Microsoft

Exploring SQL Server Data Tools in Visual Studio 2013

Contents

Azure account required for last exercise

Estimated time to complete lab is 60 minutes

To perform the last exercise of this lab where you will publish a database project to a Microsoft Azure SQL Database, you will require a Microsoft Azure account.

If you do not have an Azure account, you can request a free trial version by going to http://azure.microsoft.com/en-us/pricing/free-trial/.

Within the one-month trial version, you can perform other SQL Server 2014 hands on labs along with other tutorials available on Azure.

Note, to sign up for a free trial, you will need a mobile device that can receive text messages and a valid credit card.

You can still perform this hands on lab without an Azure account with the exception of the last exercise.

Be sure to follow the Roll Back Azure changes section at the end of this exercise after creating the Azure database so that you can make the most use of your $200 free Azure credit.

Optimized productivity–One set of tools for everything

As organizations are focused on becoming more effective, doing more with the same amount or less resource and striving to become more agile in deploying and improving applications, it becomes important to manage the skills and future skills requirements of your staff or to provision new tools. Unfortunately both these options can require a significant amount of continuing investment, as well as a delay in realizing that investment as people get up to speed and practiced in deploying the tools they have been given or the skills they have been taught. Improvements in the integration between the traditional SQL Server Management Studio capabilities and Visual Studio mean that organizations can now work with familiar toolsets and environments but allow DBAs to reduce the time required to modify, test and publish solutions within or across different versions of SQL server, as well as improve on best practice by storing all the database components in source safe allowing for version control and auditing. This means organizations can achieve a significant increase in efficiency without having to invest in tools or training and minimize their time to their realization of this efficiency gain.

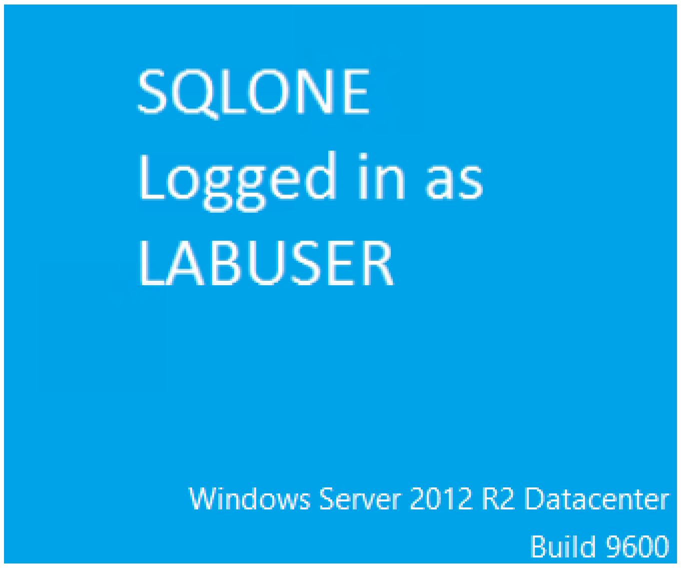
Using SSIS project to export a table to a flat file

The new release of the SQL Server Data Tools to support SQL Server 2014 enables you to support both database and business intelligence projects. From the BI standpoint, this will provide you with a tool to manage SSAS and SSRS project for SQL Server versions 2014 and lower and SSIS for SQL 2014 only. When looking at databases, the full integration with the Visual Studio SKUs allows you to provide end-to-end support for building and managing Windows Azure SQL Databases and SQL Server databases. In this lab, you will run and explore this tool.

NOTE: SQL Server Data Tools has been split into two products – SSDT and SSDTBI. SSDT is for working with database projects and is a stand-alone tool if Visual Studio is not installed. Comparatively, SSDTBI is for business intelligence project (SSAS, SSIS, SSRS.) Both are installed pn the lab environment,, so now can work with database project, BI projects, and development projects from the one interface of Visual Studio 2013. Here, you will be exploring the BI capabilities, and will use the database capabilities in the next scenario.

Connect to SQLONE computer

1. Click on **SQLONE** button on right side of the screen to connect to the **SQLONE** computer. If you see the following in the lower right corner of the screen, you can jump to step 5 below to set your screen resolution.


SQLONE
Logged in as
LABUSER

Windows Server 2012 R2 Datacenter
Build 9600

2. Click **Send Ctrl-Alt-Del** for **SQLONE** computer and then click **Switch user**.

3. Click **Send Ctrl-Alt-Del** for **SQLONE** computer again and then click **Other user**.

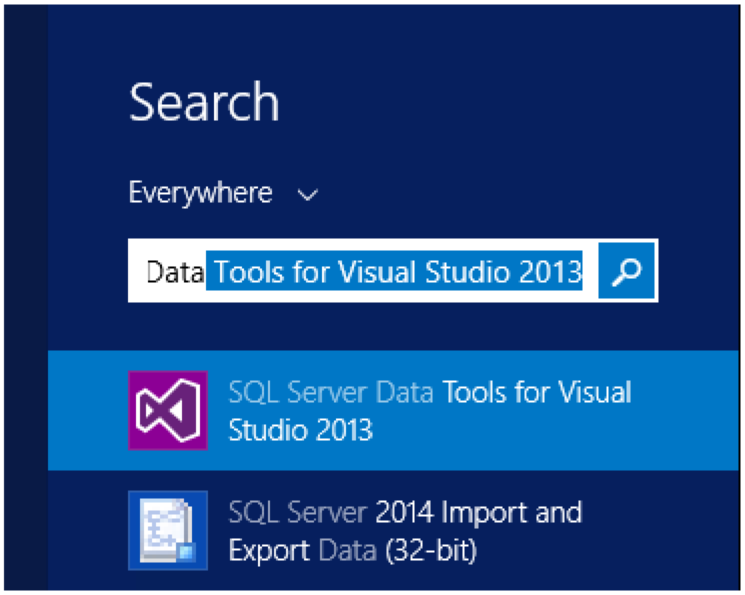
4. Log on to **SQLONE** computer as **labuser** with password **pass@word1**

Note, if you have a monitor that supports a larger screen resolution than 1024 x 768, you can change the screen resolution for the lab to go as high as 1920 x 1080. By going to a higher screen resolution, it will be easier to use SQL Server Management Studio.

5. Right click on the desktop and click on **Screen resolution**.
6. Select **1366 x 786** (a good minimum screen size for using SSMS) and click **OK**.
7. Click **Keep Changes**.
8. Resize the client **holLaunchPad Online** window for the lab to fit your screen resolution.

Create an SSIS project

1. Open **SQL Server Data Tools for Visual Studio 2013** from the start screen


Search
Everywhere
Data Tools for Visual Studio 2013
SQL Server Data Tools for Visual Studio 2013
SQL Server 2014 Import and Export Data (32-bit)

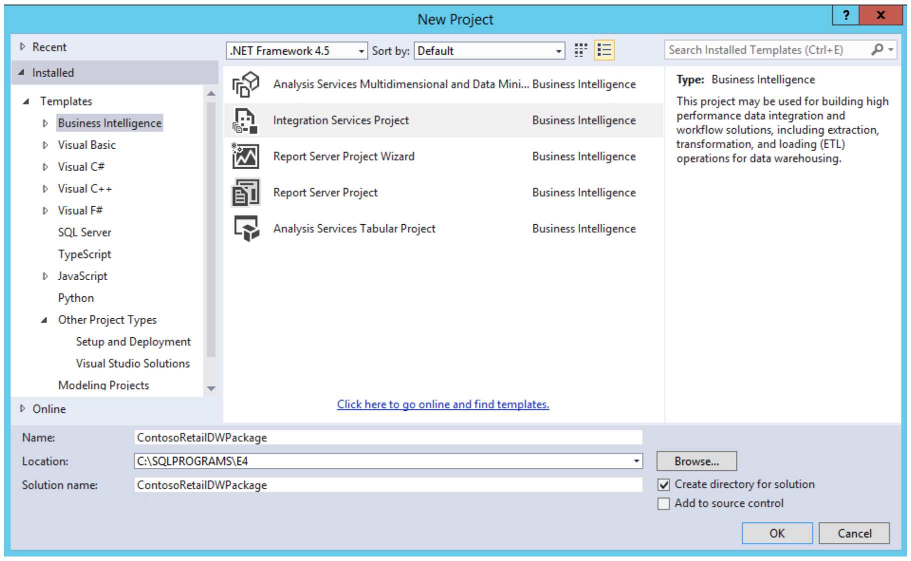
2. On the **File** menu, click **New** > **Project**
3. In the **New Project** wizard, select the **Business Intelligence** template type (in the left hand pane) then select an **Integration Services Project** (in the center pane.)
4. Enter **ContosoRetailDWPackag**e as the name for the project

Note how the solution name automatically changes as you change the solution name

5. Select the location of the project as **C:\SQLPROGRAMS\E4**

6. Ensure **Create a directory for the solution** is checked and **Add to source control** is not

You will be exploring source control later in this story


New Project
Recent
Installed
Templates
Business Intelligence
Visual Basic
Visual C#
Visual C++
Visual F#
SQL Server
TypeScript
JavaScript
Python
Other Project Types
Setup and Deployment
Visual Studio Solutions
Modeling Projects
Online
.NET Framework 4.5
Sort by: Default
Search Installed Templates (Ctrl+E)
Analysis Services Multidimensional and Data Mini... Business Intelligence
Integration Services Project Business Intelligence
Report Server Project Wizard Business Intelligence
Report Server Project Business Intelligence
Analysis Services Tabular Project Business Intelligence
Click here to go online and find templates.
Browse...
OK
Cancel

Type: Business Intelligence

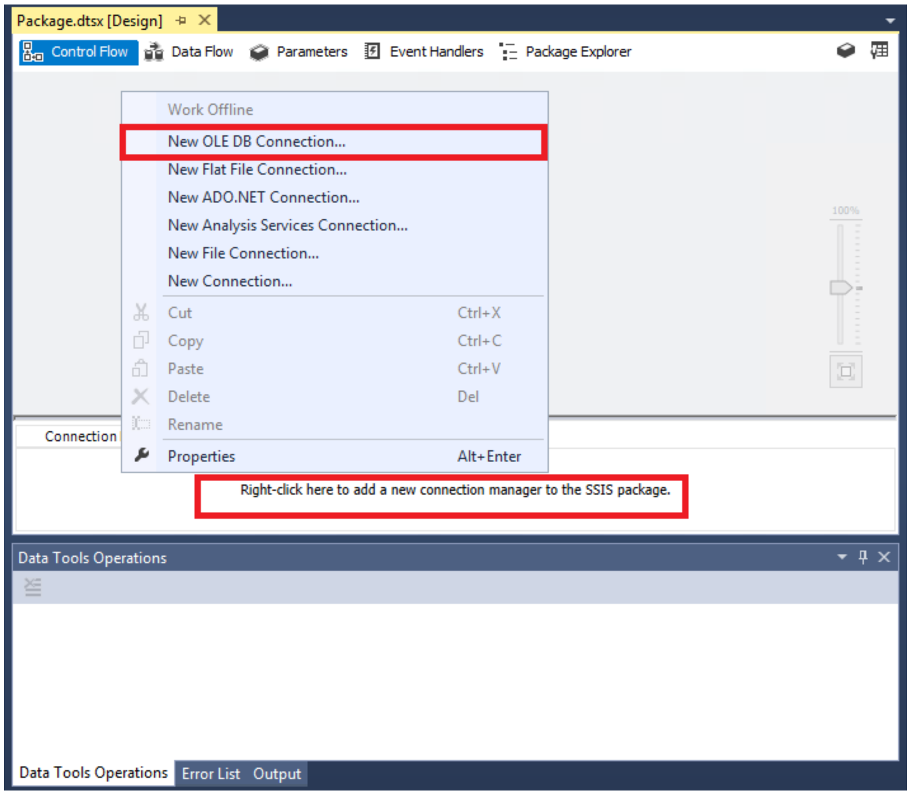
This project may be used for building high performance data integration and workflow solutions, including extraction, transformation, and loading (ETL) operations for data warehousing.

Name: ContosoRetailDWPackage
Location: C:\SQLPROGRAMS\E4
Solution name: ContosoRetailDWPackage
[✓] Create directory for solution
[ ] Add to source control

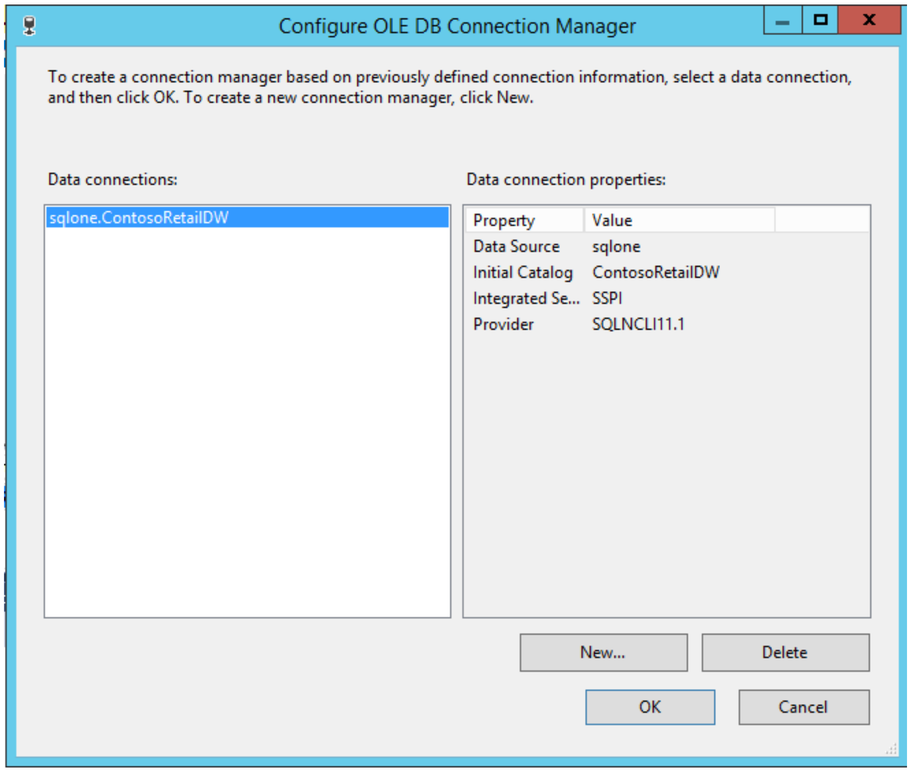
7. Click **OK** to create the project

Create an SSIS connection to ControlsRetailDW

1. Right-click in the **Connection Managers** pane at the window bottom and select **New OLE DB Connection…**



2. In the popup box, click **New**...



To create a connection manager based on previously defined connection information, select a data connection, and then click OK. To create a new connection manager, click New.

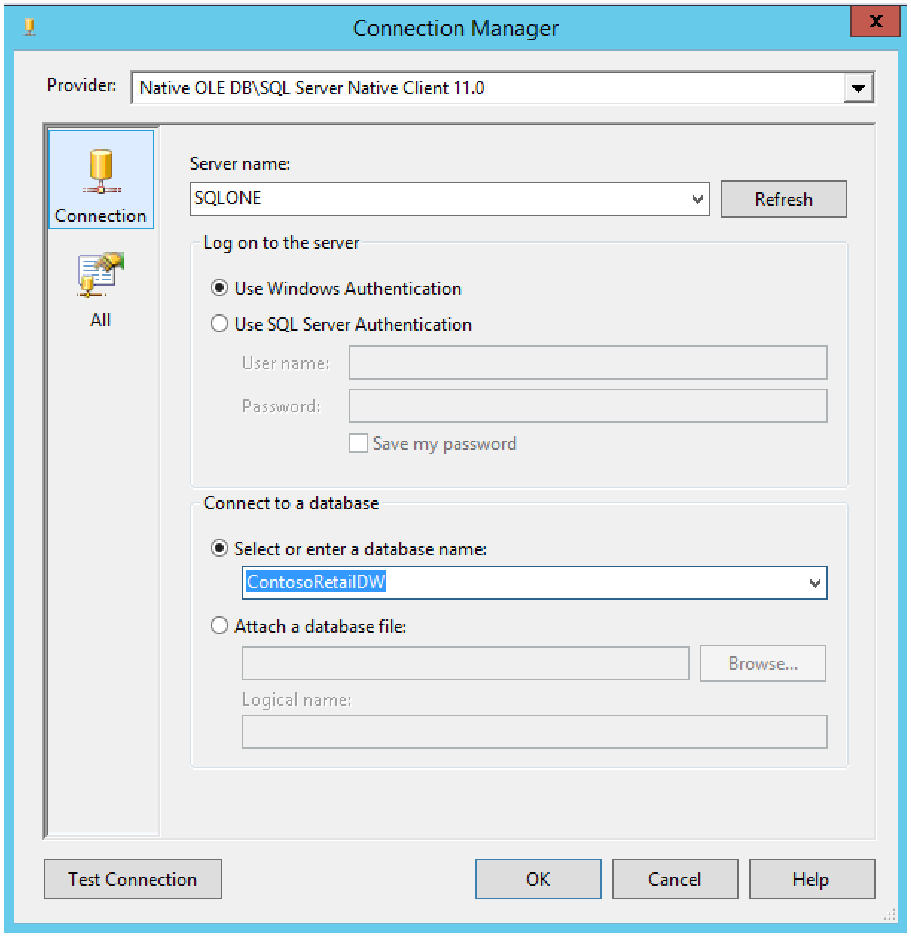
Data connections:

sqlone.ContosoRetailDW

Data connection properties:

Property	Value
Data Source	sqlone
Initial Catalog	ContosoRetailDW
Integrated Se...	SSPI
Provider	SQLNCLI11.1

3. Enter **SQLONE** as the Server name, select **Use Windows Authentication**, and select **ContosoRetailDW** as the database



Provider: Native OLE DB\SQL Server Native Client 11.0

Server name: SQLONE

Log on to the server: [✓ Use Windows Authentication] Use SQL Server Authentication

User name:

Password:

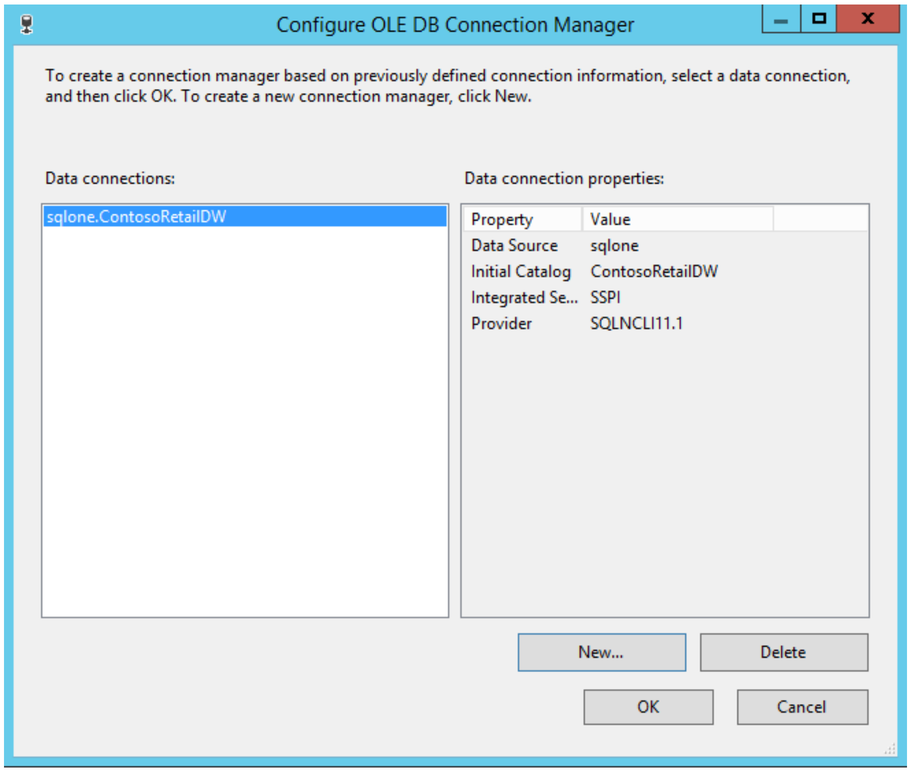
Save my password

Connect to a database: [✓ Select or enter a database name:] ContosoRetailDW

Attach a database file:

Logical name:

4. Click **OK** to create the data source, then **OK** to create the connection manager



To create a connection manager based on previously defined connection information, select a data connection, and then click OK. To create a new connection manager, click New.

Data connections:

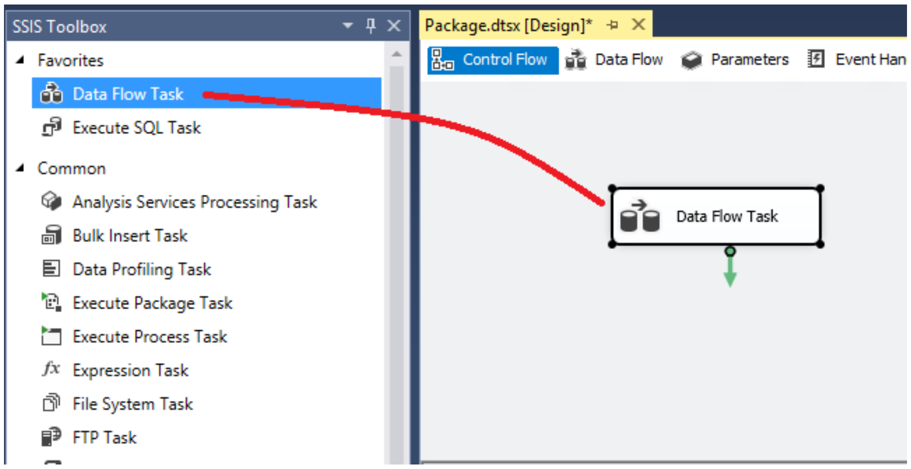
sqlone.ContosoRetailDW

Data connection properties:

Property	Value
Data Source	sqlone
Initial Catalog	ContosoRetailDW
Integrated Se...	SSPI
Provider	SQLNCLI11.1

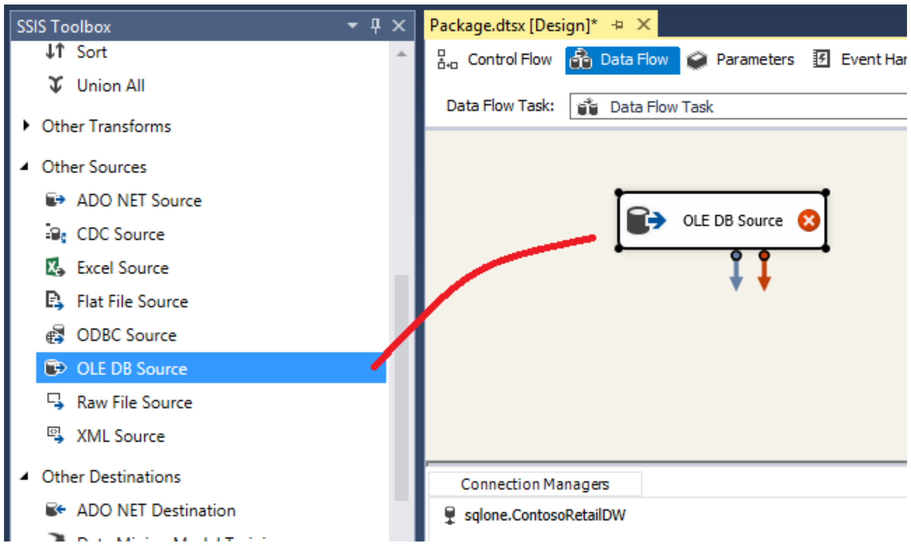
Add a data flow task

1. Drag a **Data Flow Task** control from the **SSIS Toolbox** onto the central pane

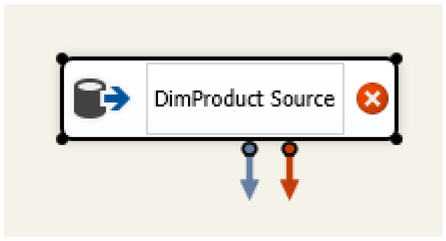


2. Double-click on the newly-created **Data Flow Task** in the central pane. This opens the Data Flow task

3. Drag an **OLE DB Source** control from the **SSIS Toolbox** (under **Other Sources**) onto the central pane



4. Click once on the text **OLE DB Source** of the new control, type **DimProduct Source** and press ENTER to change the control name

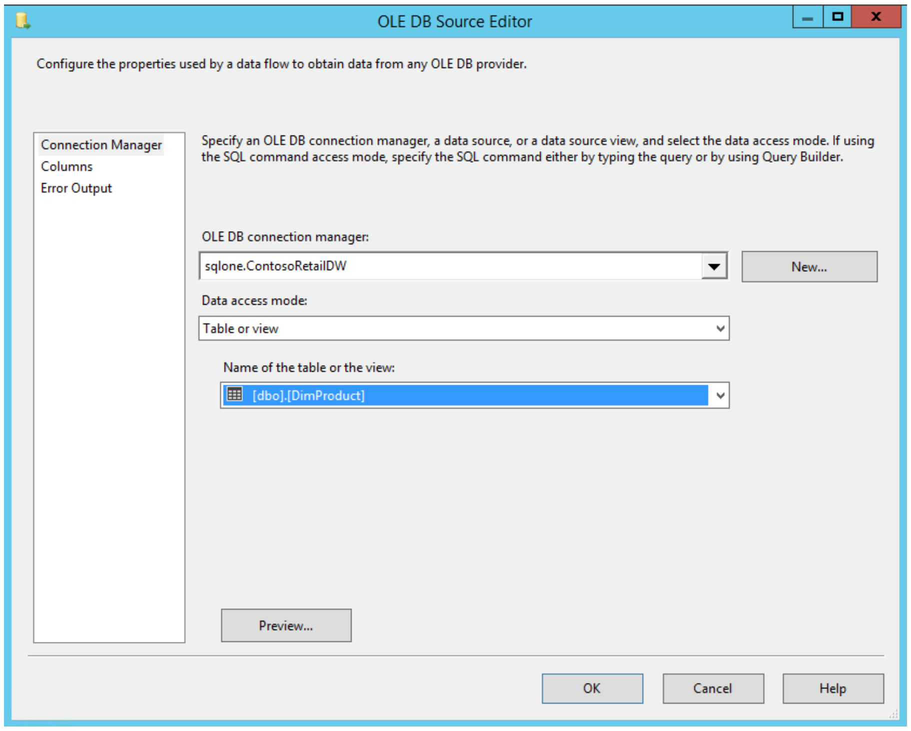


Select the table to use as the source

1. Double-click on the **DimProduct Source** control to open its editor

2. The OLE DB connection manager property will automatically be assigned to the SQLONE connection manager you created earlier

3. Select **DimProduct** from the **Name of the table or the view** dropdown box, then click **OK**.


OLE DB Source Editor

Configure the properties used by a data flow to obtain data from any OLE DB provider.

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder.

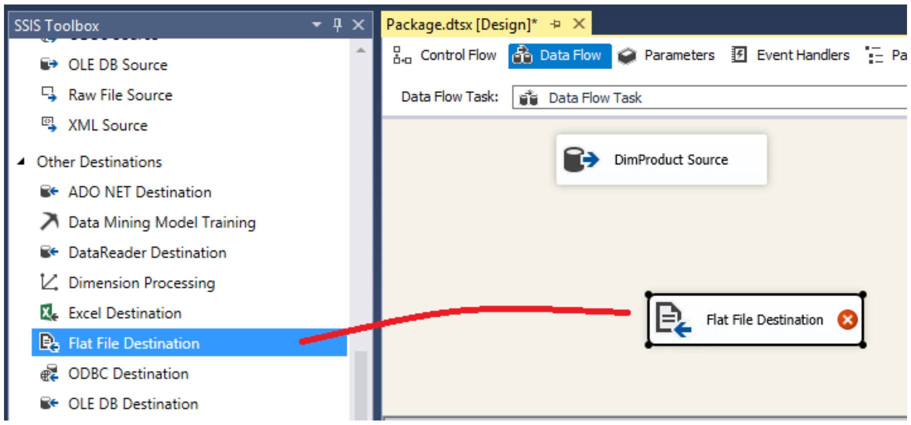
OLE DB connection manager: sqlone.ContosoRetailDW

Data access mode: Table or view

Name of the table or the view: [dbo].[DimProduct]

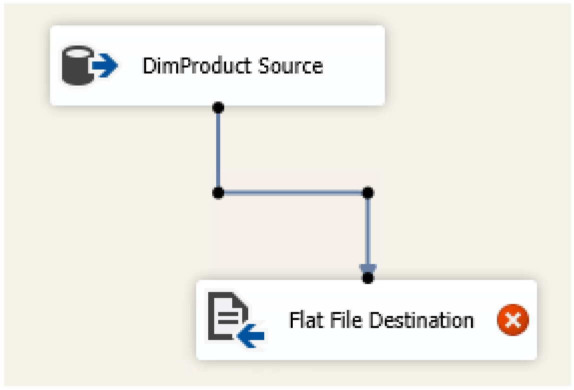
Add a destination and connect it to the data source

1. Drag a **Flat File Destination** control from the **SSIS Toolbox** (under **Other Destinations**) onto the central pane



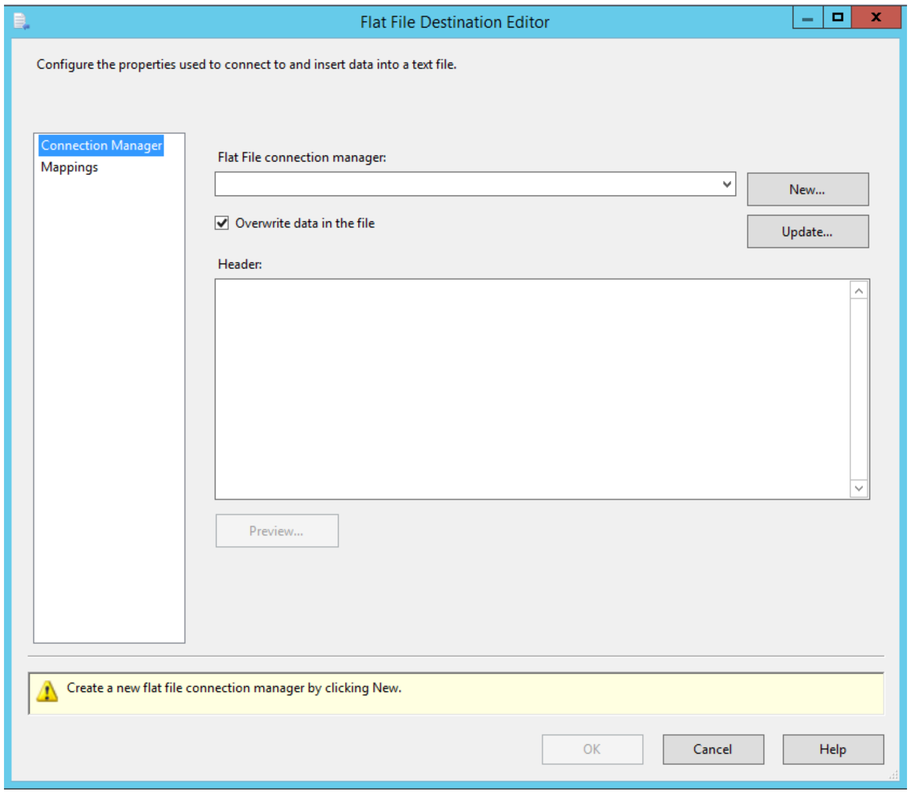
2. Click on **DimProduct Source** again

3. Once the blue arrow coming out the bottom of the **DimProduct Source** is visible, drag it so the bottom end is on top of the **Flat File Destination** control



4. Double click on the **Flat File Destination** control to open its editor

5. In the editor, click **New…** next to the **Flat File connection manager** dropdown box

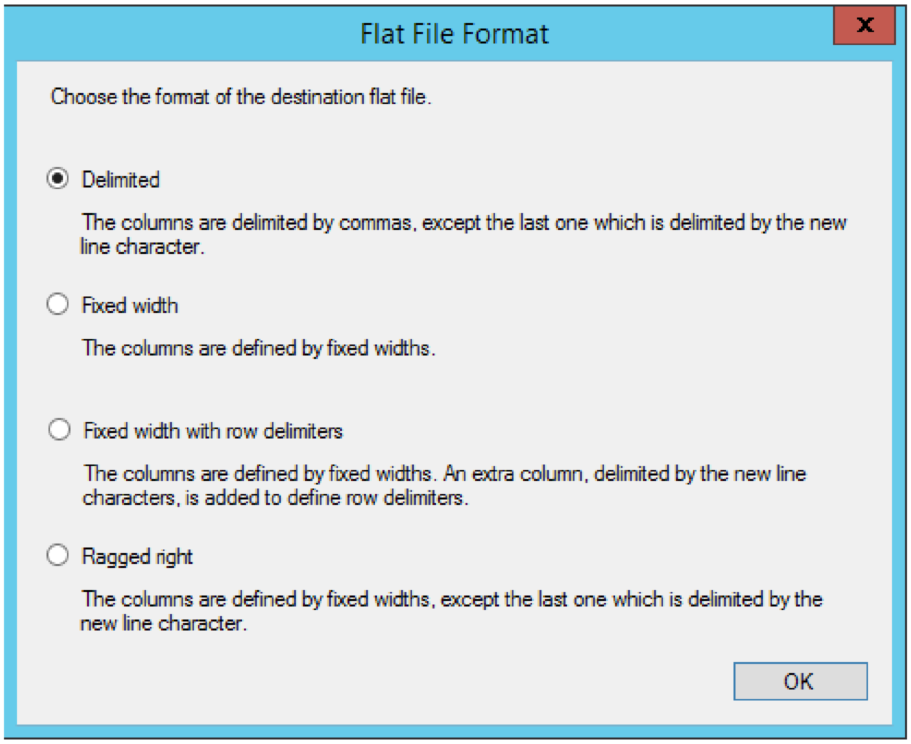


Configure the properties used to connect to and insert data into a text file.

