Module 2 | Lab

Lab Overview

SCENARIO

A local company is looking for consultants to help with the creation of a Business Intelligence solution. Currently you are one of a number of consultants they are interested in and have asked you to create a prototype ETL solution to get a better understanding of your skill level. This prototype is based on an example subset of Microsoft's AdventureWorks demonstration database and their IT team have provided both a source and destination database to start with. What you need to do is create a SQL based ETL process and send them the SQL Script as an example of your work.

LAB OVERVIEW

In this lab, you will create a SQL based ETL script. You will start by programming a simple script and then add complexity as you progress through the lab. Before starting this lab, you need to be familiar with the content of Module 2 | ETL with SQL Programming. Then, if you have not already done so, follow the instructions in the Setting up the Lab Environment section of this course to set up the lab environment.

WHAT YOU'LL NEED

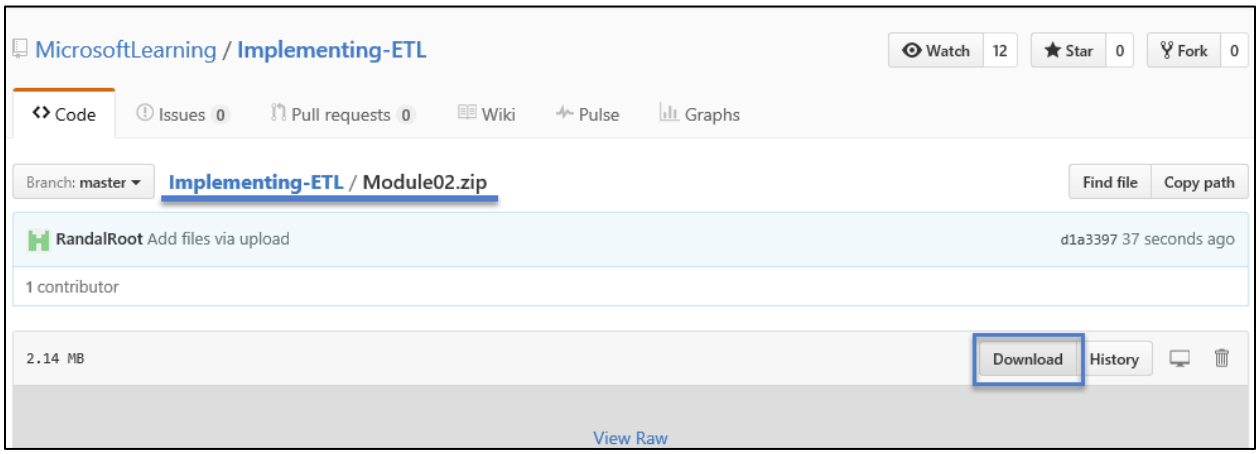
A personal or virtual computer with the SQL Server 2016 (or 2014 or 2012) installed on it
Permissions to download files on the computer
Permissions to create a new folders and files on the computer
Permissions to create databases in the SQL Server instance
A SQL database file: http://msftdbprodsamples.codeplex.com/downloads/get/354847
Several Course files: https://github.com/MicrosoftLearning/Implementing-ETL

Getting Started

To complete the labs, you must download and extract the lab resources

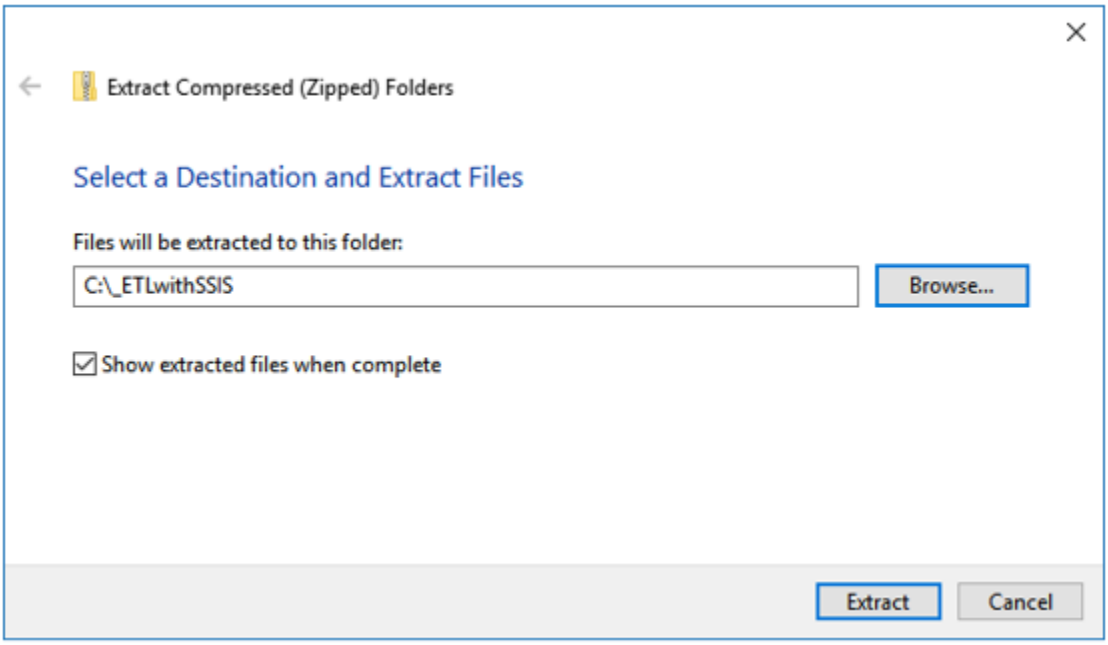
1. Open a **web browser** and navigate to https://github.com/MicrosoftLearning/Implementing-ETL (This is an external link that opens in a new window.)

