DEMO-MSSQL\SQLEXPRESS02

The WideWorldImporters database documentation.

Author: dbForge Team

Created: 10-10-2023

Servers

Servers 1

DEMO-MSSQL\SQLEXPRESS02

DEMO-MSSQL\SQLEXPRESS02

Databases 1

WideWorldImporters

Properties

Name	Value
Product	Microsoft SQL Server
Version Level	RTM
Edition	Express Edition (64-bit)
Version	16.0.1050.5
Language	English (United States)
Collation	SQL_Latin1_General_CP1_CI_AS
Platform	NT x64
OS Version	6.3 (22621)
Processors	16
Physical Memory (Mb)	32694
Clustered	False
Root Directory	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS02\MSSQL

Settings

Name	Value
Default data file path	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS02\MSSQL\DATA\
Default log file path	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS02\MSSQL\DATA\
Default backup file path	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS02\MSSQL\Backup
Recovery Interval (minutes)	0
Default index fill factor (%)	0
Default backup media retention (days)	0
Nested triggers enabled	True
Allow triggers to fire others	True
Default language	us_english
Default fulltext language	us_english
Default fulltext language LCID	1033
Network packet size	4096
Two digit year cutoff	2049
Max text replication size (bytes)	65536
Locks	0

Blocked process threshold	0
Cursor threshold	-1
Cost threshold for parallelism	5
Scans and runs startup procedures	**False**
Transform noise words	**False**
Filestream access level	Disabled
Optimize for 'ad hoc' workloads	**False**
Remote login timeout (sec)	10

User databases

User databases 1

WideWorldImporters

WideWorldImporters

Description

Properties

Name	Value
SQL Server version	Microsoft SQL Server RTM 16.0.1050.5
Compatibility level	SQL Server 2016 Community Technology Preview 3.2 (CTP 3.2) through SQL Server 2016
Last backup time	01-Jun-16 10:28:12
Last backup time of transaction log	N/A
Default index fill factor (%)	0
Creation date	10-Oct-23 17:00:40
Number of Users	4
Database Encryption Enabled	False
Database Encryption Algorithm	N/A
Database status	ONLINE
Max size	0 B
Database size (Mb)	3563.73
Unallocated data space (Mb)	1654.75
Default language	N/A
Default fulltext language	us_english

Object Types 11

- Tables
- Views
- Stored Procedures
- Table-Valued Functions
- Scalar-Valued Functions
- User-Defined Table Types
- Sequences
- Partition Functions
- Partition Schemes
- Database Roles
- Schemas

Options

Name	Value
Database collation	Latin1_General_100_CI_AS
Restrict access	MULTI_USER
Cleanly shutdown	False
Supplemental logging enabled	False
Is Read-Only	False
Auto close	False

Auto shrink	False
Database is read-only for restore log	False
Is Snapshot	False
Snapshot isolation state	False
Read committed snapshot on	False
Recovery model	SIMPLE
Page verify option	Checksum
Auto create statistics	True
Auto update statistics	True
Auto update statistics asynchronously	True
ANSI NULL default	False
ANSI NULL enabled	False
ANSI padding enabled	False
ANSI warnings enabled	False
Arithmetic abort enabled	False
Concatenating NULL yields NULL	False
Numeric roundabort enabled	False
Quoted Identifier ON	False
Recursive triggers enabled	False
Close cursors on commit	False
Local cursor by default	False
Fulltext enabled	True
Trustworthy	False
Database chaining	False
Forced parameterization	False
Master key encrypted by server	False
Published	False
Merge published	False
Is distribution database	False
Sync with backup	False
Service broker GUID	5351b45e-f7a3-458b-bef8-3218751c5130
Service broker enabled	False
Log reuse wait	Nothing
Date correlation	False
Change data capture enabled	False
Honor broker priority	False
Database owner	sa
Delayed durability	Disabled
Containment type	False

Database Files 3

Name	Type	Size	Max size	Autogrowth	File Name
WWI_Data	Rows	2,097,152 KB	N/A	65,536 KB	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS02\MSSQL\DATA\WideWorldImporters.mdf
WWI_Log	Log	102,400 KB	2,147,483,648 KB	65,536 KB	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS02\MSSQL\DATA\WideWorldImporters.ldf
WWI_InMemory_Data_1	Filestream	1,449,712 KB	N/A	0 KB	C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS02\MSSQL\DATA\WideWorldImporters_InMemory_Data_1

Tables

Objects 48

Name	Description
Application.Cities	
Application.Cities_Archive	
Application.Countries	
Application.Countries_Archive	
Application.DeliveryMethods	
Application.DeliveryMethods_Archive	
Application.PaymentMethods	
Application.PaymentMethods_Archive	
Application.People	
Application.People_Archive	
Application.StateProvinces	
Application.StateProvinces_Archive	
Application.SystemParameters	
Application.TransactionTypes	
Application.TransactionTypes_Archive	
Purchasing.PurchaseOrderLines	
Purchasing.PurchaseOrders	
Purchasing.SupplierCategories	
Purchasing.SupplierCategories_Archive	
Purchasing.Suppliers	
Purchasing.Suppliers_Archive	
Purchasing.SupplierTransactions	
Sales.BuyingGroups	
Sales.BuyingGroups_Archive	
Sales.CustomerCategories	
Sales.CustomerCategories_Archive	
Sales.Customers	
Sales.Customers_Archive	
Sales.CustomerTransactions	
Sales.InvoiceLines	
Sales.Invoices	
Sales.OrderLines	

Sales.Orders	
Sales.SpecialDeals	
Warehouse.ColdRoomTemperatures	
Warehouse.ColdRoomTemperatures_Archive	
Warehouse.Colors	
Warehouse.Colors_Archive	
Warehouse.PackageTypes	
Warehouse.PackageTypes_Archive	
Warehouse.StockGroups	
Warehouse.StockGroups_Archive	
Warehouse.StockItemHoldings	
Warehouse.StockItems	
Warehouse.StockItems_Archive	
Warehouse.StockItemStockGroups	
Warehouse.StockItemTransactions	
Warehouse.VehicleTemperatures	

Application.Cities

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	37940
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	CityID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[CityID])	False	False	
	CityName	nvarchar	50	0	0	True				False	False	
	StateProvinceID	int	4	10	0	True				False	False	
	Location	geography		0	0	False				False	False	
	LatestRecordedPopulation	bigint	8	19	0	False				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	FK_Application_Cities_StateProvinceID	StateProvinceID	False		
	PK_Application_Cities	CityID	True		

Foreign Keys

Name	Columns	Description
FK_Application_Cities_Application_People	PersonID	
FK_Application_Cities_StateProvinceID_Application_StateProvinces	StateProvinceID	

Extended Properties

Name	Value
Description	Cities that are part of any address (including geographic location)

SQL Script

```
CREATE TABLE Application.Cities (
  CityID int NOT NULL CONSTRAINT DF_Application_Cities_CityID DEFAULT (NEXT VALUE FOR [Sequences].[CityID]),
  CityName nvarchar(50) NOT NULL,
  StateProvinceID int NOT NULL,
  Location geography NULL,
  LatestRecordedPopulation bigint NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Application_Cities PRIMARY KEY CLUSTERED (CityID)
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE INDEX FK_Application_Cities_StateProvinceID
  ON Application.Cities (StateProvinceID)
  ON [PRIMARY]
GO

ALTER TABLE Application.Cities
  ADD CONSTRAINT FK_Application_Cities_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.People
  (PersonID)
GO

ALTER TABLE Application.Cities
  ADD CONSTRAINT FK_Application_Cities_StateProvinceID_Application_StateProvinces FOREIGN KEY (StateProvinceID)
  REFERENCES Application.StateProvinces (StateProvinceID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Cities that are part of any address (including geographic
location)', 'SCHEMA', N'Application', 'TABLE', N'Cities'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a city within the
database', 'SCHEMA', N'Application', 'TABLE', N'Cities', 'COLUMN', N'CityID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Formal name of the
city', 'SCHEMA', N'Application', 'TABLE', N'Cities', 'COLUMN', N'CityName'
GO

EXEC sys.sp_addextendedproperty N'Description', 'State or province for this
city', 'SCHEMA', N'Application', 'TABLE', N'Cities', 'COLUMN', N'StateProvinceID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Geographic location of the
city', 'SCHEMA', N'Application', 'TABLE', N'Cities', 'COLUMN', N'Location'
GO
```

```
EXEC sys.sp_addextendedproperty N'Description', 'Latest available population for the
City', 'SCHEMA', N'Application', 'TABLE', N'Cities', 'COLUMN', N'LatestRecordedPopulation'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Application', 'TABLE', N'Cities', 'INDEX', N'FK_Application_Cities_StateProvinceID'
GO
```

Depends On 5

- Application
- Application.People
- Application.StateProvinces
- Sequences.CityID
- WideWorldImporters

Used By 11

- Application.SystemParameters
- Purchasing.Suppliers
- Sales.Customers
- Website.Customers
- Website.Suppliers
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetCityUpdates
- Website.SearchForCustomers
- Website.SearchForSuppliers
- Application.DetermineCustomerAccess

Application.Cities_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	32
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	CityID	int	4	10	0	True				False	False	
	CityName	nvarchar	50	0	0	True				False	False	
	StateProvinceID	int	4	10	0	True				False	False	
	Location	geography		0	0	False				False	False	
	LatestRecordedPopulation	bigint	8	19	0	False				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description

	ix_Cities_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Application.Cities_Archive (
  CityID int NOT NULL,
  CityName nvarchar(50) NOT NULL,
  StateProvinceID int NOT NULL,
  Location geography NULL,
  LatestRecordedPopulation bigint NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_Cities_Archive
  ON Application.Cities_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Application
- WideWorldImporters

Used By 2

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetCityUpdates

Application.Countries

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	190
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	CountryID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[CountryID])	False	False	
	CountryName	nvarchar	60	0	0	True				False	False	
	FormalName	nvarchar	60	0	0	True				False	False	
	IsoAlpha3Code	nvarchar	3	0	0	False				False	False	
	IsoNumericCode	int	4	10	0	False				False	False	
	CountryType	nvarchar	20	0	0	False				False	False	
	LatestRecordedPopulation	bigint	8	19	0	False				False	False	
	Continent	nvarchar	30	0	0	True				False	False	
	Region	nvarchar	30	0	0	True				False	False	
	Subregion	nvarchar	30	0	0	True				False	False	

	Border	geography		0	0	False				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	PK_Application_Countries	CountryID	True		
	UQ_Application_Countries_CountryName	CountryName	True		
	UQ_Application_Countries_FormalName	FormalName	True		

Foreign Keys

Name	Columns	Description
FK_Application_Countries_Application_People	PersonID	

Extended Properties

Name	Value
Description	Countries that contain the states or provinces (including geographic boundaries)

SQL Script

```
CREATE TABLE Application.Countries (
  CountryID int NOT NULL CONSTRAINT DF_Application_Countries_CountryID DEFAULT (NEXT VALUE FOR [Sequences].
  [CountryID]),
  CountryName nvarchar(60) NOT NULL,
  FormalName nvarchar(60) NOT NULL,
  IsoAlpha3Code nvarchar(3) NULL,
  IsoNumericCode int NULL,
  CountryType nvarchar(20) NULL,
  LatestRecordedPopulation bigint NULL,
  Continent nvarchar(30) NOT NULL,
  Region nvarchar(30) NOT NULL,
  Subregion nvarchar(30) NOT NULL,
  Border geography NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Application_Countries PRIMARY KEY CLUSTERED (CountryID),
  CONSTRAINT UQ_Application_Countries_CountryName UNIQUE (CountryName),
  CONSTRAINT UQ_Application_Countries_FormalName UNIQUE (FormalName)
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE Application.Countries
  ADD CONSTRAINT FK_Application_Countries_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.
  People (PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Countries that contain the states or provinces (including geographic
boundaries)
', 'SCHEMA', N'Application', 'TABLE', N'Countries'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a country within the
database', 'SCHEMA', N'Application', 'TABLE', N'Countries', 'COLUMN', N'CountryID'
GO
```

```
EXEC sys.sp_addextendedproperty N'Description', 'Name of the
country', 'SCHEMA', N'Application', 'TABLE', N'Countries', 'COLUMN', N'CountryName'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Full formal name of the country as agreed by United
Nations', 'SCHEMA', N'Application', 'TABLE', N'Countries', 'COLUMN', N'FormalName'
GO

EXEC sys.sp_addextendedproperty N'Description', '3 letter alphabetic code assigned to the country by
ISO', 'SCHEMA', N'Application', 'TABLE', N'Countries', 'COLUMN', N'IsoAlpha3Code'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric code assigned to the country by
ISO', 'SCHEMA', N'Application', 'TABLE', N'Countries', 'COLUMN', N'IsoNumericCode'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Type of country or administrative
region', 'SCHEMA', N'Application', 'TABLE', N'Countries', 'COLUMN', N'CountryType'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Latest available population for the
country', 'SCHEMA', N'Application', 'TABLE', N'Countries', 'COLUMN', N'LatestRecordedPopulation'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Name of the
continent', 'SCHEMA', N'Application', 'TABLE', N'Countries', 'COLUMN', N'Continent'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Name of the
region', 'SCHEMA', N'Application', 'TABLE', N'Countries', 'COLUMN', N'Region'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Name of the
subregion', 'SCHEMA', N'Application', 'TABLE', N'Countries', 'COLUMN', N'Subregion'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Geographic border of the country as described by the United
Nations', 'SCHEMA', N'Application', 'TABLE', N'Countries', 'COLUMN', N'Border'
GO
```

Depends On 4

- Application
- Application.People
- Sequences.CountryID
- WideWorldImporters

Used By 4

- Application.StateProvinces
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetCityUpdates

Application.Countries_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	44
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	CountryID	int	4	10	0	True				False	False	
	CountryName	nvarchar	60	0	0	True				False	False	
	FormalName	nvarchar	60	0	0	True				False	False	
	IsoAlpha3Code	nvarchar	3	0	0	False				False	False	
	IsoNumericCode	int	4	10	0	False				False	False	
	CountryType	nvarchar	20	0	0	False				False	False	
	LatestRecordedPopulation	bigint	8	19	0	False				False	False	
	Continent	nvarchar	30	0	0	True				False	False	
	Region	nvarchar	30	0	0	True				False	False	
	Subregion	nvarchar	30	0	0	True				False	False	
	Border	geography		0	0	False				False	False	

	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_Countries_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Application.Countries_Archive (
  CountryID int NOT NULL,
  CountryName nvarchar(60) NOT NULL,
  FormalName nvarchar(60) NOT NULL,
  IsoAlpha3Code nvarchar(3) NULL,
  IsoNumericCode int NULL,
  CountryType nvarchar(20) NULL,
  LatestRecordedPopulation bigint NULL,
  Continent nvarchar(30) NOT NULL,
  Region nvarchar(30) NOT NULL,
  Subregion nvarchar(30) NOT NULL,
  Border geography NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_Countries_Archive
  ON Application.Countries_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Application
- WideWorldImporters

Used By 2

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetCityUpdates

Application.DeliveryMethods

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	10
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	DeliveryMethodID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[DeliveryMethodID])	False	False	
	DeliveryMethodName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	PK_Application_DeliveryMethods	DeliveryMethodID	True		
	UQ_Application_DeliveryMethods_DeliveryMethodName	DeliveryMethodName	True		

Foreign Keys

Name	Columns	Description
FK_Application_DeliveryMethods_Application_People	PersonID	

Extended Properties

Name	Value
Description	Ways that stock items can be delivered (ie: truck/van, post, pickup, courier, etc.

SQL Script

```
CREATE TABLE Application.DeliveryMethods (
  DeliveryMethodID int NOT NULL CONSTRAINT DF_Application_DeliveryMethods_DeliveryMethodID DEFAULT (NEXT VALUE FOR
  [Sequences].[DeliveryMethodID]),
  DeliveryMethodName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Application_DeliveryMethods PRIMARY KEY CLUSTERED (DeliveryMethodID),
  CONSTRAINT UQ_Application_DeliveryMethods_DeliveryMethodName UNIQUE (DeliveryMethodName)
)
ON [PRIMARY]
GO

ALTER TABLE Application.DeliveryMethods
  ADD CONSTRAINT FK_Application_DeliveryMethods_Application_People FOREIGN KEY (LastEditedBy) REFERENCES
  Application.People (PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Ways that stock items can be delivered (ie: truck/van, post, pickup,
courier,
etc.', 'SCHEMA', N'Application', 'TABLE', N'DeliveryMethods'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a delivery method within the
database', 'SCHEMA', N'Application', 'TABLE', N'DeliveryMethods', 'COLUMN', N'DeliveryMethodID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Full name of methods that can be used for delivery of customer
orders', 'SCHEMA', N'Application', 'TABLE', N'DeliveryMethods', 'COLUMN', N'DeliveryMethodName'
GO
```

Depends On 4

- Application
- Application.People
- Sequences.DeliveryMethodID
- WideWorldImporters

Used By 8

- Purchasing.PurchaseOrders
- Purchasing.Suppliers
- Sales.Customers
- Sales.Invoices
- Website.Customers
- Website.Suppliers
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad

Application.DeliveryMethods_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	1
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	DeliveryMethodID	int	4	10	0	True				False	False	
	DeliveryMethodName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_DeliveryMethods_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Application.DeliveryMethods_Archive (
  DeliveryMethodID int NOT NULL,
```

```
  DeliveryMethodName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_DeliveryMethods_Archive
  ON Application.DeliveryMethods_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Application
- WideWorldImporters

Used By 1

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad

Application.PaymentMethods

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	4
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	PaymentMethodID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[PaymentMethodID])	False	False	
	PaymentMethodName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	PK_Application_PaymentMethods	PaymentMethodID	True		
	UQ_Application_PaymentMethods_PaymentMethodName	PaymentMethodName	True		

Foreign Keys

Name	Columns	Description
FK_Application_PaymentMethods_Application_People	PersonID	

Extended Properties

Name	Value
Description	Ways that payments can be made (ie: cash, check, EFT, etc.

SQL Script

```
CREATE TABLE Application.PaymentMethods (
  PaymentMethodID int NOT NULL CONSTRAINT DF_Application_PaymentMethods_PaymentMethodID DEFAULT (NEXT VALUE FOR
  [Sequences].[PaymentMethodID]),
  PaymentMethodName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Application_PaymentMethods PRIMARY KEY CLUSTERED (PaymentMethodID),
  CONSTRAINT UQ_Application_PaymentMethods_PaymentMethodName UNIQUE (PaymentMethodName)
)
ON [PRIMARY]
GO

ALTER TABLE Application.PaymentMethods
  ADD CONSTRAINT FK_Application_PaymentMethods_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application
  .People (PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Ways that payments can be made (ie: cash, check, EFT,
etc.', 'SCHEMA', N'Application', 'TABLE', N'PaymentMethods'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a payment type within the
database', 'SCHEMA', N'Application', 'TABLE', N'PaymentMethods', 'COLUMN', N'PaymentMethodID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Full name of ways that customers can make payments or that suppliers
can be paid', 'SCHEMA', N'Application', 'TABLE', N'PaymentMethods', 'COLUMN', N'PaymentMethodName'
GO
```

Depends On 4

- Application
- Application.People
- Sequences.PaymentMethodID
- WideWorldImporters

Used By 5

- Purchasing.SupplierTransactions
- Sales.CustomerTransactions
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetPaymentMethodUpdates

Application.PaymentMethods_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	1
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	PaymentMethodID	int	4	10	0	True				False	False	
	PaymentMethodName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_PaymentMethods_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Application.PaymentMethods_Archive (
  PaymentMethodID int NOT NULL,
```

```
  PaymentMethodName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_PaymentMethods_Archive
  ON Application.PaymentMethods_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Application
- WideWorldImporters

Used By 2

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetPaymentMethodUpdates

Application.People

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	1099
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	PersonID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[PersonID])	False	False	
	FullName	nvarchar	50	0	0	True				False	False	
	PreferredName	nvarchar	50	0	0	True				False	False	
	SearchName	nvarchar	101	0	0	True				True	True	
	IsPermittedToLogon	bit	1	1	0	True				False	False	
	LogonName	nvarchar	50	0	0	False				False	False	
	IsExternalLogonProvider	bit	1	1	0	True				False	False	
	HashedPassword	varbinary		0	0	False				False	False	
	IsSystemUser	bit	1	1	0	True				False	False	
	IsEmployee	bit	1	1	0	True				False	False	

	IsSalesperson	bit	1	1	0	True				False	False	
	UserPreferences	nvarchar		0	0	False				False	False	
	PhoneNumber	nvarchar	20	0	0	False				False	False	
	FaxNumber	nvarchar	20	0	0	False				False	False	
	EmailAddress	nvarchar	256	0	0	False				False	False	
	Photo	varbinary		0	0	False				False	False	
	CustomFields	nvarchar		0	0	False				False	False	
	OtherLanguages	nvarchar		0	0	False				True	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Computed Columns

Name	Definition
SearchName	(concat([PreferredName],N' ',[FullName]))
OtherLanguages	(json_query([CustomFields],N'$.OtherLanguages'))

Indexes

Key	Name	Columns	Unique	Type	Description
	IX_Application_People_FullName	FullName	False		
	IX_Application_People_IsEmployee	IsEmployee	False		
	IX_Application_People_IsSalesperson	IsSalesperson	False		
	IX_Application_People_Perf_20160301_05	IsPermittedToLogon, PersonID	False		
	PK_Application_People	PersonID	True		

Foreign Keys

Name	Columns	Description
FK_Application_People_Application_People	PersonID	

Extended Properties

Name	Value
Description	People known to the application (staff, customer contacts, supplier contacts)

SQL Script

```
SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE Application.People (
  PersonID int NOT NULL CONSTRAINT DF_Application_People_PersonID DEFAULT (NEXT VALUE FOR [Sequences].[PersonID]),
  FullName nvarchar(50) NOT NULL,
  PreferredName nvarchar(50) NOT NULL,
  SearchName AS (concat([PreferredName],N' ',[FullName])) PERSISTED NOT NULL,
  IsPermittedToLogon bit NOT NULL,
  LogonName nvarchar(50) NULL,
  IsExternalLogonProvider bit NOT NULL,
```

```
  HashedPassword varbinary(max) NULL,
  IsSystemUser bit NOT NULL,
  IsEmployee bit NOT NULL,
  IsSalesperson bit NOT NULL,
  UserPreferences nvarchar(max) NULL,
  PhoneNumber nvarchar(20) NULL,
  FaxNumber nvarchar(20) NULL,
  EmailAddress nvarchar(256) NULL,
  Photo varbinary(max) NULL,
  CustomFields nvarchar(max) NULL,
  OtherLanguages AS (json_query([CustomFields],N'$.OtherLanguages')),
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Application_People PRIMARY KEY CLUSTERED (PersonID)
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE INDEX IX_Application_People_FullName
  ON Application.People (FullName)
  ON [PRIMARY]
GO

CREATE INDEX IX_Application_People_IsEmployee
  ON Application.People (IsEmployee)
  ON [PRIMARY]
GO

CREATE INDEX IX_Application_People_IsSalesperson
  ON Application.People (IsSalesperson)
  ON [PRIMARY]
GO

CREATE INDEX IX_Application_People_Perf_20160301_05
  ON Application.People (IsPermittedToLogon, PersonID)
  INCLUDE (FullName, EmailAddress)
  ON [PRIMARY]
GO

ALTER TABLE Application.People
  ADD CONSTRAINT FK_Application_People_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.People
  (PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'People known to the application (staff, customer contacts, supplier
contacts)', 'SCHEMA', N'Application', 'TABLE', N'People'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a person within the
database', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'PersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Full name for this
person', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'FullName'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Name that this person prefers to be
called', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'PreferredName'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Name to build full text search on (computed
column)', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'SearchName'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Is this person permitted to log
on?', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'IsPermittedToLogon'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Person''s system logon
name', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'LogonName'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Is logon token provided by an external
system?', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'IsExternalLogonProvider'
GO
```

```
EXEC sys.sp_addextendedproperty N'Description', 'Hash of password for users without external logon
tokens', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'HashedPassword'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Is the currently permitted to make online
access?', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'IsSystemUser'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Is this person an
employee?', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'IsEmployee'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Is this person a staff
salesperson?', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'IsSalesperson'
GO

EXEC sys.sp_addextendedproperty N'Description', 'User preferences related to the website (holds JSON
data)', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'UserPreferences'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Phone
number', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'PhoneNumber'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Fax number
', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'FaxNumber'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Email address for this
person', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'EmailAddress'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Photo of this
person', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'Photo'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Custom fields for employees and
salespeople', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'CustomFields'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Other languages spoken (computed column from custom
fields)', 'SCHEMA', N'Application', 'TABLE', N'People', 'COLUMN', N'OtherLanguages'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Improves performance of name-related
queries', 'SCHEMA', N'Application', 'TABLE', N'People', 'INDEX', N'IX_Application_People_FullName'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Allows quickly locating
employees', 'SCHEMA', N'Application', 'TABLE', N'People', 'INDEX', N'IX_Application_People_IsEmployee'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Allows quickly locating
salespeople', 'SCHEMA', N'Application', 'TABLE', N'People', 'INDEX', N'IX_Application_People_IsSalesperson'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Improves performance of order picking and
invoicing', 'SCHEMA', N'Application', 'TABLE', N'People', 'INDEX', N'IX_Application_People_Perf_20160301_05'
GO
```

Depends On 4

- Application
- Application.People
- Sequences.PersonID
- WideWorldImporters

Used By 41

- Application.Cities
- Application.Countries
- Application.DeliveryMethods
- Application.PaymentMethods

- Application.People
- Application.StateProvinces
- Application.SystemParameters
- Application.TransactionTypes
- Purchasing.PurchaseOrderLines
- Purchasing.PurchaseOrders
- Purchasing.SupplierCategories
- Purchasing.Suppliers
- Purchasing.SupplierTransactions
- Sales.BuyingGroups
- Sales.CustomerCategories
- Sales.Customers
- Sales.CustomerTransactions
- Sales.InvoiceLines
- Sales.Invoices
- Sales.OrderLines
- Sales.Orders
- Sales.SpecialDeals
- Warehouse.Colors
- Warehouse.PackageTypes
- Warehouse.StockGroups
- Warehouse.StockItemHoldings
- Warehouse.StockItems
- Warehouse.StockItemStockGroups
- Warehouse.StockItemTransactions
- Website.Customers
- Website.Suppliers
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetCustomerUpdates
- Integration.GetEmployeeUpdates
- Integration.GetSupplierUpdates
- Website.ActivateWebsiteLogon
- Website.ChangePassword
- Website.SearchForCustomers
- Website.SearchForPeople
- Website.SearchForSuppliers

Application.People_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	982
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	PersonID	int	4	10	0	True				False	False	
	FullName	nvarchar	50	0	0	True				False	False	
	PreferredName	nvarchar	50	0	0	True				False	False	
	SearchName	nvarchar	101	0	0	True				False	False	
	IsPermittedToLogon	bit	1	1	0	True				False	False	
	LogonName	nvarchar	50	0	0	False				False	False	
	IsExternalLogonProvider	bit	1	1	0	True				False	False	
	HashedPassword	varbinary		0	0	False				False	False	
	IsSystemUser	bit	1	1	0	True				False	False	
	IsEmployee	bit	1	1	0	True				False	False	
	IsSalesperson	bit	1	1	0	True				False	False	

	UserPreferences	nvarchar		0	0	False				False	False	
	PhoneNumber	nvarchar	20	0	0	False				False	False	
	FaxNumber	nvarchar	20	0	0	False				False	False	
	EmailAddress	nvarchar	256	0	0	False				False	False	
	Photo	varbinary		0	0	False				False	False	
	CustomFields	nvarchar		0	0	False				False	False	
	OtherLanguages	nvarchar		0	0	False				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_People_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Application.People_Archive (
  PersonID int NOT NULL,
  FullName nvarchar(50) NOT NULL,
  PreferredName nvarchar(50) NOT NULL,
  SearchName nvarchar(101) NOT NULL,
  IsPermittedToLogon bit NOT NULL,
  LogonName nvarchar(50) NULL,
  IsExternalLogonProvider bit NOT NULL,
  HashedPassword varbinary(max) NULL,
  IsSystemUser bit NOT NULL,
  IsEmployee bit NOT NULL,
  IsSalesperson bit NOT NULL,
  UserPreferences nvarchar(max) NULL,
  PhoneNumber nvarchar(20) NULL,
  FaxNumber nvarchar(20) NULL,
  EmailAddress nvarchar(256) NULL,
  Photo varbinary(max) NULL,
  CustomFields nvarchar(max) NULL,
  OtherLanguages nvarchar(max) NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_People_Archive
  ON Application.People_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Application
- WideWorldImporters

Used By 2

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetEmployeeUpdates

Application.StateProvinces

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	53
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	StateProvinceID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[StateProvinceID])	False	False	
	StateProvinceCode	nvarchar	5	0	0	True				False	False	
	StateProvinceName	nvarchar	50	0	0	True				False	False	
	CountryID	int	4	10	0	True				False	False	
	SalesTerritory	nvarchar	50	0	0	True				False	False	
	Border	geography		0	0	False				False	False	
	LatestRecordedPopulation	bigint	8	19	0	False				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	FK_Application_StateProvinces_CountryID	CountryID	False		
	IX_Application_StateProvinces_SalesTerritory	SalesTerritory	False		
	PK_Application_StateProvinces	StateProvinceID	True		
	UQ_Application_StateProvinces_StateProvinceName	StateProvinceName	True		

Foreign Keys

Name	Columns	Description
FK_Application_StateProvinces_Application_People	PersonID	
FK_Application_StateProvinces_CountryID_Application_Countries	CountryID	

Extended Properties

Name	Value
Description	States or provinces that contain cities (including geographic location)

SQL Script

```
CREATE TABLE Application.StateProvinces (
  StateProvinceID int NOT NULL CONSTRAINT DF_Application_StateProvinces_StateProvinceID DEFAULT (NEXT VALUE FOR
  [Sequences].[StateProvinceID]),
  StateProvinceCode nvarchar(5) NOT NULL,
  StateProvinceName nvarchar(50) NOT NULL,
  CountryID int NOT NULL,
  SalesTerritory nvarchar(50) NOT NULL,
  Border geography NULL,
  LatestRecordedPopulation bigint NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Application_StateProvinces PRIMARY KEY CLUSTERED (StateProvinceID),
  CONSTRAINT UQ_Application_StateProvinces_StateProvinceName UNIQUE (StateProvinceName)
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE INDEX FK_Application_StateProvinces_CountryID
  ON Application.StateProvinces (CountryID)
  ON [PRIMARY]
GO

CREATE INDEX IX_Application_StateProvinces_SalesTerritory
  ON Application.StateProvinces (SalesTerritory)
  ON [PRIMARY]
GO

ALTER TABLE Application.StateProvinces
  ADD CONSTRAINT FK_Application_StateProvinces_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application
  .People (PersonID)
GO

ALTER TABLE Application.StateProvinces
  ADD CONSTRAINT FK_Application_StateProvinces_CountryID_Application_Countries FOREIGN KEY (CountryID) REFERENCES
  Application.Countries (CountryID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'States or provinces that contain cities (including geographic
location)', 'SCHEMA', N'Application', 'TABLE', N'StateProvinces'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a state or province within the
database', 'SCHEMA', N'Application', 'TABLE', N'StateProvinces', 'COLUMN', N'StateProvinceID'
GO
```

```
EXEC sys.sp_addextendedproperty N'Description', 'Common code for this state or province (such as WA - Washington for the
USA)', 'SCHEMA', N'Application', 'TABLE', N'StateProvinces', 'COLUMN', N'StateProvinceCode'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Formal name of the state or
province', 'SCHEMA', N'Application', 'TABLE', N'StateProvinces', 'COLUMN', N'StateProvinceName'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Country for this
StateProvince', 'SCHEMA', N'Application', 'TABLE', N'StateProvinces', 'COLUMN', N'CountryID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Sales territory for this
StateProvince', 'SCHEMA', N'Application', 'TABLE', N'StateProvinces', 'COLUMN', N'SalesTerritory'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Geographic boundary of the state or
province', 'SCHEMA', N'Application', 'TABLE', N'StateProvinces', 'COLUMN', N'Border'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Latest available population for the
StateProvince', 'SCHEMA', N'Application', 'TABLE', N'StateProvinces', 'COLUMN', N'LatestRecordedPopulation'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Application', 'TABLE', N'StateProvinces', 'INDEX', N'FK_Application_StateProvinces_CountryID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Index used to quickly locate sales
territories', 'SCHEMA', N'Application', 'TABLE', N'StateProvinces', 'INDEX', N'IX_Application_StateProvinces_SalesTerritory
'
GO
```

Depends On 5

- Application
- Application.Countries
- Application.People
- Sequences.StateProvinceID
- WideWorldImporters

Used By 5

- Application.Cities
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetCityUpdates
- Application.DetermineCustomerAccess

Application.StateProvinces_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	78
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	StateProvinceID	int	4	10	0	True				False	False	
	StateProvinceCode	nvarchar	5	0	0	True				False	False	
	StateProvinceName	nvarchar	50	0	0	True				False	False	
	CountryID	int	4	10	0	True				False	False	
	SalesTerritory	nvarchar	50	0	0	True				False	False	
	Border	geography		0	0	False				False	False	
	LatestRecordedPopulation	bigint	8	19	0	False				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_StateProvinces_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Application.StateProvinces_Archive (
  StateProvinceID int NOT NULL,
  StateProvinceCode nvarchar(5) NOT NULL,
  StateProvinceName nvarchar(50) NOT NULL,
  CountryID int NOT NULL,
  SalesTerritory nvarchar(50) NOT NULL,
  Border geography NULL,
  LatestRecordedPopulation bigint NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_StateProvinces_Archive
  ON Application.StateProvinces_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Application
- WideWorldImporters

Used By 2

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetCityUpdates

Application.SystemParameters

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	1
Created	25-May-16 15:36:53
Last Modified	25-May-16 15:36:53

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	SystemParameterID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[SystemParameterID])	False	False	
	DeliveryAddressLine1	nvarchar	60	0	0	True				False	False	
	DeliveryAddressLine2	nvarchar	60	0	0	False				False	False	
	DeliveryCityID	int	4	10	0	True				False	False	
	DeliveryPostalCode	nvarchar	10	0	0	True				False	False	
	DeliveryLocation	geography		0	0	True				False	False	
	PostalAddressLine1	nvarchar	60	0	0	True				False	False	
	PostalAddressLine2	nvarchar	60	0	0	False				False	False	
	PostalCityID	int	4	10	0	True				False	False	
	PostalPostalCode	nvarchar	10	0	0	True				False	False	

	ApplicationSettings	nvarchar		0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	LastEditedWhen	datetime2	8	27	7	True			(sysdatetime())	False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	FK_Application_SystemParameters_DeliveryCityID	DeliveryCityID	False		
	FK_Application_SystemParameters_PostalCityID	PostalCityID	False		
	PK_Application_SystemParameters	SystemParameterID	True		

Foreign Keys

Name	Columns	Description
FK_Application_SystemParameters_Application_People	PersonID	
FK_Application_SystemParameters_DeliveryCityID_Application_Cities	CityID	
FK_Application_SystemParameters_PostalCityID_Application_Cities	CityID	

Extended Properties

Name	Value
Description	Any configurable parameters for the whole system

SQL Script

```
CREATE TABLE Application.SystemParameters (
  SystemParameterID int NOT NULL CONSTRAINT DF_Application_SystemParameters_SystemParameterID DEFAULT (NEXT VALUE
  FOR [Sequences].[SystemParameterID]),
  DeliveryAddressLine1 nvarchar(60) NOT NULL,
  DeliveryAddressLine2 nvarchar(60) NULL,
  DeliveryCityID int NOT NULL,
  DeliveryPostalCode nvarchar(10) NOT NULL,
  DeliveryLocation geography NOT NULL,
  PostalAddressLine1 nvarchar(60) NOT NULL,
  PostalAddressLine2 nvarchar(60) NULL,
  PostalCityID int NOT NULL,
  PostalPostalCode nvarchar(10) NOT NULL,
  ApplicationSettings nvarchar(max) NOT NULL,
  LastEditedBy int NOT NULL,
  LastEditedWhen datetime2 NOT NULL CONSTRAINT DF_Application_SystemParameters_LastEditedWhen DEFAULT (sysdatetime(
  )),
  CONSTRAINT PK_Application_SystemParameters PRIMARY KEY CLUSTERED (SystemParameterID)
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE INDEX FK_Application_SystemParameters_DeliveryCityID
  ON Application.SystemParameters (DeliveryCityID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Application_SystemParameters_PostalCityID
  ON Application.SystemParameters (PostalCityID)
  ON [PRIMARY]
GO

ALTER TABLE Application.SystemParameters
  ADD CONSTRAINT FK_Application_SystemParameters_Application_People FOREIGN KEY (LastEditedBy) REFERENCES
  Application.People (PersonID)
GO

ALTER TABLE Application.SystemParameters
```

```
  ADD CONSTRAINT FK_Application_SystemParameters_DeliveryCityID_Application_Cities FOREIGN KEY (DeliveryCityID)
  REFERENCES Application.Cities (CityID)
GO

ALTER TABLE Application.SystemParameters
  ADD CONSTRAINT FK_Application_SystemParameters_PostalCityID_Application_Cities FOREIGN KEY (PostalCityID)
  REFERENCES Application.Cities (CityID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Any configurable parameters for the whole
system', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for row holding system
parameters', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters', 'COLUMN', N'SystemParameterID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'First address line for the
company', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters', 'COLUMN', N'DeliveryAddressLine1'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Second address line for the
company', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters', 'COLUMN', N'DeliveryAddressLine2'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of the city for this
address', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters', 'COLUMN', N'DeliveryCityID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Postal code for the
company', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters', 'COLUMN', N'DeliveryPostalCode'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Geographic location for the company
office', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters', 'COLUMN', N'DeliveryLocation'
GO

EXEC sys.sp_addextendedproperty N'Description', 'First postal address line for the
company', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters', 'COLUMN', N'PostalAddressLine1'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Second postaladdress line for the
company', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters', 'COLUMN', N'PostalAddressLine2'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of the city for this
postaladdress', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters', 'COLUMN', N'PostalCityID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Postal code for the company when sending via
mail', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters', 'COLUMN', N'PostalPostalCode'
GO

EXEC sys.sp_addextendedproperty N'Description', 'JSON-structured application
settings', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters', 'COLUMN', N'ApplicationSettings'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters', 'INDEX', N'FK_Application_SystemParameters_DeliveryCityID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Application', 'TABLE', N'SystemParameters', 'INDEX', N'FK_Application_SystemParameters_PostalCityID'
GO
```

Depends On 5

- Application
- Application.Cities
- Application.People
- Sequences.SystemParameterID
- WideWorldImporters

Used By

No items found

Application.TransactionTypes

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	13
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	TransactionTypeID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[TransactionTypeID])	False	False	
	TransactionTypeName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	PK_Application_TransactionTypes	TransactionTypeID	True		
	UQ_Application_TransactionTypes_TransactionTypeName	TransactionTypeName	True		

Foreign Keys

Name	Columns	Description
FK_Application_TransactionTypes_Application_People	PersonID	

Extended Properties

Name	Value
Description	Types of customer, supplier, or stock transactions (ie: invoice, credit note, etc.)

SQL Script

```
CREATE TABLE Application.TransactionTypes (
  TransactionTypeID int NOT NULL CONSTRAINT DF_Application_TransactionTypes_TransactionTypeID DEFAULT (NEXT VALUE
  FOR [Sequences].[TransactionTypeID]),
  TransactionTypeName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Application_TransactionTypes PRIMARY KEY CLUSTERED (TransactionTypeID),
  CONSTRAINT UQ_Application_TransactionTypes_TransactionTypeName UNIQUE (TransactionTypeName)
)
ON [PRIMARY]
GO

ALTER TABLE Application.TransactionTypes
  ADD CONSTRAINT FK_Application_TransactionTypes_Application_People FOREIGN KEY (LastEditedBy) REFERENCES
  Application.People (PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Types of customer, supplier, or stock transactions (ie: invoice,
credit note,
etc.)', 'SCHEMA', N'Application', 'TABLE', N'TransactionTypes'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a transaction type within the
database', 'SCHEMA', N'Application', 'TABLE', N'TransactionTypes', 'COLUMN', N'TransactionTypeID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Full name of the transaction
type', 'SCHEMA', N'Application', 'TABLE', N'TransactionTypes', 'COLUMN', N'TransactionTypeName'
GO
```

Depends On 4

- Application
- Application.People
- Sequences.TransactionTypeID
- WideWorldImporters

Used By 7

- Purchasing.SupplierTransactions
- Sales.CustomerTransactions
- Warehouse.StockItemTransactions
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetTransactionTypeUpdates
- Website.InvoiceCustomerOrders

Application.TransactionTypes_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	1
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	TransactionTypeID	int	4	10	0	True				False	False	
	TransactionTypeName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_TransactionTypes_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Application.TransactionTypes_Archive (
  TransactionTypeID int NOT NULL,
```

```
  TransactionTypeName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_TransactionTypes_Archive
  ON Application.TransactionTypes_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Application
- WideWorldImporters

Used By 2

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetTransactionTypeUpdates

Purchasing.PurchaseOrderLines

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	5734
Created	25-May-16 15:36:54
Last Modified	25-May-16 15:36:54

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	PurchaseOrderLineID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[PurchaseOrderLineID])	False	False	
	PurchaseOrderID	int	4	10	0	True				False	False	
	StockItemID	int	4	10	0	True				False	False	
	OrderedOuters	int	4	10	0	True				False	False	
	Description	nvarchar	100	0	0	True				False	False	
	ReceivedOuters	int	4	10	0	True				False	False	
	PackageTypeID	int	4	10	0	True				False	False	
	ExpectedUnitPricePerOuter	decimal	9	18	2	False				False	False	
	LastReceiptDate	date	3	10	0	False				False	False	
	IsOrderLineFinalized	bit	1	1	0	True				False	False	

	LastEditedBy	int	4	10	0	True				False	False	
	LastEditedWhen	datetime2	8	27	7	True			(sysdatetime())	False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	FK_Purchasing_PurchaseOrderLines_PackageTypeID	PackageTypeID	False		
	FK_Purchasing_PurchaseOrderLines_PurchaseOrderID	PurchaseOrderID	False		
	FK_Purchasing_PurchaseOrderLines_StockItemID	StockItemID	False		
	IX_Purchasing_PurchaseOrderLines_Perf_20160301_4	IsOrderLineFinalized, StockItemID	False		
	PK_Purchasing_PurchaseOrderLines	PurchaseOrderLineID	True		

Foreign Keys

Name	Columns	Description
FK_Purchasing_PurchaseOrderLines_Application_People	PersonID	
FK_Purchasing_PurchaseOrderLines_PackageTypeID_Warehouse_PackageTypes	PackageTypeID	
FK_Purchasing_PurchaseOrderLines_PurchaseOrderID_Purchasing_PurchaseOrders	PurchaseOrderID	
FK_Purchasing_PurchaseOrderLines_StockItemID_Warehouse_StockItems	StockItemID	

Extended Properties

Name	Value
Description	Detail lines from supplier purchase orders

SQL Script

```
CREATE TABLE Purchasing.PurchaseOrderLines (
  PurchaseOrderLineID int NOT NULL CONSTRAINT DF_Purchasing_PurchaseOrderLines_PurchaseOrderLineID DEFAULT (NEXT
  VALUE FOR [Sequences].[PurchaseOrderLineID]),
  PurchaseOrderID int NOT NULL,
  StockItemID int NOT NULL,
  OrderedOuters int NOT NULL,
  Description nvarchar(100) NOT NULL,
  ReceivedOuters int NOT NULL,
  PackageTypeID int NOT NULL,
  ExpectedUnitPricePerOuter decimal(18, 2) NULL,
  LastReceiptDate date NULL,
  IsOrderLineFinalized bit NOT NULL,
  LastEditedBy int NOT NULL,
  LastEditedWhen datetime2 NOT NULL CONSTRAINT DF_Purchasing_PurchaseOrderLines_LastEditedWhen DEFAULT (sysdatetime
  ()),
  CONSTRAINT PK_Purchasing_PurchaseOrderLines PRIMARY KEY CLUSTERED (PurchaseOrderLineID)
)
ON [PRIMARY]
GO

CREATE INDEX FK_Purchasing_PurchaseOrderLines_PackageTypeID
  ON Purchasing.PurchaseOrderLines (PackageTypeID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Purchasing_PurchaseOrderLines_PurchaseOrderID
  ON Purchasing.PurchaseOrderLines (PurchaseOrderID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Purchasing_PurchaseOrderLines_StockItemID
  ON Purchasing.PurchaseOrderLines (StockItemID)
  ON [PRIMARY]
GO

CREATE INDEX IX_Purchasing_PurchaseOrderLines_Perf_20160301_4
```

```
  ON Purchasing.PurchaseOrderLines (IsOrderLineFinalized, StockItemID)
  INCLUDE (OrderedOuters, ReceivedOuters)
  ON [PRIMARY]
GO

ALTER TABLE Purchasing.PurchaseOrderLines
  ADD CONSTRAINT FK_Purchasing_PurchaseOrderLines_Application_People FOREIGN KEY (LastEditedBy) REFERENCES
  Application.People (PersonID)
GO

ALTER TABLE Purchasing.PurchaseOrderLines
  ADD CONSTRAINT FK_Purchasing_PurchaseOrderLines_PackageTypeID_Warehouse_PackageTypes FOREIGN KEY (PackageTypeID)
  REFERENCES Warehouse.PackageTypes (PackageTypeID)
GO

ALTER TABLE Purchasing.PurchaseOrderLines
  ADD CONSTRAINT FK_Purchasing_PurchaseOrderLines_PurchaseOrderID_Purchasing_PurchaseOrders FOREIGN KEY (
  PurchaseOrderID) REFERENCES Purchasing.PurchaseOrders (PurchaseOrderID)
GO

ALTER TABLE Purchasing.PurchaseOrderLines
  ADD CONSTRAINT FK_Purchasing_PurchaseOrderLines_StockItemID_Warehouse_StockItems FOREIGN KEY (StockItemID)
  REFERENCES Warehouse.StockItems (StockItemID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Detail lines from supplier purchase
orders', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a line on a purchase order within the
database', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'COLUMN', N'PurchaseOrderLineID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Purchase order that this line is associated
with', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'COLUMN', N'PurchaseOrderID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Stock item for this purchase order
line', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'COLUMN', N'StockItemID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Quantity of the stock item that is
ordered', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'COLUMN', N'OrderedOuters'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Description of the item to be supplied (Often the stock item name but could be
supplier description)', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'COLUMN', N'Description'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Total quantity of the stock item that has been received so
far', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'COLUMN', N'ReceivedOuters'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Type of package
received', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'COLUMN', N'PackageTypeID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'The unit price that we expect to be
charged', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'COLUMN', N'ExpectedUnitPricePerOuter'
GO

EXEC sys.sp_addextendedproperty N'Description', 'The last date on which this stock item was received for this purchase
order', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'COLUMN', N'LastReceiptDate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Is this purchase order line now considered finalized? (Receipted quantities
and
weights are often not precise)', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'COLUMN', N'IsOrderLineFinalized'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'INDEX', N'FK_Purchasing_PurchaseOrderLines_PackageTypeID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'INDEX', N'FK_Purchasing_PurchaseOrderLines_PurchaseOrderID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'INDEX', N'FK_Purchasing_PurchaseOrderLines_StockItemID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Improves performance of order picking and
```

```
invoicing', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrderLines', 'INDEX', N'IX_Purchasing_PurchaseOrderLines_Perf_20160301_
4'
GO
```

Depends On 7

- Purchasing
- Application.People
- Purchasing.PurchaseOrders
- Warehouse.PackageTypes
- Warehouse.StockItems
- Sequences.PurchaseOrderLineID
- WideWorldImporters

Used By 1

- Integration.GetPurchaseUpdates

Purchasing.PurchaseOrders

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	1996
Created	25-May-16 15:36:54
Last Modified	25-May-16 15:36:54

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	PurchaseOrderID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[PurchaseOrderID])	False	False	
	SupplierID	int	4	10	0	True				False	False	
	OrderDate	date	3	10	0	True				False	False	
	DeliveryMethodID	int	4	10	0	True				False	False	
	ContactPersonID	int	4	10	0	True				False	False	
	ExpectedDeliveryDate	date	3	10	0	False				False	False	
	SupplierReference	nvarchar	20	0	0	False				False	False	
	IsOrderFinalized	bit	1	1	0	True				False	False	

	Comments	nvarchar		0	0	False				False	False	
	InternalComments	nvarchar		0	0	False				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	LastEditedWhen	datetime2	8	27	7	True			(sysdatetime())	False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	FK_Purchasing_PurchaseOrders_ContactPersonID	ContactPersonID	False		
	FK_Purchasing_PurchaseOrders_DeliveryMethodID	DeliveryMethodID	False		
	FK_Purchasing_PurchaseOrders_SupplierID	SupplierID	False		
	PK_Purchasing_PurchaseOrders	PurchaseOrderID	True		

Foreign Keys

Name	Columns	Description
FK_Purchasing_PurchaseOrders_Application_People	PersonID	
FK_Purchasing_PurchaseOrders_ContactPersonID_Application_People	PersonID	
FK_Purchasing_PurchaseOrders_DeliveryMethodID_Application_DeliveryMethods	DeliveryMethodID	
FK_Purchasing_PurchaseOrders_SupplierID_Purchasing_Suppliers	SupplierID	

Extended Properties

Name	Value
Description	Details of supplier purchase orders

SQL Script

```
CREATE TABLE Purchasing.PurchaseOrders (
  PurchaseOrderID int NOT NULL CONSTRAINT DF_Purchasing_PurchaseOrders_PurchaseOrderID DEFAULT (NEXT VALUE FOR
  [Sequences].[PurchaseOrderID]),
  SupplierID int NOT NULL,
  OrderDate date NOT NULL,
  DeliveryMethodID int NOT NULL,
  ContactPersonID int NOT NULL,
  ExpectedDeliveryDate date NULL,
  SupplierReference nvarchar(20) NULL,
  IsOrderFinalized bit NOT NULL,
  Comments nvarchar(max) NULL,
  InternalComments nvarchar(max) NULL,
  LastEditedBy int NOT NULL,
  LastEditedWhen datetime2 NOT NULL CONSTRAINT DF_Purchasing_PurchaseOrders_LastEditedWhen DEFAULT (sysdatetime()),
  CONSTRAINT PK_Purchasing_PurchaseOrders PRIMARY KEY CLUSTERED (PurchaseOrderID)
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE INDEX FK_Purchasing_PurchaseOrders_ContactPersonID
  ON Purchasing.PurchaseOrders (ContactPersonID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Purchasing_PurchaseOrders_DeliveryMethodID
  ON Purchasing.PurchaseOrders (DeliveryMethodID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Purchasing_PurchaseOrders_SupplierID
```

```
  ON Purchasing.PurchaseOrders (SupplierID)
  ON [PRIMARY]
GO

ALTER TABLE Purchasing.PurchaseOrders
  ADD CONSTRAINT FK_Purchasing_PurchaseOrders_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.
  People (PersonID)
GO

ALTER TABLE Purchasing.PurchaseOrders
  ADD CONSTRAINT FK_Purchasing_PurchaseOrders_ContactPersonID_Application_People FOREIGN KEY (ContactPersonID)
  REFERENCES Application.People (PersonID)
GO

ALTER TABLE Purchasing.PurchaseOrders
  ADD CONSTRAINT FK_Purchasing_PurchaseOrders_DeliveryMethodID_Application_DeliveryMethods FOREIGN KEY (
  DeliveryMethodID) REFERENCES Application.DeliveryMethods (DeliveryMethodID)
GO

ALTER TABLE Purchasing.PurchaseOrders
  ADD CONSTRAINT FK_Purchasing_PurchaseOrders_SupplierID_Purchasing_Suppliers FOREIGN KEY (SupplierID) REFERENCES
  Purchasing.Suppliers (SupplierID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Details of supplier purchase
orders', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a purchase order within the
database', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders', 'COLUMN', N'PurchaseOrderID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Supplier for this purchase
order', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders', 'COLUMN', N'SupplierID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Date that this purchase order was
raised', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders', 'COLUMN', N'OrderDate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'How this purchase order should be
delivered', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders', 'COLUMN', N'DeliveryMethodID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'The person who is the primary contact for this purchase
order', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders', 'COLUMN', N'ContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Expected delivery date for this purchase
order', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders', 'COLUMN', N'ExpectedDeliveryDate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Supplier reference for our organization (might be our account number at
the
supplier)', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders', 'COLUMN', N'SupplierReference'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Is this purchase order now considered
finalized?', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders', 'COLUMN', N'IsOrderFinalized'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Any comments related this purchase order (comments sent to the
supplier)', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders', 'COLUMN', N'Comments'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Any internal comments related this purchase order (comments for
internal
reference only and not sent to the
supplier)', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders', 'COLUMN', N'InternalComments'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders', 'INDEX', N'FK_Purchasing_PurchaseOrders_ContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders', 'INDEX', N'FK_Purchasing_PurchaseOrders_DeliveryMethodID'
GO
```

```
EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'PurchaseOrders', 'INDEX', N'FK_Purchasing_PurchaseOrders_SupplierID'
GO
```

Depends On 6

- Purchasing
- Application.DeliveryMethods
- Application.People
- Purchasing.Suppliers
- Sequences.PurchaseOrderID
- WideWorldImporters

Used By 4

- Purchasing.PurchaseOrderLines
- Purchasing.SupplierTransactions
- Warehouse.StockItemTransactions
- Integration.GetPurchaseUpdates

Purchasing.SupplierCategories

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	9
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	SupplierCategoryID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[SupplierCategoryID])	False	False	
	SupplierCategoryName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	PK_Purchasing_SupplierCategories	SupplierCategoryID	True		
	UQ_Purchasing_SupplierCategories_SupplierCategoryName	SupplierCategoryName	True		

Foreign Keys

Name	Columns	Description
FK_Purchasing_SupplierCategories_Application_People	PersonID	

Extended Properties

Name	Value
Description	Categories for suppliers (ie novelties, toys, clothing, packaging, etc.)

SQL Script

```
CREATE TABLE Purchasing.SupplierCategories (
  SupplierCategoryID int NOT NULL CONSTRAINT DF_Purchasing_SupplierCategories_SupplierCategoryID DEFAULT (NEXT
  VALUE FOR [Sequences].[SupplierCategoryID]),
  SupplierCategoryName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Purchasing_SupplierCategories PRIMARY KEY CLUSTERED (SupplierCategoryID),
  CONSTRAINT UQ_Purchasing_SupplierCategories_SupplierCategoryName UNIQUE (SupplierCategoryName)
)
ON [PRIMARY]
GO

ALTER TABLE Purchasing.SupplierCategories
  ADD CONSTRAINT FK_Purchasing_SupplierCategories_Application_People FOREIGN KEY (LastEditedBy) REFERENCES
  Application.People (PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Categories for suppliers (ie novelties, toys, clothing, packaging,
etc.)', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierCategories'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a supplier category within the
database', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierCategories', 'COLUMN', N'SupplierCategoryID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Full name of the category that suppliers can be assigned
to', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierCategories', 'COLUMN', N'SupplierCategoryName'
GO
```

Depends On 4

- Purchasing
- Application.People
- Sequences.SupplierCategoryID
- WideWorldImporters

Used By 5

- Purchasing.Suppliers
- Website.Suppliers
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetSupplierUpdates

Purchasing.SupplierCategories_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	1
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	SupplierCategoryID	int	4	10	0	True				False	False	
	SupplierCategoryName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_SupplierCategories_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Purchasing.SupplierCategories_Archive (
  SupplierCategoryID int NOT NULL,
```

```
  SupplierCategoryName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_SupplierCategories_Archive
  ON Purchasing.SupplierCategories_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Purchasing
- WideWorldImporters

Used By 2

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetSupplierUpdates

Purchasing.Suppliers

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	13
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	SupplierID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[SupplierID])	False	False	
	SupplierName	nvarchar	100	0	0	True				False	False	
	SupplierCategoryID	int	4	10	0	True				False	False	
	PrimaryContactPersonID	int	4	10	0	True				False	False	
	AlternateContactPersonID	int	4	10	0	True				False	False	
	DeliveryMethodID	int	4	10	0	False				False	False	
	DeliveryCityID	int	4	10	0	True				False	False	

	PostalCityID	int	4	10	0	True				False	False	
	SupplierReference	nvarchar	20	0	0	False				False	False	
	BankAccountName	nvarchar	50	0	0	False				False	False	
	BankAccountBranch	nvarchar	50	0	0	False				False	False	
	BankAccountCode	nvarchar	20	0	0	False				False	False	
	BankAccountNumber	nvarchar	20	0	0	False				False	False	
	BankInternationalCode	nvarchar	20	0	0	False				False	False	
	PaymentDays	int	4	10	0	True				False	False	
	InternalComments	nvarchar		0	0	False				False	False	
	PhoneNumber	nvarchar	20	0	0	True				False	False	
	FaxNumber	nvarchar	20	0	0	True				False	False	
	WebsiteURL	nvarchar	256	0	0	True				False	False	
	DeliveryAddressLine1	nvarchar	60	0	0	True				False	False	
	DeliveryAddressLine2	nvarchar	60	0	0	False				False	False	
	DeliveryPostalCode	nvarchar	10	0	0	True				False	False	
	DeliveryLocation	geography		0	0	False				False	False	
	PostalAddressLine1	nvarchar	60	0	0	True				False	False	
	PostalAddressLine2	nvarchar	60	0	0	False				False	False	
	PostalPostalCode	nvarchar	10	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	FK_Purchasing_Suppliers_AlternateContactPersonID	AlternateContactPersonID	False		
	FK_Purchasing_Suppliers_DeliveryCityID	DeliveryCityID	False		
	FK_Purchasing_Suppliers_DeliveryMethodID	DeliveryMethodID	False		
	FK_Purchasing_Suppliers_PostalCityID	PostalCityID	False		
	FK_Purchasing_Suppliers_PrimaryContactPersonID	PrimaryContactPersonID	False		
	FK_Purchasing_Suppliers_SupplierCategoryID	SupplierCategoryID	False		
	PK_Purchasing_Suppliers	SupplierID	True		
	UQ_Purchasing_Suppliers_SupplierName	SupplierName	True		

Foreign Keys

Name	Columns	Description
FK_Purchasing_Suppliers_AlternateContactPersonID_Application_People	PersonID	

FK_Purchasing_Suppliers_Application_People	PersonID	
FK_Purchasing_Suppliers_DeliveryCityID_Application_Cities	CityID	
FK_Purchasing_Suppliers_DeliveryMethodID_Application_DeliveryMethods	DeliveryMethodID	
FK_Purchasing_Suppliers_PostalCityID_Application_Cities	CityID	
FK_Purchasing_Suppliers_PrimaryContactPersonID_Application_People	PersonID	
FK_Purchasing_Suppliers_SupplierCategoryID_Purchasing_SupplierCategories	SupplierCategoryID	

Extended Properties

Name	Value
Description	Main entity table for suppliers (organizations)

SQL Script

```
CREATE TABLE Purchasing.Suppliers (
  SupplierID int NOT NULL CONSTRAINT DF_Purchasing_Suppliers_SupplierID DEFAULT (NEXT VALUE FOR [Sequences].
  [SupplierID]),
  SupplierName nvarchar(100) NOT NULL,
  SupplierCategoryID int NOT NULL,
  PrimaryContactPersonID int NOT NULL,
  AlternateContactPersonID int NOT NULL,
  DeliveryMethodID int NULL,
  DeliveryCityID int NOT NULL,
  PostalCityID int NOT NULL,
  SupplierReference nvarchar(20) NULL,
  BankAccountName nvarchar(50) NULL,
  BankAccountBranch nvarchar(50) NULL,
  BankAccountCode nvarchar(20) NULL,
  BankAccountNumber nvarchar(20) NULL,
  BankInternationalCode nvarchar(20) NULL,
  PaymentDays int NOT NULL,
  InternalComments nvarchar(max) NULL,
  PhoneNumber nvarchar(20) NOT NULL,
  FaxNumber nvarchar(20) NOT NULL,
  WebsiteURL nvarchar(256) NOT NULL,
  DeliveryAddressLine1 nvarchar(60) NOT NULL,
  DeliveryAddressLine2 nvarchar(60) NULL,
  DeliveryPostalCode nvarchar(10) NOT NULL,
  DeliveryLocation geography NULL,
  PostalAddressLine1 nvarchar(60) NOT NULL,
  PostalAddressLine2 nvarchar(60) NULL,
  PostalPostalCode nvarchar(10) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Purchasing_Suppliers PRIMARY KEY CLUSTERED (SupplierID),
  CONSTRAINT UQ_Purchasing_Suppliers_SupplierName UNIQUE (SupplierName)
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE INDEX FK_Purchasing_Suppliers_AlternateContactPersonID
  ON Purchasing.Suppliers (AlternateContactPersonID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Purchasing_Suppliers_DeliveryCityID
  ON Purchasing.Suppliers (DeliveryCityID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Purchasing_Suppliers_DeliveryMethodID
  ON Purchasing.Suppliers (DeliveryMethodID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Purchasing_Suppliers_PostalCityID
  ON Purchasing.Suppliers (PostalCityID)
```

```
  ON [PRIMARY]
GO

CREATE INDEX FK_Purchasing_Suppliers_PrimaryContactPersonID
  ON Purchasing.Suppliers (PrimaryContactPersonID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Purchasing_Suppliers_SupplierCategoryID
  ON Purchasing.Suppliers (SupplierCategoryID)
  ON [PRIMARY]
GO

ALTER TABLE Purchasing.Suppliers
  ADD CONSTRAINT FK_Purchasing_Suppliers_AlternateContactPersonID_Application_People FOREIGN KEY (
  AlternateContactPersonID) REFERENCES Application.People (PersonID)
GO

ALTER TABLE Purchasing.Suppliers
  ADD CONSTRAINT FK_Purchasing_Suppliers_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.
  People (PersonID)
GO

ALTER TABLE Purchasing.Suppliers
  ADD CONSTRAINT FK_Purchasing_Suppliers_DeliveryCityID_Application_Cities FOREIGN KEY (DeliveryCityID) REFERENCES
  Application.Cities (CityID)
GO

ALTER TABLE Purchasing.Suppliers
  ADD CONSTRAINT FK_Purchasing_Suppliers_DeliveryMethodID_Application_DeliveryMethods FOREIGN KEY (DeliveryMethodID
  ) REFERENCES Application.DeliveryMethods (DeliveryMethodID)
GO

ALTER TABLE Purchasing.Suppliers
  ADD CONSTRAINT FK_Purchasing_Suppliers_PostalCityID_Application_Cities FOREIGN KEY (PostalCityID) REFERENCES
  Application.Cities (CityID)
GO

ALTER TABLE Purchasing.Suppliers
  ADD CONSTRAINT FK_Purchasing_Suppliers_PrimaryContactPersonID_Application_People FOREIGN KEY (
  PrimaryContactPersonID) REFERENCES Application.People (PersonID)
GO

ALTER TABLE Purchasing.Suppliers
  ADD CONSTRAINT FK_Purchasing_Suppliers_SupplierCategoryID_Purchasing_SupplierCategories FOREIGN KEY (
  SupplierCategoryID) REFERENCES Purchasing.SupplierCategories (SupplierCategoryID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Main entity table for suppliers
(organizations)', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a supplier within the
database', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'SupplierID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Supplier''s full name (usually a trading
name)', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'SupplierName'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Supplier''s
category', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'SupplierCategoryID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Primary
contact', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'PrimaryContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Alternate
contact', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'AlternateContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Standard delivery method for stock items received from this
supplier', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'DeliveryMethodID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of the delivery city for this
address', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'DeliveryCityID'
GO
```

```
EXEC sys.sp_addextendedproperty N'Description', 'ID of the mailing city for this
address', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'PostalCityID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Supplier reference for our organization (might be our account number at
the
supplier)', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'SupplierReference'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Supplier''s bank account name (ie name on the
account)', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'BankAccountName'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Supplier''s bank
branch', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'BankAccountBranch'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Supplier''s bank account code (usually a numeric reference for the bank
branch)', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'BankAccountCode'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Supplier''s bank account
number', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'BankAccountNumber'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Supplier''s bank''s international code (such as a SWIFT
code)', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'BankInternationalCode'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Number of days for payment of an invoice (ie payment
terms)', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'PaymentDays'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Internal comments (not exposed outside
organization)', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'InternalComments'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Phone
number', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'PhoneNumber'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Fax number
', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'FaxNumber'
GO

EXEC sys.sp_addextendedproperty N'Description', 'URL for the website for this
supplier', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'WebsiteURL'
GO

EXEC sys.sp_addextendedproperty N'Description', 'First delivery address line for the
supplier', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'DeliveryAddressLine1'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Second delivery address line for the
supplier', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'DeliveryAddressLine2'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Delivery postal code for the
supplier', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'DeliveryPostalCode'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Geographic location for the supplier''s
office/warehouse', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'DeliveryLocation'
GO

EXEC sys.sp_addextendedproperty N'Description', 'First postal address line for the
supplier', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'PostalAddressLine1'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Second postal address line for the
supplier', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'PostalAddressLine2'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Postal code for the supplier when sending by
mail', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'COLUMN', N'PostalPostalCode'
GO
```

```
EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'INDEX', N'FK_Purchasing_Suppliers_AlternateContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'INDEX', N'FK_Purchasing_Suppliers_DeliveryCityID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'INDEX', N'FK_Purchasing_Suppliers_DeliveryMethodID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'INDEX', N'FK_Purchasing_Suppliers_PostalCityID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'INDEX', N'FK_Purchasing_Suppliers_PrimaryContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'Suppliers', 'INDEX', N'FK_Purchasing_Suppliers_SupplierCategoryID'
GO
```

