LAB MANUAL

Course: CSC496 DATA WAREHOUSING AND DATA MINING

Department of Computer Science

Learning Procedure

1) Stage J (Journey inside-out the concept)
2) Stage a_1 (Apply the learned)
3) Stage V (Verify the accuracy)
4) Stage a_2 (Assess your work)

COMSATS University Islamabad (CUI)
Islamabad

Table of Contents

Lab # 13	Use of Decision Trees for Classification	
	K-Means Clustering	
Lab # 14	Implementing a Prediction odel	
Lab # 15	Process, Classify and Cluster Explorers	
Lab # 16	Associate, Visualize, and Select Attribute Explorer	
	Terminal Examination	

Requirements:

To complete exercise in this manual following software are required:

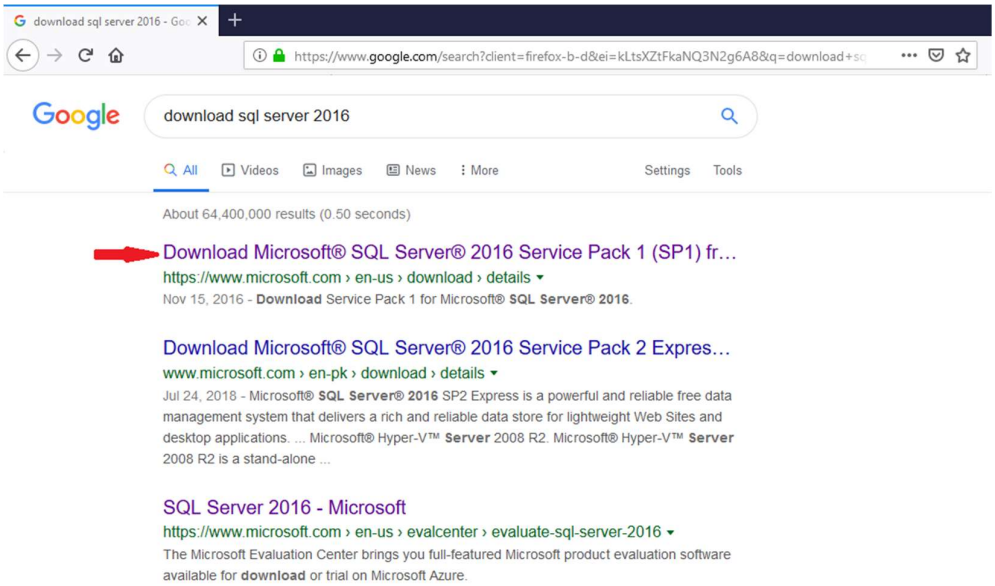
SQL SERVER 2016-Enterprise Edition or Latest

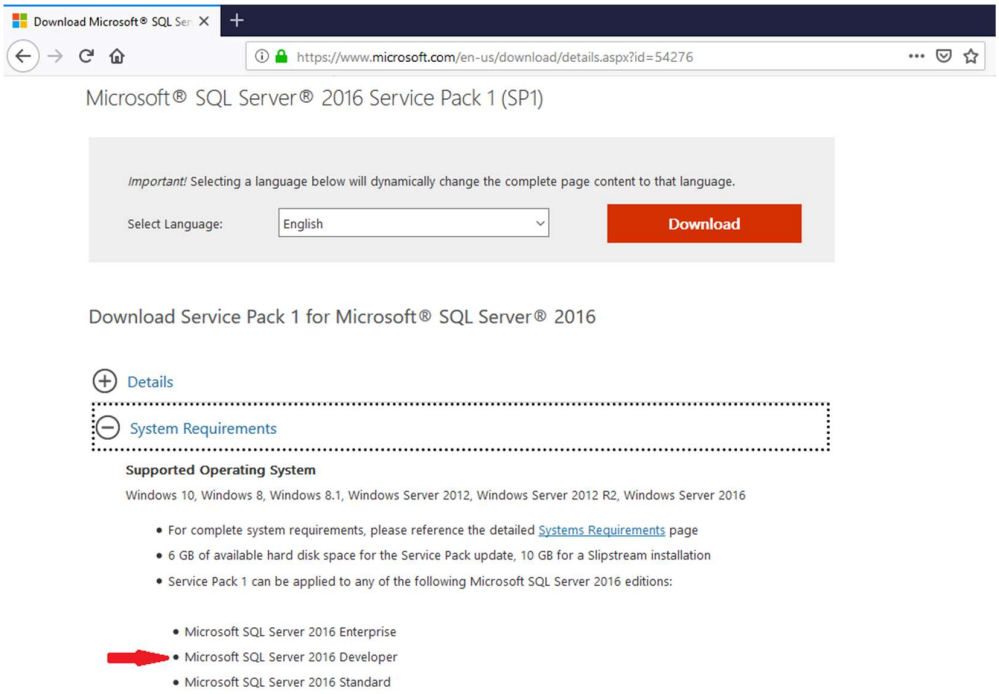
SQL SERVER Management Studio (SSMS)

SQL SERVER Data Tool (SSDT)

Finding and installing the SQL Server 2016

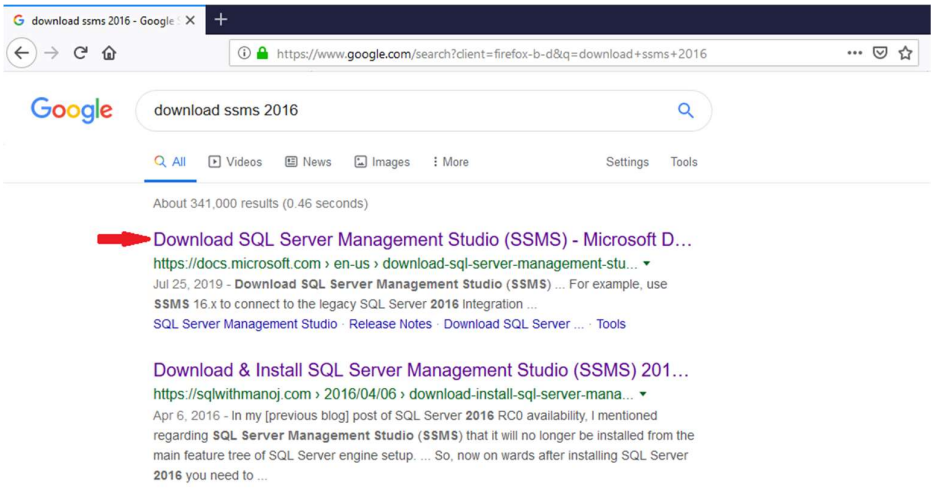
We will be downloading SQL Server 2016 Developer Edition for our data warehouse labs. The download is available on google.

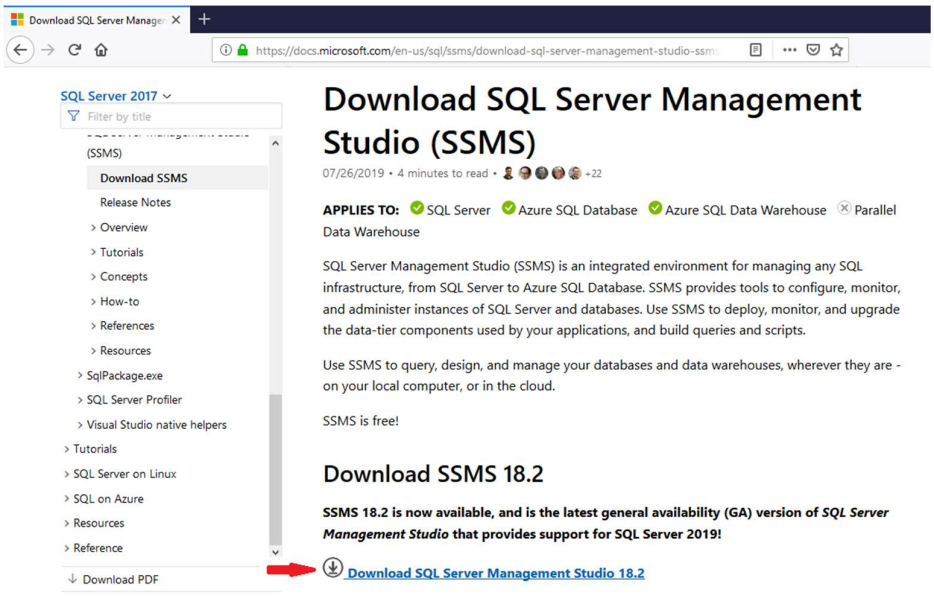




Finding and installing the SQL Server Management Studio

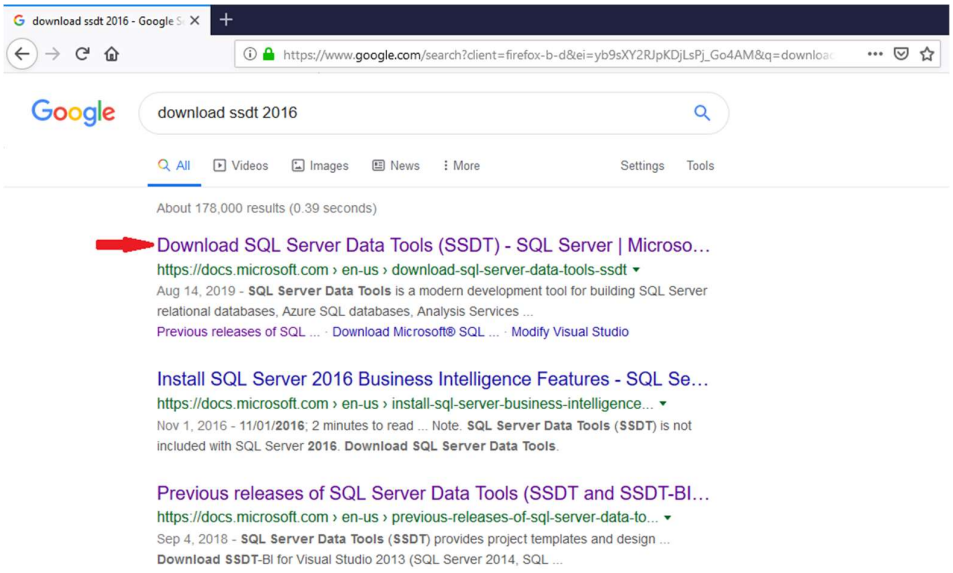
After installing the SQL Server, we need to install SQL SERVER Management Studio (SSMS). SSMS is the database management system software to run database operations.

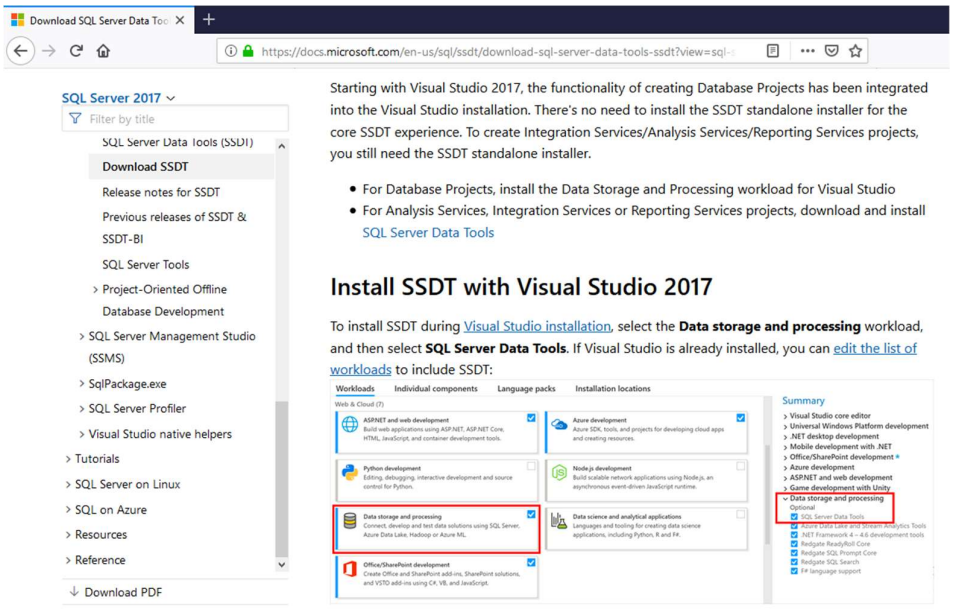


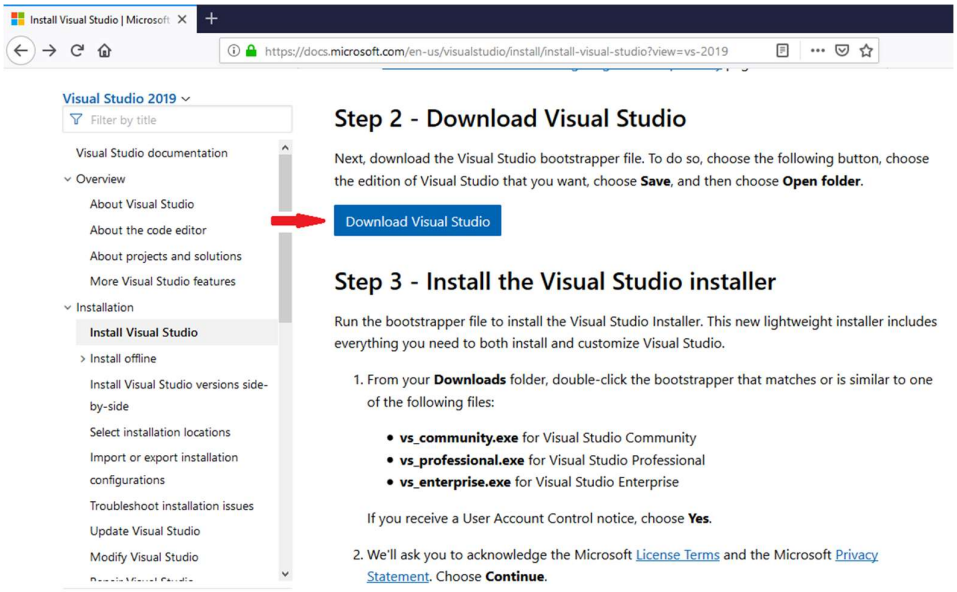


Finding and installing the SQL Server Data Tool

After installing SSMS, we need to install one more tool for the data warehouse course and that is SQL Server Data Tool (SSDT). SSDT is the tool where the ETL packages, cubes, and Reporting is done.