2. Download the course resources, by clicking on the ***Implementing-ETL/Module02.zip file***, and then clicking the Download button to save the downloaded file to your computer.



3. Locate the **Module02.zip** file, then Right-Click on it and select **Properties** from the context menu.

5. In the dialog window, check **Unblock**, then click **OK** to close. (This may be an Unblock button on some versions of Windows.)



7. Extract the file's content, by right-clicking the **Module02.zip** file, and selecting **Extract All**.

8. In the dialog window, replace the folder **path** to C:_ETLwithSSIS.

9. Click Extract. This will create a new **folder** and unzipped the **module02.zip** file.

10. Navigate to the C:_ETLwithSSIS folder.

Important:

■ **Important:** Some lab scripts use absolute file paths, please extract and use the lab resources directly from the C:_ETLwithSSIS path. This lab requires a lot of SQL programming; therefore, the answer code is supplied with this lab in case you need help.

■ **Note:** Each lab in this course is standalone, and can be completed individually.

Exercise 1 | Attaching the Source Database

1. Download and **SQL Server AdventureWorks LT 2012 database file** from the product samples page on **Codeplex**. The database file name is **AdvntureWorksLT2012_Data.mdf.** This file can be found at this URL: http://msftdbprodsamples.codeplex.com/downloads/get/354847

2. Verify that the file is in your browser's **Downloads** folder on your computer.

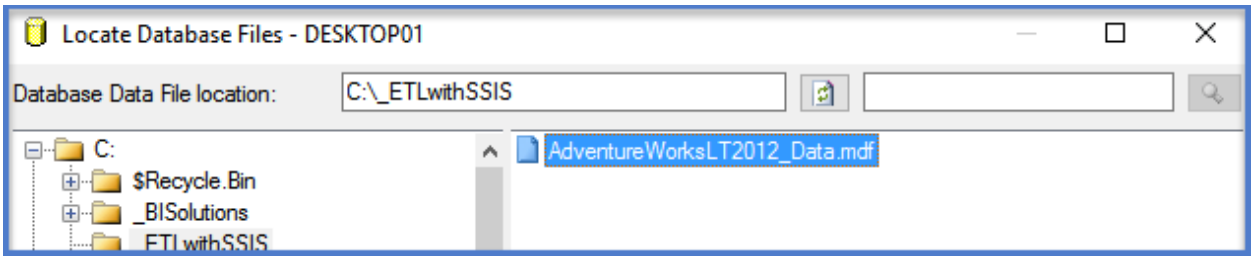
3. Copy the **AdventureWorksLT2012_Data.mdf** file to the **C:_ETLWithSSIS** folder.

4. Start **SQL Server Management Studio**, using the **Run as Administrator** option, and connect to the **Database Engine** instance.

5. Right-click **Databases** in the **Object Explorer Tree View** window, then click **Attach** in the Context Menu.

6. In the Attach Database Dialog Window, Click the **Add** button.

7. Select the **AdventureWorksLT2012_Data.mdf** database file and click **OK**. If the file is not listed, check the folder to be sure the file is there.



8. In database details, Click the **Remove** button to remove the **Log file entry**. The setup program assumes you have a log file, but there is no log file in the sample. A new log file will be created automatically when you attach the database.