☑ Overwrite data in the file

Create a new flat file connection manager by clicking New.

6. Select **Delimited** as the Flat File Format and click **OK**



Choose the format of the destination flat file.

(●) Delimited

The columns are delimited by commas, except the last one which is delimited by the new line character.

( ) Fixed width

The columns are defined by fixed widths.

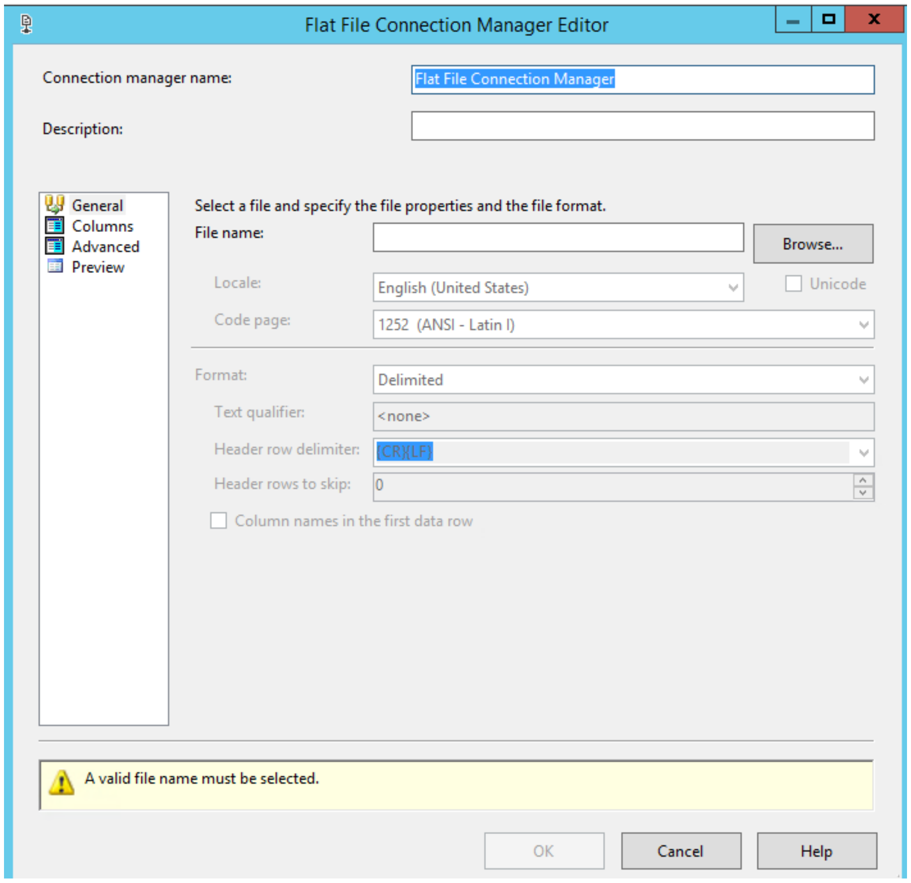
( ) Fixed width with row delimiters

The columns are defined by fixed widths. An extra column, delimited by the new line characters, is added to define row delimiters.

( ) Ragged right

The columns are defined by fixed widths, except the last one which is delimited by the new line character.

7. Click on **Browse** next to the **File name** box to select a file.



Connection manager name: Flat File Connection Manager

Description:

Select a file and specify the file properties and the file format.

File name:

Locale: English (United States)

Unicode

Code page: 1252 (ANSI - Latin I)

Format: Delimited

Text qualifier: <none>

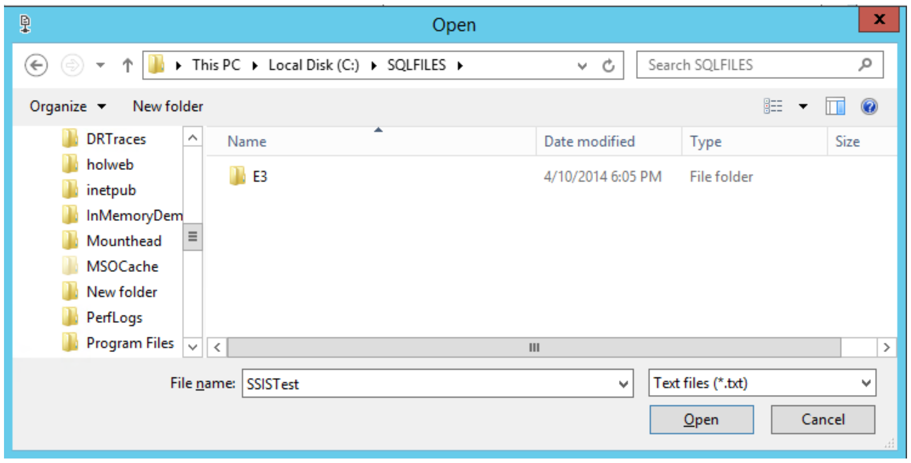
Header row delimiter: {CR}{LF}

Header rows to skip: 0

Column names in the first data row

A valid file name must be selected.

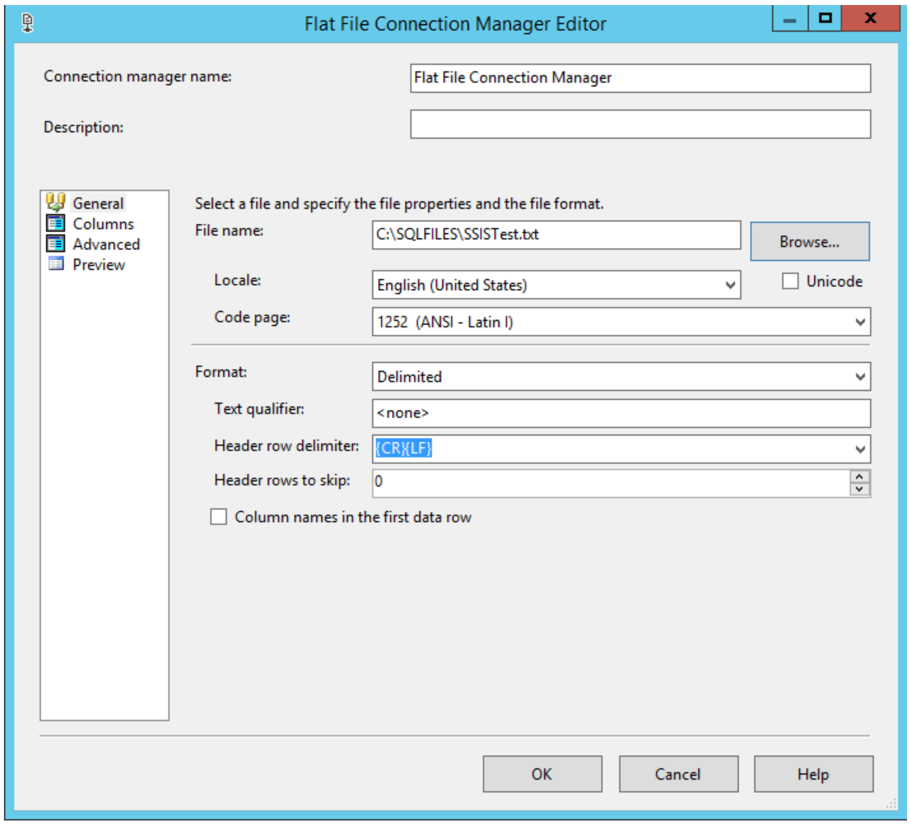
8. In the **Open** dialog, navigate to **C:\SQLFILES**, enter **SSISTest** as the file name, and then click **Open** to choose the new file.



Name	Date modified	Type	Size
E3	4/10/2014 6:05 PM	File folder	

File name: SSISTest

Text files (*.txt)



Connection manager name: Flat File Connection Manager

Description:

Select a file and specify the file properties and the file format.

File name: C:\SQLFILES\SSISTest.txt

Locale: English (United States)

Unicode

Code page: 1252 (ANSI - Latin I)

Format: Delimited

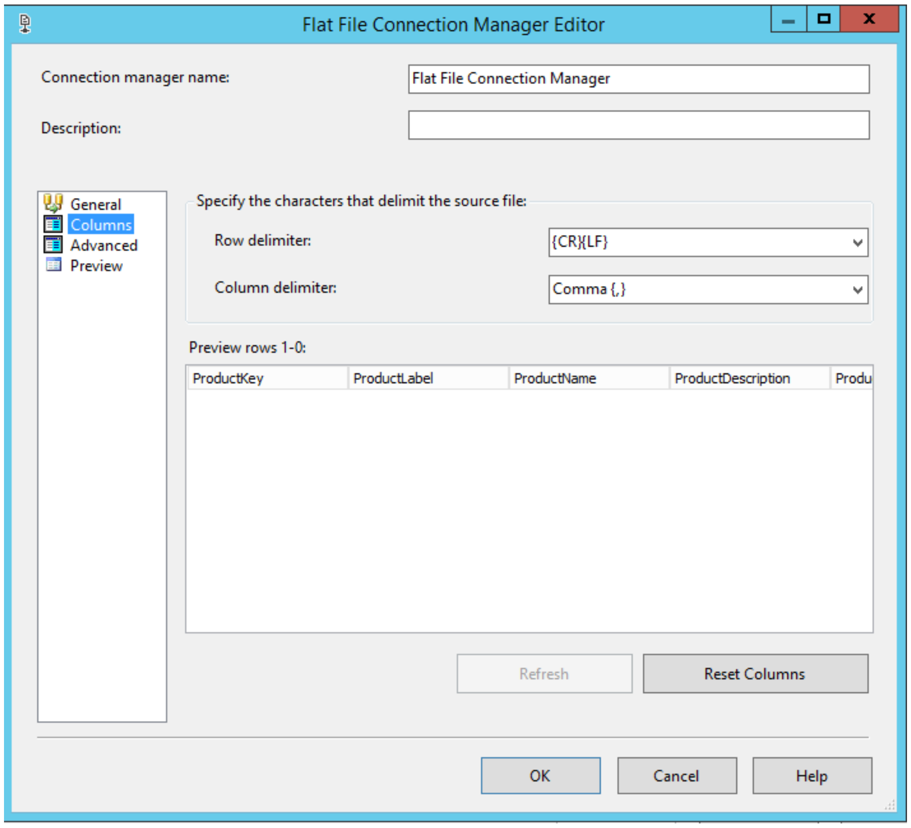
Text qualifier: <none>

Header row delimiter: {CR}{LF}

Header rows to skip: 0

Column names in the first data row

9. Look at **Columns** tab in the left to see what columns the new flat file will have


Flat File Connection Manager Editor

Connection manager name: Flat File Connection Manager

Description:

Specify the characters that delimit the source file:

Row delimiter: {CR}{LF}

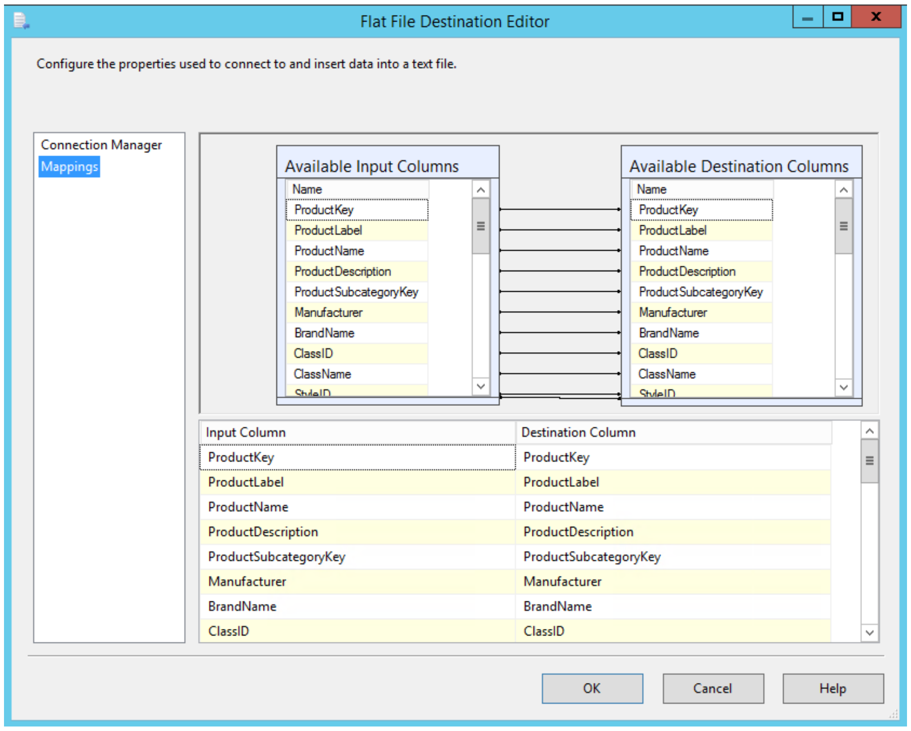
Column delimiter: Comma {,}

Preview rows 1-0:

ProductKey	ProductLabel	ProductName	ProductDescription	Produ

10. Click **OK** to create the connection manager

11. Go to the **Mappings** page of the **Flat File Destination Editor** window, and note that all columns in the **Available Input Columns** box have a line going from them to a column of the same name in the **Available Destinations columns**


Flat File Destination Editor

Configure the properties used to connect to and insert data into a text file.

Connection Manager
Mappings

Available Input Columns: Name, ProductKey, ProductLabel, ProductName, ProductDescription, ProductSubcategoryKey, Manufacturer, BrandName, ClassID, ClassName, StyleID

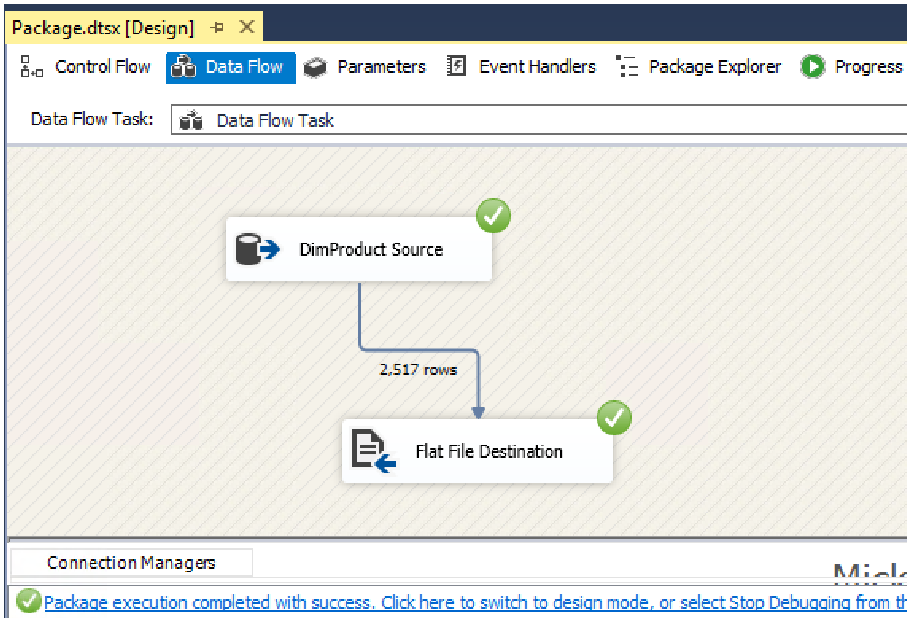
Available Destination Columns: Name, ProductKey, ProductLabel, ProductName, ProductDescription, ProductSubcategoryKey, Manufacturer, BrandName, ClassID, ClassName, StyleID

Input Column	Destination Column
ProductKey	ProductKey
ProductLabel	ProductLabel
ProductName	ProductName
ProductDescription	ProductDescription
ProductSubcategoryKey	ProductSubcategoryKey
Manufacturer	Manufacturer
BrandName	BrandName
ClassID	ClassID

12. Click **OK**

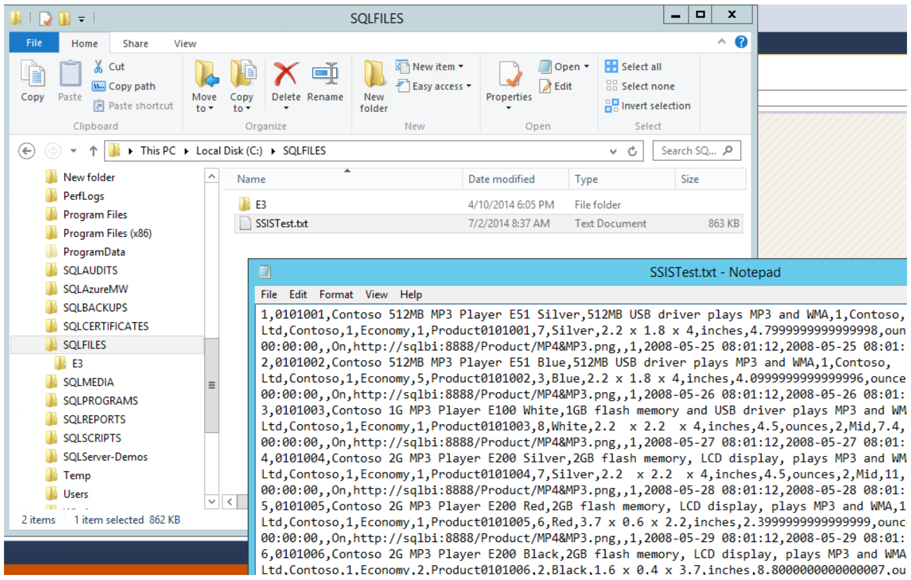
Run the package in debug mode

1. Click ▶ **Start** icon in the command bar at the window top
2. Watch the green tick appear on **DimProduct Source** and then on **Flat File Destination** showing the source and destination were both successful



Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from th

3. Once the package execution has completed with success, open **File Explorer**, navigate to **C:\SQLFILES**, and open **SSISTest.txt.** You should see all the data from **DimProduct** in this text file.



Name	Date modified	Type	Size
E3	4/10/2014 6:05 PM	File folder	
SSISTest.txt	7/2/2014 8:37 AM	Text Document	863 KB

```
1,0101001,Contoso 512MB MP3 Player E51 Silver,512MB USB driver plays MP3 and WMA,1,Contoso,
Ltd,Contoso,1,Economy,1,Product0101001,7,Silver,2.2 x 1.8 x 4,inches,4.7999999999999998,oun
00:00:00,,On,http://sqlbi:8888/Product/MP4&MP3.png,,1,2008-05-25 08:01:12,2008-05-25 08:01:
2,0101002,Contoso 512MB MP3 Player E51 Blue,512MB USB driver plays MP3 and WMA,1,Contoso,
Ltd,Contoso,1,Economy,5,Product0101002,3,Blue,2.2 x 1.8 x 4,inches,4.0999999999999996,ounce
00:00:00,,On,http://sqlbi:8888/Product/MP4&MP3.png,,1,2008-05-26 08:01:12,2008-05-26 08:01:
3,0101003,Contoso 1G MP3 Player E100 White,1GB flash memory and USB driver plays MP3 and WM
Ltd,Contoso,1,Economy,1,Product0101003,8,White,2.2  x 2.2  x 4,inches,4.5,ounces,2,Mid,7.4,
00:00:00,,On,http://sqlbi:8888/Product/MP4&MP3.png,,1,2008-05-27 08:01:12,2008-05-27 08:01:
4,0101004,Contoso 2G MP3 Player E200 Silver,2GB flash memory, LCD display, plays MP3 and WM
Ltd,Contoso,1,Economy,1,Product0101004,7,Silver,2.2  x 2.2  x 4,inches,4.5,ounces,2,Mid,11,
00:00:00,,On,http://sqlbi:8888/Product/MP4&MP3.png,,1,2008-05-28 08:01:12,2008-05-28 08:01:
5,0101005,Contoso 2G MP3 Player E200 Red,2GB flash memory, LCD display, plays MP3 and WMA,1
Ltd,Contoso,1,Economy,1,Product0101005,6,Red,3.7 x 0.6 x 2.2,inches,2.3999999999999999,ounc
00:00:00,,On,http://sqlbi:8888/Product/MP4&MP3.png,,1,2008-05-29 08:01:12,2008-05-29 08:01:
6,0101006,Contoso 2G MP3 Player E200 Black,2GB flash memory, LCD display, plays MP3 and WMA
Ltd,Contoso,1,Economy,2,Product0101006,2,Black,1.6 x 0.4 x 3.7,inches,8.8000000000000007,ou
```

4. Click on the stop icon ■ in the command bar at the window top

5. Close **SQL Server Development Tools for Visual Studio 2013** after saving the project

You just learned how to create an SSIS project from within Visual Studio 2013 to copy data from one of his database tables into a text file.

Managing and developing schemas

Visual Studio 2013 provides an integrated toolset for utilizing BI and database capability. It allows user to reverse engineer database solution from actual database and provides the ability to maintain and manage various versions of the database

In this exercise, you will use SQL Server Development Tools for Visual Studio 2013 to explore the tools to ensure that you can access, configure and run the components that relate to DBA development tasks. You will begin by looking into managing and developing schemas.

SQL Server Development Tools for Visual Studio 2013 provides a rich development environment outside of SQL Server Management studio that allows you to take advantage of a number of features that could improve the performance of the DBA team when making changes to schemas and tables. The capabilities offered by SQL Server Data Tools will allow you to do this.

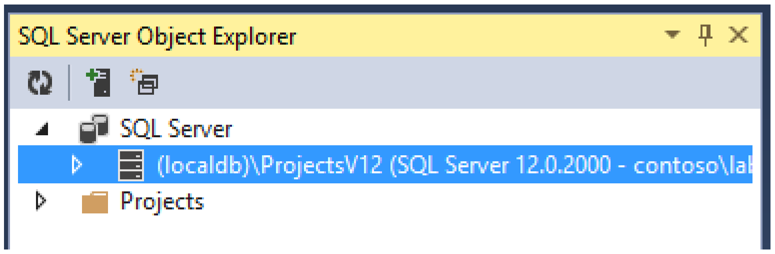
Initial Cleanup
In Windows Explorer navigate to **C:\SQLPROGRAMS\E4.**

Delete the **ContosoRetailProj** folder.

Connect to a SQL 2014 Database from Visual Studio 2013

1. Open **SQL Server Development Tools for Visual Studio 2013** from **Windows Start**

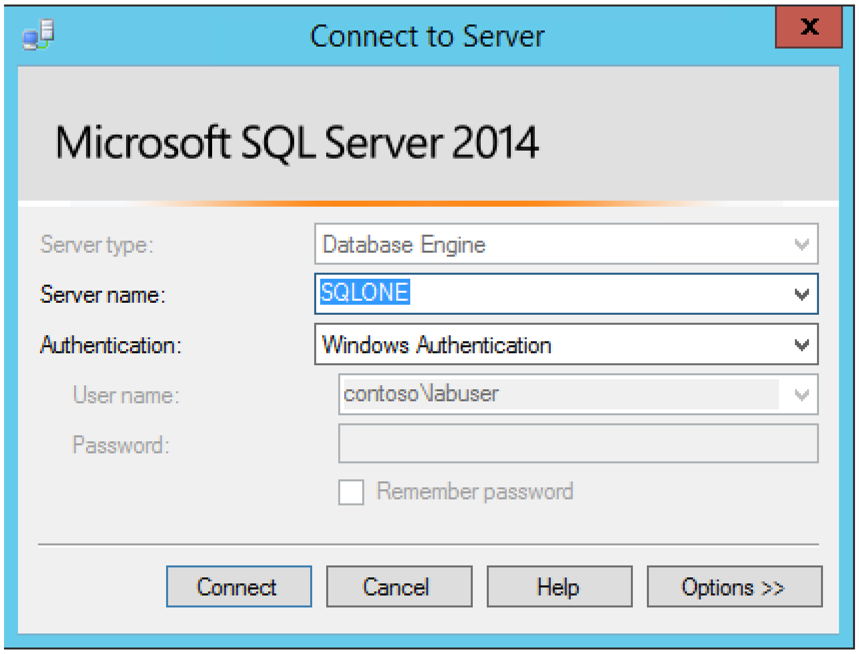
2. Select **SQL Server Object Explorer** from **View** menu


SQL Server Object Explorer
SQL Server
(localdb)\ProjectsV12 (SQL Server 12.0.2000 - contoso\la
Projects

3. In the **SQL Server Object Explorer**, right click on **SQL Server** and select **Add SQL Server…** option. Alternatively, click on **Add SQL Server** icon available on the **SQL Server Object Explorer** tab.

4. Enter **SQLONE** as the **Server name**

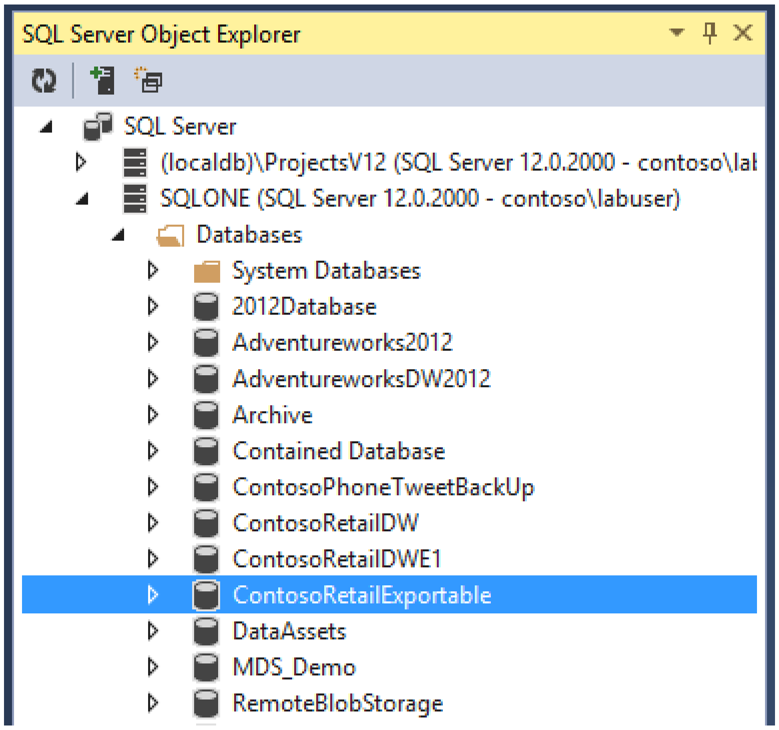
5. Select **Windows Authentication**



Server type: Database Engine
Server name: SQLONE
Authentication: Windows Authentication
User name: contoso\labuser
Password:
Remember password

6. Click on **Connect**

7. In the **SQL Server Object Explorer**, a node for **SQLONE** will now appear under the **SQL Server** node. Expand this node, then expand **Databases** folder to view all databases available in SQL Server 2014



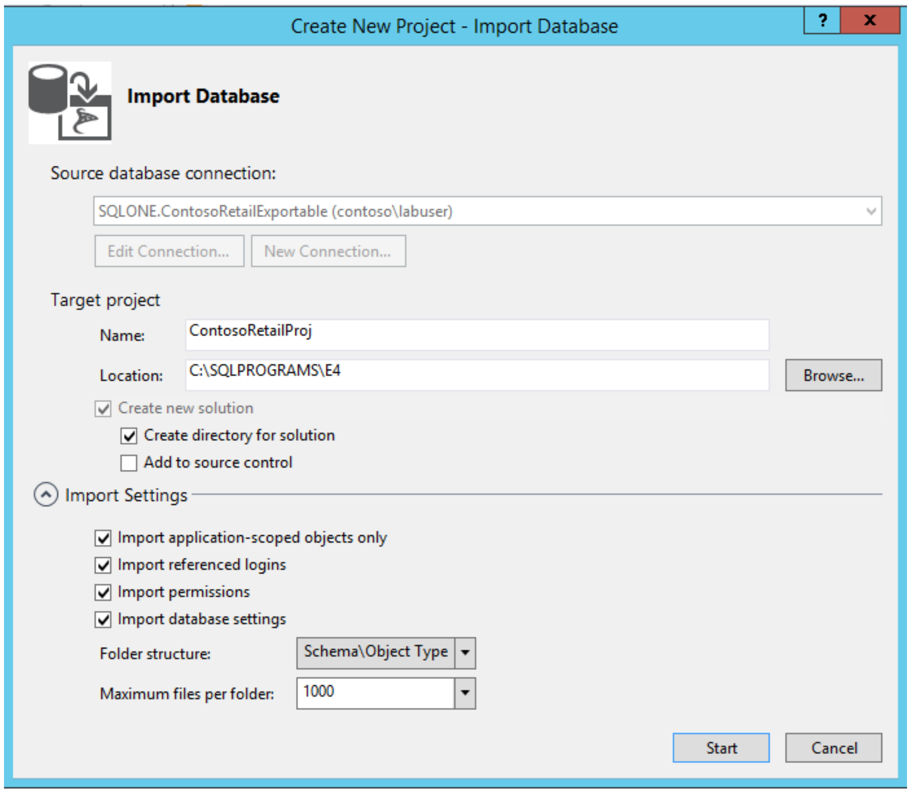
```
SQL Server
  (localdb)\ProjectsV12 (SQL Server 12.0.2000 - contoso\lal
  SQLONE (SQL Server 12.0.2000 - contoso\labuser)
    Databases
      System Databases
      2012Database
      Adventureworks2012
      AdventureworksDW2012
      Archive
      Contained Database
      ContosoPhoneTweetBackUp
      ContosoRetailDW
      ContosoRetailDWE1
      ContosoRetailExportable
      DataAssets
      MDS_Demo
      RemoteBlobStorage
```

Create a project from the ContosoRetailExportable database

1. Right click on **ContosoRetailExportable** and select **Create New project…**

2. In the **Create New Project – Import Database** wizard, change the **Target project** name to be **ContosoRetailProj** and the location to read **C:\SQLPROGRAMS\E4**, ensure **Create new solution** and **Create directory for solution** are both ticked, and **Add to source control** is not.

