:setvar DatabaseName "testDB"

USE [master];

GO

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Drop Database

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PRINT '';

PRINT '\*\*\* Dropping Database';

GO

IF EXISTS (SELECT [name] FROM [master].[sys].[databases] WHERE [name] = N'$(DatabaseName)')

DROP DATABASE $(DatabaseName);

-- If the database has any other open connections close the network connection.

IF @@ERROR = 3702

RAISERROR('$(DatabaseName) database cannot be dropped because there are still other open connections', 127, 127) WITH NOWAIT, LOG;

GO

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Create Database

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PRINT '';

PRINT '\*\*\* Creating Database';

GO

CREATE DATABASE $(DatabaseName);

GO

PRINT '';

PRINT '\*\*\* Checking for $(DatabaseName) Database';

/\* CHECK FOR DATABASE IF IT DOESN'T EXISTS, DO NOT RUN THE REST OF THE SCRIPT \*/

IF NOT EXISTS (SELECT TOP 1 1 FROM sys.databases WHERE name = N'$(DatabaseName)')

BEGIN

PRINT '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'

+char(10)+'\*\*\*\*\*\*\*\*$(DatabaseName) Database does not exist. Make sure that the script is being run in SQLCMD mode and that the variables have been correctly set.\*\*\*\*\*\*\*\*\*'

+char(10)+'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*';

SET NOEXEC ON;

END

GO

ALTER DATABASE $(DatabaseName)

SET RECOVERY SIMPLE,

ANSI\_NULLS ON,

ANSI\_PADDING ON,

ANSI\_WARNINGS ON,

ARITHABORT ON,

CONCAT\_NULL\_YIELDS\_NULL ON,

QUOTED\_IDENTIFIER ON,

NUMERIC\_ROUNDABORT OFF,

PAGE\_VERIFY CHECKSUM,

ALLOW\_SNAPSHOT\_ISOLATION OFF;

GO

USE $(DatabaseName);

GO

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Create DDL Trigger for Database

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PRINT '';

PRINT '\*\*\* Creating DDL Trigger for Database';

GO

-- Create table to store database object creation messages

-- \*\*\* WARNING: THIS TABLE IS INTENTIONALLY A HEAP - DO NOT ADD A PRIMARY KEY \*\*\*

CREATE TABLE [dbo].[DatabaseLog](

[DatabaseLogID] [int] IDENTITY (1, 1) NOT NULL,

[PostTime] [datetime] NOT NULL,

[DatabaseUser] [sysname] NOT NULL,

[Event] [sysname] NOT NULL,

[Schema] [sysname] NULL,

[Object] [sysname] NULL,

[TSQL] [nvarchar](max) NOT NULL,

[XmlEvent] [xml] NOT NULL

) ON [PRIMARY];

GO

CREATE TRIGGER [ddlDatabaseTriggerLog] ON DATABASE

FOR DDL\_DATABASE\_LEVEL\_EVENTS AS

BEGIN

SET NOCOUNT ON;

DECLARE @data XML;

DECLARE @schema sysname;

DECLARE @object sysname;

DECLARE @eventType sysname;

SET @data = EVENTDATA();

SET @eventType = @data.value('(/EVENT\_INSTANCE/EventType)[1]', 'sysname');

SET @schema = @data.value('(/EVENT\_INSTANCE/SchemaName)[1]', 'sysname');

SET @object = @data.value('(/EVENT\_INSTANCE/ObjectName)[1]', 'sysname')

IF @object IS NOT NULL

PRINT ' ' + @eventType + ' - ' + @schema + '.' + @object;

ELSE

PRINT ' ' + @eventType + ' - ' + @schema;

IF @eventType IS NULL

PRINT CONVERT(nvarchar(max), @data);

INSERT [dbo].[DatabaseLog]

(

[PostTime],

[DatabaseUser],

[Event],

[Schema],

[Object],

[TSQL],

[XmlEvent]

)

VALUES

(

GETDATE(),

CONVERT(sysname, CURRENT\_USER),

@eventType,

CONVERT(sysname, @schema),

CONVERT(sysname, @object),

@data.value('(/EVENT\_INSTANCE/TSQLCommand)[1]', 'nvarchar(max)'),

@data

);

END;

GO

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Create User Defined Functions

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Builds an ISO 8601 format date from a year, month, and day specified as integers.