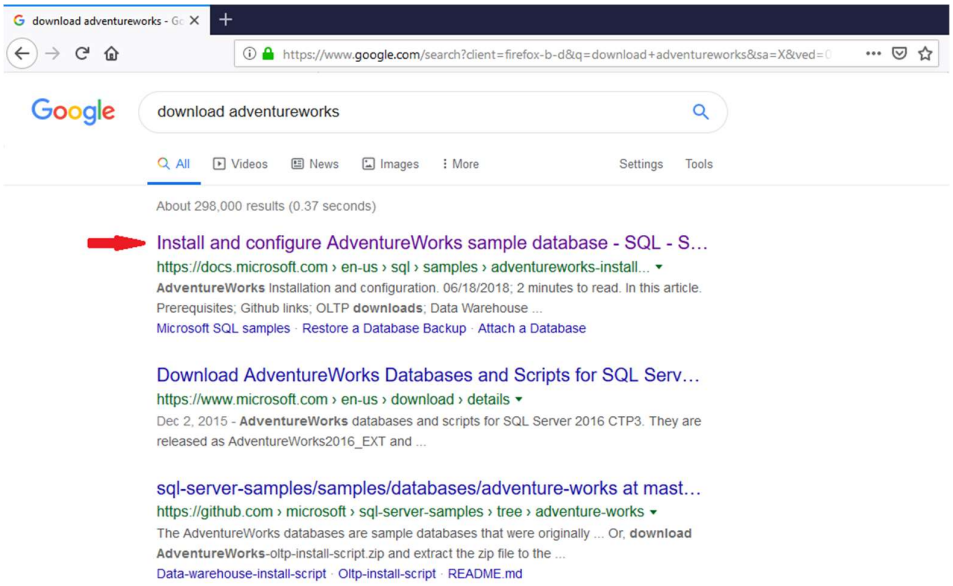


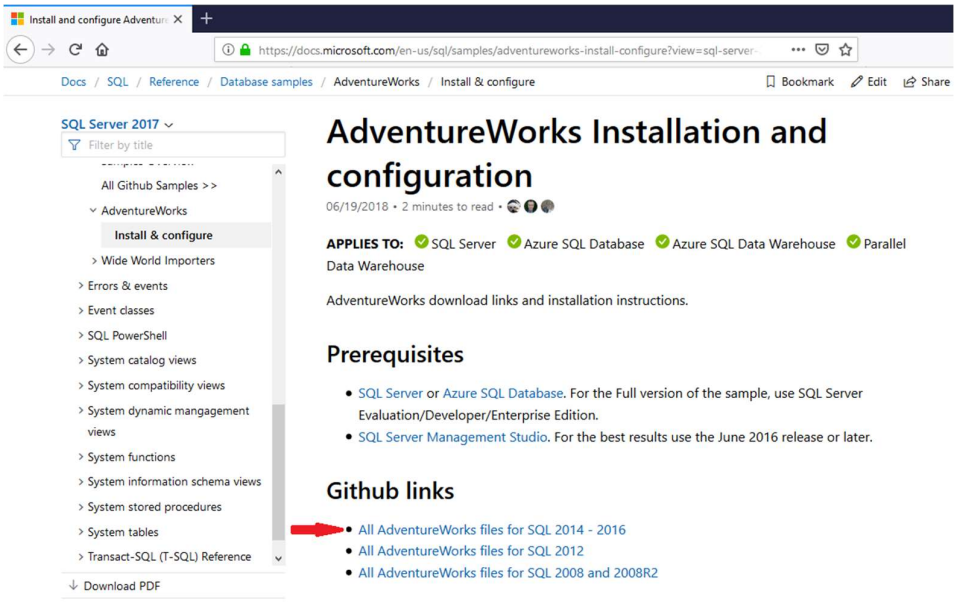


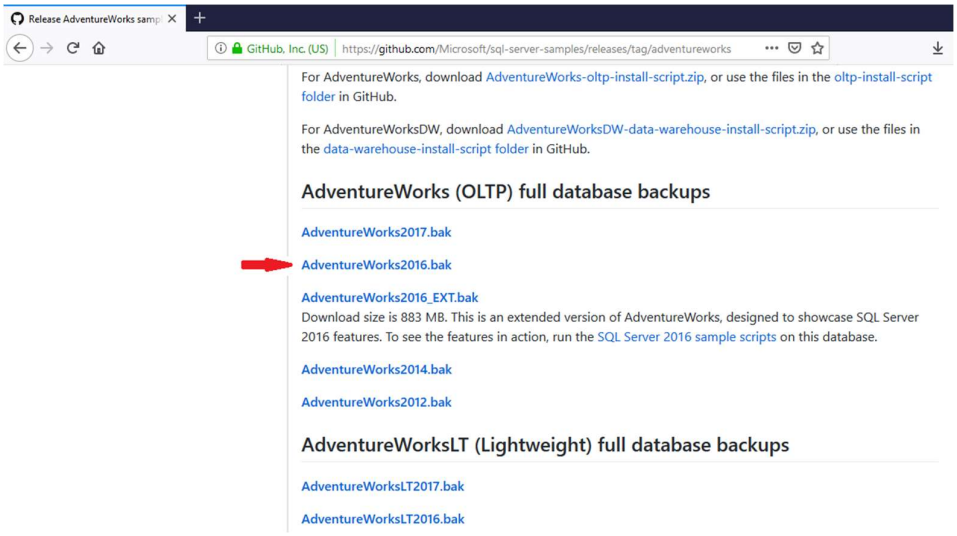
Finding and installing the sample database

After all the tools are installed and configured, we are ready to create/import the database for creating data warehouse.

We will be using Northwind and AdventureWorks database for creating dimensional model. We can download either the .bak files or the .mdf files for Northwind and AdventureWorks. For bak files you have to restore the database and for mdf files you have to attach the database.

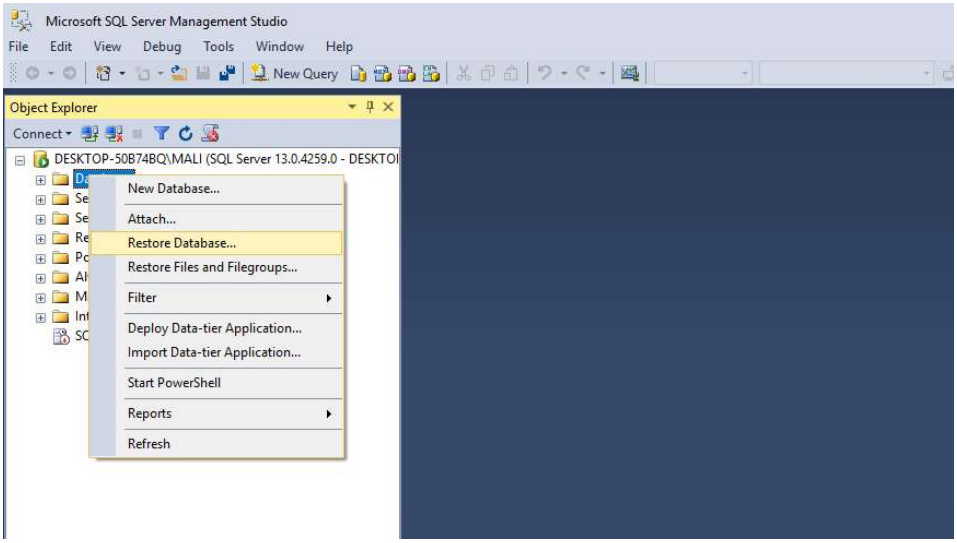

download adventureworks - Go
https://www.google.com/search?client=firefox-b-d&q=download+adventureworks&sa=X&ved=
Google
download adventureworks
All
Videos
News
Images
More
Settings
Tools
About 298,000 results (0.37 seconds)
Install and configure AdventureWorks sample database - SQL - S...
https://docs.microsoft.com › en-us › sql › samples › adventureworks-install...
AdventureWorks Installation and configuration. 06/18/2018; 2 minutes to read. In this article.
Prerequisites; Github links; OLTP downloads; Data Warehouse ...
Microsoft SQL samples · Restore a Database Backup · Attach a Database
Download AdventureWorks Databases and Scripts for SQL Serv...
https://www.microsoft.com › en-us › download › details
Dec 2, 2015 - AdventureWorks databases and scripts for SQL Server 2016 CTP3. They are released as AdventureWorks2016_EXT and ...
sql-server-samples/samples/databases/adventure-works at mast...
https://github.com › microsoft › sql-server-samples › tree › adventure-works
The AdventureWorks databases are sample databases that were originally ... Or, download AdventureWorks-oltp-install-script.zip and extract the zip file to the ...
Data-warehouse-install-script · Oltp-install-script · README.md

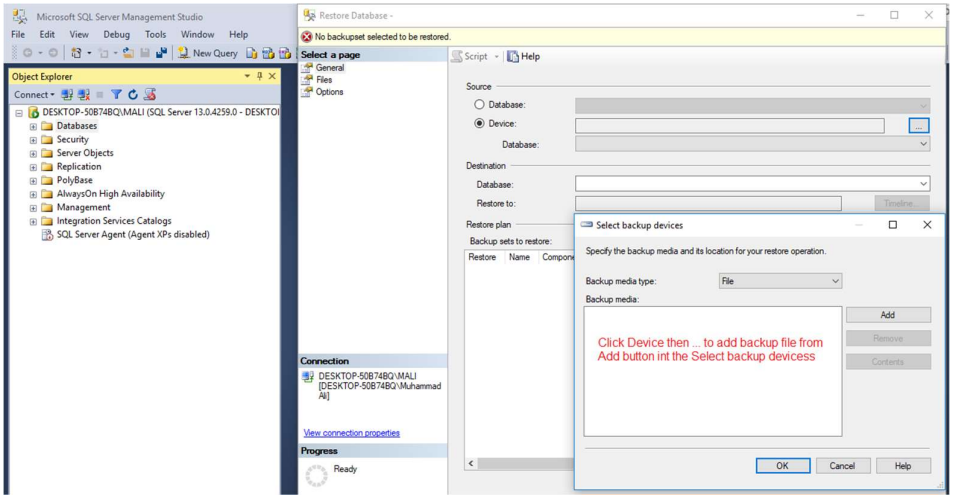

Install and configure Adventure
https://docs.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-
Docs / SQL / Reference / Database samples / AdventureWorks / Install & configure
Bookmark
Edit
Share
SQL Server 2017
Filter by title
All Github Samples >>
AdventureWorks
Install & configure
Wide World Importers
Errors & events
Event classes
SQL PowerShell
System catalog views
System compatibility views
System dynamic mangagement views
System functions
System information schema views
System stored procedures
System tables
Transact-SQL (T-SQL) Reference
Download PDF
AdventureWorks Installation and configuration
06/19/2018 • 2 minutes to read •
APPLIES TO: SQL Server Azure SQL Database Azure SQL Data Warehouse Parallel Data Warehouse
AdventureWorks download links and installation instructions.
Prerequisites
SQL Server or Azure SQL Database. For the Full version of the sample, use SQL Server Evaluation/Developer/Enterprise Edition.
SQL Server Management Studio. For the best results use the June 2016 release or later.
Github links
All AdventureWorks files for SQL 2014 - 2016
All AdventureWorks files for SQL 2012
All AdventureWorks files for SQL 2008 and 2008R2



Copy sample data file to a directory for SQL Server Data, which is Program Files, SQL Server, MS SQL Server version xx, in Backup folder. That should automatically apply the appropriate permissions.

Now open *SQL Server Management Studio* and connect to the local database server. In Databases*, right click* and select *Restore Database*

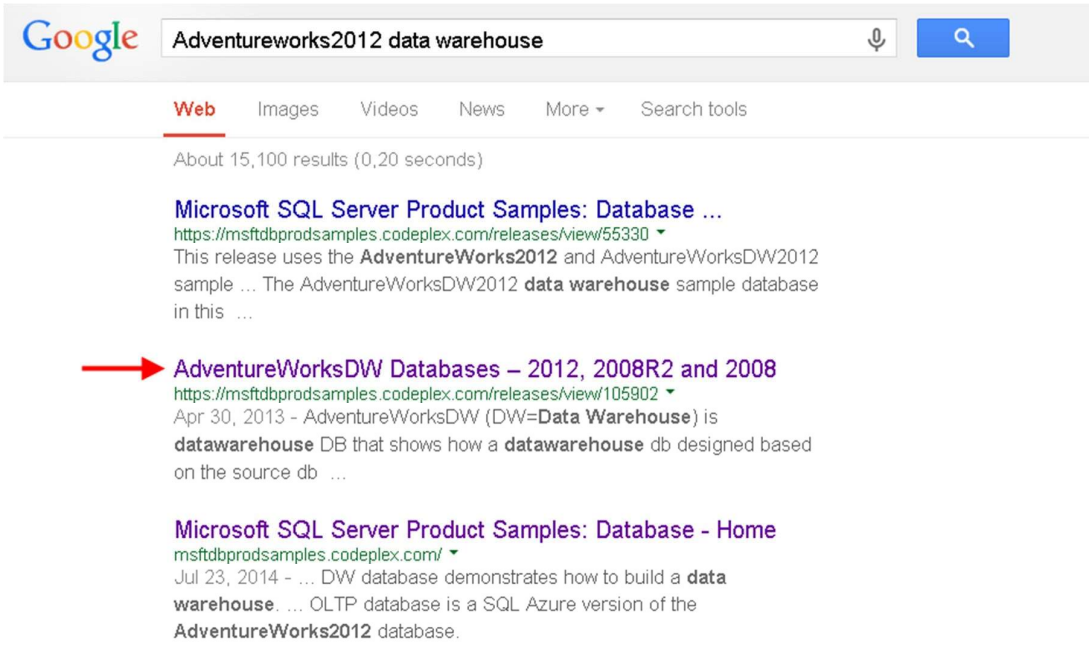


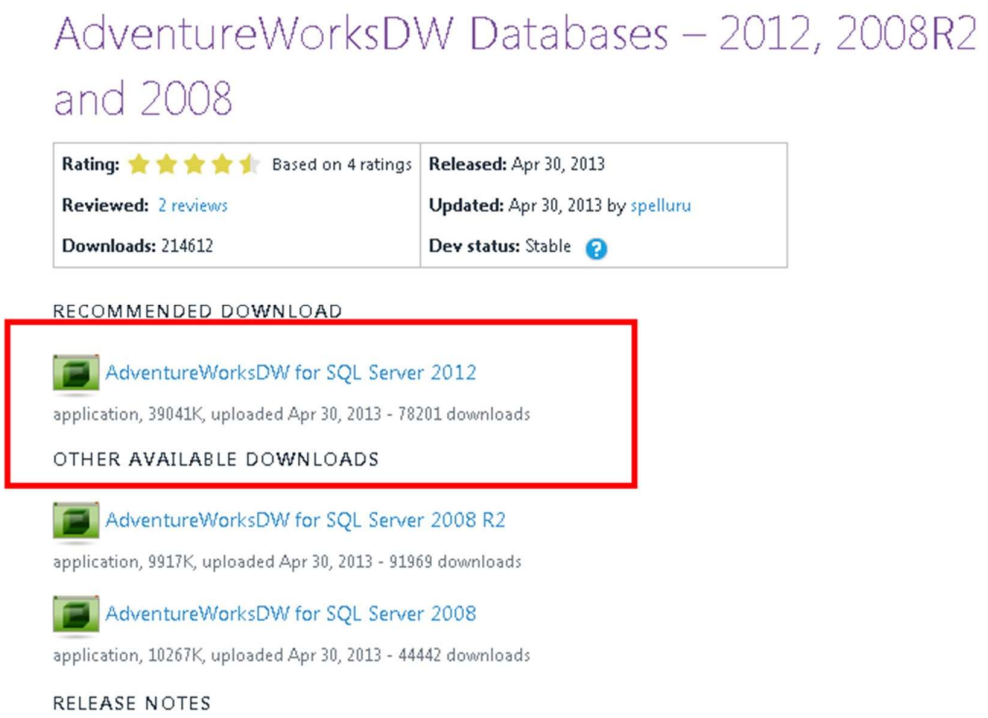


Once the backup file is loaded, we are ready to use database for creating a dimensional model.

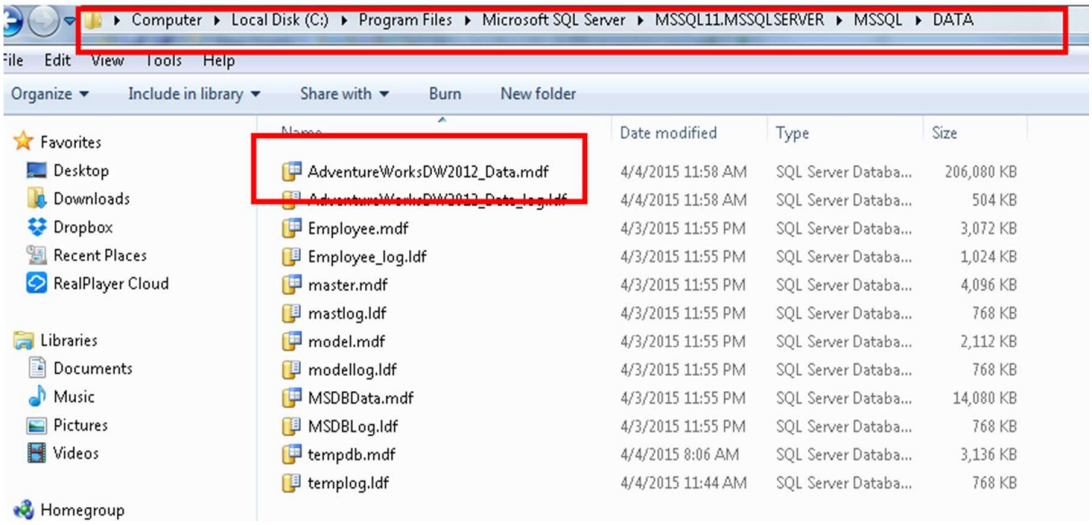
Finding and installing the sample data

Load up some sample data. The sample data used is the AdventureWorks Database, specifically the Data Warehousing version of the AdventureWorks Database.