3. In the **Import Settings** section, check all the options listed



Create New Project - Import Database

Import Database

Source database connection:
SQLONE.ContosoRetailExportable (contoso\labuser)

Target project
Name: ContosoRetailProj
Location: C:\SQLPROGRAMS\E4

- [✓] Create new solution
- [✓] Create directory for solution
- [ ] Add to source control

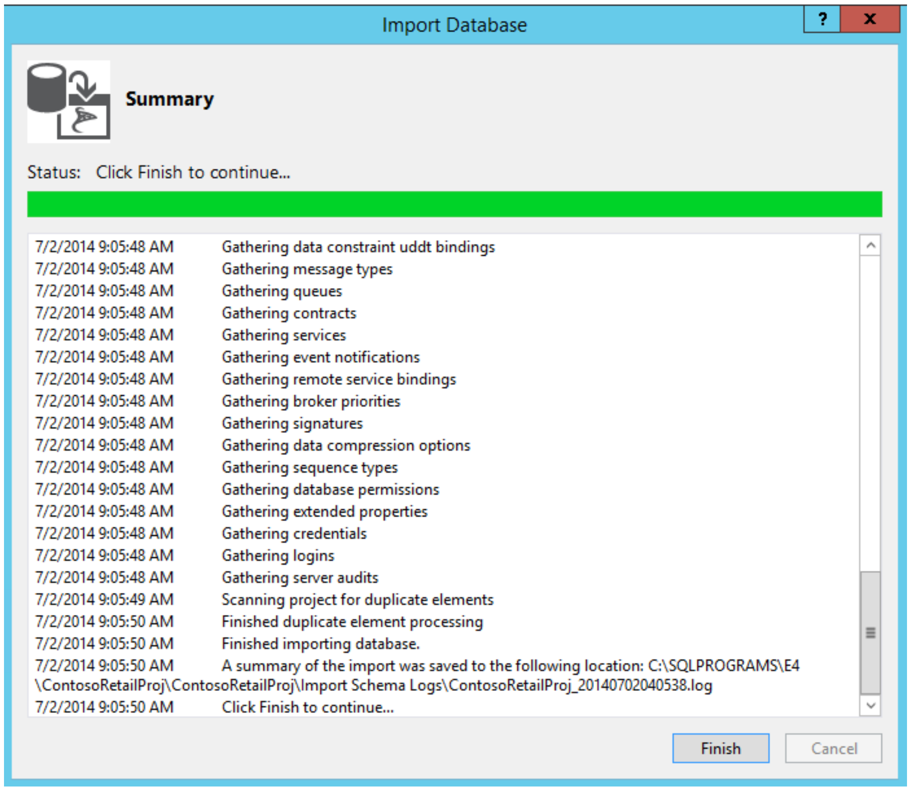
Import Settings

- [✓] Import application-scoped objects only
- [✓] Import referenced logins
- [✓] Import permissions
- [✓] Import database settings

Folder structure: Schema\Object Type
Maximum files per folder: 1000

4. Click **Start** to create the project. Click **OK** to overwrite if the message box appears.

5. Progress and summary of the import is shown as the last step. Note, it may take some time to enter the Summary page.

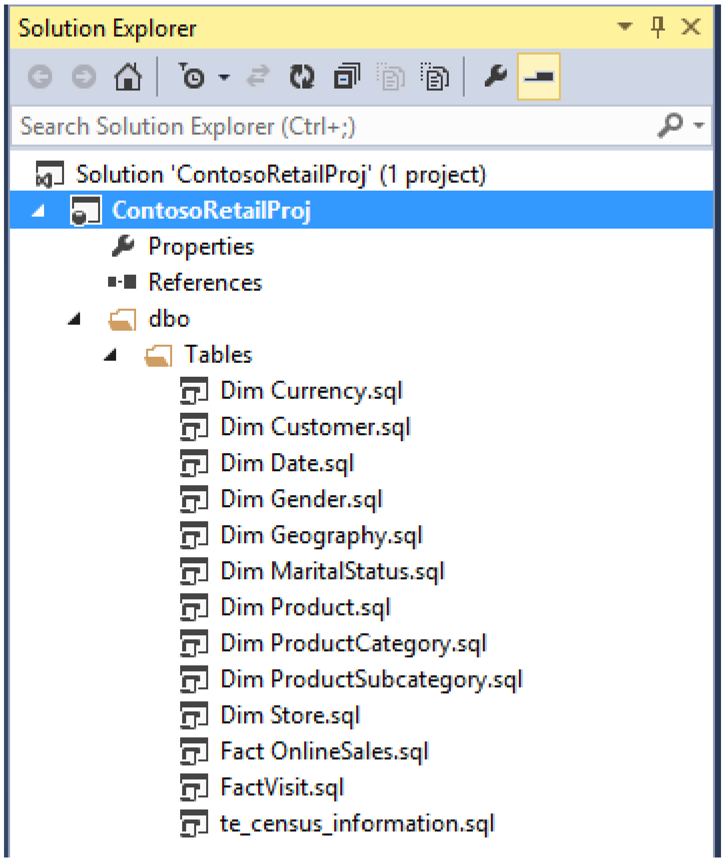


Status: Click Finish to continue...

```
7/2/2014 9:05:48 AM     Gathering data constraint uddt bindings
7/2/2014 9:05:48 AM     Gathering message types
7/2/2014 9:05:48 AM     Gathering queues
7/2/2014 9:05:48 AM     Gathering contracts
7/2/2014 9:05:48 AM     Gathering services
7/2/2014 9:05:48 AM     Gathering event notifications
7/2/2014 9:05:48 AM     Gathering remote service bindings
7/2/2014 9:05:48 AM     Gathering broker priorities
7/2/2014 9:05:48 AM     Gathering signatures
7/2/2014 9:05:48 AM     Gathering data compression options
7/2/2014 9:05:48 AM     Gathering sequence types
7/2/2014 9:05:48 AM     Gathering database permissions
7/2/2014 9:05:48 AM     Gathering extended properties
7/2/2014 9:05:48 AM     Gathering credentials
7/2/2014 9:05:48 AM     Gathering logins
7/2/2014 9:05:48 AM     Gathering server audits
7/2/2014 9:05:49 AM     Scanning project for duplicate elements
7/2/2014 9:05:50 AM     Finished duplicate element processing
7/2/2014 9:05:50 AM     Finished importing database.
7/2/2014 9:05:50 AM     A summary of the import was saved to the following location: C:\SQLPROGRAMS\E4
\ContosoRetailProj\ContosoRetailProj\Import Schema Logs\ContosoRetailProj_20140702040538.log
7/2/2014 9:05:50 AM     Click Finish to continue...
```

6. Click on **Finish** to exit the wizard

7. Click **View** -> **Solution Explorer** and expand the **dbo** and **Tables** folders.



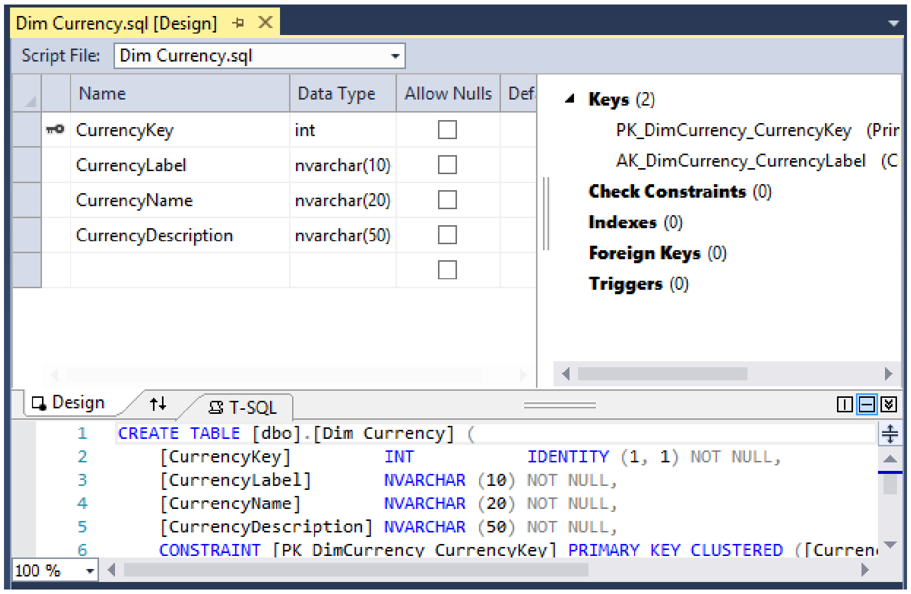
```
Solution 'ContosoRetailProj' (1 project)
  ContosoRetailProj
      Properties
      References
    dbo
      Tables
        Dim Currency.sql
        Dim Customer.sql
        Dim Date.sql
        Dim Gender.sql
        Dim Geography.sql
        Dim MaritalStatus.sql
        Dim Product.sql
        Dim ProductCategory.sql
        Dim ProductSubcategory.sql
        Dim Store.sql
        Fact OnlineSales.sql
        FactVisit.sql
        te_census_information.sql
```

*NOTE: The database objects are grouped by the Schema, so all objects in this database will be grouped under **dbo***

Browse and modify tables in Design Mode:

SQL Server Data Tools uses a local copy of the database, so any changes you make to the tables will not affect the underlying, live databases until you publish them.

1. In the **Solution Explorer**, right-click on the **Dim Currency.sql** table and select **View Designer** to view the table schema in the table designer.


Dim Currency.sql [Design]

Script File: Dim Currency.sql

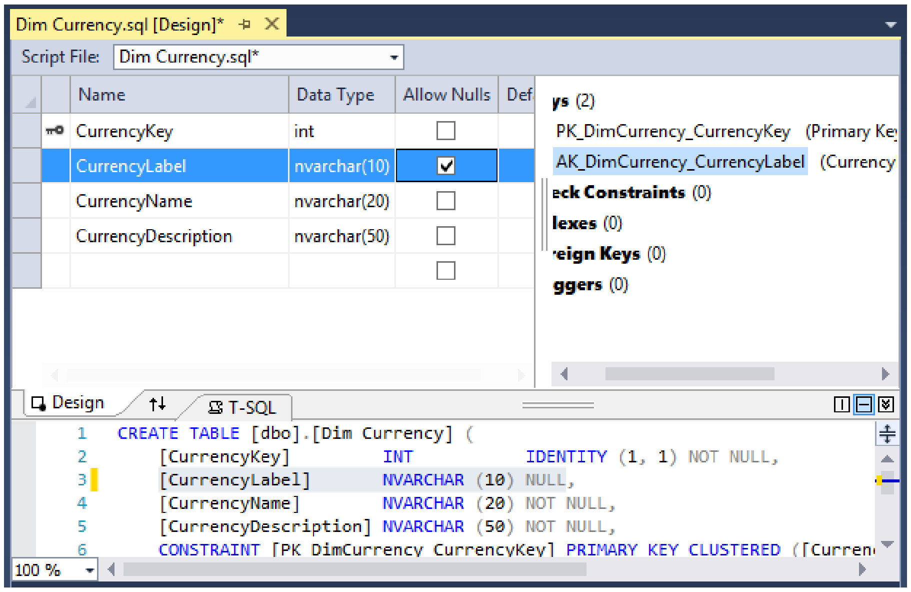
	Name	Data Type	Allow Nulls	Def
🔑	CurrencyKey	int	☐	
	CurrencyLabel	nvarchar(10)	☐	
	CurrencyName	nvarchar(20)	☐	
	CurrencyDescription	nvarchar(50)	☐	
			☐	

Keys (2)
PK_DimCurrency_CurrencyKey (Prir
AK_DimCurrency_CurrencyLabel (C
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

```
CREATE TABLE [dbo].[Dim Currency] (
    [CurrencyKey]         INT           IDENTITY (1, 1) NOT NULL,
    [CurrencyLabel]       NVARCHAR (10) NOT NULL,
    [CurrencyName]        NVARCHAR (20) NOT NULL,
    [CurrencyDescription] NVARCHAR (50) NOT NULL,
    CONSTRAINT [PK_DimCurrency_CurrencyKey] PRIMARY KEY CLUSTERED ([Curren
```

Alternatively, you can double click on table to open the table in design mode.

2. Check the **Allow Nulls** box for CurrencyLabel column. Notice how the corresponding code in the script pane is changed to **NULL** immediately.


Dim Currency.sql [Design]*

Script File: Dim Currency.sql*

	Name	Data Type	Allow Nulls	Def
🔑	CurrencyKey	int	☐	
	CurrencyLabel	nvarchar(10)	☑	
	CurrencyName	nvarchar(20)	☐	
	CurrencyDescription	nvarchar(50)	☐	
			☐	

ys (2)
PK_DimCurrency_CurrencyKey (Primary Key
AK_DimCurrency_CurrencyLabel (Currency
eck Constraints (0)
lexes (0)
eign Keys (0)
ggers (0)

```
CREATE TABLE [dbo].[Dim Currency] (
    [CurrencyKey]         INT           IDENTITY (1, 1) NOT NULL,
    [CurrencyLabel]       NVARCHAR (10) NULL,
    [CurrencyName]        NVARCHAR (20) NOT NULL,
    [CurrencyDescription] NVARCHAR (50) NOT NULL,
    CONSTRAINT [PK_DimCurrency_CurrencyKey] PRIMARY KEY CLUSTERED ([Curren
```

3. Similarly, any change to the T-SQL will be reflected in the designer

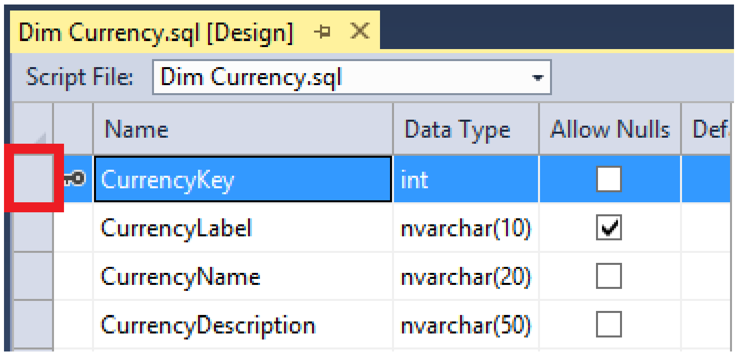
Design Mode allows for complicated and involved changes to be made intuitively, as any effects of the change are visible immediately.

4. Save the change by clicking on Save 💾 icon from the toolbar.

Change to a database object automatically updates related items in the model

The **Dim_Currency** table is referenced **by Fact_OnlineSales** via a foreign key on the **CurrencyKey** column. In this example, you will see how SSDT refactors the model when changing the primary key column name in the **Dim_Currency** table.

1. Go to the **Dim Currencty.sql [Design]** editor and select the **CurrencyKey** column in the table by clicking on the row selector.

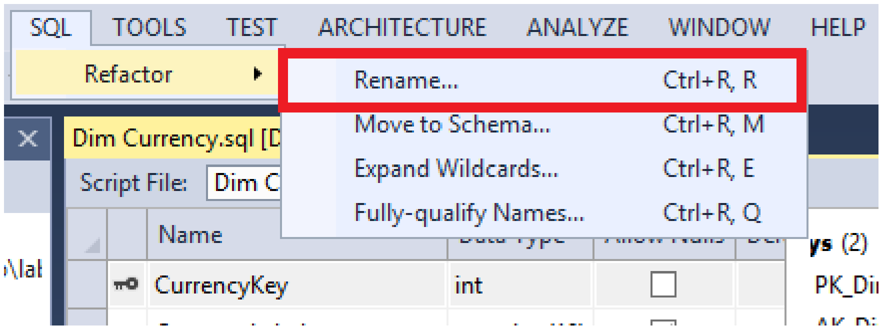


Dim Currency.sql [Design]

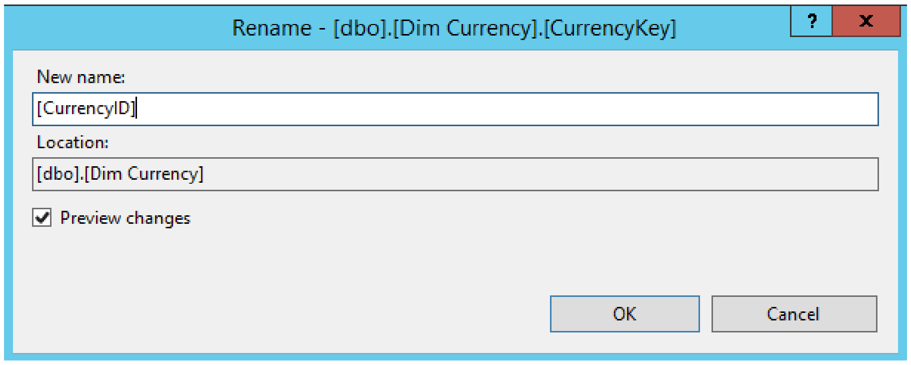
Script File: Dim Currency.sql

Name	Data Type	Allow Nulls	Def
CurrencyKey	int	☐	
CurrencyLabel	nvarchar(10)	✓	
CurrencyName	nvarchar(20)	☐	
CurrencyDescription	nvarchar(50)	☐	

2. SQL -> Refactor -> Rename


SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Refactor
Rename... Ctrl+R, R
Move to Schema... Ctrl+R, M
Expand Wildcards... Ctrl+R, E
Fully-qualify Names... Ctrl+R, Q
Dim Currency.sql [D
Script File: Dim C
Name
CurrencyKey int
ys (2)
PK_Di

3. Rename [**CurrencyKey]** to [**CurrencyId]** by selecting the value in the design view, changing it, and pressing OK.



Rename - [dbo].[Dim Currency].[CurrencyKey]

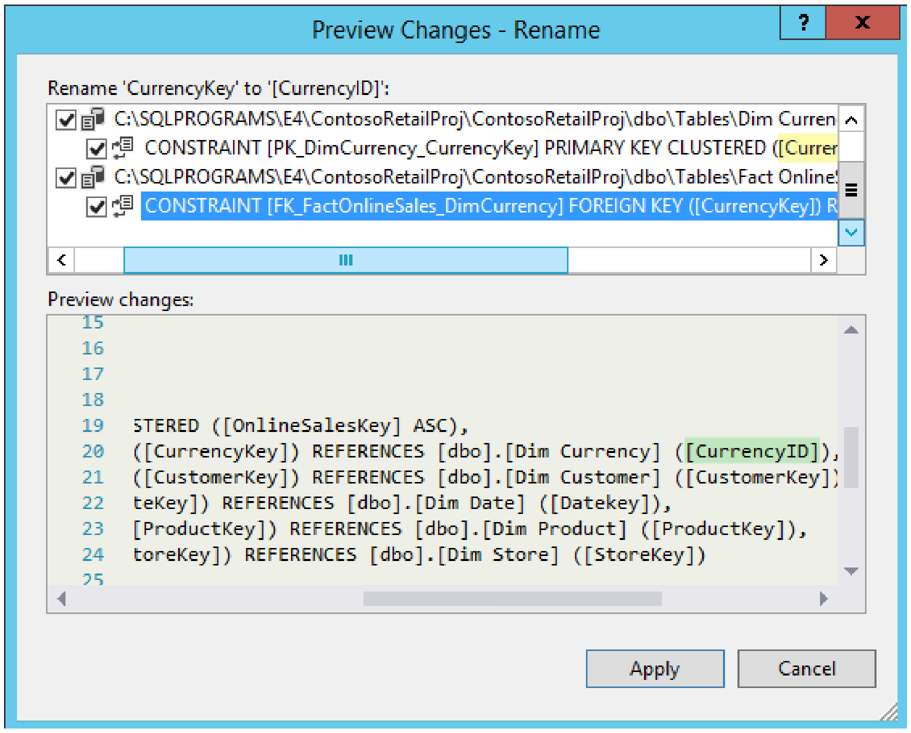
New name:
[CurrencyID]

Location:
[dbo].[Dim Currency]

✓ Preview changes

OK Cancel

4. The Preview Changes – Rename dialog allows you to see the impact of the column rename operation.


Preview Changes - Rename

Rename 'CurrencyKey' to '[CurrencyID]':

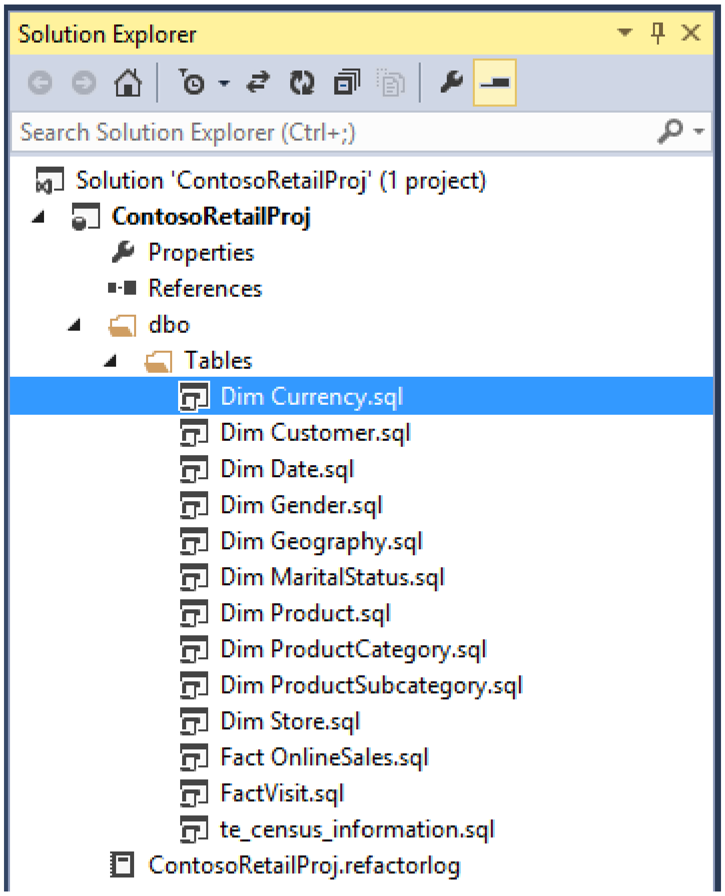
C:\SQLPROGRAMS\E4\ContosoRetailProj\ContosoRetailProj\dbo\Tables\Dim Curren
CONSTRAINT [PK_DimCurrency_CurrencyKey] PRIMARY KEY CLUSTERED ([Curre
C:\SQLPROGRAMS\E4\ContosoRetailProj\ContosoRetailProj\dbo\Tables\Fact Online
CONSTRAINT [FK_FactOnlineSales_DimCurrency] FOREIGN KEY ([CurrencyKey]) R

Preview changes:

```
STERED ([OnlineSalesKey] ASC),
([CurrencyKey]) REFERENCES [dbo].[Dim Currency] ([CurrencyID]),
([CustomerKey]) REFERENCES [dbo].[Dim Customer] ([CustomerKey])
teKey]) REFERENCES [dbo].[Dim Date] ([Datekey]),
[ProductKey]) REFERENCES [dbo].[Dim Product] ([ProductKey]),
toreKey]) REFERENCES [dbo].[Dim Store] ([StoreKey])
```

5. After examining the changes, click **Apply**.

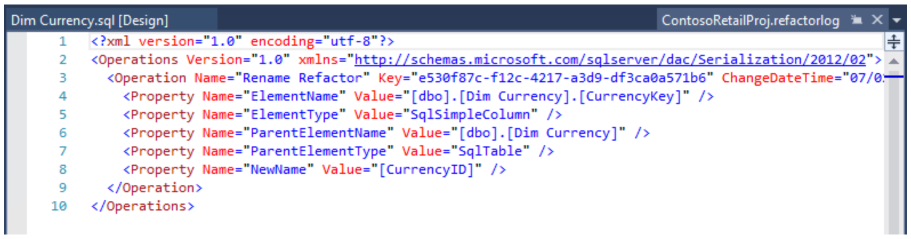
6. Notice that SSDT added a **ContosoRetailProj.refactorlog** to the project in the **Solution Explorer**.


Solution Explorer

Search Solution Explorer (Ctrl+;)

```
Solution 'ContosoRetailProj' (1 project)
◢ ContosoRetailProj
    Properties
    References
  ◢ dbo
    ◢ Tables
        Dim Currency.sql
        Dim Customer.sql
        Dim Date.sql
        Dim Gender.sql
        Dim Geography.sql
        Dim MaritalStatus.sql
        Dim Product.sql
        Dim ProductCategory.sql
        Dim ProductSubcategory.sql
        Dim Store.sql
        Fact OnlineSales.sql
        FactVisit.sql
        te_census_information.sql
    ContosoRetailProj.refactorlog
```

7. Single-click on the **ContosoRetailProj.refactorlog** file to view its contents.

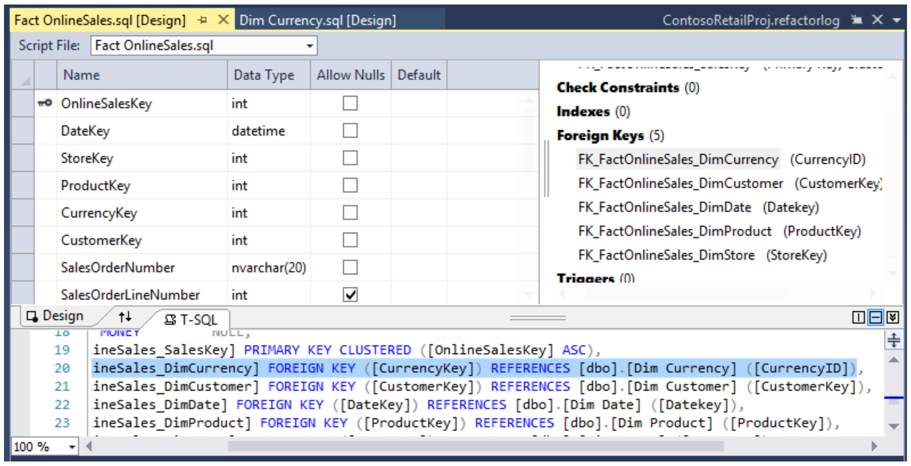

Dim Currency.sql [Design]   ContosoRetailProj.refactorlog

```
<?xml version="1.0" encoding="utf-8"?>
<Operations Version="1.0" xmlns="http://schemas.microsoft.com/sqlserver/dac/Serialization/2012/02">
  <Operation Name="Rename Refactor" Key="e530f87c-f12c-4217-a3d9-df3ca0a571b6" ChangeDateTime="07/0
    <Property Name="ElementName" Value="[dbo].[Dim Currency].[CurrencyKey]" />
    <Property Name="ElementType" Value="SqlSimpleColumn" />
    <Property Name="ParentElementName" Value="[dbo].[Dim Currency]" />
    <Property Name="ParentElementType" Value="SqlTable" />
    <Property Name="NewName" Value="[CurrencyID]" />
  </Operation>
</Operations>
```

8. Save the changes made to the table by clicking on **Save** icon from the tool bar. Alternatively, right click on the current tab and select **Save table** option.

9. From the **Solution Explorer** pane, double click on **FactOnlineSales.sql**

10. In the **T-SQL** pane in the central pane, scroll down to note that the foreign key in the table definition now refers to **CurrencyId** rather than **CurrencyKey** – this change happened automatically when you changed the **DimCurrency** table.


Fact OnlineSales.sql [Design]
Dim Currency.sql [Design]
ContosoRetailProj.refactorlog
Script File: Fact OnlineSales.sql
Design
T-SQL
100 %

Name	Data Type	Allow Nulls	Default
OnlineSalesKey	int	☐	
DateKey	datetime	☐	
StoreKey	int	☐	
ProductKey	int	☐	
CurrencyKey	int	☐	
CustomerKey	int	☐	
SalesOrderNumber	nvarchar(20)	☐	
SalesOrderLineNumber	int	✓	

Check Constraints (0)
Indexes (0)
Foreign Keys (5)
FK_FactOnlineSales_DimCurrency (CurrencyID)
FK_FactOnlineSales_DimCustomer (CustomerKey)
FK_FactOnlineSales_DimDate (Datekey)
FK_FactOnlineSales_DimProduct (ProductKey)
FK_FactOnlineSales_DimStore (StoreKey)
Triggers (0)

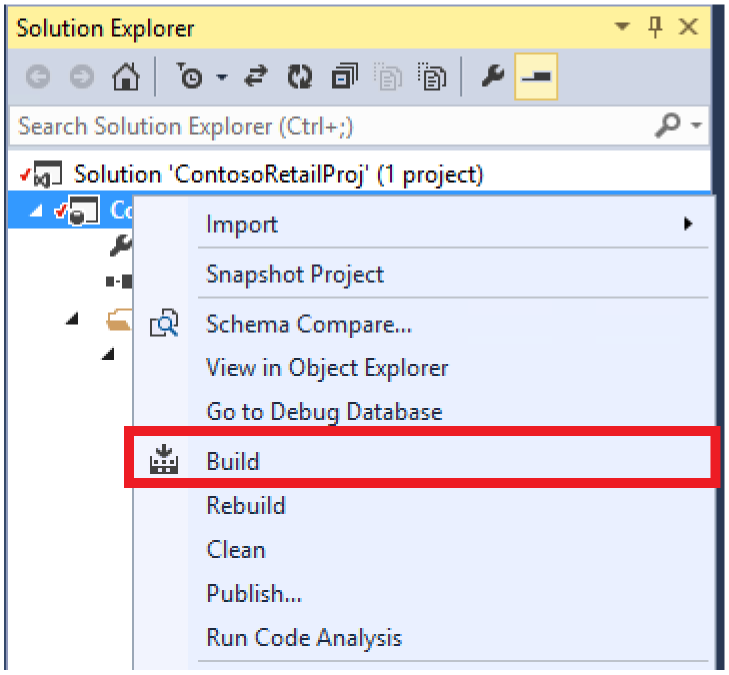
```
ineSales_SalesKey] PRIMARY KEY CLUSTERED ([OnlineSalesKey] ASC),
ineSales_DimCurrency] FOREIGN KEY ([CurrencyKey]) REFERENCES [dbo].[Dim Currency] ([CurrencyID]),
ineSales_DimCustomer] FOREIGN KEY ([CustomerKey]) REFERENCES [dbo].[Dim Customer] ([CustomerKey]),
ineSales_DimDate] FOREIGN KEY ([DateKey]) REFERENCES [dbo].[Dim Date] ([Datekey]),
ineSales_DimProduct] FOREIGN KEY ([ProductKey]) REFERENCES [dbo].[Dim Product] ([ProductKey]),
```

This makes it extremely simple to make complex and involved changes to the model. This allows you to use some of the very useful features of Visual Studio Integrated toolset, extract the database to create solutions using the database, which can be added to version control, and also opens a whole world of options for you to develop, manage and deploy the database to various environments.