-- This format of date should parse correctly regardless of SET DATEFORMAT and SET LANGUAGE.

-- See SQL Server Books Online for more details.

CREATE FUNCTION [dbo].[udfBuildISO8601Date] (@year int, @month int, @day int)

RETURNS datetime

AS

BEGIN

RETURN cast(convert(varchar, @year) + '-' + [dbo].[udfTwoDigitZeroFill](@month)

+ '-' + [dbo].[udfTwoDigitZeroFill](@day) + 'T00:00:00'

as datetime);

END;

GO

CREATE FUNCTION [dbo].[udfMinimumDate] (

@x DATETIME,

@y DATETIME

) RETURNS DATETIME

AS

BEGIN

DECLARE @z DATETIME

IF @x <= @y

SET @z = @x

ELSE

SET @z = @y

RETURN(@z)

END;

GO

-- Converts the specified integer (which should be < 100 and > -1)

-- into a two character string, zero filling from the left

-- if the number is < 10.

CREATE FUNCTION [dbo].[udfTwoDigitZeroFill] (@number int)

RETURNS char(2)

AS

BEGIN

DECLARE @result char(2);

IF @number > 9

SET @result = convert(char(2), @number);

ELSE

SET @result = convert(char(2), '0' + convert(varchar, @number));

RETURN @result;

END;

GO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Create tables

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PRINT '';

PRINT '\*\*\* Creating Tables';

GO

CREATE TABLE [dbo].[User](

[uID] [int] IDENTITY(1,1) NOT NULL,

[user\_name] [nvarchar](20) NULL,

[admin] [bit] NULL,

) ON [PRIMARY];

GO

CREATE TABLE [dbo].[Image](

[iID] [int] IDENTITY(1,1) NOT NULL,

[uID] [int] IDENTITY(1,1) NOT NULL,

[file\_name] [nvarchar](20),

[size] [int] NOT NULL,

[uploaded\_date] [datetime] NOT NULL,

[type] [nvarchar](5) NOT NULL,

[trashed] [bit] NOT NULL,

[submitted] [bit] NOT NULL,

) ON [PRIMARY];

GO

CREATE TABLE [dbo].[Log](

[lID] [int] IDENTITY(1,1) NOT NULL,

) ON [PRIMARY];

GO

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Load data

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PRINT '';

PRINT '\*\*\* Loading Data';

GO

PRINT 'Loading [dbo].[AdventureWorksDWBuildVersion]';

INSERT INTO [dbo].[AdventureWorksDWBuildVersion]

VALUES

( CONVERT(nvarchar(50), SERVERPROPERTY('ProductVersion')), CONVERT(datetime, SERVERPROPERTY('ResourceLastUpdateDateTime')));

PRINT 'Loading [dbo].[DimAccount]';

BULK INSERT [dbo].[DimAccount] FROM '$(SqlSamplesSourceDataPath)DimAccount.csv'

WITH (

CHECK\_CONSTRAINTS,

-- CODEPAGE='ACP',

DATAFILETYPE = 'widechar',

FIELDTERMINATOR= '|',

ROWTERMINATOR = '\n',

KEEPIDENTITY,

TABLOCK

);

PRINT 'Loading [dbo].[DimCurrency]';

BULK INSERT [dbo].[DimCurrency] FROM '$(SqlSamplesSourceDataPath)DimCurrency.csv'

WITH (

CHECK\_CONSTRAINTS,

-- CODEPAGE='ACP',

DATAFILETYPE = 'widechar',

FIELDTERMINATOR= '|',

ROWTERMINATOR = '\n',

KEEPIDENTITY,

TABLOCK

);