9. Click **OK** to attach just the primary database file.

Exercise 2 | Creating the Destination Database

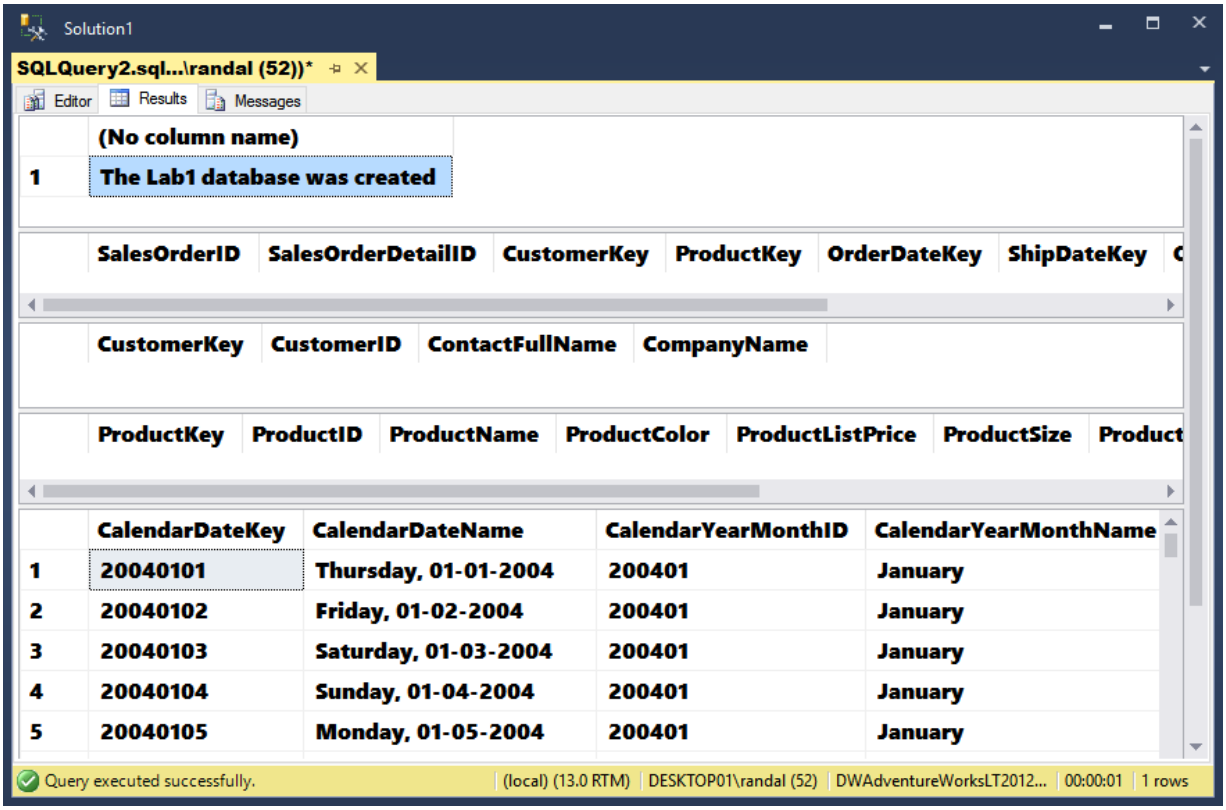
1. Start **SQL Server Management Studio** and connect to the **Database Engine** instance.

2. Use the File > Open > File menu to open the **Open File** dialog window.

3. Locate and open the **C:_ETLwithSSIS\Module02\Lab\Exercise 2 - Creating the Destination Database\Create the DWAdventureWorksLTSalesLab1 database.sql** file.

4. Use the **[! Execute]** button on the toolbar to run all of the code in the file.

5. Verify that the results of running the script look like the following image.



6. Refresh the **SSMS Object Explorer Tree** view and verify that the database was created.

Exercise 3 | Review the Developer Worksheet

1. Locate and open the **C:_ETLwithSSIS\Module02\Lab\Exercise 3 - Review the Developer Worksheet\Lab1-ETLPlanningWorksheets-DWandETLDesign.xlsx** Excel file in the course's folder.