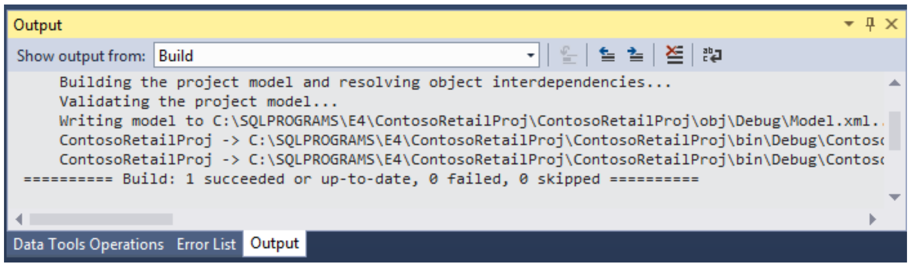
Deploy changes to a local sandbox database

You can see how your changes affect the database and if they cause any errors without deploying the changes to the main database running on SQLONE. You will deploy to the sandbox copy of the database that Visual Studio made when importing the SQLONE database into the project.

1. Right click on the project **ContosoRetailProj** from the **Solution Explorer** pane and click on **Build** to make sure there are no errors/issue in the project



2. If there are any build errors, correct the errors before proceeding



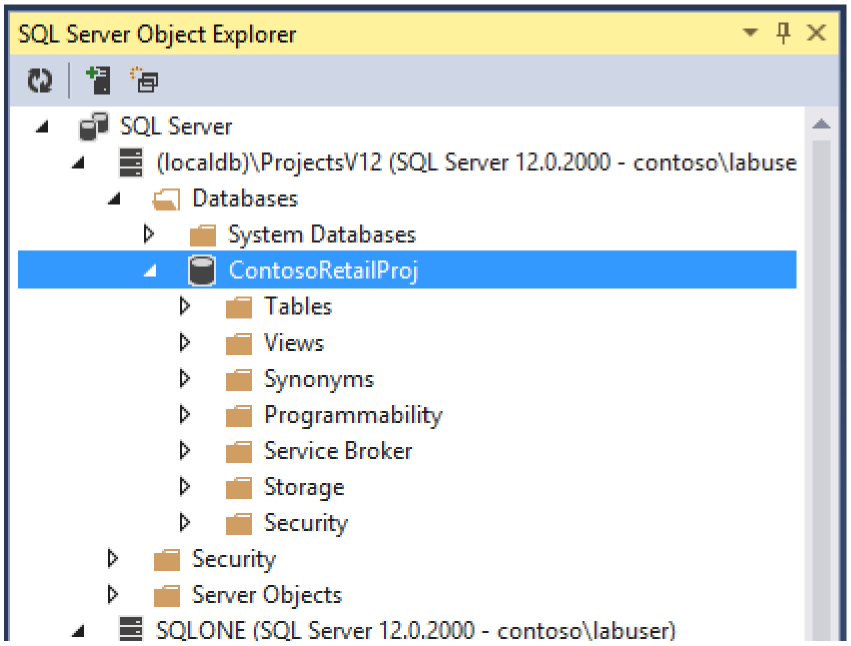
```
    Building the project model and resolving object interdependencies...
    Validating the project model...
    Writing model to C:\SQLPROGRAMS\E4\ContosoRetailProj\ContosoRetailProj\obj\Debug\Model.xml..
    ContosoRetailProj -> C:\SQLPROGRAMS\E4\ContosoRetailProj\ContosoRetailProj\bin\Debug\Contos
    ContosoRetailProj -> C:\SQLPROGRAMS\E4\ContosoRetailProj\ContosoRetailProj\bin\Debug\Contos
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

3. Click on ▶ **Start** icon from the toolbar

*This deploys the changes to a local sandbox database **(localdb)\ProjectsV12** which can be located under **SQL Server** node in the **SQL server Object Explorer** pane*

4. Expand the **Databases** node under the **(localdb)\ProjectsV12**

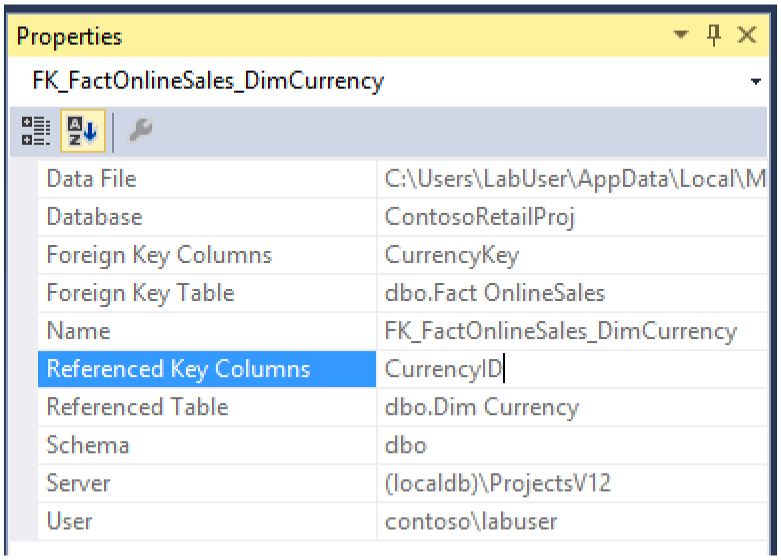
5. Right click on the **ContosoRetailProj** node and click **Refresh**


SQL Server Object Explorer

- SQL Server
  - (localdb)\ProjectsV12 (SQL Server 12.0.2000 - contoso\labuse
    - Databases
      - System Databases
      - ContosoRetailProj
        - Tables
        - Views
        - Synonyms
        - Programmability
        - Service Broker
        - Storage
        - Security
    - Security
    - Server Objects
  - SQLONE (SQL Server 12.0.2000 - contoso\labuser)

6. Expand **FactOnlineSales** and then expand **Keys**

7. Click on **FK_FactOnlineSales_DimCurrency**. Note how the properties for this foreign key relationship is shown in the **Properties** window on the right. Note the **Referenced Key Columns** to see how it has changed to **CurrencyID** reflecting the changes made in the refactor operation.

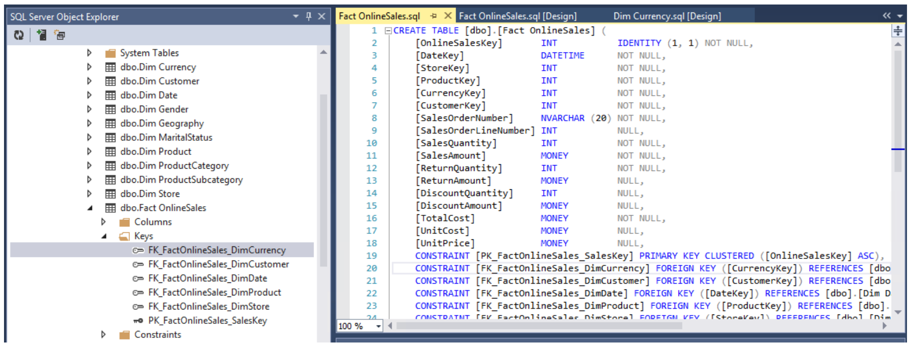

Properties

FK_FactOnlineSales_DimCurrency

Data File	C:\Users\LabUser\AppData\Local\M
Database	ContosoRetailProj
Foreign Key Columns	CurrencyKey
Foreign Key Table	dbo.Fact OnlineSales
Name	FK_FactOnlineSales_DimCurrency
Referenced Key Columns	CurrencyID
Referenced Table	dbo.Dim Currency
Schema	dbo
Server	(localdb)\ProjectsV12
User	contoso\labuser

8. If you want to see the code change for the key, go back to the SQL Server Object Explorer and double-click on the FK_FactOnlineSales_DimCurrency key.

Notice that SSDT highlights the line of code in the editor that defines the foreign key constraint.


SQL Server Object Explorer
System Tables
dbo.Dim Currency
dbo.Dim Customer
dbo.Dim Date
dbo.Dim Gender
dbo.Dim Geography
dbo.Dim MaritalStatus
dbo.Dim Product
dbo.Dim ProductCategory
dbo.Dim ProductSubcategory
dbo.Dim Store
dbo.Fact OnlineSales
Columns
Keys
FK_FactOnlineSales_DimCurrency
FK_FactOnlineSales_DimCustomer
FK_FactOnlineSales_DimDate
FK_FactOnlineSales_DimProduct
FK_FactOnlineSales_DimStore
PK_FactOnlineSales_SalesKey
Constraints
Fact OnlineSales.sql
Fact OnlineSales.sql [Design]
Dim Currency.sql [Design]
100 %

```
CREATE TABLE [dbo].[Fact OnlineSales] (
    [OnlineSalesKey]       INT            IDENTITY (1, 1) NOT NULL,
    [DateKey]              DATETIME       NOT NULL,
    [StoreKey]             INT            NOT NULL,
    [ProductKey]           INT            NOT NULL,
    [CurrencyKey]          INT            NOT NULL,
    [CustomerKey]          INT            NOT NULL,
    [SalesOrderNumber]     NVARCHAR (20)  NOT NULL,
    [SalesOrderLineNumber] INT            NULL,
    [SalesQuantity]        INT            NOT NULL,
    [SalesAmount]          MONEY          NOT NULL,
    [ReturnQuantity]       INT            NOT NULL,
    [ReturnAmount]         MONEY          NULL,
    [DiscountQuantity]     INT            NULL,
    [DiscountAmount]       MONEY          NULL,
    [TotalCost]            MONEY          NOT NULL,
    [UnitCost]             MONEY          NULL,
    [UnitPrice]            MONEY          NULL,
    CONSTRAINT [PK_FactOnlineSales_SalesKey] PRIMARY KEY CLUSTERED ([OnlineSalesKey] ASC),
    CONSTRAINT [FK_FactOnlineSales_DimCurrency] FOREIGN KEY ([CurrencyKey]) REFERENCES [dbo
    CONSTRAINT [FK_FactOnlineSales_DimCustomer] FOREIGN KEY ([CustomerKey]) REFERENCES [dbo
    CONSTRAINT [FK_FactOnlineSales_DimDate] FOREIGN KEY ([DateKey]) REFERENCES [dbo].[Dim D
    CONSTRAINT [FK_FactOnlineSales_DimProduct] FOREIGN KEY ([ProductKey]) REFERENCES [dbo].
    CONSTRAINT [FK_FactOnlineSales_DimStore] FOREIGN KEY ([StoreKey]) REFERENCES [dbo].[Dim
```

9. Exit **Visual Studio 2013** after saving the project

Above stated feature allows you to work on a local isolated copy of the database to create, test and deploy the changes before affecting a live environment. He can later deploy these changes back to the production database, if he desires.

You can see how using the SQL Server Data Tools and their increased integration and functionality with Visual Studio, the DBA's can save time and effort when changing schemas and administering the database. The ability to be able to review the changes and save the scripts that are generated will also save time and rework if he wants to deploy them into a live environment knowing that he has already been able to test the changes on a local copy knowing that the changes will work.

Source code control using Git Remote Repository

DBA's can benefit from having a more robust change, development and code repository to use when altering the databases, much in the same way developers do when writing code. By taking advantage of the integration between SQL Server and Visual Studio as well as the functionality and benefits offered by Team Foundation Server (TFS)/Git, You can now version and snapshot all the changes to the database as well as script changes, and store these in a source safe repository. Once you have your changes committed and the code saved, you will use SQL Server Data Tools to publish the new database back to SQL 2014 Server.

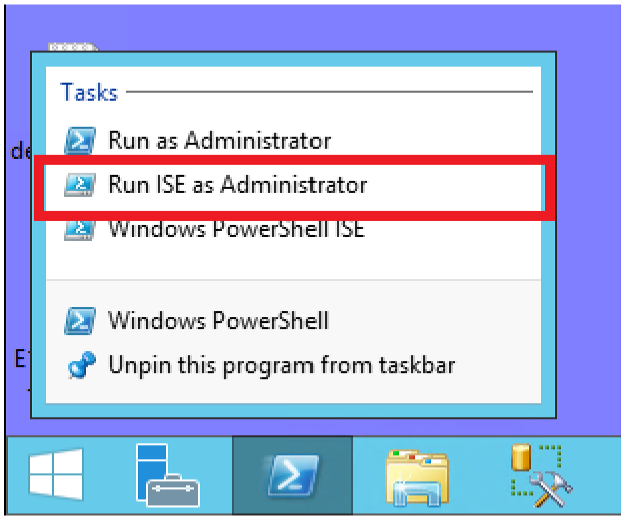
SQL Server Data Tools to allow you to adjust the target of where you wish to publish the data to. This is particularly useful when moving databases from different versions of SQL Server up to Windows Azure or if people in the team have made changes to the database that aren't compatible with Windows Azure. You can use this tool to quickly test and correct other people's modifications as required as you deploy to other platforms.

Create the Repository

You will use Git version control (an open-source form of version control more focused on flexibility.) Once you have completed this step, use that form of version control throughout the scenario.

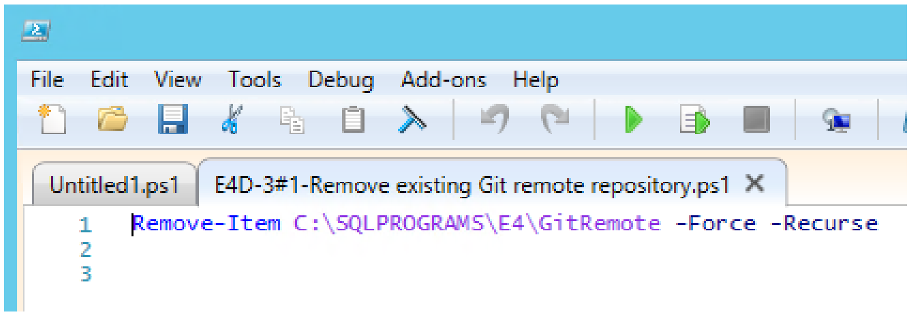
Setting up Git Remote Repository

1. Right click on the **PowerShell** icon on the toolbar and click **Run ISE as Administrator.** Click **Yes** for the **User Access Control** dialog.



2. Press Ctrl+O to open a file, navigate to **C:\SQLSCRIPTS\E4\** and select the script **E4D-3#1-Remove existing Git remote repository.ps1.** Click **Open**.

This script deletes the existing Git remote repository



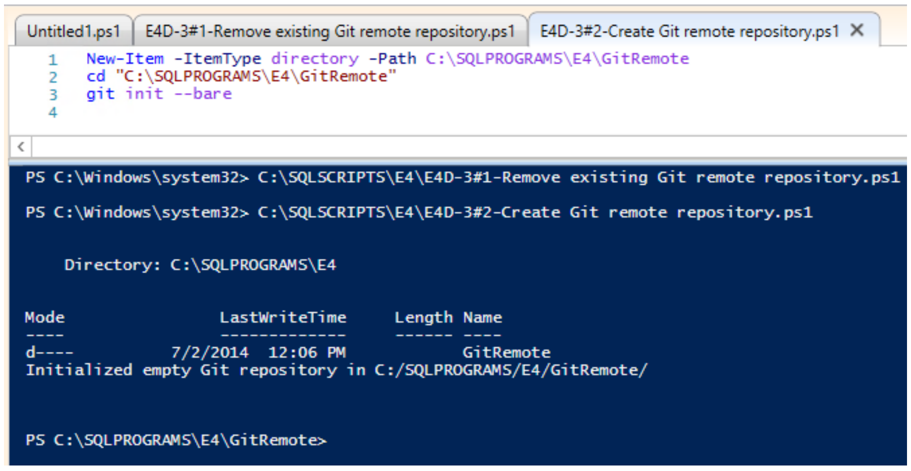
```
Remove-Item C:\SQLPROGRAMS\E4\GitRemote -Force -Recurse
```

3. Press **F5** to execute the statement.

4. Press Ctrl+O and select the script **E4D-3#2-Create Git remote repository.ps1**' and press Open.

This creates and initializes the Git remote repository at C:\SQLPROGRAMS\E4\GitRemote. Note, a Git remote repository can also be hosted online at Github.com, rather than on your local machine as is being shown here.

5. Press **F5** to execute the script.



Untitled1.ps1 | E4D-3#1-Remove existing Git remote repository.ps1 | E4D-3#2-Create Git remote repository.ps1

```
New-Item -ItemType directory -Path C:\SQLPROGRAMS\E4\GitRemote
cd "C:\SQLPROGRAMS\E4\GitRemote"
git init --bare
```

```
PS C:\Windows\system32> C:\SQLSCRIPTS\E4\E4D-3#1-Remove existing Git remote repository.ps1

PS C:\Windows\system32> C:\SQLSCRIPTS\E4\E4D-3#2-Create Git remote repository.ps1


    Directory: C:\SQLPROGRAMS\E4


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----          7/2/2014  12:06 PM            GitRemote
Initialized empty Git repository in C:/SQLPROGRAMS/E4/GitRemote/


PS C:\SQLPROGRAMS\E4\GitRemote>
```

6. Close the **Windows PowerShell ISE** program.

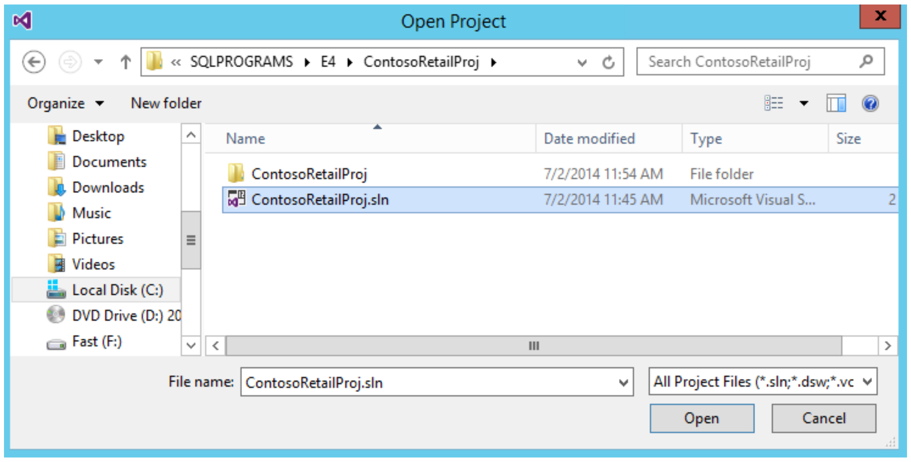
Checking a database project into source control server

Git allows version controlling and shared access to the database project. To add your project solution to a version control server, follow the steps below:

1. Open **Visual Studio 2013**

2. Press **Ctrl+Shift_O** to open a **Project/Solution**

3. From the **Open Project** window double click **ContosoRetailProj** folder, select the file **ContosoRetailProj.sln** then click **Open**. This will open the **Solution Explorer** as a pane on the right hand side.



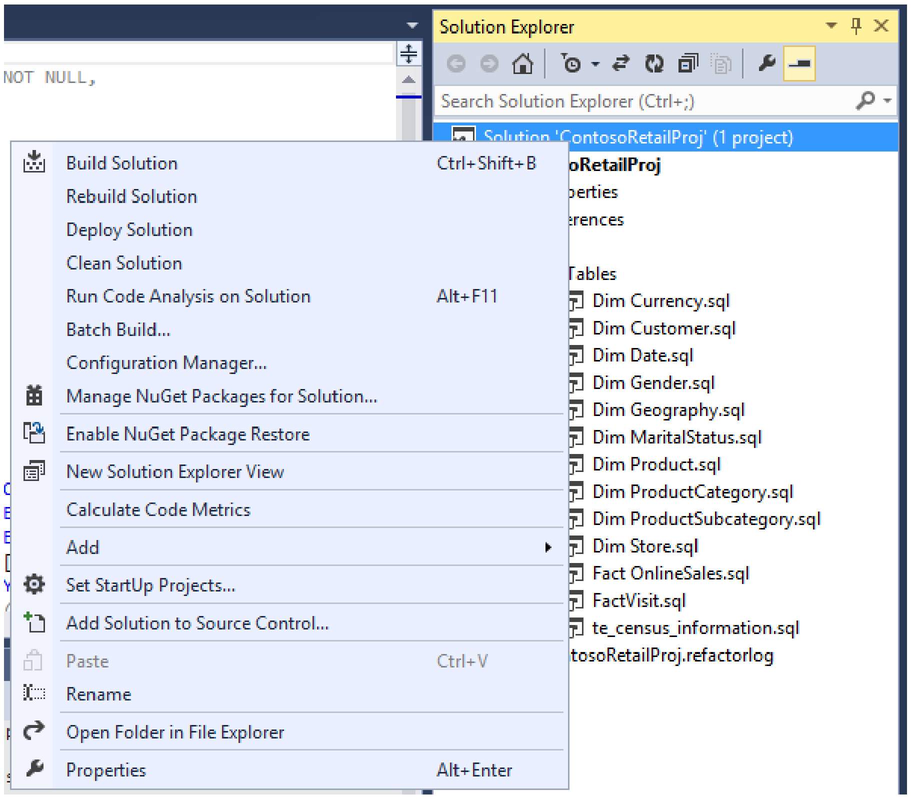
Name	Date modified	Type	Size
ContosoRetailProj	7/2/2014 11:54 AM	File folder	
ContosoRetailProj.sln	7/2/2014 11:45 AM	Microsoft Visual S...	2

File name: ContosoRetailProj.sln
All Project Files (*.sln;*.dsw;*.vc

*If not then from the **View** menu, select **Solution Explorer***

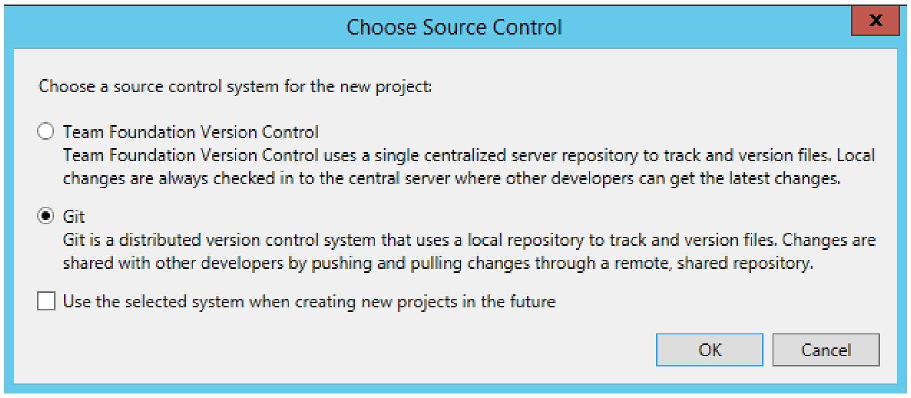
4. On the **File** menu, click **Add to Source Control.** Alternatively, right click on the project solution and click on (**Source Control**

>) **Add Solution to Source Control**. (The bracketed sections will only be present if you are using a TFS server.)