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Add Primary Keys

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PRINT '';

PRINT '\*\*\* Adding Primary Keys';

GO

SET QUOTED\_IDENTIFIER ON;

ALTER TABLE [dbo].[sysdiagrams] WITH CHECK ADD

CONSTRAINT [PK\_\_sysdiagr\_\_C2B05B616B24EA82] PRIMARY KEY CLUSTERED

(

[diagram\_id]

) ON [PRIMARY];

GO

ALTER TABLE [dbo].[DimAccount] WITH CHECK ADD

CONSTRAINT [PK\_DimAccount] PRIMARY KEY CLUSTERED

(

[AccountKey]

) ON [PRIMARY];

GO

ALTER TABLE [dbo].[DimCurrency] WITH CHECK ADD

CONSTRAINT [PK\_DimCurrency\_CurrencyKey] PRIMARY KEY CLUSTERED

(

[CurrencyKey]

) ON [PRIMARY];

GO

ALTER TABLE [dbo].[DimCustomer] WITH CHECK ADD

CONSTRAINT [PK\_DimCustomer\_CustomerKey] PRIMARY KEY CLUSTERED

(

[CustomerKey]

) ON [PRIMARY];

GO

ALTER TABLE [dbo].[DimDate] WITH CHECK ADD

CONSTRAINT [PK\_DimDate\_DateKey] PRIMARY KEY CLUSTERED

(

[DateKey]

) ON [PRIMARY];

GO

ALTER TABLE [dbo].[DimDepartmentGroup] WITH CHECK ADD

CONSTRAINT [PK\_DimDepartmentGroup] PRIMARY KEY CLUSTERED

(

[DepartmentGroupKey]

) ON [PRIMARY];

GO

ALTER TABLE [dbo].[DimEmployee] WITH CHECK ADD

CONSTRAINT [PK\_DimEmployee\_EmployeeKey] PRIMARY KEY CLUSTERED

(

[EmployeeKey]

) ON [PRIMARY];

GO

ALTER TABLE [dbo].[DimGeography] WITH CHECK ADD

CONSTRAINT [PK\_DimGeography\_GeographyKey] PRIMARY KEY CLUSTERED

(

[GeographyKey]

) ON [PRIMARY];

GO

ALTER TABLE [dbo].[DimOrganization] WITH CHECK ADD

CONSTRAINT [PK\_DimOrganization] PRIMARY KEY CLUSTERED

(

[OrganizationKey]

) ON [PRIMARY];

GO

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Add Indexes

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PRINT '';

PRINT '\*\*\* Adding Indexes';

GO

CREATE UNIQUE NONCLUSTERED INDEX [AK\_DimCurrency\_CurrencyAlternateKey] ON [dbo].[DimCurrency]([CurrencyAlternateKey]) ON [PRIMARY];

GO

CREATE UNIQUE NONCLUSTERED INDEX [IX\_DimCustomer\_CustomerAlternateKey] ON [dbo].[DimCustomer]([CustomerAlternateKey]) ON [PRIMARY];

GO

CREATE UNIQUE NONCLUSTERED INDEX [AK\_DimDate\_FullDateAlternateKey] ON [dbo].[DimDate]( [FullDateAlternateKey]) ON [PRIMARY];

GO

/\*\*\*\*\*\* Object: Index [AK\_DimProduct\_ProductAlternateKey\_StartDate] Script Date: 7/8/2016 2:39:08 PM \*\*\*\*\*\*/

ALTER TABLE [dbo].[DimProduct] ADD CONSTRAINT [AK\_DimProduct\_ProductAlternateKey\_StartDate] UNIQUE NONCLUSTERED

(

[ProductAlternateKey] ASC,

[StartDate] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

GO

/\*\*\*\*\*\* Object: Index [AK\_DimProductCategory\_ProductCategoryAlternateKey] Script Date: 7/8/2016 2:40:13 PM \*\*\*\*\*\*/