2. Review the transformations listed on the **ETL Design Tab**.

ETL Design Sheet for: DWAdventureWorksLT2012

Source Column	Source DataTyp	Source N	Destination Column	Destination DataType	Destination ?	*Keep for LAB 1*	Reason	Transformations
Customer			DimCustomers			Yes	Contains primary dimensional data	
NA	NA	NA	dbo.DimCustomers.CustomerKey	int	No	Yes	Surrogate Key	Identity (1,1)
SalesLT.Customer.CustomerID	int	No	dbo.DimCustomers.CustomerID	int	No	Yes	Original Key can be used for reference to source	
SalesLT.Customer.FirstName	nvarchar (50)	No	dbo.DimCustomers.FullName	nvarchar (200)	No	Yes	Human Friendly identifier	Rename; Cast to nvarchar(200); Merge with LastName
SalesLT.Customer.LastName	nvarchar (50)	No	dbo.DimCustomers.FullName	nvarchar (200)	No	Yes	Human Friendly identifier	Rename; Cast to nvarchar(200); Merge with FirstName
SalesLT.Customer.CompanyName	nvarchar (128)	Yes	dbo.DimCustomers.CompanyName	nvarchar (200)	No	Yes	Human Friendly identifier	Cast to nvarchar(200), change null status;
ProductCategory			DimCategories			No	Merged with DimProducts	Merge with DimProducts for first level of the Category hierachy only
SalesLT.ProductCategory.ProductCateg	int	No	dbo.DimProductsCategory.ProductCategoryID	int	No	Yes	Original Key can be used for reference to source	Join;
SalesLT.ProductCategory.Name	nvarchar (50)	No	dbo.DimProductsCategory.ProductCategoryName	nvarchar (50)	No	Yes	Human Friendly identifier	Rename;
Product			DimProducts			Yes	Contains primary dimensional data	
NA	NA	NA	dbo.DimProducts.ProductKey	int	No	Yes	Surrogate Key	Identity (1,1)
SalesLT.Product.ProductID	int	No	dbo.DimProducts.ProductID	int	No	Yes	Original Key can be used for reference to source	
SalesLT.Product.Name	nvarchar (50)	No	dbo.DimProducts.ProductName	nvarchar (50)	No	Yes	Human Friendly identifier	Rename;
SalesLT.Product.ProductNumber	nvarchar (25)	No	dbo.DimProducts.ProductNumber	nvarchar (50)	No	Yes	Original External Key can be used for reference to source	Cast to nvarchar(50)
SalesLT.Product.Color	nvarchar (15)	Yes	dbo.DimProducts.ProductColor	nvarchar (50)	No	Yes	Good for grouping aggregates	Change null status; Change null value; Cast to nvarchar(50)
SalesLT.Product.StandardCost	money (19,4)	No	dbo.DimProducts.ProductStandardCost	money (19,4)	No	Yes	Good for grouping aggregates	Rename;
SalesLT.Product.ListPrice	money (19,4)	No	dbo.DimProducts.ProductListPrice	money (19,4)	No	Yes	Good for grouping aggregates	Rename;
SalesLT.Product.Size	nvarchar (5)	Yes	dbo.DimProducts.ProductSize	nvarchar (5)	No	Yes	Good for grouping aggregates	Rename; Change null status; Change null value
SalesLT.Product.Weight	decimal (8,2)	Yes	dbo.DimProducts.ProductWeight	decimal (8,2)	Yes	Yes	Good for grouping aggregates	Rename; Leave null for proper weight calculations
SalesLT.Product.CategoryID	int	Yes	dbo.DimProducts.ProductCategoryID	int	No	Yes	Original Key can be used for reference to source	Rename; Join;
SalesOrderDetail			FactSales			Yes	Contains primary fact data	
SalesLT.SalesOrderDetail.SalesOrderID	int	No	dbo.FactSales.SalesOrderID	int	No	Yes	Original Key can be used for reference to source	
SalesLT.SalesOrderDetail.SalesOrderDetai	int	No	dbo.FactSales.SalesOrderDetailID	int	No	Yes	Original Key can be used for reference to source	
SalesLT.SalesOrderDetail.OrderQty	smallint	No	dbo.FactSales.OrderQty	smallint	No	Yes	Measure Value	
NA	NA	NA	dbo.FactSales.ProductKey	int	No	Yes	Surrogate Key used for foreign key to dimension table	Join; Lookup in dimension table
SalesLT.SalesOrderDetail.UnitPrice	money (19,4)	No	dbo.FactSales.UnitPrice	money (19,4)	No	Yes	Measure Value	
SalesLT.SalesOrderDetail.UnitPriceDisco	money (19,4)	No	dbo.FactSales.UnitPriceDiscount	money (19,4)	No	Yes	Original Key can be used to calculate LineTotal	
SalesOrderHeader			FactSales			Yes	Will be merged with FactSales based on SalesOrderID	
SalesLT.SalesOrderHeader.SalesOrderID	int	No	dbo.FactSales.SalesOrderID	int	No	Yes	Original Key can be used for reference to source	
NA	NA	NA	dbo.FactSales.OrderDateKey	int	No	Yes	Surrogate Key used for foreign key to dimension table	Join; Lookup in dimension table
NA	NA	NA	dbo.FactSales.ShipDateKey	int	No	Yes	Surrogate Key used for foreign key to dimension table	Join; Lookup in dimension table
NA	NA	NA	dbo.FactSales.CustomerKey	int	No	Yes	Surrogate Key used for foreign key to dimension table	Join; Lookup in dimension table
			DimDates			Yes	Contains primary dimensional data	[NOTE: This is already done at the start of LAB 1]