```
NOT NULL,
```

5. **Choose Source control** dialog appears
6. **Select** one of the Git option and click on **OK**


Choose Source Control

Choose a source control system for the new project:

( ) Team Foundation Version Control
Team Foundation Version Control uses a single centralized server repository to track and version files. Local changes are always checked in to the central server where other developers can get the latest changes.

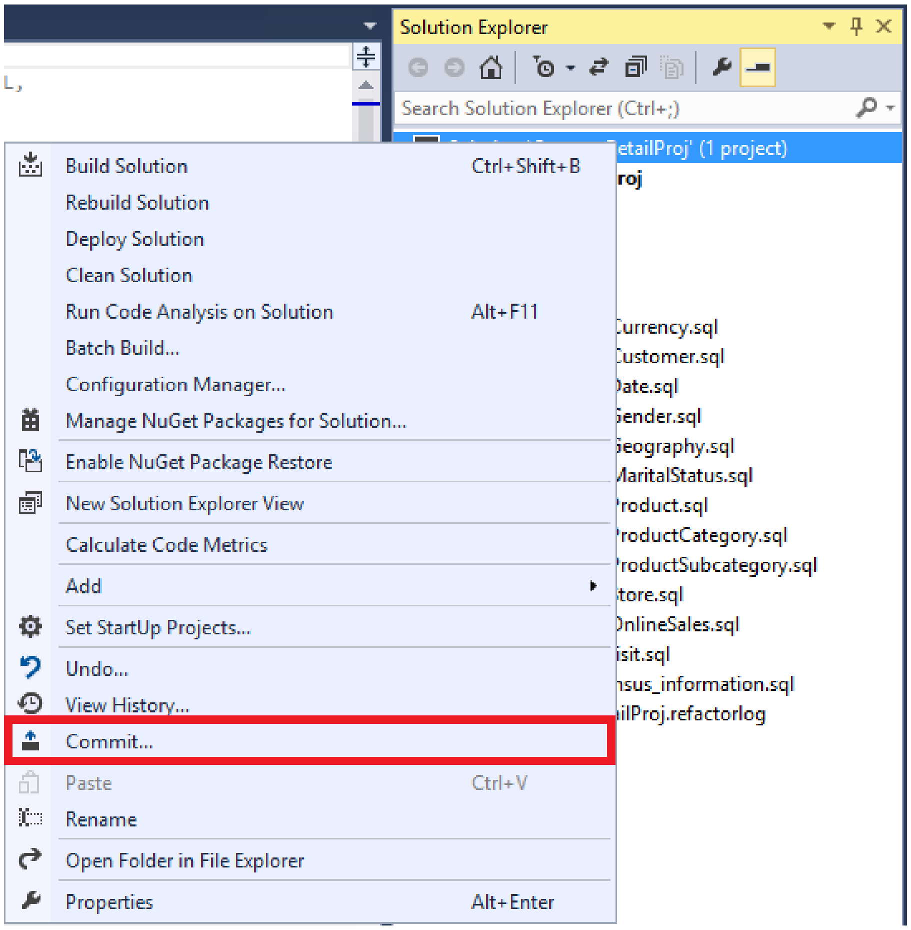
(●) Git
Git is a distributed version control system that uses a local repository to track and version files. Changes are shared with other developers by pushing and pulling changes through a remote, shared repository.

☐ Use the selected system when creating new projects in the future

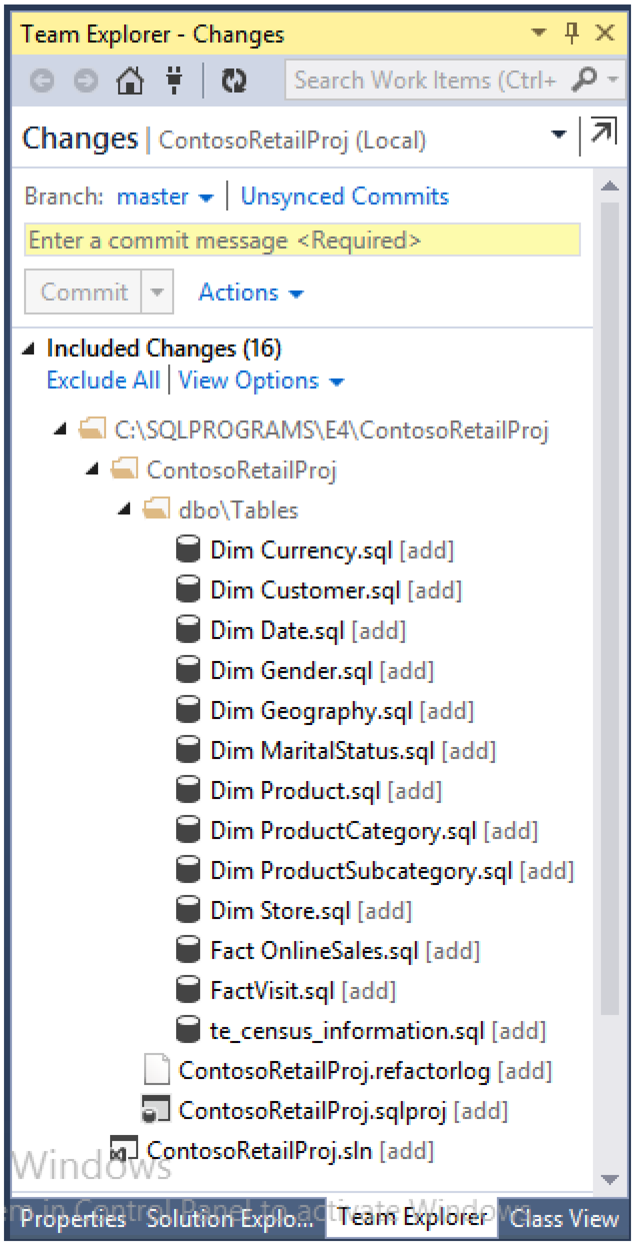
*NOTE: **Team Foundation Version Control** option uses a single centralized server repository to track and version files. Local changes are always checked in to the central server where other developers latest changes. **Git** option could be used to enable distributed version control system that uses a local repository to track and version files. Changes are shared with other developers by pulling changes through a remote, shared repository.*

7. Go to **Solution Explorer** pane

8. Right click on the solution and click on Commit... icon (may be within Source Control menu)



9. The **Team Explorer** – **Changes** pane will be displayed, the list of changes are listed


Team Explorer - Changes

Search Work Items (Ctrl+

Changes | ContosoRetailProj (Local)

Branch: master | Unsynced Commits

Enter a commit message <Required>

Commit | Actions

Included Changes (16)

Exclude All | View Options

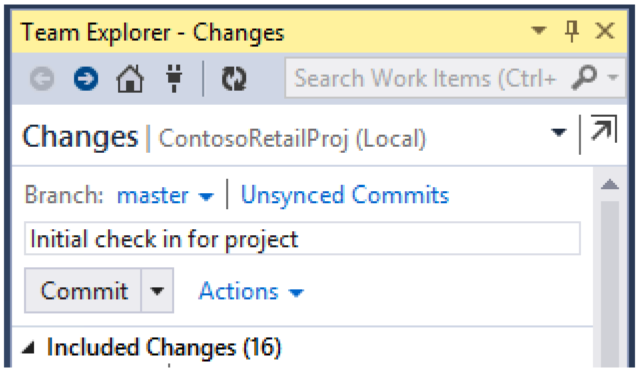
- C:\SQLPROGRAMS\E4\ContosoRetailProj
  - ContosoRetailProj
    - dbo\Tables
      - Dim Currency.sql [add]
      - Dim Customer.sql [add]
      - Dim Date.sql [add]
      - Dim Gender.sql [add]
      - Dim Geography.sql [add]
      - Dim MaritalStatus.sql [add]
      - Dim Product.sql [add]
      - Dim ProductCategory.sql [add]
      - Dim ProductSubcategory.sql [add]
      - Dim Store.sql [add]
      - Fact OnlineSales.sql [add]
      - FactVisit.sql [add]
      - te_census_information.sql [add]
    - ContosoRetailProj.refactorlog [add]
    - ContosoRetailProj.sqlproj [add]
  - ContosoRetailProj.sln [add]

Properties | Solution Explo... | Team Explorer | Class View

*NOTE: If "Configure your user name and email address before committing changes" message is shown on top of the Team Explorer – Changes pane, click on **Configure** link in the message. Provide your user name, email address and click on **Update**.*

Commit changes

1. Review the list of changes and type a commit message (e.g. **Original Version**) to help while trying to retrieve a version of code at a later point of time


Team Explorer - Changes

Changes | ContosoRetailProj (Local)

Branch: master | Unsynced Commits

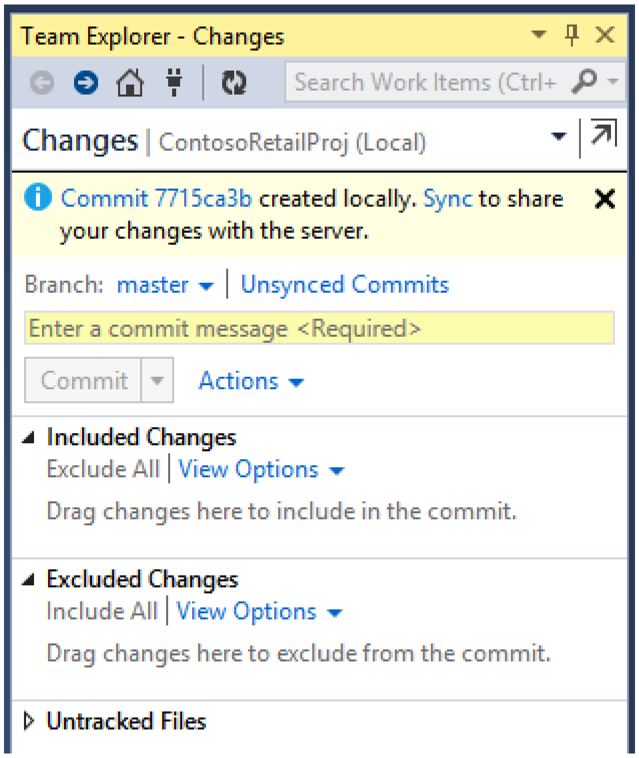
Initial check in for project

Commit | Actions

Included Changes (16)

2. Click **Commit**

NOTE: Committing the project stores the new version on your local machine until you push it onto the server


Team Explorer - Changes

Changes | ContosoRetailProj (Local)

Commit 7715ca3b created locally. Sync to share your changes with the server.

Branch: master | Unsynced Commits

Enter a commit message <Required>

Commit | Actions

Included Changes

Exclude All | View Options

Drag changes here to include in the commit.

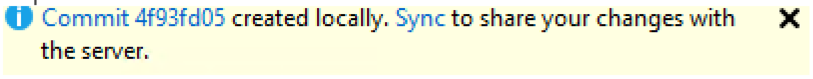
Excluded Changes

Include All | View Options

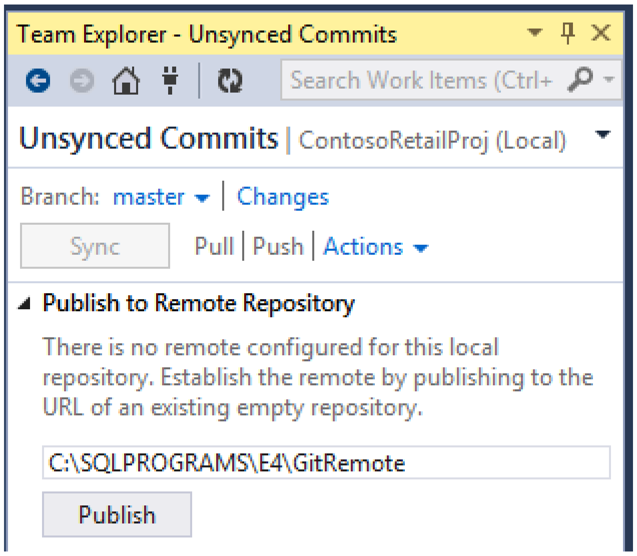
Drag changes here to exclude from the commit.

Untracked Files

3. Click on **Sync** link from the message on the top of the Team Explorer bar


Commit 4f93fd05 created locally. Sync to share your changes with the server.

4. Provide the Git URL as **C:\SQLPROGRAMS\E4\GitRemote** and click on **Publish**


Team Explorer - Unsynced Commits

Search Work Items (Ctrl+

Unsynced Commits | ContosoRetailProj (Local)

Branch: master | Changes

Sync Pull | Push | Actions

Publish to Remote Repository

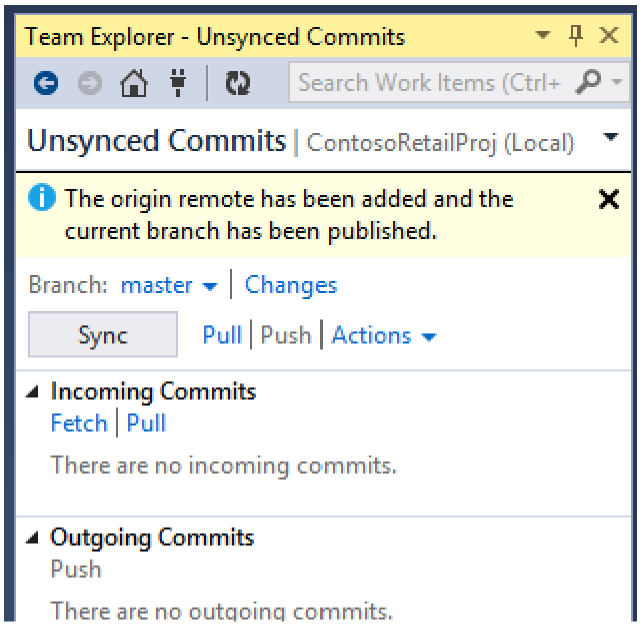
There is no remote configured for this local repository. Establish the remote by publishing to the URL of an existing empty repository.

C:\SQLPROGRAMS\E4\GitRemote

Publish

*In the **Team Explorer** a message will be displayed to inform you that the origin remote has been created a current branch has been published*

If you need to sync the local branch with the remote branch master click on the Sync button.


Team Explorer - Unsynced Commits

Search Work Items (Ctrl+

Unsynced Commits | ContosoRetailProj (Local)

The origin remote has been added and the current branch has been published.

Branch: master | Changes

Sync Pull | Push | Actions

Incoming Commits

Fetch | Pull

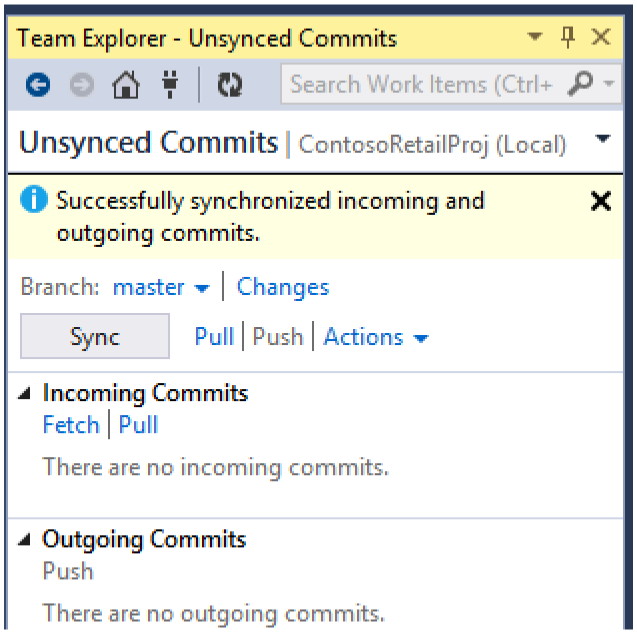
There are no incoming commits.

Outgoing Commits

Push

There are no outgoing commits.

*In the **Team Explorer** a message will then be displayed to confirm that commits have been synchronized.*


Team Explorer - Unsynced Commits

Search Work Items (Ctrl+

Unsynced Commits | ContosoRetailProj (Local)

Successfully synchronized incoming and outgoing commits.

Branch: master | Changes

Sync   Pull | Push | Actions

Incoming Commits
Fetch | Pull
There are no incoming commits.

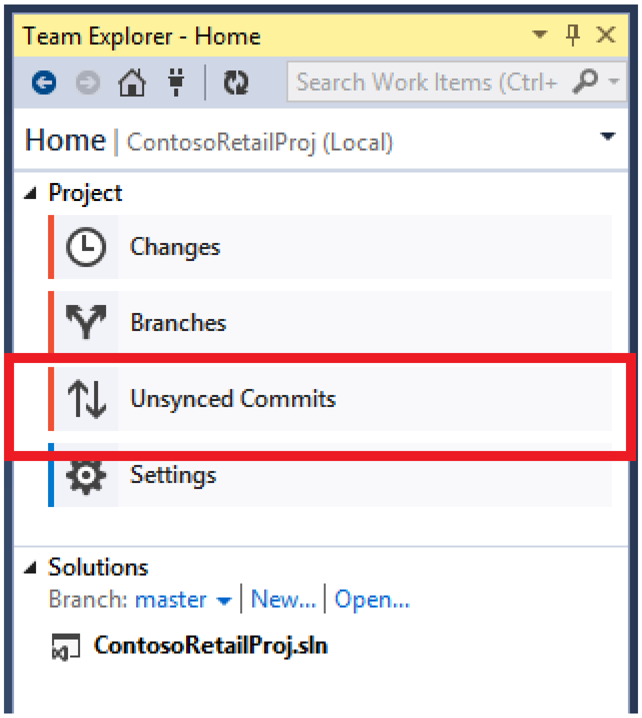
Outgoing Commits
Push
There are no outgoing commits.

Retrieve/Pull committed changes from Git

Git is designed for collaborative work across a team, so this option is to get work committed by other team members.

1. Ensure that **Team Explorer** is open at the **Unsynced Commits** view.

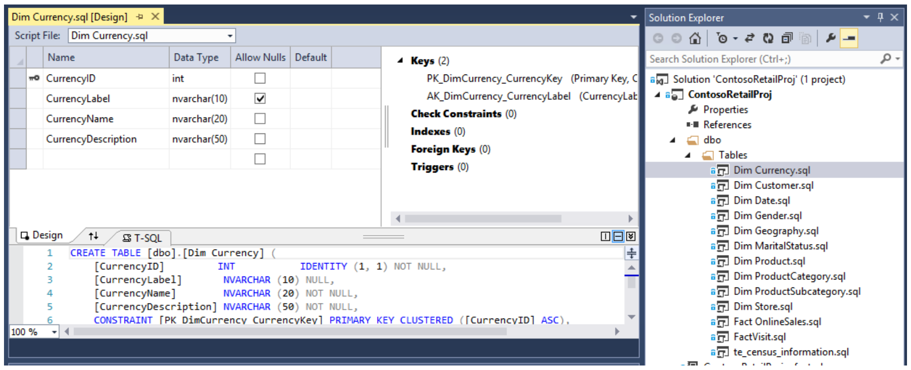
   *If it is not already then click on the **Home** icon then click on **Unsynched Commits***



2. If this project had another team member who had committed work since you last updated your local repository, their recent commits would be listed under **Incoming Commits**. You would select the commit you wanted to look at and click **fetch** (to preview the effect of the changes before integrating them into your work) or **pull** (to merge the commit with your local version.)

Modify Database Objects and check-in

1. Open **Solution Explorer** to view all the SQL objects available in the project
2. Expand the project node, followed by **dbo** and **Tables** node
3. Double-click on **DimCurrency.sql** to open it in design mode


Dim Currency.sql [Design]
Script File: Dim Currency.sql
Design
T-SQL
100 %
Solution Explorer
Search Solution Explorer (Ctrl+;)

	Name	Data Type	Allow Nulls	Default
	CurrencyID	int	☐	
	CurrencyLabel	nvarchar(10)	✓	
	CurrencyName	nvarchar(20)	☐	
	CurrencyDescription	nvarchar(50)	☐	
			☐	

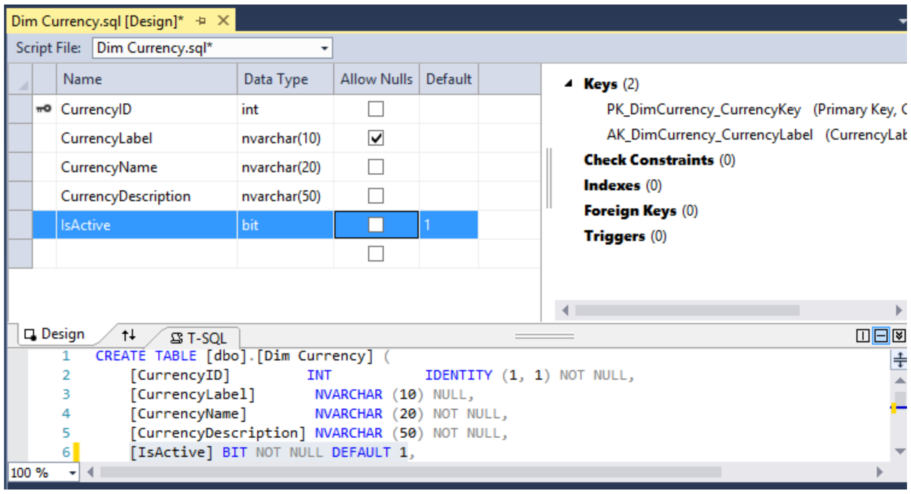
Keys (2)
PK_DimCurrency_CurrencyKey (Primary Key, C
AK_DimCurrency_CurrencyLabel (CurrencyLab
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

```
CREATE TABLE [dbo].[Dim Currency] (
    [CurrencyID]          INT           IDENTITY (1, 1) NOT NULL,
    [CurrencyLabel]       NVARCHAR (10) NULL,
    [CurrencyName]        NVARCHAR (20) NOT NULL,
    [CurrencyDescription] NVARCHAR (50) NOT NULL,
    CONSTRAINT [PK DimCurrency CurrencyKey] PRIMARY KEY CLUSTERED ([CurrencyID] ASC),
```

```
Solution 'ContosoRetailProj' (1 project)
  ContosoRetailProj
    Properties
    References
    dbo
      Tables
        Dim Currency.sql
        Dim Customer.sql
        Dim Date.sql
        Dim Gender.sql
        Dim Geography.sql
        Dim MaritalStatus.sql
        Dim Product.sql
        Dim ProductCategory.sql
        Dim ProductSubcategory.sql
        Dim Store.sql
        Fact OnlineSales.sql
        FactVisit.sql
        te_census_information.sql
```

4. Add a new non null [IsActive] column of type [Bit] and default it to have a value of 1. You can either do this by clicking in the

blank line under **CurrencyDescription** in the design section and entering the values, or by adding a new line in the CREATE TABLE script in the T-SQL view between the [CurrencyDescription] line and the first CONSTRAINT line to read [IsActive] BIT NOT NULL DEFAULT (1),



Name	Data Type	Allow Nulls	Default
CurrencyID	int	☐	
CurrencyLabel	nvarchar(10)	✓	
CurrencyName	nvarchar(20)	☐	
CurrencyDescription	nvarchar(50)	☐	
IsActive	bit	☐	1
		☐	

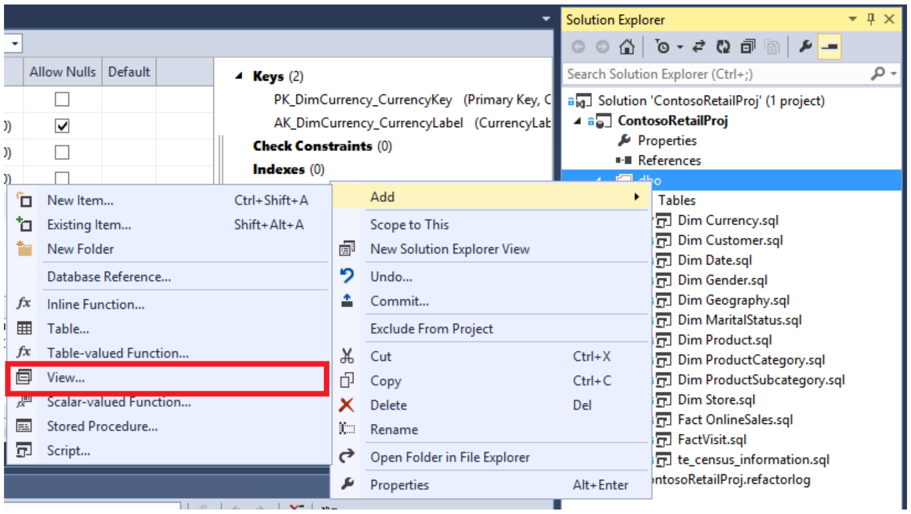
Keys (2)
PK_DimCurrency_CurrencyKey (Primary Key, C
AK_DimCurrency_CurrencyLabel (CurrencyLab
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

```
CREATE TABLE [dbo].[Dim Currency] (
    [CurrencyID]          INT            IDENTITY (1, 1) NOT NULL,
    [CurrencyLabel]       NVARCHAR (10) NULL,
    [CurrencyName]        NVARCHAR (20) NOT NULL,
    [CurrencyDescription] NVARCHAR (50) NOT NULL,
    [IsActive] BIT NOT NULL DEFAULT 1,
```

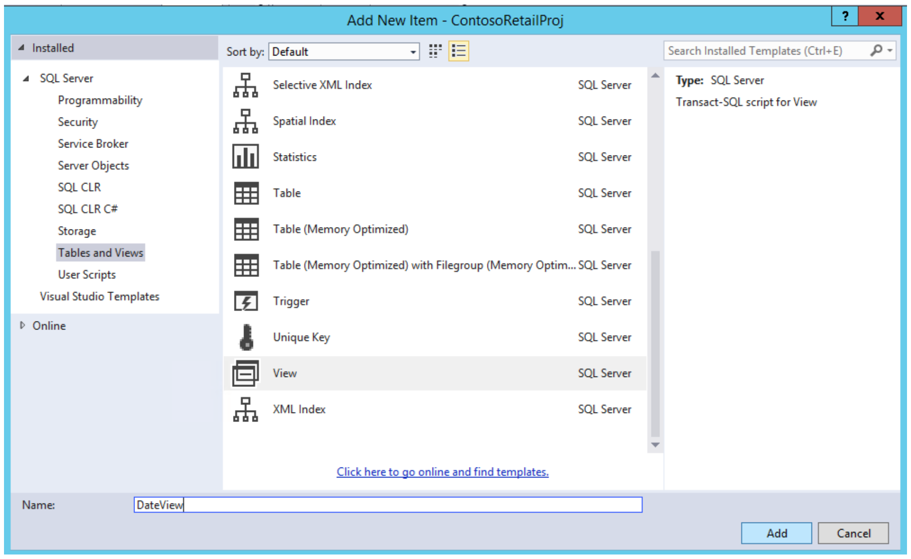
Either of these methods will cause the other view to automatically update.
By default, modifying the object definition automatically checks out file containing the definition from version control (creates a local copy of this file that you can change and then later check in to version control server if desired.)

5. On the **File** menu, click Save **DimCurrency.sql** to save the changes (on the local file.)

6. Add a new view by right clicking on the schema **dbo** in the Solution Explorer and clicking on **Add** and **View…**



7. Enter the view name **DateView** and click on **Add**

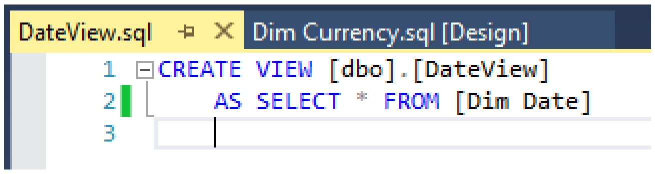


Type: SQL Server
Transact-SQL script for View

Name: DateView

8. This opens a sql script for Creating a new view

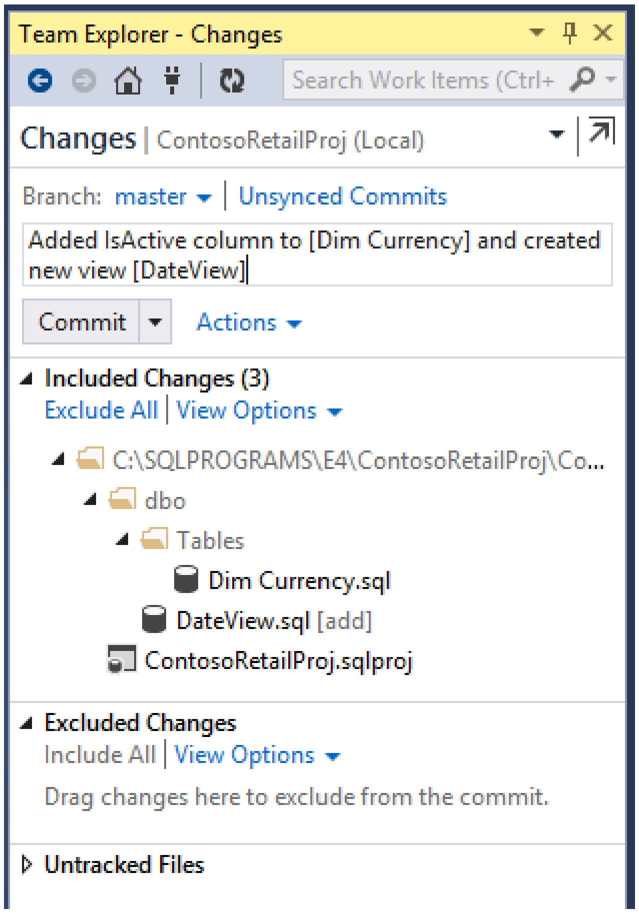
9. Replace the table name [SomeTableOrView] in the from clause with **[Dim Date]** and click on **Save** icon from the toolbar