ALTER TABLE [dbo].[DimProductCategory] ADD CONSTRAINT [AK\_DimProductCategory\_ProductCategoryAlternateKey] UNIQUE NONCLUSTERED

(

[ProductCategoryAlternateKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

GO

/\*\*\*\*\*\* Object: Index [AK\_DimProductSubcategory\_ProductSubcategoryAlternateKey] Script Date: 7/8/2016 2:40:37 PM \*\*\*\*\*\*/

ALTER TABLE [dbo].[DimProductSubcategory] ADD CONSTRAINT [AK\_DimProductSubcategory\_ProductSubcategoryAlternateKey] UNIQUE NONCLUSTERED

(

[ProductSubcategoryAlternateKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

GO

/\*\*\*\*\*\* Object: Index [AK\_DimPromotion\_PromotionAlternateKey] Script Date: 7/8/2016 2:41:46 PM \*\*\*\*\*\*/

ALTER TABLE [dbo].[DimPromotion] ADD CONSTRAINT [AK\_DimPromotion\_PromotionAlternateKey] UNIQUE NONCLUSTERED

(

[PromotionAlternateKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

GO

/\*\*\*\*\*\* Object: Index [AK\_DimReseller\_ResellerAlternateKey] Script Date: 7/8/2016 2:42:07 PM \*\*\*\*\*\*/

ALTER TABLE [dbo].[DimReseller] ADD CONSTRAINT [AK\_DimReseller\_ResellerAlternateKey] UNIQUE NONCLUSTERED

(

[ResellerAlternateKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

GO

/\*\*\*\*\*\* Object: Index [AK\_DimSalesTerritory\_SalesTerritoryAlternateKey] Script Date: 7/8/2016 2:43:13 PM \*\*\*\*\*\*/

ALTER TABLE [dbo].[DimSalesTerritory] ADD CONSTRAINT [AK\_DimSalesTerritory\_SalesTerritoryAlternateKey] UNIQUE NONCLUSTERED

(

[SalesTerritoryAlternateKey] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

GO

/\*\*\*\*\*\* Object: Index [AK\_FactCallCenter\_DateKey\_Shift] Script Date: 7/8/2016 2:43:49 PM \*\*\*\*\*\*/

ALTER TABLE [dbo].[FactCallCenter] ADD CONSTRAINT [AK\_FactCallCenter\_DateKey\_Shift] UNIQUE NONCLUSTERED

(

[DateKey] ASC,

[Shift] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

GO

CREATE UNIQUE NONCLUSTERED INDEX [UK\_principal\_name] ON [dbo].[sysdiagrams]([principal\_id], [name]) ON [PRIMARY];

GO

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Create Foreign key constraints

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PRINT '';

PRINT '\*\*\* Creating Foreign Key Constraints';

GO

ALTER TABLE [dbo].[DimAccount] ADD

CONSTRAINT [FK\_DimAccount\_DimAccount] FOREIGN KEY

(

[ParentAccountKey]

) REFERENCES [dbo].[DimAccount] ([AccountKey]);

GO

ALTER TABLE [dbo].[DimCustomer] ADD

CONSTRAINT [FK\_DimCustomer\_DimGeography] FOREIGN KEY

(

[GeographyKey]

)

REFERENCES [dbo].[DimGeography] ([GeographyKey])

ALTER TABLE [dbo].[DimDepartmentGroup] ADD

CONSTRAINT [FK\_DimDepartmentGroup\_DimDepartmentGroup] FOREIGN KEY

(

[ParentDepartmentGroupKey]

) REFERENCES [dbo].[DimDepartmentGroup] ([DepartmentGroupKey]);

GO

ALTER TABLE [dbo].[DimEmployee] ADD

CONSTRAINT [FK\_DimEmployee\_DimSalesTerritory] FOREIGN KEY

(

[SalesTerritoryKey]

) REFERENCES [dbo].[DimSalesTerritory] ([SalesTerritoryKey]),

CONSTRAINT [FK\_DimEmployee\_DimEmployee] FOREIGN KEY

(

[ParentEmployeeKey]

) REFERENCES [dbo].[DimEmployee] ([EmployeeKey]);

GO

ALTER TABLE [dbo].[DimGeography] ADD

CONSTRAINT [FK\_DimGeography\_DimSalesTerritory] FOREIGN KEY

(

[SalesTerritoryKey]

) REFERENCES [dbo].[DimSalesTerritory] ([SalesTerritoryKey]);

GO

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Add database views.