■ **NOTE:** You do not need to spend a lot of time reviewing the worksheet. Just get the general idea of how to read it and then use it as a reference for the next exercises (as needed).

3. Start **SQL Server Management Studio** and connect to the **Database Engine** instance.

4. Use the **File > Open > File** Menu to open the **C:_ETLwithSSIS\Module02\Lab\Exercise 3 - Review the Developer Worksheet\ETL Code for the DWAdventureWorksLTLab1 database.sql** file

5. Review the code in this file.

6. Use the **[! Execute]** button on the toolbar to run all of the code in the file.

7. Verify that running the script returns the following text on the **Messages** tab.

Results | Messages

```
(0 row(s) affected)

(0 row(s) affected)

(366 row(s) affected)

(0 row(s) affected)
```

Exercise 4 | Creating ETL Select Statements

1. Start **SQL Server Management Studio** and connect to the **Database Engine** instance.

2. Use the **File > Open > File** Menu to open the **C:_ETLwithSSIS\Module02\Lab\Exercise 4 - Creating ETL Select Statements\Starter_ETL Code for the DWAdventureWorksLTLab1 database.sql** file

3. Review the **code** in this file. Consider the following **transformations**:

```
  [CustomerID] = T1.CustomerID
, [CompanyName] = Cast(CompanyName as nvarchar(200))
, [ContactFullName] = Cast([FirstName] + ' ' + [LastName] as nvarchar(200))
```

4. Locate the following **code**:

```
-- DimCustomers
/*
INSERT INTO [DWAdventureWorksLT2012Lab01].[dbo].[DimCustomers]
( [CustomerID]
, [ContactFullName]
, [CompanyName]
)
 <Add your ETL Select Statement Here>
*/
go
```

5. Uncomment the **Insert statement** and add code that will **select** and **transform** the source data.

6. Test your code using the **Insert statement** provided.

7. Verify that the result of running your ETL code looks like the following image.

	CustomerKey	CustomerID	CompanyName	ContactFullName
1	1	1	A Bike Store	Orlando Gee
2	2	2	Progressive Sports	Keith Harris

8. Consider the following **transformations**:

```
  [ProductID] = T1.[ProductID]
, [ProductName] = T1.[Name]
, [ProductColor] = IsNull( Cast( T1.[Color] as nvarchar(50)), 'Not Defined')
, [ProductListPrice] =T1.[ListPrice]
, [ProductSize] = IsNull( T1.[Size], -5) -- A value could be entered, but has not
, [ProductWeight] = T1.[Weight] -- Leave null for proper weight calculations
, [ProductCategoryID] = T2.[ProductCategoryID]
, [ProductCategoryName] = T2.[Name]
```

9. Locate the following **code**:

```
-- DimProducts
/*
INSERT INTO [DWAdventureWorksLT2012Lab01].[dbo].[DimProducts]
( [ProductID]
, [ProductName]
, [ProductColor]
, [ProductListPrice]
, [ProductSize]
, [ProductWeight]
, [ProductCategoryID]
, [ProductCategoryName]
)
 <Add your ETL Select Statement Here>
*/
Go
```

10. Uncomment the **Insert statement** and add code that will **select** and **transform** the source data.

11. Test your code using the **Insert statement** provided.

12. Verify that the result of running your ETL code looks like the following **image**.

```
143 SELECT * FROM [DWAdventureWorksLT2012Lab01].[dbo].[DimProducts];
```

106 %

Results | Messages

	ProductKey	ProductID	ProductName	ProductColor	ProductListPrice	ProductSize	ProductWeight	ProductCategoryID	ProductCategoryName
1	1	771	Mountain-100 Silver, 38	Silver	3399.99	38	9230.56	5	Mountain Bikes
2	2	772	Mountain-100 Silver, 42	Silver	3399.99	42	9421.06	5	Mountain Bikes
3	3	773	Mountain-100 Silver, 44	Silver	3399.99	44	9584.36	5	Mountain Bikes