```
CREATE VIEW [dbo].[DateView]
    AS SELECT * FROM [Dim Date]
```

10. Right-click on the project (ContosoRetailProj) in the Solution Explorer pane and select **Source Control > Commit…**

11. Enter a comment to describe the change (e.g. New column [IsActive] added to <your table> table and new view <your view> added)


Team Explorer - Changes

Search Work Items (Ctrl+

Changes | ContosoRetailProj (Local)

Branch: master | Unsynced Commits

Added IsActive column to [Dim Currency] and created new view [DateView]

Commit | Actions

Included Changes (3)

Exclude All | View Options

C:\SQLPROGRAMS\E4\ContosoRetailProj\Co...

dbo

Tables

Dim Currency.sql

DateView.sql [add]

ContosoRetailProj.sqlproj

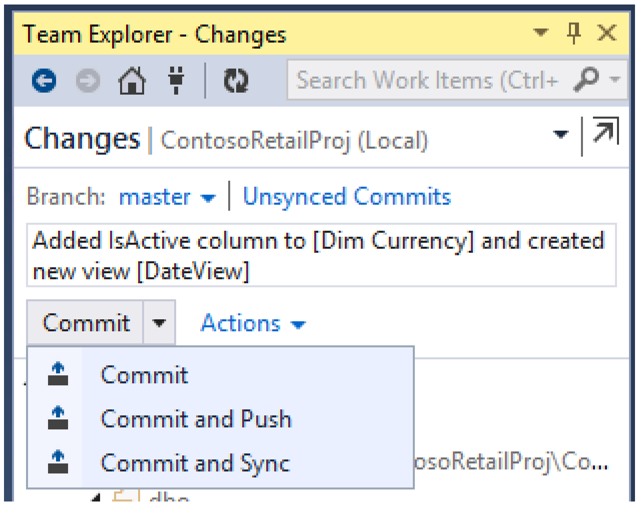
Excluded Changes

Include All | View Options

Drag changes here to exclude from the commit.

Untracked Files

12. Click on **Commit And Push** (drop down option from the **Commit** button) if using Git. Click **Yes** if prompted to confirm save.


Team Explorer - Changes

Search Work Items (Ctrl+

Changes | ContosoRetailProj (Local)

Branch: master | Unsynced Commits

Added IsActive column to [Dim Currency] and created new view [DateView]

Commit | Actions

Commit

Commit and Push

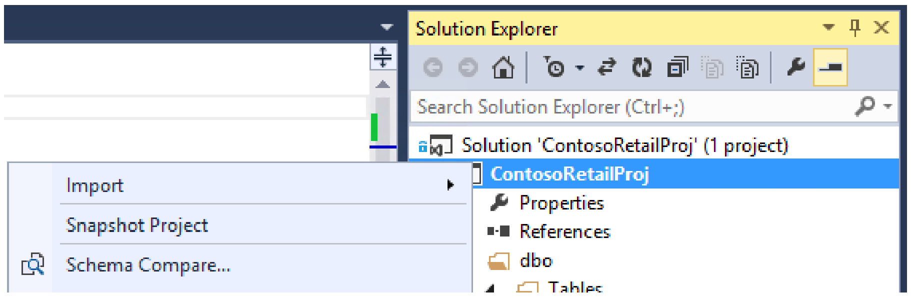
Commit and Sync

This saves a version of the database object which could be retrieved later from the Get repository. The changes checked in are also now available to other team members to view/use.

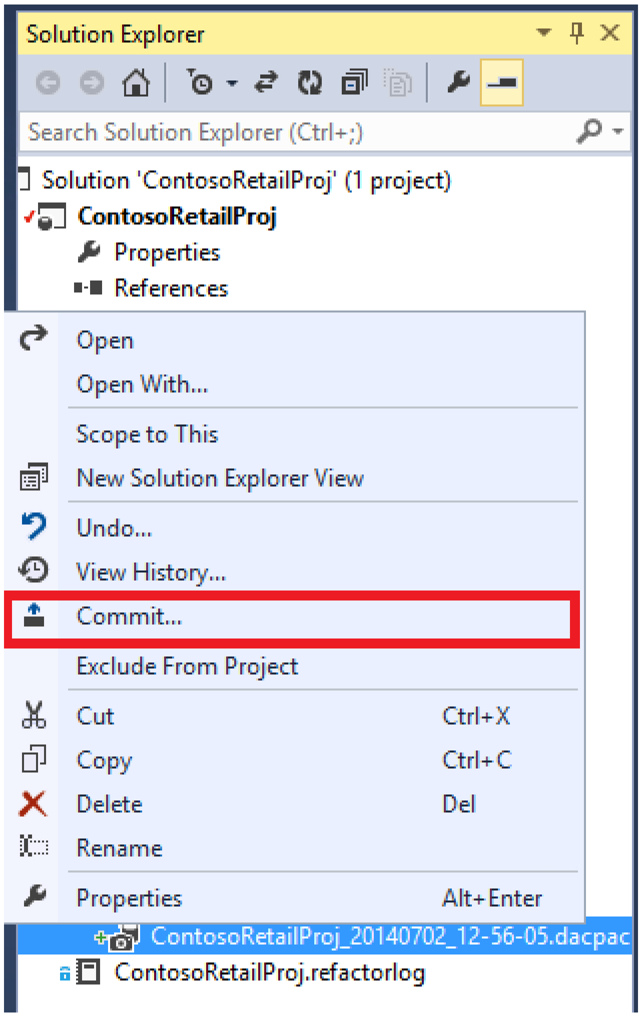
Create a database snapshot using Git

A Database snapshot captures a copy of the database project as it exists at the time you take the snapshot.

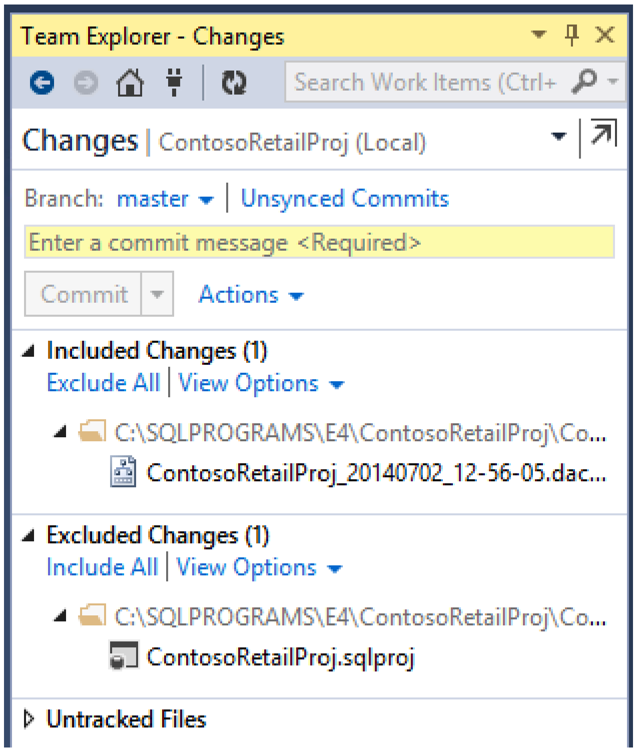
1. In Visual Studio **Solution Explorer**, right click on the project you want to create a snapshot for and choose **Snapshot Project**.



2. A **<your project>_<timestamp>.dacpac** file is created under a folder named **Snapshots** in the project



3. Check In the snapshot by right clicking on it and selecting **Commit** as after modifying database objects.

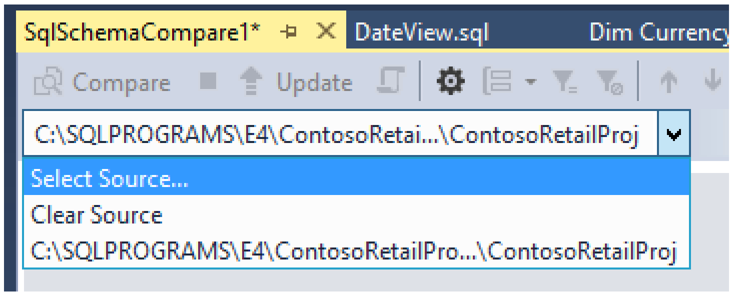

Team Explorer - Changes
Search Work Items (Ctrl+
Changes | ContosoRetailProj (Local)
Branch: master | Unsynced Commits
Enter a commit message <Required>
Commit
Actions
Included Changes (1)
Exclude All | View Options
C:\SQLPROGRAMS\E4\ContosoRetailProj\Co...
ContosoRetailProj_20140702_12-56-05.dac...
Excluded Changes (1)
Include All | View Options
C:\SQLPROGRAMS\E4\ContosoRetailProj\Co...
ContosoRetailProj.sqlproj
Untracked Files

The snapshot(s) created could be used to create new database project with the state of objects present at the time of creating the snapshot. These snapshots could also be used to compare the database schema created between two snapshots or live databases or projects. The compare schema also provides an option to generate the script to be used to move between two versions of the database.

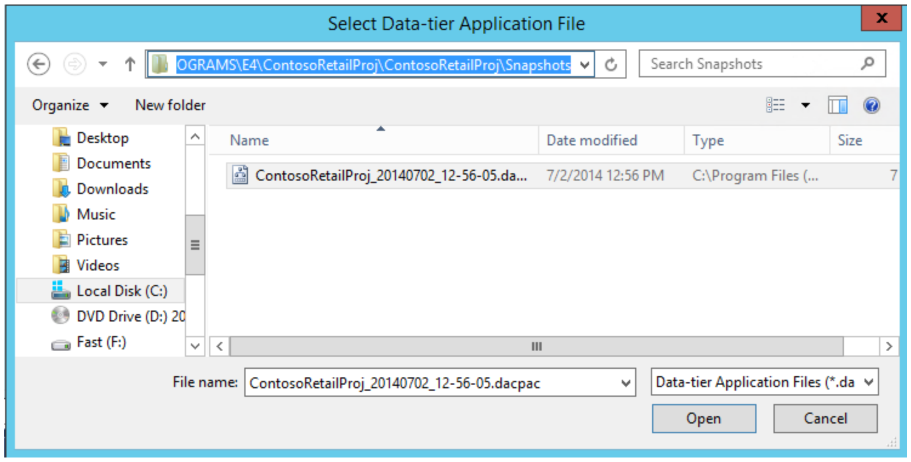
Compare Snapshots

1. Right-click on the project in the solution explorer and select **Schema Compare…**,

2. Once the **SqlSchemaCompare** tab has opened, click the drop-down arrow next to the source, choose **Select Source…**


SqlSchemaCompare1*
DateView.sql
Dim Currency
Compare
Update
C:\SQLPROGRAMS\E4\ContosoRetai...\ContosoRetailProj
Select Source...
Clear Source
C:\SQLPROGRAMS\E4\ContosoRetailPro...\ContosoRetailProj

3. In the **Select Data-tier Application File** dialog, navigate to the path **C:\SQLPROGRAMS\E4\ContosoRetailProj\ContosoRetailProj\Snapshots** directory and then select the **<your project>_<timestamp>.dacpac** file and click **Open**.


Select Data-tier Application File

OGRAMS\E4\ContosoRetailProj\ContosoRetailProj\Snapshots

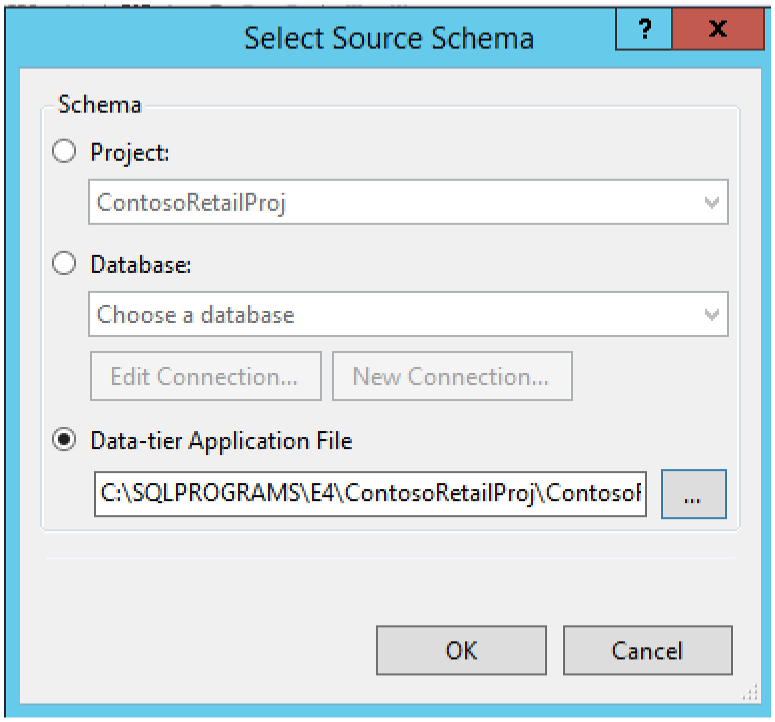
Name	Date modified	Type	Size
ContosoRetailProj_20140702_12-56-05.da...	7/2/2014 12:56 PM	C:\Program Files (...	7

File name: ContosoRetailProj_20140702_12-56-05.dacpac

Data-tier Application Files (*.da

4. In the **SqlSchemaCompare** tab, source is set to the snapshot selected.

5. Click **OK** for the **Select Source Schema** dialog.


Select Source Schema

Schema

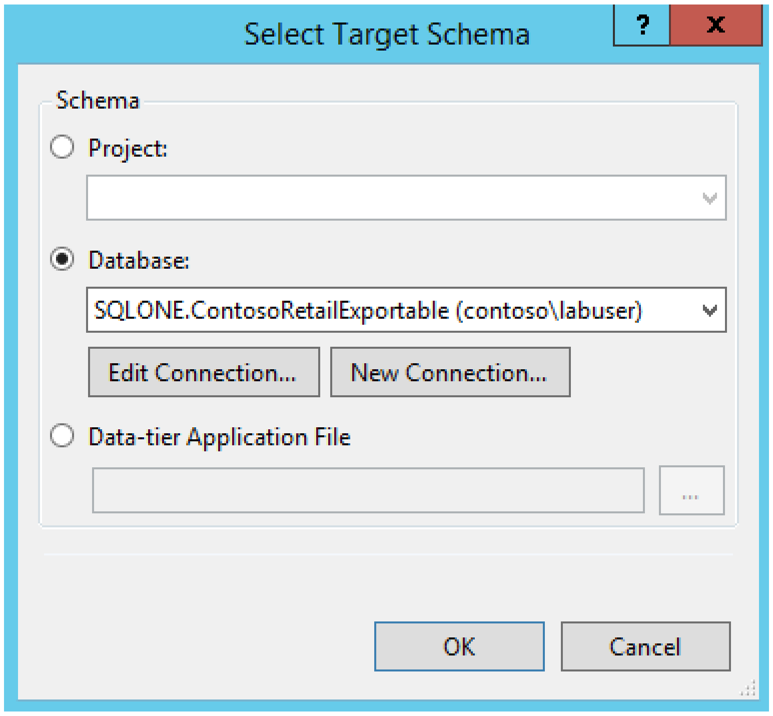
( ) Project: ContosoRetailProj

( ) Database: Choose a database

(●) Data-tier Application File: C:\SQLPROGRAMS\E4\ContosoRetailProj\Contosol

6. Click on **Select Target…** option from the target dropdown

7. Select the Database option and choose the **SQLONE.ContosoRetailExportable (contoso\labuser)** database from which this solution was created


Select Target Schema

Schema

Project:

Database: [✓]

SQLONE.ContosoRetailExportable (contoso\labuser)

Edit Connection... New Connection...

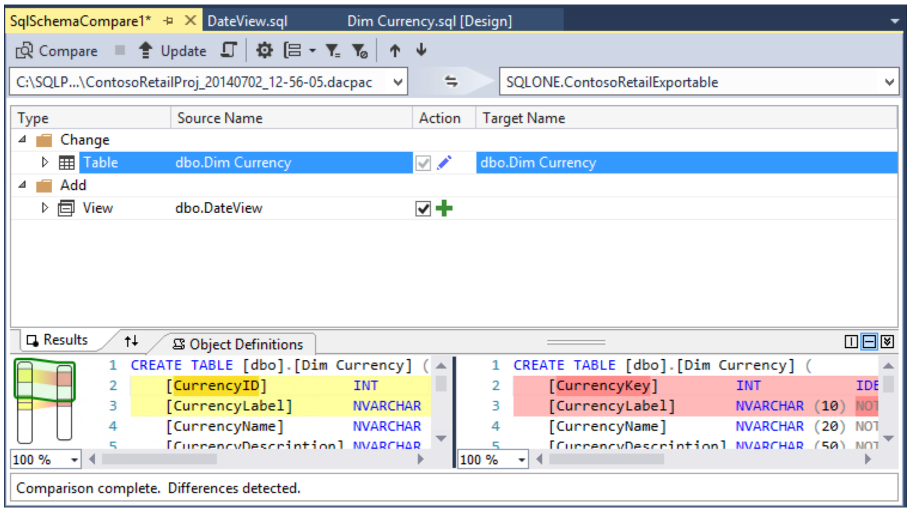
Data-tier Application File

OK Cancel

NOTE: You could also select the project or one of the previously created snapshots to compare the schema against

8. Click on **Compare** from the menu bar of the current tab

9. **Results** window shows the list of addition(s) and deletion(s) performed between the version and **Object Definitions** window highlights the changes between the version


SqlSchemaCompare1* DateView.sql Dim Currency.sql [Design]

Compare Update

C:\SQLP...\ContosoRetailProj_20140702_12-56-05.dacpac ⇆ SQLONE.ContosoRetailExportable

Type	Source Name	Action	Target Name
Change			
Table	dbo.Dim Currency	✓	dbo.Dim Currency
Add			
View	dbo.DateView	✓ +	

Results Object Definitions

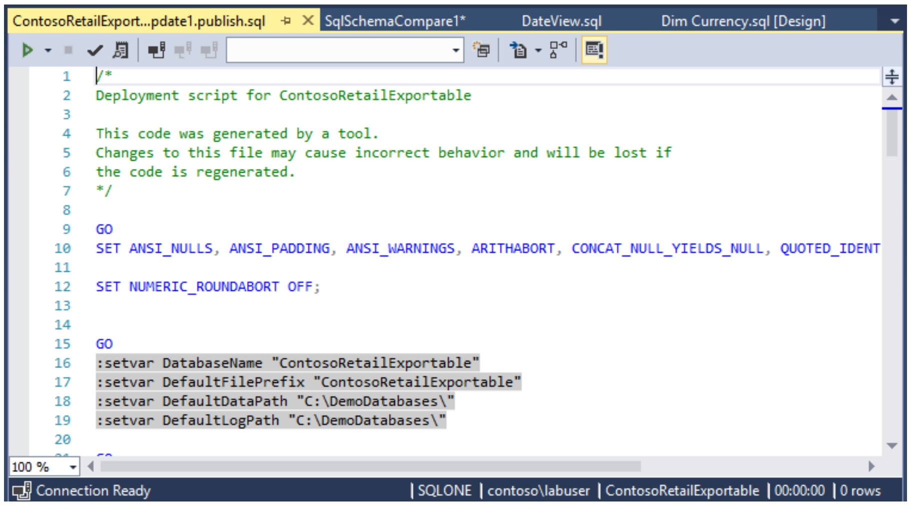
```
CREATE TABLE [dbo].[Dim Currency] (
    [CurrencyID]          INT
    [CurrencyLabel]       NVARCHAR
    [CurrencyName]        NVARCHAR
    [CurrencyDescription] NVARCHAR
```

```
CREATE TABLE [dbo].[Dim Currency] (
    [CurrencyKey]         INT             IDE
    [CurrencyLabel]       NVARCHAR (10)   NOT
    [CurrencyName]        NVARCHAR (20)   NOT
    [CurrencyDescription] NVARCHAR (50)   NOT
```

100 % 100 %

Comparison complete. Differences detected.

10. Click on in the menu bar (tooltip is **Generate Script)** from the menu to create the script to run against the source database to get to the target state
Do not run this script, as we do not want to change the source database in this case.



```
/*
Deployment script for ContosoRetailExportable

This code was generated by a tool.
Changes to this file may cause incorrect behavior and will be lost if
the code is regenerated.
*/

GO
SET ANSI_NULLS, ANSI_PADDING, ANSI_WARNINGS, ARITHABORT, CONCAT_NULL_YIELDS_NULL, QUOTED_IDENT

SET NUMERIC_ROUNDABORT OFF;


GO
:setvar DatabaseName "ContosoRetailExportable"
:setvar DefaultFilePrefix "ContosoRetailExportable"
:setvar DefaultDataPath "C:\DemoDatabases\"
:setvar DefaultLogPath "C:\DemoDatabases\"

```

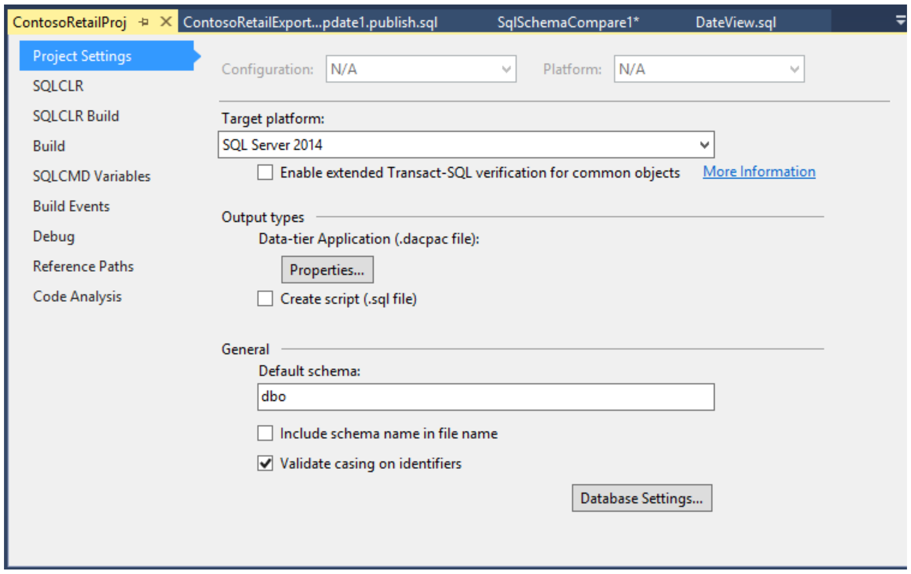
This feature enables you to browse through the schema differences visually before deployment. It also helps you in generating the delta SQL script to move between versions (if needed).

Preparing to Publish the Database Project to a Live Database

Before building/publishing the changes to any server, let us look at some of Project Properties which can be used to make Publishing to different environments quite simple.

1. Right click on the project and select **Properties** from the Solution Explorer

2. In the **Project Settings** tab, you could set the version and type of SQL server you want to publish the current solution to, which could be SQL Server 2005/2008/2012/2014 or Windows Azure SQL Database. Here, verify the Target platform is **SQL Server 2014**


ContosoRetailProj | ContosoRetailExport...pdate1.publish.sql | SqlSchemaCompare1* | DateView.sql

Project Settings
SQLCLR
SQLCLR Build
Build
SQLCMD Variables
Build Events
Debug
Reference Paths
Code Analysis

Configuration: N/A Platform: N/A

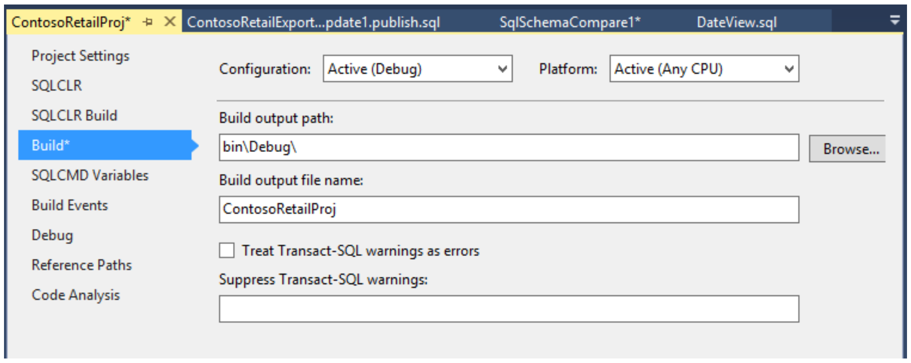
Target platform:
SQL Server 2014
☐ Enable extended Transact-SQL verification for common objects More Information

Output types
Data-tier Application (.dacpac file):
Properties...
☐ Create script (.sql file)

General
Default schema:
dbo
☐ Include schema name in file name
☑ Validate casing on identifiers
Database Settings...

NOTE: When one of the SQL servers are selected, errors specific to that version of the SQL server are highlighted during Build

3. In the **Build** tab, you could check the Treat Transact-SQL warnings as errors option. This would highlight the Transact-SQL warnings as error while building the project. **Do not check this option here.**


ContosoRetailProj* | ContosoRetailExport...pdate1.publish.sql | SqlSchemaCompare1* | DateView.sql

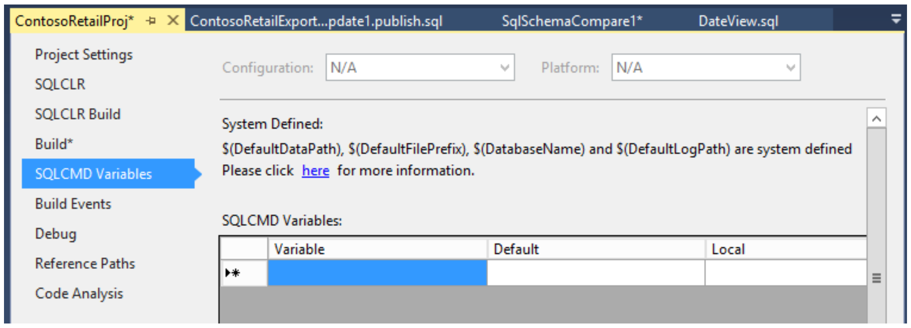
Project Settings
SQLCLR
SQLCLR Build
Build*
SQLCMD Variables
Build Events
Debug
Reference Paths
Code Analysis

Configuration: Active (Debug) Platform: Active (Any CPU)

Build output path:
bin\Debug\
Browse...
Build output file name:
ContosoRetailProj
☐ Treat Transact-SQL warnings as errors
Suppress Transact-SQL warnings:

NOTE: You could also suppress any Transact-SQL warnings you do not want to be shown as errors

4. In the **SQLCMD Variables** tab, you could define the values for system defined variables and/or user defined variables like $DatabaseName, $DefaultDataPath, etc. These variable values could be used to provide dynamic substitution for debugging or publishing to various environments and servers. Do not change anything on this page now.

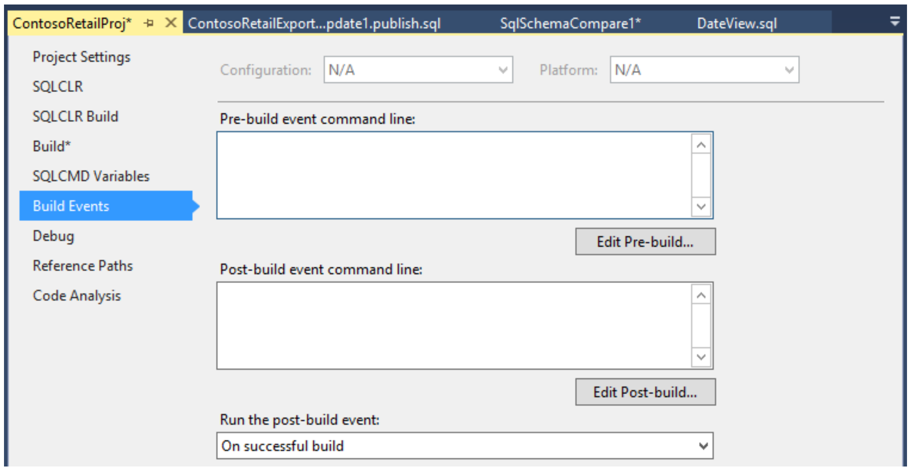


Configuration: N/A Platform: N/A

System Defined:
$(DefaultDataPath), $(DefaultFilePrefix), $(DatabaseName) and $(DefaultLogPath) are system defined
Please click here for more information.

SQLCMD Variables:

5. **Build Events** tab could be used to specify a command line to execute before the build operation starts and a command line to execute after the build operation has completed. Do not alter this settings page now.



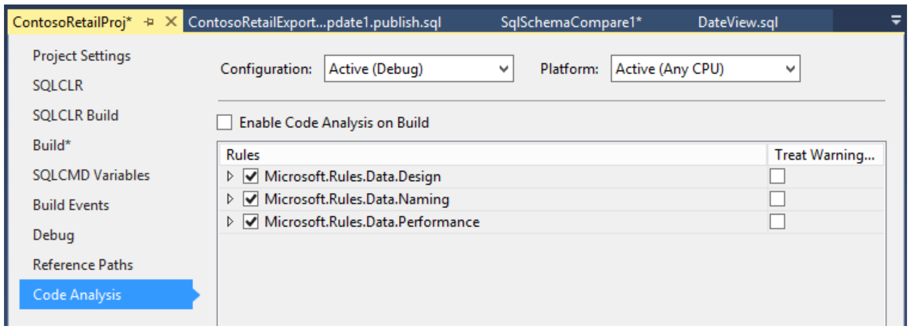
Configuration: N/A Platform: N/A

Pre-build event command line:

Post-build event command line:

Run the post-build event: On successful build

6. You can use options from **Code Analysis** tab to discover potential issues in your scripts, such as design, naming and performance problems. Rules for database projects are organized into predefined rule sets that target specific areas, and you can enable or disable any of the rules available.



Configuration: Active (Debug) Platform: Active (Any CPU)

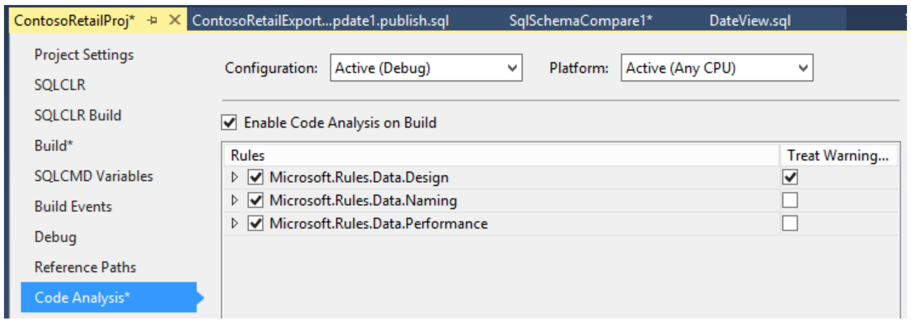
☐ Enable Code Analysis on Build

Rules	Treat Warning...
✓ Microsoft.Rules.Data.Design	☐
✓ Microsoft.Rules.Data.Naming	☐
✓ Microsoft.Rules.Data.Performance	☐

Rebuild and debug issues

7. In the **Code Analysis** tab of the project properties, check the **Enable Code Analysis on Build** option

8. Check **Treat Warning as Error** check box against Microsoft.Rules.Data.Design entry



Configuration: Active (Debug) Platform: Active (Any CPU)

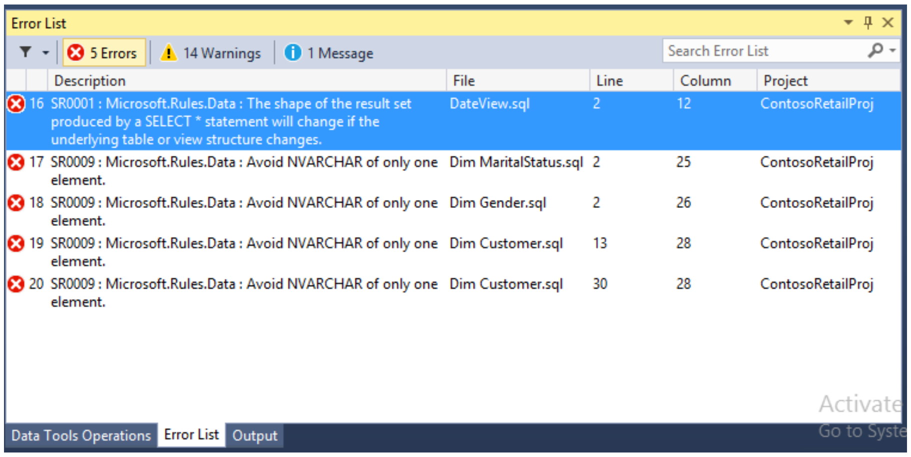
✓ Enable Code Analysis on Build

Rules	Treat Warning...
✓ Microsoft.Rules.Data.Design	✓
✓ Microsoft.Rules.Data.Naming	☐
✓ Microsoft.Rules.Data.Performance	☐

9. Click 💾 in the menu bar to save the changes

10. Right click on your project from the **Solution Explorer** pane and click on **Rebuild**

11. Go to Error List pane to view the list of errors

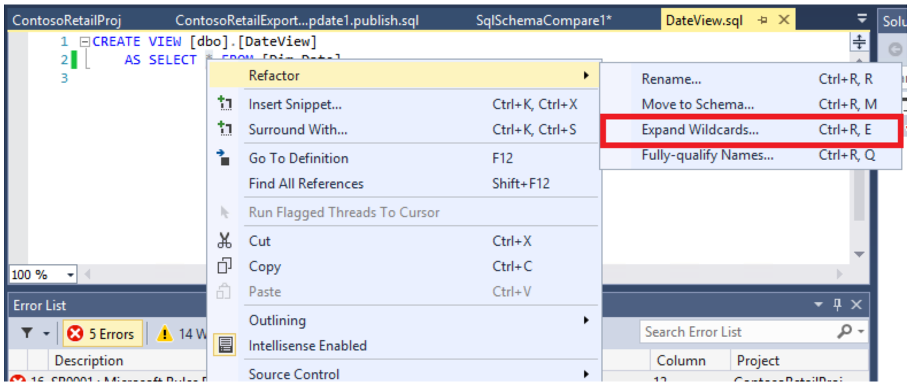


	Description	File	Line	Column	Project
16	SR0001 : Microsoft.Rules.Data : The shape of the result set produced by a SELECT * statement will change if the underlying table or view structure changes.	DateView.sql	2	12	ContosoRetailProj
17	SR0009 : Microsoft.Rules.Data : Avoid NVARCHAR of only one element.	Dim MaritalStatus.sql	2	25	ContosoRetailProj
18	SR0009 : Microsoft.Rules.Data : Avoid NVARCHAR of only one element.	Dim Gender.sql	2	26	ContosoRetailProj
19	SR0009 : Microsoft.Rules.Data : Avoid NVARCHAR of only one element.	Dim Customer.sql	13	28	ContosoRetailProj
20	SR0009 : Microsoft.Rules.Data : Avoid NVARCHAR of only one element.	Dim Customer.sql	30	28	ContosoRetailProj

12. In the **Error List** pane, double click on the error with the prefix "SR0001 : Microsoft.Rules.Data :"

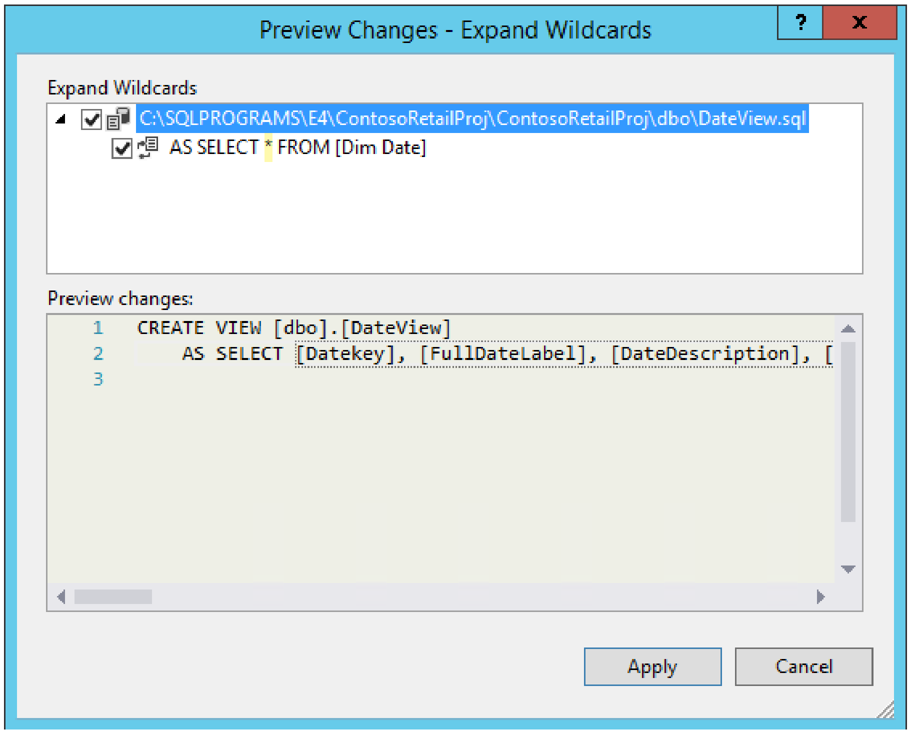
This will take you to the view definition window with asterisks (*) highlighted.

13. Rright click on asterisks and select **Refactor** and **Expand Wildcards…** option.


ContosoRetailProj
ContosoRetailExport...pdate1.publish.sql
SqlSchemaCompare1*
DateView.sql
Refactor
Insert Snippet... Ctrl+K, Ctrl+X
Surround With... Ctrl+K, Ctrl+S
Go To Definition F12
Find All References Shift+F12
Run Flagged Threads To Cursor
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Outlining
Intellisense Enabled
Source Control
Rename... Ctrl+R, R
Move to Schema... Ctrl+R, M
Expand Wildcards... Ctrl+R, E
Fully-qualify Names... Ctrl+R, Q
Error List
5 Errors
Search Error List
Description
Column
Project

```
CREATE VIEW [dbo].[DateView]
    AS SELECT * FROM [Dim Date]
```

14. In the preview changes window, click on **Apply**.

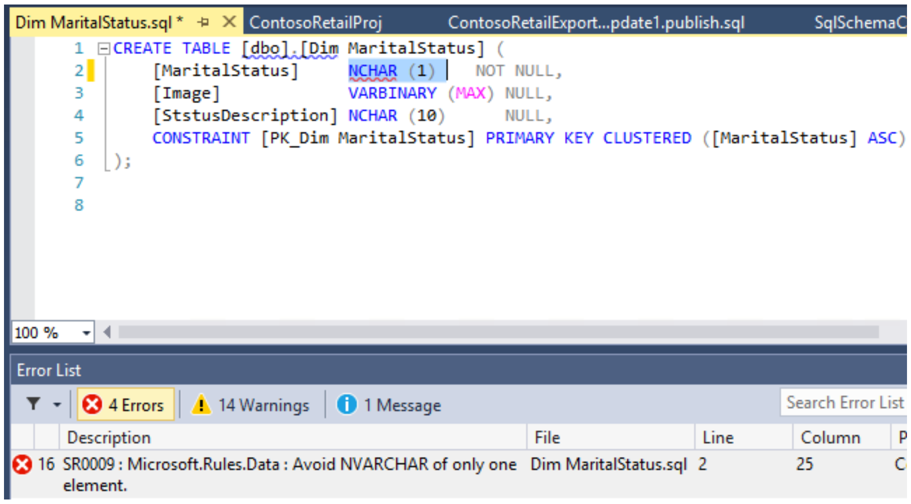

Preview Changes - Expand Wildcards
Expand Wildcards
C:\SQLPROGRAMS\E4\ContosoRetailProj\ContosoRetailProj\dbo\DateView.sql
AS SELECT * FROM [Dim Date]
Preview changes:
Apply
Cancel