Depends On 7

- Purchasing
- Application.Cities
- Application.DeliveryMethods
- Application.People
- Purchasing.SupplierCategories
- Sequences.SupplierID
- WideWorldImporters

Used By 10

- Purchasing.PurchaseOrders
- Purchasing.SupplierTransactions
- Warehouse.StockItems
- Warehouse.StockItemTransactions
- Website.Suppliers
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetSupplierUpdates
- Website.SearchForPeople
- Website.SearchForSuppliers

Purchasing.Suppliers_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	13
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	SupplierID	int	4	10	0	True				False	False	
	SupplierName	nvarchar	100	0	0	True				False	False	
	SupplierCategoryID	int	4	10	0	True				False	False	
	PrimaryContactPersonID	int	4	10	0	True				False	False	
	AlternateContactPersonID	int	4	10	0	True				False	False	
	DeliveryMethodID	int	4	10	0	False				False	False	
	DeliveryCityID	int	4	10	0	True				False	False	
	PostalCityID	int	4	10	0	True				False	False	
	SupplierReference	nvarchar	20	0	0	False				False	False	
	BankAccountName	nvarchar	50	0	0	False				False	False	
	BankAccountBranch	nvarchar	50	0	0	False				False	False	

	BankAccountCode	nvarchar	20	0	0	False				False	False	
	BankAccountNumber	nvarchar	20	0	0	False				False	False	
	BankInternationalCode	nvarchar	20	0	0	False				False	False	
	PaymentDays	int	4	10	0	True				False	False	
	InternalComments	nvarchar		0	0	False				False	False	
	PhoneNumber	nvarchar	20	0	0	True				False	False	
	FaxNumber	nvarchar	20	0	0	True				False	False	
	WebsiteURL	nvarchar	256	0	0	True				False	False	
	DeliveryAddressLine1	nvarchar	60	0	0	True				False	False	
	DeliveryAddressLine2	nvarchar	60	0	0	False				False	False	
	DeliveryPostalCode	nvarchar	10	0	0	True				False	False	
	DeliveryLocation	geography		0	0	False				False	False	
	PostalAddressLine1	nvarchar	60	0	0	True				False	False	
	PostalAddressLine2	nvarchar	60	0	0	False				False	False	
	PostalPostalCode	nvarchar	10	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_Suppliers_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Purchasing.Suppliers_Archive (
  SupplierID int NOT NULL,
  SupplierName nvarchar(100) NOT NULL,
  SupplierCategoryID int NOT NULL,
  PrimaryContactPersonID int NOT NULL,
  AlternateContactPersonID int NOT NULL,
  DeliveryMethodID int NULL,
  DeliveryCityID int NOT NULL,
  PostalCityID int NOT NULL,
  SupplierReference nvarchar(20) NULL,
  BankAccountName nvarchar(50) NULL,
  BankAccountBranch nvarchar(50) NULL,
  BankAccountCode nvarchar(20) NULL,
  BankAccountNumber nvarchar(20) NULL,
  BankInternationalCode nvarchar(20) NULL,
  PaymentDays int NOT NULL,
  InternalComments nvarchar(max) NULL,
  PhoneNumber nvarchar(20) NOT NULL,
  FaxNumber nvarchar(20) NOT NULL,
  WebsiteURL nvarchar(256) NOT NULL,
  DeliveryAddressLine1 nvarchar(60) NOT NULL,
  DeliveryAddressLine2 nvarchar(60) NULL,
  DeliveryPostalCode nvarchar(10) NOT NULL,
  DeliveryLocation geography NULL,
  PostalAddressLine1 nvarchar(60) NOT NULL,
  PostalAddressLine2 nvarchar(60) NULL,
  PostalPostalCode nvarchar(10) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
```

```
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_Suppliers_Archive
  ON Purchasing.Suppliers_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Purchasing
- WideWorldImporters

Used By 2

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetSupplierUpdates

Purchasing.SupplierTransactions

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	True
Partitioned Column	TransactionDate
Partition Scheme	PS_TransactionDate
File Groups	PRIMARY, PRIMARY, PRIMARY, PRIMARY, PRIMARY, PRIMARY
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	2353
Created	25-May-16 15:36:54
Last Modified	26-May-16 09:02:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	SupplierTransactionID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[TransactionID])	False	False	
	SupplierID	int	4	10	0	True				False	False	
	TransactionTypeID	int	4	10	0	True				False	False	
	PurchaseOrderID	int	4	10	0	False				False	False	
	PaymentMethodID	int	4	10	0	False				False	False	
	SupplierInvoiceNumber	nvarchar	20	0	0	False				False	False	

	TransactionDate	date	3	10	0	True				False	False	
	AmountExcludingTax	decimal	9	18	2	True				False	False	
	TaxAmount	decimal	9	18	2	True				False	False	
	TransactionAmount	decimal	9	18	2	True				False	False	
	OutstandingBalance	decimal	9	18	2	True				False	False	
	FinalizationDate	date	3	10	0	False				False	False	
	IsFinalized	bit	1	1	0	False				True	True	
	LastEditedBy	int	4	10	0	True				False	False	
	LastEditedWhen	datetime2	8	27	7	True			(sysdatetime())	False	False	

Computed Columns

Name	Definition
IsFinalized	(case when [FinalizationDate] IS NULL then CONVERT([bit],(0)) else CONVERT([bit],(1)) end)

Indexes

Key	Name	Columns	Unique	Type	Description
	CX_Purchasing_SupplierTransactions	TransactionDate	False		
	FK_Purchasing_SupplierTransactions_PaymentMethodID	TransactionDate, PaymentMethodID	False		
	FK_Purchasing_SupplierTransactions_PurchaseOrderID	TransactionDate, PurchaseOrderID	False		
	FK_Purchasing_SupplierTransactions_SupplierID	TransactionDate, SupplierID	False		
	FK_Purchasing_SupplierTransactions_TransactionTypeID	TransactionDate, TransactionTypeID	False		
	IX_Purchasing_SupplierTransactions_IsFinalized	TransactionDate, IsFinalized	False		
	PK_Purchasing_SupplierTransactions	SupplierTransactionID	True		

Foreign Keys

Name	Columns	Description
FK_Purchasing_SupplierTransactions_Application_People	PersonID	
FK_Purchasing_SupplierTransactions_PaymentMethodID_Application_PaymentMethods	PaymentMethodID	
FK_Purchasing_SupplierTransactions_PurchaseOrderID_Purchasing_PurchaseOrders	PurchaseOrderID	
FK_Purchasing_SupplierTransactions_SupplierID_Purchasing_Suppliers	SupplierID	
FK_Purchasing_SupplierTransactions_TransactionTypeID_Application_TransactionTypes	TransactionTypeID	

Extended Properties

Name	Value
Description	All financial transactions that are supplier-related

SQL Script

```
SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE Purchasing.SupplierTransactions (
  SupplierTransactionID int NOT NULL CONSTRAINT DF_Purchasing_SupplierTransactions_SupplierTransactionID DEFAULT (
  NEXT VALUE FOR [Sequences].[TransactionID]),
  SupplierID int NOT NULL,
  TransactionTypeID int NOT NULL,
  PurchaseOrderID int NULL,
  PaymentMethodID int NULL,
  SupplierInvoiceNumber nvarchar(20) NULL,
  TransactionDate date NOT NULL,
  AmountExcludingTax decimal(18, 2) NOT NULL,
  TaxAmount decimal(18, 2) NOT NULL,
  TransactionAmount decimal(18, 2) NOT NULL,
  OutstandingBalance decimal(18, 2) NOT NULL,
  FinalizationDate date NULL,
  IsFinalized AS (case when [FinalizationDate] IS NULL then CONVERT([bit],(0)) else CONVERT([bit],(1)) end)
  PERSISTED,
  LastEditedBy int NOT NULL,
  LastEditedWhen datetime2 NOT NULL CONSTRAINT DF_Purchasing_SupplierTransactions_LastEditedWhen DEFAULT (
  sysdatetime()),
  CONSTRAINT PK_Purchasing_SupplierTransactions PRIMARY KEY NONCLUSTERED (SupplierTransactionID) ON [PRIMARY]
)
ON PS_TransactionDate (TransactionDate)
GO

CREATE CLUSTERED INDEX CX_Purchasing_SupplierTransactions
  ON Purchasing.SupplierTransactions (TransactionDate)
  ON PS_TransactionDate (TransactionDate)
GO

CREATE INDEX FK_Purchasing_SupplierTransactions_PaymentMethodID
  ON Purchasing.SupplierTransactions (TransactionDate, PaymentMethodID)
  ON PS_TransactionDate (TransactionDate)
GO

CREATE INDEX FK_Purchasing_SupplierTransactions_PurchaseOrderID
  ON Purchasing.SupplierTransactions (TransactionDate, PurchaseOrderID)
  ON PS_TransactionDate (TransactionDate)
GO

CREATE INDEX FK_Purchasing_SupplierTransactions_SupplierID
  ON Purchasing.SupplierTransactions (TransactionDate, SupplierID)
  ON PS_TransactionDate (TransactionDate)
GO

CREATE INDEX FK_Purchasing_SupplierTransactions_TransactionTypeID
  ON Purchasing.SupplierTransactions (TransactionDate, TransactionTypeID)
  ON PS_TransactionDate (TransactionDate)
GO

CREATE INDEX IX_Purchasing_SupplierTransactions_IsFinalized
  ON Purchasing.SupplierTransactions (TransactionDate, IsFinalized)
  ON PS_TransactionDate (TransactionDate)
GO

ALTER TABLE Purchasing.SupplierTransactions
  ADD CONSTRAINT FK_Purchasing_SupplierTransactions_Application_People FOREIGN KEY (LastEditedBy) REFERENCES
  Application.People (PersonID)
GO

ALTER TABLE Purchasing.SupplierTransactions
  ADD CONSTRAINT FK_Purchasing_SupplierTransactions_PaymentMethodID_Application_PaymentMethods FOREIGN KEY (
  PaymentMethodID) REFERENCES Application.PaymentMethods (PaymentMethodID)
GO

ALTER TABLE Purchasing.SupplierTransactions
  ADD CONSTRAINT FK_Purchasing_SupplierTransactions_PurchaseOrderID_Purchasing_PurchaseOrders FOREIGN KEY (
  PurchaseOrderID) REFERENCES Purchasing.PurchaseOrders (PurchaseOrderID)
GO

ALTER TABLE Purchasing.SupplierTransactions
  ADD CONSTRAINT FK_Purchasing_SupplierTransactions_SupplierID_Purchasing_Suppliers FOREIGN KEY (SupplierID)
  REFERENCES Purchasing.Suppliers (SupplierID)
GO

ALTER TABLE Purchasing.SupplierTransactions
```

```
  ADD CONSTRAINT FK_Purchasing_SupplierTransactions_TransactionTypeID_Application_TransactionTypes FOREIGN KEY (
  TransactionTypeID) REFERENCES Application.TransactionTypes (TransactionTypeID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'All financial transactions that are supplier-
related', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used to refer to a supplier transaction within the
database', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'COLUMN', N'SupplierTransactionID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Supplier for this
transaction', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'COLUMN', N'SupplierID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Type of
transaction', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'COLUMN', N'TransactionTypeID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of an purchase order (for transactions associated with a purchase
order)', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'COLUMN', N'PurchaseOrderID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of a payment method (for transactions involving
payments)', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'COLUMN', N'PaymentMethodID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Invoice number for an invoice received from the
supplier', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'COLUMN', N'SupplierInvoiceNumber'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Date for the
transaction', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'COLUMN', N'TransactionDate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Transaction amount (excluding
tax)', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'COLUMN', N'AmountExcludingTax'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Tax amount
calculated', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'COLUMN', N'TaxAmount'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Transaction amount (including
tax)', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'COLUMN', N'TransactionAmount'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Amount still outstanding for this
transaction', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'COLUMN', N'OutstandingBalance'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Date that this transaction was finalized (if it has
been)', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'COLUMN', N'FinalizationDate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Is this transaction finalized (invoices, credits and payments have been
matched)', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'COLUMN', N'IsFinalized'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'INDEX', N'FK_Purchasing_SupplierTransactions_PaymentMe
thodID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'INDEX', N'FK_Purchasing_SupplierTransactions_PurchaseO
rderID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'INDEX', N'FK_Purchasing_SupplierTransactions_SupplierI
D'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'INDEX', N'FK_Purchasing_SupplierTransactions_Transacti
onTypeID'
```

```
GO

EXEC sys.sp_addextendedproperty N'Description', 'Index used to quickly locate unfinalized
transactions', 'SCHEMA', N'Purchasing', 'TABLE', N'SupplierTransactions', 'INDEX', N'IX_Purchasing_SupplierTransactions_
IsFinalized'
GO
```

Depends On 9

- Purchasing
- Application.PaymentMethods
- Application.People
- Application.TransactionTypes
- Purchasing.PurchaseOrders
- Purchasing.Suppliers
- PS_TransactionDate
- Sequences.TransactionID
- WideWorldImporters

Used By 1

- Integration.GetTransactionUpdates

Sales.BuyingGroups

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	2
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	BuyingGroupID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[BuyingGroupID])	False	False	
	BuyingGroupName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	PK_Sales_BuyingGroups	BuyingGroupID	True		

	UQ_Sales_BuyingGroups_BuyingGroupName	BuyingGroupName	True		

Foreign Keys

Name	Columns	Description
FK_Sales_BuyingGroups_Application_People	PersonID	

Extended Properties

Name	Value
Description	Customer organizations can be part of groups that exert greater buying power

SQL Script

```
CREATE TABLE Sales.BuyingGroups (
  BuyingGroupID int NOT NULL CONSTRAINT DF_Sales_BuyingGroups_BuyingGroupID DEFAULT (NEXT VALUE FOR [Sequences].
  [BuyingGroupID]),
  BuyingGroupName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Sales_BuyingGroups PRIMARY KEY CLUSTERED (BuyingGroupID),
  CONSTRAINT UQ_Sales_BuyingGroups_BuyingGroupName UNIQUE (BuyingGroupName)
)
ON [PRIMARY]
GO

ALTER TABLE Sales.BuyingGroups
  ADD CONSTRAINT FK_Sales_BuyingGroups_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.People
  (PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Customer organizations can be part of groups that exert greater buying
power', 'SCHEMA', N'Sales', 'TABLE', N'BuyingGroups'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a buying group within the
database', 'SCHEMA', N'Sales', 'TABLE', N'BuyingGroups', 'COLUMN', N'BuyingGroupID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Full name of a buying group that customers can be members
of', 'SCHEMA', N'Sales', 'TABLE', N'BuyingGroups', 'COLUMN', N'BuyingGroupName'
GO
```

Depends On 4

- Sales
- Application.People
- Sequences.BuyingGroupID
- WideWorldImporters

Used By 6

- Sales.Customers
- Sales.SpecialDeals
- Website.Customers
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetCustomerUpdates

Sales.BuyingGroups_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	0
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	BuyingGroupID	int	4	10	0	True				False	False	
	BuyingGroupName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_BuyingGroups_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Sales.BuyingGroups_Archive (
  BuyingGroupID int NOT NULL,
```

```
  BuyingGroupName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_BuyingGroups_Archive
  ON Sales.BuyingGroups_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Sales
- WideWorldImporters

Used By 2

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetCustomerUpdates

Sales.CustomerCategories

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	8
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	CustomerCategoryID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[CustomerCategoryID])	False	False	
	CustomerCategoryName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	PK_Sales_CustomerCategories	CustomerCategoryID	True		
	UQ_Sales_CustomerCategories_CustomerCategoryName	CustomerCategoryName	True		

Foreign Keys

Name	Columns	Description
FK_Sales_CustomerCategories_Application_People	PersonID	

Extended Properties

Name	Value
Description	Categories for customers (ie restaurants, cafes, supermarkets, etc.)

SQL Script

```
CREATE TABLE Sales.CustomerCategories (
  CustomerCategoryID int NOT NULL CONSTRAINT DF_Sales_CustomerCategories_CustomerCategoryID DEFAULT (NEXT VALUE FOR
  [Sequences].[CustomerCategoryID]),
  CustomerCategoryName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Sales_CustomerCategories PRIMARY KEY CLUSTERED (CustomerCategoryID),
  CONSTRAINT UQ_Sales_CustomerCategories_CustomerCategoryName UNIQUE (CustomerCategoryName)
)
ON [PRIMARY]
GO

ALTER TABLE Sales.CustomerCategories
  ADD CONSTRAINT FK_Sales_CustomerCategories_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.
  People (PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Categories for customers (ie restaurants, cafes, supermarkets,
etc.)', 'SCHEMA', N'Sales', 'TABLE', N'CustomerCategories'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a customer category within the
database', 'SCHEMA', N'Sales', 'TABLE', N'CustomerCategories', 'COLUMN', N'CustomerCategoryID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Full name of the category that customers can be assigned
to', 'SCHEMA', N'Sales', 'TABLE', N'CustomerCategories', 'COLUMN', N'CustomerCategoryName'
GO
```

Depends On 4

- Sales
- Application.People
- Sequences.CustomerCategoryID
- WideWorldImporters

Used By 6

- Sales.Customers
- Sales.SpecialDeals
- Website.Customers
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetCustomerUpdates

Sales.CustomerCategories_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	1
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	CustomerCategoryID	int	4	10	0	True				False	False	
	CustomerCategoryName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_CustomerCategories_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Sales.CustomerCategories_Archive (
  CustomerCategoryID int NOT NULL,
```

```
  CustomerCategoryName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_CustomerCategories_Archive
  ON Sales.CustomerCategories_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Sales
- WideWorldImporters

Used By 2

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetCustomerUpdates

Sales.Customers

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	651
Created	25-May-16 15:36:54
Last Modified	25-May-16 16:42:39

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	CustomerID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[CustomerID])	False	False	
	CustomerName	nvarchar	100	0	0	True				False	False	
	BillToCustomerID	int	4	10	0	True				False	False	
	CustomerCategoryID	int	4	10	0	True				False	False	
	BuyingGroupID	int	4	10	0	False				False	False	
	PrimaryContactPersonID	int	4	10	0	True				False	False	
	AlternateContactPersonID	int	4	10	0	False				False	False	

	DeliveryMethodID	int	4	10	0	True				False	False	
	DeliveryCityID	int	4	10	0	True				False	False	
	PostalCityID	int	4	10	0	True				False	False	
	CreditLimit	decimal	9	18	2	False				False	False	
	AccountOpenedDate	date	3	10	0	True				False	False	
	StandardDiscountPercentage	decimal	9	18	3	True				False	False	
	IsStatementSent	bit	1	1	0	True				False	False	
	IsOnCreditHold	bit	1	1	0	True				False	False	
	PaymentDays	int	4	10	0	True				False	False	
	PhoneNumber	nvarchar	20	0	0	True				False	False	
	FaxNumber	nvarchar	20	0	0	True				False	False	
	DeliveryRun	nvarchar	5	0	0	False				False	False	
	RunPosition	nvarchar	5	0	0	False				False	False	
	WebsiteURL	nvarchar	256	0	0	True				False	False	
	DeliveryAddressLine1	nvarchar	60	0	0	True				False	False	
	DeliveryAddressLine2	nvarchar	60	0	0	False				False	False	
	DeliveryPostalCode	nvarchar	10	0	0	True				False	False	
	DeliveryLocation	geography		0	0	False				False	False	
	PostalAddressLine1	nvarchar	60	0	0	True				False	False	
	PostalAddressLine2	nvarchar	60	0	0	False				False	False	
	PostalPostalCode	nvarchar	10	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	FK_Sales_Customers_AlternateContactPersonID	AlternateContactPersonID	False		
	FK_Sales_Customers_BuyingGroupID	BuyingGroupID	False		
	FK_Sales_Customers_CustomerCategoryID	CustomerCategoryID	False		
	FK_Sales_Customers_DeliveryCityID	DeliveryCityID	False		
	FK_Sales_Customers_DeliveryMethodID	DeliveryMethodID	False		
	FK_Sales_Customers_PostalCityID	PostalCityID	False		
	FK_Sales_Customers_PrimaryContactPersonID	PrimaryContactPersonID	False		
	IX_Sales_Customers_Perf_20160301_06	IsOnCreditHold, CustomerID, BillToCustomerID	False		
	PK_Sales_Customers	CustomerID	True		

	UQ_Sales_Customers_CustomerName	CustomerName	True		

Foreign Keys

Name	Columns	Description
FK_Sales_Customers_AlternateContactPersonID_Application_People	PersonID	
FK_Sales_Customers_Application_People	PersonID	
FK_Sales_Customers_BillToCustomerID_Sales_Customers	CustomerID	
FK_Sales_Customers_BuyingGroupID_Sales_BuyingGroups	BuyingGroupID	
FK_Sales_Customers_CustomerCategoryID_Sales_CustomerCategories	CustomerCategoryID	
FK_Sales_Customers_DeliveryCityID_Application_Cities	CityID	
FK_Sales_Customers_DeliveryMethodID_Application_DeliveryMethods	DeliveryMethodID	
FK_Sales_Customers_PostalCityID_Application_Cities	CityID	
FK_Sales_Customers_PrimaryContactPersonID_Application_People	PersonID	

Extended Properties

Name	Value
Description	Main entity tables for customers (organizations or individuals)

SQL Script

```
CREATE TABLE Sales.Customers (
  CustomerID int NOT NULL CONSTRAINT DF_Sales_Customers_CustomerID DEFAULT (NEXT VALUE FOR [Sequences].[CustomerID]
  ),
  CustomerName nvarchar(100) NOT NULL,
  BillToCustomerID int NOT NULL,
  CustomerCategoryID int NOT NULL,
  BuyingGroupID int NULL,
  PrimaryContactPersonID int NOT NULL,
  AlternateContactPersonID int NULL,
  DeliveryMethodID int NOT NULL,
  DeliveryCityID int NOT NULL,
  PostalCityID int NOT NULL,
  CreditLimit decimal(18, 2) NULL,
  AccountOpenedDate date NOT NULL,
  StandardDiscountPercentage decimal(18, 3) NOT NULL,
  IsStatementSent bit NOT NULL,
  IsOnCreditHold bit NOT NULL,
  PaymentDays int NOT NULL,
  PhoneNumber nvarchar(20) NOT NULL,
  FaxNumber nvarchar(20) NOT NULL,
  DeliveryRun nvarchar(5) NULL,
  RunPosition nvarchar(5) NULL,
  WebsiteURL nvarchar(256) NOT NULL,
  DeliveryAddressLine1 nvarchar(60) NOT NULL,
  DeliveryAddressLine2 nvarchar(60) NULL,
  DeliveryPostalCode nvarchar(10) NOT NULL,
  DeliveryLocation geography NULL,
  PostalAddressLine1 nvarchar(60) NOT NULL,
  PostalAddressLine2 nvarchar(60) NULL,
  PostalPostalCode nvarchar(10) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Sales_Customers PRIMARY KEY CLUSTERED (CustomerID),
  CONSTRAINT UQ_Sales_Customers_CustomerName UNIQUE (CustomerName)
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Customers_AlternateContactPersonID
  ON Sales.Customers (AlternateContactPersonID)
  ON [PRIMARY]
```

```
GO

CREATE INDEX FK_Sales_Customers_BuyingGroupID
  ON Sales.Customers (BuyingGroupID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Customers_CustomerCategoryID
  ON Sales.Customers (CustomerCategoryID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Customers_DeliveryCityID
  ON Sales.Customers (DeliveryCityID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Customers_DeliveryMethodID
  ON Sales.Customers (DeliveryMethodID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Customers_PostalCityID
  ON Sales.Customers (PostalCityID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Customers_PrimaryContactPersonID
  ON Sales.Customers (PrimaryContactPersonID)
  ON [PRIMARY]
GO

CREATE INDEX IX_Sales_Customers_Perf_20160301_06
  ON Sales.Customers (IsOnCreditHold, CustomerID, BillToCustomerID)
  INCLUDE (PrimaryContactPersonID)
  ON [PRIMARY]
GO

ALTER TABLE Sales.Customers
  ADD CONSTRAINT FK_Sales_Customers_AlternateContactPersonID_Application_People FOREIGN KEY (
  AlternateContactPersonID) REFERENCES Application.People (PersonID)
GO

ALTER TABLE Sales.Customers
  ADD CONSTRAINT FK_Sales_Customers_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.People (
  PersonID)
GO

ALTER TABLE Sales.Customers
  ADD CONSTRAINT FK_Sales_Customers_BillToCustomerID_Sales_Customers FOREIGN KEY (BillToCustomerID) REFERENCES
  Sales.Customers (CustomerID)
GO

ALTER TABLE Sales.Customers
  ADD CONSTRAINT FK_Sales_Customers_BuyingGroupID_Sales_BuyingGroups FOREIGN KEY (BuyingGroupID) REFERENCES Sales.
  BuyingGroups (BuyingGroupID)
GO

ALTER TABLE Sales.Customers
  ADD CONSTRAINT FK_Sales_Customers_CustomerCategoryID_Sales_CustomerCategories FOREIGN KEY (CustomerCategoryID)
  REFERENCES Sales.CustomerCategories (CustomerCategoryID)
GO

ALTER TABLE Sales.Customers
  ADD CONSTRAINT FK_Sales_Customers_DeliveryCityID_Application_Cities FOREIGN KEY (DeliveryCityID) REFERENCES
  Application.Cities (CityID)
GO

ALTER TABLE Sales.Customers
  ADD CONSTRAINT FK_Sales_Customers_DeliveryMethodID_Application_DeliveryMethods FOREIGN KEY (DeliveryMethodID)
  REFERENCES Application.DeliveryMethods (DeliveryMethodID)
GO

ALTER TABLE Sales.Customers
  ADD CONSTRAINT FK_Sales_Customers_PostalCityID_Application_Cities FOREIGN KEY (PostalCityID) REFERENCES
  Application.Cities (CityID)
GO

ALTER TABLE Sales.Customers
  ADD CONSTRAINT FK_Sales_Customers_PrimaryContactPersonID_Application_People FOREIGN KEY (PrimaryContactPersonID)
```

```
  REFERENCES Application.People (PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Main entity tables for customers (organizations or
individuals)', 'SCHEMA', N'Sales', 'TABLE', N'Customers'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a customer within the
database', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'CustomerID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Customer''s full name (usually a trading
name)', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'CustomerName'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Customer that this is billed to (usually the same customer but can be
another
parent company)', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'BillToCustomerID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Customer''s
category', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'CustomerCategoryID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Customer''s buying group
(optional)', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'BuyingGroupID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Primary
contact', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'PrimaryContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Alternate
contact', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'AlternateContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Standard delivery method for stock items sent to this
customer', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'DeliveryMethodID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of the delivery city for this
address', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'DeliveryCityID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of the postal city for this
address', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'PostalCityID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Credit limit for this customer (NULL if
unlimited)', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'CreditLimit'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Date this customer account was
opened', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'AccountOpenedDate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Standard discount offered to this
customer', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'StandardDiscountPercentage'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Is a statement sent to this customer? (Or do they just pay on each
invoice?)', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'IsStatementSent'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Is this customer on credit hold? (Prevents further deliveries to this
customer)', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'IsOnCreditHold'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Number of days for payment of an invoice (ie payment
terms)', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'PaymentDays'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Phone
number', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'PhoneNumber'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Fax number
', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'FaxNumber'
GO
```

```
EXEC sys.sp_addextendedproperty N'Description', 'Normal delivery run for this
customer', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'DeliveryRun'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Normal position in the delivery run for this
customer', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'RunPosition'
GO

EXEC sys.sp_addextendedproperty N'Description', 'URL for the website for this
customer', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'WebsiteURL'
GO

EXEC sys.sp_addextendedproperty N'Description', 'First delivery address line for the
customer', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'DeliveryAddressLine1'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Second delivery address line for the
customer', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'DeliveryAddressLine2'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Delivery postal code for the
customer', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'DeliveryPostalCode'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Geographic location for the customer''s
office/warehouse', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'DeliveryLocation'
GO

EXEC sys.sp_addextendedproperty N'Description', 'First postal address line for the
customer', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'PostalAddressLine1'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Second postal address line for the
customer', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'PostalAddressLine2'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Postal code for the customer when sending by
mail', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'COLUMN', N'PostalPostalCode'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'INDEX', N'FK_Sales_Customers_AlternateContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'INDEX', N'FK_Sales_Customers_BuyingGroupID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'INDEX', N'FK_Sales_Customers_CustomerCategoryID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'INDEX', N'FK_Sales_Customers_DeliveryCityID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'INDEX', N'FK_Sales_Customers_DeliveryMethodID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'INDEX', N'FK_Sales_Customers_PostalCityID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'INDEX', N'FK_Sales_Customers_PrimaryContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Improves performance of order picking and
invoicing', 'SCHEMA', N'Sales', 'TABLE', N'Customers', 'INDEX', N'IX_Sales_Customers_Perf_20160301_06'
GO
```

Depends On 9

- Sales
- Application.Cities
- Application.DeliveryMethods

- Application.People
- Sales.BuyingGroups
- Sales.CustomerCategories
- Sales.Customers
- Sequences.CustomerID
- WideWorldImporters

Used By 16

- Sales.Customers
- Sales.CustomerTransactions
- Sales.Invoices
- Sales.Orders
- Sales.SpecialDeals
- Warehouse.StockItemTransactions
- Website.Customers
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetCustomerUpdates
- Integration.GetOrderUpdates
- Integration.GetSaleUpdates
- Website.InvoiceCustomerOrders
- Website.SearchForCustomers
- Website.SearchForPeople
- Website.CalculateCustomerPrice

Sales.Customers_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	35
Created	25-May-16 15:36:54
Last Modified	25-May-16 16:42:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	CustomerID	int	4	10	0	True				False	False	
	CustomerName	nvarchar	100	0	0	True				False	False	
	BillToCustomerID	int	4	10	0	True				False	False	
	CustomerCategoryID	int	4	10	0	True				False	False	
	BuyingGroupID	int	4	10	0	False				False	False	
	PrimaryContactPersonID	int	4	10	0	True				False	False	
	AlternateContactPersonID	int	4	10	0	False				False	False	
	DeliveryMethodID	int	4	10	0	True				False	False	
	DeliveryCityID	int	4	10	0	True				False	False	
	PostalCityID	int	4	10	0	True				False	False	
	CreditLimit	decimal	9	18	2	False				False	False	

	AccountOpenedDate	date	3	10	0	True				False	False	
	StandardDiscountPercentage	decimal	9	18	3	True				False	False	
	IsStatementSent	bit	1	1	0	True				False	False	
	IsOnCreditHold	bit	1	1	0	True				False	False	
	PaymentDays	int	4	10	0	True				False	False	
	PhoneNumber	nvarchar	20	0	0	True				False	False	
	FaxNumber	nvarchar	20	0	0	True				False	False	
	DeliveryRun	nvarchar	5	0	0	False				False	False	
	RunPosition	nvarchar	5	0	0	False				False	False	
	WebsiteURL	nvarchar	256	0	0	True				False	False	
	DeliveryAddressLine1	nvarchar	60	0	0	True				False	False	
	DeliveryAddressLine2	nvarchar	60	0	0	False				False	False	
	DeliveryPostalCode	nvarchar	10	0	0	True				False	False	
	DeliveryLocation	geography		0	0	False				False	False	
	PostalAddressLine1	nvarchar	60	0	0	True				False	False	
	PostalAddressLine2	nvarchar	60	0	0	False				False	False	
	PostalPostalCode	nvarchar	10	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_Customers_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Sales.Customers_Archive (
  CustomerID int NOT NULL,
  CustomerName nvarchar(100) NOT NULL,
  BillToCustomerID int NOT NULL,
  CustomerCategoryID int NOT NULL,
  BuyingGroupID int NULL,
  PrimaryContactPersonID int NOT NULL,
  AlternateContactPersonID int NULL,
  DeliveryMethodID int NOT NULL,
  DeliveryCityID int NOT NULL,
  PostalCityID int NOT NULL,
  CreditLimit decimal(18, 2) NULL,
  AccountOpenedDate date NOT NULL,
  StandardDiscountPercentage decimal(18, 3) NOT NULL,
  IsStatementSent bit NOT NULL,
  IsOnCreditHold bit NOT NULL,
  PaymentDays int NOT NULL,
  PhoneNumber nvarchar(20) NOT NULL,
  FaxNumber nvarchar(20) NOT NULL,
  DeliveryRun nvarchar(5) NULL,
  RunPosition nvarchar(5) NULL,
  WebsiteURL nvarchar(256) NOT NULL,
  DeliveryAddressLine1 nvarchar(60) NOT NULL,
  DeliveryAddressLine2 nvarchar(60) NULL,
  DeliveryPostalCode nvarchar(10) NOT NULL,
  DeliveryLocation geography NULL,
```

```
  PostalAddressLine1 nvarchar(60) NOT NULL,
  PostalAddressLine2 nvarchar(60) NULL,
  PostalPostalCode nvarchar(10) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_Customers_Archive
  ON Sales.Customers_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Sales
- WideWorldImporters

Used By 2

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetCustomerUpdates

Sales.CustomerTransactions

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	True
Partitioned Column	TransactionDate
Partition Scheme	PS_TransactionDate
File Groups	PRIMARY, PRIMARY, PRIMARY, PRIMARY, PRIMARY, PRIMARY
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	92420
Created	25-May-16 15:36:54
Last Modified	26-May-16 09:02:33

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	CustomerTransactionID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[TransactionID])	False	False	
	CustomerID	int	4	10	0	True				False	False	
	TransactionTypeID	int	4	10	0	True				False	False	
	InvoiceID	int	4	10	0	False				False	False	
	PaymentMethodID	int	4	10	0	False				False	False	
	TransactionDate	date	3	10	0	True				False	False	

	AmountExcludingTax	decimal	9	18	2	True				False	False	
	TaxAmount	decimal	9	18	2	True				False	False	
	TransactionAmount	decimal	9	18	2	True				False	False	
	OutstandingBalance	decimal	9	18	2	True				False	False	
	FinalizationDate	date	3	10	0	False				False	False	
	IsFinalized	bit	1	1	0	False				True	True	
	LastEditedBy	int	4	10	0	True				False	False	
	LastEditedWhen	datetime2	8	27	7	True			(sysdatetime())	False	False	

Computed Columns

Name	Definition
IsFinalized	(case when [FinalizationDate] IS NULL then CONVERT([bit],(0)) else CONVERT([bit],(1)) end)

Indexes

Key	Name	Columns	Unique	Type	Description
	CX_Sales_CustomerTransactions	TransactionDate	False		
	FK_Sales_CustomerTransactions_CustomerID	TransactionDate, CustomerID	False		
	FK_Sales_CustomerTransactions_InvoiceID	TransactionDate, InvoiceID	False		
	FK_Sales_CustomerTransactions_PaymentMethodID	TransactionDate, PaymentMethodID	False		
	FK_Sales_CustomerTransactions_TransactionTypeID	TransactionDate, TransactionTypeID	False		
	IX_Sales_CustomerTransactions_IsFinalized	TransactionDate, IsFinalized	False		
	PK_Sales_CustomerTransactions	CustomerTransactionID	True		

Foreign Keys

Name	Columns	Description
FK_Sales_CustomerTransactions_Application_People	PersonID	
FK_Sales_CustomerTransactions_CustomerID_Sales_Customers	CustomerID	
FK_Sales_CustomerTransactions_InvoiceID_Sales_Invoices	InvoiceID	
FK_Sales_CustomerTransactions_PaymentMethodID_Application_PaymentMethods	PaymentMethodID	
FK_Sales_CustomerTransactions_TransactionTypeID_Application_TransactionTypes	TransactionTypeID	

Extended Properties

Name	Value
Description	All financial transactions that are customer-related

SQL Script

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE Sales.CustomerTransactions (
  CustomerTransactionID int NOT NULL CONSTRAINT DF_Sales_CustomerTransactions_CustomerTransactionID DEFAULT (NEXT
  VALUE FOR [Sequences].[TransactionID]),
  CustomerID int NOT NULL,
  TransactionTypeID int NOT NULL,
  InvoiceID int NULL,
  PaymentMethodID int NULL,
  TransactionDate date NOT NULL,
  AmountExcludingTax decimal(18, 2) NOT NULL,
  TaxAmount decimal(18, 2) NOT NULL,
  TransactionAmount decimal(18, 2) NOT NULL,
  OutstandingBalance decimal(18, 2) NOT NULL,
  FinalizationDate date NULL,
  IsFinalized AS (case when [FinalizationDate] IS NULL then CONVERT([bit],(0)) else CONVERT([bit],(1)) end)
  PERSISTED,
  LastEditedBy int NOT NULL,
  LastEditedWhen datetime2 NOT NULL CONSTRAINT DF_Sales_CustomerTransactions_LastEditedWhen DEFAULT (sysdatetime())
  ,
  CONSTRAINT PK_Sales_CustomerTransactions PRIMARY KEY NONCLUSTERED (CustomerTransactionID) ON [PRIMARY]
)
ON PS_TransactionDate (TransactionDate)
GO

CREATE CLUSTERED INDEX CX_Sales_CustomerTransactions
  ON Sales.CustomerTransactions (TransactionDate)
  ON PS_TransactionDate (TransactionDate)
GO

CREATE INDEX FK_Sales_CustomerTransactions_CustomerID
  ON Sales.CustomerTransactions (TransactionDate, CustomerID)
  ON PS_TransactionDate (TransactionDate)
GO

CREATE INDEX FK_Sales_CustomerTransactions_InvoiceID
  ON Sales.CustomerTransactions (TransactionDate, InvoiceID)
  ON PS_TransactionDate (TransactionDate)
GO

CREATE INDEX FK_Sales_CustomerTransactions_PaymentMethodID
  ON Sales.CustomerTransactions (TransactionDate, PaymentMethodID)
  ON PS_TransactionDate (TransactionDate)
GO

CREATE INDEX FK_Sales_CustomerTransactions_TransactionTypeID
  ON Sales.CustomerTransactions (TransactionDate, TransactionTypeID)
  ON PS_TransactionDate (TransactionDate)
GO

CREATE INDEX IX_Sales_CustomerTransactions_IsFinalized
  ON Sales.CustomerTransactions (TransactionDate, IsFinalized)
  ON PS_TransactionDate (TransactionDate)
GO

ALTER TABLE Sales.CustomerTransactions
  ADD CONSTRAINT FK_Sales_CustomerTransactions_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application
  .People (PersonID)
GO

ALTER TABLE Sales.CustomerTransactions
  ADD CONSTRAINT FK_Sales_CustomerTransactions_CustomerID_Sales_Customers FOREIGN KEY (CustomerID) REFERENCES Sales
  .Customers (CustomerID)
GO

ALTER TABLE Sales.CustomerTransactions
  ADD CONSTRAINT FK_Sales_CustomerTransactions_InvoiceID_Sales_Invoices FOREIGN KEY (InvoiceID) REFERENCES Sales.
  Invoices (InvoiceID)
GO

ALTER TABLE Sales.CustomerTransactions
  ADD CONSTRAINT FK_Sales_CustomerTransactions_PaymentMethodID_Application_PaymentMethods FOREIGN KEY (
  PaymentMethodID) REFERENCES Application.PaymentMethods (PaymentMethodID)
GO

ALTER TABLE Sales.CustomerTransactions
  ADD CONSTRAINT FK_Sales_CustomerTransactions_TransactionTypeID_Application_TransactionTypes FOREIGN KEY (
  TransactionTypeID) REFERENCES Application.TransactionTypes (TransactionTypeID)
GO
```

```
EXEC sys.sp_addextendedproperty N'Description', N'All financial transactions that are customer-
related', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used to refer to a customer transaction within the
database', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'COLUMN', N'CustomerTransactionID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Customer for this
transaction', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'COLUMN', N'CustomerID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Type of
transaction', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'COLUMN', N'TransactionTypeID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of an invoice (for transactions associated with an
invoice)', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'COLUMN', N'InvoiceID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of a payment method (for transactions involving
payments)', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'COLUMN', N'PaymentMethodID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Date for the
transaction', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'COLUMN', N'TransactionDate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Transaction amount (excluding
tax)', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'COLUMN', N'AmountExcludingTax'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Tax amount
calculated', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'COLUMN', N'TaxAmount'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Transaction amount (including
tax)', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'COLUMN', N'TransactionAmount'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Amount still outstanding for this
transaction', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'COLUMN', N'OutstandingBalance'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Date that this transaction was finalized (if it has
been)', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'COLUMN', N'FinalizationDate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Is this transaction finalized (invoices, credits and payments have been
matched)', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'COLUMN', N'IsFinalized'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'INDEX', N'FK_Sales_CustomerTransactions_CustomerID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'INDEX', N'FK_Sales_CustomerTransactions_InvoiceID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'INDEX', N'FK_Sales_CustomerTransactions_PaymentMethodID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'INDEX', N'FK_Sales_CustomerTransactions_TransactionTypeID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Allows quick location of unfinalized
transactions', 'SCHEMA', N'Sales', 'TABLE', N'CustomerTransactions', 'INDEX', N'IX_Sales_CustomerTransactions_IsFinalize
d'
GO
```

Depends On 8

- Sales
- Application.PaymentMethods

- Application.People
- Application.TransactionTypes
- Sales.Customers
- Sales.Invoices
- PS_TransactionDate
- Sequences.TransactionID

Used By 2

- Integration.GetTransactionUpdates
- Website.InvoiceCustomerOrders

Sales.InvoiceLines

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	222091
Created	25-May-16 15:36:54
Last Modified	27-May-16 10:06:30

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	InvoiceLineID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[InvoiceLineID])	False	False	
	InvoiceID	int	4	10	0	True				False	False	
	StockItemID	int	4	10	0	True				False	False	
	Description	nvarchar	100	0	0	True				False	False	
	PackageTypeID	int	4	10	0	True				False	False	
	Quantity	int	4	10	0	True				False	False	
	UnitPrice	decimal	9	18	2	False				False	False	

	TaxRate	decimal	9	18	3	True				False	False	
	TaxAmount	decimal	9	18	2	True				False	False	
	LineProfit	decimal	9	18	2	True				False	False	
	ExtendedPrice	decimal	9	18	2	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	LastEditedWhen	datetime2	8	27	7	True			(sysdatetime())	False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	FK_Sales_InvoiceLines_InvoiceID	InvoiceID	False		
	FK_Sales_InvoiceLines_PackageTypeID	PackageTypeID	False		
	FK_Sales_InvoiceLines_StockItemID	StockItemID	False		
	NCCX_Sales_InvoiceLines	InvoiceID, StockItemID, Quantity, UnitPrice, LineProfit, LastEditedWhen	False		
	PK_Sales_InvoiceLines	InvoiceLineID	True		

Foreign Keys

Name	Columns	Description
FK_Sales_InvoiceLines_Application_People	PersonID	
FK_Sales_InvoiceLines_InvoiceID_Sales_Invoices	InvoiceID	
FK_Sales_InvoiceLines_PackageTypeID_Warehouse_PackageTypes	PackageTypeID	
FK_Sales_InvoiceLines_StockItemID_Warehouse_StockItems	StockItemID	

Extended Properties

Name	Value
Description	Detail lines from customer invoices

SQL Script

```
CREATE TABLE Sales.InvoiceLines (
  InvoiceLineID int NOT NULL CONSTRAINT DF_Sales_InvoiceLines_InvoiceLineID DEFAULT (NEXT VALUE FOR [Sequences].
  [InvoiceLineID]),
  InvoiceID int NOT NULL,
  StockItemID int NOT NULL,
  Description nvarchar(100) NOT NULL,
  PackageTypeID int NOT NULL,
  Quantity int NOT NULL,
  UnitPrice decimal(18, 2) NULL,
  TaxRate decimal(18, 3) NOT NULL,
  TaxAmount decimal(18, 2) NOT NULL,
  LineProfit decimal(18, 2) NOT NULL,
  ExtendedPrice decimal(18, 2) NOT NULL,
  LastEditedBy int NOT NULL,
  LastEditedWhen datetime2 NOT NULL CONSTRAINT DF_Sales_InvoiceLines_LastEditedWhen DEFAULT (sysdatetime()),
  CONSTRAINT PK_Sales_InvoiceLines PRIMARY KEY CLUSTERED (InvoiceLineID)
)
ON [PRIMARY]
GO

CREATE INDEX FK_Sales_InvoiceLines_InvoiceID
  ON Sales.InvoiceLines (InvoiceID)
```

```
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_InvoiceLines_PackageTypeID
  ON Sales.InvoiceLines (PackageTypeID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_InvoiceLines_StockItemID
  ON Sales.InvoiceLines (StockItemID)
  ON [PRIMARY]
GO

CREATE COLUMNSTORE INDEX NCCX_Sales_InvoiceLines
  ON Sales.InvoiceLines (InvoiceID, StockItemID, Quantity, UnitPrice, LineProfit, LastEditedWhen)
GO

ALTER TABLE Sales.InvoiceLines
  ADD CONSTRAINT FK_Sales_InvoiceLines_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.People
  (PersonID)
GO

ALTER TABLE Sales.InvoiceLines
  ADD CONSTRAINT FK_Sales_InvoiceLines_InvoiceID_Sales_Invoices FOREIGN KEY (InvoiceID) REFERENCES Sales.Invoices (
  InvoiceID)
GO

ALTER TABLE Sales.InvoiceLines
  ADD CONSTRAINT FK_Sales_InvoiceLines_PackageTypeID_Warehouse_PackageTypes FOREIGN KEY (PackageTypeID) REFERENCES
  Warehouse.PackageTypes (PackageTypeID)
GO

ALTER TABLE Sales.InvoiceLines
  ADD CONSTRAINT FK_Sales_InvoiceLines_StockItemID_Warehouse_StockItems FOREIGN KEY (StockItemID) REFERENCES
  Warehouse.StockItems (StockItemID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Detail lines from customer
invoices', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a line on an invoice within the
database', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'COLUMN', N'InvoiceLineID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Invoice that this line is associated
with', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'COLUMN', N'InvoiceID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Stock item for this invoice
line', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'COLUMN', N'StockItemID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Description of the item supplied (Usually the stock item name but can
be
overridden)', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'COLUMN', N'Description'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Type of package
supplied', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'COLUMN', N'PackageTypeID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Quantity
supplied', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'COLUMN', N'Quantity'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Unit price
charged', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'COLUMN', N'UnitPrice'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Tax rate to be
applied', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'COLUMN', N'TaxRate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Tax amount
calculated', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'COLUMN', N'TaxAmount'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Profit made on this line item at current cost
```

```
price', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'COLUMN', N'LineProfit'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Extended line price
charged', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'COLUMN', N'ExtendedPrice'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'INDEX', N'FK_Sales_InvoiceLines_InvoiceID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'INDEX', N'FK_Sales_InvoiceLines_PackageTypeID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'InvoiceLines', 'INDEX', N'FK_Sales_InvoiceLines_StockItemID'
GO
```

Depends On 7

- Sales
- Application.People
- Sales.Invoices
- Warehouse.PackageTypes
- Warehouse.StockItems
- Sequences.InvoiceLineID
- WideWorldImporters

Used By 2

- Integration.GetSaleUpdates
- Website.InvoiceCustomerOrders

Sales.Invoices

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	67274
Created	25-May-16 15:36:54
Last Modified	25-May-16 15:36:54

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	InvoiceID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[InvoiceID])	False	False	
	CustomerID	int	4	10	0	True				False	False	
	BillToCustomerID	int	4	10	0	True				False	False	
	OrderID	int	4	10	0	False				False	False	
	DeliveryMethodID	int	4	10	0	True				False	False	
	ContactPersonID	int	4	10	0	True				False	False	
	AccountsPersonID	int	4	10	0	True				False	False	

	SalespersonPersonID	int	4	10	0	True				False	False	
	PackedByPersonID	int	4	10	0	True				False	False	
	InvoiceDate	date	3	10	0	True				False	False	
	CustomerPurchaseOrderNumber	nvarchar	20	0	0	False				False	False	
	IsCreditNote	bit	1	1	0	True				False	False	
	CreditNoteReason	nvarchar		0	0	False				False	False	
	Comments	nvarchar		0	0	False				False	False	
	DeliveryInstructions	nvarchar		0	0	False				False	False	
	InternalComments	nvarchar		0	0	False				False	False	
	TotalDryItems	int	4	10	0	True				False	False	
	TotalChillerItems	int	4	10	0	True				False	False	
	DeliveryRun	nvarchar	5	0	0	False				False	False	
	RunPosition	nvarchar	5	0	0	False				False	False	
	ReturnedDeliveryData	nvarchar		0	0	False				False	False	
	ConfirmedDeliveryTime	datetime2	8	27	7	False				True	False	
	ConfirmedReceivedBy	nvarchar	4000	0	0	False				True	False	
	LastEditedBy	int	4	10	0	True				False	False	
	LastEditedWhen	datetime2	8	27	7	True			(sysdatetime())	False	False	

Computed Columns

Name	Definition
ConfirmedDeliveryTime	(TRY_CONVERT([datetime2](7),json_value([ReturnedDeliveryData],N'$.DeliveredWhen'),(126)))
ConfirmedReceivedBy	(json_value([ReturnedDeliveryData],N'$.ReceivedBy'))

Indexes

Key	Name	Columns	Unique	Type	Description
	FK_Sales_Invoices_AccountsPersonID	AccountsPersonID	False		
	FK_Sales_Invoices_BillToCustomerID	BillToCustomerID	False		
	FK_Sales_Invoices_ContactPersonID	ContactPersonID	False		
	FK_Sales_Invoices_CustomerID	CustomerID	False		
	FK_Sales_Invoices_DeliveryMethodID	DeliveryMethodID	False		
	FK_Sales_Invoices_OrderID	OrderID	False		
	FK_Sales_Invoices_PackedByPersonID	PackedByPersonID	False		
	FK_Sales_Invoices_SalespersonPersonID	SalespersonPersonID	False		
	IX_Sales_Invoices_ConfirmedDeliveryTime	ConfirmedDeliveryTime	False		
	PK_Sales_Invoices	InvoiceID	True		

Check Constraints

Name	Columns	Condition	Description
CK_Sales_Invoices_ReturnedDeliveryData_Must_Be_Valid_JSON	ReturnedDeliveryData	([ReturnedDeliveryData] IS NULL OR isjson([ReturnedDeliveryData])<>(0))	

Foreign Keys

Name	Columns	Description
FK_Sales_Invoices_AccountsPersonID_Application_People	PersonID	
FK_Sales_Invoices_Application_People	PersonID	
FK_Sales_Invoices_BillToCustomerID_Sales_Customers	CustomerID	
FK_Sales_Invoices_ContactPersonID_Application_People	PersonID	
FK_Sales_Invoices_CustomerID_Sales_Customers	CustomerID	
FK_Sales_Invoices_DeliveryMethodID_Application_DeliveryMethods	DeliveryMethodID	
FK_Sales_Invoices_OrderID_Sales_Orders	OrderID	
FK_Sales_Invoices_PackedByPersonID_Application_People	PersonID	
FK_Sales_Invoices_SalespersonPersonID_Application_People	PersonID	

Extended Properties

Name	Value
Description	Details of customer invoices

SQL Script

```
CREATE TABLE Sales.Invoices (
  InvoiceID int NOT NULL CONSTRAINT DF_Sales_Invoices_InvoiceID DEFAULT (NEXT VALUE FOR [Sequences].[InvoiceID]),
  CustomerID int NOT NULL,
  BillToCustomerID int NOT NULL,
  OrderID int NULL,
  DeliveryMethodID int NOT NULL,
  ContactPersonID int NOT NULL,
  AccountsPersonID int NOT NULL,
  SalespersonPersonID int NOT NULL,
  PackedByPersonID int NOT NULL,
  InvoiceDate date NOT NULL,
  CustomerPurchaseOrderNumber nvarchar(20) NULL,
  IsCreditNote bit NOT NULL,
  CreditNoteReason nvarchar(max) NULL,
  Comments nvarchar(max) NULL,
  DeliveryInstructions nvarchar(max) NULL,
  InternalComments nvarchar(max) NULL,
  TotalDryItems int NOT NULL,
  TotalChillerItems int NOT NULL,
  DeliveryRun nvarchar(5) NULL,
  RunPosition nvarchar(5) NULL,
  ReturnedDeliveryData nvarchar(max) NULL,
  ConfirmedDeliveryTime AS (TRY_CONVERT([datetime2](7),json_value([ReturnedDeliveryData],N'$.DeliveredWhen'),(126))),
  ConfirmedReceivedBy AS (json_value([ReturnedDeliveryData],N'$.ReceivedBy')),
  LastEditedBy int NOT NULL,
  LastEditedWhen datetime2 NOT NULL CONSTRAINT DF_Sales_Invoices_LastEditedWhen DEFAULT (sysdatetime()),
  CONSTRAINT PK_Sales_Invoices PRIMARY KEY CLUSTERED (InvoiceID),
  CONSTRAINT CK_Sales_Invoices_ReturnedDeliveryData_Must_Be_Valid_JSON CHECK ([ReturnedDeliveryData] IS NULL OR
  isjson([ReturnedDeliveryData])<>(0))
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Invoices_AccountsPersonID
  ON Sales.Invoices (AccountsPersonID)
  ON [PRIMARY]
GO
```

```
CREATE INDEX FK_Sales_Invoices_BillToCustomerID
  ON Sales.Invoices (BillToCustomerID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Invoices_ContactPersonID
  ON Sales.Invoices (ContactPersonID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Invoices_CustomerID
  ON Sales.Invoices (CustomerID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Invoices_DeliveryMethodID
  ON Sales.Invoices (DeliveryMethodID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Invoices_OrderID
  ON Sales.Invoices (OrderID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Invoices_PackedByPersonID
  ON Sales.Invoices (PackedByPersonID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Invoices_SalespersonPersonID
  ON Sales.Invoices (SalespersonPersonID)
  ON [PRIMARY]
GO

CREATE INDEX IX_Sales_Invoices_ConfirmedDeliveryTime
  ON Sales.Invoices (ConfirmedDeliveryTime)
  INCLUDE (ConfirmedReceivedBy)
  ON [PRIMARY]
GO

ALTER TABLE Sales.Invoices
  ADD CONSTRAINT FK_Sales_Invoices_AccountsPersonID_Application_People FOREIGN KEY (AccountsPersonID) REFERENCES
  Application.People (PersonID)
GO

ALTER TABLE Sales.Invoices
  ADD CONSTRAINT FK_Sales_Invoices_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.People (
  PersonID)
GO

ALTER TABLE Sales.Invoices
  ADD CONSTRAINT FK_Sales_Invoices_BillToCustomerID_Sales_Customers FOREIGN KEY (BillToCustomerID) REFERENCES Sales
  .Customers (CustomerID)
GO

ALTER TABLE Sales.Invoices
  ADD CONSTRAINT FK_Sales_Invoices_ContactPersonID_Application_People FOREIGN KEY (ContactPersonID) REFERENCES
  Application.People (PersonID)
GO

ALTER TABLE Sales.Invoices
  ADD CONSTRAINT FK_Sales_Invoices_CustomerID_Sales_Customers FOREIGN KEY (CustomerID) REFERENCES Sales.Customers (
  CustomerID)
GO

ALTER TABLE Sales.Invoices
  ADD CONSTRAINT FK_Sales_Invoices_DeliveryMethodID_Application_DeliveryMethods FOREIGN KEY (DeliveryMethodID)
  REFERENCES Application.DeliveryMethods (DeliveryMethodID)
GO

ALTER TABLE Sales.Invoices
  ADD CONSTRAINT FK_Sales_Invoices_OrderID_Sales_Orders FOREIGN KEY (OrderID) REFERENCES Sales.Orders (OrderID)
GO

ALTER TABLE Sales.Invoices
  ADD CONSTRAINT FK_Sales_Invoices_PackedByPersonID_Application_People FOREIGN KEY (PackedByPersonID) REFERENCES
  Application.People (PersonID)
GO

ALTER TABLE Sales.Invoices
```

```
  ADD CONSTRAINT FK_Sales_Invoices_SalespersonPersonID_Application_People FOREIGN KEY (SalespersonPersonID)
  REFERENCES Application.People (PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Details of customer invoices', 'SCHEMA', N'Sales', 'TABLE', N'Invoices'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to an invoice within the
database', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'InvoiceID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Customer for this
invoice', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'CustomerID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Bill to customer for this invoice (invoices might be billed to a head
office)', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'BillToCustomerID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Sales order (if any) for this
invoice', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'OrderID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'How these stock items are beign
delivered', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'DeliveryMethodID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Customer contact for this
invoice', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'ContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Customer accounts contact for this
invoice', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'AccountsPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Salesperson for this
invoice', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'SalespersonPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Person who packed this shipment (or checked the
packing)', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'PackedByPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Date that this invoice was
raised', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'InvoiceDate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Purchase Order Number received from
customer', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'CustomerPurchaseOrderNumber'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Is this a credit note (rather than an
invoice)', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'IsCreditNote'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Reason that this credit note needed to be generated (if
applicable)', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'CreditNoteReason'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Any comments related to this invoice (sent to
customer)', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'Comments'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Any comments related to delivery (sent to
customer)', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'DeliveryInstructions'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Any internal comments related to this invoice (not sent to the
customer)', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'InternalComments'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Total number of dry packages (information for the delivery
driver)', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'TotalDryItems'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Total number of chiller packages (information for the delivery
driver)', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'TotalChillerItems'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Delivery run for this
shipment', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'DeliveryRun'
```

```
GO

EXEC sys.sp_addextendedproperty N'Description', 'Position in the delivery run for this
shipment', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'RunPosition'
GO

EXEC sys.sp_addextendedproperty N'Description', 'JSON-structured data returned from delivery devices for deliveries made
directly
by the organization', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'ReturnedDeliveryData'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Confirmed delivery date and time promoted from JSON delivery
data', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'ConfirmedDeliveryTime'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Confirmed receiver promoted from JSON delivery
data', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'COLUMN', N'ConfirmedReceivedBy'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'INDEX', N'FK_Sales_Invoices_AccountsPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'INDEX', N'FK_Sales_Invoices_BillToCustomerID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'INDEX', N'FK_Sales_Invoices_ContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'INDEX', N'FK_Sales_Invoices_CustomerID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'INDEX', N'FK_Sales_Invoices_DeliveryMethodID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'INDEX', N'FK_Sales_Invoices_OrderID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'INDEX', N'FK_Sales_Invoices_PackedByPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'INDEX', N'FK_Sales_Invoices_SalespersonPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Allows quick retrieval of invoices confirmed to have been delivered in a
given
time period', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'INDEX', N'IX_Sales_Invoices_ConfirmedDeliveryTime'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Ensures that if returned delivery data is present that it is valid
JSON', 'SCHEMA', N'Sales', 'TABLE', N'Invoices', 'CONSTRAINT', N'CK_Sales_Invoices_ReturnedDeliveryData_Must_Be_Valid_JSON'
GO
```

Depends On 7

- Sales
- Application.DeliveryMethods
- Application.People
- Sales.Customers
- Sales.Orders
- Sequences.InvoiceID
- WideWorldImporters

Used By 6

- Sales.CustomerTransactions
- Sales.InvoiceLines
- Warehouse.StockItemTransactions

Integration.GetSaleUpdates

Integration.GetTransactionUpdates

Website.InvoiceCustomerOrders

Sales.OrderLines

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	224337
Created	25-May-16 15:36:54
Last Modified	27-May-16 10:06:52

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	OrderLineID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[OrderLineID])	False	False	
3	OrderID	int	4	10	0	True				False	False	
3	StockItemID	int	4	10	0	True				False	False	
	Description	nvarchar	100	0	0	True				False	False	
	PackageTypeID	int	4	10	0	True				False	False	
	Quantity	int	4	10	0	True				False	False	
	UnitPrice	decimal	9	18	2	False				False	False	

	TaxRate	decimal	9	18	3	True				False	False	
	PickedQuantity	int	4	10	0	True				False	False	
	PickingCompletedWhen	datetime2	8	27	7	False				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	LastEditedWhen	datetime2	8	27	7	True			(sysdatetime())	False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	FK_Sales_OrderLines_OrderID	OrderID	False		
	FK_Sales_OrderLines_PackageTypeID	PackageTypeID	False		
	IX_Sales_OrderLines_AllocatedStockItems	StockItemID	False		
	IX_Sales_OrderLines_Perf_20160301_01	PickingCompletedWhen, OrderID, OrderLineID	False		
	IX_Sales_OrderLines_Perf_20160301_02	StockItemID, PickingCompletedWhen	False		
	NCCX_Sales_OrderLines	OrderID, StockItemID, Description, Quantity, UnitPrice, PickedQuantity	False		
	PK_Sales_OrderLines	OrderLineID	True		

Foreign Keys

Name	Columns	Description
FK_Sales_OrderLines_Application_People	PersonID	
FK_Sales_OrderLines_OrderID_Sales_Orders	OrderID	
FK_Sales_OrderLines_PackageTypeID_Warehouse_PackageTypes	PackageTypeID	
FK_Sales_OrderLines_StockItemID_Warehouse_StockItems	StockItemID	

Extended Properties

Name	Value
Description	Detail lines from customer orders

SQL Script

```
CREATE TABLE Sales.OrderLines (
  OrderLineID int NOT NULL CONSTRAINT DF_Sales_OrderLines_OrderLineID DEFAULT (NEXT VALUE FOR [Sequences].
  [OrderLineID]),
  OrderID int NOT NULL,
  StockItemID int NOT NULL,
  Description nvarchar(100) NOT NULL,
  PackageTypeID int NOT NULL,
  Quantity int NOT NULL,
  UnitPrice decimal(18, 2) NULL,
  TaxRate decimal(18, 3) NOT NULL,
  PickedQuantity int NOT NULL,
  PickingCompletedWhen datetime2 NULL,
  LastEditedBy int NOT NULL,
  LastEditedWhen datetime2 NOT NULL CONSTRAINT DF_Sales_OrderLines_LastEditedWhen DEFAULT (sysdatetime()),
  CONSTRAINT PK_Sales_OrderLines PRIMARY KEY CLUSTERED (OrderLineID)
)
ON [PRIMARY]
GO
```

```
CREATE INDEX FK_Sales_OrderLines_OrderID
  ON Sales.OrderLines (OrderID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_OrderLines_PackageTypeID
  ON Sales.OrderLines (PackageTypeID)
  ON [PRIMARY]
GO

CREATE INDEX IX_Sales_OrderLines_AllocatedStockItems
  ON Sales.OrderLines (StockItemID)
  INCLUDE (PickedQuantity)
  ON [PRIMARY]
GO

CREATE INDEX IX_Sales_OrderLines_Perf_20160301_01
  ON Sales.OrderLines (PickingCompletedWhen, OrderID, OrderLineID)
  INCLUDE (Quantity, StockItemID)
  ON [PRIMARY]
GO

CREATE INDEX IX_Sales_OrderLines_Perf_20160301_02
  ON Sales.OrderLines (StockItemID, PickingCompletedWhen)
  INCLUDE (OrderID, PickedQuantity)
  ON [PRIMARY]
GO

CREATE COLUMNSTORE INDEX NCCX_Sales_OrderLines
  ON Sales.OrderLines (OrderID, StockItemID, Description, Quantity, UnitPrice, PickedQuantity)
GO

ALTER TABLE Sales.OrderLines
  ADD CONSTRAINT FK_Sales_OrderLines_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.People (
  PersonID)
GO

ALTER TABLE Sales.OrderLines
  ADD CONSTRAINT FK_Sales_OrderLines_OrderID_Sales_Orders FOREIGN KEY (OrderID) REFERENCES Sales.Orders (OrderID)
GO

ALTER TABLE Sales.OrderLines
  ADD CONSTRAINT FK_Sales_OrderLines_PackageTypeID_Warehouse_PackageTypes FOREIGN KEY (PackageTypeID) REFERENCES
  Warehouse.PackageTypes (PackageTypeID)
GO

ALTER TABLE Sales.OrderLines
  ADD CONSTRAINT FK_Sales_OrderLines_StockItemID_Warehouse_StockItems FOREIGN KEY (StockItemID) REFERENCES
  Warehouse.StockItems (StockItemID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Detail lines from customer
orders', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a line on an Order within the
database', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'COLUMN', N'OrderLineID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Order that this line is associated
with', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'COLUMN', N'OrderID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Stock item for this order line (FK not indexed as separate index
exists)', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'COLUMN', N'StockItemID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Description of the item supplied (Usually the stock item name but can
be
overridden)', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'COLUMN', N'Description'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Type of package to be
supplied', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'COLUMN', N'PackageTypeID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Quantity to be
supplied', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'COLUMN', N'Quantity'
```

```
GO

EXEC sys.sp_addextendedproperty N'Description', 'Unit price to be
charged', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'COLUMN', N'UnitPrice'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Tax rate to be
applied', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'COLUMN', N'TaxRate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Quantity picked from
stock', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'COLUMN', N'PickedQuantity'
GO