13. Consider the following **transformations**:

```
  T1.[SalesOrderID]
, [SalesOrderDetailID]
, T3.[CustomerKey]
, T4.[ProductKey]
, [OrderDateKey] = T5.CalendarDateKey
, [ShippedDateKey] = T6.CalendarDateKey
, [OrderQty]
, [UnitPrice]
, [UnitPriceDiscount]
```

14. Locate the following **code**:

```
-- Fill Fact Sales
/*
INSERT INTO [DWAdventureWorksLT2012Lab01].[dbo].[FactSales]
( [SalesOrderID]
, [SalesOrderDetailID]
, [CustomerKey]
, [ProductKey]
, [OrderDateKey]
, [ShipDateKey]
, [OrderQty]
, [UnitPrice]
, [UnitPriceDiscount]
)
 <Add your ETL Select Statement Here>
*/
go
```

15. Uncomment the **Insert statement** and add code that will select and transform the source data.

16. Test your code using the **Insert statement** provided.

17. Verify that the result of running your ETL code looks like the following image.

```
152 SELECT * FROM [DWAdventureWorksLT2012Lab01].[dbo].[FactSales];
```

106 %

Results | Messages

	SalesOrderID	SalesOrderDetailID	CustomerKey	ProductKey	OrderDateKey	ShipDateKey	OrderQty	UnitPrice	UnitPriceDiscount
1	71774	110562	661	161	20040601	20040608	1	356.898	0.00
2	71774	110563	661	166	20040601	20040608	1	356.898	0.00
3	71776	110567	820	110	20040601	20040608	1	63.90	0.00

18. Run the entire **script** and verify that all of the **tables** are filled with data as shown here:

Solution1

Answer_ETL Co...\randal (53))

Editor | Results | Messages

	CustomerKey	CustomerID	CompanyName	ContactFullName
1	1	1	A Bike Store	Orlando Gee
2	2	2	Progressive Sports	Keith Harris
3	3	3	Advanced Bike Components	Donna Carreras
4	4	4	Modular Cycle Systems	Janet Gates

	ProductKey	ProductID	ProductName	ProductColor	ProductListPrice	ProductSize	ProductWeight	ProductCategoryID	Prod
1	1	771	Mountain-100 Silver, 38	Silver	3399.99	38	9230.56	5	Mou
2	2	772	Mountain-100 Silver, 42	Silver	3399.99	42	9421.06	5	Mou

	CalendarDateKey	CalendarDateName	CalendarYearMonthID	CalendarYearMonthName	CalendarYearQuarterID	CalendarYearQuarteri
1	20040101	Thursday, 01-01-2004	200401	January	200401	Q1 - 2004
2	20040102	Friday, 01-02-2004	200401	January	200401	Q1 - 2004
3	20040103	Saturday, 01-03-2004	200401	January	200401	Q1 - 2004
4	20040104	Sunday, 01-04-2004	200401	January	200401	Q1 - 2004
5	20040105	Monday, 01-05-2004	200401	January	200401	Q1 - 2004
6	20040106	Tuesday, 01-06-2004	200401	January	200401	Q1 - 2004
7	20040107	Wednesday, 01-07-2004	200401	January	200401	Q1 - 2004

	SalesOrderID	SalesOrderDetailID	CustomerKey	ProductKey	OrderDateKey	ShipDateKey	OrderQty	UnitPrice	UnitPriceDiscount
1	71774	110562	661	161	20040601	20040608	1	356.898	0.00
2	71774	110563	661	166	20040601	20040608	1	356.898	0.00
3	71776	110567	820	110	20040601	20040608	1	63.90	0.00
4	71780	110616	843	146	20040601	20040608	4	218.454	0.00

Query executed successfully. | (local) (13.0 RTM) | DESKTOP01\randal (53) | DWAdventureWorksLT2012... | 00:00:00 | 847 rows

The **Message** tab should now indicate the following rows were inserted and selected as shown in this image:

Results | Messages

(847 row(s) affected)

(295 row(s) affected)

(542 row(s) affected)

(847 row(s) affected)

(295 row(s) affected)

(366 row(s) affected)

(542 row(s) affected)

Exercise 5 | Create ETL Views

1. Start **SQL Server Management Studio** and connect to the Database Engine instance.

2. Use the **File > Open > File** Menu to open the **C:_ETLwithSSIS\Module02\Lab\Exercise 5 - Creating ETL Views\Starter_ETL Code for the DWAdventureWorksLTLab1 database with Views.sql** file.