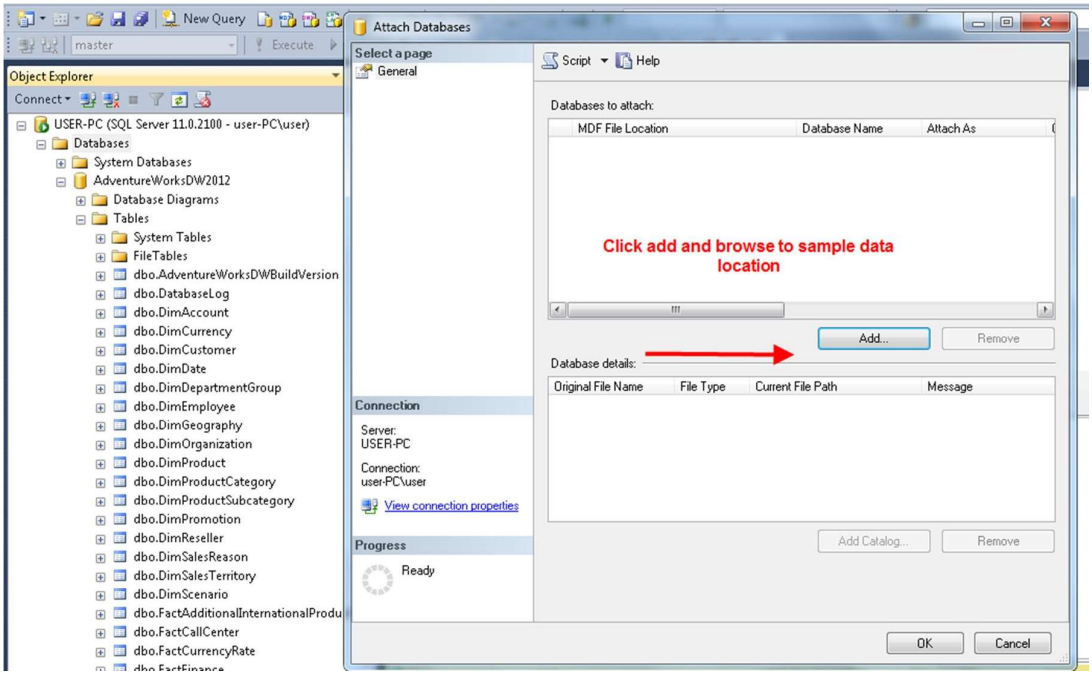




Copy sample data file to a directory for SQL Server Data, which is Program Files, SQL Server, MS SQL Server version xx, in DATA folder. That should automatically apply the appropriate permissions.



Now open *SQL Server Management Studio* and connect to the local database server. In Databases*, right click* and select *Attach*



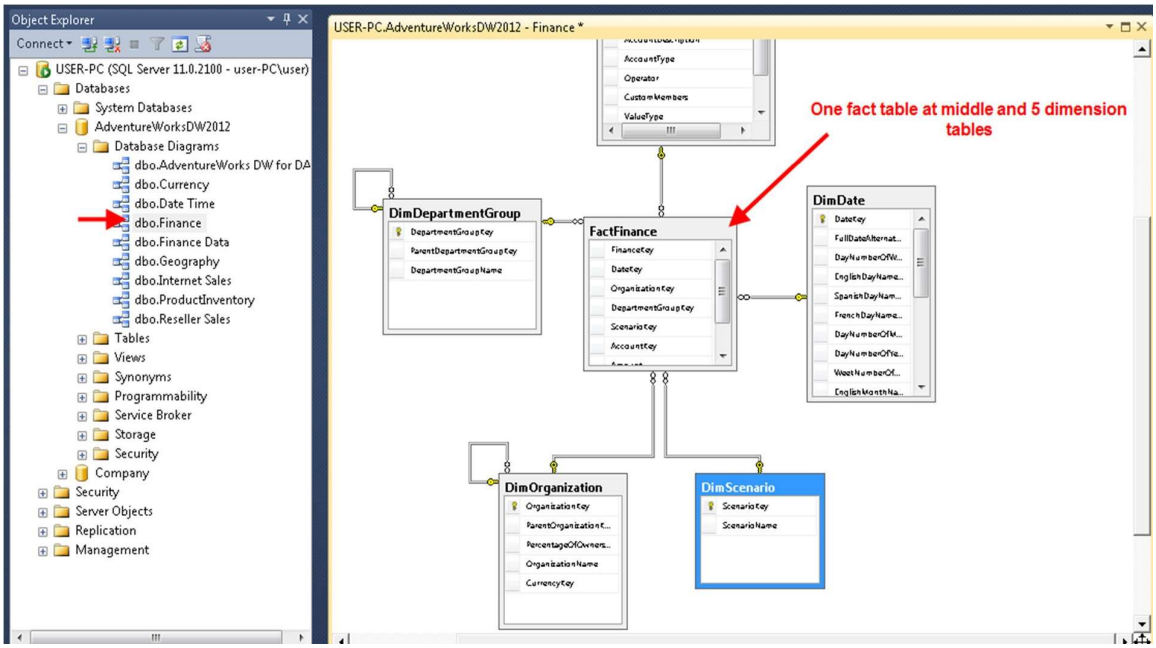
Expand the tree option for databases, AdventureWorksDW2012 will appear.

Techniques for Modeling dimension tables

There's two primary techniques ***star*** and ***snowflake***.

Star Schema

Look at the Adventureworks DW 2012 database, expand to database diagrams. Locate to *finance*, and in here you will see in the middle of the screen, one fact table called *FactFinance*, and that is surrounded by five ***dimension*** tables.

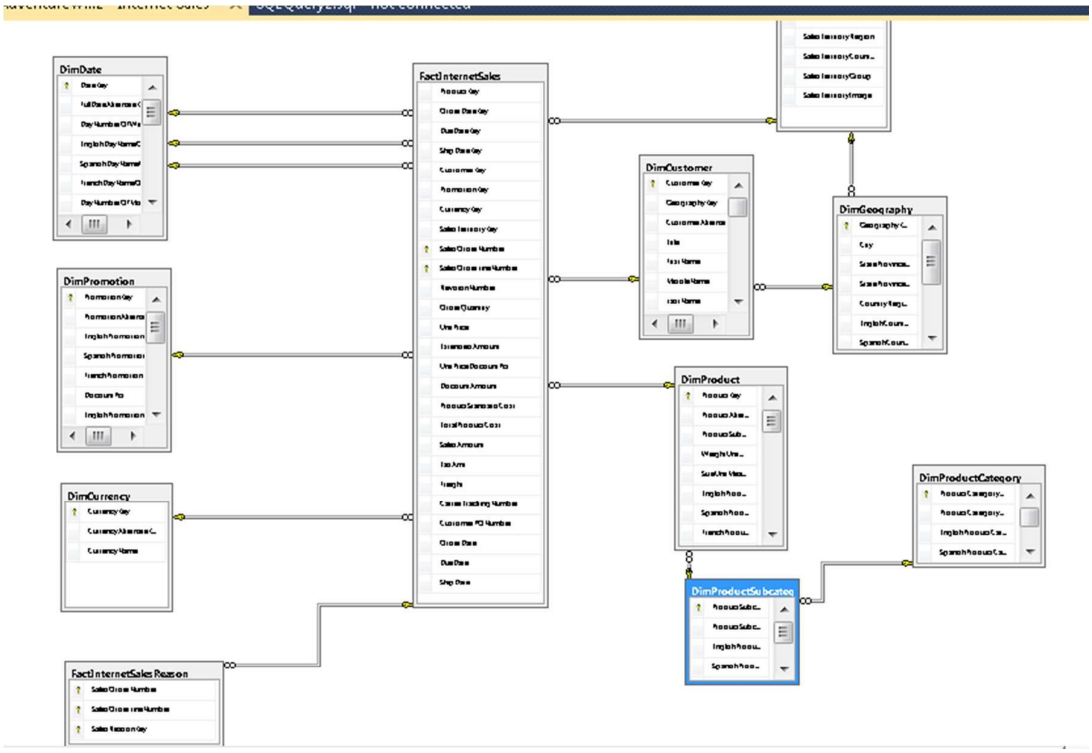


This is a common design to have multiple dimensions reference in the same fact table, and in particular, these dimension tables are not related to one another and not related to other dimension tables. They are very simple. All of the information about a dimension is contained in one table. This is called the star design.

Snowflake Schema

The other diagram is ***Internet Sales***. Here **factInternetSales** table is in the middle. Then off to the right, a connection to other dimension tables, and those dimension tables have relationships to one another, and this allows to break down a dimension into more detail, and give more options on filtering, sorting, and searching.

However, this is a more complex design. It forces to write bigger queries, more robust queries, to get the same data out of the database. It also can be a performance hit. This will create a lot more joins, and joining two large tables can be a very expensive process. Most of our dimension tables probably shouldn't be too large, but you do get into some scenarios with some large dimension tables. If we look at the ***Reseller Sales diagram***, we also see some relationships again between the different dimension tables similar to what in the *Internet Sales.*



The way schema is laid out, it could be argued that this looks like the branches of a snowflake. The dimension tables are branching off into various directions, but then those branches come back and connect to one another, and that looks a little bit like a snowflake. Therefore, this is called the *snowflake* technique.

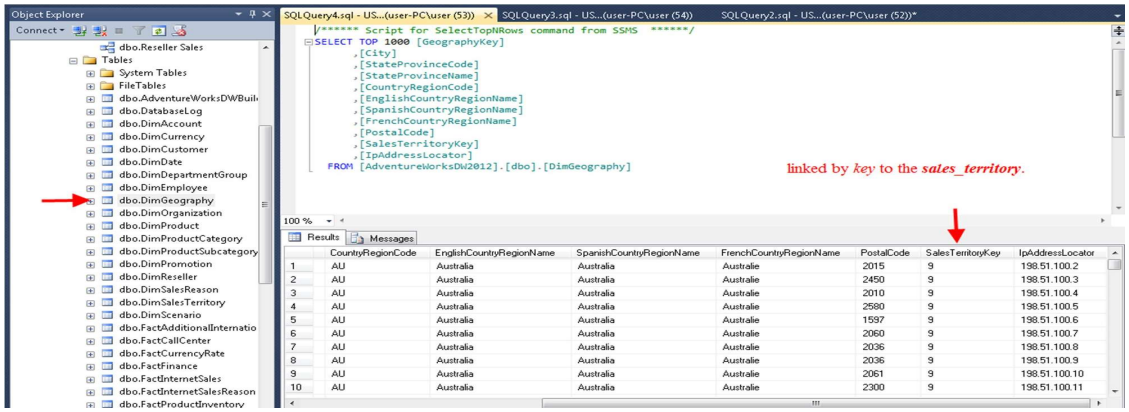
So, we have two primary techniques of structure in our dimension tables. The star technique, which is very simple. Each dimension is stored in its entirety in one table, and then contrast that with the snowflake technique.

In the snowflake technique, a dimension is split up amongst multiple tables, each one has some advantages and disadvantages. The star is the simpler way to go, and therefore will typically give better performance, and is easier to write queries against. The snowflake is a more complex design, will be more difficult to write queries, will possibly give slower performance, but it could allow us to have more robust dimensions. Realistically, most data warehouses have some of both.

It's very rare to see a data warehouse that's 100% star or 100% snowflake. Typically some of your dimensions can easily be captured in one table, and you can go ahead and use the star method there, and other dimensions just logically require multiple tables and you can use the snowflake technique there. So, you can mix these two techniques, and that is very common.

Creating view to improve implementation of data warehouse

Now let's see how SQL server views can improve the implementation of data warehouse. The first situation we select ***dimGeography*****,** that has a snowflake relationship to another dimension called **dims*alesterritory*.** Both of these dimensions are interesting in and of themselves, but they're also interesting when combine the data from the two tables. It is easier to create a view that pulls in a data from each table allowing us just to have one entity to look at all of the data. First lets look at the *geography dimension*. Right click and select the top rows.



You may noticed the geography dimension, contains information about countries, cities, states. Scroll all the way to the right, you will see it is linked by *key* to the ***sales_territory***.

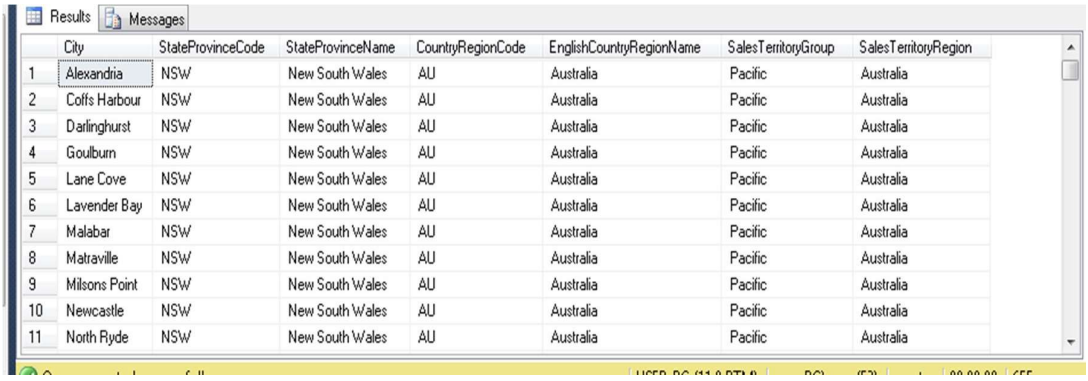
Right click *on salesterritory* table and look at the top rows and observe it contain the *Northwest United States*, *Northeast United States, Central United States*. Different sales territories, but this doesn't tell us ***what cities*** are in what territory.

In order to know what cities, or in what territory we need to look at both dimension sales territory, and the dimension geography. We can create a view to make that easier by using following code:

```
CREATE VIEW  [dbo].[Total_DimTerritoryAndGeography]
AS
SELECT dbo.DimGeography.City,
dbo.DimGeography.StateProvinceCode,
dbo.DimGeography.StateProvinceName,
dbo.DimGeography.CountryRegionCode,
dbo.DimGeography.EnglishCountryRegionName,
dbo.DimSalesTerritory.SalesTerritoryGroup,
dbo.DimSalesTerritory.SalesTerritoryRegion FROM    dbo.DimGeography
INNER JOIN dbo.DimSalesTerritory ON
dbo.DimGeography.SalesTerritoryKey =
dbo.DimSalesTerritory.SalesTerritoryKey
```

This will create a view that references the most interesting fields for both tables, the city, state, country from the geography table and also the salesterritory group and sales territory region from the territory table. The connection between the two is very simple. It is the sales territory key field, which exists in both tables.

So run this it may says command completed successfully, so now check view under list of views, if not appear refresh view. Right click newly created view Total_DimTerritoryAndGeography view and select top rows



Results | Messages

	City	StateProvinceCode	StateProvinceName	CountryRegionCode	EnglishCountryRegionName	SalesTerritoryGroup	SalesTerritoryRegion
1	Alexandria	NSW	New South Wales	AU	Australia	Pacific	Australia
2	Coffs Harbour	NSW	New South Wales	AU	Australia	Pacific	Australia
3	Darlinghurst	NSW	New South Wales	AU	Australia	Pacific	Australia
4	Goulburn	NSW	New South Wales	AU	Australia	Pacific	Australia
5	Lane Cove	NSW	New South Wales	AU	Australia	Pacific	Australia
6	Lavender Bay	NSW	New South Wales	AU	Australia	Pacific	Australia
7	Malabar	NSW	New South Wales	AU	Australia	Pacific	Australia
8	Matraville	NSW	New South Wales	AU	Australia	Pacific	Australia
9	Milsons Point	NSW	New South Wales	AU	Australia	Pacific	Australia
10	Newcastle	NSW	New South Wales	AU	Australia	Pacific	Australia
11	North Ryde	NSW	New South Wales	AU	Australia	Pacific	Australia

Now see city, state, territory group and territory region all on one line. This should make it easier and more convenient for developers to look at both of these dimensions at the same time. The other scenario where views can improve the implementation of our data warehouse comes with aggregating data. So, information like sales, profit, revenue, expense. Usually we don't want to look at those things line by line, we want to look at totals.

And maybe it's a total for a *week*, or a total for a *month*, or a total for a year but there's probably going to be some sort of grouping. So, creating a view with a *group by* clause can help us get started in that aggregation.

So, again, below is some code staged that's going to create a view.

```
CREATE VIEW  [dbo].[Total_FactInternetSales]
AS
SELECT SUM(DiscountAmount) AS Total_DiscountAmount,
                    SUM(ProductStandardCost) AS Total_ProductStandardCost,
                    SUM(TotalProductCost) AS Total_TotalProductCost,
              SUM(SalesAmount) AS Total_SalesAmount,        OrderDate,
              CustomerKey,
CurrencyKey
FROM          dbo.FactInternetSales
GROUP BY OrderDate, CustomerKey, CurrencyKey
```

Its going to sum four different fields. All of them are dealing with currency, discount amount, product cost, total product cost, and sales amount and group by three different fields, the auto date, the customer key, and the currency key. So, that will allow to run reports on a certain time frame, and or on a customer or group of customers, and or in certain currencies.

Let's run and create this view by executing the code. Select top rows from it and now we can see the data that was returned. This is now a convenience for developers, they no longer have to manually set up the group by fields, they can just pull off of this view.