EXEC sys.sp_addextendedproperty N'Description', 'When was picking of this line
completed?', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'COLUMN', N'PickingCompletedWhen'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'INDEX', N'FK_Sales_OrderLines_OrderID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'INDEX', N'FK_Sales_OrderLines_PackageTypeID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Allows quick summation of stock item quantites already allocated to
uninvoiced
orders', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'INDEX', N'IX_Sales_OrderLines_AllocatedStockItems'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Improves performance of order picking and
invoicing', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'INDEX', N'IX_Sales_OrderLines_Perf_20160301_01'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Improves performance of order picking and
invoicing', 'SCHEMA', N'Sales', 'TABLE', N'OrderLines', 'INDEX', N'IX_Sales_OrderLines_Perf_20160301_02'
GO
```

Depends On 7

- Sales
- Application.People
- Sales.Orders
- Warehouse.PackageTypes
- Warehouse.StockItems
- Sequences.OrderLineID
- WideWorldImporters

Used By 3

- Integration.GetOrderUpdates
- Website.InsertCustomerOrders
- Website.InvoiceCustomerOrders

Sales.Orders

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	69487
Created	25-May-16 15:36:54
Last Modified	25-May-16 15:36:54

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	OrderID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[OrderID])	False	False	
	CustomerID	int	4	10	0	True				False	False	
	SalespersonPersonID	int	4	10	0	True				False	False	
	PickedByPersonID	int	4	10	0	False				False	False	
	ContactPersonID	int	4	10	0	True				False	False	
	BackorderOrderID	int	4	10	0	False				False	False	
	OrderDate	date	3	10	0	True				False	False	
	ExpectedDeliveryDate	date	3	10	0	True				False	False	
	CustomerPurchaseOrderNumber	nvarchar	20	0	0	False				False	False	

	IsUndersupplyBackordered	bit	1	1	0	True				False	False	
	Comments	nvarchar		0	0	False				False	False	
	DeliveryInstructions	nvarchar		0	0	False				False	False	
	InternalComments	nvarchar		0	0	False				False	False	
	PickingCompletedWhen	datetime2	8	27	7	False				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	LastEditedWhen	datetime2	8	27	7	True			(sysdatetime())	False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	FK_Sales_Orders_ContactPersonID	ContactPersonID	False		
	FK_Sales_Orders_CustomerID	CustomerID	False		
	FK_Sales_Orders_PickedByPersonID	PickedByPersonID	False		
	FK_Sales_Orders_SalespersonPersonID	SalespersonPersonID	False		
	PK_Sales_Orders	OrderID	True		

Foreign Keys

Name	Columns	Description
FK_Sales_Orders_Application_People	PersonID	
FK_Sales_Orders_BackorderOrderID_Sales_Orders	OrderID	
FK_Sales_Orders_ContactPersonID_Application_People	PersonID	
FK_Sales_Orders_CustomerID_Sales_Customers	CustomerID	
FK_Sales_Orders_PickedByPersonID_Application_People	PersonID	
FK_Sales_Orders_SalespersonPersonID_Application_People	PersonID	

Extended Properties

Name	Value
Description	Detail of customer orders

SQL Script

```
CREATE TABLE Sales.Orders (
  OrderID int NOT NULL CONSTRAINT DF_Sales_Orders_OrderID DEFAULT (NEXT VALUE FOR [Sequences].[OrderID]),
  CustomerID int NOT NULL,
  SalespersonPersonID int NOT NULL,
  PickedByPersonID int NULL,
  ContactPersonID int NOT NULL,
  BackorderOrderID int NULL,
  OrderDate date NOT NULL,
  ExpectedDeliveryDate date NOT NULL,
  CustomerPurchaseOrderNumber nvarchar(20) NULL,
  IsUndersupplyBackordered bit NOT NULL,
  Comments nvarchar(max) NULL,
  DeliveryInstructions nvarchar(max) NULL,
  InternalComments nvarchar(max) NULL,
  PickingCompletedWhen datetime2 NULL,
  LastEditedBy int NOT NULL,
  LastEditedWhen datetime2 NOT NULL CONSTRAINT DF_Sales_Orders_LastEditedWhen DEFAULT (sysdatetime()),
  CONSTRAINT PK_Sales_Orders PRIMARY KEY CLUSTERED (OrderID)
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
```

```
GO

CREATE INDEX FK_Sales_Orders_ContactPersonID
  ON Sales.Orders (ContactPersonID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Orders_CustomerID
  ON Sales.Orders (CustomerID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Orders_PickedByPersonID
  ON Sales.Orders (PickedByPersonID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_Orders_SalespersonPersonID
  ON Sales.Orders (SalespersonPersonID)
  ON [PRIMARY]
GO

ALTER TABLE Sales.Orders
  ADD CONSTRAINT FK_Sales_Orders_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.People (
  PersonID)
GO

ALTER TABLE Sales.Orders
  ADD CONSTRAINT FK_Sales_Orders_BackorderOrderID_Sales_Orders FOREIGN KEY (BackorderOrderID) REFERENCES Sales.
  Orders (OrderID)
GO

ALTER TABLE Sales.Orders
  ADD CONSTRAINT FK_Sales_Orders_ContactPersonID_Application_People FOREIGN KEY (ContactPersonID) REFERENCES
  Application.People (PersonID)
GO

ALTER TABLE Sales.Orders
  ADD CONSTRAINT FK_Sales_Orders_CustomerID_Sales_Customers FOREIGN KEY (CustomerID) REFERENCES Sales.Customers (
  CustomerID)
GO

ALTER TABLE Sales.Orders
  ADD CONSTRAINT FK_Sales_Orders_PickedByPersonID_Application_People FOREIGN KEY (PickedByPersonID) REFERENCES
  Application.People (PersonID)
GO

ALTER TABLE Sales.Orders
  ADD CONSTRAINT FK_Sales_Orders_SalespersonPersonID_Application_People FOREIGN KEY (SalespersonPersonID)
  REFERENCES Application.People (PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Detail of customer orders', 'SCHEMA', N'Sales', 'TABLE', N'Orders'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to an order within the
database', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'OrderID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Customer for this
order', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'CustomerID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Salesperson for this
order', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'SalespersonPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Person who picked this
shipment', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'PickedByPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Customer contact for this
order', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'ContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'If this order is a backorder, this column holds the original order
number', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'BackorderOrderID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Date that this order was
raised', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'OrderDate'
```

```
GO

EXEC sys.sp_addextendedproperty N'Description', 'Expected delivery
date', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'ExpectedDeliveryDate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Purchase Order Number received from
customer', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'CustomerPurchaseOrderNumber'
GO

EXEC sys.sp_addextendedproperty N'Description', 'If items cannot be supplied are they
backordered?', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'IsUndersupplyBackordered'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Any comments related to this order (sent to
customer)', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'Comments'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Any comments related to order delivery (sent to
customer)', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'DeliveryInstructions'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Any internal comments related to this order (not sent to the
customer)', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'InternalComments'
GO

EXEC sys.sp_addextendedproperty N'Description', 'When was picking of the entire order
completed?', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'COLUMN', N'PickingCompletedWhen'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'INDEX', N'FK_Sales_Orders_ContactPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'INDEX', N'FK_Sales_Orders_CustomerID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'INDEX', N'FK_Sales_Orders_PickedByPersonID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'Orders', 'INDEX', N'FK_Sales_Orders_SalespersonPersonID'
GO
```

Depends On 6

- Sales
- Application.People
- Sales.Customers
- Sales.Orders
- Sequences.OrderID
- WideWorldImporters

Used By 7

- Sales.Invoices
- Sales.OrderLines
- Sales.Orders
- DataLoadSimulation.PopulateDataToCurrentDate
- Integration.GetOrderUpdates
- Website.InsertCustomerOrders
- Website.InvoiceCustomerOrders

Sales.SpecialDeals

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	2
Created	25-May-16 15:36:54
Last Modified	25-May-16 15:36:54

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	SpecialDealID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[SpecialDealID])	False	False	
	StockItemID	int	4	10	0	False				False	False	
	CustomerID	int	4	10	0	False				False	False	
	BuyingGroupID	int	4	10	0	False				False	False	
	CustomerCategoryID	int	4	10	0	False				False	False	
	StockGroupID	int	4	10	0	False				False	False	

	DealDescription	nvarchar	30	0	0	True				False	False	
	StartDate	date	3	10	0	True				False	False	
	EndDate	date	3	10	0	True				False	False	
	DiscountAmount	decimal	9	18	2	False				False	False	
	DiscountPercentage	decimal	9	18	3	False				False	False	
	UnitPrice	decimal	9	18	2	False				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	LastEditedWhen	datetime2	8	27	7	True			(sysdatetime())	False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	FK_Sales_SpecialDeals_BuyingGroupID	BuyingGroupID	False		
	FK_Sales_SpecialDeals_CustomerCategoryID	CustomerCategoryID	False		
	FK_Sales_SpecialDeals_CustomerID	CustomerID	False		
	FK_Sales_SpecialDeals_StockGroupID	StockGroupID	False		
	FK_Sales_SpecialDeals_StockItemID	StockItemID	False		
	PK_Sales_SpecialDeals	SpecialDealID	True		

Check Constraints

Name	Columns	Condition	Description
CK_Sales_SpecialDeals_Exactly_One_NOT_NULL_Pricing_Option_Is_Required	DiscountAmount, DiscountPercentage, UnitPrice	(((case when [DiscountAmount] IS NULL then (0) else (1) end+case when [DiscountPercentage] IS NULL then (0) else (1) end)+case when [UnitPrice] IS NULL then (0) else (1) end)=(1))	
CK_Sales_SpecialDeals_Unit_Price_Deal_Requires_Special_StockItem	StockItemID, UnitPrice	([StockItemID] IS NOT NULL AND [UnitPrice] IS NOT NULL OR [UnitPrice] IS NULL)	

Foreign Keys

Name	Columns	Description
FK_Sales_SpecialDeals_Application_People	PersonID	
FK_Sales_SpecialDeals_BuyingGroupID_Sales_BuyingGroups	BuyingGroupID	
FK_Sales_SpecialDeals_CustomerCategoryID_Sales_CustomerCategories	CustomerCategoryID	
FK_Sales_SpecialDeals_CustomerID_Sales_Customers	CustomerID	
FK_Sales_SpecialDeals_StockGroupID_Warehouse_StockGroups	StockGroupID	
FK_Sales_SpecialDeals_StockItemID_Warehouse_StockItems	StockItemID	

Extended Properties

Name	Value
Description	Special pricing (can include fixed prices, discount $ or discount %)

SQL Script

```
CREATE TABLE Sales.SpecialDeals (
  SpecialDealID int NOT NULL CONSTRAINT DF_Sales_SpecialDeals_SpecialDealID DEFAULT (NEXT VALUE FOR [Sequences].
  [SpecialDealID]),
  StockItemID int NULL,
  CustomerID int NULL,
  BuyingGroupID int NULL,
  CustomerCategoryID int NULL,
  StockGroupID int NULL,
  DealDescription nvarchar(30) NOT NULL,
  StartDate date NOT NULL,
  EndDate date NOT NULL,
  DiscountAmount decimal(18, 2) NULL,
  DiscountPercentage decimal(18, 3) NULL,
  UnitPrice decimal(18, 2) NULL,
  LastEditedBy int NOT NULL,
  LastEditedWhen datetime2 NOT NULL CONSTRAINT DF_Sales_SpecialDeals_LastEditedWhen DEFAULT (sysdatetime()),
  CONSTRAINT PK_Sales_SpecialDeals PRIMARY KEY CLUSTERED (SpecialDealID),
  CONSTRAINT CK_Sales_SpecialDeals_Exactly_One_NOT_NULL_Pricing_Option_Is_Required CHECK (((case when
  [DiscountAmount] IS NULL then (0) else (1) end+case when [DiscountPercentage] IS NULL then (0) else (1) end)+case
  when [UnitPrice] IS NULL then (0) else (1) end)=(1)),
  CONSTRAINT CK_Sales_SpecialDeals_Unit_Price_Deal_Requires_Special_StockItem CHECK ([StockItemID] IS NOT NULL AND
  [UnitPrice] IS NOT NULL OR [UnitPrice] IS NULL)
)
ON [PRIMARY]
GO

CREATE INDEX FK_Sales_SpecialDeals_BuyingGroupID
  ON Sales.SpecialDeals (BuyingGroupID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_SpecialDeals_CustomerCategoryID
  ON Sales.SpecialDeals (CustomerCategoryID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_SpecialDeals_CustomerID
  ON Sales.SpecialDeals (CustomerID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_SpecialDeals_StockGroupID
  ON Sales.SpecialDeals (StockGroupID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Sales_SpecialDeals_StockItemID
  ON Sales.SpecialDeals (StockItemID)
  ON [PRIMARY]
GO

ALTER TABLE Sales.SpecialDeals
  ADD CONSTRAINT FK_Sales_SpecialDeals_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.People
  (PersonID)
GO

ALTER TABLE Sales.SpecialDeals
  ADD CONSTRAINT FK_Sales_SpecialDeals_BuyingGroupID_Sales_BuyingGroups FOREIGN KEY (BuyingGroupID) REFERENCES
  Sales.BuyingGroups (BuyingGroupID)
GO

ALTER TABLE Sales.SpecialDeals
  ADD CONSTRAINT FK_Sales_SpecialDeals_CustomerCategoryID_Sales_CustomerCategories FOREIGN KEY (CustomerCategoryID)
  REFERENCES Sales.CustomerCategories (CustomerCategoryID)
GO

ALTER TABLE Sales.SpecialDeals
  ADD CONSTRAINT FK_Sales_SpecialDeals_CustomerID_Sales_Customers FOREIGN KEY (CustomerID) REFERENCES Sales.
  Customers (CustomerID)
```

```
GO

ALTER TABLE Sales.SpecialDeals
  ADD CONSTRAINT FK_Sales_SpecialDeals_StockGroupID_Warehouse_StockGroups FOREIGN KEY (StockGroupID) REFERENCES
  Warehouse.StockGroups (StockGroupID)
GO

ALTER TABLE Sales.SpecialDeals
  ADD CONSTRAINT FK_Sales_SpecialDeals_StockItemID_Warehouse_StockItems FOREIGN KEY (StockItemID) REFERENCES
  Warehouse.StockItems (StockItemID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Special pricing (can include fixed prices, discount $ or discount
%)', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID (sequence based) for a special
deal', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'COLUMN', N'SpecialDealID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Stock item that the deal applies to (if NULL, then only discounts are
permitted
not unit prices)', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'COLUMN', N'StockItemID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of the customer that the special pricing applies to (if NULL then
all
customers)', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'COLUMN', N'CustomerID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of the buying group that the special pricing applies to
(optional)', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'COLUMN', N'BuyingGroupID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of the customer category that the special pricing applies to
(optional)', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'COLUMN', N'CustomerCategoryID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of the stock group that the special pricing applies to
(optional)', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'COLUMN', N'StockGroupID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Description of the special
deal', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'COLUMN', N'DealDescription'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Date that the special pricing starts
from', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'COLUMN', N'StartDate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Date that the special pricing ends
on', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'COLUMN', N'EndDate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Discount per unit to be applied to sale price
(optional)', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'COLUMN', N'DiscountAmount'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Discount percentage per unit to be applied to sale price
(optional)', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'COLUMN', N'DiscountPercentage'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Special price per unit to be applied instead of sale price
(optional)', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'COLUMN', N'UnitPrice'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'INDEX', N'FK_Sales_SpecialDeals_BuyingGroupID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'INDEX', N'FK_Sales_SpecialDeals_CustomerCategoryID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'INDEX', N'FK_Sales_SpecialDeals_CustomerID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
```

```
key', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'INDEX', N'FK_Sales_SpecialDeals_StockGroupID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'INDEX', N'FK_Sales_SpecialDeals_StockItemID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Ensures that each special price row contains one and only one of
DiscountAmount,
DiscountPercentage, and
UnitPrice', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'CONSTRAINT', N'CK_Sales_SpecialDeals_Exactly_One_NOT_NULL_Pri
cing_Option_Is_
Required'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Ensures that if a specific price is allocated that it applies to a
specific
stock
item', 'SCHEMA', N'Sales', 'TABLE', N'SpecialDeals', 'CONSTRAINT', N'CK_Sales_SpecialDeals_Unit_Price_Deal_Requires_Spec
ial_StockItem'
GO
```

Depends On 9

- Sales
- Application.People
- Sales.BuyingGroups
- Sales.CustomerCategories
- Sales.Customers
- Warehouse.StockGroups
- Warehouse.StockItems
- Sequences.SpecialDealID
- WideWorldImporters

Used By 1

- Website.CalculateCustomerPrice

Warehouse.ColdRoomTemperatures

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	True
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	0
Created	26-May-16 09:02:17
Last Modified	26-May-16 09:02:18

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
🔑	ColdRoomTemperatureID	bigint	8	19	0	True	1 - 1			False	False	
	ColdRoomSensorNumber	int	4	10	0	True				False	False	
	RecordedWhen	datetime2	8	27	7	True				False	False	
	Temperature	decimal	9	10	2	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
🔑	PK__ColdRoom__F4031CE64A340F2F	ColdRoomTemperatureID	True		

SQL Script

```
CREATE TABLE Warehouse.ColdRoomTemperatures (
  ColdRoomTemperatureID bigint IDENTITY,
  ColdRoomSensorNumber int NOT NULL,
```

```
  RecordedWhen datetime2 NOT NULL,
  Temperature decimal(10, 2) NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  PRIMARY KEY NONCLUSTERED (ColdRoomTemperatureID)
)
WITH (MEMORY_OPTIMIZED = ON)
GO
```

Depends On 1

- Warehouse

Used By 3

- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Website.RecordColdRoomTemperatures

Warehouse.ColdRoomTemperatures_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	2961492
Created	25-May-16 15:36:53
Last Modified	26-May-16 09:02:25

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	ColdRoomTemperatureID	bigint	8	19	0	True				False	False	
	ColdRoomSensorNumber	int	4	10	0	True				False	False	
	RecordedWhen	datetime2	8	27	7	True				False	False	
	Temperature	decimal	9	10	2	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_ColdRoomTemperatures_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Warehouse.ColdRoomTemperatures_Archive (
  ColdRoomTemperatureID bigint NOT NULL,
  ColdRoomSensorNumber int NOT NULL,
  RecordedWhen datetime2 NOT NULL,
  Temperature decimal(10, 2) NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_ColdRoomTemperatures_Archive
  ON Warehouse.ColdRoomTemperatures_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 1

Warehouse

Used By 1

DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad

Warehouse.Colors

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	36
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:39

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	ColorID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[ColorID])	False	False	
	ColorName	nvarchar	20	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	PK_Warehouse_Colors	ColorID	True		

	UQ_Warehouse_Colors_ColorName	ColorName	True		

Foreign Keys

Name	Columns	Description
FK_Warehouse_Colors_Application_People	PersonID	

Extended Properties

Name	Value
Description	Stock items can (optionally) have colors

SQL Script

```
CREATE TABLE Warehouse.Colors (
  ColorID int NOT NULL CONSTRAINT DF_Warehouse_Colors_ColorID DEFAULT (NEXT VALUE FOR [Sequences].[ColorID]),
  ColorName nvarchar(20) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Warehouse_Colors PRIMARY KEY CLUSTERED (ColorID),
  CONSTRAINT UQ_Warehouse_Colors_ColorName UNIQUE (ColorName)
)
ON [PRIMARY]
GO

ALTER TABLE Warehouse.Colors
  ADD CONSTRAINT FK_Warehouse_Colors_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.People (
  PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Stock items can (optionally) have
colors', 'SCHEMA', N'Warehouse', 'TABLE', N'Colors'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a color within the
database', 'SCHEMA', N'Warehouse', 'TABLE', N'Colors', 'COLUMN', N'ColorID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Full name of a color that can be used to describe stock
items', 'SCHEMA', N'Warehouse', 'TABLE', N'Colors', 'COLUMN', N'ColorName'
GO
```

Depends On 4

- Warehouse
- Application.People
- Sequences.ColorID
- WideWorldImporters

Used By 4

- Warehouse.StockItems
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetStockItemUpdates

Warehouse.Colors_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	1
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:39

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	ColorID	int	4	10	0	True				False	False	
	ColorName	nvarchar	20	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_Colors_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Warehouse.Colors_Archive (
  ColorID int NOT NULL,
```

```
  ColorName nvarchar(20) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_Colors_Archive
  ON Warehouse.Colors_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Warehouse
- WideWorldImporters

Used By 1

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad

Warehouse.PackageTypes

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	14
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:39

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	PackageTypeID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[PackageTypeID])	False	False	
	PackageTypeName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	PK_Warehouse_PackageTypes	PackageTypeID	True		

	UQ_Warehouse_PackageTypes_PackageTypeName	PackageTypeName	True		

Foreign Keys

Name	Columns	Description
FK_Warehouse_PackageTypes_Application_People	PersonID	

Extended Properties

Name	Value
Description	Ways that stock items can be packaged (ie: each, box, carton, pallet, kg, etc.

SQL Script

```
CREATE TABLE Warehouse.PackageTypes (
  PackageTypeID int NOT NULL CONSTRAINT DF_Warehouse_PackageTypes_PackageTypeID DEFAULT (NEXT VALUE FOR [Sequences]
  .[PackageTypeID]),
  PackageTypeName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Warehouse_PackageTypes PRIMARY KEY CLUSTERED (PackageTypeID),
  CONSTRAINT UQ_Warehouse_PackageTypes_PackageTypeName UNIQUE (PackageTypeName)
)
ON [PRIMARY]
GO

ALTER TABLE Warehouse.PackageTypes
  ADD CONSTRAINT FK_Warehouse_PackageTypes_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.
  People (PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Ways that stock items can be packaged (ie: each, box, carton, pallet,
kg, etc.', 'SCHEMA', N'Warehouse', 'TABLE', N'PackageTypes'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a package type within the
database', 'SCHEMA', N'Warehouse', 'TABLE', N'PackageTypes', 'COLUMN', N'PackageTypeID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Full name of package types that stock items can be purchased in or sold
in', 'SCHEMA', N'Warehouse', 'TABLE', N'PackageTypes', 'COLUMN', N'PackageTypeName'
GO
```

Depends On 4

- Warehouse
- Application.People
- Sequences.PackageTypeID
- WideWorldImporters

Used By 10

- Purchasing.PurchaseOrderLines
- Sales.InvoiceLines
- Sales.OrderLines
- Warehouse.StockItems
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetOrderUpdates
- Integration.GetPurchaseUpdates
- Integration.GetSaleUpdates
- Integration.GetStockItemUpdates

Warehouse.PackageTypes_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	0
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:39

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	PackageTypeID	int	4	10	0	True				False	False	
	PackageTypeName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_PackageTypes_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Warehouse.PackageTypes_Archive (
  PackageTypeID int NOT NULL,
```

```
  PackageTypeName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_PackageTypes_Archive
  ON Warehouse.PackageTypes_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Warehouse
- WideWorldImporters

Used By 1

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad

Warehouse.StockGroups

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	10
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:39

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	StockGroupID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[StockGroupID])	False	False	
	StockGroupName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	PK_Warehouse_StockGroups	StockGroupID	True		

	UQ_Warehouse_StockGroups_StockGroupName	StockGroupName	True		

Foreign Keys

Name	Columns	Description
FK_Warehouse_StockGroups_Application_People	PersonID	

Extended Properties

Name	Value
Description	Groups for categorizing stock items (ie: novelties, toys, edible novelties, etc.)

SQL Script

```
CREATE TABLE Warehouse.StockGroups (
  StockGroupID int NOT NULL CONSTRAINT DF_Warehouse_StockGroups_StockGroupID DEFAULT (NEXT VALUE FOR [Sequences].
  [StockGroupID]),
  StockGroupName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Warehouse_StockGroups PRIMARY KEY CLUSTERED (StockGroupID),
  CONSTRAINT UQ_Warehouse_StockGroups_StockGroupName UNIQUE (StockGroupName)
)
ON [PRIMARY]
GO

ALTER TABLE Warehouse.StockGroups
  ADD CONSTRAINT FK_Warehouse_StockGroups_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.
  People (PersonID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Groups for categorizing stock items (ie: novelties, toys, edible
novelties, etc.
)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockGroups'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a stock group within the
database', 'SCHEMA', N'Warehouse', 'TABLE', N'StockGroups', 'COLUMN', N'StockGroupID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Full name of groups used to categorize stock
items', 'SCHEMA', N'Warehouse', 'TABLE', N'StockGroups', 'COLUMN', N'StockGroupName'
GO
```

Depends On 4

- Warehouse
- Application.People
- Sequences.StockGroupID
- WideWorldImporters

Used By 4

- Sales.SpecialDeals
- Warehouse.StockItemStockGroups
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad

Warehouse.StockGroups_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	1
Created	25-May-16 15:36:53
Last Modified	25-May-16 16:42:39

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	StockGroupID	int	4	10	0	True				False	False	
	StockGroupName	nvarchar	50	0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_StockGroups_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Warehouse.StockGroups_Archive (
  StockGroupID int NOT NULL,
```

```
  StockGroupName nvarchar(50) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_StockGroups_Archive
  ON Warehouse.StockGroups_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

- Warehouse
- WideWorldImporters

Used By 1

- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad

Warehouse.StockItemHoldings

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	227
Created	25-May-16 15:36:54
Last Modified	25-May-16 15:36:54

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	StockItemID	int	4	10	0	True				False	False	
	QuantityOnHand	int	4	10	0	True				False	False	
	BinLocation	nvarchar	20	0	0	True				False	False	
	LastStocktakeQuantity	int	4	10	0	True				False	False	
	LastCostPrice	decimal	9	18	2	True				False	False	
	ReorderLevel	int	4	10	0	True				False	False	
	TargetStockLevel	int	4	10	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	LastEditedWhen	datetime2	8	27	7	True			(sysdatetime())	False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	PK_Warehouse_StockItemHoldings	StockItemID	True		

Foreign Keys

Name	Columns	Description
FK_Warehouse_StockItemHoldings_Application_People	PersonID	
PKFK_Warehouse_StockItemHoldings_StockItemID_Warehouse_StockItems	StockItemID	

Extended Properties

Name	Value
Description	Non-temporal attributes for stock items

SQL Script

```
CREATE TABLE Warehouse.StockItemHoldings (
  StockItemID int NOT NULL,
  QuantityOnHand int NOT NULL,
  BinLocation nvarchar(20) NOT NULL,
  LastStocktakeQuantity int NOT NULL,
  LastCostPrice decimal(18, 2) NOT NULL,
  ReorderLevel int NOT NULL,
  TargetStockLevel int NOT NULL,
  LastEditedBy int NOT NULL,
  LastEditedWhen datetime2 NOT NULL CONSTRAINT DF_Warehouse_StockItemHoldings_LastEditedWhen DEFAULT (sysdatetime()
  ),
  CONSTRAINT PK_Warehouse_StockItemHoldings PRIMARY KEY CLUSTERED (StockItemID)
)
ON [PRIMARY]
GO

ALTER TABLE Warehouse.StockItemHoldings
  ADD CONSTRAINT FK_Warehouse_StockItemHoldings_Application_People FOREIGN KEY (LastEditedBy) REFERENCES
  Application.People (PersonID)
GO

ALTER TABLE Warehouse.StockItemHoldings
  ADD CONSTRAINT PKFK_Warehouse_StockItemHoldings_StockItemID_Warehouse_StockItems FOREIGN KEY (StockItemID)
  REFERENCES Warehouse.StockItems (StockItemID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Non-temporal attributes for stock
items', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemHoldings'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of the stock item that this holding relates to (this table holds
non-temporal
columns for stock)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemHoldings', 'COLUMN', N'StockItemID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Quantity currently on hand (if
tracked)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemHoldings', 'COLUMN', N'QuantityOnHand'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Bin location (ie location of this stock item within the
depot)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemHoldings', 'COLUMN', N'BinLocation'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Quantity at last stocktake (if
tracked)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemHoldings', 'COLUMN', N'LastStocktakeQuantity'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Unit cost price the last time this stock item was
purchased', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemHoldings', 'COLUMN', N'LastCostPrice'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Quantity below which reordering should take
```

```
place', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemHoldings', 'COLUMN', N'ReorderLevel'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Typical quantity
ordered', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemHoldings', 'COLUMN', N'TargetStockLevel'
GO
```

Depends On 4

- Warehouse
- Application.People
- Warehouse.StockItems
- WideWorldImporters

Used By 2

- Integration.GetStockHoldingUpdates
- Website.InvoiceCustomerOrders

Warehouse.StockItems

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	227
Created	25-May-16 15:36:54
Last Modified	25-May-16 16:42:39

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	StockItemID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[StockItemID])	False	False	
	StockItemName	nvarchar	100	0	0	True				False	False	
	SupplierID	int	4	10	0	True				False	False	
	ColorID	int	4	10	0	False				False	False	
	UnitPackageID	int	4	10	0	True				False	False	
	OuterPackageID	int	4	10	0	True				False	False	
	Brand	nvarchar	50	0	0	False				False	False	

	Size	nvarchar	20	0	0	False				False	False	
	LeadTimeDays	int	4	10	0	True				False	False	
	QuantityPerOuter	int	4	10	0	True				False	False	
	IsChillerStock	bit	1	1	0	True				False	False	
	Barcode	nvarchar	50	0	0	False				False	False	
	TaxRate	decimal	9	18	3	True				False	False	
	UnitPrice	decimal	9	18	2	True				False	False	
	RecommendedRetailPrice	decimal	9	18	2	False				False	False	
	TypicalWeightPerUnit	decimal	9	18	3	True				False	False	
	MarketingComments	nvarchar		0	0	False				False	False	
	InternalComments	nvarchar		0	0	False				False	False	
	Photo	varbinary		0	0	False				False	False	
	CustomFields	nvarchar		0	0	False				False	False	
	Tags	nvarchar		0	0	False				True	False	
	SearchDetails	nvarchar		0	0	True				True	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Computed Columns

Name	Definition
Tags	(json_query([CustomFields],N'$.Tags'))
SearchDetails	(concat([StockItemName],N' ',[MarketingComments]))

Indexes

Key	Name	Columns	Unique	Type	Description
	FK_Warehouse_StockItems_ColorID	ColorID	False		
	FK_Warehouse_StockItems_OuterPackageID	OuterPackageID	False		
	FK_Warehouse_StockItems_SupplierID	SupplierID	False		
	FK_Warehouse_StockItems_UnitPackageID	UnitPackageID	False		
	PK_Warehouse_StockItems	StockItemID	True		
	UQ_Warehouse_StockItems_StockItemName	StockItemName	True		

Foreign Keys

Name	Columns	Description
FK_Warehouse_StockItems_Application_People	PersonID	
FK_Warehouse_StockItems_ColorID_Warehouse_Colors	ColorID	
FK_Warehouse_StockItems_OuterPackageID_Warehouse_PackageTypes	PackageTypeID	

FK_Warehouse_StockItems_SupplierID_Purchasing_Suppliers	SupplierID	
FK_Warehouse_StockItems_UnitPackageID_Warehouse_PackageTypes	PackageTypeID	

Extended Properties

Name	Value
Description	Main entity table for stock items

SQL Script

```
CREATE TABLE Warehouse.StockItems (
  StockItemID int NOT NULL CONSTRAINT DF_Warehouse_StockItems_StockItemID DEFAULT (NEXT VALUE FOR [Sequences].
  [StockItemID]),
  StockItemName nvarchar(100) NOT NULL,
  SupplierID int NOT NULL,
  ColorID int NULL,
  UnitPackageID int NOT NULL,
  OuterPackageID int NOT NULL,
  Brand nvarchar(50) NULL,
  Size nvarchar(20) NULL,
  LeadTimeDays int NOT NULL,
  QuantityPerOuter int NOT NULL,
  IsChillerStock bit NOT NULL,
  Barcode nvarchar(50) NULL,
  TaxRate decimal(18, 3) NOT NULL,
  UnitPrice decimal(18, 2) NOT NULL,
  RecommendedRetailPrice decimal(18, 2) NULL,
  TypicalWeightPerUnit decimal(18, 3) NOT NULL,
  MarketingComments nvarchar(max) NULL,
  InternalComments nvarchar(max) NULL,
  Photo varbinary(max) NULL,
  CustomFields nvarchar(max) NULL,
  Tags AS (json_query([CustomFields],N'$.Tags')),
  SearchDetails AS (concat([StockItemName],N' ',[MarketingComments])),
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
  ValidTo datetime2 GENERATED ALWAYS AS ROW END NOT NULL,
  CONSTRAINT PK_Warehouse_StockItems PRIMARY KEY CLUSTERED (StockItemID),
  CONSTRAINT UQ_Warehouse_StockItems_StockItemName UNIQUE (StockItemName)
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE INDEX FK_Warehouse_StockItems_ColorID
  ON Warehouse.StockItems (ColorID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Warehouse_StockItems_OuterPackageID
  ON Warehouse.StockItems (OuterPackageID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Warehouse_StockItems_SupplierID
  ON Warehouse.StockItems (SupplierID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Warehouse_StockItems_UnitPackageID
  ON Warehouse.StockItems (UnitPackageID)
  ON [PRIMARY]
GO

ALTER TABLE Warehouse.StockItems
  ADD CONSTRAINT FK_Warehouse_StockItems_Application_People FOREIGN KEY (LastEditedBy) REFERENCES Application.
  People (PersonID)
GO

ALTER TABLE Warehouse.StockItems
  ADD CONSTRAINT FK_Warehouse_StockItems_ColorID_Warehouse_Colors FOREIGN KEY (ColorID) REFERENCES Warehouse.Colors
  (ColorID)
GO
```

```
ALTER TABLE Warehouse.StockItems
  ADD CONSTRAINT FK_Warehouse_StockItems_OuterPackageID_Warehouse_PackageTypes FOREIGN KEY (OuterPackageID)
  REFERENCES Warehouse.PackageTypes (PackageTypeID)
GO

ALTER TABLE Warehouse.StockItems
  ADD CONSTRAINT FK_Warehouse_StockItems_SupplierID_Purchasing_Suppliers FOREIGN KEY (SupplierID) REFERENCES
  Purchasing.Suppliers (SupplierID)
GO

ALTER TABLE Warehouse.StockItems
  ADD CONSTRAINT FK_Warehouse_StockItems_UnitPackageID_Warehouse_PackageTypes FOREIGN KEY (UnitPackageID)
  REFERENCES Warehouse.PackageTypes (PackageTypeID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Main entity table for stock
items', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used for reference to a stock item within the
database', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'StockItemID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Full name of a stock item (but not a full
description)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'StockItemName'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Usual supplier for this stock
item', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'SupplierID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Color (optional) for this stock
item', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'ColorID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Usual package for selling units of this stock
item', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'UnitPackageID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Usual package for selling outers of this stock item (ie cartons, boxes,
etc.)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'OuterPackageID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Brand for the stock item (if the item is
branded)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'Brand'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Size of this item (eg:
100mm)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'Size'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Number of days typically taken from order to receipt of this stock
item', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'LeadTimeDays'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Quantity of the stock item in an outer
package', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'QuantityPerOuter'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Does this stock item need to be in a
chiller?', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'IsChillerStock'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Barcode for this stock
item', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'Barcode'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Tax rate to be
applied', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'TaxRate'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Selling price (ex-tax) for one unit of this
product', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'UnitPrice'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Recommended retail price for this stock
item', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'RecommendedRetailPrice'
GO
```

```
EXEC sys.sp_addextendedproperty N'Description', 'Typical weight for one unit of this product
(packaged)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'TypicalWeightPerUnit'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Marketing comments for this stock item (shared outside the
organization)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'MarketingComments'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Internal comments (not exposed outside
organization)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'InternalComments'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Photo of the
product', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'Photo'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Custom fields added by system
users', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'CustomFields'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Advertising tags associated with this stock item (JSON array retrieved
from
CustomFields)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'Tags'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Combination of columns used by full text
search', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'COLUMN', N'SearchDetails'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'INDEX', N'FK_Warehouse_StockItems_ColorID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'INDEX', N'FK_Warehouse_StockItems_OuterPackageID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'INDEX', N'FK_Warehouse_StockItems_SupplierID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItems', 'INDEX', N'FK_Warehouse_StockItems_UnitPackageID'
GO
```

Depends On 7

- Warehouse
- Application.People
- Purchasing.Suppliers
- Warehouse.Colors
- Warehouse.PackageTypes
- Sequences.StockItemID
- WideWorldImporters

Used By 17

- Purchasing.PurchaseOrderLines
- Sales.InvoiceLines
- Sales.OrderLines
- Sales.SpecialDeals
- Warehouse.StockItemHoldings
- Warehouse.StockItemStockGroups
- Warehouse.StockItemTransactions
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
- Integration.GetPurchaseUpdates
- Integration.GetSaleUpdates

- Integration.GetStockItemUpdates
- Website.InsertCustomerOrders
- Website.InvoiceCustomerOrders
- Website.SearchForStockItems
- Website.SearchForStockItemsByTags
- Website.CalculateCustomerPrice

Warehouse.StockItems_Archive

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	PAGE
Row Count (~)	444
Created	25-May-16 15:36:54
Last Modified	25-May-16 16:42:39

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	StockItemID	int	4	10	0	True				False	False	
	StockItemName	nvarchar	100	0	0	True				False	False	
	SupplierID	int	4	10	0	True				False	False	
	ColorID	int	4	10	0	False				False	False	
	UnitPackageID	int	4	10	0	True				False	False	
	OuterPackageID	int	4	10	0	True				False	False	
	Brand	nvarchar	50	0	0	False				False	False	
	Size	nvarchar	20	0	0	False				False	False	
	LeadTimeDays	int	4	10	0	True				False	False	
	QuantityPerOuter	int	4	10	0	True				False	False	
	IsChillerStock	bit	1	1	0	True				False	False	

	Barcode	nvarchar	50	0	0	False				False	False	
	TaxRate	decimal	9	18	3	True				False	False	
	UnitPrice	decimal	9	18	2	True				False	False	
	RecommendedRetailPrice	decimal	9	18	2	False				False	False	
	TypicalWeightPerUnit	decimal	9	18	3	True				False	False	
	MarketingComments	nvarchar		0	0	False				False	False	
	InternalComments	nvarchar		0	0	False				False	False	
	Photo	varbinary		0	0	False				False	False	
	CustomFields	nvarchar		0	0	False				False	False	
	Tags	nvarchar		0	0	False				False	False	
	SearchDetails	nvarchar		0	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	ValidFrom	datetime2	8	27	7	True				False	False	
	ValidTo	datetime2	8	27	7	True				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	ix_StockItems_Archive	ValidTo, ValidFrom	False		

SQL Script

```
CREATE TABLE Warehouse.StockItems_Archive (
  StockItemID int NOT NULL,
  StockItemName nvarchar(100) NOT NULL,
  SupplierID int NOT NULL,
  ColorID int NULL,
  UnitPackageID int NOT NULL,
  OuterPackageID int NOT NULL,
  Brand nvarchar(50) NULL,
  Size nvarchar(20) NULL,
  LeadTimeDays int NOT NULL,
  QuantityPerOuter int NOT NULL,
  IsChillerStock bit NOT NULL,
  Barcode nvarchar(50) NULL,
  TaxRate decimal(18, 3) NOT NULL,
  UnitPrice decimal(18, 2) NOT NULL,
  RecommendedRetailPrice decimal(18, 2) NULL,
  TypicalWeightPerUnit decimal(18, 3) NOT NULL,
  MarketingComments nvarchar(max) NULL,
  InternalComments nvarchar(max) NULL,
  Photo varbinary(max) NULL,
  CustomFields nvarchar(max) NULL,
  Tags nvarchar(max) NULL,
  SearchDetails nvarchar(max) NOT NULL,
  LastEditedBy int NOT NULL,
  ValidFrom datetime2 NOT NULL,
  ValidTo datetime2 NOT NULL
)
ON [PRIMARY]
TEXTIMAGE_ON [PRIMARY]
GO

CREATE CLUSTERED INDEX ix_StockItems_Archive
  ON Warehouse.StockItems_Archive (ValidTo, ValidFrom)
  ON [PRIMARY]
GO
```

Depends On 2

Warehouse
WideWorldImporters

Used By 2

DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
Integration.GetStockItemUpdates

Warehouse.StockItemStockGroups

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	442
Created	25-May-16 15:36:54
Last Modified	25-May-16 15:36:54

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	StockItemStockGroupID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[StockItemStockGroupID])	False	False	
	StockItemID	int	4	10	0	True				False	False	
	StockGroupID	int	4	10	0	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	LastEditedWhen	datetime2	8	27	7	True			(sysdatetime())	False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	PK_Warehouse_StockItemStockGroups	StockItemStockGroupID	True		
	UQ_StockItemStockGroups_StockGroupID_Lookup	StockGroupID, StockItemID	True		

	UQ_StockItemStockGroups_StockItemID_Lookup	StockItemID, StockGroupID	True		

Foreign Keys

Name	Columns	Description
FK_Warehouse_StockItemStockGroups_Application_People	PersonID	
FK_Warehouse_StockItemStockGroups_StockGroupID_Warehouse_StockGroups	StockGroupID	
FK_Warehouse_StockItemStockGroups_StockItemID_Warehouse_StockItems	StockItemID	

Extended Properties

Name	Value
Description	Which stock items are in which stock groups

SQL Script

```
CREATE TABLE Warehouse.StockItemStockGroups (
  StockItemStockGroupID int NOT NULL CONSTRAINT DF_Warehouse_StockItemStockGroups_StockItemStockGroupID DEFAULT (
  NEXT VALUE FOR [Sequences].[StockItemStockGroupID]),
  StockItemID int NOT NULL,
  StockGroupID int NOT NULL,
  LastEditedBy int NOT NULL,
  LastEditedWhen datetime2 NOT NULL CONSTRAINT DF_Warehouse_StockItemStockGroups_LastEditedWhen DEFAULT (
  sysdatetime()),
  CONSTRAINT PK_Warehouse_StockItemStockGroups PRIMARY KEY CLUSTERED (StockItemStockGroupID),
  CONSTRAINT UQ_StockItemStockGroups_StockGroupID_Lookup UNIQUE (StockGroupID, StockItemID),
  CONSTRAINT UQ_StockItemStockGroups_StockItemID_Lookup UNIQUE (StockItemID, StockGroupID)
)
ON [PRIMARY]
GO

ALTER TABLE Warehouse.StockItemStockGroups
  ADD CONSTRAINT FK_Warehouse_StockItemStockGroups_Application_People FOREIGN KEY (LastEditedBy) REFERENCES
  Application.People (PersonID)
GO

ALTER TABLE Warehouse.StockItemStockGroups
  ADD CONSTRAINT FK_Warehouse_StockItemStockGroups_StockGroupID_Warehouse_StockGroups FOREIGN KEY (StockGroupID)
  REFERENCES Warehouse.StockGroups (StockGroupID)
GO

ALTER TABLE Warehouse.StockItemStockGroups
  ADD CONSTRAINT FK_Warehouse_StockItemStockGroups_StockItemID_Warehouse_StockItems FOREIGN KEY (StockItemID)
  REFERENCES Warehouse.StockItems (StockItemID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Which stock items are in which stock
groups', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemStockGroups'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Internal reference for this linking
row', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemStockGroups', 'COLUMN', N'StockItemStockGroupID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Stock item assigned to this stock group (FK indexed via unique
constraint)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemStockGroups', 'COLUMN', N'StockItemID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'StockGroup assigned to this stock item (FK indexed via unique
constraint)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemStockGroups', 'COLUMN', N'StockGroupID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Enforces uniqueness and indexes one side of the many to many
relationship', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemStockGroups', 'CONSTRAINT', N'UQ_StockItemStockGroups_StockGroupID
_Lookup'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Enforces uniqueness and indexes one side of the many to many
relationship', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemStockGroups', 'CONSTRAINT', N'UQ_StockItemStockGroups_StockItemID_
Lookup'
GO
```

Depends On 5

- Warehouse
- Application.People
- Warehouse.StockGroups
- Warehouse.StockItems
- Sequences.StockItemStockGroupID

Used By 1

- Website.CalculateCustomerPrice

Warehouse.StockItemTransactions

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	True
Full Text Catalog	
Full Text	
Compression	COLUMNSTORE
Row Count (~)	227861
Created	25-May-16 15:36:54
Last Modified	26-May-16 09:02:17

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
	StockItemTransactionID	int	4	10	0	True			(NEXT VALUE FOR [Sequences].[TransactionID])	False	False	
	StockItemID	int	4	10	0	True				False	False	
	TransactionTypeID	int	4	10	0	True				False	False	
	CustomerID	int	4	10	0	False				False	False	
	InvoiceID	int	4	10	0	False				False	False	

	SupplierID	int	4	10	0	False				False	False	
	PurchaseOrderID	int	4	10	0	False				False	False	
	TransactionOccurredWhen	datetime2	8	27	7	True				False	False	
	Quantity	decimal	9	18	3	True				False	False	
	LastEditedBy	int	4	10	0	True				False	False	
	LastEditedWhen	datetime2	8	27	7	True			(sysdatetime())	False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
	CCX_Warehouse_StockItemTransactions	StockItemTransactionID, StockItemID, TransactionTypeID, CustomerID, InvoiceID, SupplierID, PurchaseOrderID, TransactionOccurredWhen, Quantity, LastEditedBy, LastEditedWhen	False		
	FK_Warehouse_StockItemTransactions_CustomerID	CustomerID	False		
	FK_Warehouse_StockItemTransactions_InvoiceID	InvoiceID	False		
	FK_Warehouse_StockItemTransactions_PurchaseOrderID	PurchaseOrderID	False		
	FK_Warehouse_StockItemTransactions_StockItemID	StockItemID	False		
	FK_Warehouse_StockItemTransactions_SupplierID	SupplierID	False		
	FK_Warehouse_StockItemTransactions_TransactionTypeID	TransactionTypeID	False		
	PK_Warehouse_StockItemTransactions	StockItemTransactionID	True		

Foreign Keys

Name	Columns	Description
FK_Warehouse_StockItemTransactions_Application_People	PersonID	
FK_Warehouse_StockItemTransactions_CustomerID_Sales_Customers	CustomerID	
FK_Warehouse_StockItemTransactions_InvoiceID_Sales_Invoices	InvoiceID	
FK_Warehouse_StockItemTransactions_PurchaseOrderID_Purchasing_PurchaseOrders	PurchaseOrderID	
FK_Warehouse_StockItemTransactions_StockItemID_Warehouse_StockItems	StockItemID	
FK_Warehouse_StockItemTransactions_SupplierID_Purchasing_Suppliers	SupplierID	
FK_Warehouse_StockItemTransactions_TransactionTypeID_Application_TransactionTypes	TransactionTypeID	

Extended Properties

Name	Value
Description	Transactions covering all movements of all stock items

SQL Script

```
CREATE TABLE Warehouse.StockItemTransactions (
  StockItemTransactionID int NOT NULL CONSTRAINT DF_Warehouse_StockItemTransactions_StockItemTransactionID DEFAULT
  (NEXT VALUE FOR [Sequences].[TransactionID]),
  StockItemID int NOT NULL,
  TransactionTypeID int NOT NULL,
  CustomerID int NULL,
  InvoiceID int NULL,
  SupplierID int NULL,
  PurchaseOrderID int NULL,
  TransactionOccurredWhen datetime2 NOT NULL,
  Quantity decimal(18, 3) NOT NULL,
  LastEditedBy int NOT NULL,
  LastEditedWhen datetime2 NOT NULL CONSTRAINT DF_Warehouse_StockItemTransactions_LastEditedWhen DEFAULT (
  sysdatetime()),
  CONSTRAINT PK_Warehouse_StockItemTransactions PRIMARY KEY NONCLUSTERED (StockItemTransactionID)
)
ON [PRIMARY]
GO

CREATE CLUSTERED COLUMNSTORE INDEX CCX_Warehouse_StockItemTransactions
  ON Warehouse.StockItemTransactions
GO

CREATE INDEX FK_Warehouse_StockItemTransactions_CustomerID
  ON Warehouse.StockItemTransactions (CustomerID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Warehouse_StockItemTransactions_InvoiceID
  ON Warehouse.StockItemTransactions (InvoiceID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Warehouse_StockItemTransactions_PurchaseOrderID
  ON Warehouse.StockItemTransactions (PurchaseOrderID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Warehouse_StockItemTransactions_StockItemID
  ON Warehouse.StockItemTransactions (StockItemID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Warehouse_StockItemTransactions_SupplierID
  ON Warehouse.StockItemTransactions (SupplierID)
  ON [PRIMARY]
GO

CREATE INDEX FK_Warehouse_StockItemTransactions_TransactionTypeID
  ON Warehouse.StockItemTransactions (TransactionTypeID)
  ON [PRIMARY]
GO

ALTER TABLE Warehouse.StockItemTransactions
  ADD CONSTRAINT FK_Warehouse_StockItemTransactions_Application_People FOREIGN KEY (LastEditedBy) REFERENCES
  Application.People (PersonID)
GO

ALTER TABLE Warehouse.StockItemTransactions
  ADD CONSTRAINT FK_Warehouse_StockItemTransactions_CustomerID_Sales_Customers FOREIGN KEY (CustomerID) REFERENCES
  Sales.Customers (CustomerID)
GO

ALTER TABLE Warehouse.StockItemTransactions
  ADD CONSTRAINT FK_Warehouse_StockItemTransactions_InvoiceID_Sales_Invoices FOREIGN KEY (InvoiceID) REFERENCES
  Sales.Invoices (InvoiceID)
GO

ALTER TABLE Warehouse.StockItemTransactions
  ADD CONSTRAINT FK_Warehouse_StockItemTransactions_PurchaseOrderID_Purchasing_PurchaseOrders FOREIGN KEY (
  PurchaseOrderID) REFERENCES Purchasing.PurchaseOrders (PurchaseOrderID)
GO

ALTER TABLE Warehouse.StockItemTransactions
  ADD CONSTRAINT FK_Warehouse_StockItemTransactions_StockItemID_Warehouse_StockItems FOREIGN KEY (StockItemID)
  REFERENCES Warehouse.StockItems (StockItemID)
GO

ALTER TABLE Warehouse.StockItemTransactions
```

```
  ADD CONSTRAINT FK_Warehouse_StockItemTransactions_SupplierID_Purchasing_Suppliers FOREIGN KEY (SupplierID)
  REFERENCES Purchasing.Suppliers (SupplierID)
GO

ALTER TABLE Warehouse.StockItemTransactions
  ADD CONSTRAINT FK_Warehouse_StockItemTransactions_TransactionTypeID_Application_TransactionTypes FOREIGN KEY (
  TransactionTypeID) REFERENCES Application.TransactionTypes (TransactionTypeID)
GO

EXEC sys.sp_addextendedproperty N'Description', N'Transactions covering all movements of all stock
items', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Numeric ID used to refer to a stock item transaction within the
database', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'COLUMN', N'StockItemTransactionID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'StockItem for this
transaction', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'COLUMN', N'StockItemID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Type of
transaction', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'COLUMN', N'TransactionTypeID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Customer for this transaction (if
applicable)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'COLUMN', N'CustomerID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of an invoice (for transactions associated with an
invoice)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'COLUMN', N'InvoiceID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Supplier for this stock transaction (if
applicable)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'COLUMN', N'SupplierID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'ID of an purchase order (for transactions associated with a purchase
order)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'COLUMN', N'PurchaseOrderID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Date and time when the transaction
occurred', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'COLUMN', N'TransactionOccurredWhen'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Quantity of stock movement (positive is incoming stock, negative is
outgoing)', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'COLUMN', N'Quantity'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'INDEX', N'FK_Warehouse_StockItemTransactions_CustomerID
'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'INDEX', N'FK_Warehouse_StockItemTransactions_InvoiceID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'INDEX', N'FK_Warehouse_StockItemTransactions_PurchaseOr
derID'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'INDEX', N'FK_Warehouse_StockItemTransactions_StockItemI
D'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'INDEX', N'FK_Warehouse_StockItemTransactions_SupplierID
'
GO

EXEC sys.sp_addextendedproperty N'Description', 'Auto-created to support a foreign
key', 'SCHEMA', N'Warehouse', 'TABLE', N'StockItemTransactions', 'INDEX', N'FK_Warehouse_StockItemTransactions_Transactio
nTypeID'
GO
```

Depends On 9

- Warehouse
- Application.People
- Application.TransactionTypes
- Purchasing.PurchaseOrders
- Purchasing.Suppliers
- Sales.Customers
- Sales.Invoices
- Warehouse.StockItems
- Sequences.TransactionID

Used By 2

- Integration.GetMovementUpdates
- Website.InvoiceCustomerOrders

Warehouse.VehicleTemperatures

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	False
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	True
Full Text Catalog	
Full Text	
Compression	NONE
Row Count (~)	0
Created	26-May-16 09:02:25
Last Modified	26-May-16 09:02:25

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Rule	Default	Computed	Persisted	Description
🔑	VehicleTemperatureID	bigint	8	19	0	True	1 - 1			False	False	
	VehicleRegistration	nvarchar	20	0	0	True				False	False	
	ChillerSensorNumber	int	4	10	0	True				False	False	
	RecordedWhen	datetime2	8	27	7	True				False	False	
	Temperature	decimal	9	10	2	True				False	False	
	FullSensorData	nvarchar	1000	0	0	False				False	False	
	IsCompressed	bit	1	1	0	True				False	False	
	CompressedSensorData	varbinary		0	0	False				False	False	

Indexes

Key	Name	Columns	Unique	Type	Description
🔑	PK__VehicleT__AB2B3784F181F8BF	VehicleTemperatureID	True		

SQL Script

```
CREATE TABLE Warehouse.VehicleTemperatures (
  VehicleTemperatureID bigint IDENTITY,
  VehicleRegistration nvarchar(20) COLLATE Latin1_General_CI_AS NOT NULL,
  ChillerSensorNumber int NOT NULL,
  RecordedWhen datetime2 NOT NULL,
  Temperature decimal(10, 2) NOT NULL,
  FullSensorData nvarchar(1000) COLLATE Latin1_General_CI_AS NULL,
  IsCompressed bit NOT NULL,
  CompressedSensorData varbinary(max) NULL,
  PRIMARY KEY NONCLUSTERED (VehicleTemperatureID)
)
WITH (MEMORY_OPTIMIZED = ON)
GO
```

Depends On 1

- Warehouse

Used By 2

- Website.VehicleTemperatures
- Website.RecordVehicleTemperature

Views

Objects 3

Name	Description
Website.Customers	
Website.Suppliers	
Website.VehicleTemperatures	

Website.Customers

Description

Properties

Name	Value
Collation	Latin1_General_100_CI_AS
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Schema Bound	False
Created	25-May-16 15:38:22
Last Modified	25-May-16 15:38:22

Columns

Key	Name	Description
	CustomerID	
	CustomerName	
	CustomerCategoryName	
	PrimaryContact	
	AlternateContact	
	PhoneNumber	
	FaxNumber	
	BuyingGroupName	
	WebsiteURL	
	DeliveryMethod	
	CityName	
	DeliveryLocation	
	DeliveryRun	
	RunPosition	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER VIEW Website.Customers
AS
SELECT s.CustomerID,
       s.CustomerName,
       sc.CustomerCategoryName,
       pp.FullName AS PrimaryContact,
       ap.FullName AS AlternateContact,
       s.PhoneNumber,
```

```
        s.FaxNumber,
        bg.BuyingGroupName,
        s.WebsiteURL,
        dm.DeliveryMethodName AS DeliveryMethod,
        c.CityName AS CityName,
        s.DeliveryLocation AS DeliveryLocation,
        s.DeliveryRun,
        s.RunPosition
FROM Sales.Customers AS s
LEFT OUTER JOIN Sales.CustomerCategories AS sc
ON s.CustomerCategoryID = sc.CustomerCategoryID
LEFT OUTER JOIN [Application].People AS pp
ON s.PrimaryContactPersonID = pp.PersonID
LEFT OUTER JOIN [Application].People AS ap
ON s.AlternateContactPersonID = ap.PersonID
LEFT OUTER JOIN Sales.BuyingGroups AS bg
ON s.BuyingGroupID = bg.BuyingGroupID
LEFT OUTER JOIN [Application].DeliveryMethods AS dm
ON s.DeliveryMethodID = dm.DeliveryMethodID
LEFT OUTER JOIN [Application].Cities AS c
ON s.DeliveryCityID = c.CityID
GO
```

Depends On 7

- Website
- Sales.Customers
- Sales.CustomerCategories
- Application.People
- Sales.BuyingGroups
- Application.DeliveryMethods
- Application.Cities

Used By

No items found

Website.Suppliers

Description

Properties

Name	Value
Collation	Latin1_General_100_CI_AS
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Schema Bound	False
Created	25-May-16 15:38:22
Last Modified	25-May-16 15:38:22

Columns

Key	Name	Description
	SupplierID	
	SupplierName	
	SupplierCategoryName	
	PrimaryContact	
	AlternateContact	
	PhoneNumber	
	FaxNumber	
	WebsiteURL	
	DeliveryMethod	
	CityName	
	DeliveryLocation	
	SupplierReference	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER VIEW Website.Suppliers
AS
SELECT s.SupplierID,
       s.SupplierName,
       sc.SupplierCategoryName,
       pp.FullName AS PrimaryContact,
       ap.FullName AS AlternateContact,
       s.PhoneNumber,
       s.FaxNumber,
       s.WebsiteURL,
       dm.DeliveryMethodName AS DeliveryMethod,
       c.CityName AS CityName,
```

```
       s.DeliveryLocation AS DeliveryLocation,
       s.SupplierReference
FROM Purchasing.Suppliers AS s
LEFT OUTER JOIN Purchasing.SupplierCategories AS sc
ON s.SupplierCategoryID = sc.SupplierCategoryID
LEFT OUTER JOIN [Application].People AS pp
ON s.PrimaryContactPersonID = pp.PersonID
LEFT OUTER JOIN [Application].People AS ap
ON s.AlternateContactPersonID = ap.PersonID
LEFT OUTER JOIN [Application].DeliveryMethods AS dm
ON s.DeliveryMethodID = dm.DeliveryMethodID
LEFT OUTER JOIN [Application].Cities AS c
ON s.DeliveryCityID = c.CityID
GO
```

Depends On 6

- Website
- Purchasing.Suppliers
- Purchasing.SupplierCategories
- Application.People
- Application.DeliveryMethods
- Application.Cities

Used By

No items found

Website.VehicleTemperatures

Description

Properties

Name	Value
Collation	Latin1_General_100_CI_AS
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Schema Bound	False
Created	25-May-16 15:38:22
Last Modified	25-May-16 15:38:22

Columns

Key	Name	Description
	VehicleTemperatureID	
	VehicleRegistration	
	ChillerSensorNumber	
	RecordedWhen	
	Temperature	
	FullSensorData	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER VIEW Website.VehicleTemperatures
AS
SELECT vt.VehicleTemperatureID,
       vt.VehicleRegistration,
       vt.ChillerSensorNumber,
       vt.RecordedWhen,
       vt.Temperature,
       CASE WHEN vt.IsCompressed <> 0
            THEN CAST(DECOMPRESS(vt.CompressedSensorData) AS nvarchar(1000))
            ELSE vt.FullSensorData
       END AS FullSensorData
FROM Warehouse.VehicleTemperatures AS vt;
GO
```

Depends On 2

- Website
- Warehouse.VehicleTemperatures

Used By

No items found

Programmability

Objects 4

- Stored Procedures
- Functions
- Types
- Sequences

Stored Procedures

Objects 42

Name	Description
Application.AddRoleMemberIfNonexistent	
Application.Configuration_ApplyAuditing	
Application.Configuration_ApplyColumnstoreIndexing	
Application.Configuration_ApplyFullTextIndexing	
Application.Configuration_ApplyPartitioning	
Application.Configuration_ApplyRowLevelSecurity	
Application.Configuration_ConfigureForEnterpriseEdition	
Application.Configuration_EnableInMemory	
Application.Configuration_RemoveAuditing	
Application.Configuration_RemoveRowLevelSecurity	
Application.CreateRoleIfNonexistent	
DataLoadSimulation.Configuration_ApplyDataLoadSimulationProcedures	
DataLoadSimulation.Configuration_RemoveDataLoadSimulationProcedures	
DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad	
DataLoadSimulation.PopulateDataToCurrentDate	
DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad	
Integration.GetCityUpdates	
Integration.GetCustomerUpdates	
Integration.GetEmployeeUpdates	
Integration.GetMovementUpdates	
Integration.GetOrderUpdates	
Integration.GetPaymentMethodUpdates	
Integration.GetPurchaseUpdates	
Integration.GetSaleUpdates	
Integration.GetStockHoldingUpdates	
Integration.GetStockItemUpdates	
Integration.GetSupplierUpdates	
Integration.GetTransactionTypeUpdates	
Integration.GetTransactionUpdates	
Sequences.ReseedAllSequences	
Sequences.ReseedSequenceBeyondTableValues	
Website.ActivateWebsiteLogon	

Website.ChangePassword	
Website.InsertCustomerOrders	
Website.InvoiceCustomerOrders	
Website.RecordColdRoomTemperatures	
Website.RecordVehicleTemperature	
Website.SearchForCustomers	
Website.SearchForPeople	
Website.SearchForStockItems	
Website.SearchForStockItemsByTags	
Website.SearchForSuppliers	

Application.AddRoleMemberIfNonexistent

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@RoleName	sysname	256	
@UserName	sysname	256	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Application.AddRoleMemberIfNonexistent
@RoleName sysname,
@UserName sysname
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    IF NOT EXISTS (SELECT 1 FROM sys.database_role_members AS drm
                           INNER JOIN sys.database_principals AS dpr
                           ON drm.role_principal_id = dpr.principal_id
                           AND dpr.type = N'R'
                           INNER JOIN sys.database_principals AS dpu
                           ON drm.member_principal_id = dpu.principal_id
                           AND dpu.type = N'S'
                           WHERE dpr.name = @RoleName
                           AND dpu.name = @UserName)
    BEGIN
        BEGIN TRY

            DECLARE @SQL nvarchar(max) = N'ALTER ROLE ' + QUOTENAME(@RoleName)
                                       + N' ADD MEMBER ' + QUOTENAME(@UserName) + N';'
            EXECUTE (@SQL);

            PRINT N'User ' + @UserName + N' added to role ' + @RoleName;

        END TRY
        BEGIN CATCH
            PRINT N'Unable to add user ' + @UserName + N' to role ' + @RoleName;
            THROW;
        END CATCH;
    END;
END;
GO
```

Depends On 1

Application

Used By

No items found

Application.Configuration_ApplyAuditing

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Application.Configuration_ApplyAuditing
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @AreDatabaseAuditSpecificationsSupported bit = 0;
    DECLARE @SQL nvarchar(max);

    -- TODO !! - currently no separate test for audit
    -- but same editions with XTP support database audit specs
    IF SERVERPROPERTY(N'IsXTPSupported') <> 0 SET @AreDatabaseAuditSpecificationsSupported = 1;

    BEGIN TRY;

        IF NOT EXISTS (SELECT 1 FROM sys.server_audits WHERE name = N'WWI_Audit')
        BEGIN
            SET @SQL = N'
            USE master;

            CREATE SERVER AUDIT [WWI_Audit]
            TO APPLICATION_LOG
            WITH
            (
             QUEUE_DELAY = 1000,
             ON_FAILURE = CONTINUE
            );';
            EXECUTE (@SQL);

            PRINT N'Server audit WWI_Audit created with Application Log as a target.';
            PRINT N'For stronger security, redirect the audit to the security log or a text file in a
            secure folder.';
            PRINT N'Additional configuration is required when using the security log.';
            PRINT N'For more information see: https://technet.microsoft.com/en-us/library/cc645889.aspx.
            ';
        END;

        IF NOT EXISTS (SELECT 1 FROM sys.server_audit_specifications WHERE name = N'WWI_
        ServerAuditSpecification')
        BEGIN
            SET @SQL = N'
            USE master;

            CREATE SERVER AUDIT SPECIFICATION [WWI_ServerAuditSpecification]
            FOR SERVER AUDIT [WWI_Audit]
            ADD (AUDIT_CHANGE_GROUP),
```

```
            ADD (DATABASE_CHANGE_GROUP),
            ADD (DATABASE_OWNERSHIP_CHANGE_GROUP),
            ADD (DATABASE_ROLE_MEMBER_CHANGE_GROUP),
            ADD (FAILED_LOGIN_GROUP),
            ADD (TRACE_CHANGE_GROUP);';
            EXECUTE (@SQL);
        END;

        IF @AreDatabaseAuditSpecificationsSupported <> 0
        BEGIN
            IF NOT EXISTS (SELECT 1 FROM sys.database_audit_specifications WHERE name = N'WWI_
            DatabaseAuditSpecification')
            BEGIN
                SET @SQL = N'
                CREATE DATABASE AUDIT SPECIFICATION [WWI_DatabaseAuditSpecification]
                FOR SERVER AUDIT [WWI_Audit]
                ADD (AUDIT_CHANGE_GROUP),
                ADD (DATABASE_CHANGE_GROUP),
                ADD (DATABASE_OWNERSHIP_CHANGE_GROUP),
                ADD (DATABASE_PRINCIPAL_CHANGE_GROUP),
                ADD (DATABASE_ROLE_MEMBER_CHANGE_GROUP),
                ADD (DATABASE_OBJECT_CHANGE_GROUP),
                ADD (SELECT ON OBJECT::[Sales].[CustomerTransactions] BY [public]),
                ADD (SELECT ON OBJECT::[Purchasing].[SupplierTransactions] BY [public]);';
                EXECUTE (@SQL);
            END;
        END;

    END TRY
    BEGIN CATCH
        PRINT N'Unable to apply audit';
        THROW;
    END CATCH;
END;
GO
```