```
CREATE VIEW [dbo].[DateView]
    AS SELECT [Datekey], [FullDateLabel], [DateDescription], [
```

15. Click on **Save** from the tool bar to save the changes made to the View

16. Right click on your project from the **Solution Explorer** pane and click on **Rebuild**

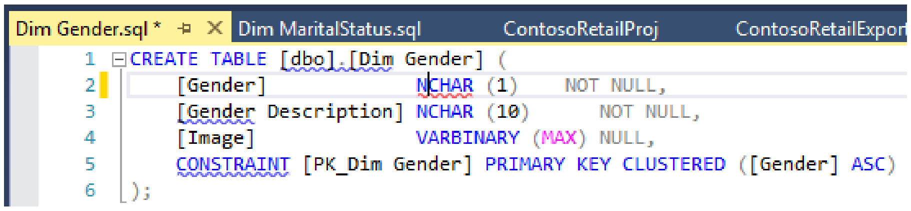
17. If there are any other build errors, correct the errors before proceeding, eg by changing NVARCHAR(1) to NCHAR(1) in necessary table definitions



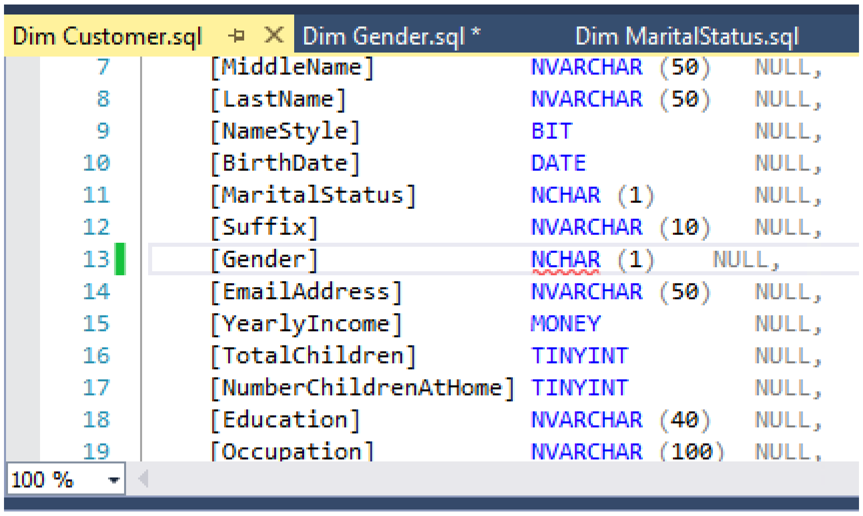
```sql
CREATE TABLE [dbo].[Dim MaritalStatus] (
    [MaritalStatus]     NCHAR (1)    NOT NULL,
    [Image]             VARBINARY (MAX) NULL,
    [StstusDescription] NCHAR (10)      NULL,
    CONSTRAINT [PK_Dim MaritalStatus] PRIMARY KEY CLUSTERED ([MaritalStatus] ASC)
);
```

4 Errors | 14 Warnings | 1 Message

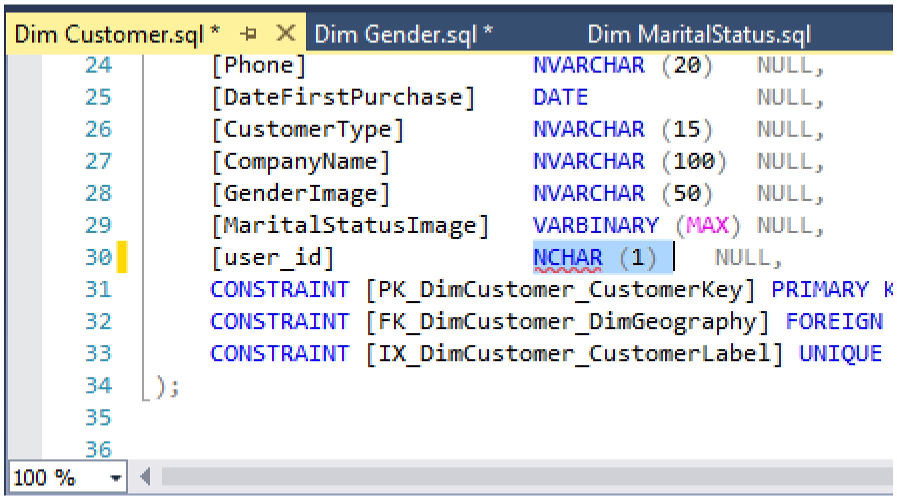
	Description	File	Line	Column	P
16	SR0009 : Microsoft.Rules.Data : Avoid NVARCHAR of only one element.	Dim MaritalStatus.sql	2	25	C



```sql
CREATE TABLE [dbo].[Dim Gender] (
    [Gender]             NCHAR (1)    NOT NULL,
    [Gender Description] NCHAR (10)   NOT NULL,
    [Image]              VARBINARY (MAX) NULL,
    CONSTRAINT [PK_Dim Gender] PRIMARY KEY CLUSTERED ([Gender] ASC)
);
```



```sql
    [MiddleName]           NVARCHAR (50)   NULL,
    [LastName]             NVARCHAR (50)   NULL,
    [NameStyle]            BIT             NULL,
    [BirthDate]            DATE            NULL,
    [MaritalStatus]        NCHAR (1)       NULL,
    [Suffix]               NVARCHAR (10)   NULL,
    [Gender]               NCHAR (1)    NULL,
    [EmailAddress]         NVARCHAR (50)   NULL,
    [YearlyIncome]         MONEY           NULL,
    [TotalChildren]        TINYINT         NULL,
    [NumberChildrenAtHome] TINYINT         NULL,
    [Education]            NVARCHAR (40)   NULL,
    [Occupation]           NVARCHAR (100)  NULL,
```

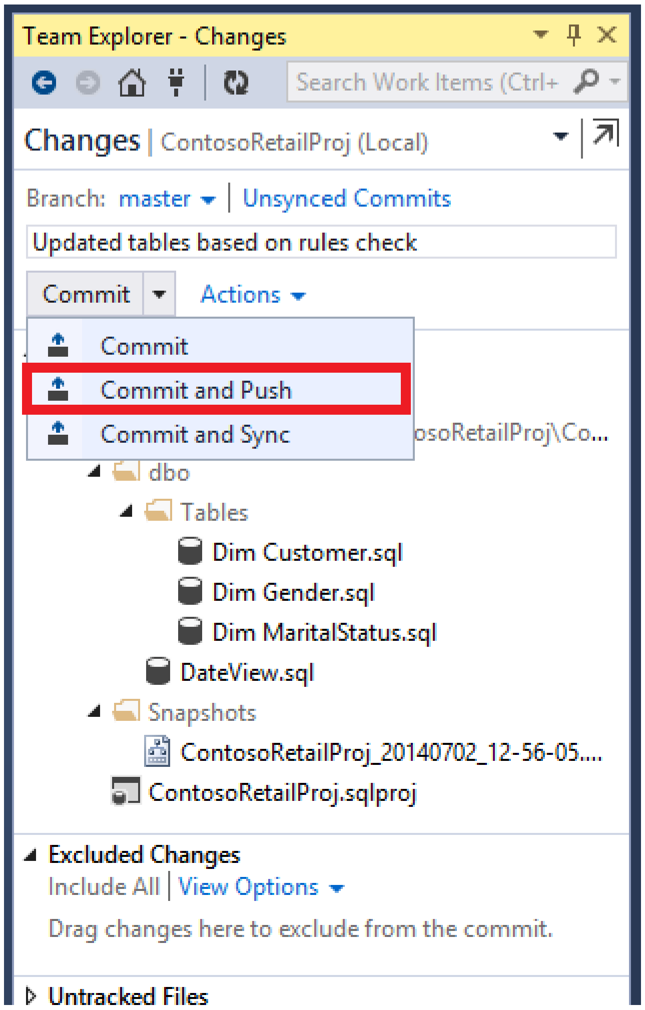


```sql
    [Phone]                NVARCHAR (20)   NULL,
    [DateFirstPurchase]    DATE            NULL,
    [CustomerType]         NVARCHAR (15)   NULL,
    [CompanyName]          NVARCHAR (100)  NULL,
    [GenderImage]          NVARCHAR (50)   NULL,
    [MaritalStatusImage]   VARBINARY (MAX) NULL,
    [user_id]              NCHAR (1)    NULL,
    CONSTRAINT [PK_DimCustomer_CustomerKey] PRIMARY K
    CONSTRAINT [FK_DimCustomer_DimGeography] FOREIGN
    CONSTRAINT [IX_DimCustomer_CustomerLabel] UNIQUE
);
```

NOTE: To deploy a database into Window Azure, all tables present in the database should have a clustered index. Error SQL71560 (Table does not have a clustered index. Clustered indexes are required for inserting data in this version of SQL Server) could be eliminated by creating a clustered index for the table specified.

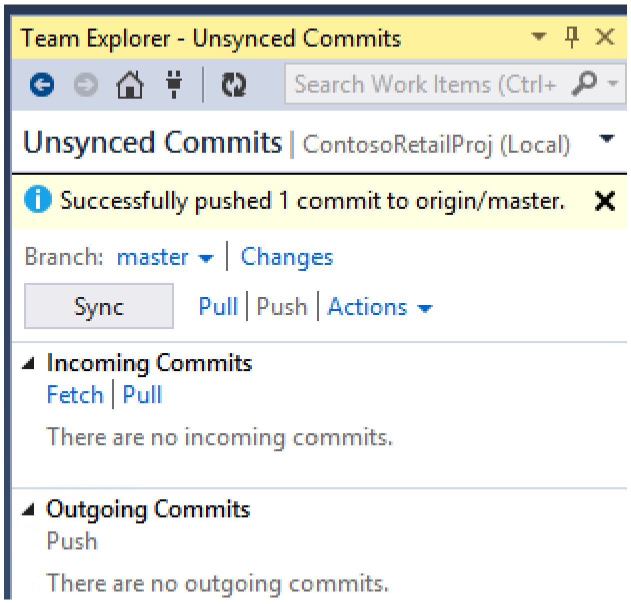
Commit changes to Git

1. Right click on the project and click on **Commit**.
2. Enter a comment and click on **Commit and Push**.



Updated tables based on rules check

3. Verify the commit action was successful.


Team Explorer - Unsynced Commits

Unsynced Commits | ContosoRetailProj (Local)

Successfully pushed 1 commit to origin/master.

Branch: master | Changes

Incoming Commits

There are no incoming commits.

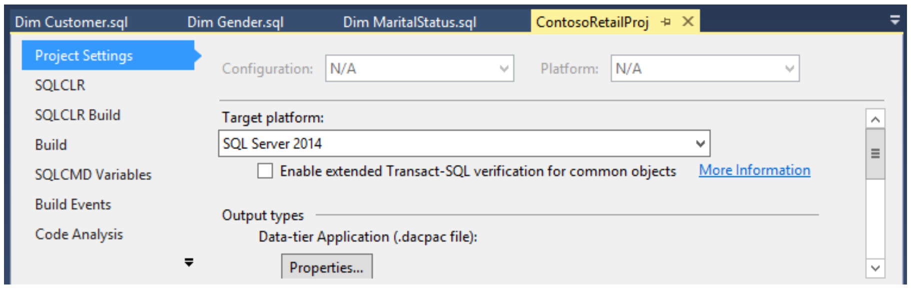
Outgoing Commits

There are no outgoing commits.

Publish the Database to SQL Server 2014 instance

1. Right click on the project and select **Properties**

2. Go to **Project Settings** tab and ensure that the **Target platform** is set to **SQL Server 2014**



Configuration: N/A Platform: N/A

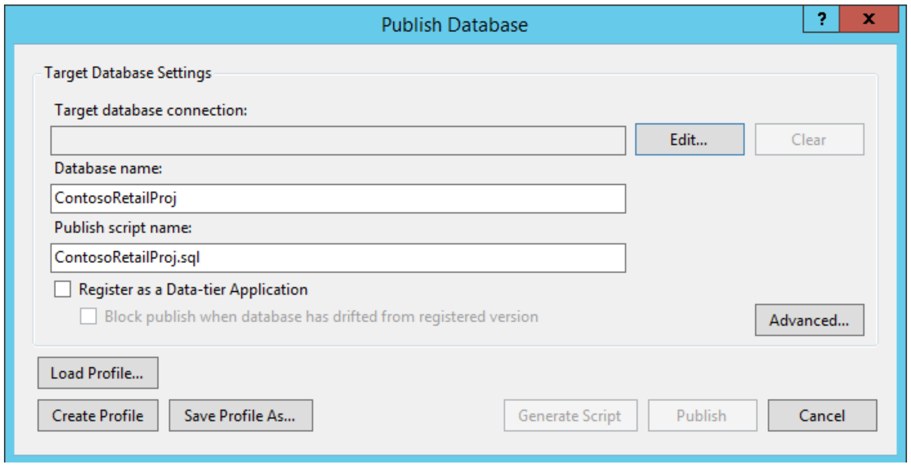
Target platform: SQL Server 2014

Enable extended Transact-SQL verification for common objects

Output types

Data-tier Application (.dacpac file):

3. Right click on the Project from the **Solution Explorer** pane and click on **Publish...**


Publish Database

Target Database Settings

Target database connection:

Database name: ContosoRetailProj

Publish script name: ContosoRetailProj.sql

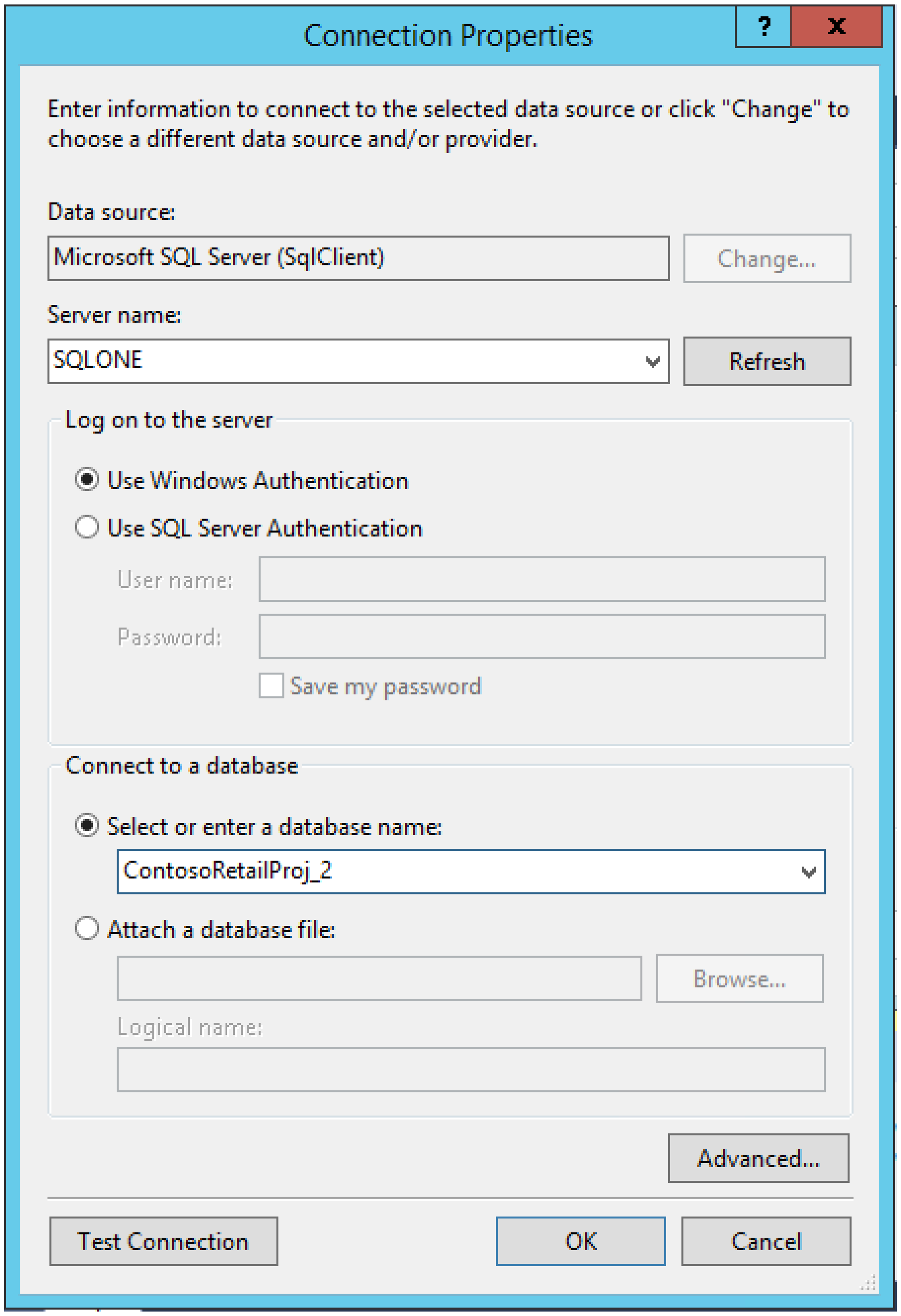
Register as a Data-tier Application

Block publish when database has drifted from registered version

4. Click on **Edit...** to open the **Connection Properties** dialog box

5. Enter **SQLONE** as the Server name, select **Use Windows Authentication**

6. Enter **ContosoRetailProj_2** as the database name
   We are deploying to a new database here. You could alternatively deploy to an existing database



Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: SQLONE

Log on to the server

● Use Windows Authentication

○ Use SQL Server Authentication

User name:

Password:

☐ Save my password

Connect to a database

● Select or enter a database name: ContosoRetailProj_2

○ Attach a database file:

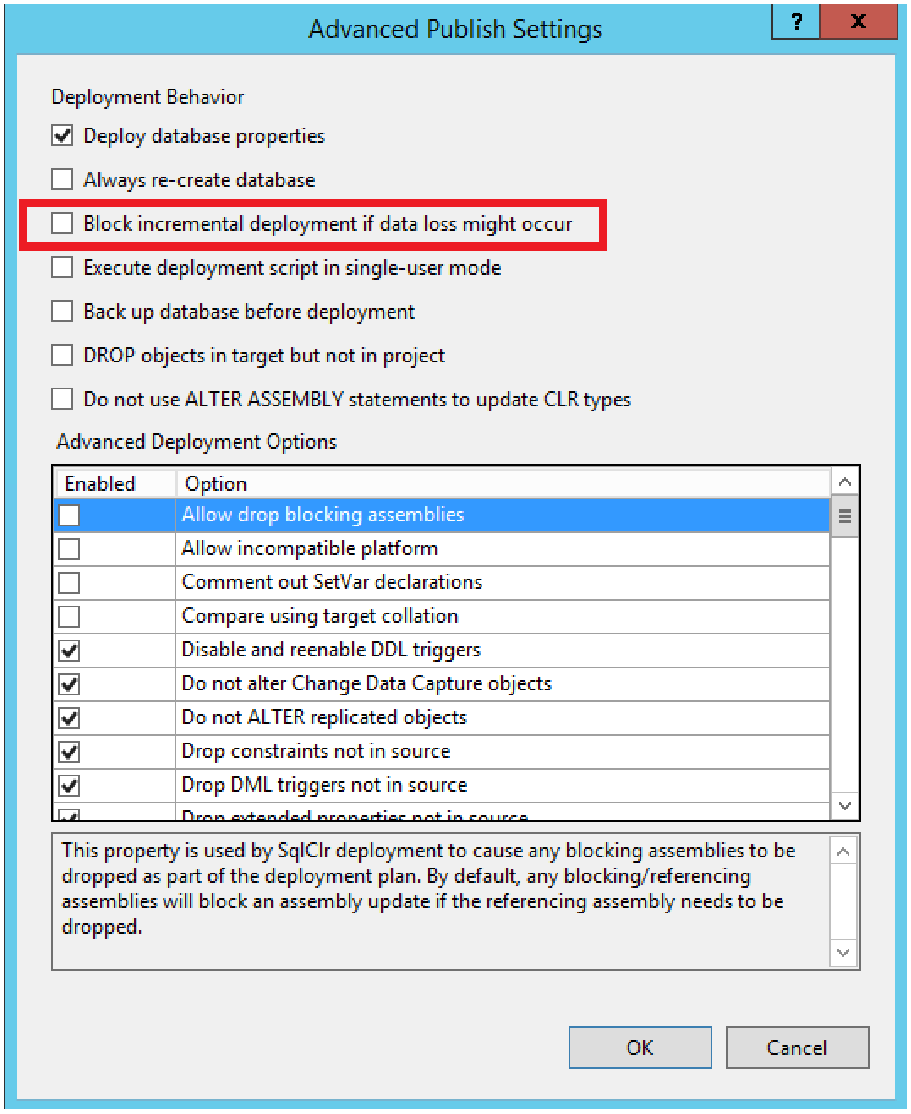
Logical name:

7. Click **OK**

8. If needed, alter the publish script name in the text box below

9. Click on **Advanced...** button to set other Publishing properties available

*NOTE: One of the options which is checked by default is **Block incremental deployment if data loss might occur**. When this option is checked, any action which could cause a possible data loss in the target database will be stopped and the deployment will fail. If you are sure about the data loss and want the change(s) to be deployed, uncheck this option and click on **OK**.*



Deployment Behavior

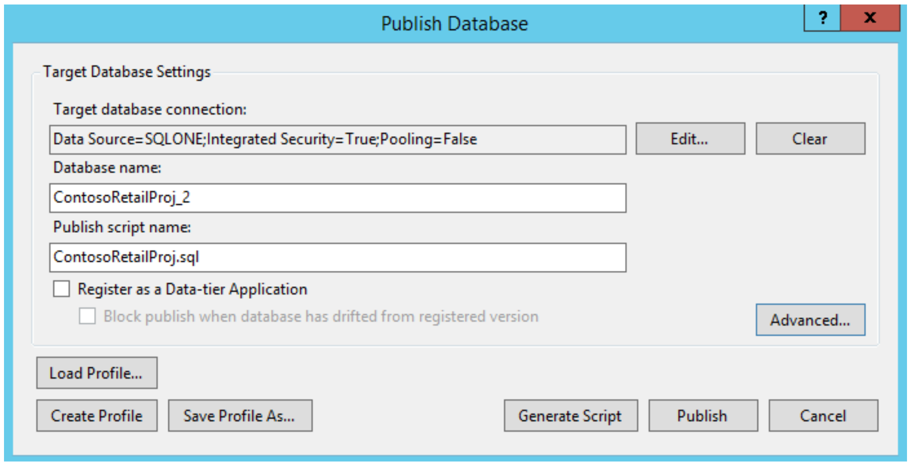
- [✓] Deploy database properties
- [ ] Always re-create database
- [ ] Block incremental deployment if data loss might occur
- [ ] Execute deployment script in single-user mode
- [ ] Back up database before deployment
- [ ] DROP objects in target but not in project
- [ ] Do not use ALTER ASSEMBLY statements to update CLR types

Advanced Deployment Options

Enabled	Option
☐	Allow drop blocking assemblies
☐	Allow incompatible platform
☐	Comment out SetVar declarations
☐	Compare using target collation
✓	Disable and reenable DDL triggers
✓	Do not alter Change Data Capture objects
✓	Do not ALTER replicated objects
✓	Drop constraints not in source
✓	Drop DML triggers not in source
✓	Drop extended properties not in source

This property is used by SqlClr deployment to cause any blocking assemblies to be dropped as part of the deployment plan. By default, any blocking/referencing assemblies will block an assembly update if the referencing assembly needs to be dropped.

10. Click on **Save Profile As** button to save the settings selected (including the options selected in the advanced publish settings)


Publish Database

Target Database Settings

Target database connection: Data Source=SQLONE;Integrated Security=True;Pooling=False

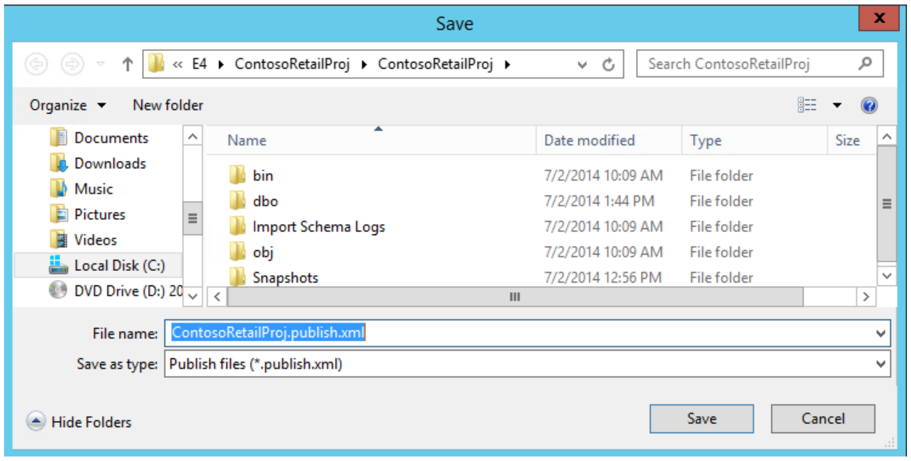
Database name: ContosoRetailProj_2

Publish script name: ContosoRetailProj.sql

Register as a Data-tier Application

Block publish when database has drifted from registered version

11. Browse to a desired location, provide a profile name and click on **Save**


Save

« E4 ▸ ContosoRetailProj ▸ ContosoRetailProj ▸

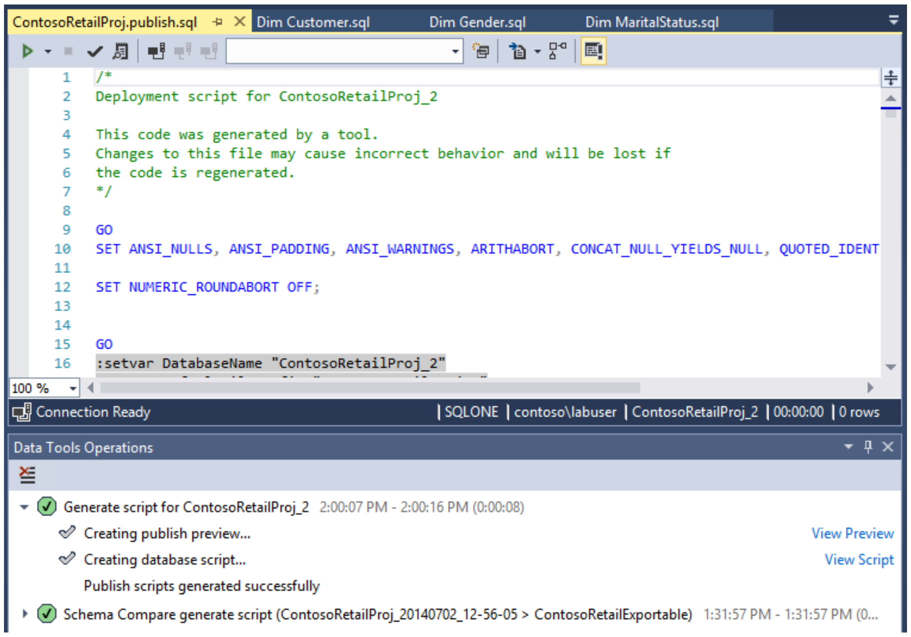
Name	Date modified	Type	Size
bin	7/2/2014 10:09 AM	File folder	
dbo	7/2/2014 1:44 PM	File folder	
Import Schema Logs	7/2/2014 10:09 AM	File folder	
obj	7/2/2014 10:09 AM	File folder	
Snapshots	7/2/2014 12:56 PM	File folder	

File name: ContosoRetailProj.publish.xml

Save as type: Publish files (*.publish.xml)

12. Click on **Generate Script** to generate a SQL script to publish the changes on the desired target server

This also helps in generating the deployment scripts which could be re-used for deploying to different environments



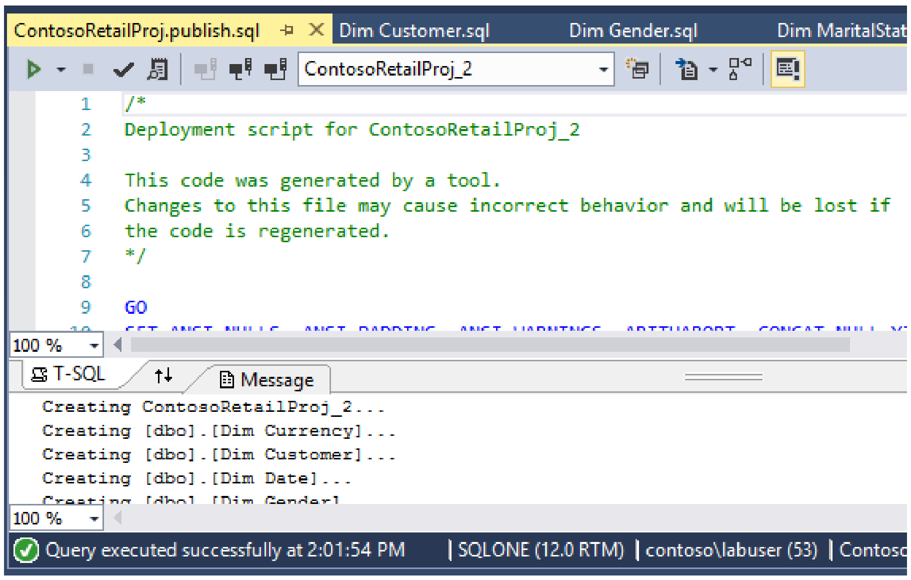
```
/*
Deployment script for ContosoRetailProj_2

This code was generated by a tool.
Changes to this file may cause incorrect behavior and will be lost if
the code is regenerated.
*/

GO
SET ANSI_NULLS, ANSI_PADDING, ANSI_WARNINGS, ARITHABORT, CONCAT_NULL_YIELDS_NULL, QUOTED_IDENT

SET NUMERIC_ROUNDABORT OFF;


GO
:setvar DatabaseName "ContosoRetailProj_2"
```

Connection Ready | SQLONE | contoso\labuser | ContosoRetailProj_2 | 00:00:00 | 0 rows

Data Tools Operations

```
Generate script for ContosoRetailProj_2   2:00:07 PM - 2:00:16 PM (0:00:08)
    Creating publish preview...                                          View Preview
    Creating database script...                                          View Script
    Publish scripts generated successfully
Schema Compare generate script (ContosoRetailProj_20140702_12-56-05 > ContosoRetailExportable)   1:31:57 PM - 1:31:57 PM (0...
```

13. Click on ▷ in the menu bar to publish the changes to target server

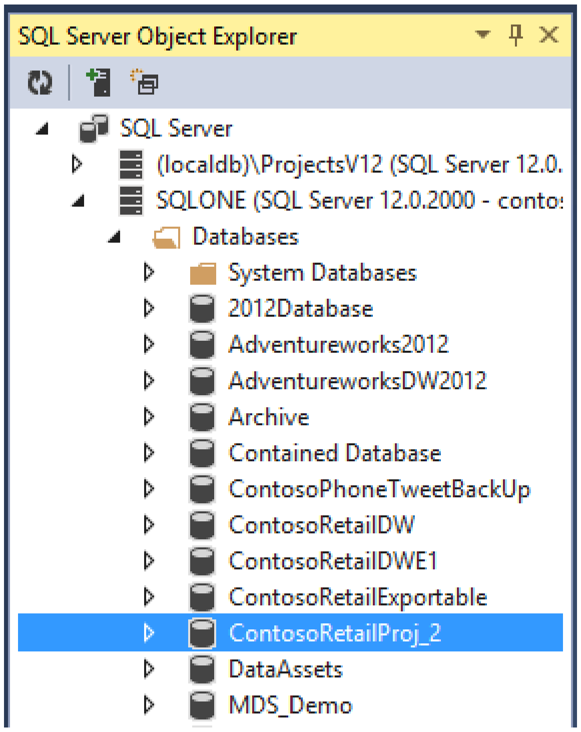


```
/*
Deployment script for ContosoRetailProj_2

This code was generated by a tool.
Changes to this file may cause incorrect behavior and will be lost if
the code is regenerated.
*/