3. Locate and review the following **Insert statements**, noting that these statements now reference **SQL Views**.

```
-- DimCustomers
INSERT INTO [DWAdventureWorksLT2012Lab01].[dbo].[DimCustomers]
( [CustomerID]
, [CompanyName]
, [ContactFullName]
)
SELECT
         [CustomerID]
       , [CompanyName]
       , [ContactFullName]
FROM [DWAdventureWorksLT2012Lab01].[dbo].[vETLDimCustomersData]
go

-- DimProducts
INSERT INTO [DWAdventureWorksLT2012Lab01].[dbo].[DimProducts]
( [ProductID]
, [ProductName]
, [ProductColor]
, [ProductListPrice]
, [ProductSize]
, [ProductWeight]
, [ProductCategoryID]
, [ProductCategoryName]
)
SELECT
        [ProductID]
       ,[ProductName]
       ,[ProductColor]
       ,[ProductListPrice]
       ,[ProductSize]
       ,[ProductWeight]
       ,[ProductCategoryID]
       ,[ProductCategoryName]
FROM [DWAdventureWorksLT2012Lab01].[dbo].[vETLDimProductsData]
go

--*********************************************************************--
-- Fill Fact Tables
--*********************************************************************--
```

```sql
-- Fill Fact Sales
INSERT INTO [DWAdventureWorksLT2012Lab01].[dbo].[FactSales]
( [SalesOrderID]
, [SalesOrderDetailID]
, [CustomerKey]
, [ProductKey]
, [OrderDateKey]
, [ShipDateKey]
, [OrderQty]
, [UnitPrice]
, [UnitPriceDiscount]
)
  SELECT
         [SalesOrderID]
       , [SalesOrderDetailID]
       , [CustomerKey]
       , [ProductKey]
       , [OrderDateKey]
       , [ShippedDateKey]
       , [OrderQty]
       , [UnitPrice]
       , [UnitPriceDiscount]
  FROM [DWAdventureWorksLT2012Lab01].[dbo].[vETLFactSalesData]
go
```

4. Locate the following **code** and add the **ETL Select statements** you created in the exercise 4 to complete each View's code.

```sql
--*********************************************************************--
-- Create ETL Views
--*********************************************************************--
If (object_id('vETLDimCustomersData') is not null) Drop View vETLDimCustomersData;
go
CREATE VIEW vETLDimCustomersData
AS
--< Add you ETL Select Statement here! >--
go

If (object_id('vETLDimProductsData') is not null) Drop View vETLDimProductsData;
go
CREATE VIEW vETLDimProductsData
AS
--< Add you ETL Select Statement here! >--
go

If (object_id('vETLFactSalesData') is not null) Drop View vETLFactSalesData;
go
CREATE VIEW vETLFactSalesData
AS
--< Add you ETL Select Statement here! >--
go
```

5. Run in entire script and verify that all of the **tables** are filled with data.

Exercise 8 | Create ETL Stored Procedures

1. Start **SQL Server Management Studio** and connect to the Database Engine instance.

2. Use the **File > Open > File** Menu to open the **C:_ETLwithSSIS\Module02\Lab\Exercise 6 - Creating an ETL Stored Procedures\Starter_ETL Code for the DWAdventureWorksLTLab1 database with Stored Procedures.sql** file.

3. Locate and review the following **SQL code**, noting how the **stored procedures** will be called to fill the **dimension** and **fact** tables:

```
--*********************************************************************--
-- Fill Dimension Tables
--*********************************************************************--

-- DimCustomers
Declare @ReturnCode int
Execute @ReturnCode = pETLFillDimCustomers
Select [Return Status for pETLFillDimCustomers ] = @ReturnCode
go

-- DimProducts
Declare @ReturnCode int
Execute @ReturnCode = pETLFillDimProducts
Select [Return Status for pETLFillDimProducts] = @ReturnCode
go

--*********************************************************************--
-- Fill Fact Tables
--*********************************************************************--

-- Fill Fact Sales
Declare @ReturnCode int
Execute @ReturnCode = pETLFillFactSales
Select [Return Status for pETLFillFactSales] = @ReturnCode
go
```