Depends On 1

Application

Used By

No items found

Application.Configuration_ApplyColumnstoreIndexing

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Application.Configuration_ApplyColumnstoreIndexing
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    IF SERVERPROPERTY(N'IsXTPSupported') = 0 -- TODO !! - currently no separate test for columnstore
    BEGIN                                    -- but same editions with XTP support columnstore
        PRINT N'Warning: Columnstore indexes cannot be created on this edition.';
    END ELSE BEGIN -- if columnstore can be created
        DECLARE @SQL nvarchar(max) = N'';

        BEGIN TRY;
   -- auto_update_statistics_async is enabled to reduce impact of auto-update stats
   -- for analytics queries run during an insert workload
   SET @SQL = N'
   ALTER DATABASE CURRENT SET AUTO_UPDATE_STATISTICS_ASYNC ON';
   SET @SQL += N';';
               EXECUTE (@SQL);

           BEGIN TRAN;

           IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = N'NCCX_Sales_OrderLines')
           BEGIN
               SET @SQL = N'
               CREATE NONCLUSTERED COLUMNSTORE INDEX NCCX_Sales_OrderLines
               ON Sales.OrderLines
               (
                OrderID,
                StockItemID,
                [Description],
                Quantity,
                UnitPrice,
                PickedQuantity
               )';

    SET @SQL += N';';
               EXECUTE (@SQL);
           END;

           IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = N'NCCX_Sales_InvoiceLines')
           BEGIN
               SET @SQL = N'
               CREATE NONCLUSTERED COLUMNSTORE INDEX NCCX_Sales_InvoiceLines
               ON Sales.InvoiceLines
```

```
                (
                 InvoiceID,
                 StockItemID,
                 Quantity,
                 UnitPrice,
                 LineProfit,
                 LastEditedWhen
                )';

    SET @SQL += N';';
                EXECUTE (@SQL);
            END;

            IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = N'CCX_Warehouse_StockItemTransactions')
            BEGIN
                SET @SQL = N'
                ALTER TABLE Warehouse.StockItemTransactions
                DROP CONSTRAINT PK_Warehouse_StockItemTransactions;';
                EXECUTE (@SQL);

                SET @SQL = N'
                ALTER TABLE Warehouse.StockItemTransactions
                ADD CONSTRAINT PK_Warehouse_StockItemTransactions PRIMARY KEY NONCLUSTERED (StockItemTransactionID);
                ';
                EXECUTE (@SQL);

                SET @SQL = N'
                CREATE CLUSTERED COLUMNSTORE INDEX CCX_Warehouse_StockItemTransactions
                ON Warehouse.StockItemTransactions;';
                EXECUTE (@SQL);

                SET @SQL = N'
                ALTER INDEX CCX_Warehouse_StockItemTransactions
                ON Warehouse.StockItemTransactions
                REORGANIZE WITH (COMPRESS_ALL_ROW_GROUPS = ON);';
                EXECUTE (@SQL);

                PRINT N'Successfully applied columnstore indexing';
            END; -- of if need to apply to stock item transactions

            COMMIT;
        END TRY
        BEGIN CATCH
            PRINT N'Unable to apply columnstore indexing';
            THROW;
        END CATCH;
    END; -- of columnstore is allowed
END;
GO
```

Depends On 1

- Application

Used By 1

- Application.Configuration_ConfigureForEnterpriseEdition

Application.Configuration_ApplyFullTextIndexing

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Application.Configuration_ApplyFullTextIndexing
WITH EXECUTE AS OWNER
AS
BEGIN
    IF SERVERPROPERTY(N'IsFullTextInstalled') = 0
    BEGIN
        PRINT N'Warning: Full text options cannot be configured because full text indexing is not installed.
        ';
    END ELSE BEGIN -- if full text is installed
        DECLARE @SQL nvarchar(max) = N'';

        IF NOT EXISTS (SELECT 1 FROM sys.fulltext_catalogs WHERE name = N'FTCatalog')
        BEGIN
            SET @SQL =  N'CREATE FULLTEXT CATALOG FTCatalog AS DEFAULT;'
            EXECUTE (@SQL);
        END;

        IF NOT EXISTS (SELECT 1 FROM sys.fulltext_indexes AS fti WHERE fti.object_id = OBJECT_ID(N'[
        Application].People'))
        BEGIN
            SET @SQL = N'
            CREATE FULLTEXT INDEX
            ON [Application].People (SearchName, CustomFields, OtherLanguages)
            KEY INDEX PK_Application_People
            WITH CHANGE_TRACKING AUTO;';
            EXECUTE (@SQL);
        END;

        IF NOT EXISTS (SELECT 1 FROM sys.fulltext_indexes AS fti WHERE fti.object_id = OBJECT_ID(N'Sales.
        Customers'))
        BEGIN
            SET @SQL = N'
            CREATE FULLTEXT INDEX
            ON Sales.Customers (CustomerName)
            KEY INDEX PK_Sales_Customers
            WITH CHANGE_TRACKING AUTO;';
            EXECUTE (@SQL);
        END;

        IF NOT EXISTS (SELECT 1 FROM sys.fulltext_indexes AS fti WHERE fti.object_id = OBJECT_ID(N'
        Purchasing.Suppliers'))
        BEGIN
            SET @SQL = N'
            CREATE FULLTEXT INDEX
            ON Purchasing.Suppliers (SupplierName)
            KEY INDEX PK_Purchasing_Suppliers
```

```
    WITH CHANGE_TRACKING AUTO;';
    EXECUTE (@SQL);
END;


IF NOT EXISTS (SELECT 1 FROM sys.fulltext_indexes AS fti WHERE fti.object_id = OBJECT_ID(N'
Warehouse.StockItems'))
BEGIN
    SET @SQL = N'CREATE FULLTEXT INDEX
    ON Warehouse.StockItems (SearchDetails, CustomFields, Tags)
    KEY INDEX PK_Warehouse_StockItems
    WITH CHANGE_TRACKING AUTO;';
    EXECUTE (@SQL);
END;

SET @SQL = N'DROP PROCEDURE IF EXISTS Website.SearchForPeople;';
EXECUTE (@SQL);

SET @SQL = N'
CREATE PROCEDURE Website.SearchForPeople
@SearchText nvarchar(1000),
@MaximumRowsToReturn int
AS
BEGIN
 SELECT p.PersonID,
 p.FullName,
 p.PreferredName,
 CASE WHEN p.IsSalesperson <> 0 THEN N''Salesperson''
 WHEN p.IsEmployee <> 0 THEN N''Employee''
 WHEN c.CustomerID IS NOT NULL THEN N''Customer''
 WHEN sp.SupplierID IS NOT NULL THEN N''Supplier''
 WHEN sa.SupplierID IS NOT NULL THEN N''Supplier''
 END AS Relationship,
 COALESCE(c.CustomerName, sp.SupplierName, sa.SupplierName, N''WWI'') AS Company
 FROM [Application].People AS p
 INNER JOIN FREETEXTTABLE([Application].People, SearchName, @SearchText, @MaximumRowsToReturn) AS ft
 ON p.PersonID = ft.[KEY]
 LEFT OUTER JOIN Sales.Customers AS c
 ON c.PrimaryContactPersonID = p.PersonID
 LEFT OUTER JOIN Purchasing.Suppliers AS sp
 ON sp.PrimaryContactPersonID = p.PersonID
 LEFT OUTER JOIN Purchasing.Suppliers AS sa
 ON sa.AlternateContactPersonID = p.PersonID
 ORDER BY ft.[RANK]
 FOR JSON AUTO, ROOT(N''People'');
END;';
EXECUTE (@SQL);

SET @SQL = N'DROP PROCEDURE IF EXISTS Website.SearchForSuppliers;';
EXECUTE (@SQL);

SET @SQL = N'
CREATE PROCEDURE Website.SearchForSuppliers
@SearchText nvarchar(1000),
@MaximumRowsToReturn int
AS
BEGIN
 SELECT s.SupplierID,
 s.SupplierName,
 c.CityName,
 s.PhoneNumber,
 s.FaxNumber ,
 p.FullName AS PrimaryContactFullName,
 p.PreferredName AS PrimaryContactPreferredName
 FROM Purchasing.Suppliers AS s
 INNER JOIN FREETEXTTABLE(Purchasing.Suppliers, SupplierName, @SearchText, @MaximumRowsToReturn) AS ft
 ON s.SupplierID = ft.[KEY]
 INNER JOIN [Application].Cities AS c
 ON s.DeliveryCityID = c.CityID
 LEFT OUTER JOIN [Application].People AS p
 ON s.PrimaryContactPersonID = p.PersonID
 ORDER BY ft.[RANK]
 FOR JSON AUTO, ROOT(N''Suppliers'');
END;';
EXECUTE (@SQL);

SET @SQL = N'DROP PROCEDURE IF EXISTS Website.SearchForCustomers;';
EXECUTE (@SQL);
```

```
        SET @SQL = N'
        CREATE PROCEDURE Website.SearchForCustomers
        @SearchText nvarchar(1000),
        @MaximumRowsToReturn int
        WITH EXECUTE AS OWNER
        AS
        BEGIN
         SELECT c.CustomerID,
         c.CustomerName,
         ct.CityName,
         c.PhoneNumber,
         c.FaxNumber,
         p.FullName AS PrimaryContactFullName,
         p.PreferredName AS PrimaryContactPreferredName
         FROM Sales.Customers AS c
         INNER JOIN FREETEXTTABLE(Sales.Customers, CustomerName, @SearchText, @MaximumRowsToReturn) AS ft
         ON c.CustomerID = ft.[KEY]
         INNER JOIN [Application].Cities AS ct
         ON c.DeliveryCityID = ct.CityID
         LEFT OUTER JOIN [Application].People AS p
         ON c.PrimaryContactPersonID = p.PersonID
         ORDER BY ft.[RANK]
         FOR JSON AUTO, ROOT(N''Customers'');
        END;';
        EXECUTE (@SQL);

        SET @SQL = N'DROP PROCEDURE IF EXISTS Website.SearchForStockItems;';
        EXECUTE (@SQL);

        SET @SQL = N'
        CREATE PROCEDURE Website.SearchForStockItems
        @SearchText nvarchar(1000),
        @MaximumRowsToReturn int
        WITH EXECUTE AS OWNER
        AS
        BEGIN
         SELECT si.StockItemID,
         si.StockItemName
         FROM Warehouse.StockItems AS si
         INNER JOIN FREETEXTTABLE(Warehouse.StockItems, SearchDetails, @SearchText, @MaximumRowsToReturn) AS ft
         ON si.StockItemID = ft.[KEY]
         ORDER BY ft.[RANK]
         FOR JSON AUTO, ROOT(N''StockItems'');
        END;';
        EXECUTE (@SQL);

        SET @SQL = N'DROP PROCEDURE IF EXISTS Website.SearchForStockItemsByTags;';
        EXECUTE (@SQL);

        SET @SQL = N'
        CREATE PROCEDURE Website.SearchForStockItemsByTags
        @SearchText nvarchar(1000),
        @MaximumRowsToReturn int
        WITH EXECUTE AS OWNER
        AS
        BEGIN
         SELECT si.StockItemID,
         si.StockItemName
         FROM Warehouse.StockItems AS si
         INNER JOIN FREETEXTTABLE(Warehouse.StockItems, Tags, @SearchText, @MaximumRowsToReturn) AS ft
         ON si.StockItemID = ft.[KEY]
         ORDER BY ft.[RANK]
         FOR JSON AUTO, ROOT(N''StockItems'');
        END;';
        EXECUTE (@SQL);

        PRINT N'Full text successfully enabled';
    END;
END;
GO
```

Depends On 1

Application

Used By 1

Application.Configuration_ConfigureForEnterpriseEdition

Application.Configuration_ApplyPartitioning

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Application.Configuration_ApplyPartitioning
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    IF SERVERPROPERTY(N'IsXTPSupported') = 0 -- TODO Check for a better way to check for partitioning
    BEGIN
    -- Currently versions that support in-memory OLTP also support partitions
        PRINT N'Warning: Partitions are not supported in this edition.';
    END ELSE BEGIN -- if partitions are permitted

        BEGIN TRAN;

        DECLARE @SQL nvarchar(max) = N'';

        IF NOT EXISTS (SELECT 1 FROM sys.partition_functions WHERE name = N'PF_TransactionDateTime')
        BEGIN
            SET @SQL =  N'
            CREATE PARTITION FUNCTION PF_TransactionDateTime(datetime)
            AS RANGE RIGHT
            FOR VALUES (N''20140101'', N''20150101'', N''20160101'', N''20170101'');';
            EXECUTE (@SQL);
        END;

        IF NOT EXISTS (SELECT 1 FROM sys.partition_functions WHERE name = N'PF_TransactionDate')
        BEGIN
            SET @SQL =  N'
            CREATE PARTITION FUNCTION PF_TransactionDate(date)
            AS RANGE RIGHT
            FOR VALUES (N''20140101'', N''20150101'', N''20160101'', N''20170101'');';
            EXECUTE (@SQL);
        END;

        IF NOT EXISTS (SELECT * FROM sys.partition_schemes WHERE name = N'PS_TransactionDateTime')
        BEGIN
            SET @SQL =  N'
            CREATE PARTITION SCHEME PS_TransactionDateTime
            AS PARTITION PF_TransactionDateTime
            ALL TO ([PRIMARY]);';
            EXECUTE (@SQL);
        END;

        IF NOT EXISTS (SELECT 1 FROM sys.partition_schemes WHERE name = N'PS_TransactionDate')
        BEGIN
```

```
    SET @SQL =  N'
    CREATE PARTITION SCHEME PS_TransactionDate
    AS PARTITION PF_TransactionDate
    ALL TO ([PRIMARY]);';
    EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = N'CX_Sales_CustomerTransactions')
BEGIN
    SET @SQL =  N'
    ALTER TABLE Sales.CustomerTransactions
    DROP CONSTRAINT PK_Sales_CustomerTransactions;';
    EXECUTE (@SQL);

    SET @SQL = N'
    ALTER TABLE Sales.CustomerTransactions
    ADD CONSTRAINT PK_Sales_CustomerTransactions PRIMARY KEY NONCLUSTERED
    (
     CustomerTransactionID
    );';
    EXECUTE (@SQL);

    SET @SQL = N'
    CREATE CLUSTERED INDEX CX_Sales_CustomerTransactions
    ON Sales.CustomerTransactions
    (
     TransactionDate
    )
    ON PS_TransactionDate(TransactionDate);';
    EXECUTE (@SQL);

    SET @SQL = N'
    CREATE INDEX FK_Sales_CustomerTransactions_CustomerID
    ON Sales.CustomerTransactions
    (
     CustomerID
    )
    WITH (DROP_EXISTING = ON)
    ON PS_TransactionDate(TransactionDate);';
    EXECUTE (@SQL);

    SET @SQL = N'
    CREATE INDEX FK_Sales_CustomerTransactions_InvoiceID
    ON Sales.CustomerTransactions
    (
     InvoiceID
    )
    WITH (DROP_EXISTING = ON)
    ON PS_TransactionDate(TransactionDate);';
    EXECUTE (@SQL);

    SET @SQL = N'
    CREATE INDEX FK_Sales_CustomerTransactions_PaymentMethodID
    ON Sales.CustomerTransactions
    (
     PaymentMethodID
    )
    WITH (DROP_EXISTING = ON)
    ON PS_TransactionDate(TransactionDate);';
    EXECUTE (@SQL);

    SET @SQL = N'
    CREATE INDEX FK_Sales_CustomerTransactions_TransactionTypeID
    ON Sales.CustomerTransactions
    (
     TransactionTypeID
    )
    WITH (DROP_EXISTING = ON)
    ON PS_TransactionDate(TransactionDate);';
    EXECUTE (@SQL);

    SET @SQL = N'
    CREATE INDEX IX_Sales_CustomerTransactions_IsFinalized
    ON Sales.CustomerTransactions
    (
     IsFinalized
    )
    WITH (DROP_EXISTING = ON)
    ON PS_TransactionDate(TransactionDate);';
```

```
    EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = N'CX_Purchasing_SupplierTransactions')
BEGIN
    SET @SQL =  N'
    ALTER TABLE Purchasing.SupplierTransactions
    DROP CONSTRAINT PK_Purchasing_SupplierTransactions;';
    EXECUTE (@SQL);

    SET @SQL =  N'
    ALTER TABLE Purchasing.SupplierTransactions
    ADD CONSTRAINT PK_Purchasing_SupplierTransactions PRIMARY KEY NONCLUSTERED
    (
     SupplierTransactionID
    );';
    EXECUTE (@SQL);

    SET @SQL =  N'
    CREATE CLUSTERED INDEX CX_Purchasing_SupplierTransactions
    ON Purchasing.SupplierTransactions
    (
     TransactionDate
    )
    ON PS_TransactionDate(TransactionDate);';
    EXECUTE (@SQL);

    SET @SQL =  N'
    CREATE INDEX FK_Purchasing_SupplierTransactions_PaymentMethodID
    ON Purchasing.SupplierTransactions
    (
     PaymentMethodID
    )
    WITH (DROP_EXISTING = ON)
    ON PS_TransactionDate(TransactionDate);';
    EXECUTE (@SQL);

    SET @SQL =  N'
    CREATE INDEX FK_Purchasing_SupplierTransactions_PurchaseOrderID
    ON Purchasing.SupplierTransactions
    (
     PurchaseOrderID
    )
    WITH (DROP_EXISTING = ON)
    ON PS_TransactionDate(TransactionDate);';
    EXECUTE (@SQL);

    SET @SQL =  N'
    CREATE INDEX FK_Purchasing_SupplierTransactions_SupplierID
    ON Purchasing.SupplierTransactions
    (
     SupplierID
    )
    WITH (DROP_EXISTING = ON)
    ON PS_TransactionDate(TransactionDate);';
    EXECUTE (@SQL);

    SET @SQL =  N'
    CREATE INDEX FK_Purchasing_SupplierTransactions_TransactionTypeID
    ON Purchasing.SupplierTransactions
    (
     TransactionTypeID
    )
    WITH (DROP_EXISTING = ON)
    ON PS_TransactionDate(TransactionDate);';
    EXECUTE (@SQL);

    SET @SQL =  N'
    CREATE INDEX IX_Purchasing_SupplierTransactions_IsFinalized
    ON Purchasing.SupplierTransactions
    (
     IsFinalized
    )
    WITH (DROP_EXISTING = ON)
    ON PS_TransactionDate(TransactionDate);';
    EXECUTE (@SQL);
END;
```

```
        COMMIT;

        PRINT N'Partitioning successfully enabled';
    END;
END;
GO
```

Depends On 1

Application

Used By 1

Application.Configuration_ConfigureForEnterpriseEdition

Application.Configuration_ApplyRowLevelSecurity

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Application.Configuration_ApplyRowLevelSecurity
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @SQL nvarchar(max);

    BEGIN TRY;

        SET @SQL = N'DROP SECURITY POLICY IF EXISTS [Application].FilterCustomersBySalesTerritoryRole;';
        EXECUTE (@SQL);

        SET @SQL = N'DROP FUNCTION IF EXISTS [Application].DetermineCustomerAccess;';
        EXECUTE (@SQL);

        SET @SQL = N'
        CREATE FUNCTION [Application].DetermineCustomerAccess(@CityID int)
        RETURNS TABLE
        WITH SCHEMABINDING
        AS
        RETURN (SELECT 1 AS AccessResult
         WHERE IS_ROLEMEMBER(N''db_owner'') <> 0
         OR IS_ROLEMEMBER((SELECT sp.SalesTerritory
         FROM [Application].Cities AS c
         INNER JOIN [Application].StateProvinces AS sp
         ON c.StateProvinceID = sp.StateProvinceID
         WHERE c.CityID = @CityID) + N'' Sales'') <> 0
         OR (ORIGINAL_LOGIN() = N''Website''
         AND EXISTS (SELECT 1
         FROM [Application].Cities AS c
         INNER JOIN [Application].StateProvinces AS sp
         ON c.StateProvinceID = sp.StateProvinceID
         WHERE c.CityID = @CityID
         AND sp.SalesTerritory = SESSION_CONTEXT(N''SalesTerritory''))));';
        EXECUTE (@SQL);

        SET @SQL = N'
        CREATE SECURITY POLICY [Application].FilterCustomersBySalesTerritoryRole
        ADD FILTER PREDICATE [Application].DetermineCustomerAccess(DeliveryCityID)
        ON Sales.Customers,
        ADD BLOCK PREDICATE [Application].DetermineCustomerAccess(DeliveryCityID)
        ON Sales.Customers AFTER UPDATE;';
        EXECUTE (@SQL);
```

```
        PRINT N'Successfully applied row level security';
    END TRY
    BEGIN CATCH
        PRINT N'Unable to apply row level security';
  PRINT ERROR_MESSAGE();
        THROW 51000, N'Unable to apply row level security', 1;
    END CATCH;
END;
GO
```

Depends On 1

Application

Used By 1

DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad

Application.Configuration_ConfigureForEnterpriseEdition

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Application.Configuration_ConfigureForEnterpriseEdition
AS
BEGIN

    EXEC [Application].[Configuration_ApplyColumnstoreIndexing];

    IF SERVERPROPERTY(N'IsFullTextInstalled') = 0
    BEGIN
        EXEC [Application].[Configuration_ApplyFullTextIndexing];
    END;

    EXEC [Application].[Configuration_EnableInMemory];

    EXEC [Application].[Configuration_ApplyPartitioning];

END;
GO
```

Depends On 5

- Application
- Application.Configuration_ApplyColumnstoreIndexing
- Application.Configuration_ApplyFullTextIndexing
- Application.Configuration_EnableInMemory
- Application.Configuration_ApplyPartitioning

Used By

No items found

Application.Configuration_EnableInMemory

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Application.Configuration_EnableInMemory
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    IF SERVERPROPERTY(N'IsXTPSupported') = 0
    BEGIN
        PRINT N'Warning: In-memory tables cannot be created on this edition.';
    END ELSE BEGIN -- if in-memory can be created

  DECLARE @SQL nvarchar(max) = N'';

  BEGIN TRY
   IF CAST(SERVERPROPERTY(N'EngineEdition') AS int) <> 5   -- Not an Azure SQL DB
   BEGIN
    DECLARE @SQLDataFolder nvarchar(max) = (SELECT SUBSTRING(df.physical_name, 1, CHARINDEX(N'
    WideWorldImporters.mdf', df.physical_name, 1)
    - 1)
                                            FROM sys.database_files AS df
                                            WHERE df.file_id = 1);
    DECLARE @MemoryOptimizedFilegroupFolder nvarchar(max) = @SQLDataFolder + N'WideWorldImporters_InMemory_Data_
    1';

    IF NOT EXISTS (SELECT 1 FROM sys.filegroups WHERE name = N'WWI_InMemory_Data')
    BEGIN
        SET @SQL = N'
        ALTER DATABASE WideWorldImporters
        ADD FILEGROUP WWI_InMemory_Data CONTAINS MEMORY_OPTIMIZED_DATA;';
     EXECUTE (@SQL);

     SET @SQL = N'
     ALTER DATABASE WideWorldImporters
     ADD FILE (name = N''WWI_InMemory_Data_1'', filename = '''
                 + @MemoryOptimizedFilegroupFolder + N''')
                 TO FILEGROUP WWI_InMemory_Data;';
     EXECUTE (@SQL);

     SET @SQL = N'
     ALTER DATABASE WideWorldImporters
     SET MEMORY_OPTIMIZED_ELEVATE_TO_SNAPSHOT = ON;';
     EXECUTE (@SQL);
    END;
            END;
```

```
IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = N'ColdRoomTemperatures' AND is_memory_optimized <> 0)
BEGIN

    SET @SQL = N'
    ALTER TABLE Warehouse.ColdRoomTemperatures SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE Warehouse.ColdRoomTemperatures DROP PERIOD FOR SYSTEM_TIME;
    ALTER TABLE Warehouse.ColdRoomTemperatures DROP CONSTRAINT PK_Warehouse_ColdRoomTemperatures;';
    EXECUTE (@SQL);

    SET @SQL = N'
    EXEC dbo.sp_rename @objname = N''Warehouse.ColdRoomTemperatures'',
     @newname = N''ColdRoomTemperatures_Backup'',
     @objtype = N''OBJECT'';';
    EXECUTE (@SQL);

    SET @SQL = N'
    CREATE TABLE Warehouse.ColdRoomTemperatures
    (
     ColdRoomTemperatureID bigint IDENTITY(1,1) NOT NULL
     PRIMARY KEY NONCLUSTERED,
     ColdRoomSensorNumber int NOT NULL,
     RecordedWhen datetime2(7) NOT NULL,
     Temperature decimal(10, 2) NOT NULL,
     ValidFrom datetime2(7) NOT NULL,
     ValidTo datetime2(7) NOT NULL
    ) WITH (MEMORY_OPTIMIZED = ON ,DURABILITY = SCHEMA_AND_DATA);';
    EXECUTE (@SQL);

    SET @SQL = N'
    SET IDENTITY_INSERT Warehouse.ColdRoomTemperatures ON;

    INSERT Warehouse.ColdRoomTemperatures (ColdRoomTemperatureID, ColdRoomSensorNumber, RecordedWhen,
    Temperature,   ValidFrom, ValidTo)
    SELECT ColdRoomTemperatureID, ColdRoomSensorNumber, RecordedWhen, Temperature, ValidFrom, ValidTo
    FROM Warehouse.ColdRoomTemperatures_Backup;

    SET IDENTITY_INSERT Warehouse.ColdRoomTemperatures OFF;';
    EXECUTE (@SQL);

    SET @SQL = N'DROP TABLE Warehouse.ColdRoomTemperatures_Backup;';
    EXECUTE (@SQL);

    SET @SQL = N'
    ALTER TABLE Warehouse.ColdRoomTemperatures
    ADD PERIOD FOR SYSTEM_TIME(ValidFrom, ValidTo);';
    EXECUTE (@SQL);

    SET @SQL = N'
    ALTER TABLE Warehouse.ColdRoomTemperatures
    SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = Warehouse.ColdRoomTemperatures_Archive, DATA_
    CONSISTENCY_CHECK = ON));';
    EXECUTE (@SQL);

END; -- of if we need to move ColdRoomTemperatures

IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = N'VehicleTemperatures' AND is_memory_optimized <> 0)
BEGIN

    SET @SQL = N'
    ALTER TABLE Warehouse.VehicleTemperatures DROP CONSTRAINT PK_Warehouse_VehicleTemperatures;';
    EXECUTE (@SQL);

    SET @SQL = N'
    EXEC dbo.sp_rename @objname = N''Warehouse.VehicleTemperatures'',
     @newname = N''VehicleTemperatures_Backup'',
     @objtype = N''OBJECT'';';
    EXECUTE (@SQL);

    SET @SQL = N'
    CREATE TABLE Warehouse.VehicleTemperatures
    (
     VehicleTemperatureID bigint IDENTITY(1,1) NOT NULL
     PRIMARY KEY NONCLUSTERED,
     VehicleRegistration nvarchar(20) COLLATE Latin1_General_CI_AS NOT NULL,
     ChillerSensorNumber int NOT NULL,
     RecordedWhen datetime2(7) NOT NULL,
     Temperature decimal(10, 2) NOT NULL,
     FullSensorData nvarchar(1000) COLLATE Latin1_General_CI_AS NULL,
     IsCompressed bit NOT NULL,
```

```
                CompressedSensorData varbinary(max) NULL
                ) WITH (MEMORY_OPTIMIZED = ON , DURABILITY = SCHEMA_AND_DATA);';
                EXECUTE (@SQL);

                SET @SQL = N'
                SET IDENTITY_INSERT Warehouse.VehicleTemperatures ON;

                INSERT Warehouse.VehicleTemperatures
                (VehicleTemperatureID, VehicleRegistration, ChillerSensorNumber, RecordedWhen, Temperature,
                FullSensorData, IsCompressed, CompressedSensorData)  SELECT VehicleTemperatureID,
                VehicleRegistration, ChillerSensorNumber, RecordedWhen, Temperature, FullSensorData, IsCompressed,
                CompressedSensorData FROM Warehouse.VehicleTemperatures_Backup;

                SET IDENTITY_INSERT Warehouse.VehicleTemperatures OFF;';
                EXECUTE (@SQL);

                SET @SQL = N'DROP TABLE Warehouse.VehicleTemperatures_Backup;';
                EXECUTE (@SQL);

            END; -- of if we need to move VehicleTemperatures

            -- Drop the procedures that are used by the table types

            SET @SQL = N'DROP PROCEDURE IF EXISTS Website.InvoiceCustomerOrders;';
            EXECUTE (@SQL);
            SET @SQL = N'DROP PROCEDURE IF EXISTS Website.InsertCustomerOrders;';
            EXECUTE (@SQL);
SET @SQL = N'DROP PROCEDURE IF EXISTS Website.RecordColdRoomTemperatures;';
EXECUTE (@SQL);

            -- Drop the table types

            SET @SQL = N'DROP TYPE IF EXISTS Website.OrderIDList;';
            EXECUTE (@SQL);
            SET @SQL = N'DROP TYPE IF EXISTS Website.OrderLineList;';
            EXECUTE (@SQL);
            SET @SQL = N'DROP TYPE IF EXISTS Website.OrderList;';
            EXECUTE (@SQL);
            SET @SQL = N'DROP TYPE IF EXISTS Website.SensorDataList;';
            EXECUTE (@SQL);

            -- Create the new table types

            SET @SQL = N'
            CREATE TYPE Website.OrderIDList AS TABLE
            (
             OrderID int PRIMARY KEY NONCLUSTERED
            )
            WITH (MEMORY_OPTIMIZED = ON);';
            EXECUTE (@SQL);

            SET @SQL = N'
            CREATE TYPE Website.OrderList AS TABLE
            (
             OrderReference int PRIMARY KEY NONCLUSTERED,
             CustomerID int,
             ContactPersonID int,
             ExpectedDeliveryDate date,
             CustomerPurchaseOrderNumber nvarchar(20),
             IsUndersupplyBackordered bit,
             Comments nvarchar(max),
             DeliveryInstructions nvarchar(max)
            )
            WITH (MEMORY_OPTIMIZED = ON);';
            EXECUTE (@SQL);

            SET @SQL = N'
            CREATE TYPE Website.OrderLineList AS TABLE
            (
             OrderReference int,
             StockItemID int,
             [Description] nvarchar(100),
             Quantity int,
             INDEX IX_Website_OrderLineList NONCLUSTERED (OrderReference)
            )
            WITH (MEMORY_OPTIMIZED = ON);';
            EXECUTE (@SQL);
```

```
SET @SQL = N'
CREATE TYPE Website.SensorDataList AS TABLE
(
 SensorDataListID int IDENTITY(1,1) PRIMARY KEY NONCLUSTERED,
 ColdRoomSensorNumber int,
 RecordedWhen datetime2(7),
 Temperature decimal(18,2)
)
WITH (MEMORY_OPTIMIZED = ON);';
EXECUTE (@SQL);

SET @SQL = N'
CREATE PROCEDURE Website.InvoiceCustomerOrders
@OrdersToInvoice Website.OrderIDList READONLY,
@PackedByPersonID int,
@InvoicedByPersonID int
WITH EXECUTE AS OWNER
AS
BEGIN
 SET NOCOUNT ON;
 SET XACT_ABORT ON;

 DECLARE @InvoicesToGenerate TABLE
 (
 OrderID int PRIMARY KEY,
 InvoiceID int NOT NULL,
 TotalDryItems int NOT NULL,
 TotalChillerItems int NOT NULL
 );

 BEGIN TRY;

 -- Check that all orders exist, have been fully picked, and not already invoiced. Also allocate
new invoice numbers.  INSERT @InvoicesToGenerate (OrderID, InvoiceID, TotalDryItems,
TotalChillerItems)  SELECT oti.OrderID,
 NEXT VALUE FOR Sequences.InvoiceID,
 COALESCE((SELECT SUM(CASE WHEN si.IsChillerStock <> 0 THEN 0 ELSE 1 END)
 FROM Sales.OrderLines AS ol
 INNER JOIN Warehouse.StockItems AS si
 ON ol.StockItemID = si.StockItemID
 WHERE ol.OrderID = oti.OrderID), 0),
 COALESCE((SELECT SUM(CASE WHEN si.IsChillerStock <> 0 THEN 1 ELSE 0 END)
 FROM Sales.OrderLines AS ol
 INNER JOIN Warehouse.StockItems AS si
 ON ol.StockItemID = si.StockItemID
 WHERE ol.OrderID = oti.OrderID), 0)
 FROM @OrdersToInvoice AS oti
 INNER JOIN Sales.Orders AS o
 ON oti.OrderID = o.OrderID
 WHERE NOT EXISTS (SELECT 1 FROM Sales.Invoices AS i
 WHERE i.OrderID = oti.OrderID)
 AND o.PickingCompletedWhen IS NOT NULL;

 IF EXISTS (SELECT 1 FROM @OrdersToInvoice AS oti WHERE NOT EXISTS (SELECT 1 FROM @
InvoicesToGenerate AS itg WHERE itg.OrderID = oti.OrderID))  BEGIN
 PRINT N''At least one order ID either does not exist, is not picked, or is already invoiced'
';  THROW 51000, N''At least one orderID either does not exist, is not picked, or is already
invoiced'', 1;  END;

 BEGIN TRAN;

 INSERT Sales.Invoices
 (InvoiceID, CustomerID, BillToCustomerID, OrderID, DeliveryMethodID, ContactPersonID,
AccountsPersonID,   SalespersonPersonID, PackedByPersonID, InvoiceDate,
CustomerPurchaseOrderNumber,   IsCreditNote, CreditNoteReason, Comments,
DeliveryInstructions, InternalComments,   TotalDryItems, TotalChillerItems, DeliveryRun,
RunPosition,   ReturnedDeliveryData,
 LastEditedBy, LastEditedWhen)
 SELECT itg.InvoiceID, c.CustomerID, c.BillToCustomerID, itg.OrderID, c.DeliveryMethodID, o.
ContactPersonID, btc.PrimaryContactPersonID,  o.SalespersonPersonID, @PackedByPersonID,
SYSDATETIME(), o.CustomerPurchaseOrderNumber,  0, NULL, NULL, c.DeliveryAddressLine1 + N''
, '' + c.DeliveryAddressLine2, NULL,  itg.TotalDryItems, itg.TotalChillerItems, c.
DeliveryRun, c.RunPosition,   JSON_MODIFY(N''{"Events": []}'', N''append $.Events'',
 JSON_MODIFY(JSON_MODIFY(JSON_MODIFY(N''{ }'', N''$.Event'', N''Ready for collection''
),   N''$.EventTime'', CONVERT(nvarchar(20), SYSDATETIME(), 126)),
 N''$.ConNote'', N''EAN-125-'' + CAST(itg.InvoiceID + 1050 AS nvarchar(20)))),
   @InvoicedByPersonID, SYSDATETIME()
 FROM @InvoicesToGenerate AS itg
 INNER JOIN Sales.Orders AS o
```

```
 ON itg.OrderID = o.OrderID
 INNER JOIN Sales.Customers AS c
 ON o.CustomerID = c.CustomerID
 INNER JOIN Sales.Customers AS btc
 ON btc.CustomerID = c.BillToCustomerID;

 INSERT Sales.InvoiceLines
 (InvoiceID, StockItemID, [Description], PackageTypeID,
 Quantity, UnitPrice, TaxRate, TaxAmount, LineProfit, ExtendedPrice,
 LastEditedBy, LastEditedWhen)
 SELECT itg.InvoiceID, ol.StockItemID, ol.[Description], ol.PackageTypeID,
 ol.PickedQuantity, ol.UnitPrice, ol.TaxRate,
 ROUND(ol.PickedQuantity * ol.UnitPrice * ol.TaxRate / 100.0, 2),
 ROUND(ol.PickedQuantity * (ol.UnitPrice - sih.LastCostPrice), 2),
 ROUND(ol.PickedQuantity * ol.UnitPrice, 2)
 + ROUND(ol.PickedQuantity * ol.UnitPrice * ol.TaxRate / 100.0, 2),
 @InvoicedByPersonID, SYSDATETIME()
 FROM @InvoicesToGenerate AS itg
 INNER JOIN Sales.OrderLines AS ol
 ON itg.OrderID = ol.OrderID
 INNER JOIN Warehouse.StockItems AS si
 ON ol.StockItemID = si.StockItemID
 INNER JOIN Warehouse.StockItemHoldings AS sih
 ON si.StockItemID = sih.StockItemID
 ORDER BY ol.OrderID, ol.OrderLineID;

 INSERT Warehouse.StockItemTransactions
 (StockItemID, TransactionTypeID, CustomerID, InvoiceID, SupplierID, PurchaseOrderID,
 TransactionOccurredWhen, Quantity, LastEditedBy, LastEditedWhen)
 SELECT il.StockItemID, (SELECT TransactionTypeID FROM [Application].TransactionTypes WHERE
TransactionTypeName = N''Stock Issue''),  i.CustomerID, i.InvoiceID, NULL, NULL,
 SYSDATETIME(), 0 - il.Quantity, @InvoicedByPersonID, SYSDATETIME()
 FROM @InvoicesToGenerate AS itg
 INNER JOIN Sales.InvoiceLines AS il
 ON itg.InvoiceID = il.InvoiceID
 INNER JOIN Sales.Invoices AS i
 ON il.InvoiceID = i.InvoiceID
 ORDER BY il.InvoiceID, il.InvoiceLineID;

 WITH StockItemTotals
 AS
 (
 SELECT il.StockItemID, SUM(il.Quantity) AS TotalQuantity
 FROM Sales.InvoiceLines aS il
 WHERE il.InvoiceID IN (SELECT InvoiceID FROM @InvoicesToGenerate)
 GROUP BY il.StockItemID
 )
 UPDATE sih
 SET sih.QuantityOnHand -= sit.TotalQuantity,
 sih.LastEditedBy = @InvoicedByPersonID,
 sih.LastEditedWhen = SYSDATETIME()
 FROM Warehouse.StockItemHoldings AS sih
 INNER JOIN StockItemTotals AS sit
 ON sih.StockItemID = sit.StockItemID;

 INSERT Sales.CustomerTransactions
 (CustomerID, TransactionTypeID, InvoiceID, PaymentMethodID,
 TransactionDate, AmountExcludingTax, TaxAmount, TransactionAmount,
 OutstandingBalance, FinalizationDate, LastEditedBy, LastEditedWhen)
 SELECT i.BillToCustomerID,
 (SELECT TransactionTypeID FROM [Application].TransactionTypes WHERE TransactionTypeName
= N''Customer Invoice''),  itg.InvoiceID,
 NULL,
 SYSDATETIME(),
 (SELECT SUM(il.ExtendedPrice - il.TaxAmount) FROM Sales.InvoiceLines AS il WHERE il.
InvoiceID = itg.InvoiceID),  (SELECT SUM(il.TaxAmount) FROM Sales.InvoiceLines AS il
WHERE il.InvoiceID = itg.InvoiceID),  (SELECT SUM(il.ExtendedPrice) FROM Sales.
InvoiceLines AS il WHERE il.InvoiceID = itg.InvoiceID),  (SELECT SUM(il.ExtendedPrice)
FROM Sales.InvoiceLines AS il WHERE il.InvoiceID = itg.InvoiceID),  NULL,
 @InvoicedByPersonID,
 SYSDATETIME()
 FROM @InvoicesToGenerate AS itg
 INNER JOIN Sales.Invoices AS i
 ON itg.InvoiceID = i.InvoiceID;

 COMMIT;

 END TRY
```

```
            BEGIN CATCH
            IF XACT_STATE() <> 0 ROLLBACK;
            PRINT N''Unable to invoice these orders'';
            THROW;
            RETURN -1;
            END CATCH;

            RETURN 0;
           END;';
           EXECUTE (@SQL);

SET @SQL = N'
CREATE PROCEDURE Website.RecordColdRoomTemperatures
@SensorReadings Website.SensorDataList READONLY
WITH NATIVE_COMPILATION, SCHEMABINDING, EXECUTE AS OWNER
AS
BEGIN ATOMIC WITH
(
 TRANSACTION ISOLATION LEVEL = SNAPSHOT,
 LANGUAGE = N''English''
)
 BEGIN TRY

 DECLARE @NumberOfReadings int = (SELECT MAX(SensorDataListID) FROM @SensorReadings);
 DECLARE @Counter int = (SELECT MIN(SensorDataListID) FROM @SensorReadings);

 DECLARE @ColdRoomSensorNumber int;
 DECLARE @RecordedWhen datetime2(7);
 DECLARE @Temperature decimal(18,2);

 -- note that we cannot use a merge here because multiple readings might exist for each sensor

 WHILE @Counter <= @NumberOfReadings
 BEGIN
 SELECT @ColdRoomSensorNumber = ColdRoomSensorNumber,
 @RecordedWhen = RecordedWhen,
 @Temperature = Temperature
 FROM @SensorReadings
 WHERE SensorDataListID = @Counter;

 UPDATE Warehouse.ColdRoomTemperatures
 SET RecordedWhen = @RecordedWhen,
 Temperature = @Temperature
 WHERE ColdRoomSensorNumber = @ColdRoomSensorNumber;

 IF @@ROWCOUNT = 0
 BEGIN
 INSERT Warehouse.ColdRoomTemperatures
 (ColdRoomSensorNumber, RecordedWhen, Temperature)
 VALUES (@ColdRoomSensorNumber, @RecordedWhen, @Temperature);
 END;

 SET @Counter += 1;
 END;

 END TRY
 BEGIN CATCH
 THROW 51000, N''Unable to apply the sensor data'', 2;

 RETURN 1;
 END CATCH;
END;';
EXECUTE (@SQL);

           SET @SQL = N'
           CREATE PROCEDURE Website.InsertCustomerOrders
           @Orders Website.OrderList READONLY,
           @OrderLines Website.OrderLineList READONLY,
           @OrdersCreatedByPersonID int,
           @SalespersonPersonID int
           WITH EXECUTE AS OWNER
           AS
           BEGIN
            SET NOCOUNT ON;
            SET XACT_ABORT ON;

            DECLARE @OrdersToGenerate AS TABLE
            (
            OrderReference int PRIMARY KEY, -- reference from the application
```

```
            OrderID int
            );

            -- allocate the new order numbers

            INSERT @OrdersToGenerate (OrderReference, OrderID)
            SELECT OrderReference, NEXT VALUE FOR Sequences.OrderID
            FROM @Orders;

            BEGIN TRY

            BEGIN TRAN;

            INSERT Sales.Orders
            (OrderID, CustomerID, SalespersonPersonID, PickedByPersonID, ContactPersonID,
           BackorderOrderID, OrderDate,   ExpectedDeliveryDate, CustomerPurchaseOrderNumber,
           IsUndersupplyBackordered, Comments, DeliveryInstructions, InternalComments,
           PickingCompletedWhen, LastEditedBy, LastEditedWhen)  SELECT otg.OrderID, o.CustomerID, @
           SalespersonPersonID, NULL, o.ContactPersonID, NULL, SYSDATETIME(),  o.
           ExpectedDeliveryDate, o.CustomerPurchaseOrderNumber, o.IsUndersupplyBackordered, o.Comments, o.
           DeliveryInstructions, NULL,  NULL, @OrdersCreatedByPersonID, SYSDATETIME()
            FROM @OrdersToGenerate AS otg
            INNER JOIN @Orders AS o
            ON otg.OrderReference = o.OrderReference;

            INSERT Sales.OrderLines
            (OrderID, StockItemID, [Description], PackageTypeID, Quantity, UnitPrice,
            TaxRate, PickedQuantity, PickingCompletedWhen, LastEditedBy, LastEditedWhen)
            SELECT otg.OrderID, ol.StockItemID, ol.[Description], si.UnitPackageID, ol.Quantity,
            Website.CalculateCustomerPrice(o.CustomerID, ol.StockItemID, SYSDATETIME()),
            si.TaxRate, 0, NULL, @OrdersCreatedByPersonID, SYSDATETIME()
            FROM @OrdersToGenerate AS otg
            INNER JOIN @OrderLines AS ol
            ON otg.OrderReference = ol.OrderReference
            INNER JOIN @Orders AS o
            ON ol.OrderReference = o.OrderReference
            INNER JOIN Warehouse.StockItems AS si
            ON ol.StockItemID = si.StockItemID;

            COMMIT;

            END TRY
            BEGIN CATCH
            IF XACT_STATE() <> 0 ROLLBACK;
            PRINT N''Unable to create the customer orders.'';
            THROW;
            RETURN -1;
            END CATCH;

            RETURN 0;
           END;';
           EXECUTE (@SQL);

        END TRY
        BEGIN CATCH
            PRINT N'Unable to apply in-memory tables';
            THROW;
        END CATCH;
    END; -- of in-memory is allowed
END;
GO
```

Depends On 1

Application

Used By 1

Application.Configuration_ConfigureForEnterpriseEdition

Application.Configuration_RemoveAuditing

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Application.Configuration_RemoveAuditing
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @AreDatabaseAuditSpecificationsSupported bit = 0;
    DECLARE @SQL nvarchar(max);

    -- TODO !! - currently no separate test for audit
    -- but same editions with XTP support database audit specs
    IF SERVERPROPERTY(N'IsXTPSupported') <> 0 SET @AreDatabaseAuditSpecificationsSupported = 1;

    BEGIN TRY;

        IF @AreDatabaseAuditSpecificationsSupported <> 0
        BEGIN
            IF EXISTS (SELECT 1 FROM sys.database_audit_specifications WHERE name = N'WWI_
            DatabaseAuditSpecification')
            BEGIN
                SET @SQL = N'
                DROP DATABASE AUDIT SPECIFICATION WWI_DatabaseAuditSpecification;';
                EXECUTE (@SQL);
            END;
        END;

        IF EXISTS (SELECT 1 FROM sys.server_audit_specifications WHERE name = N'WWI_
        ServerAuditSpecification')
        BEGIN
            SET @SQL = N'
            USE master;

            DROP SERVER AUDIT SPECIFICATION WWI_ServerAuditSpecification;';
            EXECUTE (@SQL);
        END;

        IF EXISTS (SELECT 1 FROM sys.server_audits WHERE name = N'WWI_Audit')
        BEGIN
            SET @SQL = N'
            USE master;

            DROP SERVER AUDIT [WWI_Audit];';
            EXECUTE (@SQL);

        END;
```

```
    END TRY
    BEGIN CATCH
        PRINT N'Unable to remove audit';
        THROW;
    END CATCH;
END;
GO
```

Depends On 1

Application

Used By

No items found

Application.Configuration_RemoveRowLevelSecurity

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Application.Configuration_RemoveRowLevelSecurity
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @SQL nvarchar(max);

    BEGIN TRY;

        SET @SQL = N'DROP SECURITY POLICY IF EXISTS [Application].FilterCustomersBySalesTerritoryRole;';
        EXECUTE (@SQL);

        SET @SQL = N'DROP FUNCTION IF EXISTS [Application].DetermineCustomerAccess;';
        EXECUTE (@SQL);

        PRINT N'Successfully removed row level security';
    END TRY
    BEGIN CATCH
        PRINT N'Unable to remove row level security';
        THROW 51000, N'Unable to remove row level security', 1;
    END CATCH;
END;
GO
```

Depends On 1

Application

Used By 1

DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad

Application.CreateRoleIfNonexistent

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@RoleName	sysname	256	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Application.CreateRoleIfNonexistent
@RoleName sysname
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    IF NOT EXISTS (SELECT 1 FROM sys.database_principals WHERE name = @RoleName AND type = N'R')
    BEGIN
        BEGIN TRY

            DECLARE @SQL nvarchar(max) = N'CREATE ROLE ' + QUOTENAME(@RoleName) + N';'
            EXECUTE (@SQL);

            PRINT N'Role ' + @RoleName + N' created';