Results | Messages

	Total_DiscountAmount	Total_ProductStandardCost	Total_TotalProductCost	Total_SalesAmount	OrderDate	CustomerKey	CurrencyKey
1	0	11.2163	11.2163	29.99	2008-03-05 00:00:00.000	17822	100
2	0	432.8647	432.8647	804.48	2008-06-27 00:00:00.000	15547	100
3	0	1518.7864	1518.7864	2443.35	2007-06-13 00:00:00.000	24506	6
4	0	12.0728	12.0728	32.28	2007-08-12 00:00:00.000	23088	100
5	0	1291.7958	1291.7958	2389.98	2008-05-15 00:00:00.000	12827	100
6	0	1082.51	1082.51	1700.99	2008-05-17 00:00:00.000	21986	6
7	0	1518.7864	1518.7864	2443.35	2006-07-04 00:00:00.000	13085	100
8	0	1105.81	1105.81	2049.0982	2006-09-11 00:00:00.000	28559	100
9	0	474.5311	474.5311	777.34	2007-11-09 00:00:00.000	28854	100
10	0	1269.3558	1269.3558	2329.98	2008-06-17 00:00:00.000	17767	100
11	0	2171.2942	2171.2942	3578.27	2005-09-30 00:00:00.000	17326	6

Adding Index to views:

This implementation however would not provide a performance increase. If we want to provide a performance increase, we'll need to do is attach an ***index*** to this view. The way the view is now with no indexes, it does not create a new copy of the data. And every time we select from

the view, it goes to the tables, pulls the data, performs all the necessary math, and then displays the data to the user. If we had an index to the view, that forces the machine to create a secondary copy of the data that's already aggregated.

So, now when someone wants to look at this view, it doesn't go back to the tables and do all the math again. It will just look directly at the view with the pre-aggregated totals. So, that would be possibly a significant performance increase on many of our queries but, be aware it could be a performance decrease when we have to load new data. Because every time we load new data into this data warehouse, we are going to have to update the view. The machine will handle that automatically but it will increase the time it takes to load data.

```
ALTER VIEW  [dbo].[Total_FactInternetSales]
WITH SCHEMABINDING
AS
SELECT SUM(DiscountAmount) AS Total_DiscountAmount,
                    SUM(ProductStandardCost) AS Total_ProductStandardCost,
                   SUM(TotalProductCost) AS Total_TotalProductCost,
                 SUM(SalesAmount) AS Total_SalesAmount,
                        OrderDate,
              CustomerKey,
              CurrencyKey,
               COUNT_BIG(*) as RecordCount
 FROM         dbo.FactInternetSales
GROUP BY OrderDate, CustomerKey, CurrencyKey
```

Two changes are added to view:

schemabinding which creates a relationship between view and the underlying tables preventing a change from one, without changing the other.

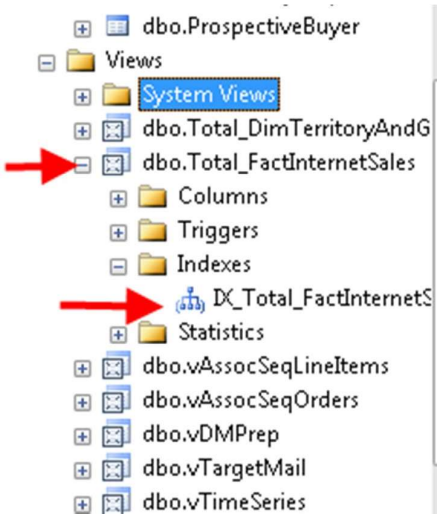
COUNT_BIG count the number of records and this is a requirement in order to add an index onto any view that has a *GROUP BY*.

execute this and see command completed successfully. Now add an index to this view using following query:

```
CREATE UNIQUE CLUSTERED INDEX  [IX_Total_FactInternetSales]
ON [dbo].[Total_FactInternetSales] (OrderDate, CustomerKey, CurrencyKey)
```

Its create a unique clustered index because the first index on a view has to be a unique clustered index. After that you can create non-clustered. Here it indexed on *order date*, *customer key*

and *currency key* . Execute that and now over in the views we can expand it, look at the indexes, and we do in fact see one index coming up under view.



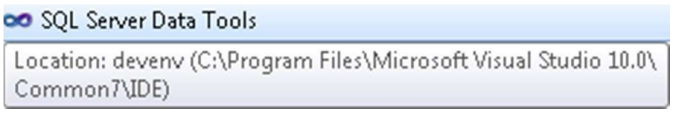
Index is added on view. In the background the machine had made a secondary copy of the data. So, we still have the original data in the original tables. We also have a secondary copy of that data in the view that is pre-aggregated. So, if we want to make a call to this view, the machine doesn't have to do all of the math to do the group by, it just has to read the data that it has aggregated. That can be a significant performance increase to us.

Introduction to ETL with SSIS

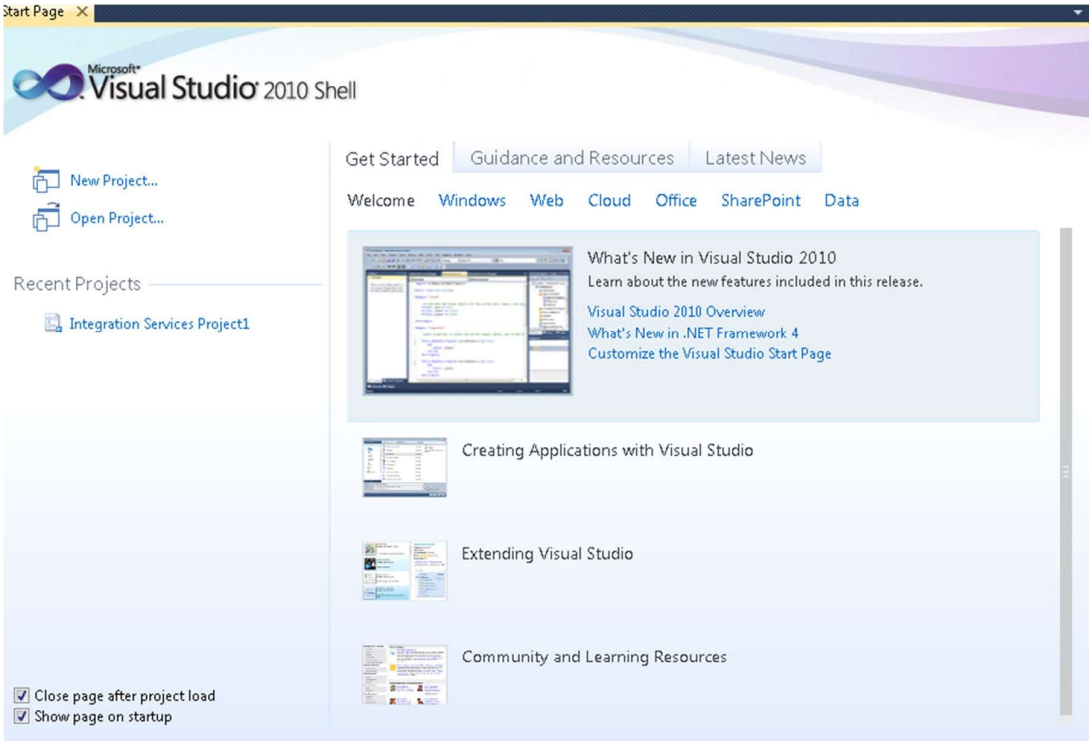
Now let's start **SQL Server Integration Services (SSIS)** to develop an ETL solution to extract transform load. ETL is a very important part of a data warehouse when you first setting up a data warehouse a lot of your work goes into designing and building the table structure that will hold the data. Once that's done, it typically doesn't change very often and therefore there's not a lot of maintenance to be done there. On the other hand, the ETL process often does have some changes and some maintenance that needs to be done.

ETL is often importing data from an external entity, in other words data we have a little or no control over. Now when something peculiar happens to that data, ETL process has to react to that. ETL process is often changed after the initial deployment. A lot of the day-to-day work of the data warehouse administrator does involve ETL. In order to develop ETL process, integration services will be used tool that is called ***SQL Server Data Tools (SSDT).***

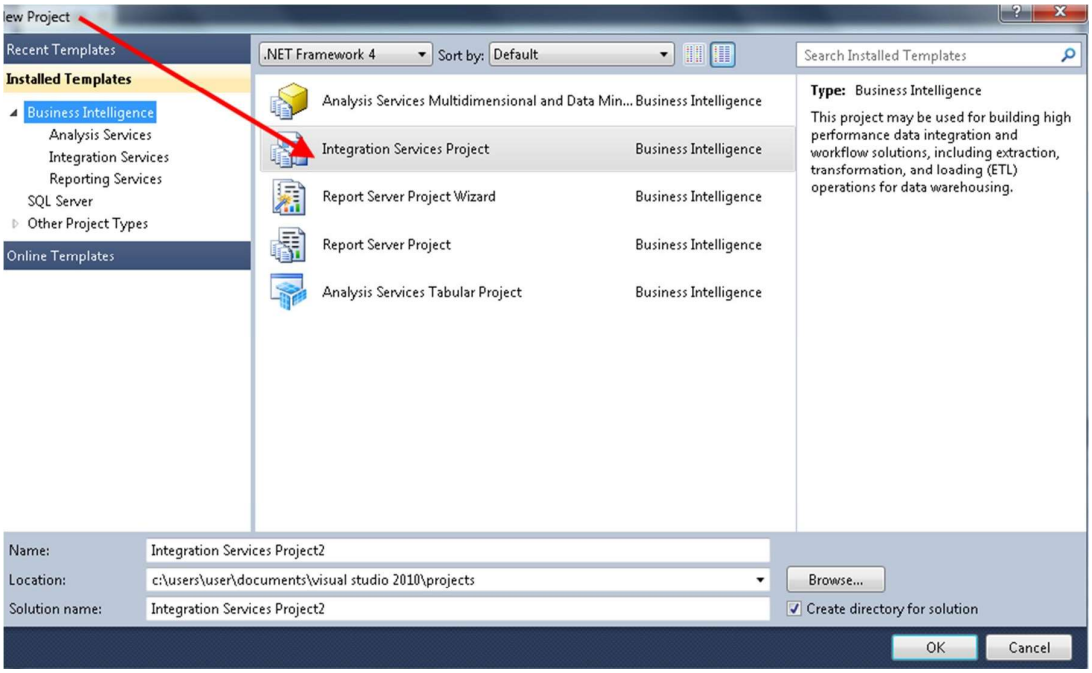
In previous releases of SQL Server, this was called *Business Intelligence Studio*. The tool has been renamed to ***SQL Server Data Tools***. From program locate to SQL Server Data Tools:



This is the starting page for SQL Server Integration Tools

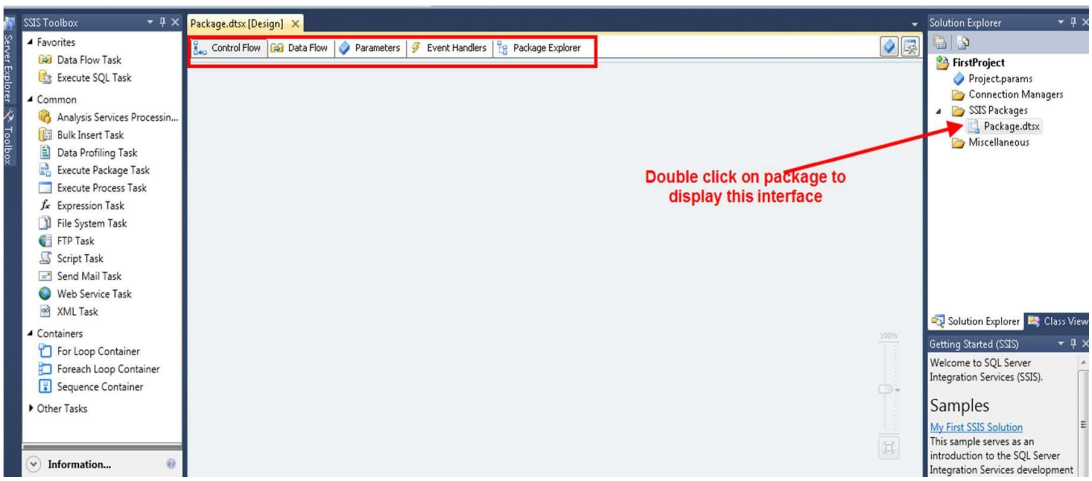


Create a *new project* and look for an *integration services project*. At the bottom it will ask for a name and location. In this instance just accept the defaults and this takes you to an interface where we can develop the package.



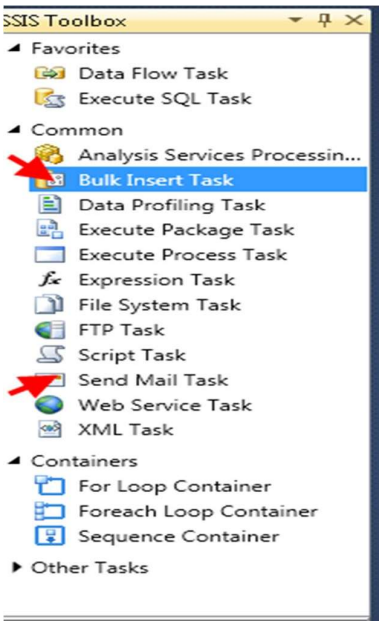
In SQL Server Integration Services primary unit of work is called ***a package.*** A package can handle an entire ETL solution, or you could do just one part of it in one package and an additional part in a different package. For example, one package for *extract*, a second package for *transform*, and a third package for *load*.

In this exercise we're going to include everything in one package. Double click on package, it gives the interface presented following.

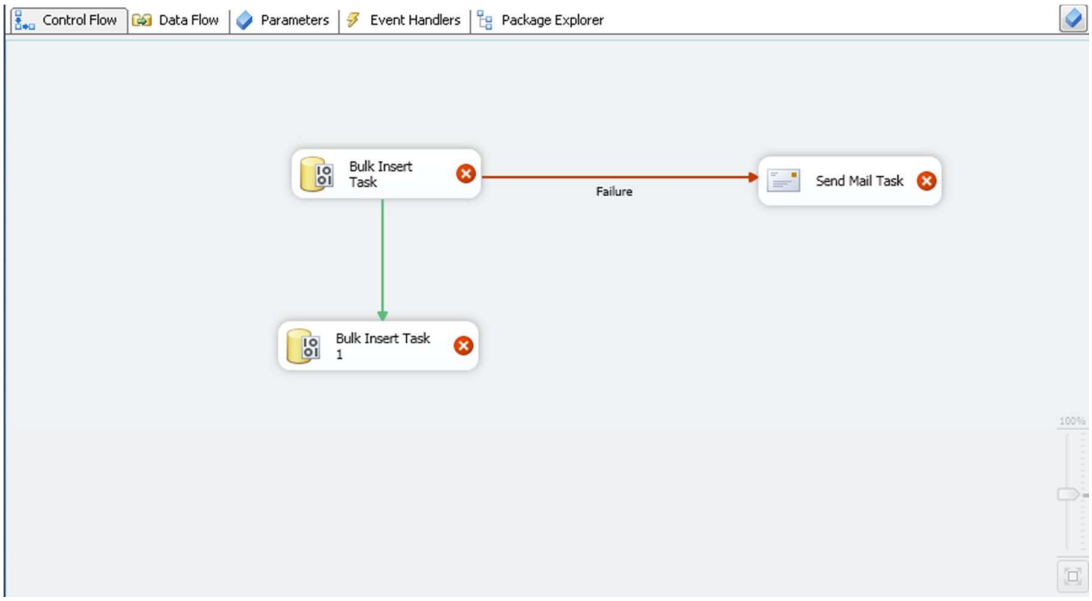


We can choose *Control Flow*, *Data Flow*, *Parameters, Event Handlers, and Package Explorer.*

Select **Control Flow**, and over to the left a tool box where drag certain tools onto the work area that will allow to accomplish certain tasks. Common thing we do in an ETL solution is a ***bulk insert***, drag ***Bulk Insert Task***.



Right-click on it, look at the properties and configure different properties to allow *Bulk Insert Task* to find some data and insert it somewhere.



Green arrow that's coming out of the bottom of the box are called ***Precedent constraints.*** They allow package to flow from one task to another.