4. Locate and review the following **SQL code**:

```
--*********************************************************************--
-- Create an ETL Stored Procedures
--*********************************************************************--
/*
If (object_id('pETLProcedureTemplate') is not null) Drop Procedure pETLProcedureTemplate;
go
CREATE -- ETL Stored Procedure Template
PROCEDURE pETLProcedureTemplate
AS
       /*************************************************************
       Desc: <Desc Goes Here>
       ChangeLog: When, Who, What
       20160101,RRoot,Created Procedure
       *************************************************************/
Begin -- Procedure Code
```

```
 Declare
   @RC int = 0;
 Begin Try
  Begin Transaction;
  -- ETL Code  ---------------------------------------------------------------

   Select 3/1 -- Replace this test code!;

  -- ETL Code  ---------------------------------------------------------------
  Commit Transaction;
  Set @RC = 100; -- Success
 End Try
 Begin Catch
  Rollback Tran;
  Set @RC = -100; -- Failure
 End Catch
 Return @RC;
End -- Procedure Code
;
go

--Declare @ReturnCode int
--Execute @ReturnCode = pETLProcedureTemplate
--Select @ReturnCode
--go


If (object_id('pETLFillDimCustomers') is not null) Drop Procedure pETLFillDimCustomers;
go
CREATE  -- ETL Stored Procedure for DimCustomers
PROCEDURE pETLFillDimCustomers
< Add Code here to complete the stored procedure >


If (object_id('pETLFillDimProducts') is not null) Drop Procedure pETLFillDimProducts;
go
CREATE  -- ETL Stored Procedure for DimProducts
PROCEDURE pETLFillDimProducts
< Add Code here to complete the stored procedure >


If (object_id('pETLFillFactSales') is not null) Drop Procedure pETLFillFactSales;
go
CREATE  -- ETL Stored Procedure for FactSales
PROCEDURE pETLFillFactSales
< Add Code here to complete the stored procedure >

*/
```

5. Uncomment this **code** and add the **code** to create **stored procedures** that will fill the tables using the inserts statements you created in **Exercise 5**.

6. Run in entire **script** and verify that all of the **tables** are filled with data as shown here:

Solution1

Answer_ETL Co...\randal (53))

Editor | Results | Messages

	CustomerKey	CustomerID	CompanyName	ContactFullName
1	1	1	A Bike Store	Orlando Gee
2	2	2	Progressive Sports	Keith Harris
3	3	3	Advanced Bike Components	Donna Carreras
4	4	4	Modular Cycle Systems	Janet Gates

	ProductKey	ProductID	ProductName	ProductColor	ProductListPrice	ProductSize	ProductWeight	ProductCategoryID	Prod
1	1	771	Mountain-100 Silver, 38	Silver	3399.99	38	9230.56	5	Mou
2	2	772	Mountain-100 Silver, 42	Silver	3399.99	42	9421.06	5	Mou

	CalendarDateKey	CalendarDateName	CalendarYearMonthID	CalendarYearMonthName	CalendarYearQuarterID	CalendarYearQuarteri
1	20040101	Thursday, 01-01-2004	200401	January	200401	Q1 - 2004
2	20040102	Friday, 01-02-2004	200401	January	200401	Q1 - 2004
3	20040103	Saturday, 01-03-2004	200401	January	200401	Q1 - 2004
4	20040104	Sunday, 01-04-2004	200401	January	200401	Q1 - 2004
5	20040105	Monday, 01-05-2004	200401	January	200401	Q1 - 2004
6	20040106	Tuesday, 01-06-2004	200401	January	200401	Q1 - 2004
7	20040107	Wednesday, 01-07-2004	200401	January	200401	Q1 - 2004

	SalesOrderID	SalesOrderDetailID	CustomerKey	ProductKey	OrderDateKey	ShipDateKey	OrderQty	UnitPrice	UnitPriceDiscount
1	71774	110562	661	161	20040601	20040608	1	356.898	0.00
2	71774	110563	661	166	20040601	20040608	1	356.898	0.00
3	71776	110567	820	110	20040601	20040608	1	63.90	0.00
4	71780	110616	843	146	20040601	20040608	4	218.454	0.00

Query executed successfully. | (local) (13.0 RTM) | DESKTOP01\randal (53) | DWAdventureWorksLT2012... | 00:00:00 | 847 rows

The **Message** tab should now indicate the following rows were inserted and selected as shown in this image:

Results | Messages

(847 row(s) affected)

(295 row(s) affected)

(542 row(s) affected)

(847 row(s) affected)

(295 row(s) affected)

(366 row(s) affected)

(542 row(s) affected)