        END TRY
        BEGIN CATCH
            PRINT N'Unable to create role ' + @RoleName;
            THROW;
        END CATCH;
    END;
END;
GO
```

Depends On 1

Application

Used By

No items found

DataLoadSimulation.Configuration_ApplyDataLoadSimulationProcedures

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE DataLoadSimulation.Configuration_ApplyDataLoadSimulationProcedures
WITH EXECUTE AS OWNER
AS
BEGIN
 SET NOCOUNT ON;
 SET XACT_ABORT ON;

 EXEC DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad;

 DECLARE @SQL nvarchar(max);

 IF NOT EXISTS (SELECT 1 FROM sys.objects WHERE name = N'GetAreaCode'
                                          AND type = N'FN'
           AND SCHEMA_NAME(schema_id) = N'DataLoadSimulation')
 BEGIN
  SET @SQL = N'
  CREATE FUNCTION DataLoadSimulation.GetAreaCode
  (
   @StateProvinceCode nvarchar(2)
  )
  RETURNS INT
  WITH EXECUTE AS OWNER
  AS
  BEGIN
   DECLARE @AreaCode int;

   WITH AreaCodes
   AS
   (
   SELECT StateProvinceCode, AreaCode
   FROM
   (VALUES (''NJ'', 201),
   (''DC'', 202),
   (''CT'', 203),
   (''MB'', 204),
   (''AL'', 205),
   (''WA'', 206),
   (''ME'', 207),
   (''ID'', 208),
   (''CA'', 209),
   (''TX'', 210),
   (''NY'', 212),
   (''CA'', 213),
```

```
    (''TX'', 214),
    (''PA'', 215),
    (''OH'', 216),
    (''IL'', 217),
    (''MN'', 218),
    (''IN'', 219),
    (''OH'', 220),
    (''IL'', 224),
    (''LA'', 225),
    (''ON'', 226),
    (''MS'', 228),
    (''GA'', 229),
    (''MI'', 231),
    (''OH'', 234),
    (''BC'', 236),
    (''FL'', 239),
    (''MD'', 240),
    (''MI'', 248),
    (''BC'', 250),
    (''AL'', 251),
    (''NC'', 252),
    (''WA'', 253),
    (''TX'', 254),
    (''AL'', 256),
    (''IN'', 260),
    (''WI'', 262),
    (''PA'', 267),
    (''MI'', 269),
    (''KY'', 270),
    (''PA'', 272),
    (''VA'', 276),
    (''MI'', 278),
    (''TX'', 281),
    (''OH'', 283),
    (''ON'', 289),
    (''MD'', 301),
    (''DE'', 302),
    (''CO'', 303),
    (''WV'', 304),
    (''FL'', 305),
    (''SK'', 306),
    (''WY'', 307),
    (''NE'', 308),
    (''IL'', 309),
    (''CA'', 310),
    (''IL'', 312),
    (''MI'', 313),
    (''MO'', 314),
    (''NY'', 315),
    (''KS'', 316),
    (''IN'', 317),
    (''LA'', 318),
    (''IA'', 319),
    (''MN'', 320),
    (''FL'', 321),
    (''CA'', 323),
    (''TX'', 325),
    (''OH'', 330),
    (''IL'', 331),
    (''AL'', 334),
    (''NC'', 336),
    (''LA'', 337),
    (''MA'', 339),
    (''VI'', 340),
    (''CA'', 341),
    (''ON'', 343),
    (''NY'', 347),
    (''MA'', 351),
    (''FL'', 352),
    (''WA'', 360),
    (''TX'', 361),
    (''ON'', 365),
    (''CA'', 369),
    (''OH'', 380),
    (''UT'', 385),
    (''FL'', 386),
    (''RI'', 401),
    (''NE'', 402),
    (''AB'', 403),
```

```
    (''GA'', 404),
    (''OK'', 405),
    (''MT'', 406),
    (''FL'', 407),
    (''CA'', 408),
    (''TX'', 409),
    (''MD'', 410),
    (''PA'', 412),
    (''MA'', 413),
    (''WI'', 414),
    (''CA'', 415),
    (''ON'', 416),
    (''MO'', 417),
    (''QC'', 418),
    (''OH'', 419),
    (''TN'', 423),
    (''CA'', 424),
    (''WA'', 425),
    (''TX'', 430),
    (''MB'', 431),
    (''TX'', 432),
    (''VA'', 434),
    (''UT'', 435),
    (''ON'', 437),
    (''QC'', 438),
    (''OH'', 440),
    (''CA'', 442),
    (''MD'', 443),
    (''QC'', 450),
    (''OR'', 458),
    (''IL'', 464),
    (''TX'', 469),
    (''GA'', 470),
    (''CT'', 475),
    (''GA'', 478),
    (''AR'', 479),
    (''AZ'', 480),
    (''QC'', 481),
    (''PA'', 484),
    (''AR'', 501),
    (''KY'', 502),
    (''OR'', 503),
    (''LA'', 504),
    (''NM'', 505),
    (''NB'', 506),
    (''MN'', 507),
    (''MA'', 508),
    (''WA'', 509),
    (''CA'', 510),
    (''TX'', 512),
    (''OH'', 513),
    (''QC'', 514),
    (''IA'', 515),
    (''NY'', 516),
    (''MI'', 517),
    (''NY'', 518),
    (''ON'', 519),
    (''AZ'', 520),
    (''CA'', 530),
    (''OK'', 539),
    (''VA'', 540),
    (''OR'', 541),
    (''ON'', 548),
    (''NJ'', 551),
    (''MO'', 557),
    (''CA'', 559),
    (''FL'', 561),
    (''CA'', 562),
    (''IA'', 563),
    (''WA'', 564),
    (''OH'', 567),
    (''PA'', 570),
    (''VA'', 571),
    (''MO'', 573),
    (''IN'', 574),
    (''NM'', 575),
    (''QC'', 579),
    (''OK'', 580),
```

```
    (''NY'', 585),
    (''MI'', 586),
    (''AB'', 587),
    (''MS'', 601),
    (''AZ'', 602),
    (''NH'', 603),
    (''BC'', 604),
    (''SD'', 605),
    (''KY'', 606),
    (''NY'', 607),
    (''WI'', 608),
    (''NJ'', 609),
    (''PA'', 610),
    (''MN'', 612),
    (''ON'', 613),
    (''OH'', 614),
    (''TN'', 615),
    (''MI'', 616),
    (''MA'', 617),
    (''IL'', 618),
    (''CA'', 619),
    (''KS'', 620),
    (''AZ'', 623),
    (''CA'', 626),
    (''CA'', 627),
    (''CA'', 628),
    (''TN'', 629),
    (''IL'', 630),
    (''NY'', 631),
    (''MO'', 636),
    (''SK'', 639),
    (''IA'', 641),
    (''NY'', 646),
    (''ON'', 647),
    (''CA'', 650),
    (''MN'', 651),
    (''CA'', 657),
    (''MO'', 660),
    (''CA'', 661),
    (''MS'', 662),
    (''CA'', 669),
    (''MP'', 670),
    (''GU'', 671),
    (''GA'', 678),
    (''MI'', 679),
    (''WV'', 681),
    (''TX'', 682),
    (''FL'', 689),
    (''ND'', 701),
    (''NV'', 702),
    (''VA'', 703),
    (''NC'', 704),
    (''ON'', 705),
    (''GA'', 706),
    (''CA'', 707),
    (''IL'', 708),
    (''NL'', 709),
    (''IA'', 712),
    (''TX'', 713),
    (''CA'', 714),
    (''WI'', 715),
    (''NY'', 716),
    (''PA'', 717),
    (''NY'', 718),
    (''CO'', 719),
    (''CO'', 720),
    (''PA'', 724),
    (''NV'', 725),
    (''FL'', 727),
    (''TN'', 731),
    (''NJ'', 732),
    (''MI'', 734),
    (''TX'', 737),
    (''OH'', 740),
    (''CA'', 747),
    (''FL'', 754),
    (''VA'', 757),
    (''CA'', 760),
    (''GA'', 762),
```

```
    (''MN'', 763),
    (''CA'', 764),
    (''IN'', 765),
    (''MS'', 769),
    (''GA'', 770),
    (''FL'', 772),
    (''IL'', 773),
    (''MA'', 774),
    (''NV'', 775),
    (''BC'', 778),
    (''IL'', 779),
    (''AB'', 780),
    (''MA'', 781),
    (''NS'', 782),
    (''KS'', 785),
    (''FL'', 786),
    (''PR'', 787),
    (''UT'', 801),
    (''VT'', 802),
    (''SC'', 803),
    (''VA'', 804),
    (''CA'', 805),
    (''TX'', 806),
    (''ON'', 807),
    (''HI'', 808),
    (''MI'', 810),
    (''IN'', 812),
    (''FL'', 813),
    (''PA'', 814),
    (''IL'', 815),
    (''MO'', 816),
    (''TX'', 817),
    (''CA'', 818),
    (''QC'', 819),
    (''AB'', 825),
    (''NC'', 828),
    (''TX'', 830),
    (''CA'', 831),
    (''TX'', 832),
    (''PA'', 835),
    (''SC'', 843),
    (''NY'', 845),
    (''IL'', 847),
    (''NJ'', 848),
    (''FL'', 850),
    (''NJ'', 856),
    (''MA'', 857),
    (''CA'', 858),
    (''KY'', 859),
    (''CT'', 860),
    (''NJ'', 862),
    (''FL'', 863),
    (''SC'', 864),
    (''TN'', 865),
    (''YT'', 867),
    (''AR'', 870),
    (''IL'', 872),
    (''QC'', 873),
    (''PA'', 878),
    (''TN'', 901),
    (''NS'', 902),
    (''TX'', 903),
    (''FL'', 904),
    (''ON'', 905),
    (''MI'', 906),
    (''AK'', 907),
    (''NJ'', 908),
    (''CA'', 909),
    (''NC'', 910),
    (''GA'', 912),
    (''KS'', 913),
    (''NY'', 914),
    (''TX'', 915),
    (''CA'', 916),
    (''NY'', 917),
    (''OK'', 918),
    (''NC'', 919),
    (''WI'', 920),
```

```
  (''CA'', 925),
  (''FL'', 927),
  (''AZ'', 928),
  (''NY'', 929),
  (''TN'', 931),
  (''CA'', 935),
  (''TX'', 936),
  (''OH'', 937),
  (''PR'', 939),
  (''TX'', 940),
  (''FL'', 941),
  (''MI'', 947),
  (''CA'', 949),
  (''CA'', 951),
  (''MN'', 952),
  (''FL'', 954),
  (''TX'', 956),
  (''NM'', 957),
  (''CT'', 959),
  (''CO'', 970),
  (''OR'', 971),
  (''TX'', 972),
  (''NJ'', 973),
  (''MO'', 975),
  (''MA'', 978),
  (''TX'', 979),
  (''NC'', 980),
  (''NC'', 984),
  (''LA'', 985),
  (''MI'', 989)
  ) AS AreaCodes(StateProvinceCode, AreaCode)
  )
  SELECT TOP(1) @AreaCode = AreaCode FROM AreaCodes AS ac WHERE ac.StateProvinceCode = @StateProvinceCode;

  RETURN @AreaCode;
 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                            AND name = N'
                                            ActivateWebsiteLogins')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.ActivateWebsiteLogons
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 WITH EXECUTE AS OWNER
 AS
 BEGIN
  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  -- Approximately 1 in 8 days has a new website activation

  DECLARE @NumberOfLogonsToActivate int = CASE WHEN (RAND() * 8) <= 1 THEN 1 ELSE 0 END;

  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Activating '' + CAST(@NumberOfLogonsToActivate AS nvarchar(20)) + N'' logons'';
  END;

  DECLARE @Counter int = 0;
  DECLARE @PersonID int;
  DECLARE @EmailAddress nvarchar(256);
  DECLARE @HashedPassword varbinary(max);
  DECLARE @FullName nvarchar(50);
  DECLARE @UserPreferences nvarchar(max) = (SELECT UserPreferences FROM [Application].People WHERE PersonID = 1)
 ;
  WHILE @Counter < @NumberOfLogonsToActivate
  BEGIN
  SELECT TOP(1) @PersonID = PersonID,
  @EmailAddress = EmailAddress,
  @FullName = FullName
  FROM [Application].People
  WHERE IsPermittedToLogon = 0 AND PersonID <> 1
  ORDER BY NEWID();
```

```
  UPDATE [Application].People
  SET IsPermittedToLogon = 1,
  LogonName = @EmailAddress,
  HashedPassword = HASHBYTES(N''SHA2_256'', N''SQLRocks!00'' + @FullName),
  UserPreferences = @UserPreferences,
  [ValidFrom] = @StartingWhen
  WHERE PersonID = @PersonID;

  SET @Counter += 1;
  END;
 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                            AND name = N'AddCustomers')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.AddCustomers
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 WITH EXECUTE AS OWNER
 AS
 BEGIN

  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  -- add a customer one in 15 days average
  DECLARE @NumberOfCustomersToAdd int = (SELECT TOP(1) Quantity
  FROM (VALUES (0), (0), (0), (0), (0),
  (0), (0), (0), (0), (0),
  (0), (0), (0), (0), (0),
  (0), (0), (0), (0), (0),
  (0), (0), (0), (0), (1)) AS q(Quantity)
  ORDER BY NEWID());
  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Adding '' + CAST(@NumberOfCustomersToAdd AS nvarchar(20)) + N'' customers'';
  END;

  DECLARE @Counter int = 0;
  DECLARE @CityID int;
  DECLARE @CityName nvarchar(max);
  DECLARE @CityStateProvinceID int;
  DECLARE @CityStateProvinceCode nvarchar(5);
  DECLARE @AreaCode int;
  DECLARE @CustomerCategoryID int;


  DECLARE @CustomerID int;
  DECLARE @PrimaryContactFullName nvarchar(50);
  DECLARE @PrimaryContactPersonID int;
  DECLARE @PrimaryContactFirstName nvarchar(50);
  DECLARE @DeliveryMethodID int = (SELECT DeliveryMethodID FROM [Application].DeliveryMethods
  WHERE DeliveryMethodName = N''Delivery Van'');
  DECLARE @DeliveryAddressLine1 nvarchar(max);
  DECLARE @DeliveryAddressLine2 nvarchar(max);
  DECLARE @DeliveryPostalCode nvarchar(max);
  DECLARE @PostalAddressLine1 nvarchar(max);
  DECLARE @PostalAddressLine2 nvarchar(max);
  DECLARE @PostalPostalCode nvarchar(max);
  DECLARE @StreetSuffix nvarchar(max);
  DECLARE @CompanySuffix nvarchar(max);
  DECLARE @StorePrefix nvarchar(max);
  DECLARE @CreditLimit int;

  WHILE @Counter < @NumberOfCustomersToAdd
  BEGIN
  WITH NamesToUse
  AS
  (
  SELECT FirstName, LastName, FullName
  FROM
  (VALUES (''Mark'', ''Korjus'', ''Mark Korjus''),
```

```
(''Emil'', ''Bojin'', ''Emil Bojin''),
(''Hue'', ''Ton'', ''Hue Ton''),
(''Leonardo'', ''Jozic'', ''Leonardo Jozic''),
(''Ivana'', ''Hadrabova'', ''Ivana Hadrabova''),
(''Hakan'', ''Akbulut'', ''Hakan Akbulut''),
(''Jayanti'', ''Pandit'', ''Jayanti Pandit''),
(''Judit'', ''Gyenes'', ''Judit Gyenes''),
(''Coralie'', ''Monty'', ''Coralie Monty''),
(''Hai'', ''Banh'', ''Hai Banh''),
(''Manuel'', ''Jaramillo'', ''Manuel Jaramillo''),
(''Damodar'', ''Shenoy'', ''Damodar Shenoy''),
(''Jatindra'', ''Bandopadhyay'', ''Jatindra Bandopadhyay''),
(''Kanan'', ''Malakar'', ''Kanan Malakar''),
(''Miloslav'', ''Fisar'', ''Miloslav Fisar''),
(''Sylvie'', ''Laramee'', ''Sylvie Laramee''),
(''Rene'', ''Saucier'', ''Rene Saucier''),
(''Aruna'', ''Cheema'', ''Aruna Cheema''),
(''Jagdish'', ''Shergill'', ''Jagdish Shergill''),
(''Gopichand'', ''Dutta'', ''Gopichand Dutta''),
(''Adrian'', ''Lindqvist'', ''Adrian Lindqvist''),
(''Renata'', ''Michnova'', ''Renata Michnova''),
(''Gunnar'', ''Bjorklund'', ''Gunnar Bjorklund''),
(''Binoba'', ''Dey'', ''Binoba Dey''),
(''Stefan'', ''Selezeanu'', ''Stefan Selezeanu''),
(''Amolik'', ''Chakraborty'', ''Amolik Chakraborty''),
(''Mai'', ''Ton'', ''Mai Ton''),
(''Rajendra'', ''Mulye'', ''Rajendra Mulye''),
(''Sushila'', ''Baruah'', ''Sushila Baruah''),
(''Jibek'', ''Juniskyzy'', ''Jibek Juniskyzy''),
(''Rabindra'', ''Kaul'', ''Rabindra Kaul''),
(''Lucia'', ''Hinojosa'', ''Lucia Hinojosa''),
(''Maija'', ''Lukstina'', ''Maija Lukstina''),
(''Rajanikant'', ''Pandit'', ''Rajanikant Pandit''),
(''Nichole '', ''Deslauriers'', ''Nichole Deslauriers''),
(''Max'', ''Shand'', ''Max Shand''),
(''Farzana'', ''Abbasi'', ''Farzana Abbasi''),
(''Ekambar'', ''Bhuiyan'', ''Ekambar Bhuiyan''),
(''Dhanishta'', ''Pullela'', ''Dhanishta Pullela''),
(''Busarakham'', ''Kitjakarn'', ''Busarakham Kitjakarn''),
(''Manjunatha'', ''Karnik'', ''Manjunatha Karnik''),
(''Bianca'', ''Lack'', ''Bianca Lack''),
(''Viktoria'', ''Hudecova'', ''Viktoria Hudecova''),
(''Haarati'', ''Pendyala'', ''Haarati Pendyala''),
(''Bhagavateeprasaad'', ''Malladi'', ''Bhagavateeprasaad Malladi''),
(''Aykut'', ''ozkan'', ''Aykut ozkan''),
(''Essie'', ''Wimmer'', ''Essie Wimmer''),
(''Ivan'', ''Ignatyev'', ''Ivan Ignatyev''),
(''Sohail'', ''Shasthri'', ''Sohail Shasthri''),
(''Nils'', ''Kaulins'', ''Nils Kaulins''),
(''Suresh'', ''Singh'', ''Suresh Singh''),
(''Christian'', ''Couet'', ''Christian Couet''),
(''Tami'', ''Braggs'', ''Tami Braggs''),
(''Ian'', ''Olofsson'', ''Ian Olofsson''),
(''Juan'', ''Roy'', ''Juan Roy''),
(''Chandrani'', ''Dey'', ''Chandrani Dey''),
(''Esther'', ''Jobrani'', ''Esther Jobrani''),
(''Kristi'', ''Kuusik'', ''Kristi Kuusik''),
(''Abhaya'', ''Paruchuri'', ''Abhaya Paruchuri''),
(''Sung-Hwan'', ''Yoo'', ''Sung-Hwan Yoo''),
(''Amet'', ''Shergill'', ''Amet Shergill''),
(''Damla'', ''Yavuz'', ''Damla Yavuz''),
(''Naveen'', ''Scindia'', ''Naveen Scindia''),
(''Anurupa'', ''Mitra'', ''Anurupa Mitra''),
(''Raymond'', ''Beauchamp'', ''Raymond Beauchamp''),
(''Tara'', ''Kotadia'', ''Tara Kotadia''),
(''Arnost'', ''Hovorka'', ''Arnost Hovorka''),
(''Aive'', ''Petrov'', ''Aive Petrov''),
(''Tomo'', ''Vidovic'', ''Tomo Vidovic''),
(''Arundhati'', ''Majumdar'', ''Arundhati Majumdar''),
(''Marcela'', ''Mencikova'', ''Marcela Mencikova''),
(''Cosmina'', ''Leonte'', ''Cosmina Leonte''),
(''Linda'', ''Ohl'', ''Linda Ohl''),
(''Gulzar'', ''Sarkar'', ''Gulzar Sarkar''),
(''Carol'', ''Antonescu'', ''Carol Antonescu''),
(''Kyung-Soon'', ''Pak'', ''Kyung-Soon Pak''),
(''Jaroslav'', ''Fisar'', ''Jaroslav Fisar''),
(''Amrita'', ''Ganguly'', ''Amrita Ganguly''),
(''Śani'', ''Shasthri'', ''Śani Shasthri''),
(''Ivan'', ''Arenas'', ''Ivan Arenas''),
```

```
('''Miljan''', ''Stojanovic'', ''Miljan Stojanovic''),
('''Tereza''', ''Cermakova'', ''Tereza Cermakova''),
('''Harendra''', ''Sonkar'', ''Harendra Sonkar''),
('''Taj''', ''Syme'', ''Taj Syme''),
('''Rajeev''', ''Sandhu'', ''Rajeev Sandhu''),
('''Alok''', ''Sridhara'', ''Alok Sridhara''),
('''Falgun''', ''Bagchi'', ''Falgun Bagchi''),
('''Kashi''', ''Singh'', ''Kashi Singh''),
('''Bong-Soo''', ''Ha'', ''Bong-Soo Ha''),
('''Damodara''', ''Trivedi'', ''Damodara Trivedi''),
('''Nguyen''', ''Banh'', ''Nguyen Banh''),
('''Lan''', ''Bach'', ''Lan Bach''),
('''Surya''', ''Kulkarni'', ''Surya Kulkarni''),
('''Afsar-ud-Din''', ''Zare'', ''Afsar-ud-Din Zare''),
('''Dita''', ''Kreslina'', ''Dita Kreslina''),
('''TunC''', ''Polat'', ''TunC Polat''),
('''Aleksandra''', ''Semjonov'', ''Aleksandra Semjonov''),
('''Bianh''', ''Banh'', ''Bianh Banh''),
('''Promita''', ''Chattopadhyay'', ''Promita Chattopadhyay''),
('''Alessandro''', ''Sagese'', ''Alessandro Sagese''),
('''Dinh''', ''Mai'', ''Dinh Mai''),
('''Cam''', ''Dinh'', ''Cam Dinh''),
('''Shyam''', ''Sarma'', ''Shyam Sarma''),
('''Ramesh''', ''Das'', ''Ramesh Das''),
('''Inna''', ''Kask'', ''Inna Kask''),
('''Luis''', ''Saucedo'', ''Luis Saucedo''),
('''Ilgonis''', ''Prieditis'', ''Ilgonis Prieditis''),
('''Min-ji''', ''Nan'', ''Min-ji Nan''),
('''Risto''', ''Lepmets'', ''Risto Lepmets''),
('''Vjekoslava''', ''Brkic'', ''Vjekoslava Brkic''),
('''Spidols''', ''Podnieks'', ''Spidols Podnieks''),
('''Orions''', ''Podnieks'', ''Orions Podnieks''),
('''Kristine''', ''Zvaigzne'', ''Kristine Zvaigzne''),
('''Kalyani''', ''Benjaree'', ''Kalyani Benjaree''),
('''Gadhar''', ''Das'', ''Gadhar Das''),
('''Sashi''', ''Dev'', ''Sashi Dev''),
('''Bhadram''', ''Kamasamudram'', ''Bhadram Kamasamudram''),
('''Som''', ''Mukherjee'', ''Som Mukherjee''),
('''Kyle''', ''Redd'', ''Kyle Redd''),
('''Śani''', ''Sarkar'', ''Śani Sarkar''),
('''Narendra''', ''Tickoo'', ''Narendra Tickoo''),
('''Ganesh''', ''Majumdar'', ''Ganesh Majumdar''),
('''Anusuya''', ''Dutta'', ''Anusuya Dutta''),
('''Katarina''', ''Filipovic'', ''Katarina Filipovic''),
('''Dhanya''', ''Mokkapati'', ''Dhanya Mokkapati''),
('''Mehmet''', ''Arslan'', ''Mehmet Arslan''),
('''Gita''', ''Bhutia'', ''Gita Bhutia''),
('''Tapas''', ''Sikdar'', ''Tapas Sikdar''),
('''Lucija''', ''Cosic'', ''Lucija Cosic''),
('''Vitalijs''', ''Baltins'', ''Vitalijs Baltins''),
('''Kanchana''', ''Dutta'', ''Kanchana Dutta''),
('''Elvira''', ''Konovalova'', ''Elvira Konovalova''),
('''Preecha''', ''Suppamongkon'', ''Preecha Suppamongkon''),
('''Min-ji''', ''Shim'', ''Min-ji Shim''),
('''Noora''', ''Piili'', ''Noora Piili''),
('''Arshagouhi''', ''Deilami'', ''Arshagouhi Deilami''),
('''Risto''', ''Lill'', ''Risto Lill''),
('''Emma''', ''Van Zant'', ''Emma Van Zant''),
('''Hardi''', ''Laurits'', ''Hardi Laurits''),
('''Zoltan''', ''Gero'', ''Zoltan Gero''),
('''Soner''', ''Guler'', ''Soner Guler''),
('''Abhra''', ''Ganguly'', ''Abhra Ganguly''),
('''Fabrice''', ''Cloutier'', ''Fabrice Cloutier''),
('''Yonca''', ''Basturk'', ''Yonca Basturk''),
('''Nandita''', ''Bhuiyan'', ''Nandita Bhuiyan''),
('''Omar''', ''Lind'', ''Omar Lind''),
('''Mai''', ''Thai'', ''Mai Thai''),
('''David''', ''Novacek '', ''David Novacek ''),
('''Adriana''', ''Pena'', ''Adriana Pena''),
('''Rato''', ''Novakovic'', ''Rato Novakovic''),
('''Neelam''', ''Ahmadi'', ''Neelam Ahmadi''),
('''Phoung''', ''Du'', ''Phoung Du''),
('''Luca''', ''Barese'', ''Luca Barese''),
('''Aasaajyoeti''', ''Bhogireddy'', ''Aasaajyoeti Bhogireddy''),
('''Catherine''', ''Potts'', ''Catherine Potts''),
('''Aishwarya''', ''Tottempudi'', ''Aishwarya Tottempudi''),
('''Aarti''', ''Kommineni'', ''Aarti Kommineni''),
('''Lilli''', ''Peetre'', ''Lilli Peetre''),
```

```
(''Lassi'', ''Santala'', ''Lassi Santala''),
(''Umut'', ''Acar'', ''Umut Acar''),
(''Kevin'', ''Rummo'', ''Kevin Rummo''),
(''Nargis'', ''Shakiba'', ''Nargis Shakiba''),
(''Irma'', ''Berzina'', ''Irma Berzina''),
(''Irma'', ''Auzina'', ''Irma Auzina''),
(''Manindra'', ''Sidhu'', ''Manindra Sidhu''),
(''Aita'', ''Kasesalu'', ''Aita Kasesalu''),
(''Narayan'', ''Ogra'', ''Narayan Ogra''),
(''Amrita'', ''Shetty'', ''Amrita Shetty''),
(''Logan'', ''Dixon'', ''Logan Dixon''),
(''Celik'', ''TunC'', ''Celik TunC''),
(''David'', ''Jaramillo'', ''David Jaramillo''),
(''Gagan'', ''Sengupta'', ''Gagan Sengupta''),
(''Kalpana'', ''Sen'', ''Kalpana Sen''),
(''Charline'', ''Monjeau'', ''Charline Monjeau''),
(''Essie'', ''Braggs'', ''Essie Braggs''),
(''Teresa'', ''Fields'', ''Teresa Fields''),
(''Ron'', ''Williams'', ''Ron Williams''),
(''Daniela'', ''Lo Duca'', ''Daniela Lo Duca''),
(''Ashutosh'', ''Bandopadhyay'', ''Ashutosh Bandopadhyay''),
(''Cristina'', ''Angelo'', ''Cristina Angelo''),
(''Indranil'', ''Prabhupāda'', ''Indranil Prabhupāda''),
(''Julia'', ''Eder'', ''Julia Eder''),
(''Baebeesaroejini'', ''Veturi'', ''Baebeesaroejini Veturi''),
(''Giovanna'', ''Loggia'', ''Giovanna Loggia''),
(''Nicola'', ''Dellucci'', ''Nicola Dellucci''),
(''Pavel'', ''Bures'', ''Pavel Bures''),
(''Bhaamini'', ''Palagummi'', ''Bhaamini Palagummi''),
(''Cyrus'', ''Zardindoost'', ''Cyrus Zardindoost''),
(''Jautrite'', ''Avotina'', ''Jautrite Avotina''),
(''Matija'', ''Rusl'', ''Matija Rusl''),
(''Daniella'', ''Cavalcante'', ''Daniella Cavalcante''),
(''Vedrana'', ''Kovacevic'', ''Vedrana Kovacevic''),
(''Isa'', ''Hulsegge'', ''Isa Hulsegge''),
(''Ivana'', ''Popov'', ''Ivana Popov''),
(''Tuulikki'', ''Linna'', ''Tuulikki Linna''),
(''Allan'', ''Olofsson'', ''Allan Olofsson''),
(''Cosmin'', ''Vulpes'', ''Cosmin Vulpes''),
(''Dipti'', ''Shah'', ''Dipti Shah''),
(''Teresa'', ''Borgen'', ''Teresa Borgen''),
(''Veronika'', ''Necesana'', ''Veronika Necesana''),
(''Alfonso'', ''Barese'', ''Alfonso Barese''),
(''Erik'', ''Malk'', ''Erik Malk''),
(''Deepa'', ''Nandamuri'', ''Deepa Nandamuri''),
(''Arka'', ''Chatterjee'', ''Arka Chatterjee''),
(''Veronika'', ''Svancarova'', ''Veronika Svancarova''),
(''Felipe'', ''Robles'', ''Felipe Robles''),
(''Tami'', ''Shuler'', ''Tami Shuler''),
(''Flynn'', ''Moresby'', ''Flynn Moresby''),
(''Harsha'', ''Raju'', ''Harsha Raju''),
(''Aishwarya'', ''Dantuluri'', ''Aishwarya Dantuluri''),
(''Truman'', ''Schmidt'', ''Truman Schmidt''),
(''Divyendu'', ''Sen'', ''Divyendu Sen''),
(''Nhung'', ''Ton'', ''Nhung Ton''),
(''Cuneyt'', ''Arslan'', ''Cuneyt Arslan''),
(''Drishti'', ''Bose'', ''Drishti Bose''),
(''Farzana'', ''Habibi'', ''Farzana Habibi''),
(''Angelica'', ''Nilsson'', ''Angelica Nilsson''),
(''Arjun'', ''Bhowmick'', ''Arjun Bhowmick''),
(''Salamans'', ''Karklins'', ''Salamans Karklins''),
(''Hyun-Shik'', ''Lee'', ''Hyun-Shik Lee''),
(''Anand'', ''Mudaliyar'', ''Anand Mudaliyar''),
(''Carlos'', ''Aguayo'', ''Carlos Aguayo''),
(''Sharmila'', ''Bhutia'', ''Sharmila Bhutia''),
(''Hanita'', ''Nookala'', ''Hanita Nookala''),
(''Ondrej'', ''Polak'', ''Ondrej Polak''),
(''Serdar'', ''ozden'', ''Serdar ozden''),
(''Serdar'', ''ozCelik'', ''Serdar ozCelik''),
(''Javiera'', ''Laureano'', ''Javiera Laureano''),
(''Rafael'', ''Azevedo'', ''Rafael Azevedo''),
(''Raj'', ''Verma'', ''Raj Verma''),
(''Philippe'', ''Bellefeuille'', ''Philippe Bellefeuille''),
(''Arda'', ''Gunes'', ''Arda Gunes''),
(''Marcello'', ''Longo'', ''Marcello Longo''),
(''Marcela'', ''Antunes'', ''Marcela Antunes''),
(''Matteo'', ''Cattaneo'', ''Matteo Cattaneo''),
(''Prasad'', ''Raju'', ''Prasad Raju''),
(''Peep'', ''Lill'', ''Peep Lill''),
```

```
 (''Chompoo'', ''Atitarn'', ''Chompoo Atitarn''),
 (''Emma'', ''Salpa'', ''Emma Salpa''),
 (''Le'', ''Chu'', ''Le Chu''),
 (''Kailash'', ''Mittal'', ''Kailash Mittal''),
 (''Pinja'', ''Pekkanen'', ''Pinja Pekkanen''),
 (''Karita'', ''Jantunen'', ''Karita Jantunen''),
 (''Antonio Carlos'', ''Rocha'', ''Antonio Carlos Rocha''),
 (''Kim-ly'', ''Vanh'', ''Kim-ly Vanh''),
 (''Cuc'', ''Du'', ''Cuc Du''),
 (''Chaowalit'', ''Rojumanong'', ''Chaowalit Rojumanong''),
 (''Maria'', ''Nechita'', ''Maria Nechita''),
 (''Shirley'', ''Doane'', ''Shirley Doane''),
 (''Roberto'', ''Sal'', ''Roberto Sal''),
 (''Damyanti'', ''Bhamidipati'', ''Damyanti Bhamidipati''),
 (''Aleksandrs'', ''Purins'', ''Aleksandrs Purins''),
 (''Alen'', ''Kustrin'', ''Alen Kustrin''),
 (''Urve'', ''Kasesalu'', ''Urve Kasesalu''),
 (''David'', ''Serbanescu'', ''David Serbanescu''),
 (''Nadir'', ''Seddigh'', ''Nadir Seddigh''),
 (''Dhirendro'', ''Ghatak'', ''Dhirendro Ghatak''),
 (''Monika'', ''Kozakova'', ''Monika Kozakova''),
 (''Riccardo'', ''Esposito'', ''Riccardo Esposito''),
 (''Aleksandra'', ''Abola'', ''Aleksandra Abola''),
 (''Agrita'', ''Abele'', ''Agrita Abele''),
 (''Sabrina'', ''Baresi'', ''Sabrina Baresi''),
 (''Mudar'', ''Mihajlovik'', ''Mudar Mihajlovik''),
 (''Liga'', ''Dumina'', ''Liga Dumina''),
 (''Buu'', ''Tran'', ''Buu Tran''),
 (''Annette '', ''Hetu'', ''Annette Hetu''),
 (''Sami'', ''Lundin'', ''Sami Lundin''),
 (''Sylvie'', ''Methot'', ''Sylvie Methot''),
 (''Petr'', ''Spousta'', ''Petr Spousta''),
 (''Lorenzo'', ''Howland'', ''Lorenzo Howland''),
 (''Fatima'', ''Pulido'', ''Fatima Pulido''),
 (''Rui'', ''Carvalho'', ''Rui Carvalho'')) AS Names(FirstName, LastName, FullName)
 ),
 UnusedNames
 AS
 (
 SELECT *
 FROM NamesToUse AS ntu
 WHERE NOT EXISTS (SELECT 1 FROM [Application].People AS p WHERE p.FullName = ntu.FullName)
 )
 SELECT TOP(1) @PrimaryContactFullName = un.FullName,
 @PrimaryContactFirstName = un.FirstName
 FROM UnusedNames AS un
 ORDER BY NEWID();

 SET @CustomerID = NEXT VALUE FOR Sequences.CustomerID;
 SET @CustomerCategoryID = (SELECT TOP(1) CustomerCategoryID
 FROM Sales.CustomerCategories
 WHERE CustomerCategoryName IN (N''Novelty Shop'', N''Supermarket'', N''
Computer Store'', N''Gift Store'', N''Corporate'')  ORDER BY NEWID());
 SET @CityID = (SELECT TOP(1) CityID FROM [Application].Cities AS c
 ORDER BY NEWID());
 SET @CityName = (SELECT CityName FROM [Application].Cities WHERE CityID = @CityID);
 SET @CityStateProvinceID = (SELECT StateProvinceID FROM [Application].Cities WHERE CityID = @CityID);
 SET @CityStateProvinceCode = (SELECT StateProvinceCode
 FROM [Application].StateProvinces
 WHERE StateProvinceID = @CityStateProvinceID);
 SET @AreaCode = DataLoadSimulation.GetAreaCode(@CityStateProvinceCode);
 SET @StreetSuffix = (SELECT TOP(1) StreetType
 FROM (VALUES(N''Street''), (N''Lane''), (N''Avenue''), (N''Boulevard''), (N''
Crescent''), (N''Road'')) AS st(StreetType)  ORDER BY NEWID());
 SET @CompanySuffix = (SELECT TOP(1) CompanySuffix
 FROM (VALUES(N''Inc''), (N''Corp''), (N''LLC'')) AS cs(CompanySuffix)
 ORDER BY NEWID());
 SET @StorePrefix = (SELECT TOP(1) StorePrefix
 FROM (VALUES(N''Shop''), (N''Suite''), (N''Unit'')) AS sp(StorePrefix)
 ORDER BY NEWID());
 SET @CreditLimit = CEILING(RAND() * 30) * 100 + 1000;

 SET @DeliveryAddressLine1 = @StorePrefix + N'' '' + CAST(CEILING(RAND() * 30) + 1 AS nvarchar(20));
 SET @DeliveryAddressLine2 = CAST(CEILING(RAND() * 2000) + 1 AS nvarchar(20)) + N'' ''
 + (SELECT TOP(1) PreferredName FROM [Application].People ORDER BY NEWID())
 + N'' '' + @StreetSuffix;
 SET @DeliveryPostalCode = CAST(CEILING(RAND() * 800) + 90000 AS nvarchar(20));
 SET @PostalAddressLine1 = N''PO Box '' + CAST(CEILING(RAND() * 10000) + 10 AS nvarchar(20));
```

```
  SET @PostalAddressLine2 = (SELECT TOP(1) PreferredName FROM [Application].People ORDER BY NEWID()) + N''
 ville'';  SET @PostalPostalCode = @DeliveryPostalCode;

  SET @PrimaryContactPersonID = NEXT VALUE FOR Sequences.PersonID;

  BEGIN TRAN;

  INSERT [Application].People
  (PersonID, FullName, PreferredName, IsPermittedToLogon, LogonName,
  IsExternalLogonProvider, HashedPassword, IsSystemUser, IsEmployee,
  IsSalesperson, UserPreferences, PhoneNumber, FaxNumber,
  EmailAddress, LastEditedBy, ValidFrom, ValidTo)
  VALUES
  (@PrimaryContactPersonID, @PrimaryContactFullName, @PrimaryContactFirstName, 0, N''NO LOGON'',
  0, NULL, 0, 0,
  0, NULL, N''('' + CAST(@AreaCode AS nvarchar(20)) + N'') 555-0100'', N''('' + CAST(@AreaCode AS
 nvarchar(20)) + N'') 555-0101'',  LOWER(REPLACE(@PrimaryContactFirstName, N'''''''', N'''')) + N''@
 example.com'', 1, @CurrentDateTime, @EndOfTime);
  INSERT Sales.Customers
  (CustomerID, CustomerName, BillToCustomerID, CustomerCategoryID,
  BuyingGroupID, PrimaryContactPersonID, AlternateContactPersonID, DeliveryMethodID,
  DeliveryCityID, PostalCityID, CreditLimit, AccountOpenedDate, StandardDiscountPercentage,
  IsStatementSent, IsOnCreditHold, PaymentDays, PhoneNumber, FaxNumber,
  DeliveryRun, RunPosition, WebsiteURL, DeliveryAddressLine1, DeliveryAddressLine2,
  DeliveryPostalCode, DeliveryLocation, PostalAddressLine1, PostalAddressLine2,
  PostalPostalCode, LastEditedBy, ValidFrom, ValidTo)
  VALUES
  (@CustomerID, @PrimaryContactFullName, @CustomerID, @CustomerCategoryID,
  NULL, @PrimaryContactPersonID, NULL, @DeliveryMethodID,
  @CityID, @CityID, @CreditLimit, @StartingWhen, 0,
  0, 0, 7, N''('' + CAST(@AreaCode AS nvarchar(20)) + N'') 555-0100'', N''('' + CAST(@AreaCode AS
 nvarchar(20)) + N'') 555-0101'',  NULL, NULL, N''http://www.microsoft.com/'', @DeliveryAddressLine1, @
 DeliveryAddressLine2,  @DeliveryPostalCode, (SELECT TOP(1) Location FROM [Application].Cities WHERE
 CityID = @CityID),  @PostalAddressLine1, @PostalAddressLine2,
  @PostalPostalCode, 1, @CurrentDateTime, @EndOfTime);

  COMMIT;

  SET @Counter += 1;
  END;
 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                            AND name = N'
                                            AddSpecialDeals')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.AddSpecialDeals
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 AS
 BEGIN

  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  IF CAST(@CurrentDateTime AS date) = ''20151231''
  BEGIN
  BEGIN TRAN;

  INSERT Sales.SpecialDeals
  (StockItemID, CustomerID, BuyingGroupID, CustomerCategoryID, StockGroupID,
  DealDescription, StartDate, EndDate, DiscountAmount, DiscountPercentage,
  UnitPrice, LastEditedBy, LastEditedWhen)
  VALUES
  (NULL, NULL, (SELECT BuyingGroupID FROM Sales.BuyingGroups WHERE BuyingGroupName = N''Wingtip Toys''),
    NULL, (SELECT StockGroupID FROM Warehouse.StockGroups WHERE StockGroupName = N''USB Novelties''),
  N''10% 1st qtr USB Wingtip'', ''20160101'', ''20160331'', NULL, 10, NULL,
  2, @StartingWhen);

  INSERT Sales.SpecialDeals
  (StockItemID, CustomerID, BuyingGroupID, CustomerCategoryID, StockGroupID,
  DealDescription, StartDate, EndDate, DiscountAmount, DiscountPercentage,
  UnitPrice, LastEditedBy, LastEditedWhen)
  VALUES
```

```
  (NULL, NULL, (SELECT BuyingGroupID FROM Sales.BuyingGroups WHERE BuyingGroupName = N''Tailspin Toys'')
 ,   NULL, (SELECT StockGroupID FROM Warehouse.StockGroups WHERE StockGroupName = N''USB Novelties''),
  N''15% 2nd qtr USB Tailspin'', ''20160401'', ''20160630'', NULL, 15, NULL,
  2, @StartingWhen);

  COMMIT;

  END;

  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Adding special deals'';
  END;

 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                          AND name = N'
                                          AddStockItems')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.AddStockItems
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 AS
 BEGIN

  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  DECLARE @NumberOfStockItems int = 0;

  IF CAST(@CurrentDateTime AS date) = ''20160101''
  BEGIN
  SET @NumberOfStockItems = 2;

  BEGIN TRAN;

  INSERT Warehouse.StockItems (StockItemID, StockItemName, SupplierID, ColorID, UnitPackageID,
 OuterPackageID, Brand, Size, LeadTimeDays, QuantityPerOuter, IsChillerStock, Barcode, TaxRate, UnitPrice,
 RecommendedRetailPrice, TypicalWeightPerUnit, MarketingComments, InternalComments, Photo, CustomFields,
 LastEditedBy, ValidFrom, ValidTo) VALUES (220,''Novelty chilli chocolates 250g'',(SELECT SupplierID FROM
 Purchasing.Suppliers WHERE SupplierName = N''A Datum Corporation''),(SELECT ColorID FROM Warehouse.Colors WHERE
 ColorName = N''NULL''),(SELECT PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Bag''),(
 SELECT PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Carton''),NULL,''250g'',3,24,1,''
 8302039293929'',10,8.55,12.23,0.25,''Watch your friends faces when they eat these'',NULL,NULL,NULL,1,@
 CurrentDateTime,@EndOfTime);  INSERT Warehouse.StockItems (StockItemID, StockItemName, SupplierID, ColorID,
  UnitPackageID, OuterPackageID, Brand, Size, LeadTimeDays, QuantityPerOuter, IsChillerStock, Barcode, TaxRate,
 UnitPrice, RecommendedRetailPrice, TypicalWeightPerUnit, MarketingComments, InternalComments, Photo, CustomFields,
  LastEditedBy, ValidFrom, ValidTo) VALUES (221,''Novelty chilli chocolates 500g'',(SELECT SupplierID FROM
 Purchasing.Suppliers WHERE SupplierName = N''A Datum Corporation''),(SELECT ColorID FROM Warehouse.Colors WHERE
 ColorName = N''NULL''),(SELECT PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Bag''),(
 SELECT PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Carton''),NULL,''500g'',3,12,1,''
 8302039293647'',10,14.5,20.74,0.5,''Watch your friends faces when they eat these'',NULL,NULL,NULL,1,@
 CurrentDateTime,@EndOfTime);  INSERT Warehouse.StockItemHoldings (StockItemID, QuantityOnHand, BinLocation,
  LastStocktakeQuantity, LastCostPrice, ReorderLevel, TargetStockLevel, LastEditedBy, LastEditedWhen) VALUES (220,
 360,''CH-1'',360,4.75,240,500,1,@CurrentDateTime);  INSERT Warehouse.StockItemHoldings (StockItemID,
 QuantityOnHand, BinLocation, LastStocktakeQuantity, LastCostPrice, ReorderLevel, TargetStockLevel, LastEditedBy,
 LastEditedWhen) VALUES (221,12,''CH-2'',12,8.75,120,250,1,@CurrentDateTime);  INSERT Warehouse.
 StockItemStockGroups (StockItemID, StockGroupID, LastEditedBy, LastEditedWhen) SELECT 220, sg.StockGroupID, 1, @
 CurrentDateTime FROM Warehouse.StockGroups AS sg WHERE sg.StockGroupName IN (N''Novelty Items'', N''Edible
 Novelties'');  INSERT Warehouse.StockItemStockGroups (StockItemID, StockGroupID, LastEditedBy,
 LastEditedWhen) SELECT 221, sg.StockGroupID, 1, @CurrentDateTime FROM Warehouse.StockGroups AS sg WHERE sg.
 StockGroupName IN (N''Novelty Items'', N''Edible Novelties'');
  COMMIT;

  END ELSE IF CAST(@CurrentDateTime AS date) = ''20160102''
  BEGIN
  SET @NumberOfStockItems = 2;

  BEGIN TRAN;

  INSERT Warehouse.StockItems (StockItemID, StockItemName, SupplierID, ColorID, UnitPackageID,
 OuterPackageID, Brand, Size, LeadTimeDays, QuantityPerOuter, IsChillerStock, Barcode, TaxRate, UnitPrice,
```

```
RecommendedRetailPrice, TypicalWeightPerUnit, MarketingComments, InternalComments, Photo, CustomFields,
LastEditedBy, ValidFrom, ValidTo) VALUES (222,''Chocolate beetles 250g'',(SELECT SupplierID FROM Purchasing.
Suppliers WHERE SupplierName = N''A Datum Corporation''),(SELECT ColorID FROM Warehouse.Colors WHERE ColorName =
N''NULL''),(SELECT PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Bag''),(SELECT
PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Carton''),NULL,''250g'',3,24,1,''
8792838293820'',10,8.55,12.23,0.25,''Perfect for your child''''s party'',NULL,NULL,NULL,1,@CurrentDateTime,@
EndOfTime);  INSERT Warehouse.StockItems (StockItemID, StockItemName, SupplierID, ColorID, UnitPackageID,
OuterPackageID, Brand, Size, LeadTimeDays, QuantityPerOuter, IsChillerStock, Barcode, TaxRate, UnitPrice,
RecommendedRetailPrice, TypicalWeightPerUnit, MarketingComments, InternalComments, Photo, CustomFields,
LastEditedBy, ValidFrom, ValidTo) VALUES (223,''Chocolate echidnas 250g'',(SELECT SupplierID FROM Purchasing.
Suppliers WHERE SupplierName = N''A Datum Corporation''),(SELECT ColorID FROM Warehouse.Colors WHERE ColorName =
N''NULL''),(SELECT PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Bag''),(SELECT
PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Carton''),NULL,''250g'',3,24,1,''
8792838293728'',10,8.55,12.23,0.25,''Perfect for your child''''s party'',NULL,NULL,NULL,1,@CurrentDateTime,@
EndOfTime);  INSERT Warehouse.StockItemHoldings (StockItemID, QuantityOnHand, BinLocation,
LastStocktakeQuantity, LastCostPrice, ReorderLevel, TargetStockLevel, LastEditedBy, LastEditedWhen) VALUES (222,
120,''CH-3'',120,4.75,240,500,1,@CurrentDateTime);  INSERT Warehouse.StockItemHoldings (StockItemID,
QuantityOnHand, BinLocation, LastStocktakeQuantity, LastCostPrice, ReorderLevel, TargetStockLevel, LastEditedBy,
LastEditedWhen) VALUES (223,120,''CH-4'',120,4.75,240,500,1,@CurrentDateTime);  INSERT Warehouse.
StockItemStockGroups (StockItemID, StockGroupID, LastEditedBy, LastEditedWhen) SELECT 222, sg.StockGroupID, 1, @
CurrentDateTime FROM Warehouse.StockGroups AS sg WHERE sg.StockGroupName IN (N''Novelty Items'', N''Edible
Novelties'');  INSERT Warehouse.StockItemStockGroups (StockItemID, StockGroupID, LastEditedBy,
LastEditedWhen) SELECT 223, sg.StockGroupID, 1, @CurrentDateTime FROM Warehouse.StockGroups AS sg WHERE sg.
StockGroupName IN (N''Novelty Items'', N''Edible Novelties'');
 COMMIT;

 END ELSE IF CAST(@CurrentDateTime AS date) = ''20160104''
 BEGIN
 SET @NumberOfStockItems = 2;

 BEGIN TRAN;

 INSERT Warehouse.StockItems (StockItemID, StockItemName, SupplierID, ColorID, UnitPackageID,
OuterPackageID, Brand, Size, LeadTimeDays, QuantityPerOuter, IsChillerStock, Barcode, TaxRate, UnitPrice,
RecommendedRetailPrice, TypicalWeightPerUnit, MarketingComments, InternalComments, Photo, CustomFields,
LastEditedBy, ValidFrom, ValidTo) VALUES (224,''Chocolate frogs 250g'',(SELECT SupplierID FROM Purchasing.
Suppliers WHERE SupplierName = N''A Datum Corporation''),(SELECT ColorID FROM Warehouse.Colors WHERE ColorName =
N''NULL''),(SELECT PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Bag''),(SELECT
PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Carton''),NULL,''250g'',3,24,1,''
8792838293987'',10,8.55,12.23,0.25,''Perfect for your child''''s party'',NULL,NULL,NULL,1,@CurrentDateTime,@
EndOfTime);  INSERT Warehouse.StockItems (StockItemID, StockItemName, SupplierID, ColorID, UnitPackageID,
OuterPackageID, Brand, Size, LeadTimeDays, QuantityPerOuter, IsChillerStock, Barcode, TaxRate, UnitPrice,
RecommendedRetailPrice, TypicalWeightPerUnit, MarketingComments, InternalComments, Photo, CustomFields,
LastEditedBy, ValidFrom, ValidTo) VALUES (225,''Chocolate sharks 250g'',(SELECT SupplierID FROM Purchasing.
Suppliers WHERE SupplierName = N''A Datum Corporation''),(SELECT ColorID FROM Warehouse.Colors WHERE ColorName =
N''NULL''),(SELECT PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Bag''),(SELECT
PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Carton''),NULL,''250g'',3,24,1,''
8792838293234'',10,8.55,12.23,0.25,''Perfect for your child''''s party'',NULL,NULL,NULL,1,@CurrentDateTime,@
EndOfTime);  INSERT Warehouse.StockItemHoldings (StockItemID, QuantityOnHand, BinLocation,
LastStocktakeQuantity, LastCostPrice, ReorderLevel, TargetStockLevel, LastEditedBy, LastEditedWhen) VALUES (224,
144,''CH-5'',144,4.75,240,500,1,@CurrentDateTime);  INSERT Warehouse.StockItemHoldings (StockItemID,
QuantityOnHand, BinLocation, LastStocktakeQuantity, LastCostPrice, ReorderLevel, TargetStockLevel, LastEditedBy,
LastEditedWhen) VALUES (225,160,''CH-6'',160,4.75,240,500,1,@CurrentDateTime);  INSERT Warehouse.
StockItemStockGroups (StockItemID, StockGroupID, LastEditedBy, LastEditedWhen) SELECT 224, sg.StockGroupID, 1, @
CurrentDateTime FROM Warehouse.StockGroups AS sg WHERE sg.StockGroupName IN (N''Novelty Items'', N''Edible
Novelties'');  INSERT Warehouse.StockItemStockGroups (StockItemID, StockGroupID, LastEditedBy,
LastEditedWhen) SELECT 225, sg.StockGroupID, 1, @CurrentDateTime FROM Warehouse.StockGroups AS sg WHERE sg.
StockGroupName IN (N''Novelty Items'', N''Edible Novelties'');
 COMMIT;

 END ELSE IF CAST (@CurrentDateTime AS date) = ''20160105''
 BEGIN
 SET @NumberOfStockItems = 2;

 BEGIN TRAN;

 INSERT Warehouse.StockItems (StockItemID, StockItemName, SupplierID, ColorID, UnitPackageID,
OuterPackageID, Brand, Size, LeadTimeDays, QuantityPerOuter, IsChillerStock, Barcode, TaxRate, UnitPrice,
RecommendedRetailPrice, TypicalWeightPerUnit, MarketingComments, InternalComments, Photo, CustomFields,
LastEditedBy, ValidFrom, ValidTo) VALUES (226,''White chocolate snow balls 250g'',(SELECT SupplierID FROM
Purchasing.Suppliers WHERE SupplierName = N''A Datum Corporation''),(SELECT ColorID FROM Warehouse.Colors WHERE
ColorName = N''NULL''),(SELECT PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Bag''),(
SELECT PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Carton''),NULL,''250g'',3,24,1,''
8792838293236'',10,8.55,12.23,0.25,''Perfect for your child''''s party'',NULL,NULL,NULL,1,@CurrentDateTime,@
EndOfTime);  INSERT Warehouse.StockItems (StockItemID, StockItemName, SupplierID, ColorID, UnitPackageID,
OuterPackageID, Brand, Size, LeadTimeDays, QuantityPerOuter, IsChillerStock, Barcode, TaxRate, UnitPrice,
RecommendedRetailPrice, TypicalWeightPerUnit, MarketingComments, InternalComments, Photo, CustomFields,
LastEditedBy, ValidFrom, ValidTo) VALUES (227,''White chocolate moon rocks 250g'',(SELECT SupplierID FROM
Purchasing.Suppliers WHERE SupplierName = N''A Datum Corporation''),(SELECT ColorID FROM Warehouse.Colors WHERE
```

```
 ColorName = N''NULL''),(SELECT PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Bag''),(
 SELECT PackageTypeID FROM Warehouse.PackageTypes WHERE PackageTypeName = N''Carton''),NULL,''250g'',3,24,1,''
 8792838293289'',10,8.55,12.23,0.25,''Perfect for your child''''s party'',NULL,NULL,NULL,1,@CurrentDateTime,@
 EndOfTime);  INSERT Warehouse.StockItemHoldings (StockItemID, QuantityOnHand, BinLocation,
 LastStocktakeQuantity, LastCostPrice, ReorderLevel, TargetStockLevel, LastEditedBy, LastEditedWhen) VALUES (226,
 24,''CH-7'',24,4.75,240,500,1,@CurrentDateTime);  INSERT Warehouse.StockItemHoldings (StockItemID,
 QuantityOnHand, BinLocation, LastStocktakeQuantity, LastCostPrice, ReorderLevel, TargetStockLevel, LastEditedBy,
 LastEditedWhen) VALUES (227,48,''CH-8'',48,4.75,240,500,1,@CurrentDateTime);  INSERT Warehouse.
 StockItemStockGroups (StockItemID, StockGroupID, LastEditedBy, LastEditedWhen) SELECT 226, sg.StockGroupID, 1, @
 CurrentDateTime FROM Warehouse.StockGroups AS sg WHERE sg.StockGroupName IN (N''Novelty Items'', N''Edible
 Novelties'');  INSERT Warehouse.StockItemStockGroups (StockItemID, StockGroupID, LastEditedBy,
 LastEditedWhen) SELECT 227, sg.StockGroupID, 1, @CurrentDateTime FROM Warehouse.StockGroups AS sg WHERE sg.
 StockGroupName IN (N''Novelty Items'', N''Edible Novelties'');
  COMMIT;
  END;

  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Adding '' + CAST(@NumberOfStockItems AS nvarchar(20)) + N'' stock items'';
  END;

 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                            AND name = N'
                                            ChangePasswords')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.ChangePasswords
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 WITH EXECUTE AS OWNER
 AS
 BEGIN
  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  -- 1 in 4 days will be 1 password change, 1 in 8 days will be 2 passwords changed

  DECLARE @NumberOfPasswordsToChange int = (SELECT TOP(1) Quantity
  FROM (VALUES (0), (0), (0), (0), (0), (1), (1), (2)) AS q(Quantity)
  ORDER BY NEWID());
  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Changing '' + CAST(@NumberOfPasswordsToChange AS nvarchar(20)) + N'' passwords'';
  END;

  DECLARE @Counter int = 0;
  DECLARE @PersonID int;
  DECLARE @EmailAddress nvarchar(256);
  DECLARE @HashedPassword varbinary(max);
  DECLARE @FullName nvarchar(50);

  WHILE @Counter < @NumberOfPasswordsToChange
  BEGIN
  SELECT TOP(1) @PersonID = PersonID,
  @EmailAddress = EmailAddress,
  @FullName = FullName
  FROM [Application].People
  WHERE IsPermittedToLogon <> 0 AND PersonID <> 1
  ORDER BY NEWID();

  UPDATE [Application].People
  SET HashedPassword = HASHBYTES(N''SHA2_256'', N''SQLRocks!00'' + @FullName),
  [ValidFrom] = @StartingWhen
  WHERE PersonID = @PersonID;

  SET @Counter += 1;
  END;
 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
```

```
                                        AND name = N'
                                        CreateCustomerOrders')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.CreateCustomerOrders
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @NumberOfCustomerOrders int,
 @IsSilentMode bit
 WITH EXECUTE AS OWNER
 AS
 BEGIN
  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Creating '' + CAST(@NumberOfCustomerOrders AS nvarchar(20)) + N'' customer orders'';
  END;

  DECLARE @OrderCounter int = 0;
  DECLARE @OrderLineCounter int = 0;
  DECLARE @CustomerID int;
  DECLARE @OrderID int;
  DECLARE @PrimaryContactPersonID int;
  DECLARE @SalespersonPersonID int;
  DECLARE @ExpectedDeliveryDate date = DATEADD(day, 1, @CurrentDateTime);
  DECLARE @OrderDateTime datetime = @StartingWhen;
  DECLARE @NumberOfOrderLines int;
  DECLARE @StockItemID int;
  DECLARE @StockItemName nvarchar(100);
  DECLARE @UnitPackageID int;
  DECLARE @QuantityPerOuter int;
  DECLARE @Quantity int;
  DECLARE @CustomerPrice decimal(18,2);
  DECLARE @TaxRate decimal(18,3);

  -- No deliveries on weekends

  SET DATEFIRST 7;

  WHILE DATEPART(weekday, @ExpectedDeliveryDate) IN (1, 7)
  BEGIN
  SET @ExpectedDeliveryDate = DATEADD(day, 1, @ExpectedDeliveryDate);
  END;

  -- Generate the required orders

  WHILE @OrderCounter < @NumberOfCustomerOrders
  BEGIN

  BEGIN TRAN;

  SET @OrderID = NEXT VALUE FOR Sequences.OrderID;

  SELECT TOP(1) @CustomerID = c.CustomerID,
  @PrimaryContactPersonID = c.PrimaryContactPersonID
  FROM Sales.Customers AS c
  WHERE c.IsOnCreditHold = 0
  ORDER BY NEWID();

  SET @SalespersonPersonID = (SELECT TOP(1) PersonID
  FROM [Application].People
  WHERE IsSalesperson <> 0
  ORDER BY NEWID());

  INSERT Sales.Orders
  (OrderID, CustomerID, SalespersonPersonID, PickedByPersonID, ContactPersonID, BackorderOrderID,
 OrderDate,   ExpectedDeliveryDate, CustomerPurchaseOrderNumber, IsUndersupplyBackordered, Comments,
 DeliveryInstructions, InternalComments,   PickingCompletedWhen, LastEditedBy, LastEditedWhen)
  VALUES
  (@OrderID, @CustomerID, @SalespersonPersonID, NULL, @PrimaryContactPersonID, NULL, @CurrentDateTime,
  @ExpectedDeliveryDate, CAST(CEILING(RAND() * 10000) + 10000 AS nvarchar(20)), 1, NULL, NULL, NULL,
  NULL, 1, @OrderDateTime);

  SET @NumberOfOrderLines = 1 + CEILING(RAND() * 4);
  SET @OrderLineCounter = 0;
```

```
  WHILE @OrderLineCounter < @NumberOfOrderLines
  BEGIN
  SELECT TOP(1) @StockItemID = si.StockItemID,
  @StockItemName = si.StockItemName,
  @UnitPackageID = si.UnitPackageID,
  @QuantityPerOuter = si.QuantityPerOuter,
  @TaxRate = si.TaxRate
  FROM Warehouse.StockItems AS si
  WHERE NOT EXISTS (SELECT 1 FROM Sales.OrderLines AS ol
  WHERE ol.OrderID = @OrderID
  AND ol.StockItemID = si.StockItemID)
  ORDER BY NEWID();

  SET @Quantity = @QuantityPerOuter * (1 + FLOOR(RAND() * 10));
  SET @CustomerPrice = Website.CalculateCustomerPrice(@CustomerID, @StockItemID, @CurrentDateTime);

  INSERT Sales.OrderLines
  (OrderID, StockItemID, [Description], PackageTypeID, Quantity, UnitPrice,
  TaxRate, PickedQuantity, PickingCompletedWhen, LastEditedBy, LastEditedWhen)
  VALUES
  (@OrderID, @StockItemID, @StockItemName, @UnitPackageID, @Quantity, @CustomerPrice,
  @TaxRate, 0, NULL, 1, @StartingWhen);

  SET @OrderLineCounter += 1;
  END;

  COMMIT;

  SET @OrderCounter += 1;
  END;
 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                          AND name = N'
                                          InvoicePickedOrders')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.InvoicePickedOrders
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 AS
 BEGIN
  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Invoicing picked orders'';
  END;

  DECLARE @OrderID int;
  DECLARE @InvoiceID int;
  DECLARE @PickingCompletedWhen datetime;
  DECLARE @BackorderOrderID int;
  DECLARE @BillToCustomerID int;
  DECLARE @InvoicingPersonID int = (SELECT TOP(1) PersonID FROM [Application].People WHERE IsEmployee <>
 0 ORDER BY NEWID());  DECLARE @PackedByPersonID int = (SELECT TOP(1) PersonID FROM [Application].People WHERE
 IsEmployee <> 0 ORDER BY NEWID());  DECLARE @TotalDryItems int;
  DECLARE @TotalChillerItems int;
  DECLARE @TransactionAmount decimal(18,2);
  DECLARE @TaxAmount decimal(18,2);
  DECLARE @ReturnedDeliveryData nvarchar(max);
  DECLARE @DeliveryEvent nvarchar(max);

  DECLARE OrderList CURSOR FAST_FORWARD READ_ONLY
  FOR
  SELECT o.OrderID, o.PickingCompletedWhen, c.BillToCustomerID
  FROM Sales.Orders AS o
  INNER JOIN Sales.Customers AS c
  ON o.CustomerID = c.CustomerID
  WHERE NOT EXISTS (SELECT 1 FROM Sales.Invoices AS i WHERE i.OrderID = o.OrderID) -- not already invoiced
  AND c.IsOnCreditHold = 0 -- and customer not on
 credit hold  AND ((o.PickingCompletedWhen IS NOT NULL) -- order
 completely picked  OR (o.PickingCompletedWhen IS NULL -- order
```

```
not picked but customer happy  AND o.IsUndersupplyBackordered <> 0
 -- for part shipments and at least one  AND EXISTS (SELECT 1 FROM Sales.OrderLines AS ol
 -- order line has been picked  WHERE ol.OrderID = o.OrderID
 AND ol.PickingCompletedWhen IS NOT NULL)));

 OPEN OrderList;
 FETCH NEXT FROM OrderList INTO @OrderID, @PickingCompletedWhen, @BillToCustomerID;

 WHILE @@FETCH_STATUS = 0
 BEGIN
 IF @PickingCompletedWhen IS NULL
 BEGIN -- need to reorder undersupplied items

 BEGIN TRAN;

 SET @BackorderOrderID = NEXT VALUE FOR Sequences.OrderID;
 SET @PickingCompletedWhen = @StartingWhen;

 -- create the backorder order
 INSERT Sales.Orders
 (OrderID, CustomerID, SalespersonPersonID, PickedByPersonID, ContactPersonID, BackorderOrderID,
 OrderDate, ExpectedDeliveryDate, CustomerPurchaseOrderNumber, IsUndersupplyBackordered,
 Comments, DeliveryInstructions, InternalComments, PickingCompletedWhen, LastEditedBy,
LastEditedWhen)  SELECT @BackorderOrderID, o.CustomerID, o.SalespersonPersonID, NULL, o.
ContactPersonID, NULL,  o.OrderDate, o.ExpectedDeliveryDate, o.CustomerPurchaseOrderNumber, 1,
 o.Comments, o.DeliveryInstructions, o.InternalComments, NULL, @InvoicingPersonID, @
StartingWhen   FROM Sales.Orders AS o
 WHERE o.OrderID = @OrderID;

 -- move the items that haven''t been supplied to the new order
 UPDATE Sales.OrderLines
 SET OrderID = @BackorderOrderID,
 LastEditedBy = @InvoicingPersonID,
 LastEditedWhen = @StartingWhen
 WHERE OrderID = @OrderID
 AND PickingCompletedWhen IS NULL;

 -- flag the original order as backordered and picking completed
 UPDATE Sales.Orders
 SET BackorderOrderID = @BackorderOrderID,
 PickingCompletedWhen = @PickingCompletedWhen,
 LastEditedBy = @InvoicingPersonID,
 LastEditedWhen = @StartingWhen
 WHERE OrderID = @OrderID;

 COMMIT;
 END;

 SELECT @TotalDryItems = SUM(CASE WHEN si.IsChillerStock <> 0 THEN 0 ELSE 1 END),
 @TotalChillerItems = SUM(CASE WHEN si.IsChillerStock <> 0 THEN 1 ELSE 0 END)
 FROM Sales.OrderLines AS ol
 INNER JOIN Warehouse.StockItems AS si
 ON ol.StockItemID = si.StockItemID
 WHERE ol.OrderID = @OrderID;

 -- now invoice whatever is left on the order
 BEGIN TRAN;

 SET @InvoiceID = NEXT VALUE FOR Sequences.InvoiceID;

 SET @ReturnedDeliveryData = N''{"Events": []}'';
 SET @DeliveryEvent = N''{ }'';

 SET @DeliveryEvent = JSON_MODIFY(@DeliveryEvent, N''$.Event'', N''Ready for collection'');
 SET @DeliveryEvent = JSON_MODIFY(@DeliveryEvent, N''$.EventTime'', CONVERT(nvarchar(20), @StartingWhen,
126));  SET @DeliveryEvent = JSON_MODIFY(@DeliveryEvent, N''$.ConNote'', N''EAN-125-'' + CAST(@InvoiceID +
1050 AS nvarchar(20)));
 SET @ReturnedDeliveryData = JSON_MODIFY(@ReturnedDeliveryData, N''append $.Events'', JSON_QUERY(@
DeliveryEvent));
 INSERT Sales.Invoices
 (InvoiceID, CustomerID, BillToCustomerID, OrderID, DeliveryMethodID, ContactPersonID,
AccountsPersonID,   SalespersonPersonID, PackedByPersonID, InvoiceDate, CustomerPurchaseOrderNumber,
 IsCreditNote, CreditNoteReason, Comments, DeliveryInstructions, InternalComments,
 TotalDryItems, TotalChillerItems,
 DeliveryRun, RunPosition, ReturnedDeliveryData, LastEditedBy, LastEditedWhen)
 SELECT @InvoiceID, c.CustomerID, @BillToCustomerID, @OrderID, c.DeliveryMethodID, o.ContactPersonID, btc.
PrimaryContactPersonID,  o.SalespersonPersonID, @PackedByPersonID, @StartingWhen, o.
CustomerPurchaseOrderNumber,  0, NULL, NULL, c.DeliveryAddressLine1 + N'', '' + c.
DeliveryAddressLine2, NULL,  @TotalDryItems, @TotalChillerItems,
```

```
 c.DeliveryRun, c.RunPosition, @ReturnedDeliveryData, @InvoicingPersonID, @StartingWhen
 FROM Sales.Orders AS o
 INNER JOIN Sales.Customers AS c
 ON o.CustomerID = c.CustomerID
 INNER JOIN Sales.Customers AS btc
 ON btc.CustomerID = c.BillToCustomerID
 WHERE o.OrderID = @OrderID;

 INSERT Sales.InvoiceLines
 (InvoiceID, StockItemID, [Description], PackageTypeID,
 Quantity, UnitPrice, TaxRate, TaxAmount, LineProfit, ExtendedPrice,
 LastEditedBy, LastEditedWhen)
 SELECT @InvoiceID, ol.StockItemID, ol.[Description], ol.PackageTypeID,
 ol.PickedQuantity, ol.UnitPrice, ol.TaxRate,
 ROUND(ol.PickedQuantity * ol.UnitPrice * ol.TaxRate / 100.0, 2),
 ROUND(ol.PickedQuantity * (ol.UnitPrice - sih.LastCostPrice), 2),
 ROUND(ol.PickedQuantity * ol.UnitPrice, 2)
 + ROUND(ol.PickedQuantity * ol.UnitPrice * ol.TaxRate / 100.0, 2),
 @InvoicingPersonID, @StartingWhen
 FROM Sales.OrderLines AS ol
 INNER JOIN Warehouse.StockItems AS si
 ON ol.StockItemID = si.StockItemID
 INNER JOIN Warehouse.StockItemHoldings AS sih
 ON si.StockItemID = sih.StockItemID
 WHERE ol.OrderID = @OrderID
 ORDER BY ol.OrderLineID;

 INSERT Warehouse.StockItemTransactions
 (StockItemID, TransactionTypeID, CustomerID, InvoiceID, SupplierID, PurchaseOrderID,
 TransactionOccurredWhen, Quantity, LastEditedBy, LastEditedWhen)
 SELECT il.StockItemID, (SELECT TransactionTypeID FROM [Application].TransactionTypes WHERE
TransactionTypeName = N''Stock Issue''),  i.CustomerID, i.InvoiceID, NULL, NULL,
 @StartingWhen, 0 - il.Quantity, @InvoicingPersonID, @StartingWhen
 FROM Sales.InvoiceLines AS il
 INNER JOIN Sales.Invoices AS i
 ON il.InvoiceID = i.InvoiceID
 WHERE i.InvoiceID = @InvoiceID
 ORDER BY il.InvoiceLineID;

 WITH StockItemTotals
 AS
 (
 SELECT il.StockItemID, SUM(il.Quantity) AS TotalQuantity
 FROM Sales.InvoiceLines aS il
 WHERE il.InvoiceID = @InvoiceID
 GROUP BY il.StockItemID
 )
 UPDATE sih
 SET sih.QuantityOnHand -= sit.TotalQuantity,
 sih.LastEditedBy = @InvoicingPersonID,
 sih.LastEditedWhen = @StartingWhen
 FROM Warehouse.StockItemHoldings AS sih
 INNER JOIN StockItemTotals AS sit
 ON sih.StockItemID = sit.StockItemID;

 SELECT @TransactionAmount = SUM(il.ExtendedPrice),
 @TaxAmount = SUM(il.TaxAmount)
 FROM Sales.InvoiceLines AS il
 WHERE il.InvoiceID = @InvoiceID;

 INSERT Sales.CustomerTransactions
 (CustomerID, TransactionTypeID, InvoiceID, PaymentMethodID,
 TransactionDate, AmountExcludingTax, TaxAmount, TransactionAmount,
 OutstandingBalance, FinalizationDate, LastEditedBy, LastEditedWhen)
 VALUES
 (@BillToCustomerID, (SELECT TransactionTypeID FROM [Application].TransactionTypes WHERE
TransactionTypeName = N''Customer Invoice''),  @InvoiceID, NULL,
 @StartingWhen, @TransactionAmount - @TaxAmount, @TaxAmount, @TransactionAmount,
 @TransactionAmount, NULL, @InvoicingPersonID, @StartingWhen);

 COMMIT;
 FETCH NEXT FROM OrderList INTO @OrderID, @PickingCompletedWhen, @BillToCustomerID;
 END;

 CLOSE OrderList;
 DEALLOCATE OrderList;

END;';
```

```
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                            AND name = N'
                                            MakeTemporalChanges')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.MakeTemporalChanges
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 AS
 BEGIN

  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  DECLARE @Counter int;
  DECLARE @RowsToModify int;
  DECLARE @StaffMember int = (SELECT TOP(1) PersonID FROM [Application].People WHERE IsEmployee <> 0
 ORDER BY NEWID());
  IF DAY(@StartingWhen) = 1 AND MONTH(@StartingWhen) = 7
  BEGIN
  SET @Counter = 0;
  SET @RowsToModify = CEILING(RAND() * 20);

  WHILE @Counter < @RowsToModify
  BEGIN
  UPDATE [Application].Cities
  SET LatestRecordedPopulation = LatestRecordedPopulation * 1.04,
  LastEditedBy = @StaffMember,
  ValidFrom = @StartingWhen
  WHERE CityID = (SELECT TOP(1) CityID FROM [Application].Cities ORDER BY NEWID());
  SET @Counter += 1;
  END;
  END;

  IF DAY(@StartingWhen) = 1 AND MONTH(@StartingWhen) = 7
  BEGIN
  SET @Counter = 0;
  SET @RowsToModify = CEILING(RAND() * 20);

  WHILE @Counter < @RowsToModify
  BEGIN
  UPDATE [Application].StateProvinces
  SET LatestRecordedPopulation = LatestRecordedPopulation * 1.04,
  LastEditedBy = @StaffMember,
  ValidFrom = @StartingWhen
  WHERE StateProvinceID = (SELECT TOP(1) StateProvinceID FROM [Application].StateProvinces ORDER BY
 NEWID());  SET @Counter += 1;
  END;
  END;

  IF DAY(@StartingWhen) = 1 AND MONTH(@StartingWhen) = 7
  BEGIN
  SET @Counter = 0;
  SET @RowsToModify = CEILING(RAND() * 20);

  WHILE @Counter < @RowsToModify
  BEGIN
  UPDATE [Application].Countries
  SET LatestRecordedPopulation = LatestRecordedPopulation * 1.04,
  LastEditedBy = @StaffMember,
  ValidFrom = @StartingWhen
  WHERE CountryID = (SELECT TOP(1) CountryID FROM [Application].Countries ORDER BY NEWID());
  SET @Counter += 1;
  END;
  END;

  IF CAST(@StartingWhen AS date) = ''20150101''
  BEGIN
  UPDATE [Application].DeliveryMethods
  SET DeliveryMethodName = N''Chilled Van'',
  LastEditedBy = @StaffMember,
  ValidFrom = @StartingWhen
  WHERE DeliveryMethodName = N''Van with Chiller'';
  END;
```

```
IF CAST(@StartingWhen AS date) = ''20160101''
BEGIN
UPDATE [Application].PaymentMethods
SET PaymentMethodName = N''Credit-Card'',
LastEditedBy = @StaffMember,
ValidFrom = @StartingWhen
WHERE PaymentMethodName = N''Credit Card'';

INSERT [Application].TransactionTypes
(TransactionTypeName, LastEditedBy, ValidFrom, ValidTo)
VALUES
(N''Contra'', @StaffMember, @StartingWhen, @EndOfTime);

UPDATE [Application].TransactionTypes
SET TransactionTypeName = N''Customer Contra'',
LastEditedBy = @StaffMember,
ValidFrom = DATEADD(minute, 5, @StartingWhen)
WHERE TransactionTypeName = N''Contra'';

UPDATE Warehouse.Colors
SET ColorName = N''Steel Gray'',
LastEditedBy = @StaffMember,
ValidFrom = @StartingWhen
WHERE ColorName = N''Gray'';

INSERT Warehouse.PackageTypes
(PackageTypeName, LastEditedBy, ValidFrom, ValidTo)
VALUES
(N''Bin'', @StaffMember, @StartingWhen, @EndOfTime);

DELETE Warehouse.PackageTypes WHERE PackageTypeName = N''Bin'';
UPDATE Warehouse.StockGroups
SET StockGroupName = N''Furry Footwear'',
LastEditedBy = @StaffMember,
ValidFrom = @StartingWhen
WHERE StockGroupName = N''Footwear'';
END;

IF CAST(@StartingWhen AS date) = ''20150101''
BEGIN
UPDATE Purchasing.SupplierCategories
SET SupplierCategoryName = N''Courier Services Supplier'',
LastEditedBy = @StaffMember,
ValidFrom = @StartingWhen
WHERE SupplierCategoryName = N''Courier'';
END;

IF CAST(@StartingWhen AS date) = ''20140101''
BEGIN
INSERT Sales.CustomerCategories
(CustomerCategoryName, LastEditedBy, ValidFrom, ValidTo)
VALUES
(N''Retailer'', 1, @StartingWhen, @EndOfTime);

UPDATE Sales.CustomerCategories
SET CustomerCategoryName = N''General Retailer'',
LastEditedBy = @StaffMember,
ValidFrom = DATEADD(minute, 15, @StartingWhen)
WHERE CustomerCategoryName = N''Retailer'';
END;

IF DAY(@StartingWhen) = 1 AND MONTH(@StartingWhen) = 7
BEGIN
SET @Counter = 0;
SET @RowsToModify = CEILING(RAND() * 20);

WHILE @Counter < @RowsToModify
BEGIN
UPDATE Sales.Customers
SET CreditLimit = CreditLimit * 1.05,
LastEditedBy = @StaffMember,
ValidFrom = @StartingWhen
WHERE CustomerID = (SELECT TOP(1) CustomerID FROM Sales.Customers WHERE CreditLimit > 0 ORDER BY
NEWID());  SET @Counter += 1;
END;
END;
```