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PRINT '';

PRINT '\*\*\* Creating Table Views';

GO

-- vDMPrep will be used as a data source by the other data mining views.

-- Uses DW data at customer, product, day, etc. granularity and

-- gets region, model, year, month, etc.

CREATE VIEW [dbo].[vDMPrep]

AS

SELECT

pc.[EnglishProductCategoryName]

,Coalesce(p.[ModelName], p.[EnglishProductName]) AS [Model]

,c.[CustomerKey]

,s.[SalesTerritoryGroup] AS [Region]

,CASE

WHEN Month(GetDate()) < Month(c.[BirthDate])

THEN DateDiff(yy,c.[BirthDate],GetDate()) - 1

WHEN Month(GetDate()) = Month(c.[BirthDate])

AND Day(GetDate()) < Day(c.[BirthDate])

THEN DateDiff(yy,c.[BirthDate],GetDate()) - 1

ELSE DateDiff(yy,c.[BirthDate],GetDate())

END AS [Age]

,CASE

WHEN c.[YearlyIncome] < 40000 THEN 'Low'

WHEN c.[YearlyIncome] > 60000 THEN 'High'

ELSE 'Moderate'

END AS [IncomeGroup]

,d.[CalendarYear]

,d.[FiscalYear]

,d.[MonthNumberOfYear] AS [Month]

,f.[SalesOrderNumber] AS [OrderNumber]

,f.SalesOrderLineNumber AS LineNumber

,f.OrderQuantity AS Quantity

,f.ExtendedAmount AS Amount

FROM

[dbo].[FactInternetSales] f

INNER JOIN [dbo].[DimDate] d

ON f.[OrderDateKey] = d.[DateKey]

INNER JOIN [dbo].[DimProduct] p

ON f.[ProductKey] = p.[ProductKey]

INNER JOIN [dbo].[DimProductSubcategory] psc

ON p.[ProductSubcategoryKey] = psc.[ProductSubcategoryKey]

INNER JOIN [dbo].[DimProductCategory] pc

ON psc.[ProductCategoryKey] = pc.[ProductCategoryKey]

INNER JOIN [dbo].[DimCustomer] c

ON f.[CustomerKey] = c.[CustomerKey]

INNER JOIN [dbo].[DimGeography] g

ON c.[GeographyKey] = g.[GeographyKey]

INNER JOIN [dbo].[DimSalesTerritory] s

ON g.[SalesTerritoryKey] = s.[SalesTerritoryKey]

;

GO

/\*\*\*\*\*\* Object: View [dbo].[vTimeSeries] Script Date: 7/8/2016 3:09:56 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- vTimeSeries view supports the creation of time series data mining models.

-- - Replaces earlier bike models with successor models.

-- - Abbreviates model names to improve readability in mining model viewer

-- - Concatenates model and region so that table only has one input.

-- - Creates a date field indexed to monthly reporting date for use in prediction.

CREATE VIEW [dbo].[vTimeSeries]

AS

SELECT

CASE [Model]

WHEN 'Mountain-100' THEN 'M200'

WHEN 'Road-150' THEN 'R250'

WHEN 'Road-650' THEN 'R750'

WHEN 'Touring-1000' THEN 'T1000'

ELSE Left([Model], 1) + Right([Model], 3)

END + ' ' + [Region] AS [ModelRegion]

,(Convert(Integer, [CalendarYear]) \* 100) + Convert(Integer, [Month]) AS [TimeIndex]

,Sum([Quantity]) AS [Quantity]

,Sum([Amount]) AS [Amount]