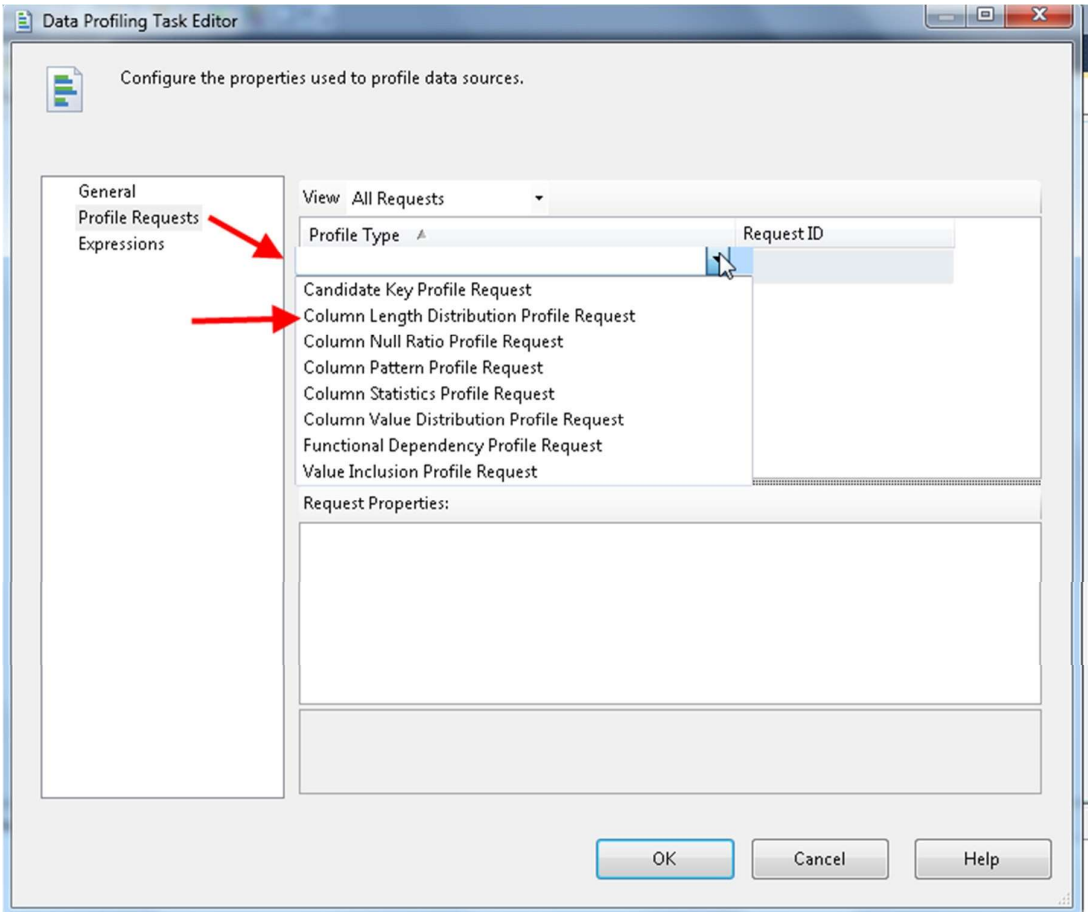
The green arrow, the default arrow is a ***success constraint***. What actually here is that if the bulk insert succeeds, it will move on to the next ***Bulk Insert***. We may want to do something if the task fails. For example, send an email to an administrator in that case drag a ***send mail task*** on to the screen.

From the Bulk Insert Task drag another precedence constraint to the **Send Mail Task.** Now it's currently green meaning it is success. *Rright-click* on it, and change it to failure turned it red, and it is labelled with the phrase, ***Failure***. Now run package, the Bulk Insert Task will run. If it's succeeds, it will follow the green arrow and go to the second *bulk Insert Task*. If it fails, it will go to the *send mail task* following the red arrow or the red president constraint.

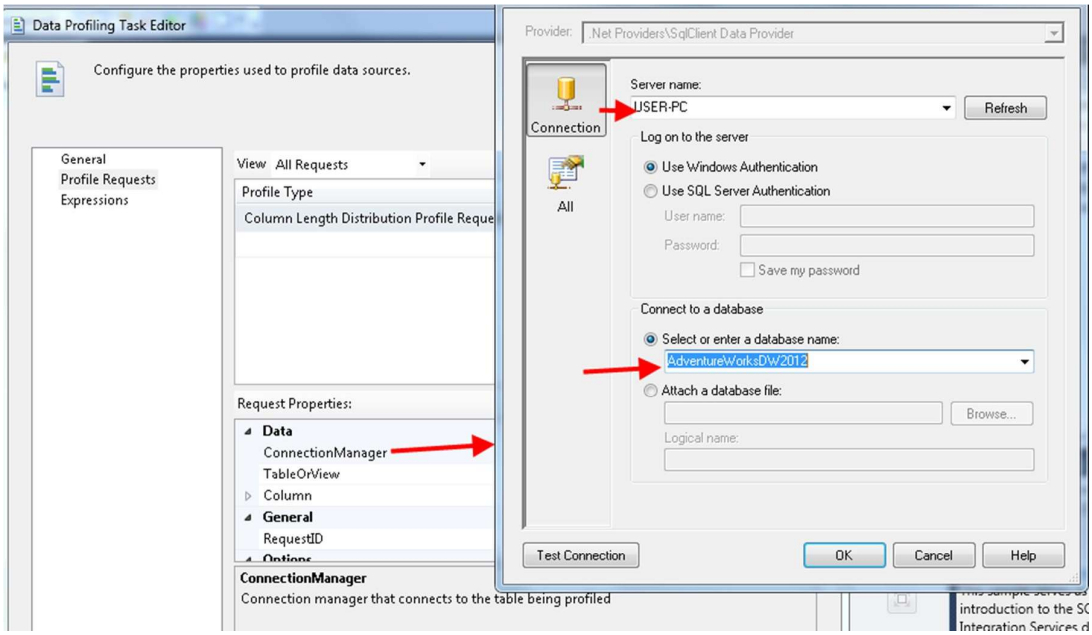
Exploring source data

When we have a new data source, we need to integrate into our ETL solution. There is some concerns about data e.g. what data types are being used? How often is a column null? How often column has a very small value or a very large value? SQL server integration services have some tools that can help to explore the data and answer these types of questions.

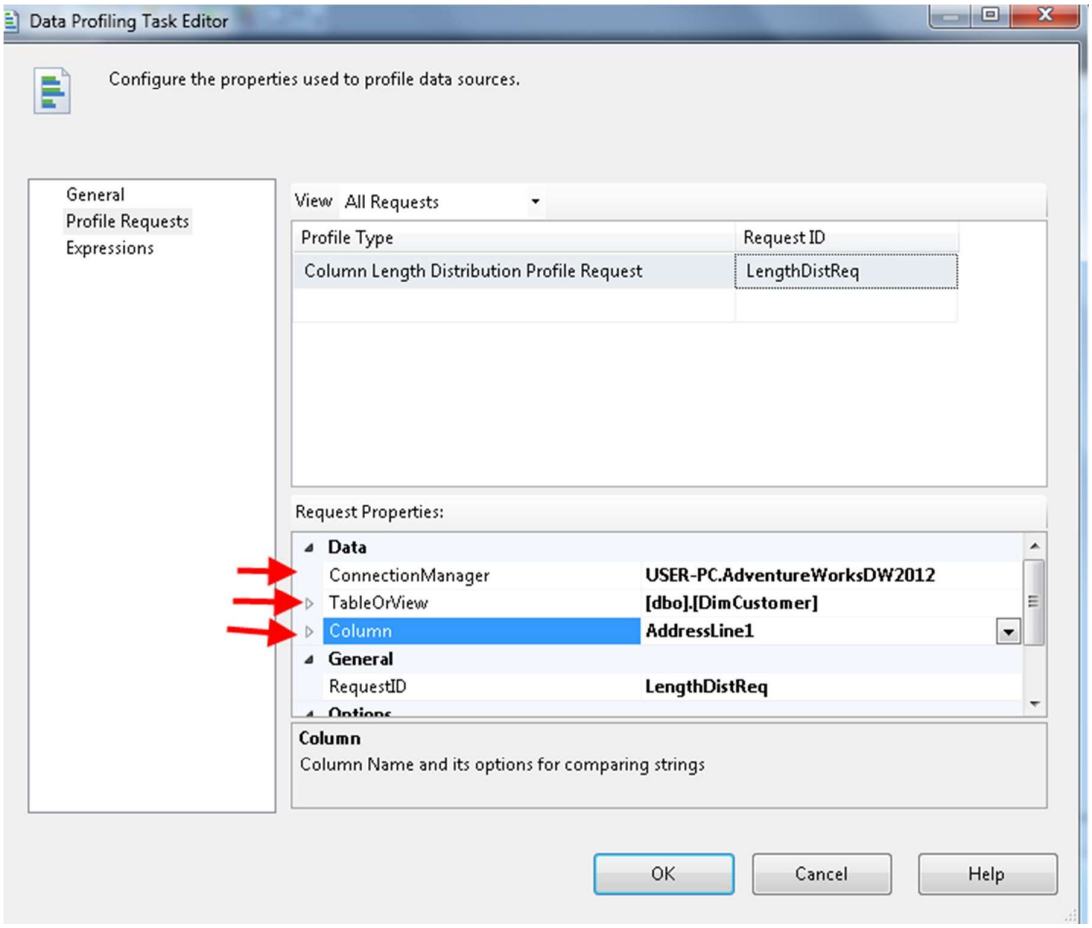
- Open SQL server data tools and create a new project it would be an integration services project.
- In the tool box locate ***data profiling task***, drag that onto the screen.
- Double-clicking to get some options on configuring.
- Explore ***profile request section*****,** look at is a column *length distribution profile request,* which give information about a particular column of data; how often have large value, how often have small values.



- Once click on RequestId it will select ***LengthDistRequest***. At the bottom it ask to make a connection and make a new connection; to an existing SQL server look for A***dventureworksdw2012*** database.

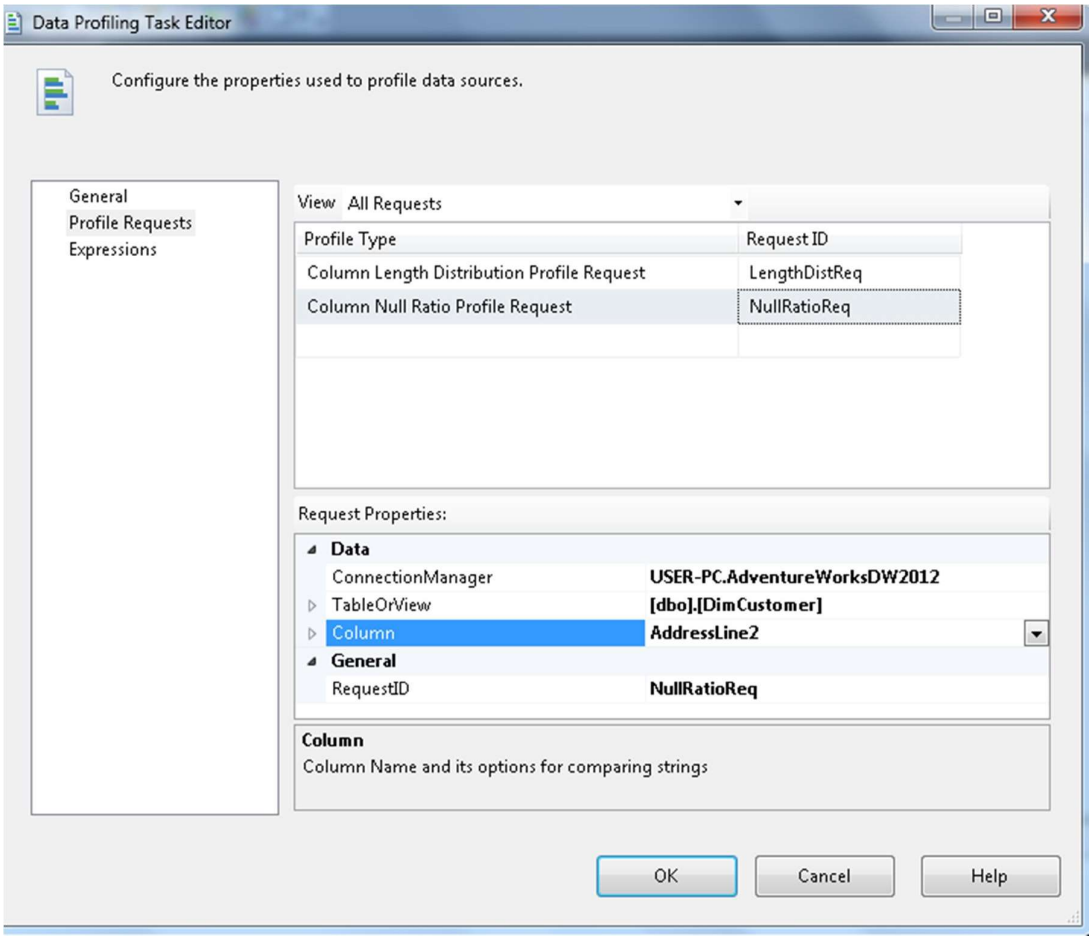


- Next is table which you are interested in and in this case it is the **customers** table which in database is named *dimcustomer* because it is a dimension and next select column to ***addressline.***
- Click ok

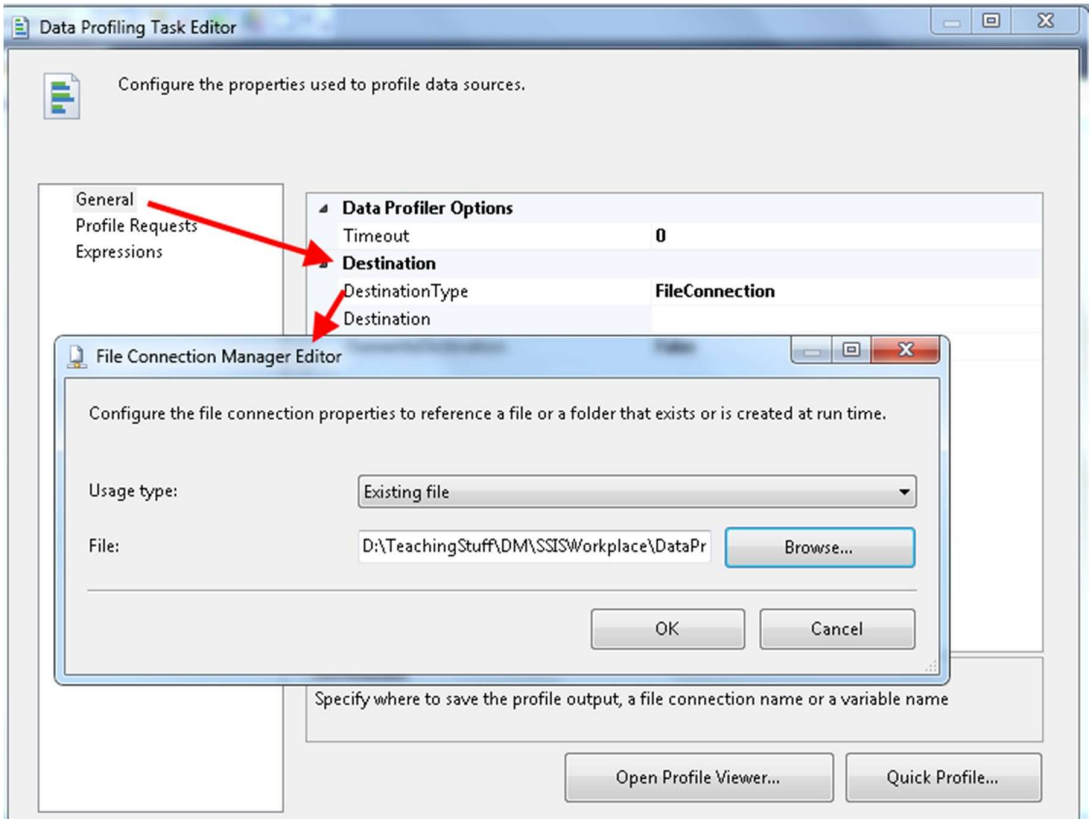


Add one more request by repeating the above step and that will be a column ***nullratioprofilerequest***. It simply answering the question how often is this particular column have the null value. Use connection and table as previously.

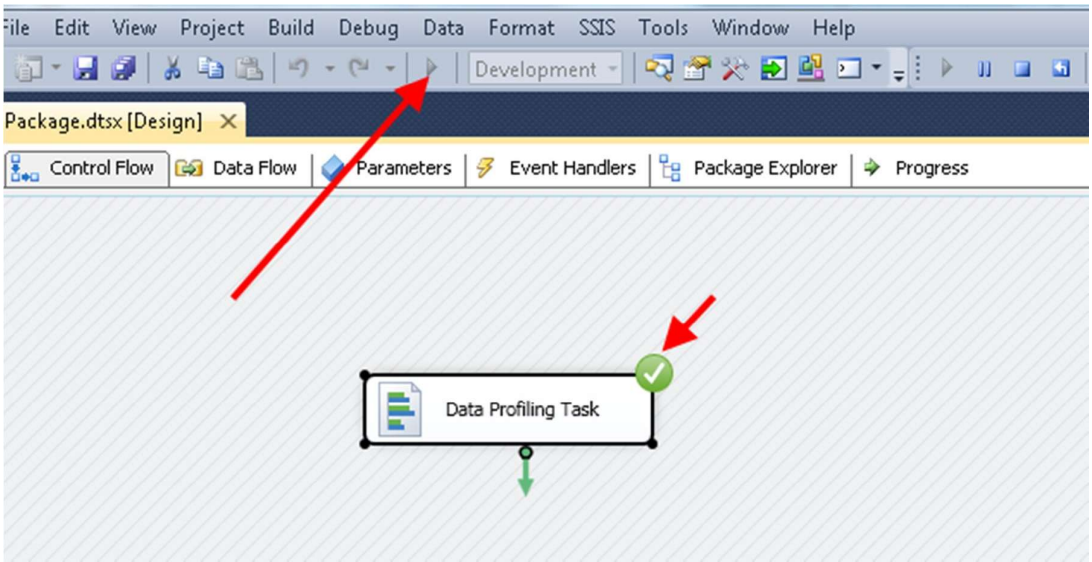
But select different column *Addressline2* rather than address line1.