DataLoadSimulation.Configuration_ApplyDataLoadSimulationProcedures

```
  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Modifying a few temporal items '';
  END;

 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                            AND name = N'PaySuppliers')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.PaySuppliers
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 AS
 BEGIN
  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Paying suppliers'';
  END;

  DECLARE @StaffMemberPersonID int = (SELECT TOP(1) PersonID
  FROM [Application].People
  WHERE IsEmployee <> 0
  ORDER BY NEWID());

  DECLARE @TransactionsToPay TABLE
  (
  SupplierTransactionID int,
  SupplierID int,
  PurchaseOrderID int NULL,
  SupplierInvoiceNumber nvarchar(20) NULL,
  OutstandingBalance decimal(18,2)
  );

  INSERT @TransactionsToPay
  (SupplierTransactionID, SupplierID, PurchaseOrderID, SupplierInvoiceNumber, OutstandingBalance)
  SELECT SupplierTransactionID, SupplierID, PurchaseOrderID, SupplierInvoiceNumber, OutstandingBalance
  FROM Purchasing.SupplierTransactions
  WHERE IsFinalized = 0;

  BEGIN TRAN;

  UPDATE Purchasing.SupplierTransactions
  SET OutstandingBalance = 0,
  FinalizationDate = @StartingWhen,
  LastEditedBy = @StaffMemberPersonID,
  LastEditedWhen = @StartingWhen
  WHERE SupplierTransactionID IN (SELECT SupplierTransactionID FROM @TransactionsToPay);

  INSERT Purchasing.SupplierTransactions
  (SupplierID, TransactionTypeID, PurchaseOrderID, PaymentMethodID,
  SupplierInvoiceNumber, TransactionDate, AmountExcludingTax, TaxAmount, TransactionAmount,
  OutstandingBalance, FinalizationDate, LastEditedBy, LastEditedWhen)
  SELECT ttp.SupplierID, (SELECT TransactionTypeID FROM [Application].TransactionTypes WHERE
 TransactionTypeName = N''Supplier Payment Issued''),  NULL, (SELECT PaymentMethodID FROM [Application].
 PaymentMethods WHERE PaymentMethodName = N''EFT''),  NULL, CAST(@StartingWhen AS date), 0, 0, 0 - SUM(
 ttp.OutstandingBalance),  0, CAST(@StartingWhen AS date), @StaffMemberPersonID, @StartingWhen
  FROM @TransactionsToPay AS ttp
  GROUP BY ttp.SupplierID;

  COMMIT;

 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                            AND name = N'
                                            PerformStocktake')
BEGIN
```

```
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.PerformStocktake
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 AS
 BEGIN
  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Performing stocktake'';
  END;

  DECLARE @StaffMemberPersonID int = (SELECT TOP(1) PersonID
  FROM [Application].People
  WHERE IsEmployee <> 0
  ORDER BY NEWID());

  DECLARE @Counter int = 0;
  DECLARE @NumberOfAdjustedStockItems int = (SELECT CEILING(RAND() * 5));
  DECLARE @StockItemIDToAdjust int;
  DECLARE @QuantityToAdjust int;

  BEGIN TRAN;

  UPDATE Warehouse.StockItemHoldings
  SET LastStocktakeQuantity = QuantityOnHand,
  LastEditedBy = @StaffMemberPersonID,
  LastEditedWhen = @StartingWhen;

  WHILE @Counter < @NumberOfAdjustedStockItems
  BEGIN
  SET @QuantityToAdjust = 5 - CEILING(RAND() * 10);
  SET @StockItemIDToAdjust = (SELECT TOP(1) StockItemID
  FROM Warehouse.StockItemHoldings
  WHERE (QuantityOnHand + @QuantityToAdjust) >= 0
  ORDER BY NEWID());

  IF @StockItemIDToAdjust IS NOT NULL
  BEGIN

  UPDATE Warehouse.StockItemHoldings
  SET LastStocktakeQuantity += @QuantityToAdjust,
  LastEditedBy = @StaffMemberPersonID,
  LastEditedWhen = @StartingWhen
  WHERE StockItemID = @StockItemIDToAdjust;

  INSERT Warehouse.StockItemTransactions
  (StockItemID, TransactionTypeID, CustomerID, InvoiceID, SupplierID, PurchaseOrderID,
  TransactionOccurredWhen, Quantity, LastEditedBy, LastEditedWhen)
  VALUES
  (@StockItemIDToAdjust,
  (SELECT TransactionTypeID FROM [Application].TransactionTypes WHERE TransactionTypeName = N''
 Stock Adjustment at Stocktake''),  NULL, NULL, NULL, NULL,
  @StartingWhen, @QuantityToAdjust, @StaffMemberPersonID, @StartingWhen);
  END;
  SET @Counter += 1;
  END;

  COMMIT;
 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                          AND name = N'
                                          PickStockForCustomerOrders')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.PickStockForCustomerOrders
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 AS
```

```
BEGIN
 SET NOCOUNT ON;
 SET XACT_ABORT ON;

 IF @IsSilentMode = 0
 BEGIN
 PRINT N''Picking stock for customer orders'';
 END;

 SET XACT_ABORT ON;

 DECLARE @UninvoicedOrders TABLE
 (
 OrderID int PRIMARY KEY
 );

 INSERT @UninvoicedOrders
 SELECT o.OrderID
 FROM Sales.Orders AS o
 WHERE NOT EXISTS (SELECT 1 FROM Sales.Invoices AS i WHERE i.OrderID = o.OrderID);

 DECLARE @StockAlreadyAllocated TABLE
 (
 StockItemID int PRIMARY KEY,
 QuantityAllocated int
 );

 WITH StockAlreadyAllocated
 AS
 (
 SELECT ol.StockItemID, SUM(ol.PickedQuantity) AS TotalPickedQuantity
 FROM Sales.OrderLines AS ol
 INNER JOIN @UninvoicedOrders AS uo
 ON ol.OrderID = uo.OrderID
 WHERE ol.PickingCompletedWhen IS NULL
 GROUP BY ol.StockItemID
 )
 INSERT @StockAlreadyAllocated (StockItemID, QuantityAllocated)
 SELECT sa.StockItemID, sa.TotalPickedQuantity
 FROM StockAlreadyAllocated AS sa;

 DECLARE OrderLineList CURSOR FAST_FORWARD READ_ONLY
 FOR
 SELECT ol.OrderID, ol.OrderLineID, ol.StockItemID, ol.Quantity
 FROM Sales.OrderLines AS ol
 WHERE ol.PickingCompletedWhen IS NULL
 ORDER BY ol.OrderID, ol.OrderLineID;

 DECLARE @OrderID int;
 DECLARE @OrderLineID int;
 DECLARE @StockItemID int;
 DECLARE @Quantity int;
 DECLARE @AvailableStock int;
 DECLARE @PickingPersonID int = (SELECT TOP(1) PersonID
 FROM [Application].People
 WHERE IsEmployee <> 0
 ORDER BY NEWID());

 OPEN OrderLineList;
 FETCH NEXT FROM OrderLineList INTO @OrderID, @OrderLineID, @StockItemID, @Quantity;

 WHILE @@FETCH_STATUS = 0
 BEGIN
 -- work out available stock for this stock item (on hand less allocated)
 SET @AvailableStock = (SELECT QuantityOnHand FROM Warehouse.StockItemHoldings AS sih WHERE sih.
StockItemID = @StockItemID);  SET @AvailableStock -= COALESCE((SELECT QuantityAllocated FROM @
StockAlreadyAllocated AS saa WHERE saa.StockItemID = @StockItemID), 0);
 IF @AvailableStock >= @Quantity
 BEGIN
 BEGIN TRAN;

 MERGE @StockAlreadyAllocated AS saa
 USING (VALUES (@StockItemID, @Quantity)) AS sa(StockItemID, Quantity)
 ON saa.StockItemID = sa.StockItemID
 WHEN MATCHED THEN
 UPDATE SET saa.QuantityAllocated += sa.Quantity
 WHEN NOT MATCHED THEN
 INSERT (StockItemID, QuantityAllocated) VALUES (sa.StockItemID, sa.Quantity);
```

```
  -- reserve the required stock
  UPDATE Sales.OrderLines
  SET PickedQuantity = @Quantity,
  PickingCompletedWhen = @StartingWhen,
  LastEditedBy = @PickingPersonID,
  LastEditedWhen = @StartingWhen
  WHERE OrderLineID = @OrderLineID;

  -- mark the order as ready to invoice (picking complete) if all lines picked
  IF NOT EXISTS (SELECT 1 FROM Sales.OrderLines AS ol
  WHERE ol.OrderID = @OrderID
  AND ol.PickingCompletedWhen IS NULL)
  BEGIN
  UPDATE Sales.Orders
  SET PickingCompletedWhen = @StartingWhen,
  PickedByPersonID = @PickingPersonID,
  LastEditedBy = @PickingPersonID,
  LastEditedWhen = @StartingWhen
  WHERE OrderID = @OrderID;
  END;

  COMMIT;
  END;

  FETCH NEXT FROM OrderLineList INTO @OrderID, @OrderLineID, @StockItemID, @Quantity;
  END;

  CLOSE OrderLineList;
  DEALLOCATE OrderLineList;

 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                          AND name = N'
                                          PlaceSupplierOrders')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.PlaceSupplierOrders
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 AS
 BEGIN
  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Placing supplier orders'';
  END;

  DECLARE @ContactPersonID int = (SELECT TOP(1) PersonID FROM [Application].People WHERE IsEmployee <> 0
 ORDER BY NEWID());
  DECLARE @Orders TABLE
  (
  SupplierID int,
  PurchaseOrderID int NULL,
  DeliveryMethodID int,
  ContactPersonID int,
  SupplierReference nvarchar(20) NULL
  );

  DECLARE @OrderLines TABLE
  (
  StockItemID int,
  [Description] nvarchar(100),
  SupplierID int,
  QuantityOfOuters int,
  LeadTimeDays int,
  OuterPackageID int,
  LastOuterCostPrice decimal(18,2)
  );

  BEGIN TRAN;
```

```
WITH StockItemsToCheck
AS
(
SELECT si.StockItemID,
si.StockItemName AS [Description],
si.SupplierID,
sih.TargetStockLevel,
sih.ReorderLevel,
si.QuantityPerOuter,
si.LeadTimeDays,
si.OuterPackageID,
sih.QuantityOnHand,
sih.LastCostPrice,
COALESCE((SELECT SUM(ol.Quantity)
FROM Sales.OrderLines AS ol
WHERE ol.StockItemID = si.StockItemID
AND ol.PickingCompletedWhen IS NULL), 0) AS StockNeededForOrders,
COALESCE((SELECT si.QuantityPerOuter * SUM(pol.OrderedOuters - pol.ReceivedOuters)
FROM Purchasing.PurchaseOrderLines AS pol
WHERE pol.StockItemID = si.StockItemID
AND pol.IsOrderLineFinalized = 0), 0) AS StockOnOrder
FROM Warehouse.StockItems AS si
INNER JOIN Warehouse.StockItemHoldings AS sih
ON si.StockItemID = sih.StockItemID
),
StockItemsToOrder
AS
(
SELECT sitc.StockItemID,
sitc.[Description],
sitc.SupplierID,
(sitc.QuantityOnHand + sitc.StockOnOrder - sitc.StockNeededForOrders) AS EffectiveStockLevel,
sitc.TargetStockLevel,
sitc.QuantityPerOuter,
sitc.LeadTimeDays,
sitc.OuterPackageID,
sitc.LastCostPrice
FROM StockItemsToCheck AS sitc
WHERE (sitc.QuantityOnHand + sitc.StockOnOrder - sitc.StockNeededForOrders) < sitc.ReorderLevel
AND sitc.QuantityPerOuter <> 0
)
INSERT @OrderLines (StockItemID, [Description], SupplierID, QuantityOfOuters, LeadTimeDays, OuterPackageID,
LastOuterCostPrice)  SELECT sito.StockItemID,
sito.[Description],
sito.SupplierID,
CEILING((sito.TargetStockLevel - sito.EffectiveStockLevel) / sito.QuantityPerOuter) AS OutersRequired,
sito.LeadTimeDays,
sito.OuterPackageID,
ROUND(sito.LastCostPrice * sito.QuantityPerOuter, 2) AS LastOuterCostPrice
FROM StockItemsToOrder AS sito;

INSERT @Orders (SupplierID, PurchaseOrderID, DeliveryMethodID, ContactPersonID, SupplierReference)

SELECT s.SupplierID, NEXT VALUE FOR Sequences.PurchaseOrderID, s.DeliveryMethodID,
(SELECT TOP(1) PersonID FROM [Application].People WHERE IsEmployee <> 0),
s.SupplierReference
FROM Purchasing.Suppliers AS s
WHERE s.SupplierID IN (SELECT SupplierID FROM @OrderLines);

INSERT Purchasing.PurchaseOrders
(PurchaseOrderID, SupplierID, OrderDate, DeliveryMethodID, ContactPersonID,
ExpectedDeliveryDate, SupplierReference, IsOrderFinalized, Comments,
InternalComments, LastEditedBy, LastEditedWhen)
SELECT o.PurchaseOrderID, o.SupplierID, CAST(@StartingWhen AS date), o.DeliveryMethodID, o.ContactPersonID,
DATEADD(day, (SELECT MAX(LeadTimeDays) FROM @OrderLines), CAST(@StartingWhen AS date)),
o.SupplierReference, 0, NULL,
NULL, 1, @StartingWhen
FROM @Orders AS o;

INSERT Purchasing.PurchaseOrderLines
(PurchaseOrderID, StockItemID, OrderedOuters, [Description],
ReceivedOuters, PackageTypeID, ExpectedUnitPricePerOuter, LastReceiptDate,
IsOrderLineFinalized, LastEditedBy, LastEditedWhen)
SELECT o.PurchaseOrderID, ol.StockItemID, ol.QuantityOfOuters, ol.[Description],
0, ol.OuterPackageID, ol.LastOuterCostPrice, NULL,
0, @ContactPersonID, @StartingWhen
FROM @OrderLines AS ol
INNER JOIN @Orders AS o
ON ol.SupplierID = o.SupplierID;
```

```
  COMMIT;
 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                          AND name = N'
                                          ProcessCustomerPayments')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.ProcessCustomerPayments
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 WITH EXECUTE AS OWNER
 AS
 BEGIN
  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Processing customer payments'';
  END;

  DECLARE @StaffMemberPersonID int = (SELECT TOP(1) PersonID
  FROM [Application].People
  WHERE IsEmployee <> 0
  ORDER BY NEWID());

  DECLARE @TransactionsToReceive TABLE
  (
  CustomerTransactionID int,
  CustomerID int,
  InvoiceID int NULL,
  OutstandingBalance decimal(18,2)
  );

  INSERT @TransactionsToReceive
  (CustomerTransactionID, CustomerID, InvoiceID, OutstandingBalance)
  SELECT CustomerTransactionID, CustomerID, InvoiceID, OutstandingBalance
  FROM Sales.CustomerTransactions
  WHERE IsFinalized = 0;

  BEGIN TRAN;

  UPDATE Sales.CustomerTransactions
  SET OutstandingBalance = 0,
  FinalizationDate = @StartingWhen,
  LastEditedBy = @StaffMemberPersonID,
  LastEditedWhen = @StartingWhen
  WHERE CustomerTransactionID IN (SELECT CustomerTransactionID FROM @TransactionsToReceive);

  INSERT Sales.CustomerTransactions
  (CustomerID, TransactionTypeID, InvoiceID, PaymentMethodID, TransactionDate,
  AmountExcludingTax, TaxAmount, TransactionAmount, OutstandingBalance,
  FinalizationDate, LastEditedBy, LastEditedWhen)
  SELECT ttr.CustomerID, (SELECT TransactionTypeID FROM [Application].TransactionTypes WHERE
 TransactionTypeName = N''Customer Payment Received''),  NULL, (SELECT PaymentMethodID FROM [Application]
 .PaymentMethods WHERE PaymentMethodName = N''EFT''),  CAST(@StartingWhen AS date), 0, 0, 0 - SUM(ttr.
 OutstandingBalance),  0, CAST(@StartingWhen AS date), @StaffMemberPersonID, @StartingWhen
  FROM @TransactionsToReceive AS ttr
  GROUP BY ttr.CustomerID;

  COMMIT;

 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                          AND name = N'
                                          ReceivePurchaseOrders')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.ReceivePurchaseOrders
```

```
@CurrentDateTime datetime2(7),
@StartingWhen datetime,
@EndOfTime datetime2(7),
@IsSilentMode bit
AS
BEGIN
 SET NOCOUNT ON;
 SET XACT_ABORT ON;

 IF @IsSilentMode = 0
 BEGIN
 PRINT N''Receiving stock from purchase orders'';
 END;

 DECLARE @StaffMemberPersonID int = (SELECT TOP(1) PersonID
 FROM [Application].People
 WHERE IsEmployee <> 0
 ORDER BY NEWID());
 DECLARE @PurchaseOrderID int;
 DECLARE @SupplierID int;
 DECLARE @TotalExcludingTax decimal(18,2);
 DECLARE @TotalIncludingTax decimal(18,2);

 DECLARE PurchaseOrderList CURSOR FAST_FORWARD READ_ONLY
 FOR
 SELECT PurchaseOrderID, SupplierID
 FROM Purchasing.PurchaseOrders AS po
 WHERE po.IsOrderFinalized = 0
 AND po.ExpectedDeliveryDate >= @StartingWhen;

 OPEN PurchaseOrderList;
 FETCH NEXT FROM PurchaseOrderList INTO @PurchaseOrderID, @SupplierID;

 WHILE @@FETCH_STATUS = 0
 BEGIN

 BEGIN TRAN;

 UPDATE Purchasing.PurchaseOrderLines
 SET ReceivedOuters = OrderedOuters,
 IsOrderLineFinalized = 1,
 LastReceiptDate = CAST(@StartingWhen as date),
 LastEditedBy = @StaffMemberPersonID,
 LastEditedWhen = @StartingWhen
 WHERE PurchaseOrderID = @PurchaseOrderID;

 UPDATE sih
 SET sih.QuantityOnHand += pol.ReceivedOuters * si.QuantityPerOuter,
 sih.LastEditedBy = @StaffMemberPersonID,
 sih.LastEditedWhen = @StartingWhen
 FROM Warehouse.StockItemHoldings AS sih
 INNER JOIN Purchasing.PurchaseOrderLines AS pol
 ON sih.StockItemID = pol.StockItemID
 INNER JOIN Warehouse.StockItems AS si
 ON sih.StockItemID = si.StockItemID;

 INSERT Warehouse.StockItemTransactions
 (StockItemID, TransactionTypeID, CustomerID, InvoiceID, SupplierID, PurchaseOrderID,
 TransactionOccurredWhen, Quantity, LastEditedBy, LastEditedWhen)
 SELECT pol.StockItemID, (SELECT TransactionTypeID FROM [Application].TransactionTypes WHERE
TransactionTypeName = N''Stock Receipt''),  NULL, NULL, @SupplierID, pol.PurchaseOrderID,
 @StartingWhen, pol.ReceivedOuters * si.QuantityPerOuter, @StaffMemberPersonID, @StartingWhen
 FROM Purchasing.PurchaseOrderLines AS pol
 INNER JOIN Warehouse.StockItems AS si
 ON pol.StockItemID = si.StockItemID
 WHERE pol.PurchaseOrderID = @PurchaseOrderID;

 UPDATE Purchasing.PurchaseOrders
 SET IsOrderFinalized = 1,
 LastEditedBy = @StaffMemberPersonID,
 LastEditedWhen = @StartingWhen
 WHERE PurchaseOrderID = @PurchaseOrderID;

 SELECT @TotalExcludingTax = SUM(ROUND(pol.OrderedOuters * pol.ExpectedUnitPricePerOuter,2)),
 @TotalIncludingTax = SUM(ROUND(pol.OrderedOuters * pol.ExpectedUnitPricePerOuter,2))
 + SUM(ROUND(pol.OrderedOuters * pol.ExpectedUnitPricePerOuter * si.TaxRate /
100.0,2))  FROM Purchasing.PurchaseOrderLines AS pol
 INNER JOIN Warehouse.StockItems AS si
 ON pol.StockItemID = si.StockItemID
```

```
  WHERE pol.PurchaseOrderID = @PurchaseOrderID;

  INSERT Purchasing.SupplierTransactions
  (SupplierID, TransactionTypeID, PurchaseOrderID, PaymentMethodID,
  SupplierInvoiceNumber, TransactionDate, AmountExcludingTax,
  TaxAmount, TransactionAmount, OutstandingBalance,
  FinalizationDate, LastEditedBy, LastEditedWhen)
  VALUES
  (@SupplierID, (SELECT TransactionTypeID FROM [Application].TransactionTypes WHERE TransactionTypeName
 = N''Supplier Invoice''),  @PurchaseOrderID, (SELECT PaymentMethodID FROM [Application].
 PaymentMethods WHERE PaymentMethodName = N''EFT''),  CAST(CEILING(RAND() * 10000) AS nvarchar(20)),
 CAST(@StartingWhen AS date), @TotalExcludingTax,  @TotalIncludingTax - @TotalExcludingTax, @
 TotalIncludingTax, @TotalIncludingTax,  NULL, @StaffMemberPersonID, @StartingWhen);

  COMMIT;

  FETCH NEXT FROM PurchaseOrderList INTO @PurchaseOrderID, @SupplierID;
  END;

  CLOSE PurchaseOrderList;
  DEALLOCATE PurchaseOrderList;
 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                            AND name = N'
                                            RecordColdRoomTemperatures')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.RecordColdRoomTemperatures
 @AverageSecondsBetweenReadings int,
 @NumberOfSensors int,
 @CurrentDateTime datetime2(7),
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 WITH EXECUTE AS OWNER
 AS
 BEGIN

  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Recording cold room temperatures'';
  END;

  DECLARE @TimeCounter datetime2(7) = CAST(@CurrentDateTime AS date);
  DECLARE @SensorCounter int;
  DECLARE @DelayInSeconds int;
  DECLARE @TimeToFinishForTheDay datetime2(7) = DATEADD(second, -30, DATEADD(day, 1, @TimeCounter));
  DECLARE @Temperature decimal(10,2);

  WHILE @TimeCounter < @TimeToFinishForTheDay
  BEGIN
  SET @SensorCounter = 0;
  WHILE @SensorCounter < @NumberOfSensors
  BEGIN
  SET @Temperature = 3 + RAND() * 2;

  DELETE Warehouse.ColdRoomTemperatures
  OUTPUT deleted.ColdRoomTemperatureID,
  deleted.ColdRoomSensorNumber,
  deleted.RecordedWhen,
  deleted.Temperature,
  deleted.ValidFrom,
  @TimeCounter
  INTO Warehouse.ColdRoomTemperatures_Archive
  WHERE ColdRoomSensorNumber = @SensorCounter + 1;

  INSERT Warehouse.ColdRoomTemperatures
  (ColdRoomSensorNumber, RecordedWhen, Temperature, ValidFrom, ValidTo)
  VALUES
  (@SensorCounter + 1, @TimeCounter, @Temperature, @TimeCounter, @EndOfTime);

  SET @SensorCounter += 1;
  END;
```

```
 SET @DelayInSeconds = CEILING(RAND() * @AverageSecondsBetweenReadings);
 SET @TimeCounter = DATEADD(second, @DelayInSeconds, @TimeCounter);
 END;
END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                            AND name = N'
                                            RecordDeliveryVanTemperatures')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.RecordDeliveryVanTemperatures
 @AverageSecondsBetweenReadings int,
 @NumberOfSensors int,
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @IsSilentMode bit
 WITH EXECUTE AS OWNER
 AS
 BEGIN

  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Recording delivery van temperatures'';
  END;

  DECLARE @VehicleRegistration nvarchar(20) = N''WWI-321-A'';

  DECLARE @TimeCounter datetime2(7) = @StartingWhen;
  DECLARE @SensorCounter int;
  DECLARE @DelayInSeconds int;
  DECLARE @MidnightToday datetime2(7) = CAST(@StartingWhen AS date);
  DECLARE @TimeToFinishForTheDay datetime2(7) = DATEADD(hour, 16, @MidnightToday);
  DECLARE @Temperature decimal(10,2);
  DECLARE @FullSensorData nvarchar(1000);
  DECLARE @Latitude decimal(18,7);
  DECLARE @Longitude decimal(18,7);
  DECLARE @IsCompressed bit;

  WHILE @TimeCounter < @TimeToFinishForTheDay
  BEGIN
  SET @SensorCounter = 0;
  WHILE @SensorCounter < @NumberOfSensors
  BEGIN
  SET @Temperature = 3 + RAND() * 2;
  SET @Latitude = 37.78352 + RAND() * 30;
  SET @Longitude = -122.4169 + RAND() * 40;

  SET @IsCompressed = CASE WHEN @TimeCounter < ''20160101'' THEN 1 ELSE 0 END;

  SET @FullSensorData = N''{"Recordings": ''
  + N''['' 
  + N''{"type":"Feature", "geometry": {"type":"Point", "coordinates":[''
  + CAST(@Longitude AS nvarchar(20)) + N'','' + CAST(@Latitude AS nvarchar(20))
  + N''] }, "properties":{"rego":"'' + STRING_ESCAPE(@VehicleRegistration, N''json'')
  + N''","sensor":"'' + CAST(@SensorCounter + 1 AS nvarchar(20))
  + N'',"when":"'' + CONVERT(nvarchar(30), @TimeCounter, 126)
  + N''","temp":'' + CAST(@Temperature AS nvarchar(20))
  + N''}} ]'';

  INSERT Warehouse.VehicleTemperatures
  (VehicleRegistration, ChillerSensorNumber,
  RecordedWhen, Temperature,
  FullSensorData, IsCompressed, CompressedSensorData)
  VALUES
  (@VehicleRegistration, @SensorCounter + 1,
  @TimeCounter, @Temperature,
  CASE WHEN @IsCompressed = 0 THEN @FullSensorData END,
  @IsCompressed,
  CASE WHEN @IsCompressed <> 0 THEN COMPRESS(@FullSensorData) END);

  SET @SensorCounter += 1;
  END;
  SET @DelayInSeconds = CEILING(RAND() * @AverageSecondsBetweenReadings);
  SET @TimeCounter = DATEADD(second, @DelayInSeconds, @TimeCounter);
```

```
  END;
 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                          AND name = N'
                                          RecordInvoiceDeliveries')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.RecordInvoiceDeliveries
 @CurrentDateTime datetime2(7),
 @StartingWhen datetime,
 @EndOfTime datetime2(7),
 @IsSilentMode bit
 WITH EXECUTE AS OWNER
 AS
 BEGIN
  SET NOCOUNT ON;
  SET XACT_ABORT ON;

  IF @IsSilentMode = 0
  BEGIN
  PRINT N''Recording invoice deliveries'';
  END;

  DECLARE @DeliveryDriverPersonID int = (SELECT TOP(1) PersonID
  FROM [Application].People
  WHERE IsEmployee <> 0
  ORDER BY NEWID());

  DECLARE @ReturnedDeliveryData nvarchar(max);
  DECLARE @InvoiceID int;
  DECLARE @CustomerName nvarchar(100);
  DECLARE @PrimaryContactFullName nvarchar(50);
  DECLARE @Latitude decimal(18,7);
  DECLARE @Longitude decimal(18,7);
  DECLARE @DeliveryAttemptWhen datetime2(7);
  DECLARE @Counter int = 0;
  DECLARE @DeliveryEvent nvarchar(max);
  DECLARE @IsDelivered bit;

  DECLARE InvoiceList CURSOR FAST_FORWARD READ_ONLY
  FOR
  SELECT i.InvoiceID, i.ReturnedDeliveryData, c.CustomerName, p.FullName, ct.[Location].Lat, ct.[Location].Long
  FROM Sales.Invoices AS i
  INNER JOIN Sales.Customers AS c
  ON i.CustomerID = c.CustomerID
  INNER JOIN [Application].Cities AS ct
  ON c.DeliveryCityID = ct.CityID
  INNER JOIN [Application].People AS p
  ON c.PrimaryContactPersonID = p.PersonID
  WHERE i.ConfirmedDeliveryTime IS NULL
  AND i.InvoiceDate < CAST(@StartingWhen AS date)
  ORDER BY i.InvoiceID;

  OPEN InvoiceList;
  FETCH NEXT FROM InvoiceList INTO @InvoiceID, @ReturnedDeliveryData, @CustomerName, @PrimaryContactFullName, @
 Latitude, @Longitude;
  WHILE @@FETCH_STATUS = 0
  BEGIN
  SET @Counter += 1;
  SET @DeliveryAttemptWhen = DATEADD(minute, @Counter * 5, @StartingWhen);

  SET @DeliveryEvent = N''{ }'';
  SET @DeliveryEvent = JSON_MODIFY(@DeliveryEvent, N''$.Event'', N''DeliveryAttempt'');
  SET @DeliveryEvent = JSON_MODIFY(@DeliveryEvent, N''$.EventTime'', CONVERT(nvarchar(20), @
 DeliveryAttemptWhen, 126));  SET @DeliveryEvent = JSON_MODIFY(@DeliveryEvent, N''$.ConNote'', N''EAN-125-''
  + CAST(@InvoiceID + 1050 AS nvarchar(20)));  SET @DeliveryEvent = JSON_MODIFY(@DeliveryEvent, N''$.
 DriverID'', @DeliveryDriverPersonID);  SET @DeliveryEvent = JSON_MODIFY(@DeliveryEvent, N''$.Latitude'', @
 Latitude);  SET @DeliveryEvent = JSON_MODIFY(@DeliveryEvent, N''$.Longitude'', @Longitude);

  SET @IsDelivered = 0;

  IF RAND() < 0.1 -- 10 % chance of non-delivery on this attempt
  BEGIN
  SET @DeliveryEvent = JSON_MODIFY(@DeliveryEvent, N''$.Comment'', N''Receiver not present'');
  END ELSE BEGIN -- delivered
```

```
  SET @DeliveryEvent = JSON_MODIFY(@DeliveryEvent, N''$.Status'', N''Delivered'');
  SET @IsDelivered = 1;
  END;

  SET @ReturnedDeliveryData = JSON_MODIFY(@ReturnedDeliveryData, N''append $.Events'', JSON_QUERY(@
 DeliveryEvent));  SET @ReturnedDeliveryData = JSON_MODIFY(@ReturnedDeliveryData, N''$.DeliveredWhen'',
 CONVERT(nvarchar(20), @DeliveryAttemptWhen, 126));  SET @ReturnedDeliveryData = JSON_MODIFY(@
 ReturnedDeliveryData, N''$.ReceivedBy'', @PrimaryContactFullName);
  UPDATE Sales.Invoices
  SET ReturnedDeliveryData = @ReturnedDeliveryData,
  LastEditedBy = @DeliveryDriverPersonID,
  LastEditedWhen = @StartingWhen
  WHERE InvoiceID = @InvoiceID;

  FETCH NEXT FROM InvoiceList INTO @InvoiceID, @ReturnedDeliveryData, @CustomerName, @
 PrimaryContactFullName, @Latitude, @Longitude;  END;

  CLOSE InvoiceList;
  DEALLOCATE InvoiceList;
 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                            AND name = N'
                                            UpdateCustomFields')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.UpdateCustomFields
 @CurrentDateTime AS date
 WITH EXECUTE AS OWNER
 AS
 BEGIN
  DECLARE @StartingWhen datetime2(7) = CAST(@CurrentDateTime AS date);

  SET @StartingWhen = DATEADD(hour, 23, @StartingWhen);

  -- Populate custom data for stock items

  UPDATE Warehouse.StockItems
  SET CustomFields = N''{ "CountryOfManufacture": ''
  + CASE WHEN IsChillerStock <> 0 THEN N''"USA", "ShelfLife": "7 days"''
  WHEN StockItemName LIKE N''%USB food%'' THEN N''"Japan"''
  ELSE N''"China"''
  END
  + N'', "Tags": []''
  + CASE WHEN Size IN (N''S'', N''XS'', N''XXS'', N''3XS'') THEN N'', "Range": "Children"''
  WHEN Size IN (N''M'') THEN N'', "Range": "Teens/Young Adult"''
  WHEN Size IN (N''L'', N''XL'', N''XXL'', N''3XL'', N''4XL'', N''5XL'', N''6XL'', N''
 7XL'') THEN N'', "Range": "Adult"''  ELSE N''''
  END
  + CASE WHEN StockItemName LIKE N''RC %'' THEN N'', "MinimumAge": "10"''
  ELSE N''''
  END
  + N'' }'',
  ValidFrom = @StartingWhen;

  SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);

  UPDATE Warehouse.StockItems
  SET CustomFields = JSON_MODIFY(CustomFields, N''append $.Tags'', N''Radio Control''),
  ValidFrom = @StartingWhen
  WHERE StockItemName LIKE N''RC %'';

  SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);

  UPDATE Warehouse.StockItems
  SET CustomFields = JSON_MODIFY(CustomFields, N''append $.Tags'', N''Realistic Sound''),
  ValidFrom = @StartingWhen
  WHERE StockItemName LIKE N''RC %'';

  SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);

  UPDATE Warehouse.StockItems
  SET CustomFields = JSON_MODIFY(CustomFields, N''append $.Tags'', N''Vintage''),
  ValidFrom = @StartingWhen
  WHERE StockItemName LIKE N''%vintage%'';

  SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);
```

```
UPDATE Warehouse.StockItems
SET CustomFields = JSON_MODIFY(CustomFields, N''append $.Tags'', N''Halloween Fun''),
ValidFrom = @StartingWhen
WHERE StockItemName LIKE N''%halloween%'';

SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);

UPDATE Warehouse.StockItems
SET CustomFields = JSON_MODIFY(CustomFields, N''append $.Tags'', N''Super Value''),
ValidFrom = @StartingWhen
WHERE StockItemName LIKE N''%pack of%'';

SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);

UPDATE Warehouse.StockItems
SET CustomFields = JSON_MODIFY(CustomFields, N''append $.Tags'', N''So Realistic''),
ValidFrom = @StartingWhen
WHERE StockItemName LIKE N''%ride on%'';

SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);

UPDATE Warehouse.StockItems
SET CustomFields = JSON_MODIFY(CustomFields, N''append $.Tags'', N''Comfortable''),
ValidFrom = @StartingWhen
WHERE StockItemName LIKE N''%slipper%'';

SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);

UPDATE Warehouse.StockItems
SET CustomFields = JSON_MODIFY(CustomFields, N''append $.Tags'', N''Long Battery Life''),
ValidFrom = @StartingWhen
WHERE StockItemName LIKE N''%slipper%'';

SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);

UPDATE Warehouse.StockItems
SET CustomFields = JSON_MODIFY(CustomFields, N''append $.Tags'', CASE WHEN StockItemID % 2 = 0 THEN N''32GB''
ELSE N''16GB'' END),  ValidFrom = @StartingWhen
WHERE StockItemName LIKE N''%USB food%'';

SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);

UPDATE Warehouse.StockItems
SET CustomFields = JSON_MODIFY(CustomFields, N''append $.Tags'', N''Comedy''),
ValidFrom = @StartingWhen
WHERE StockItemName LIKE N''%joke%'';

SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);

UPDATE Warehouse.StockItems
SET CustomFields = JSON_MODIFY(CustomFields, N''append $.Tags'', N''USB Powered''),
ValidFrom = @StartingWhen
WHERE StockItemName LIKE N''%USB%'';

SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);

UPDATE si
SET si.CustomFields = JSON_MODIFY(si.CustomFields, N''append $.Tags'', N''Limited Stock''),
ValidFrom = @StartingWhen
FROM Warehouse.StockItems AS si
WHERE EXISTS (SELECT 1
FROM Warehouse.StockItemStockGroups AS sisg
INNER JOIN Warehouse.StockGroups AS sg
ON sisg.StockGroupID = sg.StockGroupID
WHERE si.StockItemID = sisg.StockItemID
AND sg.StockGroupName LIKE N''%Packaging%'');

-- populate custom data for employees and salespeople

SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);

DECLARE EmployeeList CURSOR FAST_FORWARD READ_ONLY
FOR
SELECT PersonID, IsSalesperson
FROM [Application].People
WHERE IsEmployee <> 0;
```

```
DECLARE @EmployeeID int;
DECLARE @IsSalesperson bit;
DECLARE @CustomFields nvarchar(max);
DECLARE @JobTitle nvarchar(max);
DECLARE @NumberOfAdditionalLanguages int;
DECLARE @LanguageCounter int;
DECLARE @OtherLanguages TABLE ( LanguageName nvarchar(50) );
DECLARE @LanguageName nvarchar(50);

OPEN EmployeeList;
FETCH NEXT FROM EmployeeList INTO @EmployeeID, @IsSalesperson;

WHILE @@FETCH_STATUS = 0
BEGIN
SET @CustomFields = N''{ "OtherLanguages": [] }'';

SET @NumberOfAdditionalLanguages = FLOOR(RAND() * 4);
DELETE @OtherLanguages;
SET @LanguageCounter = 0;
WHILE @LanguageCounter < @NumberOfAdditionalLanguages
BEGIN
SET @LanguageName = (SELECT TOP(1) alias
FROM sys.syslanguages
WHERE alias NOT LIKE N''%English%''
AND alias NOT LIKE N''%Brazil%''
ORDER BY NEWID());
IF @LanguageName LIKE N''%Chinese%'' SET @LanguageName = N''Chinese'';
IF NOT EXISTS (SELECT 1 FROM @OtherLanguages WHERE LanguageName = @LanguageName)
BEGIN
INSERT @OtherLanguages (LanguageName) VALUES(@LanguageName);
SET @CustomFields = JSON_MODIFY(@CustomFields, N''append $.OtherLanguages'', @LanguageName);
END;
SET @LanguageCounter += 1;
END;

SET @CustomFields = JSON_MODIFY(@CustomFields, N''$.HireDate'',
CONVERT(nvarchar(20), DATEADD(day, 0 - CEILING(RAND() * 2000) - 100, ''
20130101''), 126));
SET @JobTitle = N''Team Member'';
SET @JobTitle = CASE WHEN RAND() < 0.05 THEN N''General Manager''
WHEN RAND() < 0.1 THEN N''Manager''
WHEN RAND() < 0.15 THEN N''Accounts Controller''
WHEN RAND() < 0.2 THEN N''Warehouse Supervisor''
ELSE @JobTitle
END;
SET @CustomFields = JSON_MODIFY(@CustomFields, N''$.Title'', @JobTitle);

IF @IsSalesperson <> 0
BEGIN
SET @CustomFields = JSON_MODIFY(@CustomFields, N''$.PrimarySalesTerritory'',
(SELECT TOP(1) SalesTerritory FROM [Application].StateProvinces ORDER
BY NEWID()));  SET @CustomFields = JSON_MODIFY(@CustomFields, N''$.CommissionRate'',
CAST(CAST(RAND() * 5 AS decimal(18,2)) AS nvarchar(20)));
END;

UPDATE [Application].People
SET CustomFields = @CustomFields,
ValidFrom = @StartingWhen
WHERE PersonID = @EmployeeID;

FETCH NEXT FROM EmployeeList INTO @EmployeeID, @IsSalesperson;
END;

CLOSE EmployeeList;
DEALLOCATE EmployeeList;

-- Set user preferences

SET @StartingWhen = DATEADD(minute, 1, @StartingWhen);

UPDATE Application.People
SET UserPreferences = N''{"theme":"''+ (CASE (PersonID % 7)
WHEN 0 THEN ''ui-darkness''
WHEN 1 THEN ''blitzer''
WHEN 2 THEN ''humanity''
WHEN 3 THEN ''dark-hive''
WHEN 4 THEN ''ui-darkness''
WHEN 5 THEN ''le-frog''
WHEN 6 THEN ''black-tie''
```

```
  ELSE ''ui-lightness''
  END)
  + N''","dateFormat":"'' + CASE (PersonID % 10)
  WHEN 0 THEN ''mm/dd/yy''
  WHEN 1 THEN ''yy-mm-dd''
  WHEN 2 THEN ''dd/mm/yy''
  WHEN 3 THEN ''DD, MM d, yy''
  WHEN 4 THEN ''dd/mm/yy''
  WHEN 5 THEN ''dd/mm/yy''
  WHEN 6 THEN ''mm/dd/yy''
  ELSE ''mm/dd/yy''
  END
  + N''","timeZone": "PST"''
  + N'',"table":{"pagingType":"'' + CASE (PersonID % 5)
  WHEN 0 THEN ''numbers''
  WHEN 1 THEN ''full_numbers''
  WHEN 2 THEN ''full''
  WHEN 3 THEN ''simple_numbers''
  ELSE ''simple''
  END
  + N''","pageLength": '' + CASE (PersonID % 5)
  WHEN 0 THEN ''10''
  WHEN 1 THEN ''25''
  WHEN 2 THEN ''50''
  WHEN 3 THEN ''10''
  ELSE ''10''
  END + N''},"favoritesOnDashboard":true}'',
  ValidFrom = @StartingWhen
  WHERE UserPreferences IS NOT NULL;
 END;';
 EXECUTE (@SQL);
END;

IF NOT EXISTS (SELECT 1 FROM sys.procedures WHERE SCHEMA_NAME(schema_id) = N'DataLoadSimulation'
                                          AND name = N'
                                          DailyProcessToCreateHistory')
BEGIN
 SET @SQL = N'
 CREATE PROCEDURE DataLoadSimulation.DailyProcessToCreateHistory
 @StartDate date,
 @EndDate date,
 @AverageNumberOfCustomerOrdersPerDay int,
 @SaturdayPercentageOfNormalWorkDay int,
 @SundayPercentageOfNormalWorkDay int,
 @UpdateCustomFields bit,
 @IsSilentMode bit,
 @AreDatesPrinted bit
 AS
 BEGIN
  SET NOCOUNT ON;

  DECLARE @CurrentDateTime datetime2(7) = @StartDate;
  DECLARE @EndOfTime datetime2(7) = ''99991231 23:59:59.9999999'';
  DECLARE @StartingWhen datetime;
  DECLARE @NumberOfCustomerOrders int;
  DECLARE @IsWeekday bit;
  DECLARE @IsSaturday bit;
  DECLARE @IsSunday bit;
  DECLARE @IsMonday bit;
  DECLARE @Weekday int;
  DECLARE @IsStaffOnly bit;

  SET DATEFIRST 7; -- Week begins on Sunday

  EXEC DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad;

  WHILE @CurrentDateTime <= @EndDate
  BEGIN
  IF @AreDatesPrinted <> 0 OR @IsSilentMode = 0
  BEGIN
  PRINT SUBSTRING(DATENAME(weekday, @CurrentDateTime), 1,3) + N'' '' + CONVERT(nvarchar(20), @
 CurrentDateTime, 107);  PRINT N'' '';
  END;

  -- Calculate the days of the week - different processing happens on each day
  SET @Weekday = DATEPART(weekday, @CurrentDateTime);
  SET @IsSaturday = 0;
  SET @IsSunday = 0;
```

```
 SET @IsMonday = 0;
 SET @IsWeekday = 1;

 IF @Weekday = 7
 BEGIN
 SET @IsSaturday = 1;
 SET @IsWeekday = 0;
 END;
 IF @Weekday = 1
 BEGIN
 SET @IsSunday = 1;
 SET @IsWeekday = 0;
 END;
 IF @Weekday = 2 SET @IsMonday = 1;

 -- Purchase orders
 IF @IsWeekday <> 0
 BEGIN
 -- Start receiving purchase orders at 7AM on weekdays
 SET @StartingWhen = DATEADD(hour, 7, @CurrentDateTime);
 EXEC DataLoadSimulation.ReceivePurchaseOrders @CurrentDateTime, @StartingWhen, @EndOfTime, @
IsSilentMode;  END;

 -- Password changes
 SET @StartingWhen = DATEADD(hour, 8, @CurrentDateTime);
 EXEC DataLoadSimulation.ChangePasswords @CurrentDateTime, @StartingWhen, @EndOfTime, @IsSilentMode;

 -- Activate new website users
 SET @StartingWhen = DATEADD(minute, 10, DATEADD(hour, 8, @CurrentDateTime));
 EXEC DataLoadSimulation.ActivateWebsiteLogons @CurrentDateTime, @StartingWhen, @EndOfTime, @IsSilentMode;

 -- Payments to suppliers
 IF DATEPART(weekday, @CurrentDateTime) = 2
 BEGIN
 SET @StartingWhen = DATEADD(hour, 9, @CurrentDateTime); -- Suppliers are paid on Monday mornings
 EXEC DataLoadSimulation.PaySuppliers @CurrentDateTime, @StartingWhen, @EndOfTime, @IsSilentMode;
 END;

 -- Customer orders received
 SET @StartingWhen = DATEADD(hour, 10, @CurrentDateTime);
 SET @NumberOfCustomerOrders = @AverageNumberOfCustomerOrdersPerDay / 2
 + CEILING(RAND() * @AverageNumberOfCustomerOrdersPerDay);
 SET @NumberOfCustomerOrders = CASE DATEPART(weekday, @CurrentDateTime)
 WHEN 7
 THEN FLOOR(@NumberOfCustomerOrders * @
SaturdayPercentageOfNormalWorkDay / 100)  WHEN 1
 THEN FLOOR(@NumberOfCustomerOrders * @SundayPercentageOfNormalWorkDay /
 100)  ELSE @NumberOfCustomerOrders
 END;
 SET @NumberOfCustomerOrders = FLOOR(@NumberOfCustomerOrders * CASE WHEN YEAR(@StartingWhen) = 2013 THEN 1.0
 WHEN YEAR(@StartingWhen) = 2014 THEN 1.12
 WHEN YEAR(@StartingWhen) = 2015 THEN 1.21
 WHEN YEAR(@StartingWhen) = 2016 THEN 1.23
 ELSE 1.26
 END);
 EXEC DataLoadSimulation.CreateCustomerOrders @CurrentDateTime, @StartingWhen, @EndOfTime, @
NumberOfCustomerOrders, @IsSilentMode;
 -- Pick any customer orders that can be picked
 SET @StartingWhen = DATEADD(hour, 11, @CurrentDateTime);
 EXEC DataLoadSimulation.PickStockForCustomerOrders @CurrentDateTime, @StartingWhen, @EndOfTime, @
IsSilentMode;
 -- Process any payments from customers
 IF @Weekday <> 0
 BEGIN
 SET @StartingWhen = DATEADD(minute, 30, DATEADD(hour, 11, @CurrentDateTime));
 EXEC DataLoadSimulation.ProcessCustomerPayments @CurrentDateTime, @StartingWhen, @EndOfTime, @
IsSilentMode;  END;

 -- Invoice orders that have been fully picked
 SET @StartingWhen = DATEADD(hour, 12, @CurrentDateTime);
 EXEC DataLoadSimulation.InvoicePickedOrders @CurrentDateTime, @StartingWhen, @EndOfTime, @IsSilentMode;

 -- Place supplier orders
 IF @Weekday <> 0
 BEGIN
 SET @StartingWhen = DATEADD(hour, 13, @CurrentDateTime);
 EXEC DataLoadSimulation.PlaceSupplierOrders @CurrentDateTime, @StartingWhen, @EndOfTime, @
IsSilentMode;  END;
```

```
    -- End of quarter stock take
    IF (MONTH(@CurrentDateTime) = 1 AND DAY(@CurrentDateTime) = 31)
    OR (MONTH(@CurrentDateTime) = 4 AND DAY(@CurrentDateTime) = 30)
    OR (MONTH(@CurrentDateTime) = 7 AND DAY(@CurrentDateTime) = 31)
    OR (MONTH(@CurrentDateTime) = 10 AND DAY(@CurrentDateTime) = 31)
    BEGIN
    SET @StartingWhen = DATEADD(hour, 14, @CurrentDateTime);
    EXEC DataLoadSimulation.PerformStocktake @CurrentDateTime, @StartingWhen, @EndOfTime, @IsSilentMode;
    END;

    -- Record invoice deliveries
    SET @StartingWhen = DATEADD(hour, 7, @CurrentDateTime);
    EXEC DataLoadSimulation.RecordInvoiceDeliveries @CurrentDateTime, @StartingWhen, @EndOfTime, @
   IsSilentMode;
    -- Add customers
    IF @Weekday <> 0
    BEGIN
    SET @StartingWhen = DATEADD(hour, 15, @CurrentDateTime);
    EXEC DataLoadSimulation.AddCustomers @CurrentDateTime, @StartingWhen, @EndOfTime, @IsSilentMode;
    END;

    -- Add stock items
    SET @StartingWhen = DATEADD(hour, 16, @CurrentDateTime);
    EXEC DataLoadSimulation.AddStockItems @CurrentDateTime, @StartingWhen, @EndOfTime, @IsSilentMode;

    -- Add special deals
    SET @StartingWhen = DATEADD(hour, 16, @CurrentDateTime);
    EXEC DataLoadSimulation.AddSpecialDeals @CurrentDateTime, @StartingWhen, @EndOfTime, @IsSilentMode;

    -- Temporal changes
    SET @StartingWhen = DATEADD(hour, 16, @CurrentDateTime);
    EXEC DataLoadSimulation.MakeTemporalChanges @CurrentDateTime, @StartingWhen, @EndOfTime, @IsSilentMode;

    -- Record delivery van temperatures
    IF @CurrentDateTime >= ''20160101''
    BEGIN
    SET @StartingWhen = DATEADD(hour, 7, @CurrentDateTime);
    EXEC DataLoadSimulation.RecordDeliveryVanTemperatures 300, 2, @CurrentDateTime, @StartingWhen, @
   IsSilentMode;  END;

    -- Record cold room temperatures
    IF @CurrentDateTime >= ''20151220''
    BEGIN
    EXEC DataLoadSimulation.RecordColdRoomTemperatures 30, 4, @CurrentDateTime, @EndOfTime, @IsSilentMode;
    END;

    IF @IsSilentMode = 0
    BEGIN
    PRINT N'' '';
    END;

    SET @CurrentDateTime = DATEADD(day, 1, @CurrentDateTime);
    END; -- of processing each day

    IF @UpdateCustomFields <> 0
    BEGIN
    EXEC DataLoadSimulation.UpdateCustomFields @EndDate;
    END;

    EXEC DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad;

    EXEC Sequences.ReseedAllSequences;

    -- Ensure RLS is applied
    EXEC [Application].Configuration_ApplyRowLevelSecurity
   END;';
   EXECUTE (@SQL);
  END;

  EXEC DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad;
END;
GO
```

Depends On 3

- DataLoadSimulation
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad

DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad

Used By 1

DataLoadSimulation.PopulateDataToCurrentDate

DataLoadSimulation.Configuration_RemoveDataLoadSimulationProcedures

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE DataLoadSimulation.Configuration_RemoveDataLoadSimulationProcedures
WITH EXECUTE AS OWNER
AS
BEGIN
 SET NOCOUNT ON;
 SET XACT_ABORT ON;

 DROP PROCEDURE IF EXISTS DataLoadSimulation.ActivateWebsiteLogons;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.AddCustomers;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.AddSpecialDeals;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.AddStockItems;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.ChangePasswords;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.CreateCustomerOrders;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.DailyProcessToCreateHistory;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.InvoicePickedOrders;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.MakeTemporalChanges;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.PaySuppliers;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.PerformStocktake;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.PickStockForCustomerOrders;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.PlaceSupplierOrders;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.ProcessCustomerPayments;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.ReceivePurchaseOrders;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.RecordColdRoomTemperatures;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.RecordDeliveryVanTemperatures;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.RecordInvoiceDeliveries;
 DROP PROCEDURE IF EXISTS DataLoadSimulation.UpdateCustomFields;
 DROP FUNCTION IF EXISTS DataLoadSimulation.GetAreaCode;
END;
GO
```

Depends On 1

DataLoadSimulation

Used By 1

DataLoadSimulation.PopulateDataToCurrentDate

DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
AS BEGIN
    -- Disables the temporal nature of the temporal tables before a simulated data load
    SET NOCOUNT ON;

    IF EXISTS (SELECT 1 FROM sys.procedures WHERE name = N'Configuration_RemoveRowLevelSecurity')
    BEGIN
        EXEC [Application].Configuration_RemoveRowLevelSecurity;
    END;

    DECLARE @SQL nvarchar(max) = N'';
    DECLARE @CrLf nvarchar(2) = NCHAR(13) + NCHAR(10);
    DECLARE @Indent nvarchar(4) = N' ';
    DECLARE @SchemaName sysname;
    DECLARE @TableName sysname;
    DECLARE @NormalColumnList nvarchar(max);
    DECLARE @NormalColumnListWithDPrefix nvarchar(max);
    DECLARE @PrimaryKeyColumn sysname;
    DECLARE @TemporalFromColumnName sysname = N'ValidFrom';
    DECLARE @TemporalToColumnName sysname = N'ValidTo';
    DECLARE @TemporalTableSuffix nvarchar(max) = N'Archive';
    DECLARE @LastEditedByColumnName sysname;

    ALTER TABLE [Application].[Cities] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Application].[Cities] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Application].[Countries] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Application].[Countries] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Application].[DeliveryMethods] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Application].[DeliveryMethods] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Application].[PaymentMethods] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Application].[PaymentMethods] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Application].[People] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Application].[People] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Application].[StateProvinces] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Application].[StateProvinces] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Application].[TransactionTypes] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Application].[TransactionTypes] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Purchasing].[SupplierCategories] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Purchasing].[SupplierCategories] DROP PERIOD FOR SYSTEM_TIME;
```

```
    ALTER TABLE [Purchasing].[Suppliers] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Purchasing].[Suppliers] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Sales].[BuyingGroups] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Sales].[BuyingGroups] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Sales].[CustomerCategories] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Sales].[CustomerCategories] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Sales].[Customers] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Sales].[Customers] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Warehouse].[ColdRoomTemperatures] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Warehouse].[ColdRoomTemperatures] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Warehouse].[Colors] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Warehouse].[Colors] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Warehouse].[PackageTypes] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Warehouse].[PackageTypes] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Warehouse].[StockGroups] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Warehouse].[StockGroups] DROP PERIOD FOR SYSTEM_TIME;

    ALTER TABLE [Warehouse].[StockItems] SET (SYSTEM_VERSIONING = OFF);
    ALTER TABLE [Warehouse].[StockItems] DROP PERIOD FOR SYSTEM_TIME;

    SET @SQL = N'';
    SET @SchemaName = N'Application';
    SET @TableName = N'Cities';
    SET @PrimaryKeyColumn = N'CityID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [CityID], [CityName], [StateProvinceID], [Location], [LatestRecordedPopulation],';
    SET @NormalColumnListWithDPrefix = N' d.[CityID], d.[CityName], d.[StateProvinceID], d.[Location], d.[
    LatestRecordedPopulation],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
              + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
              + N'AFTER INSERT, UPDATE' + @CrLf
              + N'AS' + @CrLf
              + N'BEGIN' + @CrLf
              + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
              + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
              + @Indent + N'BEGIN' + @CrLf
              + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                  + N' must be updated when simulating data loads'
                                  ', 1;' + @CrLf
              + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
              + @Indent + N'END;' + @Crlf + @CrLf
              + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
              + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
              QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                  + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                  @CrLf
              + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
              + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                  + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                  ' + QUOTENAME(@TemporalFromColumnName) +
                                  @CrLf
              + @Indent + N'FROM inserted AS i' + @CrLf
              + @Indent + N'INNER JOIN deleted AS d' + @CrLf
              + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
              + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Application';
```

```
    SET @TableName = N'Countries';
    SET @PrimaryKeyColumn = N'CountryID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [CountryID], [CountryName], [FormalName], [IsoAlpha3Code], [IsoNumericCode], [
    CountryType], [LatestRecordedPopulation], [Continent], [Region], [Subregion], [Border],';
    SET @NormalColumnListWithDPrefix = N' d.[CountryID], d.[CountryName], d.[FormalName], d.[IsoAlpha3Code], d.[
    IsoNumericCode], d.[CountryType], d.[LatestRecordedPopulation], d.[Continent], d.[Region], d.[Subregion], d.[
    Border],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
                + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
                + N'AFTER INSERT, UPDATE' + @CrLf
                + N'AS' + @CrLf
                + N'BEGIN' + @CrLf
                + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
                + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
                + @Indent + N'BEGIN' + @CrLf
                + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                    + N' must be updated when simulating data loads'
                                    ', 1;' + @CrLf
                + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
                + @Indent + N'END;' + @Crlf + @CrLf
                + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
                + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
                QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                    @CrLf
                + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
                + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                    ' + QUOTENAME(@TemporalFromColumnName) +
                                    @CrLf
                + @Indent + N'FROM inserted AS i' + @CrLf
                + @Indent + N'INNER JOIN deleted AS d' + @CrLf
                + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
                + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Application';
    SET @TableName = N'DeliveryMethods';
    SET @PrimaryKeyColumn = N'DeliveryMethodID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [DeliveryMethodID], [DeliveryMethodName],';
    SET @NormalColumnListWithDPrefix = N' d.[DeliveryMethodID], d.[DeliveryMethodName],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
                + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
                + N'AFTER INSERT, UPDATE' + @CrLf
                + N'AS' + @CrLf
                + N'BEGIN' + @CrLf
                + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
                + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
                + @Indent + N'BEGIN' + @CrLf
                + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                    + N' must be updated when simulating data loads'
                                    ', 1;' + @CrLf
                + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
                + @Indent + N'END;' + @Crlf + @CrLf
                + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
                + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
```

```
                QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                    @CrLf
                + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
                + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                    ' + QUOTENAME(@TemporalFromColumnName) +
                                    @CrLf
                + @Indent + N'FROM inserted AS i' + @CrLf
                + @Indent + N'INNER JOIN deleted AS d' + @CrLf
                + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
                + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Application';
    SET @TableName = N'PaymentMethods';
    SET @PrimaryKeyColumn = N'PaymentMethodID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [PaymentMethodID], [PaymentMethodName],';
    SET @NormalColumnListWithDPrefix = N' d.[PaymentMethodID], d.[PaymentMethodName],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
                + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
                + N'AFTER INSERT, UPDATE' + @CrLf
                + N'AS' + @CrLf
                + N'BEGIN' + @CrLf
                + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
                + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
                + @Indent + N'BEGIN' + @CrLf
                + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                    + N' must be updated when simulating data loads'
                                    ', 1;' + @CrLf
                + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
                + @Indent + N'END;' + @Crlf + @CrLf
                + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
                + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
                QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                    @CrLf
                + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
                + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                    ' + QUOTENAME(@TemporalFromColumnName) +
                                    @CrLf
                + @Indent + N'FROM inserted AS i' + @CrLf
                + @Indent + N'INNER JOIN deleted AS d' + @CrLf
                + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
                + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Application';
    SET @TableName = N'People';
    SET @PrimaryKeyColumn = N'PersonID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [PersonID], [FullName], [PreferredName], [SearchName], [IsPermittedToLogon], [
    LogonName], [IsExternalLogonProvider], [HashedPassword], [IsSystemUser], [IsEmployee], [IsSalesperson], [
    UserPreferences], [PhoneNumber], [FaxNumber], [EmailAddress], [Photo], [CustomFields], [OtherLanguages],';
    SET @NormalColumnListWithDPrefix = N' d.[PersonID], d.[FullName], d.[PreferredName], d.[SearchName], d.[
    IsPermittedToLogon], d.[LogonName], d.[IsExternalLogonProvider], d.[HashedPassword], d.[IsSystemUser], d.[
    IsEmployee], d.[IsSalesperson], d.[UserPreferences], d.[PhoneNumber], d.[FaxNumber], d.[EmailAddress], d.[Photo]
    , d.[CustomFields], d.[OtherLanguages],';
```

```
    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
                + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
                + N'AFTER INSERT, UPDATE' + @CrLf
                + N'AS' + @CrLf
                + N'BEGIN' + @CrLf
                + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
                + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
                + @Indent + N'BEGIN' + @CrLf
                + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                    + N' must be updated when simulating data loads'
                                    ', 1;' + @CrLf
                + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
                + @Indent + N'END;' + @Crlf + @CrLf
                + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
                + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
                QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                    @CrLf
                + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
                + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                    ' + QUOTENAME(@TemporalFromColumnName) +
                                    @CrLf
                + @Indent + N'FROM inserted AS i' + @CrLf
                + @Indent + N'INNER JOIN deleted AS d' + @CrLf
                + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
                + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Application';
    SET @TableName = N'StateProvinces';
    SET @PrimaryKeyColumn = N'StateProvinceID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [StateProvinceID], [StateProvinceCode], [StateProvinceName], [CountryID], [
    SalesTerritory], [Border], [LatestRecordedPopulation],';
    SET @NormalColumnListWithDPrefix = N' d.[StateProvinceID], d.[StateProvinceCode], d.[StateProvinceName], d.[
    CountryID], d.[SalesTerritory], d.[Border], d.[LatestRecordedPopulation],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
                + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
                + N'AFTER INSERT, UPDATE' + @CrLf
                + N'AS' + @CrLf
                + N'BEGIN' + @CrLf
                + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
                + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
                + @Indent + N'BEGIN' + @CrLf
                + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                    + N' must be updated when simulating data loads'
                                    ', 1;' + @CrLf
                + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
                + @Indent + N'END;' + @Crlf + @CrLf
                + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
                + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
                QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                    @CrLf
                + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
                + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
```

```
                                + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                ' + QUOTENAME(@TemporalFromColumnName) +
                                @CrLf
                + @Indent + N'FROM inserted AS i' + @CrLf
                + @Indent + N'INNER JOIN deleted AS d' + @CrLf
                + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
                + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Application';
    SET @TableName = N'TransactionTypes';
    SET @PrimaryKeyColumn = N'TransactionTypeID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [TransactionTypeID], [TransactionTypeName],';
    SET @NormalColumnListWithDPrefix = N' d.[TransactionTypeID], d.[TransactionTypeName],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
                + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
                + N'AFTER INSERT, UPDATE' + @CrLf
                + N'AS' + @CrLf
                + N'BEGIN' + @CrLf
                + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
                + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
                + @Indent + N'BEGIN' + @CrLf
                + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                + N' must be updated when simulating data loads'
                                ', 1;' + @CrLf
                + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
                + @Indent + N'END;' + @Crlf + @CrLf
                + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
                + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
                QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                @CrLf
                + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
                + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                ' + QUOTENAME(@TemporalFromColumnName) +
                                @CrLf
                + @Indent + N'FROM inserted AS i' + @CrLf
                + @Indent + N'INNER JOIN deleted AS d' + @CrLf
                + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
                + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Purchasing';
    SET @TableName = N'SupplierCategories';
    SET @PrimaryKeyColumn = N'SupplierCategoryID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [SupplierCategoryID], [SupplierCategoryName],';
    SET @NormalColumnListWithDPrefix = N' d.[SupplierCategoryID], d.[SupplierCategoryName],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
                + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
                + N'AFTER INSERT, UPDATE' + @CrLf
                + N'AS' + @CrLf
                + N'BEGIN' + @CrLf
```

```
                + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
                + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
                + @Indent + N'BEGIN' + @CrLf
                + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                    + N' must be updated when simulating data loads'
                                    ', 1;' + @CrLf
                + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
                + @Indent + N'END;' + @Crlf + @CrLf
                + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
                + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
                QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                    @CrLf
                + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
                + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                    ' + QUOTENAME(@TemporalFromColumnName) +
                                    @CrLf
                + @Indent + N'FROM inserted AS i' + @CrLf
                + @Indent + N'INNER JOIN deleted AS d' + @CrLf
                + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
                + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Purchasing';
    SET @TableName = N'Suppliers';
    SET @PrimaryKeyColumn = N'SupplierID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [SupplierID], [SupplierName], [SupplierCategoryID], [PrimaryContactPersonID], [
    AlternateContactPersonID], [DeliveryMethodID], [DeliveryCityID], [PostalCityID], [SupplierReference], [
    BankAccountName], [BankAccountBranch], [BankAccountCode], [BankAccountNumber], [BankInternationalCode], [
    PaymentDays], [InternalComments], [PhoneNumber], [FaxNumber], [WebsiteURL], [DeliveryAddressLine1], [
    DeliveryAddressLine2], [DeliveryPostalCode], [DeliveryLocation], [PostalAddressLine1], [PostalAddressLine2], [
    PostalPostalCode],';
    SET @NormalColumnListWithDPrefix = N' d.[SupplierID], d.[SupplierName], d.[SupplierCategoryID], d.[
    PrimaryContactPersonID], d.[AlternateContactPersonID], d.[DeliveryMethodID], d.[DeliveryCityID], d.[
    PostalCityID], d.[SupplierReference], d.[BankAccountName], d.[BankAccountBranch], d.[BankAccountCode], d.[
    BankAccountNumber], d.[BankInternationalCode], d.[PaymentDays], d.[InternalComments], d.[PhoneNumber], d.[
    FaxNumber], d.[WebsiteURL], d.[DeliveryAddressLine1], d.[DeliveryAddressLine2], d.[DeliveryPostalCode], d.[
    DeliveryLocation], d.[PostalAddressLine1], d.[PostalAddressLine2], d.[PostalPostalCode],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
                + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
                + N'AFTER INSERT, UPDATE' + @CrLf
                + N'AS' + @CrLf
                + N'BEGIN' + @CrLf
                + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
                + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
                + @Indent + N'BEGIN' + @CrLf
                + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                    + N' must be updated when simulating data loads'
                                    ', 1;' + @CrLf
                + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
                + @Indent + N'END;' + @Crlf + @CrLf
                + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
                + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
                QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                    @CrLf
                + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
                + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                    ' + QUOTENAME(@TemporalFromColumnName) +
                                    @CrLf
```

```
                + @Indent + N'FROM inserted AS i' + @CrLf
                + @Indent + N'INNER JOIN deleted AS d' + @CrLf
                + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
                + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Sales';
    SET @TableName = N'BuyingGroups';
    SET @PrimaryKeyColumn = N'BuyingGroupID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [BuyingGroupID], [BuyingGroupName],';
    SET @NormalColumnListWithDPrefix = N' d.[BuyingGroupID], d.[BuyingGroupName],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
                + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
                + N'AFTER INSERT, UPDATE' + @CrLf
                + N'AS' + @CrLf
                + N'BEGIN' + @CrLf
                + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
                + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
                + @Indent + N'BEGIN' + @CrLf
                + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                    + N' must be updated when simulating data loads'
                                    ', 1;' + @CrLf
                + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
                + @Indent + N'END;' + @Crlf + @CrLf
                + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
                + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
                QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                    @CrLf
                + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
                + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                    ' + QUOTENAME(@TemporalFromColumnName) +
                                    @CrLf
                + @Indent + N'FROM inserted AS i' + @CrLf
                + @Indent + N'INNER JOIN deleted AS d' + @CrLf
                + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
                + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Sales';
    SET @TableName = N'CustomerCategories';
    SET @PrimaryKeyColumn = N'CustomerCategoryID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [CustomerCategoryID], [CustomerCategoryName],';
    SET @NormalColumnListWithDPrefix = N' d.[CustomerCategoryID], d.[CustomerCategoryName],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
                + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
                + N'AFTER INSERT, UPDATE' + @CrLf
                + N'AS' + @CrLf
                + N'BEGIN' + @CrLf
                + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
                + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
                + @Indent + N'BEGIN' + @CrLf
```

```
            + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                + N' must be updated when simulating data loads'
                                ', 1;' + @CrLf
            + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
            + @Indent + N'END;' + @Crlf + @CrLf
            + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
            + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
            QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                @CrLf
            + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
            + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                ' + QUOTENAME(@TemporalFromColumnName) +
                                @CrLf
            + @Indent + N'FROM inserted AS i' + @CrLf
            + @Indent + N'INNER JOIN deleted AS d' + @CrLf
            + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
            + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Sales';
    SET @TableName = N'Customers';
    SET @PrimaryKeyColumn = N'CustomerID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [CustomerID], [CustomerName], [BillToCustomerID], [CustomerCategoryID], [
    BuyingGroupID], [PrimaryContactPersonID], [AlternateContactPersonID], [DeliveryMethodID], [DeliveryCityID], [
    PostalCityID], [CreditLimit], [AccountOpenedDate], [StandardDiscountPercentage], [IsStatementSent], [
    IsOnCreditHold], [PaymentDays], [PhoneNumber], [FaxNumber], [DeliveryRun], [RunPosition], [WebsiteURL], [
    DeliveryAddressLine1], [DeliveryAddressLine2], [DeliveryPostalCode], [DeliveryLocation], [PostalAddressLine1], [
    PostalAddressLine2], [PostalPostalCode],';
    SET @NormalColumnListWithDPrefix = N' d.[CustomerID], d.[CustomerName], d.[BillToCustomerID], d.[
    CustomerCategoryID], d.[BuyingGroupID], d.[PrimaryContactPersonID], d.[AlternateContactPersonID], d.[
    DeliveryMethodID], d.[DeliveryCityID], d.[PostalCityID], d.[CreditLimit], d.[AccountOpenedDate], d.[
    StandardDiscountPercentage], d.[IsStatementSent], d.[IsOnCreditHold], d.[PaymentDays], d.[PhoneNumber], d.[
    FaxNumber], d.[DeliveryRun], d.[RunPosition], d.[WebsiteURL], d.[DeliveryAddressLine1], d.[DeliveryAddressLine2]
    , d.[DeliveryPostalCode], d.[DeliveryLocation], d.[PostalAddressLine1], d.[PostalAddressLine2], d.[
    PostalPostalCode],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
            + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
            + N'AFTER INSERT, UPDATE' + @CrLf
            + N'AS' + @CrLf
            + N'BEGIN' + @CrLf
            + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
            + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
            + @Indent + N'BEGIN' + @CrLf
            + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                + N' must be updated when simulating data loads'
                                ', 1;' + @CrLf
            + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
            + @Indent + N'END;' + @Crlf + @CrLf
            + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
            + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
            QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                @CrLf
            + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
            + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                ' + QUOTENAME(@TemporalFromColumnName) +
                                @CrLf
            + @Indent + N'FROM inserted AS i' + @CrLf
            + @Indent + N'INNER JOIN deleted AS d' + @CrLf
```

```
            + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
            + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Warehouse';
    SET @TableName = N'ColdRoomTemperatures';
    SET @PrimaryKeyColumn = N'ColdRoomTemperatureID';
    SET @LastEditedByColumnName = N'';
    SET @NormalColumnList = N' [ColdRoomTemperatureID], [ColdRoomSensorNumber], [RecordedWhen], [Temperature],';
    SET @NormalColumnListWithDPrefix = N' d.[ColdRoomTemperatureID], d.[ColdRoomSensorNumber], d.[RecordedWhen],
     d.[Temperature],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
            + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
            + N'AFTER INSERT, UPDATE' + @CrLf
            + N'AS' + @CrLf
            + N'BEGIN' + @CrLf
            + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
            + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
            + @Indent + N'BEGIN' + @CrLf
            + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                        + N' must be updated when simulating data loads'
                        ', 1;' + @CrLf
            + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
            + @Indent + N'END;' + @Crlf + @CrLf
            + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
            + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
            QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                        + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                        @CrLf
            + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
            + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                        + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                        ' + QUOTENAME(@TemporalFromColumnName) +
                        @CrLf
            + @Indent + N'FROM inserted AS i' + @CrLf
            + @Indent + N'INNER JOIN deleted AS d' + @CrLf
            + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
            + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Warehouse';
    SET @TableName = N'Colors';
    SET @PrimaryKeyColumn = N'ColorID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [ColorID], [ColorName],';
    SET @NormalColumnListWithDPrefix = N' d.[ColorID], d.[ColorName],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
            + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
            + N'AFTER INSERT, UPDATE' + @CrLf
            + N'AS' + @CrLf
            + N'BEGIN' + @CrLf
            + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
            + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
            + @Indent + N'BEGIN' + @CrLf
            + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
```

```
                                + N' must be updated when simulating data loads'
                                ', 1;' + @CrLf
                + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
                + @Indent + N'END;' + @Crlf + @CrLf
                + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
                + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
                QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                @CrLf
                + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
                + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                ' + QUOTENAME(@TemporalFromColumnName) +
                                @CrLf
                + @Indent + N'FROM inserted AS i' + @CrLf
                + @Indent + N'INNER JOIN deleted AS d' + @CrLf
                + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
                + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Warehouse';
    SET @TableName = N'PackageTypes';
    SET @PrimaryKeyColumn = N'PackageTypeID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [PackageTypeID], [PackageTypeName],';
    SET @NormalColumnListWithDPrefix = N' d.[PackageTypeID], d.[PackageTypeName],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
                + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
                + N'AFTER INSERT, UPDATE' + @CrLf
                + N'AS' + @CrLf
                + N'BEGIN' + @CrLf
                + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
                + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
                + @Indent + N'BEGIN' + @CrLf
                + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                + N' must be updated when simulating data loads'
                                ', 1;' + @CrLf
                + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
                + @Indent + N'END;' + @Crlf + @CrLf
                + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
                + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
                QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                @CrLf
                + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
                + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                ' + QUOTENAME(@TemporalFromColumnName) +
                                @CrLf
                + @Indent + N'FROM inserted AS i' + @CrLf
                + @Indent + N'INNER JOIN deleted AS d' + @CrLf
                + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
                + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Warehouse';
    SET @TableName = N'StockGroups';
    SET @PrimaryKeyColumn = N'StockGroupID';
```

```
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [StockGroupID], [StockGroupName],';
    SET @NormalColumnListWithDPrefix = N' d.[StockGroupID], d.[StockGroupName],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
                + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
                + N'AFTER INSERT, UPDATE' + @CrLf
                + N'AS' + @CrLf
                + N'BEGIN' + @CrLf
                + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
                + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
                + @Indent + N'BEGIN' + @CrLf
                + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                    + N' must be updated when simulating data loads'
                                    ', 1;' + @CrLf
                + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
                + @Indent + N'END;' + @Crlf + @CrLf
                + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
                + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
                QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                    @CrLf
                + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
                + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                    + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                    ' + QUOTENAME(@TemporalFromColumnName) +
                                    @CrLf
                + @Indent + N'FROM inserted AS i' + @CrLf
                + @Indent + N'INNER JOIN deleted AS d' + @CrLf
                + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
                + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

    SET @SQL = N'';
    SET @SchemaName = N'Warehouse';
    SET @TableName = N'StockItems';
    SET @PrimaryKeyColumn = N'StockItemID';
    SET @LastEditedByColumnName = N'LastEditedBy';
    SET @NormalColumnList = N' [StockItemID], [StockItemName], [SupplierID], [ColorID], [UnitPackageID], [
    OuterPackageID], [Brand], [Size], [LeadTimeDays], [QuantityPerOuter], [IsChillerStock], [Barcode], [TaxRate], [
    UnitPrice], [RecommendedRetailPrice], [TypicalWeightPerUnit], [MarketingComments], [InternalComments], [Photo],
    [CustomFields], [Tags], [SearchDetails],';
    SET @NormalColumnListWithDPrefix = N' d.[StockItemID], d.[StockItemName], d.[SupplierID], d.[ColorID], d.[
    UnitPackageID], d.[OuterPackageID], d.[Brand], d.[Size], d.[LeadTimeDays], d.[QuantityPerOuter], d.[
    IsChillerStock], d.[Barcode], d.[TaxRate], d.[UnitPrice], d.[RecommendedRetailPrice], d.[TypicalWeightPerUnit],
    d.[MarketingComments], d.[InternalComments], d.[Photo], d.[CustomFields], d.[Tags], d.[SearchDetails],';

    SET @SQL = N'DROP TRIGGER IF EXISTS ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify];'
    EXECUTE (@SQL);

    SET @SQL = N'CREATE TRIGGER ' + QUOTENAME(@SchemaName) + N'.
[TR_' + @SchemaName + N'_' + @TableName + N'_DataLoad_Modify]' + @CrLf
                + N'ON ' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + @CrLf
                + N'AFTER INSERT, UPDATE' + @CrLf
                + N'AS' + @CrLf
                + N'BEGIN' + @CrLf
                + @Indent + N'SET NOCOUNT ON;' + @CrLf + @CrLf
                + @Indent + N'IF NOT UPDATE(' + QUOTENAME(@TemporalFromColumnName) + N')' + @CrLf
                + @Indent + N'BEGIN' + @CrLf
                + @Indent + @Indent + N'THROW 51000, ''' + QUOTENAME(@TemporalFromColumnName)
                                    + N' must be updated when simulating data loads'
                                    ', 1;' + @CrLf
                + @Indent + @Indent + N'ROLLBACK TRAN;' + @CrLf
                + @Indent + N'END;' + @Crlf + @CrLf
                + @Indent + N'INSERT
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName + N'_' + @TemporalTableSuffix) + @CrLf
                + @Indent + @Indent + N'(' + @NormalColumnList + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  THEN
```

```
                QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                + QUOTENAME(@TemporalFromColumnName) + N',' + QUOTENAME(@TemporalToColumnName) +  +
                                @CrLf
                + @Indent + N'SELECT' + @NormalColumnListWithDPrefix + CASE WHEN COALESCE(@LastEditedByColumnName, ) <>  T
HEN
                + QUOTENAME(@LastEditedByColumnName) + N', ' ELSE N'' END
                                + N' d.' + QUOTENAME(@TemporalFromColumnName) + N', i.
                                ' + QUOTENAME(@TemporalFromColumnName) +
                                @CrLf
                + @Indent + N'FROM inserted AS i' + @CrLf
                + @Indent + N'INNER JOIN deleted AS d' + @CrLf
                + @Indent + N'ON i.' + QUOTENAME(@PrimaryKeyColumn) + N' =
d.' + QUOTENAME(@PrimaryKeyColumn) + N';' + @CrLf
                + N'END;';
    IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = @TableName AND is_memory_optimized <> 0)
    BEGIN
        EXECUTE (@SQL);
    END;

END;
GO
```