GO
```

```
Creating ContosoRetailProj_2...
Creating [dbo].[Dim Currency]...
Creating [dbo].[Dim Customer]...
Creating [dbo].[Dim Date]...
Creating [dbo].[Dim Gender]
```

Query executed successfully at 2:01:54 PM | SQLONE (12.0 RTM) | contoso\labuser (53) | Contoso

14. To verify the changes, open the **SQL Server Object Explorer** pane (from the **menu** select **View** then **SQL Server Object Explorer**), expand **SQL Server** then **SQLONE** then **Databases** and you should see **ContosoRetailProj_2**



SQL Server Object Explorer

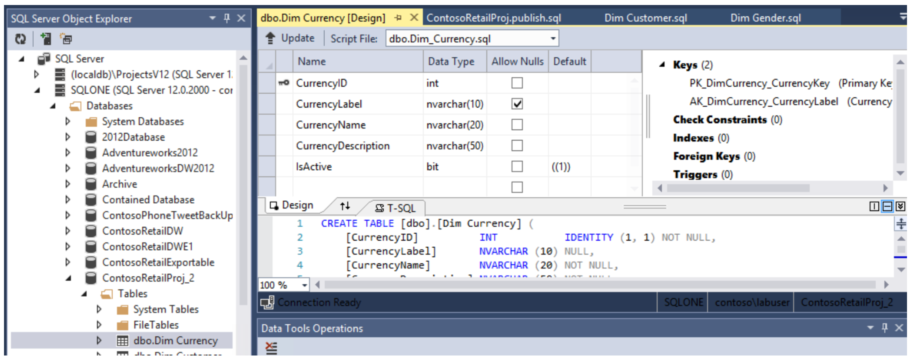
- SQL Server
  - (localdb)\ProjectsV12 (SQL Server 12.0.
  - SQLONE (SQL Server 12.0.2000 - conto:
    - Databases
      - System Databases
      - 2012Database
      - Adventureworks2012
      - AdventureworksDW2012
      - Archive
      - Contained Database
      - ContosoPhoneTweetBackUp
      - ContosoRetailDW
      - ContosoRetailDWE1
      - ContosoRetailExportable
      - ContosoRetailProj_2
      - DataAssets
      - MDS_Demo

NOTE: If the database is not shown then press the ***refresh*** *icon* *you will need to repeat the expand steps above*

15. Expand the **Server** and the **Database** folder

16. Locate the database to which the changes were deployed and expand ContosoRetailProj_2 followed by the database object folder (e.g. Tables, View, etc.)

17. Double-click on any database object (eg the dbo.Dim Currency table) to open it in design view and see



SQL Server Object Explorer

- SQL Server
  - (localdb)\ProjectsV12 (SQL Server 1
  - SQLONE (SQL Server 12.0.2000 - cor
    - Databases
      - System Databases
      - 2012Database
      - Adventureworks2012
      - AdventureworksDW2012
      - Archive
      - Contained Database
      - ContosoPhoneTweetBackUp
      - ContosoRetailDW
      - ContosoRetailDWE1
      - ContosoRetailExportable
      - ContosoRetailProj_2
        - Tables
          - System Tables
          - FileTables
          - dbo.Dim Currency

dbo.Dim Currency [Design] | ContosoRetailProj.publish.sql | Dim Customer.sql | Dim Gender.sql

Update | Script File: dbo.Dim_Currency.sql

Name	Data Type	Allow Nulls	Default
CurrencyID	int	☐	
CurrencyLabel	nvarchar(10)	✓	
CurrencyName	nvarchar(20)	☐	
CurrencyDescription	nvarchar(50)	☐	
IsActive	bit	☐	((1))
		☐	

- **Keys** (2)
  - PK_DimCurrency_CurrencyKey (Primary Ke
  - AK_DimCurrency_CurrencyLabel (Currency
- **Check Constraints** (0)
- **Indexes** (0)
- **Foreign Keys** (0)
- **Triggers** (0)

Design | T-SQL

```
CREATE TABLE [dbo].[Dim Currency] (
    [CurrencyID]          INT            IDENTITY (1, 1) NOT NULL,
    [CurrencyLabel]       NVARCHAR (10) NULL,
    [CurrencyName]        NVARCHAR (20) NOT NULL,
```

100 %

Connection Ready | SQLONE | contoso\labuser | ContosoRetailProj_2

Data Tools Operations

Visual Studio 2013 gives you a single interface to manage the end to end development of an application. It provides the ability the have a fully managed development process and source safe repository. Also helps you in managing the entire life cycle in a better and efficient way.

This feature allows DBA's to work more effectively and have a source safe repository for all their code, code changes and model changes allowing them to easily develop solutions and enhancements and publish these changes from a single toolset. This makes collaboration more effective and saves the team time and effort.

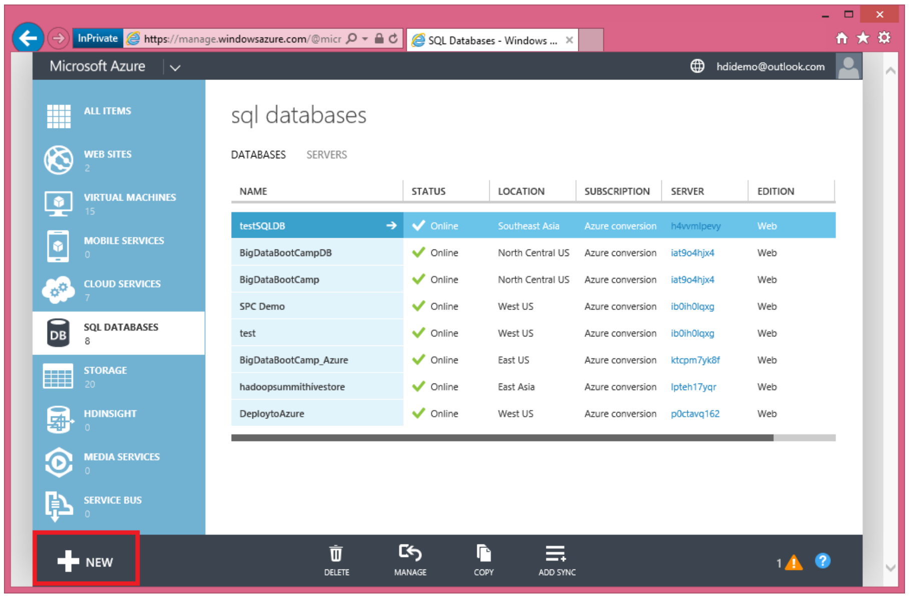
Publish project to Azure SQL Database

Create the target Azure SQL Database

1. If you are not already in Azure Management Portal, open Internet Explorer from the start screen and browse to https://manage.windowsazure.com/ then sign in using your Azure credentials.

2. In the blue navigation bar, click on **SQL DATABASES**.

3. In the grey options bar, click **+New**.



sql databases

DATABASES SERVERS

NAME	STATUS	LOCATION	SUBSCRIPTION	SERVER	EDITION
testSQLDB	✓ Online	Southeast Asia	Azure conversion	h4vvmlpevy	Web
BigDataBootCampDB	✓ Online	North Central US	Azure conversion	iat9o4hjx4	Web
BigDataBootCamp	✓ Online	North Central US	Azure conversion	iat9o4hjx4	Web
SPC Demo	✓ Online	West US	Azure conversion	ib0ih0lqxg	Web
test	✓ Online	West US	Azure conversion	ib0ih0lqxg	Web
BigDataBootCamp_Azure	✓ Online	East US	Azure conversion	ktcpm7yk8f	Web
hadoopsummithivestore	✓ Online	East Asia	Azure conversion	lpteh17yqr	Web
DeploytoAzure	✓ Online	West US	Azure conversion	p0ctavq162	Web

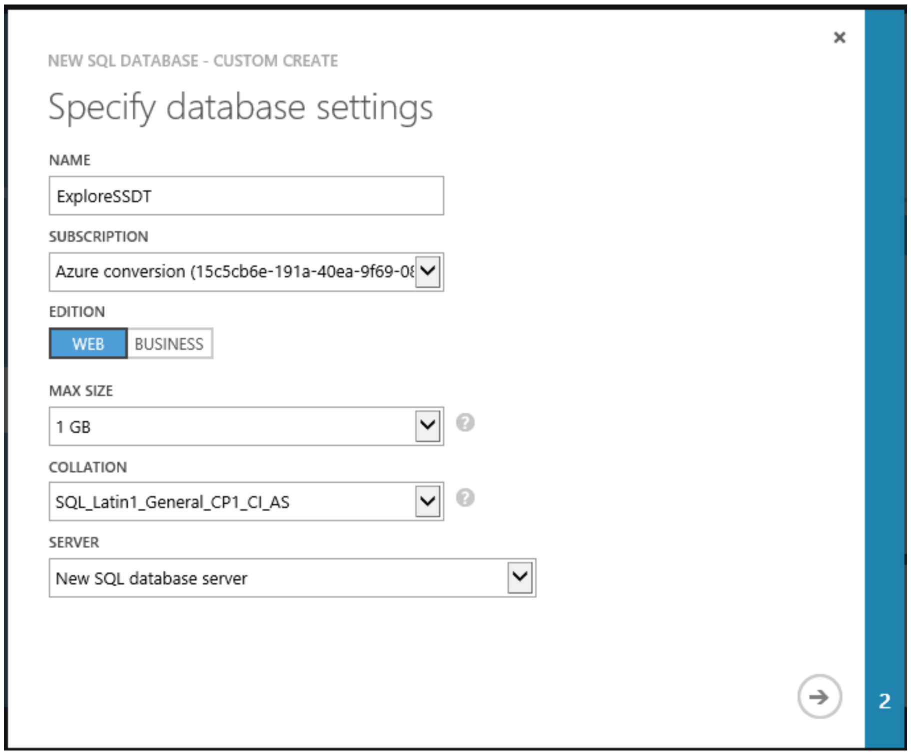
4. Select **Custom Create**.

5. Enter a name for the database as **ContosoRetailProj** and select your subscription if asked.

6. Select **SQL_Latin1_General_CP1_CI-AS** as the collation.

7. Choose **New SQL database server** as the **Server** and click the next arrow.

You can now close the lab environment.



NEW SQL DATABASE - CUSTOM CREATE

Specify database settings

NAME
ExploreSSDT

SUBSCRIPTION
Azure conversion (15c5cb6e-191a-40ea-9f69-0

EDITION
[✓ WEB] BUSINESS

MAX SIZE
1 GB

COLLATION
SQL_Latin1_General_CP1_CI_AS

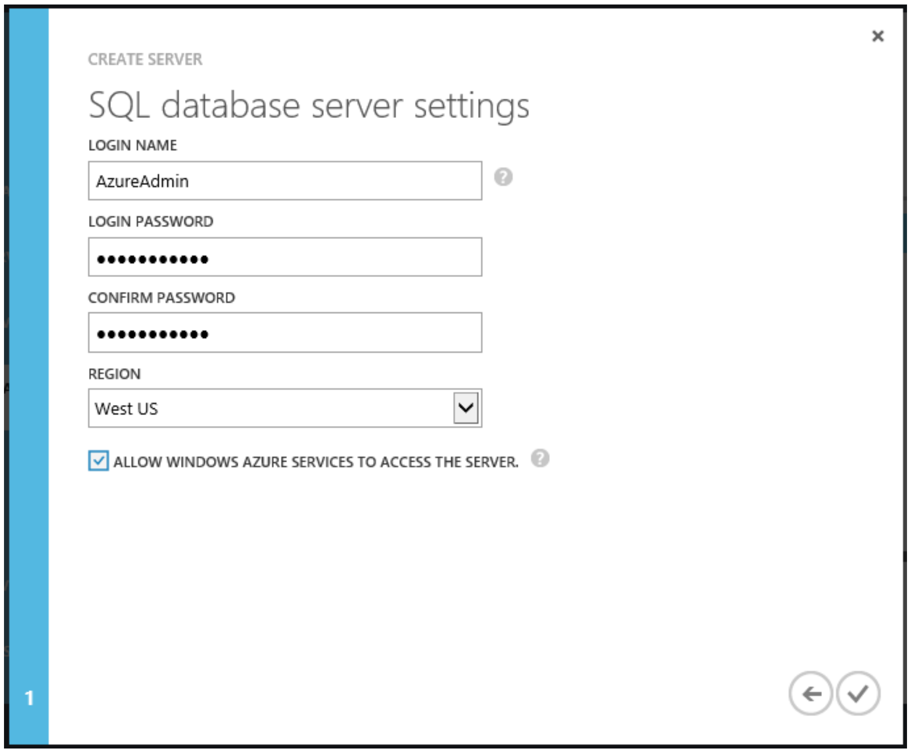
SERVER
New SQL database server

2

8. Enter a **LOGIN NAME** as **AzureAdmin** and **LOGIN PASSWORD** as **Pass@word12**

These define an SQL user which will initially be the only user who can access the database, and is not the same as your Azure login. You can create other SQL users later if desired (not covered in this tutorial.)

9. Choose **West US** as the region, leave **Allow Windows Azure Services to Access the Server** ticked, and click the check mark to create the database.



CREATE SERVER

SQL database server settings

LOGIN NAME: AzureAdmin

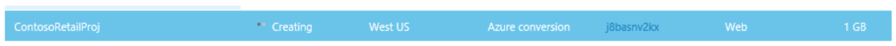
LOGIN PASSWORD: ●●●●●●●●●●●

CONFIRM PASSWORD: ●●●●●●●●●●●

REGION: West US

[✓] ALLOW WINDOWS AZURE SERVICES TO ACCESS THE SERVER.

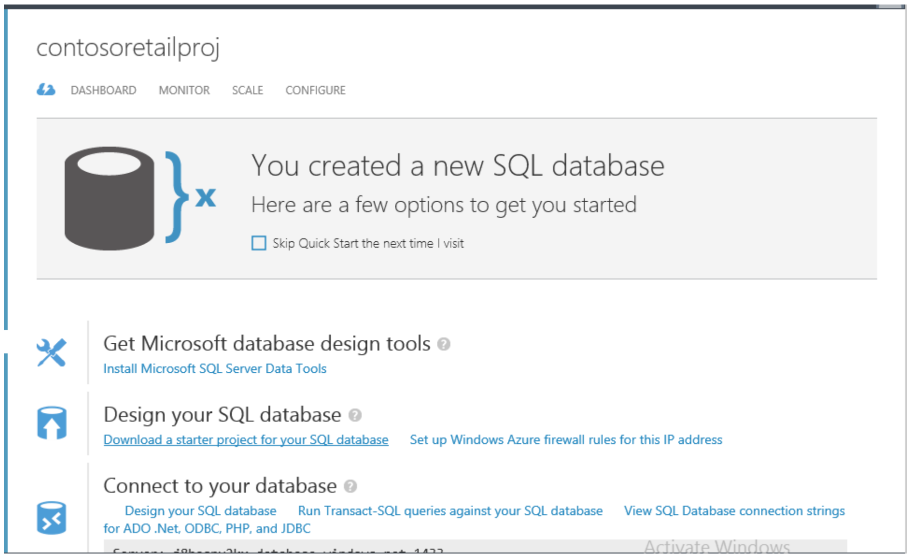
10. Copy the generated server name to the clipboard by selecting the name in the list and pressing **Ctrl+C**.



ContosoRetailProj	Creating	West US	Azure conversion	j8basnv2kx	Web	1 GB

You now need to open a firewall port on the Azure SQL Database so you can connect using your computer's IP address.

11. Click on the database name you created to display the database overview page.


contosoretailproj

DASHBOARD MONITOR SCALE CONFIGURE

You created a new SQL database

Here are a few options to get you started

☐ Skip Quick Start the next time I visit

Get Microsoft database design tools

Install Microsoft SQL Server Data Tools

Design your SQL database

Download a starter project for your SQL database   Set up Windows Azure firewall rules for this IP address

Connect to your database

Design your SQL database   Run Transact-SQL queries against your SQL database   View SQL Database connection strings for ADO .Net, ODBC, PHP, and JDBC

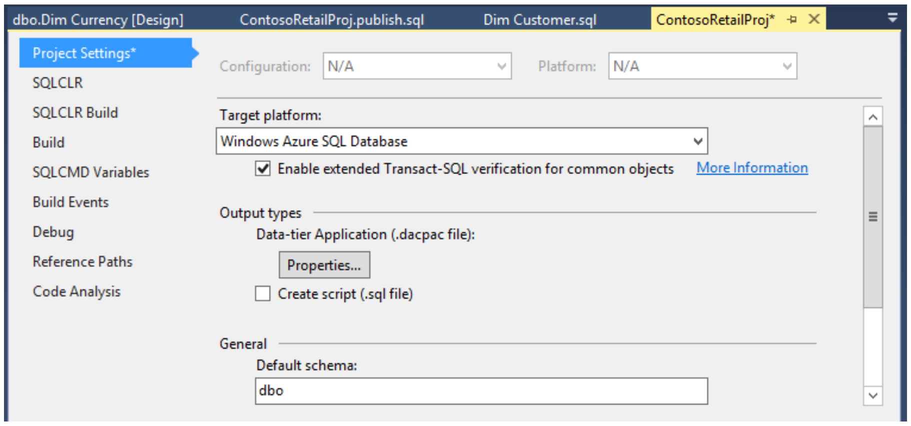
12. Click on **the Set up Windows Azure firewall rules for this IP address** link.

13. Click **Yes** to respond to the prompt to update the firewall rules.

Publish the Database to Azure SQL Database:

1. Right click on the project in the Solution Explorer and select **Properties**

2. Go to **Project Settings** tab and ensure that the **Target platform** is set to **Windows Azure SQL Database**


dbo.Dim Currency [Design]   ContosoRetailProj.publish.sql   Dim Customer.sql   ContosoRetailProj*

Project Settings*
SQLCLR
SQLCLR Build
Build
SQLCMD Variables
Build Events
Debug
Reference Paths
Code Analysis

Configuration: N/A   Platform: N/A

Target platform:
Windows Azure SQL Database
☑ Enable extended Transact-SQL verification for common objects   More Information

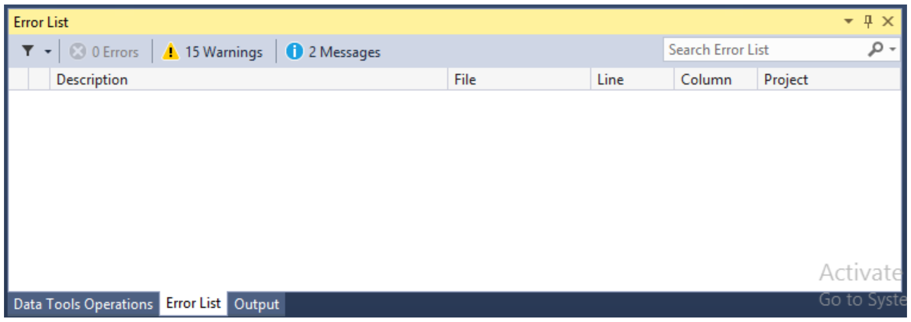
Output types
Data-tier Application (.dacpac file):
Properties...
☐ Create script (.sql file)

General
Default schema:
dbo

3. Click **File > Save Selected** in the menu or us CTL-S (press Ctrl key and S key down at the same time)

4. Right click on the project from **Solution Explorer** pane and click on **Build**

5. Open the **Error List** pane at the bottom of the window.
   Any problems for deploying to Azure will be displayed here.


Error List

0 Errors | 15 Warnings | 2 Messages | Search Error List

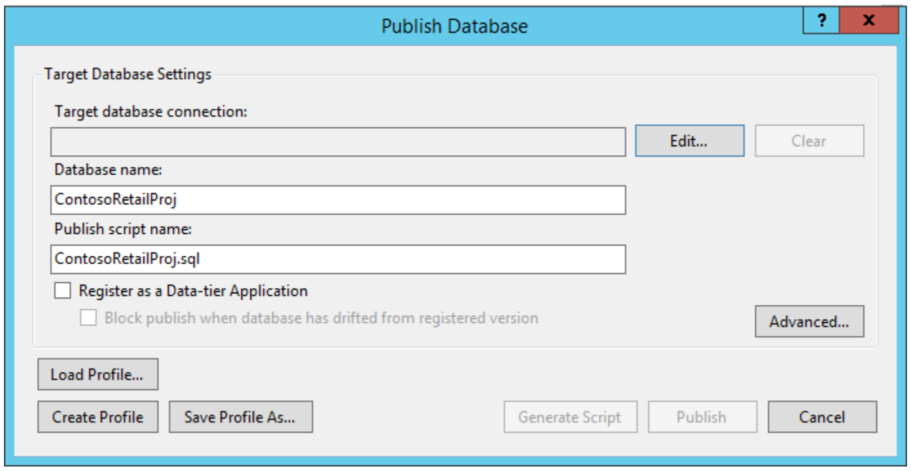
Description	File	Line	Column	Project

Data Tools Operations | Error List | Output

6. If there are any errors, double-click on them to go to the location of the error and then fix it. You can ignore warnings.

NOTE: To deploy a database into Window Azure, all tables present in the database should have a clustered index. Error SQL71560 (Table does not have a clustered index. Clustered indexes are required for inserting data in this version of SQL Server) could be eliminated by creating a clustered index for the table specified in the list.

7. Right click on the Project from the **Solution Explorer** pane and click on **Publish...**


Publish Database

Target Database Settings

Target database connection: [blank] Edit... Clear

Database name: ContosoRetailProj

Publish script name: ContosoRetailProj.sql

☐ Register as a Data-tier Application

☐ Block publish when database has drifted from registered version

Advanced...

Load Profile... | Create Profile | Save Profile As... | Generate Script | Publish | Cancel

8. Click on **Edit...** to open the **Connection Properties** dialog box

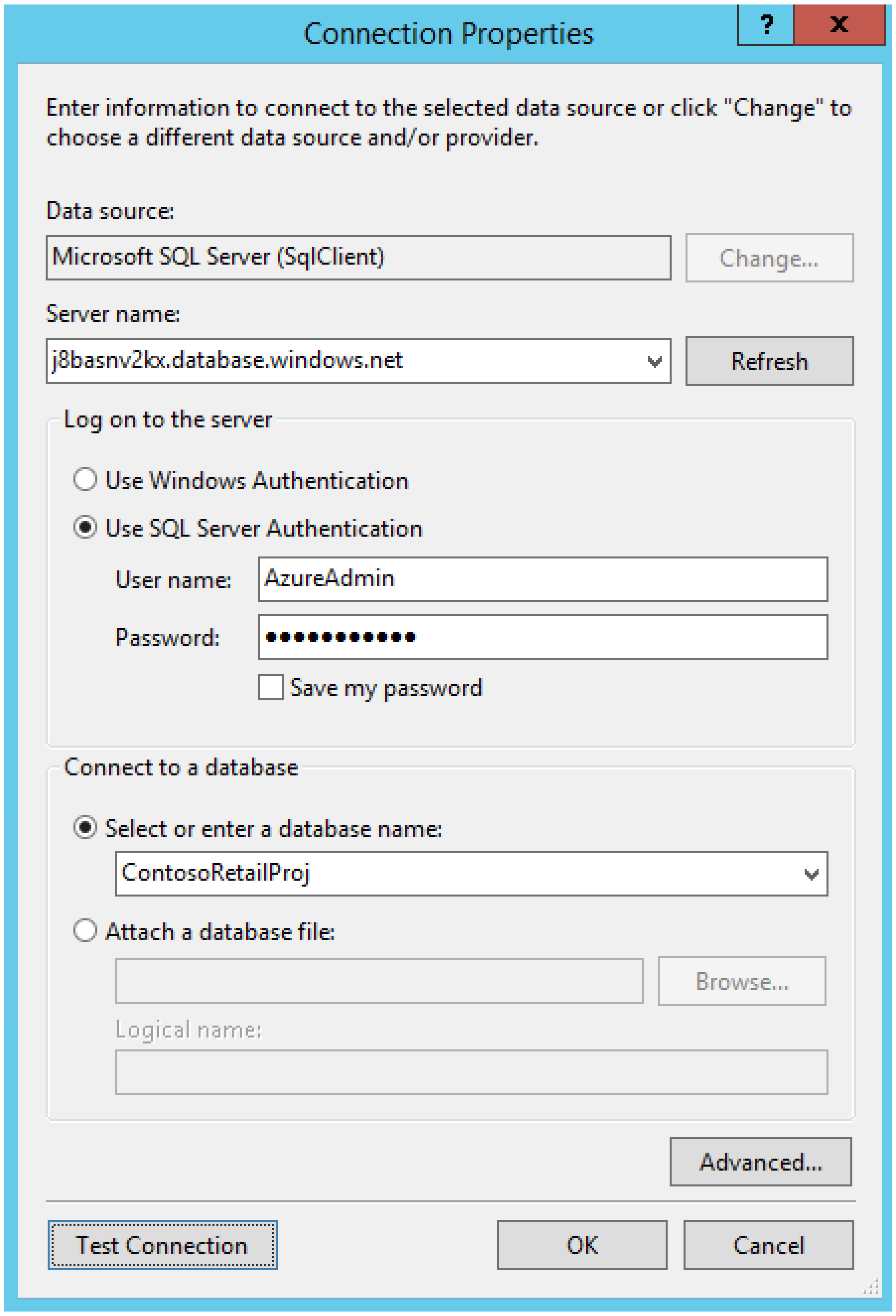
9. Enter the Windows Azure SQL Server name, by pasting the server name followed by **database.windows.net**

Example: n04vaq4nef.database.windows.net

10. Select **Use SQL Server Authentication**

11. Enter the **User name** as **AzureAdmin** and **Password** as **Pass@word12** to connect to the server

12. Enter ContosoRetailProj as the database name


Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: j8basnv2kx.database.windows.net

Log on to the server

Use Windows Authentication [✓ Use SQL Server Authentication]

User name: AzureAdmin

Password: ●●●●●●●●●●●

Save my password [ ]

Connect to a database

[✓ Select or enter a database name:] ContosoRetailProj

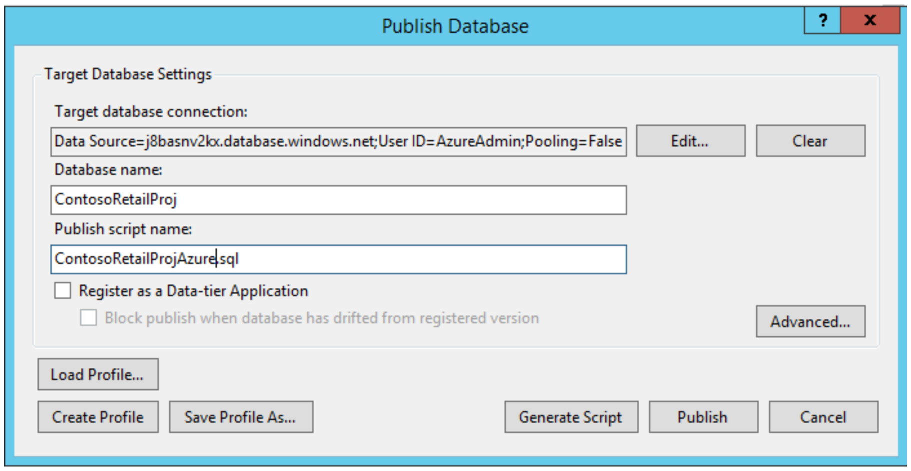
Attach a database file:

Logical name:

13. Click Test Connection to make sure you have the server connection information correct and the OK to dismiss the message box.

14. Click on **OK**

15. Change the publish script name to **ContosoRetailProjAzure.sql**


Publish Database

Target Database Settings

Target database connection: Data Source=j8basnv2kx.database.windows.net;User ID=AzureAdmin;Pooling=False

Database name: ContosoRetailProj

Publish script name: ContosoRetailProjAzure.sql

☐ Register as a Data-tier Application

☐ Block publish when database has drifted from registered version

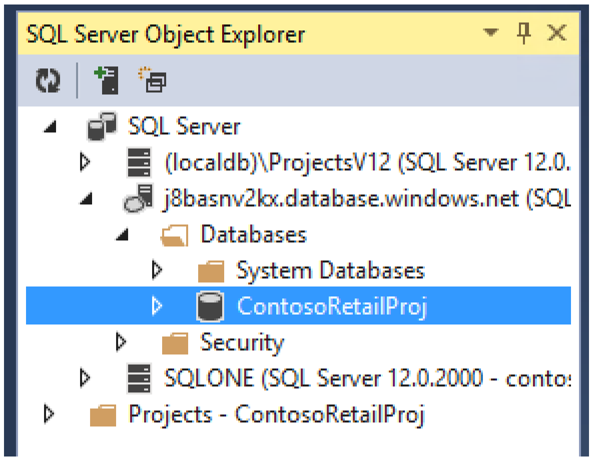
16. Click **Publish** to publish the changes directly.

NOTE: This may take a significant amount of time to process

Generating a script also helps in generating the deployment scripts which could be re-used for deploying to different environments

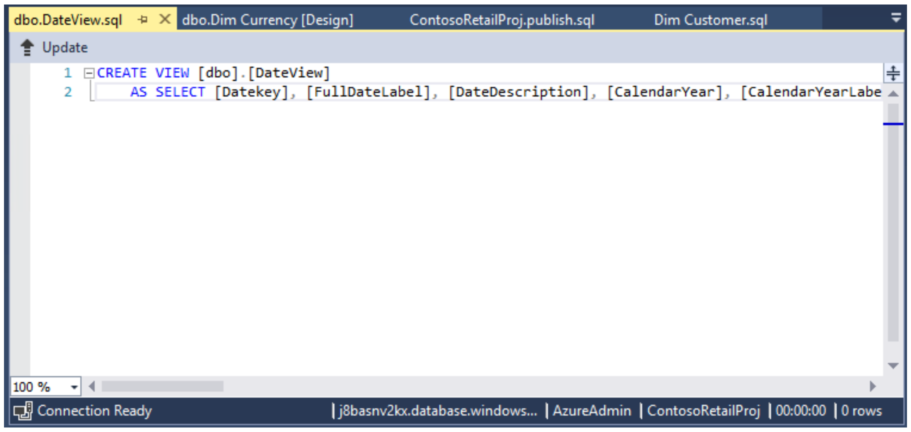
17. To verify the changes, right click on the Windows Azure SQL Server/Database in the **SQL Server Object Explorer** pane and click on **Refresh**.

18. Expand the **Server** and the **Database** folder


SQL Server Object Explorer
SQL Server
(localdb)\ProjectsV12 (SQL Server 12.0.
j8basnv2kx.database.windows.net (SQL
Databases
System Databases
ContosoRetailProj
Security
SQLONE (SQL Server 12.0.2000 - conto:
Projects - ContosoRetailProj

19. Locate the database to which the changes were deployed and expand the database followed by the database object folder (e.g. Tables, View, etc.)

20. Double click on the **View dbo.DataView** to open the designer and verify the published changes



```
CREATE VIEW [dbo].[DateView]
    AS SELECT [Datekey], [FullDateLabel], [DateDescription], [CalendarYear], [CalendarYearLabe
```

Connection Ready | j8basnv2kx.database.windows... | AzureAdmin | ContosoRetailProj | 00:00:00 | 0 rows

Visual Studio 2013 gives you a single interface to handle modification and publication with no intermediate steps. It saves a lot of time which is spent on integrating independent changes and different versions of development. You now have one go to tool to handle all your development and deployment needs. With Visual Studio 2013 it has become more agile to deploying and manage changes and applications between platforms.

Roll back Azure changes

Rollback Azure changes

1. Open the Azure Management Portal at https://manage.windowsazure.com/ and sign in with your credentials

2. Delete Azure databases and servers

   i. Click on **SQL DATABASES** in the navigation pane

   ii. Click **SERVERS** at the page top and select the server you want to delete (if you are not sure which server you created during this experience, click on the server name then **Databases** at the top of the page to see a list of databases hosted on this server.)

   iii. Click **DELETE** in the options pane at the bottom of the page

   iv. Carry out the confirmation steps and click the check mark.

You can now close this lab.

Terms of use