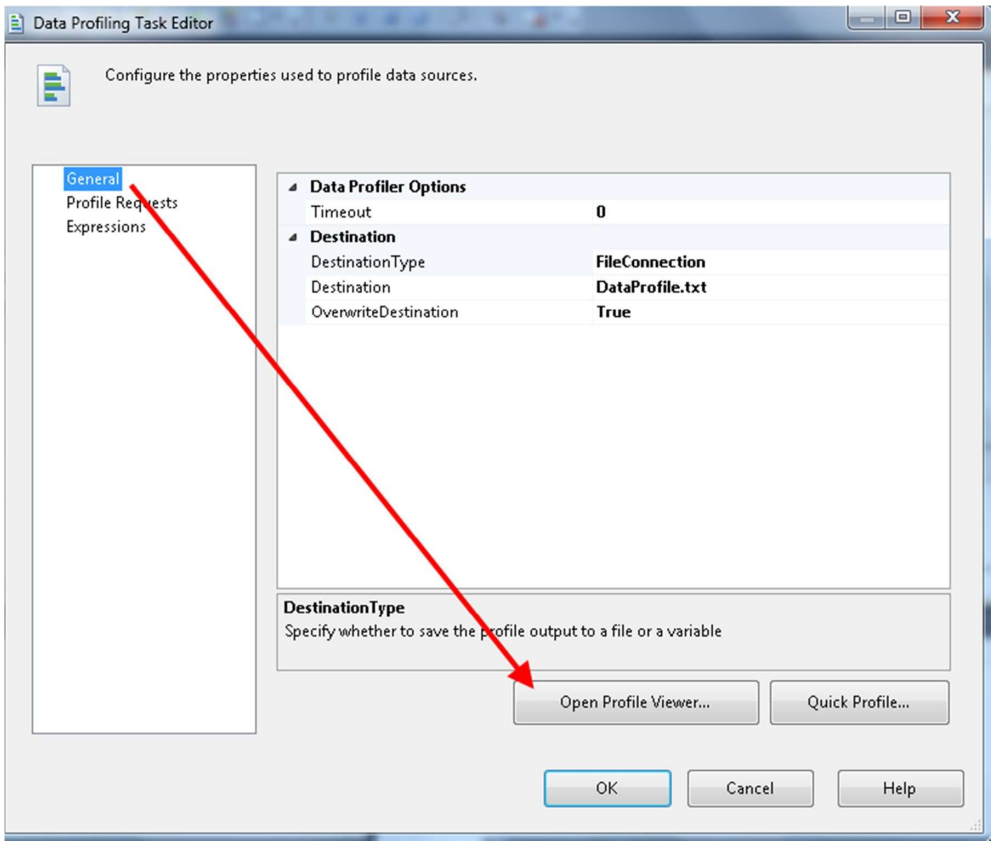
- It also need a destination for this data, SQL Server Integration Services would like to write this to an XML file. So on the General section set a destination.
- This needs to be an existing file already on your machine or create a new file I have one called *Data Profile* and change the overwrite to True to overwrite that file.



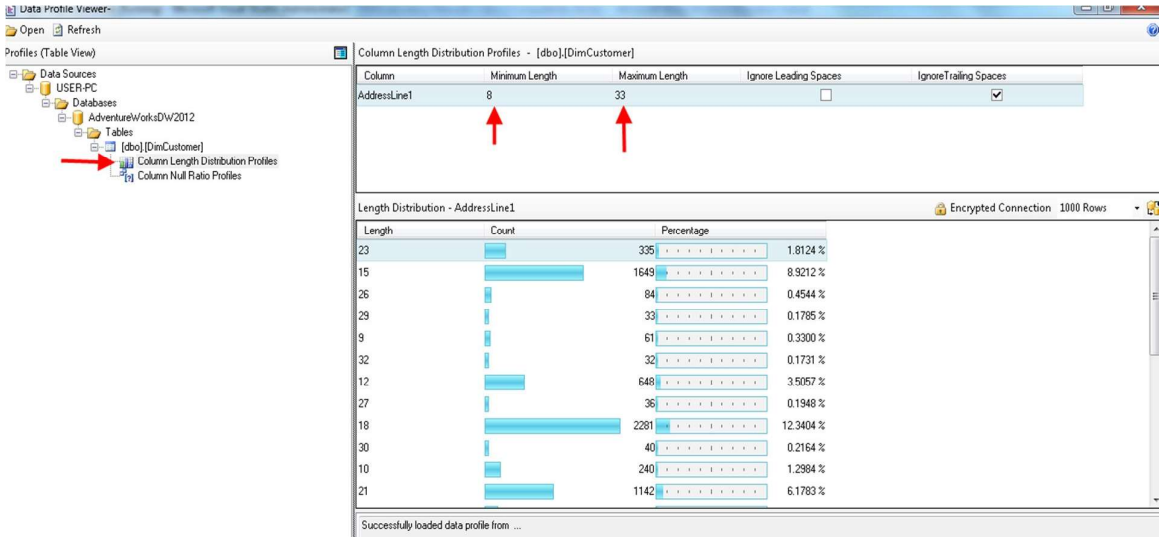
- Now ***data profile task ready*** to go. Click OK
- In order to execute the package, on the toolbar look for green triangle pointing to the right. And that is the play button.
- Package will run on the Data Profiling Task. We now have a green check mark, implying it ran successfully.



- While it is still running double click the Data Profiling Task, that will bring up the interface where we configured it before and look for option in the lower right to *Open Profile Viewer.*



- Maximize this and select is the column distribution profile, for address line1 the minimum length is eight, the maximum length is 33.



- It shows some statics of distribution and if sort by length it will show just a few with length of 8 just a few with length of 33, and the majority of them with length in the 13 to 20 range.

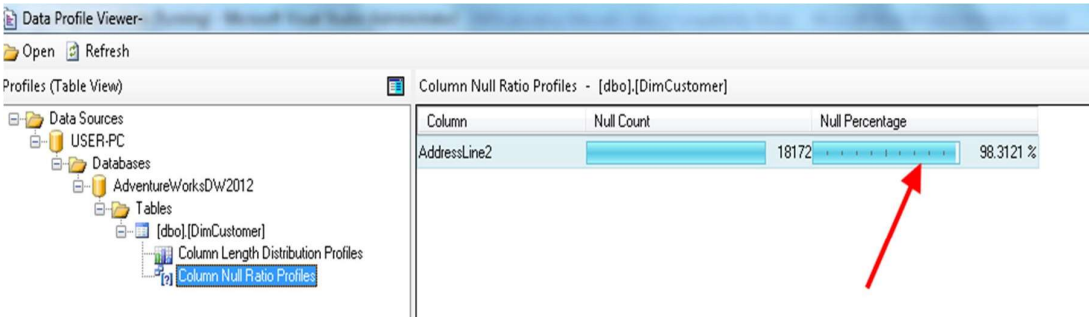
Column	Minimum Length	Maximum Length	Ignore Leading Spaces	IgnoreTrailing Spaces
AddressLine1	8	33	☐	☑

Length Distribution - AddressLine1

Length	Count	Percentage
8	5	0.0271 %
9	61	0.3300 %
10	240	1.2984 %
11	262	1.4174 %
12	648	3.5057 %
13	970	5.2478 %
14	1288	6.9682 %
15	1649	8.9212 %
16	2220	12.0104 %
17	2360	12.7678 %
18	2281	12.3404 %
19	1922	10.3982 %
20	1639	8.8671 %
21	1142	6.1783 %
22	759	4.1063 %
23	335	1.8124 %
24	200	1.0820 %
25	171	0.9251 %
26	84	0.4544 %
27	36	0.1948 %

This information is going to help to choose a data type. If this data were to be imported into data warehouse, it is important to define a data type that can handle at least 33 characters, probably not much more than that.

- Look at the column *nullratio* in the tree view to the left. It tells quite simply that for *addres line2* it is 98.3121% of the time.
- That starts to make question, do it need to import when it 98% null. So here it's a candidate for not even importing it in the first place.



Implementing control flow

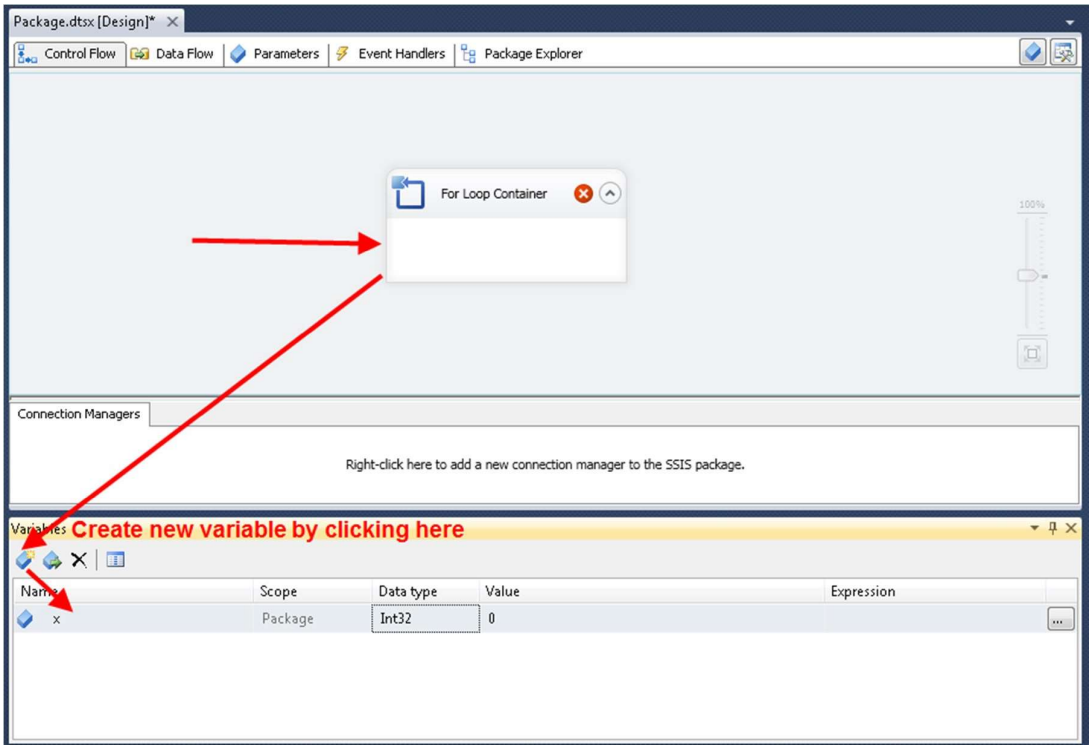
Implementing loops

This section talk about, controlling the flow of a SQL Server integration services package. When work with control the flow, we are typically worried about two things.

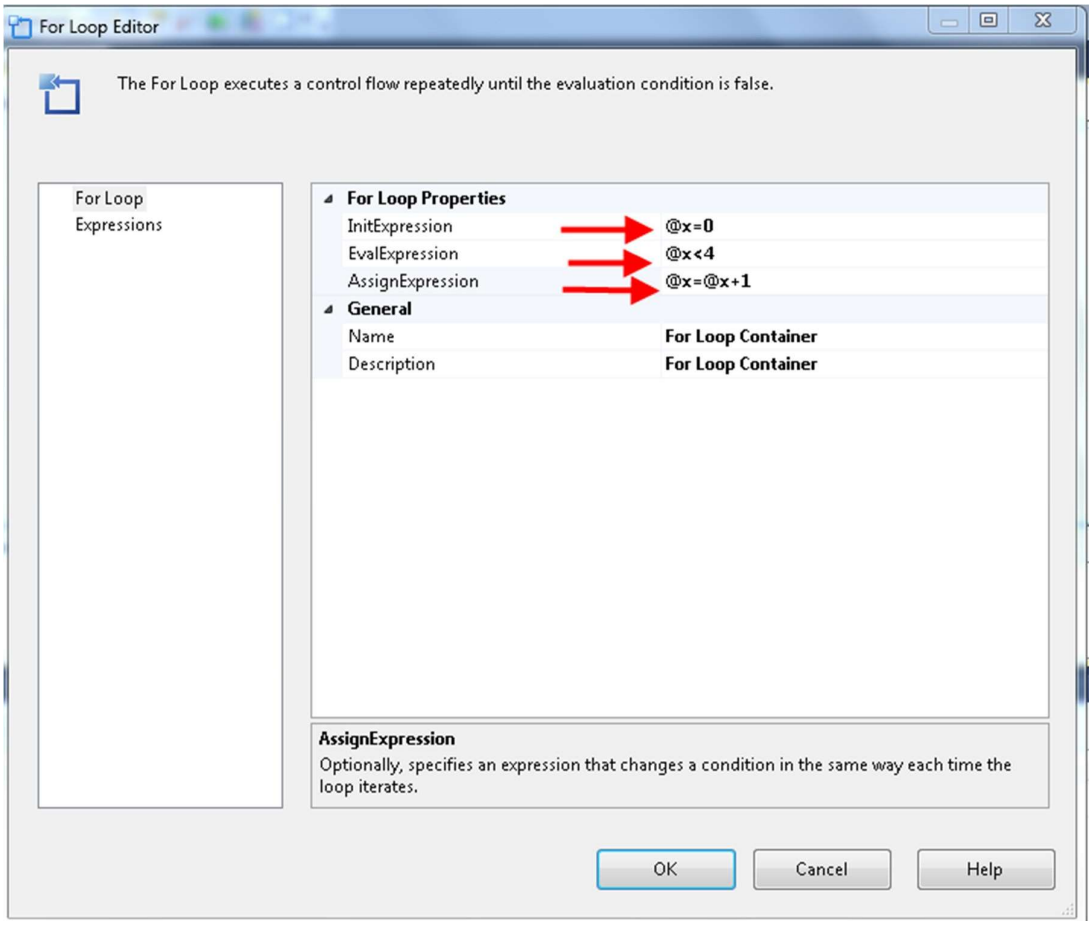
- It should have sort of looping structure to allow a certain chunk of code to be executed more than once.
- It should have some sort of decision structure, where we can conditionally execute a task or not execute it, based on the value of some variable.

Let's talk about looping, first.

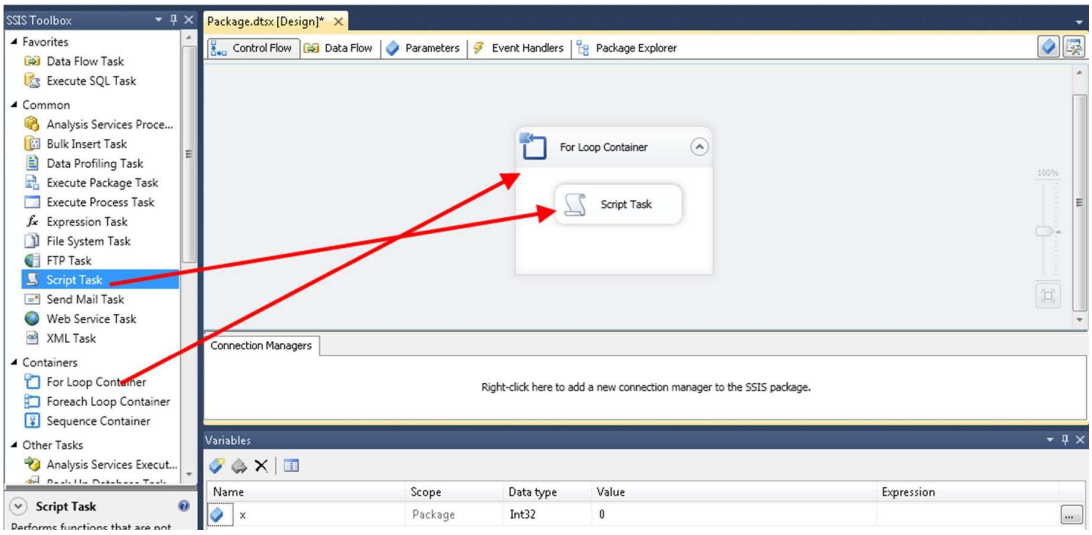
- Create a new project, and it would be an integration services project.
- Default name, default location, and on the Control Flow area, look at the ***For Loop Container*** drag one of those onto the Control Flow surface and set up some initial conditions about the For Loop.
- In doing that, we need a variable, create a variable called x. You first need to open the variable's window, if that's not already open, on the Design surface, you can rightclick and go to Variables.
- That will open the Variables window. In the Toolbar, all the way to the left is ***Add Variable***, name it ***x*** and the default data type of integer.