Depends On 19

- DataLoadSimulation
- Application.Configuration_RemoveRowLevelSecurity
- Application.Cities
- Application.Countries
- Application.DeliveryMethods
- Application.PaymentMethods
- Application.People
- Application.StateProvinces
- Application.TransactionTypes
- Purchasing.SupplierCategories
- Purchasing.Suppliers
- Sales.BuyingGroups
- Sales.CustomerCategories
- Sales.Customers
- Warehouse.ColdRoomTemperatures
- Warehouse.Colors
- Warehouse.PackageTypes
- Warehouse.StockGroups
- Warehouse.StockItems

Used By 1

- DataLoadSimulation.Configuration_ApplyDataLoadSimulationProcedures

DataLoadSimulation.PopulateDataToCurrentDate

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	
Assembly	

Parameters

Name	Data Type	Length	Description
@AverageNumberOfCustomerOrdersPerDay	int	4	
@SaturdayPercentageOfNormalWorkDay	int	4	
@SundayPercentageOfNormalWorkDay	int	4	
@IsSilentMode	bit	1	
@AreDatesPrinted	bit	1	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE DataLoadSimulation.PopulateDataToCurrentDate
@AverageNumberOfCustomerOrdersPerDay int,
@SaturdayPercentageOfNormalWorkDay int,
@SundayPercentageOfNormalWorkDay int,
@IsSilentMode bit,
@AreDatesPrinted bit
AS
BEGIN
    SET NOCOUNT ON;

 EXEC DataLoadSimulation.Configuration_ApplyDataLoadSimulationProcedures;

    DECLARE @CurrentMaximumDate date = COALESCE((SELECT MAX(OrderDate) FROM Sales.Orders), '20121231');
    DECLARE @StartingDate date = DATEADD(day, 1, @CurrentMaximumDate);
    DECLARE @EndingDate date = CAST(DATEADD(day, -1, SYSDATETIME()) AS date);

    EXEC DataLoadSimulation.DailyProcessToCreateHistory
        @StartDate = @StartingDate,
        @EndDate = @EndingDate,
        @AverageNumberOfCustomerOrdersPerDay = @AverageNumberOfCustomerOrdersPerDay,
        @SaturdayPercentageOfNormalWorkDay = @SaturdayPercentageOfNormalWorkDay,
        @SundayPercentageOfNormalWorkDay = @SundayPercentageOfNormalWorkDay,
        @UpdateCustomFields = 0, -- they were done in the initial load
        @IsSilentMode = @IsSilentMode,
        @AreDatesPrinted = @AreDatesPrinted;

 EXEC DataLoadSimulation.Configuration_RemoveDataLoadSimulationProcedures;
END;
GO
```

Depends On 4

- DataLoadSimulation
- DataLoadSimulation.Configuration_ApplyDataLoadSimulationProcedures
- Sales.Orders
- DataLoadSimulation.Configuration_RemoveDataLoadSimulationProcedures

Used By

No items found

DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad
AS BEGIN
    -- Re-enables the temporal nature of the temporal tables after a simulated data load
    SET NOCOUNT ON;

    IF EXISTS (SELECT 1 FROM sys.procedures WHERE name = N'Configuration_ApplyRowLevelSecurity')
    BEGIN
        EXEC [Application].Configuration_ApplyRowLevelSecurity;
    END;

    DROP TRIGGER IF EXISTS [Application].[TR_Application_Cities_DataLoad_Modify];
    ALTER TABLE [Application].[Cities] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Application].[Cities] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Application].[Cities_Archive],
    DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Application].[TR_Application_Countries_DataLoad_Modify];
    ALTER TABLE [Application].[Countries] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Application].[Countries] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Application].
    [Countries_Archive], DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Application].[TR_Application_DeliveryMethods_DataLoad_Modify];
    ALTER TABLE [Application].[DeliveryMethods] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Application].[DeliveryMethods] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Application].
    [DeliveryMethods_Archive], DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Application].[TR_Application_PaymentMethods_DataLoad_Modify];
    ALTER TABLE [Application].[PaymentMethods] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Application].[PaymentMethods] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Application].
    [PaymentMethods_Archive], DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Application].[TR_Application_People_DataLoad_Modify];
    ALTER TABLE [Application].[People] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Application].[People] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Application].[People_Archive],
    DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Application].[TR_Application_StateProvinces_DataLoad_Modify];
    ALTER TABLE [Application].[StateProvinces] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Application].[StateProvinces] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Application].
    [StateProvinces_Archive], DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Application].[TR_Application_TransactionTypes_DataLoad_Modify];
    ALTER TABLE [Application].[TransactionTypes] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Application].[TransactionTypes] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Application].
    [TransactionTypes_Archive], DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Purchasing].[TR_Purchasing_SupplierCategories_DataLoad_Modify];
```

```
    ALTER TABLE [Purchasing].[SupplierCategories] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Purchasing].[SupplierCategories] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Purchasing].
    [SupplierCategories_Archive], DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Purchasing].[TR_Purchasing_Suppliers_DataLoad_Modify];
    ALTER TABLE [Purchasing].[Suppliers] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Purchasing].[Suppliers] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Purchasing].
    [Suppliers_Archive], DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Sales].[TR_Sales_BuyingGroups_DataLoad_Modify];
    ALTER TABLE [Sales].[BuyingGroups] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Sales].[BuyingGroups] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Sales].[BuyingGroups_Archive],
    DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Sales].[TR_Sales_CustomerCategories_DataLoad_Modify];
    ALTER TABLE [Sales].[CustomerCategories] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Sales].[CustomerCategories] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Sales].
    [CustomerCategories_Archive], DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Sales].[TR_Sales_Customers_DataLoad_Modify];
    ALTER TABLE [Sales].[Customers] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Sales].[Customers] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Sales].[Customers_Archive],
    DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Warehouse].[TR_Warehouse_ColdRoomTemperatures_DataLoad_Modify];
    ALTER TABLE [Warehouse].[ColdRoomTemperatures] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Warehouse].[ColdRoomTemperatures] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Warehouse].
    [ColdRoomTemperatures_Archive], DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Warehouse].[TR_Warehouse_Colors_DataLoad_Modify];
    ALTER TABLE [Warehouse].[Colors] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Warehouse].[Colors] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Warehouse].[Colors_Archive],
    DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Warehouse].[TR_Warehouse_PackageTypes_DataLoad_Modify];
    ALTER TABLE [Warehouse].[PackageTypes] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Warehouse].[PackageTypes] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Warehouse].
    [PackageTypes_Archive], DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Warehouse].[TR_Warehouse_StockGroups_DataLoad_Modify];
    ALTER TABLE [Warehouse].[StockGroups] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Warehouse].[StockGroups] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Warehouse].
    [StockGroups_Archive], DATA_CONSISTENCY_CHECK = ON));

    DROP TRIGGER IF EXISTS [Warehouse].[TR_Warehouse_StockItems_DataLoad_Modify];
    ALTER TABLE [Warehouse].[StockItems] ADD PERIOD FOR SYSTEM_TIME([ValidFrom], [ValidTo]);
    ALTER TABLE [Warehouse].[StockItems] SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = [Warehouse].
    [StockItems_Archive], DATA_CONSISTENCY_CHECK = ON));

END;
GO
```

Depends On 36

- DataLoadSimulation
- Application.Configuration_ApplyRowLevelSecurity
- Application.Cities
- Application.Cities_Archive
- Application.Countries
- Application.Countries_Archive
- Application.DeliveryMethods
- Application.DeliveryMethods_Archive
- Application.PaymentMethods
- Application.PaymentMethods_Archive
- Application.People
- Application.People_Archive
- Application.StateProvinces
- Application.StateProvinces_Archive
- Application.TransactionTypes
- Application.TransactionTypes_Archive

- Purchasing.SupplierCategories
- Purchasing.SupplierCategories_Archive
- Purchasing.Suppliers
- Purchasing.Suppliers_Archive
- Sales.BuyingGroups
- Sales.BuyingGroups_Archive
- Sales.CustomerCategories
- Sales.CustomerCategories_Archive
- Sales.Customers
- Sales.Customers_Archive
- Warehouse.ColdRoomTemperatures
- Warehouse.ColdRoomTemperatures_Archive
- Warehouse.Colors
- Warehouse.Colors_Archive
- Warehouse.PackageTypes
- Warehouse.PackageTypes_Archive
- Warehouse.StockGroups
- Warehouse.StockGroups_Archive
- Warehouse.StockItems
- Warehouse.StockItems_Archive

Used By 1

- DataLoadSimulation.Configuration_ApplyDataLoadSimulationProcedures

Integration.GetCityUpdates

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@LastCutoff	datetime2	8	
@NewCutoff	datetime2	8	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Integration.GetCityUpdates
@LastCutoff datetime2(7),
@NewCutoff datetime2(7)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @EndOfTime datetime2(7) = '99991231 23:59:59.9999999';
    DECLARE @InitialLoadDate date = '20130101';

    CREATE TABLE #CityChanges
    (
        [WWI City ID] int,
        City nvarchar(50),
        [State Province] nvarchar(50),
        Country nvarchar(50),
        Continent nvarchar(30),
        [Sales Territory] nvarchar(50),
        Region nvarchar(30),
        Subregion nvarchar(30),
        [Location] geography,
        [Latest Recorded Population] bigint,
        [Valid From] datetime2(7),
        [Valid To] datetime2(7) NULL
    );

    DECLARE @CountryID int;
    DECLARE @StateProvinceID int;
    DECLARE @CityID int;
    DECLARE @ValidFrom datetime2(7);

    -- first need to find any country changes that have occurred since initial load

    DECLARE CountryChangeList CURSOR FAST_FORWARD READ_ONLY
```

```
FOR
SELECT co.CountryID,
       co.ValidFrom
FROM [Application].Countries_Archive AS co
WHERE co.ValidFrom > @LastCutoff
AND co.ValidFrom <= @NewCutoff
AND co.ValidFrom <> @InitialLoadDate
UNION ALL
SELECT co.CountryID,
       co.ValidFrom
FROM [Application].Countries AS co
WHERE co.ValidFrom > @LastCutoff
AND co.ValidFrom <= @NewCutoff
AND co.ValidFrom <> @InitialLoadDate
ORDER BY ValidFrom;

OPEN CountryChangeList;
FETCH NEXT FROM CountryChangeList INTO @CountryID, @ValidFrom;

WHILE @@FETCH_STATUS = 0
BEGIN
    INSERT #CityChanges
        ([WWI City ID], City, [State Province], Country, Continent, [Sales Territory], Region, Subregion,
         [Location], [Latest Recorded Population], [Valid From], [Valid To])
    SELECT c.CityID, c.CityName, sp.StateProvinceName, co.CountryName, co.Continent, sp.SalesTerritory, co.
    Region, co.Subregion,
           c.[Location], COALESCE(c.LatestRecordedPopulation, 0), @ValidFrom, NULL
    FROM [Application].Cities FOR SYSTEM_TIME AS OF @ValidFrom AS c
    INNER JOIN [Application].StateProvinces FOR SYSTEM_TIME AS OF @ValidFrom AS sp
    ON c.StateProvinceID = sp.StateProvinceID
    INNER JOIN [Application].Countries FOR SYSTEM_TIME AS OF @ValidFrom AS co
    ON sp.CountryID = co.CountryID
    WHERE co.CountryID = @CountryID;

    FETCH NEXT FROM CountryChangeList INTO @CountryID, @ValidFrom;
END;

CLOSE CountryChangeList;
DEALLOCATE CountryChangeList;

-- next need to find any stateprovince changes that have occurred since initial load

DECLARE StateProvinceChangeList CURSOR FAST_FORWARD READ_ONLY
FOR
SELECT sp.StateProvinceID,
       sp.ValidFrom
FROM [Application].StateProvinces_Archive AS sp
WHERE sp.ValidFrom > @LastCutoff
AND sp.ValidFrom <= @NewCutoff
AND sp.ValidFrom <> @InitialLoadDate
UNION ALL
SELECT sp.StateProvinceID,
       sp.ValidFrom
FROM [Application].StateProvinces AS sp
WHERE sp.ValidFrom > @LastCutoff
AND sp.ValidFrom <= @NewCutoff
AND sp.ValidFrom <> @InitialLoadDate
ORDER BY ValidFrom;

OPEN StateProvinceChangeList;
FETCH NEXT FROM StateProvinceChangeList INTO @StateProvinceID, @ValidFrom;

WHILE @@FETCH_STATUS = 0
BEGIN
    INSERT #CityChanges
        ([WWI City ID], City, [State Province], Country, Continent, [Sales Territory], Region, Subregion,
         [Location], [Latest Recorded Population], [Valid From], [Valid To])
    SELECT c.CityID, c.CityName, sp.StateProvinceName, co.CountryName, co.Continent, sp.SalesTerritory, co.
    Region, co.Subregion,
           c.[Location], COALESCE(c.LatestRecordedPopulation, 0), @ValidFrom, NULL
    FROM [Application].Cities FOR SYSTEM_TIME AS OF @ValidFrom AS c
    INNER JOIN [Application].StateProvinces FOR SYSTEM_TIME AS OF @ValidFrom AS sp
    ON c.StateProvinceID = sp.StateProvinceID
    INNER JOIN [Application].Countries FOR SYSTEM_TIME AS OF @ValidFrom AS co
    ON sp.CountryID = co.CountryID
    WHERE sp.StateProvinceID = @StateProvinceID;

    FETCH NEXT FROM StateProvinceChangeList INTO @StateProvinceID, @ValidFrom;
END;
```

```
    CLOSE StateProvinceChangeList;
    DEALLOCATE StateProvinceChangeList;

    -- finally need to find any city changes that have occurred, including during the initial load

    DECLARE CityChangeList CURSOR FAST_FORWARD READ_ONLY
    FOR
    SELECT c.CityID,
           c.ValidFrom
    FROM [Application].Cities_Archive AS c
    WHERE c.ValidFrom > @LastCutoff
    AND c.ValidFrom <= @NewCutoff
    UNION ALL
    SELECT c.CityID,
           c.ValidFrom
    FROM [Application].Cities AS c
    WHERE c.ValidFrom > @LastCutoff
    AND c.ValidFrom <= @NewCutoff
    ORDER BY ValidFrom;

    OPEN CityChangeList;
    FETCH NEXT FROM CityChangeList INTO @CityID, @ValidFrom;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        INSERT #CityChanges
            ([WWI City ID], City, [State Province], Country, Continent, [Sales Territory], Region, Subregion,
             [Location], [Latest Recorded Population], [Valid From], [Valid To])
        SELECT c.CityID, c.CityName, sp.StateProvinceName, co.CountryName, co.Continent, sp.SalesTerritory, co.
        Region, co.Subregion,
               c.[Location], COALESCE(c.LatestRecordedPopulation, 0), @ValidFrom, NULL
        FROM [Application].Cities FOR SYSTEM_TIME AS OF @ValidFrom AS c
        INNER JOIN [Application].StateProvinces FOR SYSTEM_TIME AS OF @ValidFrom AS sp
        ON c.StateProvinceID = sp.StateProvinceID
        INNER JOIN [Application].Countries FOR SYSTEM_TIME AS OF @ValidFrom AS co
        ON sp.CountryID = co.CountryID
        WHERE c.CityID = @CityID;

        FETCH NEXT FROM CityChangeList INTO @CityID, @ValidFrom;
    END;

    CLOSE CityChangeList;
    DEALLOCATE CityChangeList;

    -- add an index to make lookups faster

    CREATE INDEX IX_CityChanges ON #CityChanges ([WWI City ID], [Valid From]);

    -- work out the [Valid To] value by taking the [Valid From] of any row that's for the same city but later
    -- otherwise take the end of time

    UPDATE cc
    SET [Valid To] = COALESCE((SELECT MIN([Valid From]) FROM #CityChanges AS cc2
                                                       WHERE cc2.[WWI City ID] = cc.[WWI City ID]
                                                       AND cc2.[Valid From] > cc.[Valid From]), @EndOfTime)
    FROM #CityChanges AS cc;

    SELECT [WWI City ID], City, [State Province], Country, Continent, [Sales Territory],
           Region, Subregion, [Location] geography, [Latest Recorded Population], [Valid From],
           [Valid To]
    FROM #CityChanges
    ORDER BY [Valid From];

    DROP TABLE #CityChanges;

    RETURN 0;
END;
GO
```

Depends On 7

- Integration
- Application.Countries_Archive
- Application.Countries
- Application.Cities

- Application.StateProvinces
- Application.StateProvinces_Archive
- Application.Cities_Archive

Used By

No items found

Integration.GetCustomerUpdates

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@LastCutoff	datetime2	8	
@NewCutoff	datetime2	8	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Integration.GetCustomerUpdates
@LastCutoff datetime2(7),
@NewCutoff datetime2(7)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @EndOfTime datetime2(7) = '99991231 23:59:59.9999999';
    DECLARE @InitialLoadDate date = '20130101';

    CREATE TABLE #CustomerChanges
    (
        [WWI Customer ID] int,
        Customer nvarchar(100),
        [Bill To Customer] nvarchar(100),
        Category nvarchar(50),
        [Buying Group] nvarchar(50),
        [Primary Contact] nvarchar(50),
        [Postal Code] nvarchar(10),
        [Valid From] datetime2(7),
        [Valid To] datetime2(7) NULL
    );

    DECLARE @BuyingGroupID int;
    DECLARE @CustomerCategoryID int;
    DECLARE @CustomerID int;
    DECLARE @ValidFrom datetime2(7);

    -- first need to find any buying group changes that have occurred since initial load

    DECLARE BuyingGroupChangeList CURSOR FAST_FORWARD READ_ONLY
    FOR
    SELECT bg.BuyingGroupID,
           bg.ValidFrom
```

```
FROM Sales.BuyingGroups_Archive AS bg
WHERE bg.ValidFrom > @LastCutoff
AND bg.ValidFrom <= @NewCutoff
AND bg.ValidFrom <> @InitialLoadDate
UNION ALL
SELECT bg.BuyingGroupID,
       bg.ValidFrom
FROM Sales.BuyingGroups AS bg
WHERE bg.ValidFrom > @LastCutoff
AND bg.ValidFrom <= @NewCutoff
AND bg.ValidFrom <> @InitialLoadDate
ORDER BY ValidFrom;

OPEN BuyingGroupChangeList;
FETCH NEXT FROM BuyingGroupChangeList INTO @BuyingGroupID, @ValidFrom;

WHILE @@FETCH_STATUS = 0
BEGIN
    INSERT #CustomerChanges
        ([WWI Customer ID], Customer, [Bill To Customer], Category,
         [Buying Group], [Primary Contact], [Postal Code],
         [Valid From], [Valid To])
    SELECT c.CustomerID, c.CustomerName, bt.CustomerName, cc.CustomerCategoryName,
           bg.BuyingGroupName, p.FullName, c.DeliveryPostalCode,
           c.ValidFrom, c.ValidTo
    FROM Sales.Customers FOR SYSTEM_TIME AS OF @ValidFrom AS c
    INNER JOIN Sales.BuyingGroups FOR SYSTEM_TIME AS OF @ValidFrom AS bg
    ON c.BuyingGroupID = bg.BuyingGroupID
    INNER JOIN Sales.CustomerCategories FOR SYSTEM_TIME AS OF @ValidFrom AS cc
    ON c.CustomerCategoryID = cc.CustomerCategoryID
    INNER JOIN Sales.Customers FOR SYSTEM_TIME AS OF @ValidFrom AS bt
    ON c.BillToCustomerID = bt.CustomerID
    INNER JOIN [Application].People FOR SYSTEM_TIME AS OF @ValidFrom AS p
    ON c.PrimaryContactPersonID = p.PersonID
    WHERE c.BuyingGroupID = @BuyingGroupID;

    FETCH NEXT FROM BuyingGroupChangeList INTO @BuyingGroupID, @ValidFrom;
END;

CLOSE BuyingGroupChangeList;
DEALLOCATE BuyingGroupChangeList;

-- next need to find any customer category changes that have occurred since initial load

DECLARE CustomerCategoryChangeList CURSOR FAST_FORWARD READ_ONLY
FOR
SELECT cc.CustomerCategoryID,
       cc.ValidFrom
FROM Sales.CustomerCategories_Archive AS cc
WHERE cc.ValidFrom > @LastCutoff
AND cc.ValidFrom <= @NewCutoff
AND cc.ValidFrom <> @InitialLoadDate
UNION ALL
SELECT cc.CustomerCategoryID,
       cc.ValidFrom
FROM Sales.CustomerCategories AS cc
WHERE cc.ValidFrom > @LastCutoff
AND cc.ValidFrom <= @NewCutoff
AND cc.ValidFrom <> @InitialLoadDate
ORDER BY ValidFrom;

OPEN CustomerCategoryChangeList;
FETCH NEXT FROM CustomerCategoryChangeList INTO @CustomerCategoryID, @ValidFrom;

WHILE @@FETCH_STATUS = 0
BEGIN
    INSERT #CustomerChanges
        ([WWI Customer ID], Customer, [Bill To Customer], Category,
         [Buying Group], [Primary Contact], [Postal Code],
         [Valid From], [Valid To])
    SELECT c.CustomerID, c.CustomerName, bt.CustomerName, cc.CustomerCategoryName,
           bg.BuyingGroupName, p.FullName, c.DeliveryPostalCode,
           c.ValidFrom, c.ValidTo
    FROM Sales.Customers FOR SYSTEM_TIME AS OF @ValidFrom AS c
    INNER JOIN Sales.BuyingGroups FOR SYSTEM_TIME AS OF @ValidFrom AS bg
    ON c.BuyingGroupID = bg.BuyingGroupID
    INNER JOIN Sales.CustomerCategories FOR SYSTEM_TIME AS OF @ValidFrom AS cc
    ON c.CustomerCategoryID = cc.CustomerCategoryID
    INNER JOIN Sales.Customers FOR SYSTEM_TIME AS OF @ValidFrom AS bt
```

```
    ON c.BillToCustomerID = bt.CustomerID
    INNER JOIN [Application].People FOR SYSTEM_TIME AS OF @ValidFrom AS p
    ON c.PrimaryContactPersonID = p.PersonID
    WHERE cc.CustomerCategoryID = @CustomerCategoryID;

    FETCH NEXT FROM CustomerCategoryChangeList INTO @CustomerCategoryID, @ValidFrom;
END;

CLOSE CustomerCategoryChangeList;
DEALLOCATE CustomerCategoryChangeList;

-- finally need to find any customer changes that have occurred, including during the initial load

DECLARE CustomerChangeList CURSOR FAST_FORWARD READ_ONLY
FOR
SELECT c.CustomerID,
       c.ValidFrom
FROM Sales.Customers_Archive AS c
WHERE c.ValidFrom > @LastCutoff
AND c.ValidFrom <= @NewCutoff
UNION ALL
SELECT c.CustomerID,
       c.ValidFrom
FROM Sales.Customers AS c
WHERE c.ValidFrom > @LastCutoff
AND c.ValidFrom <= @NewCutoff
ORDER BY ValidFrom;

OPEN CustomerChangeList;
FETCH NEXT FROM CustomerChangeList INTO @CustomerID, @ValidFrom;

WHILE @@FETCH_STATUS = 0
BEGIN
    INSERT #CustomerChanges
        ([WWI Customer ID], Customer, [Bill To Customer], Category,
         [Buying Group], [Primary Contact], [Postal Code],
         [Valid From], [Valid To])
    SELECT c.CustomerID, c.CustomerName, bt.CustomerName, cc.CustomerCategoryName,
           bg.BuyingGroupName, p.FullName, c.DeliveryPostalCode,
           c.ValidFrom, c.ValidTo
    FROM Sales.Customers FOR SYSTEM_TIME AS OF @ValidFrom AS c
    INNER JOIN Sales.BuyingGroups FOR SYSTEM_TIME AS OF @ValidFrom AS bg
    ON c.BuyingGroupID = bg.BuyingGroupID
    INNER JOIN Sales.CustomerCategories FOR SYSTEM_TIME AS OF @ValidFrom AS cc
    ON c.CustomerCategoryID = cc.CustomerCategoryID
    INNER JOIN Sales.Customers FOR SYSTEM_TIME AS OF @ValidFrom AS bt
    ON c.BillToCustomerID = bt.CustomerID
    INNER JOIN [Application].People FOR SYSTEM_TIME AS OF @ValidFrom AS p
    ON c.PrimaryContactPersonID = p.PersonID
    WHERE c.CustomerID = @CustomerID;

    FETCH NEXT FROM CustomerChangeList INTO @CustomerID, @ValidFrom;
END;

CLOSE CustomerChangeList;
DEALLOCATE CustomerChangeList;

-- add an index to make lookups faster

CREATE INDEX IX_CustomerChanges ON #CustomerChanges ([WWI Customer ID], [Valid From]);

-- work out the [Valid To] value by taking the [Valid From] of any row that's for the same customer but later
-- otherwise take the end of time

UPDATE cc
SET [Valid To] = COALESCE((SELECT MIN([Valid From]) FROM #CustomerChanges AS cc2
                                                   WHERE cc2.[WWI Customer ID] = cc.[WWI Customer ID]
                                                   AND cc2.[Valid From] > cc.[Valid From]), @EndOfTime)
FROM #CustomerChanges AS cc;

SELECT [WWI Customer ID], Customer, [Bill To Customer], Category,
       [Buying Group], [Primary Contact], [Postal Code],
       [Valid From], [Valid To]
FROM #CustomerChanges
ORDER BY [Valid From];

DROP TABLE #CustomerChanges;
```

```
    RETURN 0;
END;
GO
```

Depends On 8

- Integration
- Sales.BuyingGroups_Archive
- Sales.BuyingGroups
- Sales.Customers
- Sales.CustomerCategories
- Application.People
- Sales.CustomerCategories_Archive
- Sales.Customers_Archive

Used By

No items found

Integration.GetEmployeeUpdates

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@LastCutoff	datetime2	8	
@NewCutoff	datetime2	8	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Integration.GetEmployeeUpdates
@LastCutoff datetime2(7),
@NewCutoff datetime2(7)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @EndOfTime datetime2(7) = '99991231 23:59:59.9999999';
    DECLARE @InitialLoadDate date = '20130101';

    CREATE TABLE #EmployeeChanges
    (
        [WWI Employee ID] int,
        Employee nvarchar(50),
        [Preferred Name] nvarchar(50),
        [Is Salesperson] bit,
        Photo varbinary(max),
        [Valid From] datetime2(7),
        [Valid To] datetime2(7)
    );

    DECLARE @PersonID int;
    DECLARE @ValidFrom datetime2(7);

    -- need to find any employee changes that have occurred, including during the initial load

    DECLARE EmployeeChangeList CURSOR FAST_FORWARD READ_ONLY
    FOR
    SELECT p.PersonID,
           p.ValidFrom
    FROM [Application].People_Archive AS p
    WHERE p.ValidFrom > @LastCutoff
    AND p.ValidFrom <= @NewCutoff
    AND p.IsEmployee <> 0
```

```
    UNION ALL
    SELECT p.PersonID,
           p.ValidFrom
    FROM [Application].People AS p
    WHERE p.ValidFrom > @LastCutoff
    AND p.ValidFrom <= @NewCutoff
    AND p.IsEmployee <> 0
    ORDER BY ValidFrom;

    OPEN EmployeeChangeList;
    FETCH NEXT FROM EmployeeChangeList INTO @PersonID, @ValidFrom;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        INSERT #EmployeeChanges
            ([WWI Employee ID], Employee, [Preferred Name], [Is Salesperson], Photo,
             [Valid From], [Valid To])
        SELECT p.PersonID, p.FullName, p.PreferredName, p.IsSalesperson, p.Photo,
               p.ValidFrom, p.ValidTo
        FROM [Application].People FOR SYSTEM_TIME AS OF @ValidFrom AS p
        WHERE p.PersonID = @PersonID;

        FETCH NEXT FROM EmployeeChangeList INTO @PersonID, @ValidFrom;
    END;

    CLOSE EmployeeChangeList;
    DEALLOCATE EmployeeChangeList;

    -- add an index to make lookups faster

    CREATE INDEX IX_EmployeeChanges ON #EmployeeChanges ([WWI Employee ID], [Valid From]);

    -- work out the [Valid To] value by taking the [Valid From] of any row that's for the same entry but later
    -- otherwise take the end of time

    UPDATE cc
    SET [Valid To] = COALESCE((SELECT MIN([Valid From]) FROM #EmployeeChanges AS cc2
                                                       WHERE cc2.[WWI Employee ID] = cc.[WWI Employee ID]
                                                       AND cc2.[Valid From] > cc.[Valid From]), @EndOfTime)
    FROM #EmployeeChanges AS cc;

    SELECT [WWI Employee ID], Employee, [Preferred Name], [Is Salesperson], Photo,
           [Valid From], [Valid To]
    FROM #EmployeeChanges
    ORDER BY [Valid From];

    DROP TABLE #EmployeeChanges;

    RETURN 0;
END;
GO
```

Depends On 3

- Integration
- Application.People_Archive
- Application.People

Used By

No items found

Integration.GetMovementUpdates

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@LastCutoff	datetime2	8	
@NewCutoff	datetime2	8	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Integration.GetMovementUpdates
@LastCutoff datetime2(7),
@NewCutoff datetime2(7)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @EndOfTime datetime2(7) = '99991231 23:59:59.9999999';
    DECLARE @InitialLoadDate date = '20130101';

    SELECT CAST(sit.TransactionOccurredWhen AS date) AS [Date Key],
           sit.StockItemTransactionID AS [WWI Stock Item Transaction ID],
           sit.InvoiceID AS [WWI Invoice ID],
           sit.PurchaseOrderID AS [WWI Purchase Order ID],
           CAST(sit.Quantity AS int) AS Quantity,
           sit.StockItemID AS [WWI Stock Item ID],
           sit.CustomerID AS [WWI Customer ID],
           sit.SupplierID AS [WWI Supplier ID],
           sit.TransactionTypeID AS [WWI Transaction Type ID],
           sit.TransactionOccurredWhen AS [Transaction Occurred When]
    FROM Warehouse.StockItemTransactions AS sit
    WHERE sit.LastEditedWhen > @LastCutoff
    AND sit.LastEditedWhen <= @NewCutoff
    ORDER BY sit.StockItemTransactionID;

    RETURN 0;
END;
GO
```

Depends On 2

- Integration
- Warehouse.StockItemTransactions

Used By

No items found

Integration.GetOrderUpdates

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@LastCutoff	datetime2	8	
@NewCutoff	datetime2	8	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Integration.GetOrderUpdates
@LastCutoff datetime2(7),
@NewCutoff datetime2(7)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @EndOfTime datetime2(7) = '99991231 23:59:59.9999999';
    DECLARE @InitialLoadDate date = '20130101';


    SELECT CAST(o.OrderDate AS date) AS [Order Date Key],
           CAST(ol.PickingCompletedWhen AS date) AS [Picked Date Key],
           o.OrderID AS [WWI Order ID],
           o.BackorderOrderID AS [WWI Backorder ID],
           ol.[Description],
           pt.PackageTypeName AS Package,
           ol.Quantity AS Quantity,
           ol.UnitPrice AS [Unit Price],
           ol.TaxRate AS [Tax Rate],
           ROUND(ol.Quantity * ol.UnitPrice, 2) AS [Total Excluding Tax],
           ROUND(ol.Quantity * ol.UnitPrice * ol.TaxRate / 100.0, 2) AS [Tax Amount],
           ROUND(ol.Quantity * ol.UnitPrice, 2) + ROUND(ol.Quantity * ol.UnitPrice * ol.TaxRate / 100.0, 2) AS
           [Total Including Tax],
           c.DeliveryCityID AS [WWI City ID],
           c.CustomerID AS [WWI Customer ID],
           ol.StockItemID AS [WWI Stock Item ID],
           o.SalespersonPersonID AS [WWI Salesperson ID],
           o.PickedByPersonID AS [WWI Picker ID],
           CASE WHEN ol.LastEditedWhen > o.LastEditedWhen THEN ol.LastEditedWhen ELSE o.LastEditedWhen END AS
           [Last Modified When]
    FROM Sales.Orders AS o
    INNER JOIN Sales.OrderLines AS ol
    ON o.OrderID = ol.OrderID
```

```
    INNER JOIN Warehouse.PackageTypes AS pt
    ON ol.PackageTypeID = pt.PackageTypeID
    INNER JOIN Sales.Customers AS c
    ON c.CustomerID = o.CustomerID
    WHERE CASE WHEN ol.LastEditedWhen > o.LastEditedWhen THEN ol.LastEditedWhen ELSE o.LastEditedWhen END >
    @LastCutoff
    AND CASE WHEN ol.LastEditedWhen > o.LastEditedWhen THEN ol.LastEditedWhen ELSE o.LastEditedWhen END <=
    @NewCutoff
    ORDER BY o.OrderID;

    RETURN 0;
END;
GO
```

Depends On 5

- Integration
- Sales.Orders
- Sales.OrderLines
- Warehouse.PackageTypes
- Sales.Customers

Used By

No items found

Integration.GetPaymentMethodUpdates

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@LastCutoff	datetime2	8	
@NewCutoff	datetime2	8	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Integration.GetPaymentMethodUpdates
@LastCutoff datetime2(7),
@NewCutoff datetime2(7)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @EndOfTime datetime2(7) = '99991231 23:59:59.9999999';
    DECLARE @InitialLoadDate date = '20130101';

    CREATE TABLE #PaymentMethodChanges
    (
        [WWI Payment Method ID] int,
        [Payment Method] nvarchar(50),
        [Valid From] datetime2(7),
        [Valid To] datetime2(7)
    );

    DECLARE @PaymentMethodID int;
    DECLARE @ValidFrom datetime2(7);

    -- need to find any payment method changes that have occurred, including during the initial load

    DECLARE ChangeList CURSOR FAST_FORWARD READ_ONLY
    FOR
    SELECT p.PaymentMethodID,
           p.ValidFrom
    FROM [Application].PaymentMethods_Archive AS p
    WHERE p.ValidFrom > @LastCutoff
    AND p.ValidFrom <= @NewCutoff
    UNION ALL
    SELECT p.PaymentMethodID,
           p.ValidFrom
    FROM [Application].PaymentMethods AS p
```

```
    WHERE p.ValidFrom > @LastCutoff
    AND p.ValidFrom <= @NewCutoff
    ORDER BY ValidFrom;

    OPEN ChangeList;
    FETCH NEXT FROM ChangeList INTO @PaymentMethodID, @ValidFrom;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        INSERT #PaymentMethodChanges
            ([WWI Payment Method ID], [Payment Method], [Valid From], [Valid To])
        SELECT p.PaymentMethodID, p.PaymentMethodName, p.ValidFrom, p.ValidTo
        FROM [Application].PaymentMethods FOR SYSTEM_TIME AS OF @ValidFrom AS p
        WHERE p.PaymentMethodID = @PaymentMethodID;

        FETCH NEXT FROM ChangeList INTO @PaymentMethodID, @ValidFrom;
    END;

    CLOSE ChangeList;
    DEALLOCATE ChangeList;

    -- add an index to make lookups faster

    CREATE INDEX IX_PaymentMethodChanges ON #PaymentMethodChanges ([WWI Payment Method ID], [Valid From]);

    -- work out the [Valid To] value by taking the [Valid From] of any row that's for the same entry but later
    -- otherwise take the end of time

    UPDATE cc
    SET [Valid To] = COALESCE((SELECT MIN([Valid From]) FROM #PaymentMethodChanges AS cc2
                                                        WHERE cc2.[WWI Payment Method ID] = cc.
                                                        [WWI Payment Method ID]
                                                        AND cc2.[Valid From] > cc.[Valid From]), @EndOfTime)
    FROM #PaymentMethodChanges AS cc;

    SELECT [WWI Payment Method ID], [Payment Method], [Valid From], [Valid To]
    FROM #PaymentMethodChanges
    ORDER BY [Valid From];

    DROP TABLE #PaymentMethodChanges;

    RETURN 0;
END;
GO
```

Depends On 3

- Integration
- Application.PaymentMethods_Archive
- Application.PaymentMethods

Used By

No items found

Integration.GetPurchaseUpdates

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@LastCutoff	datetime2	8	
@NewCutoff	datetime2	8	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Integration.GetPurchaseUpdates
@LastCutoff datetime2(7),
@NewCutoff datetime2(7)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @EndOfTime datetime2(7) = '99991231 23:59:59.9999999';
    DECLARE @InitialLoadDate date = '20130101';

    SELECT CAST(po.OrderDate AS date) AS [Date Key],
           po.PurchaseOrderID AS [WWI Purchase Order ID],
           pol.OrderedOuters AS [Ordered Outers],
           pol.OrderedOuters * si.QuantityPerOuter AS [Ordered Quantity],
           pol.ReceivedOuters AS [Received Outers],
           pt.PackageTypeName AS Package,
           pol.IsOrderLineFinalized AS [Is Order Finalized],
           po.SupplierID AS [WWI Supplier ID],
           pol.StockItemID AS [WWI Stock Item ID],
           CASE WHEN pol.LastEditedWhen > po.LastEditedWhen THEN pol.LastEditedWhen ELSE po.LastEditedWhen END AS
           [Last Modified When]
    FROM Purchasing.PurchaseOrders AS po
    INNER JOIN Purchasing.PurchaseOrderLines AS pol
    ON po.PurchaseOrderID = pol.PurchaseOrderID
    INNER JOIN Warehouse.StockItems AS si
    ON pol.StockItemID = si.StockItemID
    INNER JOIN Warehouse.PackageTypes AS pt
    ON pol.PackageTypeID = pt.PackageTypeID
    WHERE CASE WHEN pol.LastEditedWhen > po.LastEditedWhen THEN pol.LastEditedWhen ELSE po.LastEditedWhen END >
    @LastCutoff
    AND CASE WHEN pol.LastEditedWhen > po.LastEditedWhen THEN pol.LastEditedWhen ELSE po.LastEditedWhen END <=
    @NewCutoff
    ORDER BY po.PurchaseOrderID;
```

```
    RETURN 0;
END;
GO
```

Depends On 5

- Integration
- Purchasing.PurchaseOrders
- Purchasing.PurchaseOrderLines
- Warehouse.StockItems
- Warehouse.PackageTypes

Used By

No items found

Integration.GetSaleUpdates

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@LastCutoff	datetime2	8	
@NewCutoff	datetime2	8	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Integration.GetSaleUpdates
@LastCutoff datetime2(7),
@NewCutoff datetime2(7)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @EndOfTime datetime2(7) = '99991231 23:59:59.9999999';
    DECLARE @InitialLoadDate date = '20130101';

    SELECT CAST(i.InvoiceDate AS date) AS [Invoice Date Key],
           CAST(i.ConfirmedDeliveryTime AS date) AS [Delivery Date Key],
           i.InvoiceID AS [WWI Invoice ID],
           il.[Description],
           pt.PackageTypeName AS Package,
           il.Quantity,
           il.UnitPrice AS [Unit Price],
           il.TaxRate AS [Tax Rate],
           il.ExtendedPrice - il.TaxAmount AS [Total Excluding Tax],
           il.TaxAmount AS [Tax Amount],
           il.LineProfit AS Profit,
           il.ExtendedPrice AS [Total Including Tax],
           CASE WHEN si.IsChillerStock = 0 THEN il.Quantity ELSE 0 END AS [Total Dry Items],
           CASE WHEN si.IsChillerStock <> 0 THEN il.Quantity ELSE 0 END AS [Total Chiller Items],
           c.DeliveryCityID AS [WWI City ID],
           i.CustomerID AS [WWI Customer ID],
           i.BillToCustomerID AS [WWI Bill To Customer ID],
           il.StockItemID AS [WWI Stock Item ID],
           i.SalespersonPersonID AS [WWI Saleperson ID],
           CASE WHEN il.LastEditedWhen > i.LastEditedWhen THEN il.LastEditedWhen ELSE i.LastEditedWhen END AS
           [Last Modified When]
    FROM Sales.Invoices AS i
    INNER JOIN Sales.InvoiceLines AS il
    ON i.InvoiceID = il.InvoiceID
```

```
    INNER JOIN Warehouse.StockItems AS si
    ON il.StockItemID = si.StockItemID
    INNER JOIN Warehouse.PackageTypes AS pt
    ON il.PackageTypeID = pt.PackageTypeID
    INNER JOIN Sales.Customers AS c
    ON i.CustomerID = c.CustomerID
    INNER JOIN Sales.Customers AS bt
    ON i.BillToCustomerID = bt.CustomerID
    WHERE CASE WHEN il.LastEditedWhen > i.LastEditedWhen THEN il.LastEditedWhen ELSE i.LastEditedWhen END >
    @LastCutoff
    AND CASE WHEN il.LastEditedWhen > i.LastEditedWhen THEN il.LastEditedWhen ELSE i.LastEditedWhen END <=
    @NewCutoff
    ORDER BY i.InvoiceID, il.InvoiceLineID;

    RETURN 0;
END;
GO
```

Depends On 6

- Integration
- Sales.Invoices
- Sales.InvoiceLines
- Warehouse.StockItems
- Warehouse.PackageTypes
- Sales.Customers

Used By

No items found

Integration.GetStockHoldingUpdates

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Integration.GetStockHoldingUpdates
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    SELECT sih.QuantityOnHand AS [Quantity On Hand],
           sih.BinLocation AS [Bin Location],
           sih.LastStocktakeQuantity AS [Last Stocktake Quantity],
           sih.LastCostPrice AS [Last Cost Price],
           sih.ReorderLevel AS [Reorder Level],
           sih.TargetStockLevel AS [Target Stock Level],
           sih.StockItemID AS [WWI Stock Item ID]
    FROM Warehouse.StockItemHoldings AS sih
    ORDER BY sih.StockItemID;

    RETURN 0;
END;
GO
```

Depends On 2

- Integration
- Warehouse.StockItemHoldings

Used By

No items found

Integration.GetStockItemUpdates

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@LastCutoff	datetime2	8	
@NewCutoff	datetime2	8	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Integration.GetStockItemUpdates
@LastCutoff datetime2(7),
@NewCutoff datetime2(7)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @EndOfTime datetime2(7) = '99991231 23:59:59.9999999';
    DECLARE @InitialLoadDate date = '20130101';

    CREATE TABLE #StockItemChanges
    (
        [WWI Stock Item ID] int,
        [Stock Item] nvarchar(100),
        Color nvarchar(20),
        [Selling Package] nvarchar(50),
        [Buying Package] nvarchar(50),
        Brand nvarchar(50),
        Size nvarchar(20),
        [Lead Time Days] int,
        [Quantity Per Outer] int,
        [Is Chiller Stock] bit,
        Barcode nvarchar(50),
        [Tax Rate] decimal(18,3),
        [Unit Price] decimal(18,2),
        [Recommended Retail Price] decimal(18,2),
        [Typical Weight Per Unit] decimal(18,3),
        Photo varbinary(max),
        [Valid From] datetime2(7),
        [Valid To] datetime2(7)
    );

    DECLARE @StockItemID int;
    DECLARE @ValidFrom datetime2(7);
```

```
    -- need to find any StockItem changes that have occurred, including during the initial load

    DECLARE StockItemChangeList CURSOR FAST_FORWARD READ_ONLY
    FOR
    SELECT c.StockItemID,
           c.ValidFrom
    FROM Warehouse.StockItems_Archive AS c
    WHERE c.ValidFrom > @LastCutoff
    AND c.ValidFrom <= @NewCutoff
    UNION ALL
    SELECT c.StockItemID,
           c.ValidFrom
    FROM Warehouse.StockItems AS c
    WHERE c.ValidFrom > @LastCutoff
    AND c.ValidFrom <= @NewCutoff
    ORDER BY ValidFrom;

    OPEN StockItemChangeList;
    FETCH NEXT FROM StockItemChangeList INTO @StockItemID, @ValidFrom;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        INSERT #StockItemChanges
            ([WWI Stock Item ID], [Stock Item], Color, [Selling Package],
             [Buying Package], Brand, Size, [Lead Time Days], [Quantity Per Outer],
             [Is Chiller Stock], Barcode, [Tax Rate], [Unit Price], [Recommended Retail Price],
             [Typical Weight Per Unit], Photo, [Valid From], [Valid To])
        SELECT si.StockItemID, si.StockItemName, c.ColorName, spt.PackageTypeName,
               bpt.PackageTypeName, si.Brand, si.Size, si.LeadTimeDays, si.QuantityPerOuter,
               si.IsChillerStock, si.Barcode, si.LeadTimeDays, si.UnitPrice, si.RecommendedRetailPrice,
               si.TypicalWeightPerUnit, si.Photo, si.ValidFrom, si.ValidTo
        FROM Warehouse.StockItems FOR SYSTEM_TIME AS OF @ValidFrom AS si
        INNER JOIN Warehouse.PackageTypes FOR SYSTEM_TIME AS OF @ValidFrom AS spt
        ON si.UnitPackageID = spt.PackageTypeID
        INNER JOIN Warehouse.PackageTypes FOR SYSTEM_TIME AS OF @ValidFrom AS bpt
        ON si.OuterPackageID = bpt.PackageTypeID
        LEFT OUTER JOIN Warehouse.Colors FOR SYSTEM_TIME AS OF @ValidFrom AS c
        ON si.ColorID = c.ColorID
        WHERE si.StockItemID = @StockItemID;

        FETCH NEXT FROM StockItemChangeList INTO @StockItemID, @ValidFrom;
    END;

    CLOSE StockItemChangeList;
    DEALLOCATE StockItemChangeList;

    -- add an index to make lookups faster

    CREATE INDEX IX_StockItemChanges ON #StockItemChanges ([WWI Stock Item ID], [Valid From]);

    -- work out the [Valid To] value by taking the [Valid From] of any row that's for the same StockItem but later
    -- otherwise take the end of time

    UPDATE cc
    SET [Valid To] = COALESCE((SELECT MIN([Valid From]) FROM #StockItemChanges AS cc2
                                                       WHERE cc2.[WWI Stock Item ID] = cc.[WWI Stock Item ID]
                                                       AND cc2.[Valid From] > cc.[Valid From]), @EndOfTime)
    FROM #StockItemChanges AS cc;

    SELECT [WWI Stock Item ID], [Stock Item],
           ISNULL(Color, N'N/A') AS Color,
           [Selling Package], [Buying Package],
           ISNULL(Brand, N'N/A') AS Brand,
           ISNULL(Size, N'N/A') AS Size,
           [Lead Time Days], [Quantity Per Outer], [Is Chiller Stock],
           ISNULL(Barcode, N'N/A') AS Barcode,
           [Tax Rate], [Unit Price], [Recommended Retail Price], [Typical Weight Per Unit],
           Photo, [Valid From], [Valid To]
    FROM #StockItemChanges
    ORDER BY [Valid From];

    DROP TABLE #StockItemChanges;

    RETURN 0;
END;
GO
```

Depends On 5

- Integration
- Warehouse.StockItems_Archive
- Warehouse.StockItems
- Warehouse.PackageTypes
- Warehouse.Colors

Used By

No items found

Integration.GetSupplierUpdates

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@LastCutoff	datetime2	8	
@NewCutoff	datetime2	8	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Integration.GetSupplierUpdates
@LastCutoff datetime2(7),
@NewCutoff datetime2(7)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @EndOfTime datetime2(7) = '99991231 23:59:59.9999999';
    DECLARE @InitialLoadDate date = '20130101';

    CREATE TABLE #SupplierChanges
    (
        [WWI Supplier ID] int,
        Supplier nvarchar(100),
        Category nvarchar(50),
        [Primary Contact] nvarchar(50),
        [Supplier Reference] nvarchar(20),
        [Payment Days] int,
        [Postal Code] nvarchar(10),
        [Valid From] datetime2(7),
        [Valid To] datetime2(7)
    );

    DECLARE @SupplierCategoryID int;
    DECLARE @SupplierID int;
    DECLARE @ValidFrom datetime2(7);

    -- need to find any Supplier category changes that have occurred since initial load

    DECLARE SupplierCategoryChangeList CURSOR FAST_FORWARD READ_ONLY
    FOR
    SELECT cc.SupplierCategoryID,
           cc.ValidFrom
    FROM Purchasing.SupplierCategories_Archive AS cc
```

```
    WHERE cc.ValidFrom > @LastCutoff
    AND cc.ValidFrom <= @NewCutoff
    AND cc.ValidFrom <> @InitialLoadDate
    UNION ALL
    SELECT cc.SupplierCategoryID,
           cc.ValidFrom
    FROM Purchasing.SupplierCategories AS cc
    WHERE cc.ValidFrom > @LastCutoff
    AND cc.ValidFrom <= @NewCutoff
    AND cc.ValidFrom <> @InitialLoadDate
    ORDER BY ValidFrom;

    OPEN SupplierCategoryChangeList;
    FETCH NEXT FROM SupplierCategoryChangeList INTO @SupplierCategoryID, @ValidFrom;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        INSERT #SupplierChanges
            ([WWI Supplier ID], Supplier, Category, [Primary Contact], [Supplier Reference],
             [Payment Days], [Postal Code], [Valid From], [Valid To])
        SELECT s.SupplierID, s.SupplierName, sc.SupplierCategoryName, p.FullName, s.SupplierReference,
               s.PaymentDays, s.DeliveryPostalCode, s.ValidFrom, s.ValidTo
        FROM Purchasing.Suppliers FOR SYSTEM_TIME AS OF @ValidFrom AS s
        INNER JOIN Purchasing.SupplierCategories FOR SYSTEM_TIME AS OF @ValidFrom AS sc
        ON s.SupplierCategoryID = sc.SupplierCategoryID
        INNER JOIN [Application].People FOR SYSTEM_TIME AS OF @ValidFrom AS p
        ON s.PrimaryContactPersonID = p.PersonID
        WHERE sc.SupplierCategoryID = @SupplierCategoryID;

        FETCH NEXT FROM SupplierCategoryChangeList INTO @SupplierCategoryID, @ValidFrom;
    END;

    CLOSE SupplierCategoryChangeList;
    DEALLOCATE SupplierCategoryChangeList;

    -- finally need to find any Supplier changes that have occurred, including during the initial load

    DECLARE SupplierChangeList CURSOR FAST_FORWARD READ_ONLY
    FOR
    SELECT c.SupplierID,
           c.ValidFrom
    FROM Purchasing.Suppliers_Archive AS c
    WHERE c.ValidFrom > @LastCutoff
    AND c.ValidFrom <= @NewCutoff
    UNION ALL
    SELECT c.SupplierID,
           c.ValidFrom
    FROM Purchasing.Suppliers AS c
    WHERE c.ValidFrom > @LastCutoff
    AND c.ValidFrom <= @NewCutoff
    ORDER BY ValidFrom;

    OPEN SupplierChangeList;
    FETCH NEXT FROM SupplierChangeList INTO @SupplierID, @ValidFrom;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        INSERT #SupplierChanges
            ([WWI Supplier ID], Supplier, Category, [Primary Contact], [Supplier Reference],
             [Payment Days], [Postal Code], [Valid From], [Valid To])
        SELECT s.SupplierID, s.SupplierName, sc.SupplierCategoryName, p.FullName, s.SupplierReference,
               s.PaymentDays, s.DeliveryPostalCode, s.ValidFrom, s.ValidTo
        FROM Purchasing.Suppliers FOR SYSTEM_TIME AS OF @ValidFrom AS s
        INNER JOIN Purchasing.SupplierCategories FOR SYSTEM_TIME AS OF @ValidFrom AS sc
        ON s.SupplierCategoryID = sc.SupplierCategoryID
        INNER JOIN [Application].People FOR SYSTEM_TIME AS OF @ValidFrom AS p
        ON s.PrimaryContactPersonID = p.PersonID
        WHERE s.SupplierID = @SupplierID;

        FETCH NEXT FROM SupplierChangeList INTO @SupplierID, @ValidFrom;
    END;

    CLOSE SupplierChangeList;
    DEALLOCATE SupplierChangeList;

    -- add an index to make lookups faster

    CREATE INDEX IX_SupplierChanges ON #SupplierChanges ([WWI Supplier ID], [Valid From]);
```

```
    -- work out the [Valid To] value by taking the [Valid From] of any row that's for the same Supplier but later
    -- otherwise take the end of time

    UPDATE cc
    SET [Valid To] = COALESCE((SELECT MIN([Valid From]) FROM #SupplierChanges AS cc2
                                                        WHERE cc2.[WWI Supplier ID] = cc.[WWI Supplier ID]
                                                        AND cc2.[Valid From] > cc.[Valid From]), @EndOfTime)
    FROM #SupplierChanges AS cc;

    SELECT [WWI Supplier ID], Supplier, Category, [Primary Contact],
           [Supplier Reference], [Payment Days], [Postal Code],
           [Valid From], [Valid To]
    FROM #SupplierChanges
    ORDER BY [Valid From];

    DROP TABLE #SupplierChanges;

    RETURN 0;
END;
GO
```

Depends On 6

- Integration
- Purchasing.SupplierCategories_Archive
- Purchasing.SupplierCategories
- Purchasing.Suppliers
- Application.People
- Purchasing.Suppliers_Archive

Used By

No items found

Integration.GetTransactionTypeUpdates

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@LastCutoff	datetime2	8	
@NewCutoff	datetime2	8	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Integration.GetTransactionTypeUpdates
@LastCutoff datetime2(7),
@NewCutoff datetime2(7)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @EndOfTime datetime2(7) = '99991231 23:59:59.9999999';
    DECLARE @InitialLoadDate date = '20130101';

    CREATE TABLE #TransactionTypeChanges
    (
        [WWI Transaction Type ID] int,
        [Transaction Type] nvarchar(50),
        [Valid From] datetime2(7),
        [Valid To] datetime2(7)
    );

    DECLARE @TransactionTypeID int;
    DECLARE @ValidFrom datetime2(7);

    -- need to find any Transaction Type changes that have occurred, including during the initial load

    DECLARE ChangeList CURSOR FAST_FORWARD READ_ONLY
    FOR
    SELECT tt.TransactionTypeID,
           tt.ValidFrom
    FROM [Application].TransactionTypes_Archive AS tt
    WHERE tt.ValidFrom > @LastCutoff
    AND tt.ValidFrom <= @NewCutoff
    UNION ALL
    SELECT tt.TransactionTypeID,
           tt.ValidFrom
    FROM [Application].TransactionTypes AS tt
```

```
    WHERE tt.ValidFrom > @LastCutoff
    AND tt.ValidFrom <= @NewCutoff
    ORDER BY ValidFrom;

    OPEN ChangeList;
    FETCH NEXT FROM ChangeList INTO @TransactionTypeID, @ValidFrom;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        INSERT #TransactionTypeChanges
            ([WWI Transaction Type ID], [Transaction Type], [Valid From], [Valid To])
        SELECT p.TransactionTypeID, p.TransactionTypeName, p.ValidFrom, p.ValidTo
        FROM [Application].TransactionTypes FOR SYSTEM_TIME AS OF @ValidFrom AS p
        WHERE p.TransactionTypeID = @TransactionTypeID;

        FETCH NEXT FROM ChangeList INTO @TransactionTypeID, @ValidFrom;
    END;

    CLOSE ChangeList;
    DEALLOCATE ChangeList;

    -- add an index to make lookups faster

    CREATE INDEX IX_TransactionTypeChanges ON #TransactionTypeChanges ([WWI Transaction Type ID], [Valid From]);

    -- work out the [Valid To] value by taking the [Valid From] of any row that's for the same entry but later
    -- otherwise take the end of time

    UPDATE cc
    SET [Valid To] = COALESCE((SELECT MIN([Valid From]) FROM #TransactionTypeChanges AS cc2
                                                        WHERE cc2.[WWI Transaction Type ID] = cc.
                                                        [WWI Transaction Type ID]
                                                        AND cc2.[Valid From] > cc.[Valid From]), @EndOfTime)
    FROM #TransactionTypeChanges AS cc;

    SELECT [WWI Transaction Type ID], [Transaction Type], [Valid From], [Valid To]
    FROM #TransactionTypeChanges
    ORDER BY [Valid From];

    DROP TABLE #TransactionTypeChanges;

    RETURN 0;
END;
GO
```

Depends On 3

- Integration
- Application.TransactionTypes_Archive
- Application.TransactionTypes

Used By

No items found

Integration.GetTransactionUpdates

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@LastCutoff	datetime2	8	
@NewCutoff	datetime2	8	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Integration.GetTransactionUpdates
@LastCutoff datetime2(7),
@NewCutoff datetime2(7)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @EndOfTime datetime2(7) = '99991231 23:59:59.9999999';
    DECLARE @InitialLoadDate date = '20130101';

    SELECT CAST(ct.TransactionDate AS date) AS [Date Key],
           ct.CustomerTransactionID AS [WWI Customer Transaction ID],
           CAST(NULL AS int) AS [WWI Supplier Transaction ID],
           ct.InvoiceID AS [WWI Invoice ID],
           CAST(NULL AS int) AS [WWI Purchase Order ID],
           CAST(NULL AS nvarchar(20)) AS [Supplier Invoice Number],
           ct.AmountExcludingTax AS [Total Excluding Tax],
           ct.TaxAmount AS [Tax Amount],
           ct.TransactionAmount AS [Total Including Tax],
           ct.OutstandingBalance AS [Outstanding Balance],
           ct.IsFinalized AS [Is Finalized],
           COALESCE(i.CustomerID, ct.CustomerID) AS [WWI Customer ID],
           ct.CustomerID AS [WWI Bill To Customer ID],
           CAST(NULL AS int) AS [WWI Supplier ID],
           ct.TransactionTypeID AS [WWI Transaction Type ID],
           ct.PaymentMethodID AS [WWI Payment Method ID],
           ct.LastEditedWhen AS [Last Modified When]
    FROM Sales.CustomerTransactions AS ct
    LEFT OUTER JOIN Sales.Invoices AS i
    ON ct.InvoiceID = i.InvoiceID
    WHERE ct.LastEditedWhen > @LastCutoff
    AND ct.LastEditedWhen <= @NewCutoff