,CalendarYear

,[Month]

,[dbo].[udfBuildISO8601Date] ([CalendarYear], [Month], 25)

as ReportingDate

FROM

[dbo].[vDMPrep]

WHERE

[Model] IN ('Mountain-100', 'Mountain-200', 'Road-150', 'Road-250',

'Road-650', 'Road-750', 'Touring-1000')

GROUP BY

CASE [Model]

WHEN 'Mountain-100' THEN 'M200'

WHEN 'Road-150' THEN 'R250'

WHEN 'Road-650' THEN 'R750'

WHEN 'Touring-1000' THEN 'T1000'

ELSE Left(Model,1) + Right(Model,3)

END + ' ' + [Region]

,(Convert(Integer, [CalendarYear]) \* 100) + Convert(Integer, [Month])

,CalendarYear

,[Month]

,[dbo].[udfBuildISO8601Date] ([CalendarYear], [Month], 25);

GO

/\*\*\*\*\*\* Object: View [dbo].[vTargetMail] Script Date: 7/8/2016 3:09:56 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- vTargetMail supports targeted mailing data model

-- Uses vDMPrep to determine if a customer buys a bike and joins to DimCustomer

CREATE VIEW [dbo].[vTargetMail]

AS

SELECT

c.[CustomerKey],

c.[GeographyKey],

c.[CustomerAlternateKey],

c.[Title],

c.[FirstName],

c.[MiddleName],

c.[LastName],

c.[NameStyle],

c.[BirthDate],

c.[MaritalStatus],

c.[Suffix],

c.[Gender],

c.[EmailAddress],

c.[YearlyIncome],

c.[TotalChildren],

c.[NumberChildrenAtHome],

c.[EnglishEducation],

c.[SpanishEducation],

c.[FrenchEducation],

c.[EnglishOccupation],

c.[SpanishOccupation],

c.[FrenchOccupation],

c.[HouseOwnerFlag],

c.[NumberCarsOwned],

c.[AddressLine1],

c.[AddressLine2],

c.[Phone],

c.[DateFirstPurchase],

c.[CommuteDistance],

x.[Region],

x.[Age],

CASE x.[Bikes]

WHEN 0 THEN 0

ELSE 1

END AS [BikeBuyer]

FROM

[dbo].[DimCustomer] c INNER JOIN (

SELECT

[CustomerKey]

,[Region]

,[Age]

,Sum(

CASE [EnglishProductCategoryName]

WHEN 'Bikes' THEN 1

ELSE 0

END) AS [Bikes]

FROM

[dbo].[vDMPrep]

GROUP BY

[CustomerKey]

,[Region]

,[Age]

) AS [x]

ON c.[CustomerKey] = x.[CustomerKey]

;

GO

/\*\*\*\*\*\* Object: View [dbo].[vAssocSeqOrders] Script Date: 7/8/2016 3:09:56 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

/\* vAssocSeqOrders supports assocation and sequence clustering data mmining models.

- Limits data to FY2004.

- Creates order case table and line item nested table.\*/

CREATE VIEW [dbo].[vAssocSeqOrders]

AS

SELECT DISTINCT OrderNumber, CustomerKey, Region, IncomeGroup

FROM dbo.vDMPrep

WHERE (FiscalYear = '2013')

GO

/\*\*\*\*\*\* Object: View [dbo].[vAssocSeqLineItems] Script Date: 7/8/2016 3:09:56 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[vAssocSeqLineItems]

AS

SELECT OrderNumber, LineNumber, Model

FROM dbo.vDMPrep

WHERE (FiscalYear = '2013')

GO

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Drop DDL Trigger for Database

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PRINT '';

PRINT '\*\*\* Disabling DDL Trigger for Database';

GO

DISABLE TRIGGER [ddlDatabaseTriggerLog]

ON DATABASE;

GO

USE [master];

GO

PRINT 'Finished - ' + CONVERT(varchar, GETDATE(), 121);

GO

SET NOEXEC OFF