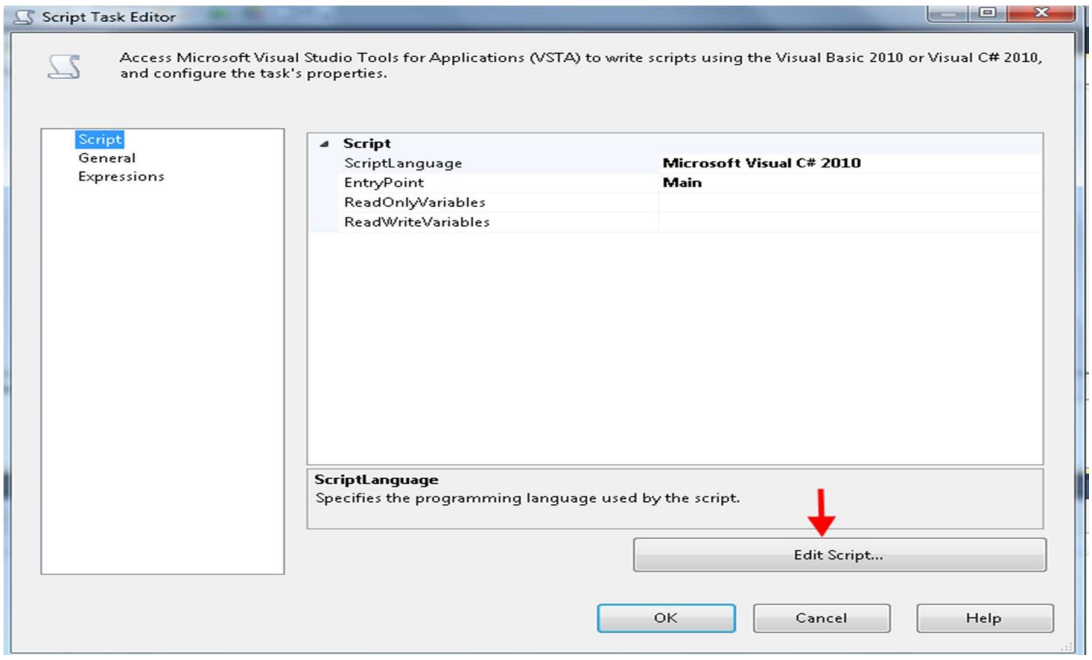
- Now, double click on the For Loop Container. Set initial expression to be x =0 in SQL Server Integration Service need to use the @ sign to symbolize that what's coming is a variable. So, it's at @x= 0,
- Then we need to set an evaluation expression and other condition as below



- ***Drag Script Task over*** the loop container.

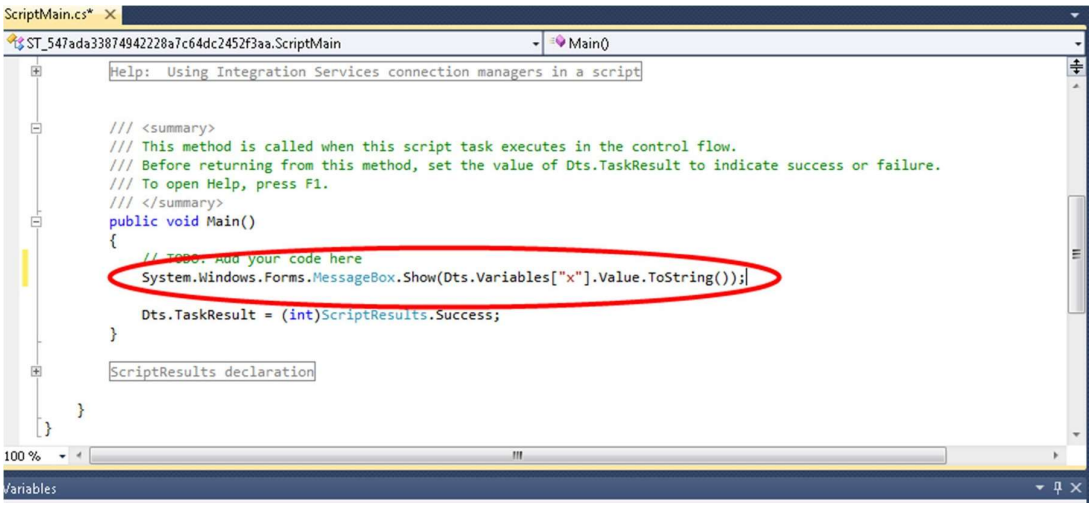


- Double click on Script Task lower right have a button for *Edit Script*, and that will pull up a new instance of Visual Studio with some C# code already written, including a function called Main.



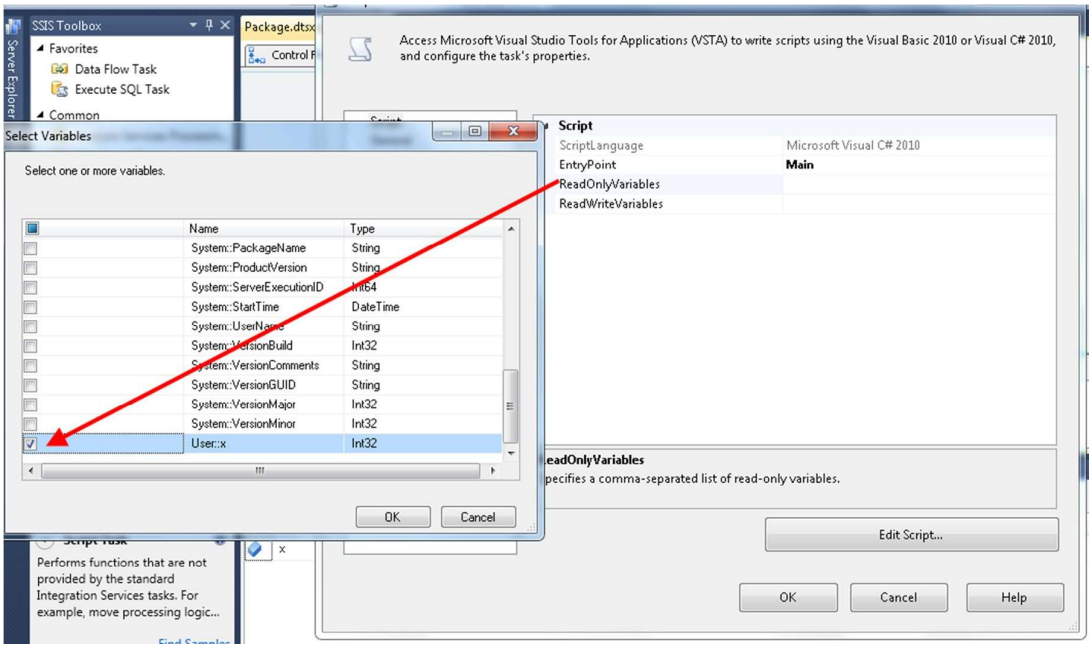
- Inside of Main, add the code that I would like to run just show a message box with the variable, x, so that we can see that x is, in fact, incrementing.
- The syntax we are referencing in SSIS variable inside of my C# code is as follows:

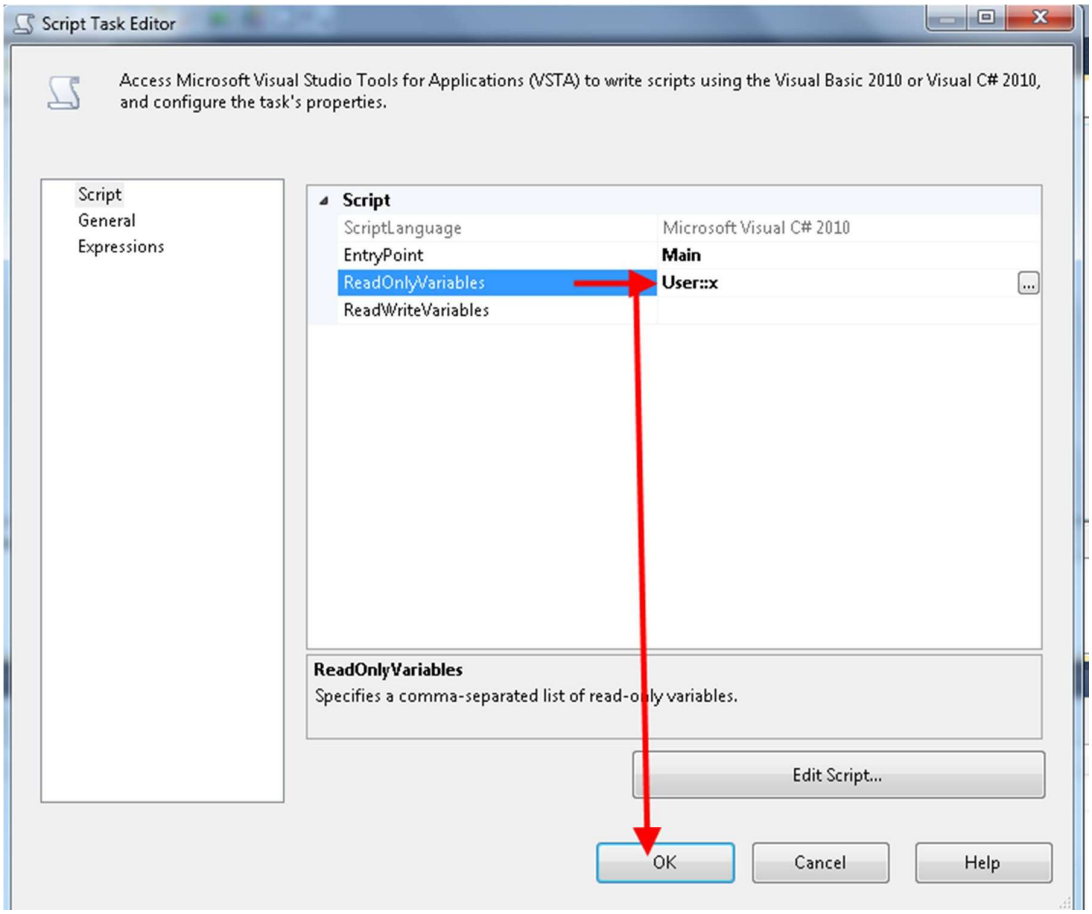
```
System.Windows.Forms.MessageBox.Show(Dts.Variables["x"].Value.ToString());
```



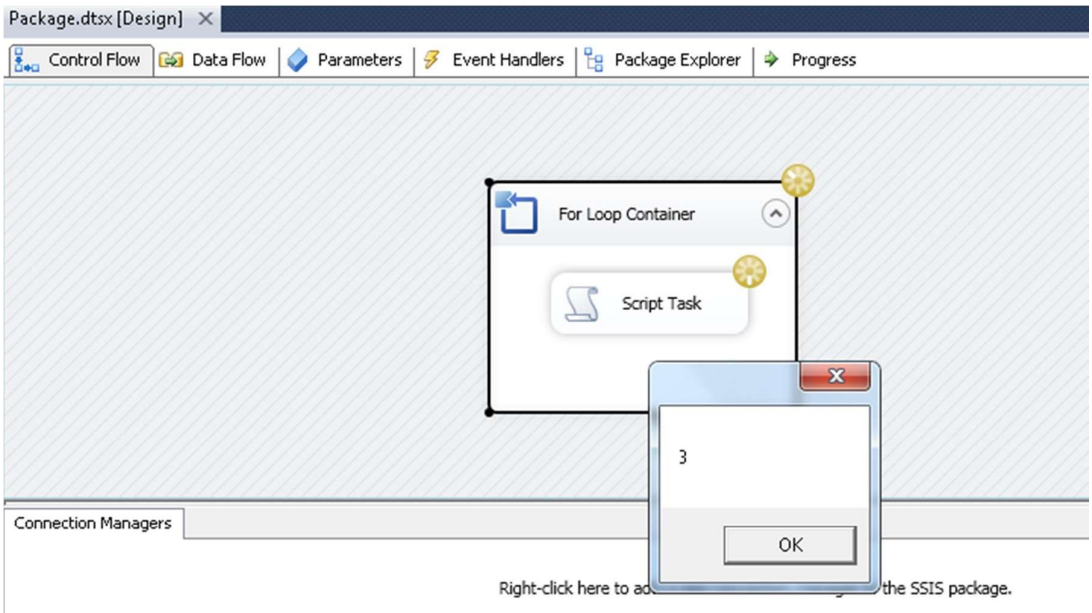
Dts is a legacy name, the product Microsoft had before SQL Server Integration Services was called Data Transformation Services or dts.

- Save and close this window. Go Back to *Script Task Editor* dialog double click and set *Readonlyvariable* to x.





Click OK, and you should be able to run this, and see the loop come up multiple times.

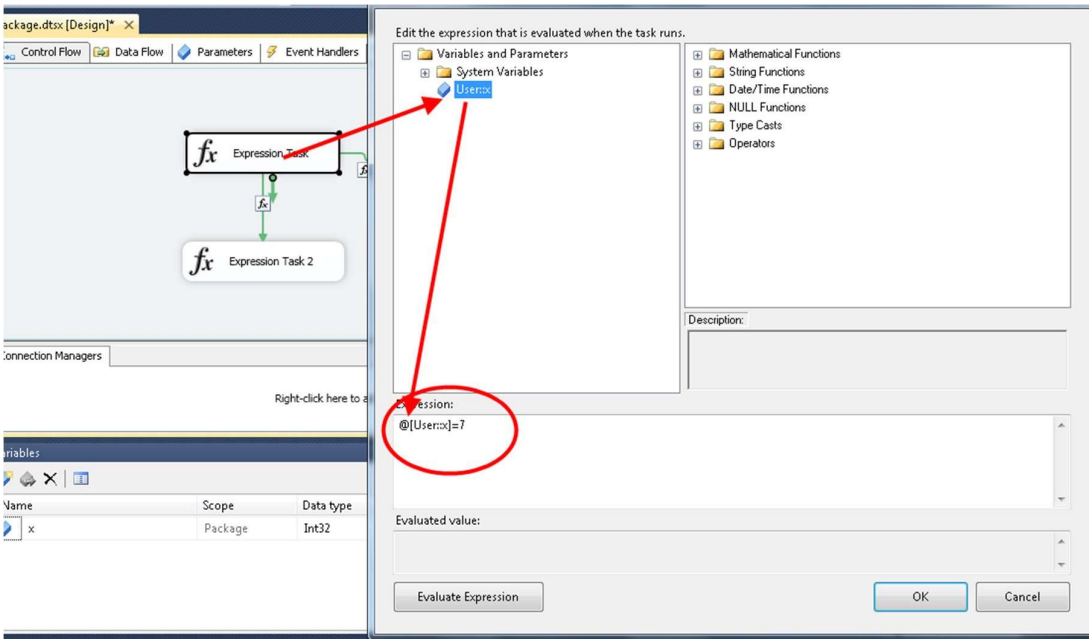


Once the value of x got 4 loop will stop.

Practice for reach loop by yourself.

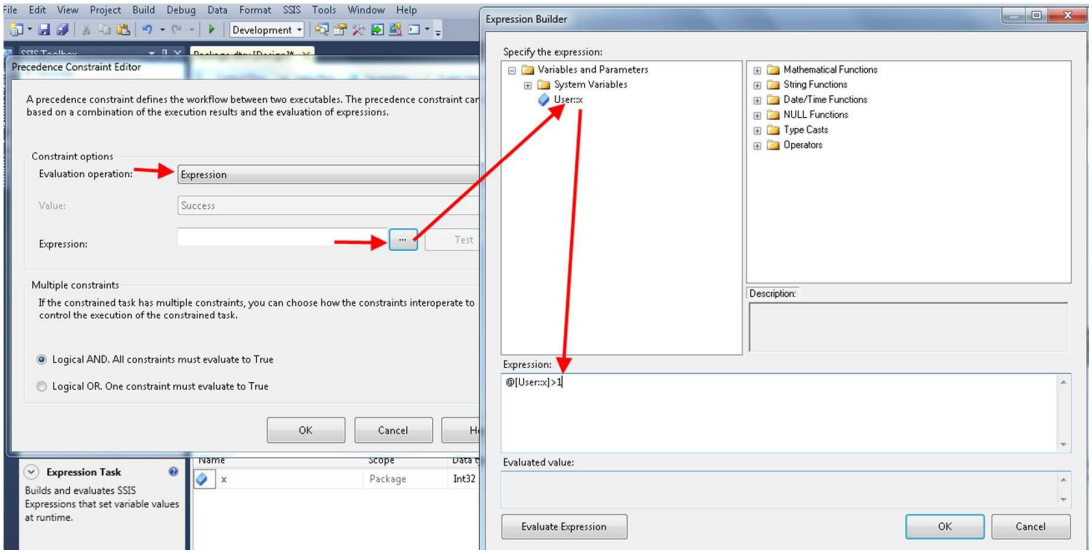
Implementing IF-Else Logic

Delete all item from design area or create a new Project go ahead with ***Expression Task***. Double click on that **Expression Task**, and the expression, assign a value to the variable x=7



Create two more Expressions Tasks to have the destinations where we can possibly go out of two direction based on condition.