    UNION ALL
```

```
    SELECT CAST(st.TransactionDate AS date) AS [Date Key],
           CAST(NULL AS int) AS [WWI Customer Transaction ID],
           st.SupplierTransactionID AS [WWI Supplier Transaction ID],
           CAST(NULL AS int) AS [WWI Invoice ID],
           st.PurchaseOrderID AS [WWI Purchase Order ID],
           st.SupplierInvoiceNumber AS [Supplier Invoice Number],
           st.AmountExcludingTax AS [Total Excluding Tax],
           st.TaxAmount AS [Tax Amount],
           st.TransactionAmount AS [Total Including Tax],
           st.OutstandingBalance AS [Outstanding Balance],
           st.IsFinalized AS [Is Finalized],
           CAST(NULL AS int) AS [WWI Customer ID],
           CAST(NULL AS int) AS [WWI Bill To Customer ID],
           st.SupplierID AS [WWI Supplier ID],
           st.TransactionTypeID AS [WWI Transaction Type ID],
           st.PaymentMethodID AS [WWI Payment Method ID],
           st.LastEditedWhen AS [Last Modified When]
    FROM Purchasing.SupplierTransactions AS st
    WHERE st.LastEditedWhen > @LastCutoff
    AND st.LastEditedWhen <= @NewCutoff;

    RETURN 0;
END;
GO
```

Depends On 4

- Integration
- Sales.CustomerTransactions
- Sales.Invoices
- Purchasing.SupplierTransactions

Used By

No items found

Sequences.ReseedAllSequences

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	
Assembly	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Sequences.ReseedAllSequences
AS BEGIN
    -- Ensures that the next sequence values are above the maximum value of the related table columns
    SET NOCOUNT ON;

    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'BuyingGroupID', @SchemaName = 'Sales', @TableName =
'BuyingGroups', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'CityID', @SchemaName = 'Application', @TableName = '
Cities', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'ColorID', @SchemaName = 'Warehouse', @TableName = 'C
olors', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'CountryID', @SchemaName = 'Application', @TableName =
 'Countries', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'CustomerCategoryID', @SchemaName = 'Sales', @TableNa
me = '
    CustomerCategories', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'CustomerID', @SchemaName = 'Sales', @TableName = 'Cu
stomers', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'DeliveryMethodID', @SchemaName = 'Application', @Tab
leName = '
    DeliveryMethods', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'InvoiceID', @SchemaName = 'Sales', @TableName = 'Inv
oices', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'InvoiceLineID', @SchemaName = 'Sales', @TableName =
'InvoiceLines', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'OrderID', @SchemaName = 'Sales', @TableName = 'Order
s', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'OrderLineID', @SchemaName = 'Sales', @TableName = 'O
rderLines', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'PackageTypeID', @SchemaName = 'Warehouse', @TableName
 = 'PackageTypes', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'PaymentMethodID', @SchemaName = 'Application', @Tabl
eName = '
    PaymentMethods', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'PersonID', @SchemaName = 'Application', @TableName =
'People', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'PurchaseOrderID', @SchemaName = 'Purchasing', @Table
Name = '
    PurchaseOrders', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'PurchaseOrderLineID', @SchemaName = 'Purchasing', @T
ableName = '
    PurchaseOrderLines', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'SpecialDealID', @SchemaName = 'Sales', @TableName =
'SpecialDeals', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'StateProvinceID', @SchemaName = 'Application', @Tabl
eName = '
```

```
    StateProvinces', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'StockGroupID', @SchemaName = 'Warehouse', @TableName
= 'StockGroups', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'StockItemID', @SchemaName = 'Warehouse', @TableName =
 'StockItems', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'StockItemStockGroupID', @SchemaName = 'Warehouse', @
TableName = '
    StockItemStockGroups', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'SupplierCategoryID', @SchemaName = 'Purchasing', @Ta
bleName = '
    SupplierCategories', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'SupplierID', @SchemaName = 'Purchasing', @TableName =
 'Suppliers', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'SystemParameterID', @SchemaName = 'Application', @Ta
bleName = '
    SystemParameters', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'TransactionID', @SchemaName = 'Purchasing', @TableNa
me = '
    SupplierTransactions', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'TransactionID', @SchemaName = 'Sales', @TableName =
'
    CustomerTransactions', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'TransactionID', @SchemaName = 'Warehouse', @TableName
 = '
    StockItemTransactions', @ColumnName = ;
    EXEC Sequences.ReseedSequenceBeyondTableValues @SequenceName = 'TransactionTypeID', @SchemaName = 'Application', @Ta
bleName = '
    TransactionTypes', @ColumnName = ;
END;
GO
```

Depends On 2

- Sequences
- Sequences.ReseedSequenceBeyondTableValues

Used By

No items found

Sequences.ReseedSequenceBeyondTableValues

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	
Assembly	

Parameters

Name	Data Type	Length	Description
@SequenceName	sysname	256	
@SchemaName	sysname	256	
@TableName	sysname	256	
@ColumnName	sysname	256	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Sequences.ReseedSequenceBeyondTableValues
@SequenceName sysname,
@SchemaName sysname,
@TableName sysname,
@ColumnName sysname
AS BEGIN
    -- Ensures that the next sequence value is above the maximum value of the supplied table column
    SET NOCOUNT ON;

    DECLARE @SQL nvarchar(max);
    DECLARE @CurrentTableMaximumValue bigint;
    DECLARE @NewSequenceValue bigint;
    DECLARE @CurrentSequenceMaximumValue bigint
        = (SELECT CAST(current_value AS bigint) FROM sys.sequences
                                                WHERE name = @SequenceName
                                                AND SCHEMA_NAME(schema_id) = );
    CREATE TABLE #CurrentValue
    (
        CurrentValue bigint
    )

    SET @SQL = N'INSERT #CurrentValue (CurrentValue) SELECT COALESCE(MAX(' + QUOTENAME(@ColumnName) + N'), 0) FROM
' + QUOTENAME(@SchemaName) + N'.' + QUOTENAME(@TableName) + N';';
    EXECUTE (@SQL);
    SET @CurrentTableMaximumValue = (SELECT CurrentValue FROM #CurrentValue);
    DROP TABLE #CurrentValue;

    IF @CurrentTableMaximumValue >= @CurrentSequenceMaximumValue
    BEGIN
        SET @NewSequenceValue = @CurrentTableMaximumValue + 1;
        SET @SQL = N'ALTER SEQUENCE Sequences.' + QUOTENAME(@SequenceName) + N' RESTART WITH
' + CAST(@NewSequenceValue AS nvarchar(20)) + N';';
```

```
        EXECUTE (@SQL);
    END;
END;
GO
```

Depends On 1

- Sequences

Used By 1

- Sequences.ReseedAllSequences

Website.ActivateWebsiteLogon

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@PersonID	int	4	
@LogonName	nvarchar	50	
@InitialPassword	nvarchar	40	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Website.ActivateWebsiteLogon
@PersonID int,
@LogonName nvarchar(50),
@InitialPassword nvarchar(40)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    UPDATE [Application].People
    SET IsPermittedToLogon = 1,
        LogonName = @LogonName,
        HashedPassword = HASHBYTES(N'SHA2_256', @InitialPassword + FullName),
        UserPreferences = (SELECT UserPreferences FROM [Application].People WHERE PersonID = 1)
        -- Person 1 has User Preferences template
    WHERE PersonID = @PersonID
    AND PersonID <> 1
    AND IsPermittedToLogon = 0;

    IF @@ROWCOUNT = 0
    BEGIN
        PRINT N'The PersonID must be valid, must not be person 1, and must not already be enabled';
        THROW 51000, N'Invalid PersonID', 1;
        RETURN -1;
    END;
END;
GO
```

Depends On 2

- Website
- Application.People

Used By

No items found

Website.ChangePassword

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@PersonID	int	4	
@OldPassword	nvarchar	40	
@NewPassword	nvarchar	40	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Website.ChangePassword
@PersonID int,
@OldPassword nvarchar(40),
@NewPassword nvarchar(40)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    UPDATE [Application].People
    SET IsPermittedToLogon = 1,
        HashedPassword = HASHBYTES(N'SHA2_256', @NewPassword + FullName)
    WHERE PersonID = @PersonID
    AND PersonID <> 1
    AND HashedPassword = HASHBYTES(N'SHA2_256', @OldPassword + FullName);

    IF @@ROWCOUNT = 0
    BEGIN
        PRINT N'The PersonID must be valid, and the old password must be valid.';
        PRINT N'If the user has also changed name, please contact the IT staff to assist.';
        THROW 51000, N'Invalid Password Change', 1;
        RETURN -1;
    END;
END;
GO
```

Depends On 2

- Website
- Application.People

Used By

No items found

Website.InsertCustomerOrders

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@Orders	OrderList	max	
@OrderLines	OrderLineList	max	
@OrdersCreatedByPersonID	int	4	
@SalespersonPersonID	int	4	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Website.InsertCustomerOrders
@Orders Website.OrderList READONLY,
@OrderLines Website.OrderLineList READONLY,
@OrdersCreatedByPersonID int,
@SalespersonPersonID int
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @OrdersToGenerate AS TABLE
    (
        OrderReference int PRIMARY KEY,   -- reference from the application
        OrderID int
    );

    -- allocate the new order numbers

    INSERT @OrdersToGenerate (OrderReference, OrderID)
    SELECT OrderReference, NEXT VALUE FOR Sequences.OrderID
    FROM @Orders;

    BEGIN TRY

        BEGIN TRAN;

        INSERT Sales.Orders
            (OrderID, CustomerID, SalespersonPersonID, PickedByPersonID, ContactPersonID, BackorderOrderID,
            OrderDate,
             ExpectedDeliveryDate, CustomerPurchaseOrderNumber, IsUndersupplyBackordered, Comments,
             DeliveryInstructions, InternalComments,
```

```
            PickingCompletedWhen, LastEditedBy, LastEditedWhen)
        SELECT otg.OrderID, o.CustomerID, @SalespersonPersonID, NULL, o.ContactPersonID, NULL, SYSDATETIME(),
               o.ExpectedDeliveryDate, o.CustomerPurchaseOrderNumber, o.IsUndersupplyBackordered, o.Comments, o.
               DeliveryInstructions, NULL,
               NULL, @OrdersCreatedByPersonID, SYSDATETIME()
        FROM @OrdersToGenerate AS otg
        INNER JOIN @Orders AS o
        ON otg.OrderReference = o.OrderReference;

        INSERT Sales.OrderLines
            (OrderID, StockItemID, [Description], PackageTypeID, Quantity, UnitPrice,
             TaxRate, PickedQuantity, PickingCompletedWhen, LastEditedBy, LastEditedWhen)
        SELECT otg.OrderID, ol.StockItemID, ol.[Description], si.UnitPackageID, ol.Quantity,
               Website.CalculateCustomerPrice(o.CustomerID, ol.StockItemID, SYSDATETIME()),
               si.TaxRate, 0, NULL, @OrdersCreatedByPersonID, SYSDATETIME()
        FROM @OrdersToGenerate AS otg
        INNER JOIN @OrderLines AS ol
        ON otg.OrderReference = ol.OrderReference
  INNER JOIN @Orders AS o
  ON ol.OrderReference = o.OrderReference
        INNER JOIN Warehouse.StockItems AS si
        ON ol.StockItemID = si.StockItemID;

        COMMIT;

    END TRY
    BEGIN CATCH
        IF XACT_STATE() <> 0 ROLLBACK;
        PRINT N'Unable to create the customer orders.';
        THROW;
        RETURN -1;
    END CATCH;

    RETURN 0;
END;
GO
```

Depends On 8

- Website
- Website.OrderList
- Website.OrderLineList
- Sequences.OrderID
- Sales.Orders
- Sales.OrderLines
- Website.CalculateCustomerPrice
- Warehouse.StockItems

Used By

No items found

Website.InvoiceCustomerOrders

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@OrdersToInvoice	OrderIDList	max	
@PackedByPersonID	int	4	
@InvoicedByPersonID	int	4	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Website.InvoiceCustomerOrders
@OrdersToInvoice Website.OrderIDList READONLY,
@PackedByPersonID int,
@InvoicedByPersonID int
WITH EXECUTE AS OWNER
AS
BEGIN
    SET NOCOUNT ON;
    SET XACT_ABORT ON;

    DECLARE @InvoicesToGenerate TABLE
    (
        OrderID int PRIMARY KEY,
        InvoiceID int NOT NULL,
        TotalDryItems int NOT NULL,
        TotalChillerItems int NOT NULL
    );

    BEGIN TRY;


        -- Check that all orders exist, have been fully picked, and not already invoiced. Also allocate new invoice
numbers.
        INSERT @InvoicesToGenerate (OrderID, InvoiceID, TotalDryItems, TotalChillerItems)
        SELECT oti.OrderID,
               NEXT VALUE FOR Sequences.InvoiceID,
               COALESCE((SELECT SUM(CASE WHEN si.IsChillerStock <> 0 THEN 0 ELSE 1 END)
                         FROM Sales.OrderLines AS ol
                         INNER JOIN Warehouse.StockItems AS si
                         ON ol.StockItemID = si.StockItemID
                         WHERE ol.OrderID = oti.OrderID), 0),
               COALESCE((SELECT SUM(CASE WHEN si.IsChillerStock <> 0 THEN 1 ELSE 0 END)
                         FROM Sales.OrderLines AS ol
                         INNER JOIN Warehouse.StockItems AS si
```

```
                ON ol.StockItemID = si.StockItemID
                WHERE ol.OrderID = oti.OrderID), 0)
FROM @OrdersToInvoice AS oti
INNER JOIN Sales.Orders AS o
ON oti.OrderID = o.OrderID
WHERE NOT EXISTS (SELECT 1 FROM Sales.Invoices AS i
                          WHERE i.OrderID = oti.OrderID)
AND o.PickingCompletedWhen IS NOT NULL;

IF EXISTS (SELECT 1 FROM @OrdersToInvoice AS oti WHERE NOT EXISTS (SELECT 1 FROM @InvoicesToGenerate AS itg
WHERE itg.OrderID = oti.OrderID))
BEGIN
    PRINT N'At least one order ID either does not exist, is not picked, or is already invoiced';
    THROW 51000, N'At least one orderID either does not exist, is not picked, or is already
    invoiced', 1;
END;

BEGIN TRAN;

INSERT Sales.Invoices
    (InvoiceID, CustomerID, BillToCustomerID, OrderID, DeliveryMethodID, ContactPersonID, AccountsPersonID,
     SalespersonPersonID, PackedByPersonID, InvoiceDate, CustomerPurchaseOrderNumber,
     IsCreditNote, CreditNoteReason, Comments, DeliveryInstructions, InternalComments,
     TotalDryItems, TotalChillerItems,  DeliveryRun, RunPosition,
     ReturnedDeliveryData,
     LastEditedBy, LastEditedWhen)
SELECT itg.InvoiceID, c.CustomerID, c.BillToCustomerID, itg.OrderID, c.DeliveryMethodID, o.ContactPersonID,
btc.PrimaryContactPersonID,
       o.SalespersonPersonID, @PackedByPersonID, SYSDATETIME(), o.CustomerPurchaseOrderNumber,
       0, NULL, NULL, c.DeliveryAddressLine1 + N', ' + c.DeliveryAddressLine2, NULL,
       itg.TotalDryItems, itg.TotalChillerItems, c.DeliveryRun, c.RunPosition,
       JSON_MODIFY(N'{"Events": []}', N'append $.Events',
           JSON_MODIFY(JSON_MODIFY(JSON_MODIFY(N'{ }', N'$.Event', N'Ready for collection'),
           N'$.EventTime', CONVERT(nvarchar(20), SYSDATETIME(), 126)),
           N'$.ConNote', N'EAN-125-' + CAST(itg.InvoiceID + 1050 AS nvarchar(20)))),
       @InvoicedByPersonID, SYSDATETIME()
FROM @InvoicesToGenerate AS itg
INNER JOIN Sales.Orders AS o
ON itg.OrderID = o.OrderID
INNER JOIN Sales.Customers AS c
ON o.CustomerID = c.CustomerID
INNER JOIN Sales.Customers AS btc
ON btc.CustomerID = c.BillToCustomerID;

INSERT Sales.InvoiceLines
    (InvoiceID, StockItemID, [Description], PackageTypeID,
     Quantity, UnitPrice, TaxRate, TaxAmount, LineProfit, ExtendedPrice,
     LastEditedBy, LastEditedWhen)
SELECT itg.InvoiceID, ol.StockItemID, ol.[Description], ol.PackageTypeID,
       ol.PickedQuantity, ol.UnitPrice, ol.TaxRate,
       ROUND(ol.PickedQuantity * ol.UnitPrice * ol.TaxRate / 100.0, 2),
       ROUND(ol.PickedQuantity * (ol.UnitPrice - sih.LastCostPrice), 2),
       ROUND(ol.PickedQuantity * ol.UnitPrice, 2)
         + ROUND(ol.PickedQuantity * ol.UnitPrice * ol.TaxRate / 100.0, 2),
       @InvoicedByPersonID, SYSDATETIME()
FROM @InvoicesToGenerate AS itg
INNER JOIN Sales.OrderLines AS ol
ON itg.OrderID = ol.OrderID
INNER JOIN Warehouse.StockItems AS si
ON ol.StockItemID = si.StockItemID
INNER JOIN Warehouse.StockItemHoldings AS sih
ON si.StockItemID = sih.StockItemID
ORDER BY ol.OrderID, ol.OrderLineID;

INSERT Warehouse.StockItemTransactions
    (StockItemID, TransactionTypeID, CustomerID, InvoiceID, SupplierID, PurchaseOrderID,
     TransactionOccurredWhen, Quantity, LastEditedBy, LastEditedWhen)
SELECT il.StockItemID, (SELECT TransactionTypeID FROM [Application].TransactionTypes WHERE
TransactionTypeName = N'Stock Issue'),
       i.CustomerID, i.InvoiceID, NULL, NULL,
       SYSDATETIME(), 0 - il.Quantity, @InvoicedByPersonID, SYSDATETIME()
FROM @InvoicesToGenerate AS itg
INNER JOIN Sales.InvoiceLines AS il
ON itg.InvoiceID = il.InvoiceID
INNER JOIN Sales.Invoices AS i
ON il.InvoiceID = i.InvoiceID
ORDER BY il.InvoiceID, il.InvoiceLineID;

WITH StockItemTotals
```

```
        AS
        (
            SELECT il.StockItemID, SUM(il.Quantity) AS TotalQuantity
            FROM Sales.InvoiceLines aS il
            WHERE il.InvoiceID IN (SELECT InvoiceID FROM @InvoicesToGenerate)
            GROUP BY il.StockItemID
        )
        UPDATE sih
        SET sih.QuantityOnHand -= sit.TotalQuantity,
            sih.LastEditedBy = @InvoicedByPersonID,
            sih.LastEditedWhen = SYSDATETIME()
        FROM Warehouse.StockItemHoldings AS sih
        INNER JOIN StockItemTotals AS sit
        ON sih.StockItemID = sit.StockItemID;

        INSERT Sales.CustomerTransactions
            (CustomerID, TransactionTypeID, InvoiceID, PaymentMethodID,
             TransactionDate, AmountExcludingTax, TaxAmount, TransactionAmount,
             OutstandingBalance, FinalizationDate, LastEditedBy, LastEditedWhen)
        SELECT i.BillToCustomerID,
               (SELECT TransactionTypeID FROM [Application].TransactionTypes WHERE TransactionTypeName = ),
               itg.InvoiceID,
               NULL,
               SYSDATETIME(),
               (SELECT SUM(il.ExtendedPrice - il.TaxAmount) FROM Sales.InvoiceLines AS il WHERE il.InvoiceID = itg.
               InvoiceID),
               (SELECT SUM(il.TaxAmount) FROM Sales.InvoiceLines AS il WHERE il.InvoiceID = itg.InvoiceID),
               (SELECT SUM(il.ExtendedPrice) FROM Sales.InvoiceLines AS il WHERE il.InvoiceID = itg.InvoiceID),
               (SELECT SUM(il.ExtendedPrice) FROM Sales.InvoiceLines AS il WHERE il.InvoiceID = itg.InvoiceID),
               NULL,
               @InvoicedByPersonID,
               SYSDATETIME()
        FROM @InvoicesToGenerate AS itg
        INNER JOIN Sales.Invoices AS i
        ON itg.InvoiceID = i.InvoiceID;

        COMMIT;

    END TRY
    BEGIN CATCH
        IF XACT_STATE() <> 0 ROLLBACK;
        PRINT N'Unable to invoice these orders';
        THROW;
        RETURN -1;
    END CATCH;

    RETURN 0;
END;
GO
```

Depends On 13

- Website
- Website.OrderIDList
- Sequences.InvoiceID
- Sales.OrderLines
- Warehouse.StockItems
- Sales.Orders
- Sales.Invoices
- Sales.Customers
- Sales.InvoiceLines
- Warehouse.StockItemHoldings
- Warehouse.StockItemTransactions
- Application.TransactionTypes
- Sales.CustomerTransactions

Used By

No items found

Website.RecordColdRoomTemperatures

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@SensorReadings	SensorDataList	max	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Website.RecordColdRoomTemperatures
@SensorReadings Website.SensorDataList READONLY
WITH NATIVE_COMPILATION, SCHEMABINDING, EXECUTE AS OWNER
AS
BEGIN ATOMIC WITH
(
 TRANSACTION ISOLATION LEVEL = SNAPSHOT,
 LANGUAGE = N'English'
)
    BEGIN TRY

  DECLARE @NumberOfReadings int = (SELECT MAX(SensorDataListID) FROM @SensorReadings);
  DECLARE @Counter int = (SELECT MIN(SensorDataListID) FROM @SensorReadings);

  DECLARE @ColdRoomSensorNumber int;
  DECLARE @RecordedWhen datetime2(7);
  DECLARE @Temperature decimal(18,2);

  -- note that we cannot use a merge here because multiple readings might exist for each sensor

  WHILE @Counter <= @NumberOfReadings
  BEGIN
   SELECT @ColdRoomSensorNumber = ColdRoomSensorNumber,
          @RecordedWhen = RecordedWhen,
      @Temperature = Temperature
   FROM @SensorReadings
   WHERE SensorDataListID = @Counter;

   UPDATE Warehouse.ColdRoomTemperatures
    SET RecordedWhen = @RecordedWhen,
        Temperature = @Temperature
   WHERE ColdRoomSensorNumber = @ColdRoomSensorNumber;

   IF @@ROWCOUNT = 0
   BEGIN
    INSERT Warehouse.ColdRoomTemperatures
     (ColdRoomSensorNumber, RecordedWhen, Temperature)
    VALUES (@ColdRoomSensorNumber, @RecordedWhen, @Temperature);
```

```
    END;

    SET @Counter += 1;
  END;

    END TRY
    BEGIN CATCH
        THROW 51000, N'Unable to apply the sensor data', 2;

        RETURN 1;
    END CATCH;
END;
GO
```

Depends On 3

- Website
- Website.SensorDataList
- Warehouse.ColdRoomTemperatures

Used By

No items found

Website.RecordVehicleTemperature

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@FullSensorDataArray	nvarchar	1000	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Website.RecordVehicleTemperature
@FullSensorDataArray nvarchar(1000)
WITH EXECUTE AS OWNER
AS
BEGIN
    SET XACT_ABORT ON;

    DECLARE @CrLf nchar(2) = nchar(13) + nchar(10);
    DECLARE @HelpMessage nvarchar(max) = N'JSON sensor data is invalid. An example of what is required is as
    follows:' + @CrLf + @CrLf
              + N'{"Recordings":' + @CrLf
              + N' [' + @CrLf
              + N' {"type":"Feature", "geometry": {"type":"Point", "coordinates":[-89.7600464,
              50.4742420] }, "properties":{"rego":"WWI-321-A","sensor":1,"when":"2016-01-
01T07:00:00","temp":3.96}},' + @CrLf
              + N' {"type":"Feature", "geometry": {"type":"Point", "coordinates":[-89.7600464,
              50.4742420] }, "properties":{"rego":"WWI-321-A","sensor":2,"when":"2016-01-
01T07:00:00","temp":3.98}}' + @CrLf
              + N' ]' + @CrLf
              + N'}';

    IF ISJSON(@FullSensorDataArray) = 0
    BEGIN
        PRINT @HelpMessage;
        THROW 51000, N'FullSensorDataArray must be valid JSON data', 1;
        RETURN 1;
    END;

    BEGIN TRY

        BEGIN TRAN;

        INSERT Warehouse.VehicleTemperatures
            (VehicleRegistration, ChillerSensorNumber, RecordedWhen, Temperature,
    FullSensorData, IsCompressed, CompressedSensorData)
  SELECT VehicleRegistration, ChillerSensorNumber, RecordedWhen, Temperature,
         FullSensorData, 0, NULL
  FROM OPENJSON(@FullSensorDataArray, N'$.Recordings')
```

```
        WITH ( VehicleRegistration nvarchar(40) N'$.properties.rego',
               ChillerSensorNumber int N'$.properties.sensor',
            RecordedWhen datetime2(7) N'$.properties.when',
            Temperature decimal(18,2) N'$.properties.temp',
            FullSensorData nvarchar(max) N'$' AS JSON);

        IF @@ROWCOUNT = 0
        BEGIN
            PRINT N'Warning: No valid sensor data found';
            PRINT @HelpMessage;
        END;

        COMMIT;

    END TRY
    BEGIN CATCH
        PRINT @HelpMessage;

        THROW 51000, N'Valid JSON was supplied but does not match the temperature recordings array
        structure', 2;

        IF XACT_STATE() <> 0 ROLLBACK TRAN;

        RETURN 1;
    END CATCH;
END;
GO
```

Depends On 2

- Website
- Warehouse.VehicleTemperatures

Used By

No items found

Website.SearchForCustomers

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@SearchText	nvarchar	1000	
@MaximumRowsToReturn	int	4	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Website.SearchForCustomers
@SearchText nvarchar(1000),
@MaximumRowsToReturn int
WITH EXECUTE AS OWNER
AS
BEGIN
    SELECT TOP(@MaximumRowsToReturn)
           c.CustomerID,
           c.CustomerName,
           ct.CityName,
           c.PhoneNumber,
           c.FaxNumber,
           p.FullName AS PrimaryContactFullName,
           p.PreferredName AS PrimaryContactPreferredName
    FROM Sales.Customers AS c
    INNER JOIN [Application].Cities AS ct
    ON c.DeliveryCityID = ct.CityID
    LEFT OUTER JOIN [Application].People AS p
    ON c.PrimaryContactPersonID = p.PersonID
    WHERE CONCAT(c.CustomerName, N' ', p.FullName, N' ', p.PreferredName) LIKE N'%' + @SearchText + N'%'
    ORDER BY c.CustomerName
    FOR JSON AUTO, ROOT(N'Customers');
END;
GO
```

Depends On 4

- Website
- Sales.Customers
- Application.Cities
- Application.People

Used By

No items found

Website.SearchForPeople

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	
Assembly	

Parameters

Name	Data Type	Length	Description
@SearchText	nvarchar	1000	
@MaximumRowsToReturn	int	4	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Website.SearchForPeople
@SearchText nvarchar(1000),
@MaximumRowsToReturn int
AS
BEGIN
    SELECT TOP(@MaximumRowsToReturn)
           p.PersonID,
           p.FullName,
           p.PreferredName,
           CASE WHEN p.IsSalesperson <> 0 THEN N'Salesperson'
                WHEN p.IsEmployee <> 0 THEN N'Employee'
                WHEN c.CustomerID IS NOT NULL THEN N'Customer'
                WHEN sp.SupplierID IS NOT NULL THEN N'Supplier'
                WHEN sa.SupplierID IS NOT NULL THEN N'Supplier'
           END AS Relationship,
           COALESCE(c.CustomerName, sp.SupplierName, sa.SupplierName, N'WWI') AS Company
    FROM [Application].People AS p
    LEFT OUTER JOIN Sales.Customers AS c
    ON c.PrimaryContactPersonID = p.PersonID
    LEFT OUTER JOIN Purchasing.Suppliers AS sp
    ON sp.PrimaryContactPersonID = p.PersonID
    LEFT OUTER JOIN Purchasing.Suppliers AS sa
    ON sa.AlternateContactPersonID = p.PersonID
    WHERE p.SearchName LIKE N'%' + @SearchText + N'%'
    ORDER BY p.FullName
    FOR JSON AUTO, ROOT(N'People');
END;
GO
```

Depends On 4

- Website
- Application.People
- Sales.Customers

Purchasing.Suppliers

Used By

No items found

Website.SearchForStockItems

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@SearchText	nvarchar	1000	
@MaximumRowsToReturn	int	4	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Website.SearchForStockItems
@SearchText nvarchar(1000),
@MaximumRowsToReturn int
WITH EXECUTE AS OWNER
AS
BEGIN
    SELECT TOP(@MaximumRowsToReturn)
           si.StockItemID,
           si.StockItemName
    FROM Warehouse.StockItems AS si
    WHERE si.SearchDetails LIKE N'%' + @SearchText + N'%'
    ORDER BY si.StockItemName
    FOR JSON AUTO, ROOT(N'StockItems');
END;
GO
```

Depends On 2

- Website
- Warehouse.StockItems

Used By

No items found

Website.SearchForStockItemsByTags

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@SearchText	nvarchar	1000	
@MaximumRowsToReturn	int	4	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Website.SearchForStockItemsByTags
@SearchText nvarchar(1000),
@MaximumRowsToReturn int
WITH EXECUTE AS OWNER
AS
BEGIN
    SELECT TOP(@MaximumRowsToReturn)
           si.StockItemID,
           si.StockItemName
    FROM Warehouse.StockItems AS si
    WHERE si.Tags LIKE N'%' + @SearchText + N'%'
    ORDER BY si.StockItemName
    FOR JSON AUTO, ROOT(N'StockItems');
END;
GO
```

Depends On 2

- Website
- Warehouse.StockItems

Used By

No items found

Website.SearchForSuppliers

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	OWNER
Assembly	

Parameters

Name	Data Type	Length	Description
@SearchText	nvarchar	1000	
@MaximumRowsToReturn	int	4	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER PROCEDURE Website.SearchForSuppliers
@SearchText nvarchar(1000),
@MaximumRowsToReturn int
WITH EXECUTE AS OWNER
AS
BEGIN
    SELECT TOP(@MaximumRowsToReturn)
           s.SupplierID,
           s.SupplierName,
           c.CityName,
           s.PhoneNumber,
           s.FaxNumber ,
           p.FullName AS PrimaryContactFullName,
           p.PreferredName AS PrimaryContactPreferredName
    FROM Purchasing.Suppliers AS s
    INNER JOIN [Application].Cities AS c
    ON s.DeliveryCityID = c.CityID
    LEFT OUTER JOIN [Application].People AS p
    ON s.PrimaryContactPersonID = p.PersonID
    WHERE CONCAT(s.SupplierName, N' ', p.FullName, N' ', p.PreferredName) LIKE N'%' + @SearchText + N'%'
    ORDER BY s.SupplierName
    FOR JSON AUTO, ROOT(N'Suppliers');
END;
GO
```

Depends On 4

- Website
- Purchasing.Suppliers
- Application.Cities
- Application.People

Used By

No items found

Functions

Objects 2

- Table-Valued Functions
- Scalar-Valued Functions

Table-Valued Functions

Objects 1

Name	Description
Application.DetermineCustomerAccess	

Application.DetermineCustomerAccess

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	
Null on Null Input	False
Schema Bound	True
Assembly	

Parameters

Name	Data Type	Length	Description
@CityID	int	4	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER FUNCTION Application.DetermineCustomerAccess(@CityID int)
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN (SELECT 1 AS AccessResult
        WHERE IS_ROLEMEMBER(N'db_owner') <> 0
        OR IS_ROLEMEMBER((SELECT sp.SalesTerritory
                          FROM [Application].Cities AS c
                          INNER JOIN [Application].StateProvinces AS sp
                          ON c.StateProvinceID = sp.StateProvinceID
                          WHERE c.CityID = @CityID) + N' Sales') <> 0
     OR (ORIGINAL_LOGIN() = N'Website'
      AND EXISTS (SELECT 1
                  FROM [Application].Cities AS c
            INNER JOIN [Application].StateProvinces AS sp
            ON c.StateProvinceID = sp.StateProvinceID
            WHERE c.CityID = @CityID
            AND sp.SalesTerritory = SESSION_CONTEXT(N'SalesTerritory'))));
GO
```

Depends On 3

- Application
- Application.Cities
- Application.StateProvinces

Used By

No items found

Scalar-Valued Functions

Objects 1

Name	Description
Website.CalculateCustomerPrice	

Website.CalculateCustomerPrice

Description

Properties

Name	Value
ANSI Nulls ON	True
Quoted Identifier ON	True
Encrypted	False
Execute As	
Null on Null Input	False
Schema Bound	False
Assembly	

Parameters

Name	Data Type	Length	Description
(Result)	decimal	9	
@CustomerID	int	4	
@StockItemID	int	4	
@PricingDate	date	3	

SQL Script

```
SET QUOTED_IDENTIFIER, ANSI_NULLS ON
GO

CREATE OR ALTER FUNCTION Website.CalculateCustomerPrice
(
    @CustomerID int,
    @StockItemID int,
    @PricingDate date
)
RETURNS decimal(18,2)
WITH EXECUTE AS OWNER
AS
BEGIN
    DECLARE @CalculatedPrice decimal(18,2);
    DECLARE @UnitPrice decimal(18,2);
    DECLARE @LowestUnitPrice decimal(18,2);
    DECLARE @HighestDiscountAmount decimal(18,2);
    DECLARE @HighestDiscountPercentage decimal(18,3);
    DECLARE @BuyingGroupID int;
    DECLARE @CustomerCategoryID int;
    DECLARE @DiscountedUnitPrice decimal(18,2);

    SELECT @BuyingGroupID = BuyingGroupID,
           @CustomerCategoryID = CustomerCategoryID
    FROM Sales.Customers
    WHERE CustomerID = @CustomerID;

    SELECT @UnitPrice = si.UnitPrice
    FROM Warehouse.StockItems AS si
    WHERE si.StockItemID = @StockItemID;
```

```
    SET @CalculatedPrice = @UnitPrice;

    SET @LowestUnitPrice = (SELECT MIN(sd.UnitPrice)
                            FROM Sales.SpecialDeals AS sd
                            WHERE ((sd.StockItemID = @StockItemID) OR (sd.StockItemID IS NULL))
                            AND ((sd.CustomerID = @CustomerID) OR (sd.CustomerID IS NULL))
                            AND ((sd.BuyingGroupID = @BuyingGroupID) OR (sd.BuyingGroupID IS NULL))
                            AND ((sd.CustomerCategoryID = @CustomerCategoryID) OR (sd.CustomerCategoryID IS NULL))
                            AND ((sd.StockGroupID IS NULL) OR EXISTS (SELECT 1 FROM Warehouse.StockItemStockGroups
                            AS sisg
                                                                      WHERE sisg.StockItemID =
                                                                      @StockItemID
                                                                      AND sisg.StockGroupID = sd.
                                                                      StockGroupID))
                            AND sd.UnitPrice IS NOT NULL
                            AND @PricingDate BETWEEN sd.StartDate AND sd.EndDate);

    IF @LowestUnitPrice IS NOT NULL AND @LowestUnitPrice < @UnitPrice
    BEGIN
        SET @CalculatedPrice = @LowestUnitPrice;
    END;

    SET @HighestDiscountAmount = (SELECT MAX(sd.DiscountAmount)
                                  FROM Sales.SpecialDeals AS sd
                                  WHERE ((sd.StockItemID = @StockItemID) OR (sd.StockItemID IS NULL))
                                  AND ((sd.CustomerID = @CustomerID) OR (sd.CustomerID IS NULL))
                                  AND ((sd.BuyingGroupID = @BuyingGroupID) OR (sd.BuyingGroupID IS NULL))
                                  AND ((sd.CustomerCategoryID = @CustomerCategoryID) OR (sd.CustomerCategoryID IS
                                  NULL))
                                  AND ((sd.StockGroupID IS NULL) OR EXISTS (SELECT 1 FROM Warehouse.
                                  StockItemStockGroups AS sisg
                                                                            WHERE sisg.StockItemID =
                                                                            @StockItemID
                                                                            AND sisg.StockGroupID = sd.
                                                                            StockGroupID))
                                  AND sd.DiscountAmount IS NOT NULL
                                  AND @PricingDate BETWEEN sd.StartDate AND sd.EndDate);

    IF @HighestDiscountAmount IS NOT NULL AND (@UnitPrice - @HighestDiscountAmount) < @CalculatedPrice
    BEGIN
        SET @CalculatedPrice = @UnitPrice - @HighestDiscountAmount;
    END;

    SET @HighestDiscountPercentage = (SELECT MAX(sd.DiscountPercentage)
                                      FROM Sales.SpecialDeals AS sd
                                      WHERE ((sd.StockItemID = @StockItemID) OR (sd.StockItemID IS NULL))
                                      AND ((sd.CustomerID = @CustomerID) OR (sd.CustomerID IS NULL))
                                      AND ((sd.BuyingGroupID = @BuyingGroupID) OR (sd.BuyingGroupID IS NULL))
                                      AND ((sd.CustomerCategoryID = @CustomerCategoryID) OR (sd.CustomerCategoryID
                                      IS NULL))
                                      AND ((sd.StockGroupID IS NULL) OR EXISTS (SELECT 1 FROM Warehouse.
                                      StockItemStockGroups AS sisg
                                                                                WHERE sisg.StockItemID =
                                                                                @StockItemID
                                                                                AND sisg.StockGroupID = sd
                                                                                .StockGroupID))
                                      AND sd.DiscountPercentage IS NOT NULL
                                      AND @PricingDate BETWEEN sd.StartDate AND sd.EndDate);

    IF @HighestDiscountPercentage IS NOT NULL
    BEGIN
        SET @DiscountedUnitPrice = ROUND(@UnitPrice * @HighestDiscountPercentage / 100.0, 2);
        IF @DiscountedUnitPrice < @CalculatedPrice SET @CalculatedPrice = @DiscountedUnitPrice;
    END;


    RETURN @CalculatedPrice;
END;
GO
```

Types

Objects 1

User-Defined Table Types

User-Defined Table Types

Objects 4

Name	Description
Website.OrderIDList	
Website.OrderLineList	
Website.OrderList	
Website.SensorDataList	

Website.OrderIDList

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	
Row Count (~)	
Created	
Last Modified	

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Default	Computed	Persisted	Description
🔑	OrderID	int	4	10	0	True			False	False	

SQL Script

```
CREATE TYPE Website.OrderIDList AS TABLE (
  OrderID int NOT NULL,
  PRIMARY KEY NONCLUSTERED (OrderID)
)
WITH (MEMORY_OPTIMIZED = ON)
GO
```

Depends On 1

Website

Used By 1

Website.InvoiceCustomerOrders

Website.OrderLineList

Description

Properties

Name	Value
Collation	Latin1_General_100_CI_AS
File Group	
Filestream File Group	
Is Partitioned	
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	
Row Count (~)	
Created	
Last Modified	

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Default	Computed	Persisted	Description
	OrderReference	int	4	10	0	False			False	False	
	StockItemID	int	4	10	0	False			False	False	
	Description	nvarchar	100	0	0	False			False	False	
	Quantity	int	4	10	0	False			False	False	

SQL Script

```
CREATE TYPE Website.OrderLineList AS TABLE (
  OrderReference int NULL,
  StockItemID int NULL,
  Description nvarchar(100) NULL,
  Quantity int NULL,
  INDEX IX_Website_OrderLineList (OrderReference)
)
WITH (MEMORY_OPTIMIZED = ON)
GO
```

Depends On 1

- Website

Used By 1

- Website.InsertCustomerOrders

Website.OrderList

Description

Properties

Name	Value
Collation	Latin1_General_100_CI_AS
File Group	
Filestream File Group	
Is Partitioned	
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	
Row Count (~)	
Created	
Last Modified	

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Default	Computed	Persisted	Description
🔑	OrderReference	int	4	10	0	True			False	False	
	CustomerID	int	4	10	0	False			False	False	
	ContactPersonID	int	4	10	0	False			False	False	
	ExpectedDeliveryDate	date	3	10	0	False			False	False	
	CustomerPurchaseOrderNumber	nvarchar	20	0	0	False			False	False	
	IsUndersupplyBackordered	bit	1	1	0	False			False	False	
	Comments	nvarchar		0	0	False			False	False	
	DeliveryInstructions	nvarchar		0	0	False			False	False	

SQL Script

```
CREATE TYPE Website.OrderList AS TABLE (
  OrderReference int NOT NULL,
  CustomerID int NULL,
```

```
  ContactPersonID int NULL,
  ExpectedDeliveryDate date NULL,
  CustomerPurchaseOrderNumber nvarchar(20) NULL,
  IsUndersupplyBackordered bit NULL,
  Comments nvarchar(max) NULL,
  DeliveryInstructions nvarchar(max) NULL,
  PRIMARY KEY NONCLUSTERED (OrderReference)
)
WITH (MEMORY_OPTIMIZED = ON)
GO
```

Depends On 1

Website

Used By 1

Website.InsertCustomerOrders

Website.SensorDataList

Description

Properties

Name	Value
Collation	
File Group	
Filestream File Group	
Is Partitioned	
Partitioned Column	
Partition Scheme	
File Groups	
Filestream Partition Scheme	
Filestream File Groups	
Heap	False
Full Text Catalog	
Full Text	
Compression	
Row Count (~)	
Created	
Last Modified	

Columns

Key	Name	Data Type	Length	Precision	Scale	Not Null	Identity	Default	Computed	Persisted	Description
🔑	SensorDataListID	int	4	10	0	True	1 - 1		False	False	
	ColdRoomSensorNumber	int	4	10	0	False			False	False	
	RecordedWhen	datetime2	8	27	7	False			False	False	
	Temperature	decimal	9	18	2	False			False	False	

SQL Script

```
CREATE TYPE Website.SensorDataList AS TABLE (
  SensorDataListID int IDENTITY,
  ColdRoomSensorNumber int NULL,
  RecordedWhen datetime2 NULL,
  Temperature decimal(18, 2) NULL,
  PRIMARY KEY NONCLUSTERED (SensorDataListID)
)
WITH (MEMORY_OPTIMIZED = ON)
GO
```

Depends On 1

Website

Used By 1

Website.RecordColdRoomTemperatures

Sequences

Objects 26

Name	Description
Sequences.BuyingGroupID	
Sequences.CityID	
Sequences.ColorID	
Sequences.CountryID	
Sequences.CustomerCategoryID	
Sequences.CustomerID	
Sequences.DeliveryMethodID	
Sequences.InvoiceID	
Sequences.InvoiceLineID	
Sequences.OrderID	
Sequences.OrderLineID	
Sequences.PackageTypeID	
Sequences.PaymentMethodID	
Sequences.PersonID	
Sequences.PurchaseOrderID	
Sequences.PurchaseOrderLineID	
Sequences.SpecialDealID	
Sequences.StateProvinceID	
Sequences.StockGroupID	
Sequences.StockItemID	
Sequences.StockItemStockGroupID	
Sequences.SupplierCategoryID	
Sequences.SupplierID	
Sequences.SystemParameterID	
Sequences.TransactionID	
Sequences.TransactionTypeID	

Sequences.BuyingGroupID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	3
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.BuyingGroupID
  AS int
  START WITH 3
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Sales.BuyingGroups

Sequences.CityID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	38187
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.CityID
  AS int
  START WITH 38187
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Application.Cities

Sequences.ColorID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	37
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.ColorID
  AS int
  START WITH 37
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Warehouse.Colors

Sequences.CountryID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	242
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.CountryID
  AS int
  START WITH 242
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Application.Countries

Sequences.CustomerCategoryID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	9
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.CustomerCategoryID
  AS int
  START WITH 9
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Sales.CustomerCategories

Sequences.CustomerID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	1050
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.CustomerID
  AS int
  START WITH 1050
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Sales.Customers

Sequences.DeliveryMethodID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	11
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.DeliveryMethodID
  AS int
  START WITH 11
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Application.DeliveryMethods

Sequences.InvoiceID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	67275
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.InvoiceID
  AS int
  START WITH 67275
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

- Sequences

Used By 2

- Sales.Invoices
- Website.InvoiceCustomerOrders

Sequences.InvoiceLineID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	222092
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.InvoiceLineID
  AS int
  START WITH 222092
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Sales.InvoiceLines

Sequences.OrderID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	69488
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.OrderID
  AS int
  START WITH 69488
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

- Sequences

Used By 2

- Sales.Orders
- Website.InsertCustomerOrders

Sequences.OrderLineID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	224338
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.OrderLineID
  AS int
  START WITH 224338
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Sales.OrderLines

Sequences.PackageTypeID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	15
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.PackageTypeID
  AS int
  START WITH 15
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Warehouse.PackageTypes

Sequences.PaymentMethodID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	5
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.PaymentMethodID
  AS int
  START WITH 5
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Application.PaymentMethods

Sequences.PersonID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	3250
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.PersonID
  AS int
  START WITH 3250
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Application.People

Sequences.PurchaseOrderID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	1997
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.PurchaseOrderID
  AS int
  START WITH 1997
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Purchasing.PurchaseOrders

Sequences.PurchaseOrderLineID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	5735
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.PurchaseOrderLineID
  AS int
  START WITH 5735
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Purchasing.PurchaseOrderLines

Sequences.SpecialDealID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	3
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.SpecialDealID
  AS int
  START WITH 3
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Sales.SpecialDeals

Sequences.StateProvinceID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	54
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.StateProvinceID
  AS int
  START WITH 54
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Application.StateProvinces

Sequences.StockGroupID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	11
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.StockGroupID
  AS int
  START WITH 11
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Warehouse.StockGroups

Sequences.StockItemID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	228
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.StockItemID
  AS int
  START WITH 228
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Warehouse.StockItems

Sequences.StockItemStockGroupID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	443
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.StockItemStockGroupID
  AS int
  START WITH 443
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Warehouse.StockItemStockGroups

Sequences.SupplierCategoryID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	10
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.SupplierCategoryID
  AS int
  START WITH 10
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Purchasing.SupplierCategories

Sequences.SupplierID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	14
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.SupplierID
  AS int
  START WITH 14
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Purchasing.Suppliers

Sequences.SystemParameterID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	2
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.SystemParameterID
  AS int
  START WITH 2
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Application.SystemParameters

Sequences.TransactionID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	322635
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.TransactionID
  AS int
  START WITH 322635
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

- Sequences

Used By 3

- Purchasing.SupplierTransactions
- Sales.CustomerTransactions
- Warehouse.StockItemTransactions

Sequences.TransactionTypeID

Description

Properties

Name	Value
Owner	Sequences
Data Type	int
Start With	14
Increment By	1
Min value	-2147483648
Max value	2147483647
Cycle	False
Cache	

SQL Script

```
CREATE SEQUENCE Sequences.TransactionTypeID
  AS int
  START WITH 14
  INCREMENT BY 1
  NO CYCLE
  CACHE
GO
```

Depends On 1

Sequences

Used By 1

Application.TransactionTypes

Storage

Objects 2

- Partition Functions
- Partition Schemes

Partition Functions

Objects 2

Name	Description
PF_TransactionDate	
PF_TransactionDateTime	

PF_TransactionDate

Description

Properties

Name	Value
Owner	WideWorldImporters
Range	Right
Input Parameter Data Type	date

Values 4

'2014-01-01'
'2015-01-01'
'2016-01-01'
'2017-01-01'

SQL Script

```
CREATE PARTITION FUNCTION PF_TransactionDate (date)
  AS RANGE RIGHT FOR VALUES ('2014-01-01', '2015-01-01', '2016-01-01', '2017-01-01')
GO
```

Depends On

No items found

Used By 1

PS_TransactionDate

PF_TransactionDateTime

Description

Properties

Name	Value
Owner	WideWorldImporters
Range	Right
Input Parameter Data Type	datetime

Values 4

'2014-01-01 00:00:00.000'
'2015-01-01 00:00:00.000'
'2016-01-01 00:00:00.000'
'2017-01-01 00:00:00.000'

SQL Script

```
CREATE PARTITION FUNCTION PF_TransactionDateTime (datetime)
  AS RANGE RIGHT FOR VALUES ('2014-01-01 00:00:00.000', '2015-01-01 00:00:00.000', '2016-01-01 00:00:00.000', '2017-01-
01 00:00:00.000')
GO
```

Depends On

No items found

Used By 1

PS_TransactionDateTime

Partition Schemes

Objects 2

Name	Description
PS_TransactionDate	
PS_TransactionDateTime	

PS_TransactionDate

Description

Properties

Name	Value
Owner	WideWorldImporters
Function	PF_TransactionDate

FileGroups 6

PRIMARY
PRIMARY
PRIMARY
PRIMARY
PRIMARY
PRIMARY

SQL Script

```
CREATE PARTITION SCHEME PS_TransactionDate
  AS PARTITION PF_TransactionDate ALL TO ([PRIMARY])
GO
```

Depends On 1

PF_TransactionDate

Used By 2

Purchasing.SupplierTransactions
Sales.CustomerTransactions

PS_TransactionDateTime

Description

Properties

Name	Value
Owner	WideWorldImporters
Function	PF_TransactionDateTime

FileGroups 6

PRIMARY
PRIMARY
PRIMARY
PRIMARY
PRIMARY
PRIMARY

SQL Script

```
CREATE PARTITION SCHEME PS_TransactionDateTime
  AS PARTITION PF_TransactionDateTime ALL TO ([PRIMARY])
GO
```

Depends On 1

PF_TransactionDateTime

Used By

No items found

Security

Objects 2

- Roles
- Schemas

Roles

Objects 1

Database Roles

Database Roles

Objects 19

Name	Description
db_accessadmin	
db_backupoperator	
db_datareader	
db_datawriter	
db_ddladmin	
db_denydatareader	
db_denydatawriter	
db_owner	
db_securityadmin	
External Sales	
Far West Sales	
Great Lakes Sales	
Mideast Sales	
New England Sales	
Plains Sales	
public	
Rocky Mountain Sales	
Southeast Sales	
Southwest Sales	

db_accessadmin

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE db_accessadmin
GO
```

Depends On 1

dbo

Used By

No items found

db_backupoperator

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE db_backupoperator
GO
```

Depends On 1

dbo

Used By

No items found

db_datareader

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE db_datareader
GO
```

Depends On 1

dbo

Used By

No items found

db_datawriter

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE db_datawriter
GO
```

Depends On 1

dbo

Used By

No items found

db_ddladmin

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE db_ddladmin
GO
```

Depends On 1

dbo

Used By

No items found

db_denydatareader

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE db_denydatareader
GO
```

Depends On 1

dbo

Used By

No items found

db_denydatawriter

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE db_denydatawriter
GO
```

Depends On 1

dbo

Used By

No items found

db_owner

Properties

Name	Value
Owner	dbo

Members 1

dbo

SQL Script

```
CREATE ROLE db_owner
GO

EXEC sp_addrolemember N'db_owner', N'dbo'
GO
```

Depends On 1

dbo

Used By

No items found

db_securityadmin

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE db_securityadmin
GO
```

Depends On 1

dbo

Used By

No items found

External Sales

Description

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE [External Sales]
GO
```

Depends On 1

dbo

Used By

No items found

Far West Sales

Description

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE [Far West Sales]
GO
```

Depends On 1

dbo

Used By

No items found

Great Lakes Sales

Description

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE [Great Lakes Sales]
GO
```

Depends On 1

dbo

Used By

No items found

Mideast Sales

Description

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE [Mideast Sales]
GO
```

Depends On 1

dbo

Used By

No items found

New England Sales

Description

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE [New England Sales]
GO
```

Depends On 1

dbo

Used By

No items found

Plains Sales

Description

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE [Plains Sales]
GO
```

Depends On 1

dbo

Used By

No items found

public

Properties

Name	Value
Owner	dbo

Members

No items found

Database Level Permissions

Type	Action
Grant	VIEW ANY COLUMN ENCRYPTION KEY DEFINITION
Grant	VIEW ANY COLUMN MASTER KEY DEFINITION

SQL Script

```
CREATE ROLE [public]
GO
```

Depends On 1

dbo

Used By

No items found

Rocky Mountain Sales

Description

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE [Rocky Mountain Sales]
GO
```

Depends On 1

dbo

Used By

No items found

Southeast Sales

Description

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE [Southeast Sales]
GO
```

Depends On 1

dbo

Used By

No items found

Southwest Sales

Description

Properties

Name	Value
Owner	dbo

Members

No items found

SQL Script

```
CREATE ROLE [Southwest Sales]
GO
```

Depends On 1

dbo

Used By

No items found

Schemas

Objects 23

Name	Description
Application	
DataLoadSimulation	
db_accessadmin	
db_backupoperator	
db_datareader	
db_datawriter	
db_ddladmin	
db_denydatareader	
db_denydatawriter	
db_owner	
db_securityadmin	
dbo	
guest	
INFORMATION_SCHEMA	
Integration	
PowerBI	
Purchasing	
Reports	
Sales	
Sequences	
sys	
Warehouse	
Website	

Application

Description

Properties

Name	Value
Owner	dbo

Extended Properties

Name	Value
Description	Tables common across the application. Used for categorization and lookup lists, system parameters and people (users and contacts)

SQL Script

```
CREATE SCHEMA Application AUTHORIZATION dbo
GO

EXEC sys.sp_addextendedproperty N'Description', N'Tables common across the application. Used for categorization and
lookup lists,
system parameters and people (users and contacts)', 'SCHEMA', N'Application'
GO
```

Depends On 1

- dbo

Used By 27

- Application.Cities
- Application.Cities_Archive
- Application.Countries
- Application.Countries_Archive
- Application.DeliveryMethods
- Application.DeliveryMethods_Archive
- Application.PaymentMethods
- Application.PaymentMethods_Archive
- Application.People
- Application.People_Archive
- Application.StateProvinces
- Application.StateProvinces_Archive
- Application.SystemParameters
- Application.TransactionTypes
- Application.TransactionTypes_Archive
- Application.AddRoleMemberIfNonexistent
- Application.Configuration_ApplyAuditing
- Application.Configuration_ApplyColumnstoreIndexing
- Application.Configuration_ApplyFullTextIndexing
- Application.Configuration_ApplyPartitioning
- Application.Configuration_ApplyRowLevelSecurity
- Application.Configuration_ConfigureForEnterpriseEdition
- Application.Configuration_EnableInMemory

 Application.Configuration_RemoveAuditing

 Application.Configuration_RemoveRowLevelSecurity

 Application.CreateRoleIfNonexistent

 Application.DetermineCustomerAccess

DataLoadSimulation

Description

Properties

Name	Value
Owner	dbo

Extended Properties

Name	Value
Description	Tables and procedures used only during simulated data loading operations

SQL Script

```
CREATE SCHEMA DataLoadSimulation AUTHORIZATION dbo
GO

EXEC sys.sp_addextendedproperty N'Description', N'Tables and procedures used only during simulated data loading
operations', 'SCHEMA', N'DataLoadSimulation'
GO
```

Depends On 1

- dbo

Used By 5

- DataLoadSimulation.Configuration_ApplyDataLoadSimulationProcedures
- DataLoadSimulation.Configuration_RemoveDataLoadSimulationProcedures
- DataLoadSimulation.DeactivateTemporalTablesBeforeDataLoad
- DataLoadSimulation.PopulateDataToCurrentDate
- DataLoadSimulation.ReactivateTemporalTablesAfterDataLoad

db_accessadmin

Properties

Name	Value
Owner	db_accessadmin

SQL Script

```
CREATE SCHEMA db_accessadmin AUTHORIZATION db_accessadmin
GO
```

Depends On

No items found

Used By

No items found

db_backupoperator

Properties

Name	Value
Owner	db_backupoperator

SQL Script

```
CREATE SCHEMA db_backupoperator AUTHORIZATION db_backupoperator
GO
```

Depends On

No items found

Used By

No items found

db_datareader

Properties

Name	Value
Owner	db_datareader

SQL Script

```
CREATE SCHEMA db_datareader AUTHORIZATION db_datareader
GO
```

Depends On

No items found

Used By

No items found

db_datawriter

Properties

Name	Value
Owner	db_datawriter

SQL Script

```
CREATE SCHEMA db_datawriter AUTHORIZATION db_datawriter
GO
```

Depends On

No items found

Used By

No items found

db_ddladmin

Properties

Name	Value
Owner	db_ddladmin

SQL Script

```
CREATE SCHEMA db_ddladmin AUTHORIZATION db_ddladmin
GO
```

Depends On

No items found

Used By

No items found

db_denydatareader

Properties

Name	Value
Owner	db_denydatareader

SQL Script

```
CREATE SCHEMA db_denydatareader AUTHORIZATION db_denydatareader
GO
```

Depends On

No items found

Used By

No items found

db_denydatawriter

Properties

Name	Value
Owner	db_denydatawriter

SQL Script

```
CREATE SCHEMA db_denydatawriter AUTHORIZATION db_denydatawriter
GO
```

Depends On

No items found

Used By

No items found

db_owner

Properties

Name	Value
Owner	db_owner

SQL Script

```
CREATE SCHEMA db_owner AUTHORIZATION db_owner
GO
```

Depends On

No items found

Used By

No items found

db_securityadmin

Properties

Name	Value
Owner	db_securityadmin

SQL Script

```
CREATE SCHEMA db_securityadmin AUTHORIZATION db_securityadmin
GO
```

Depends On

No items found

Used By

No items found

dbo

Description

Properties

Name	Value
Owner	dbo

SQL Script

```
CREATE SCHEMA dbo AUTHORIZATION dbo
GO
```

Depends On

No items found

Used By

No items found

guest

Description

Properties

Name	Value
Owner	guest

SQL Script

```
CREATE SCHEMA guest AUTHORIZATION guest
GO
```

Depends On

No items found

Used By

No items found

INFORMATION_SCHEMA

Properties

Name	Value
Owner	INFORMATION_SCHEMA

SQL Script

```
CREATE SCHEMA INFORMATION_SCHEMA AUTHORIZATION INFORMATION_SCHEMA
GO
```

Depends On

No items found

Used By

No items found

Integration

Description

Properties

Name	Value
Owner	dbo

Extended Properties

Name	Value
Description	Tables and procedures required for integration with the data warehouse

SQL Script

```
CREATE SCHEMA Integration AUTHORIZATION dbo
GO

EXEC sys.sp_addextendedproperty N'Description', N'Tables and procedures required for integration with the data
warehouse', 'SCHEMA', N'Integration'
GO
```

Depends On 1

- dbo

Used By 13

- Integration.GetCityUpdates
- Integration.GetCustomerUpdates
- Integration.GetEmployeeUpdates
- Integration.GetMovementUpdates
- Integration.GetOrderUpdates
- Integration.GetPaymentMethodUpdates
- Integration.GetPurchaseUpdates
- Integration.GetSaleUpdates
- Integration.GetStockHoldingUpdates
- Integration.GetStockItemUpdates
- Integration.GetSupplierUpdates
- Integration.GetTransactionTypeUpdates
- Integration.GetTransactionUpdates

PowerBI

Description

Properties

Name	Value
Owner	dbo

Extended Properties

Name	Value
Description	Views and stored procedures that provide the only access for the Power BI dashboard system

SQL Script

```
CREATE SCHEMA PowerBI AUTHORIZATION dbo
GO

EXEC sys.sp_addextendedproperty N'Description', N'Views and stored procedures that provide the only access for the Power
BI
dashboard system', 'SCHEMA', N'PowerBI'
GO
```

Depends On 1

dbo

Used By

No items found

Purchasing

Description

Properties

Name	Value
Owner	dbo

Extended Properties

Name	Value
Description	Details of suppliers and of purchasing of stock items

SQL Script

```
CREATE SCHEMA Purchasing AUTHORIZATION dbo
GO

EXEC sys.sp_addextendedproperty N'Description', N'Details of suppliers and of purchasing of stock
items', 'SCHEMA', N'Purchasing'
GO
```

Depends On 1

- dbo

Used By 7

- Purchasing.PurchaseOrderLines
- Purchasing.PurchaseOrders
- Purchasing.SupplierCategories
- Purchasing.SupplierCategories_Archive
- Purchasing.Suppliers
- Purchasing.Suppliers_Archive
- Purchasing.SupplierTransactions

Reports

Description

Properties

Name	Value
Owner	dbo

Extended Properties

Name	Value
Description	Views and stored procedures that provide the only access for the reporting system

SQL Script

```
CREATE SCHEMA Reports AUTHORIZATION dbo
GO

EXEC sys.sp_addextendedproperty N'Description', N'Views and stored procedures that provide the only access for the
reporting
system', 'SCHEMA', N'Reports'
GO
```

Depends On 1

dbo

Used By

No items found

Sales

Description

Properties

Name	Value
Owner	dbo

Extended Properties

Name	Value
Description	Details of customers, salespeople, and of sales of stock items

SQL Script

```
CREATE SCHEMA Sales AUTHORIZATION dbo
GO

EXEC sys.sp_addextendedproperty N'Description', N'Details of customers, salespeople, and of sales of stock
items', 'SCHEMA', N'Sales'
GO
```

Depends On 1

- dbo

Used By 12

- Sales.BuyingGroups
- Sales.BuyingGroups_Archive
- Sales.CustomerCategories
- Sales.CustomerCategories_Archive
- Sales.Customers
- Sales.Customers_Archive
- Sales.CustomerTransactions
- Sales.InvoiceLines
- Sales.Invoices
- Sales.OrderLines
- Sales.Orders
- Sales.SpecialDeals

Sequences

Description

Properties

Name	Value
Owner	dbo

Extended Properties

Name	Value
Description	Holds sequences used by all tables in the application

SQL Script

```
CREATE SCHEMA Sequences AUTHORIZATION dbo
GO

EXEC sys.sp_addextendedproperty N'Description', N'Holds sequences used by all tables in the
application', 'SCHEMA', N'Sequences'
GO
```

Depends On 1

- dbo

Used By 28

- Sequences.ReseedAllSequences
- Sequences.ReseedSequenceBeyondTableValues
- Sequences.BuyingGroupID
- Sequences.CityID
- Sequences.ColorID
- Sequences.CountryID
- Sequences.CustomerCategoryID
- Sequences.CustomerID
- Sequences.DeliveryMethodID
- Sequences.InvoiceID
- Sequences.InvoiceLineID
- Sequences.OrderID
- Sequences.OrderLineID
- Sequences.PackageTypeID
- Sequences.PaymentMethodID
- Sequences.PersonID
- Sequences.PurchaseOrderID
- Sequences.PurchaseOrderLineID
- Sequences.SpecialDealID
- Sequences.StateProvinceID
- Sequences.StockGroupID
- Sequences.StockItemID
- Sequences.StockItemStockGroupID
- Sequences.SupplierCategoryID

- Sequences.SupplierID
- Sequences.SystemParameterID
- Sequences.TransactionID
- Sequences.TransactionTypeID

sys

Properties

Name	Value
Owner	sys

SQL Script

```
CREATE SCHEMA sys AUTHORIZATION sys
GO
```

Depends On

No items found

Used By

No items found

Warehouse

Description

Properties

Name	Value
Owner	dbo

Extended Properties

Name	Value
Description	Details of stock items, their holdings and transactions

SQL Script

```
CREATE SCHEMA Warehouse AUTHORIZATION dbo
GO

EXEC sys.sp_addextendedproperty N'Description', N'Details of stock items, their holdings and
transactions', 'SCHEMA', N'Warehouse'
GO
```

Depends On 1

- dbo

Used By 14

- Warehouse.ColdRoomTemperatures
- Warehouse.ColdRoomTemperatures_Archive
- Warehouse.Colors
- Warehouse.Colors_Archive
- Warehouse.PackageTypes
- Warehouse.PackageTypes_Archive
- Warehouse.StockGroups
- Warehouse.StockGroups_Archive
- Warehouse.StockItemHoldings
- Warehouse.StockItems
- Warehouse.StockItems_Archive
- Warehouse.StockItemStockGroups
- Warehouse.StockItemTransactions
- Warehouse.VehicleTemperatures

Website

Description

Properties

Name	Value
Owner	dbo

Extended Properties

Name	Value
Description	Views and stored procedures that provide the only access for the application website

SQL Script

```
CREATE SCHEMA Website AUTHORIZATION dbo
GO

EXEC sys.sp_addextendedproperty N'Description', N'Views and stored procedures that provide the only access for the
application
website', 'SCHEMA', N'Website'
GO
```

Depends On 1

- dbo

Used By 19

- Website.Customers
- Website.Suppliers
- Website.VehicleTemperatures
- Website.ActivateWebsiteLogon
- Website.ChangePassword
- Website.InsertCustomerOrders
- Website.InvoiceCustomerOrders
- Website.RecordColdRoomTemperatures
- Website.RecordVehicleTemperature
- Website.SearchForCustomers
- Website.SearchForPeople
- Website.SearchForStockItems
- Website.SearchForStockItemsByTags
- Website.SearchForSuppliers
- Website.CalculateCustomerPrice
- Website.OrderIDList
- Website.OrderLineList
- Website.OrderList
- Website.SensorDataList