Connect with precedent constraint and right click to edit. Change Evaluation operation that to an expression, and that expression will be in the variable of x >1.

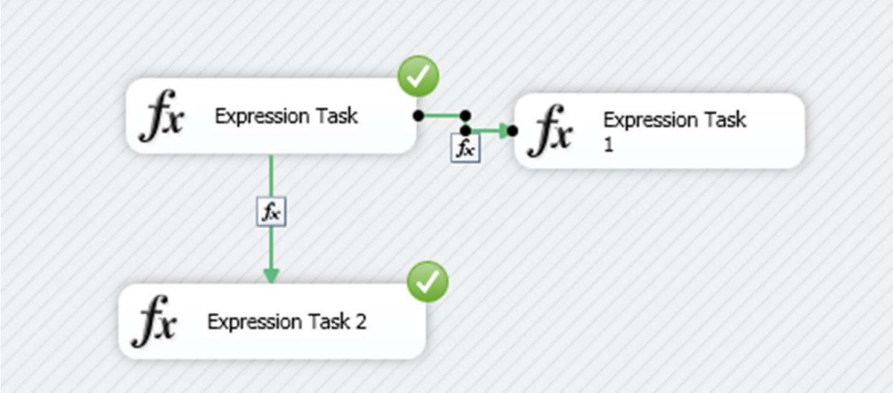


So that means now this precedent constraint whenever the value of x >1, this precedent constraint will evaluate to true. If the value of x >! 1 this precedent constraint will evaluate to false.

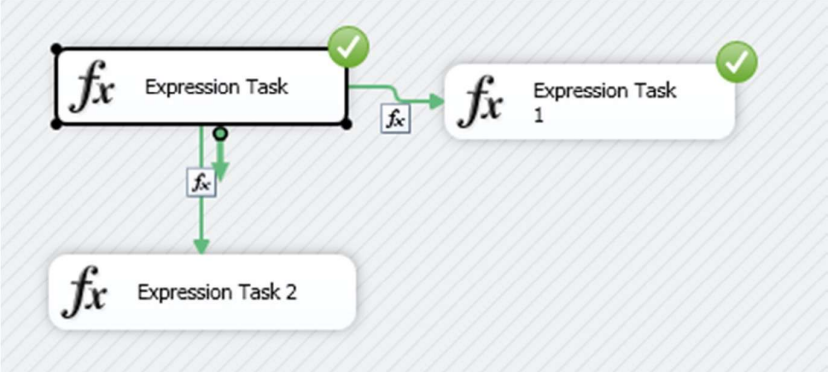
Now drag an additional constraint to the other expression task. Right-click on that one also and Edit, again need to change this to an expression. This expression will be the opposite of what we did before. Rather than x > 1, do x < 1.

Now some branching logic is implemented, if the value of x > 1, follow one path. If the value of x < 1, follow a different path.

Test this out by running since in first Expression Task, value of x to be 7 so we should follow the path of > 1. Then we see that run successfully.



Now change the Expression Task to set the value of x= 0 , where it was initially set 7 and now we should follow the path of x < 1, and see it will a different path.



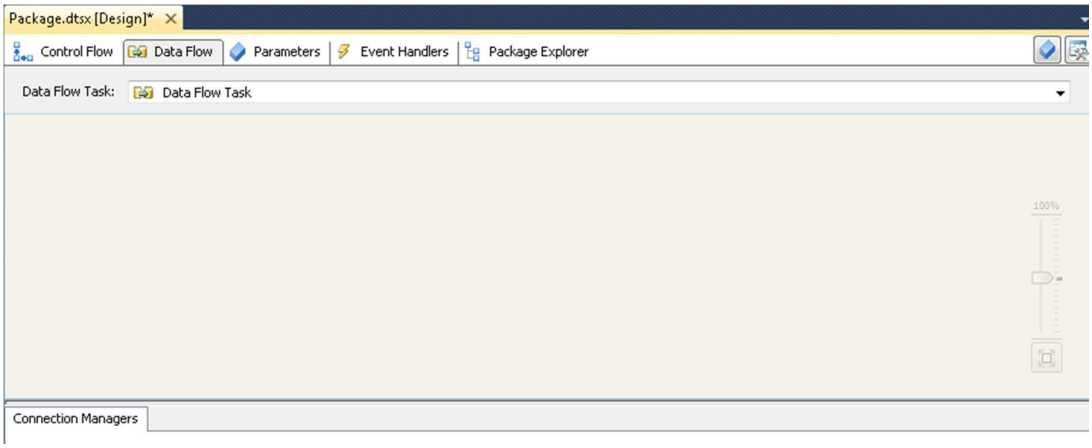
So this is a way to implement IF THEN logic in SQL Server Integration Services, it's maybe not 100% intuitive to a new user; but once you get used to it, it is a quick and easy way to implement some branching logic.

Implementing data flow

Open SQL Server Data Tool

Create a new project and name it (ResellerPackage, we are going to use it while deploying SSIS)

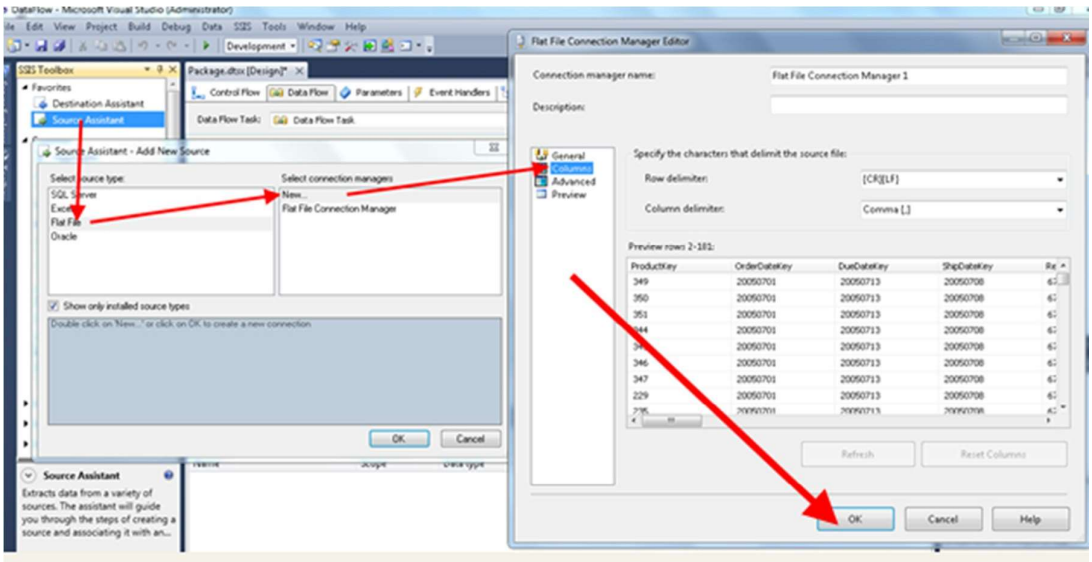
Under Control flow tab from toolbox drag ***Data Flow*** Task onto the control surface, and then double-click on that Data Flow Task and now notice environment has changed a little bit.



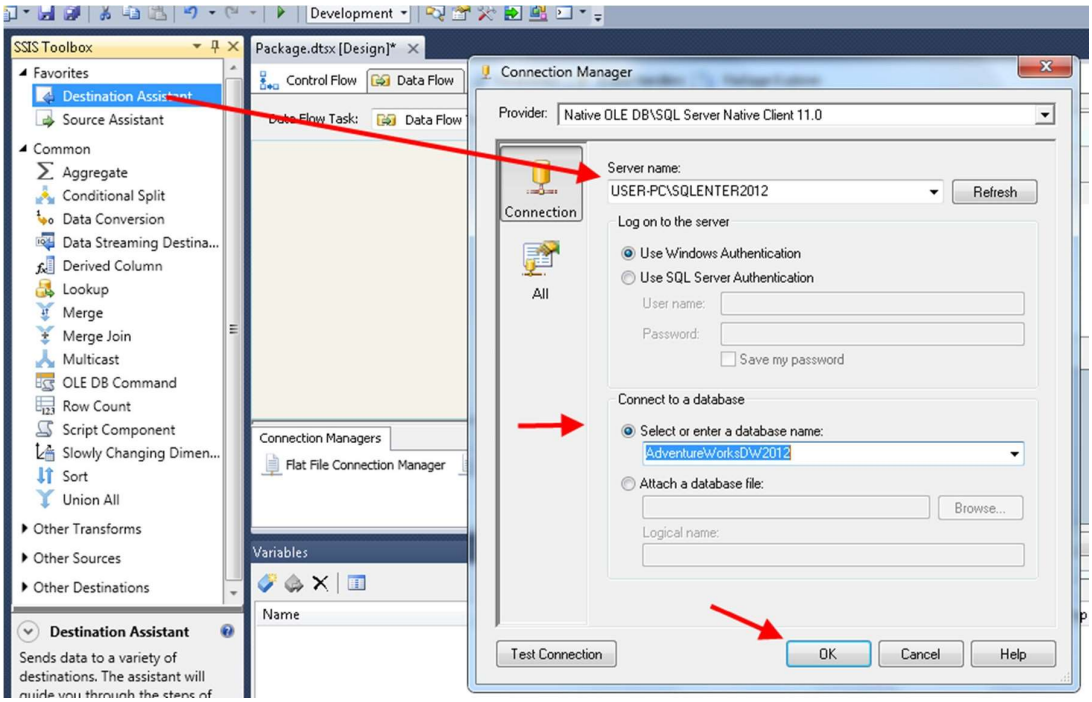
Every Data Flow Task need a source and a destination. In other words, it's a place where pulling data out of and a place where inserting data into.

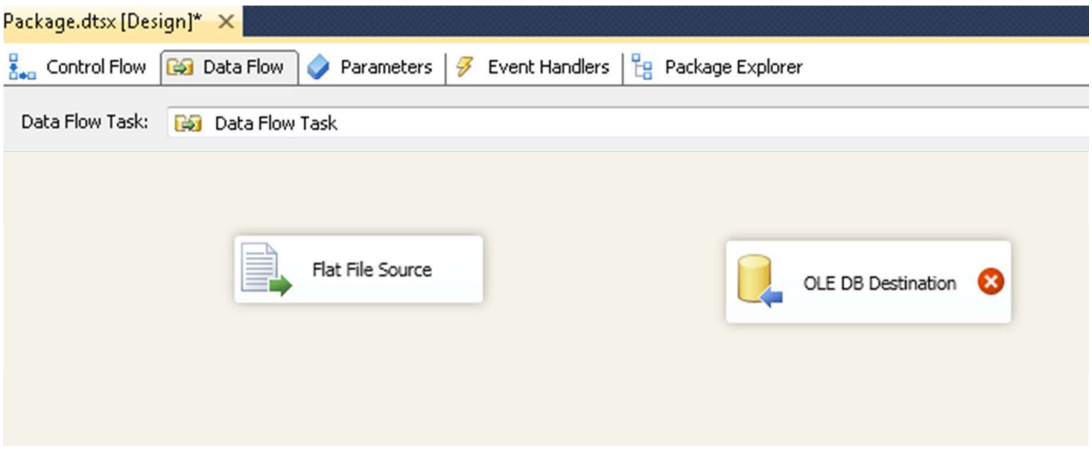
So first, drag a ***Source Assistant*** on to work surface.

This particular source will be a *Flat File* and New connection browse for the file in the *ResellerSales* (available with manual), Looking at the columns, notice it's mostly keys that links it to dimension tables. But then there's also some information about the pricing of the products. So that looks good for a source.

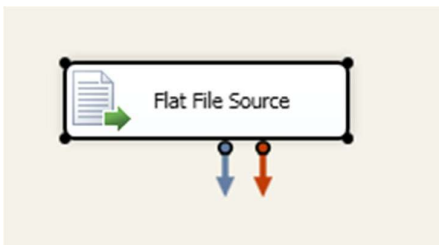


Now drag next task *Destination Assistant* on to the screen. And select SQL Server, a New connection. Use local SQL Server and the database AdventureWorksDW2012.





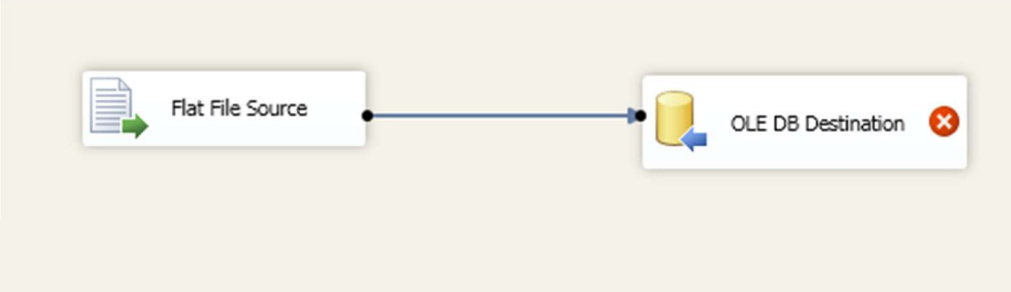
When click on the *Flat File Source*, you notice two arrows coming out of the bottom of it.



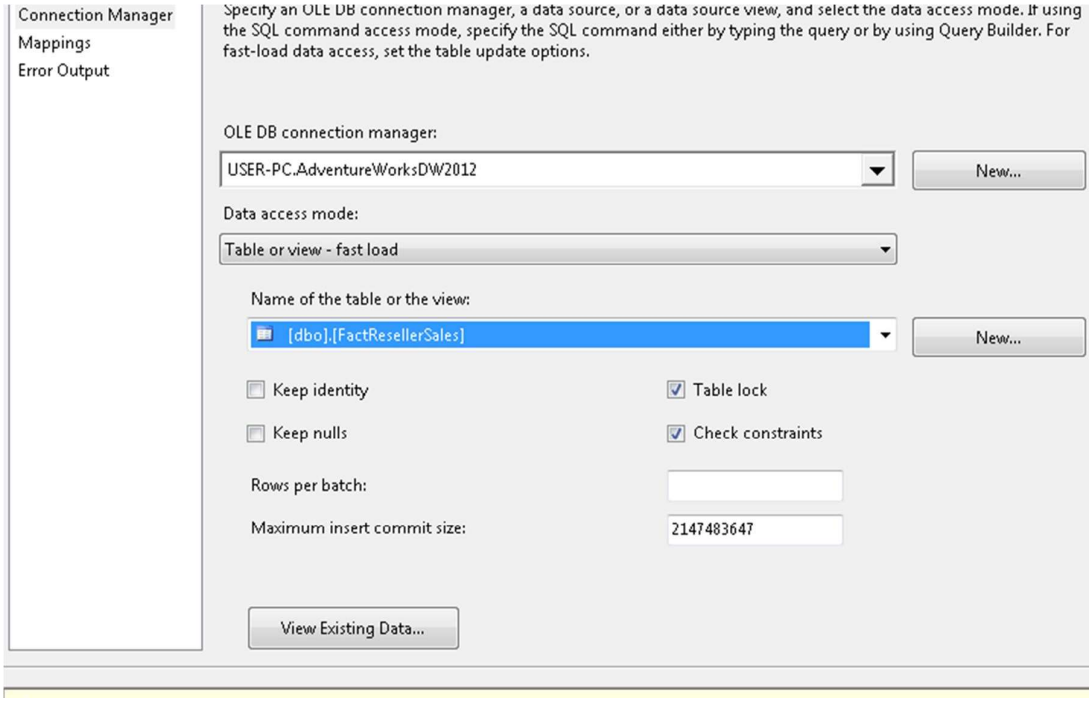
These arrows are different than the arrows in the Control Flow area. In Control Flow, an arrow is a *precedence constraint*. It controls what task runs after the current task based on success or failure. But here in the Data Flow area, the arrows represent data that's flowing from one task to another.

The blue arrow represents good data, data that was successfully read or successfully transformed. The red arrow represents bad data, data in which there has been an error such as, unable to read the entire row, or unable to perform a specific transformation.

Drag blue arrow that over to destination, so now flow of data from a Flat File Source into an OLE DB destination.



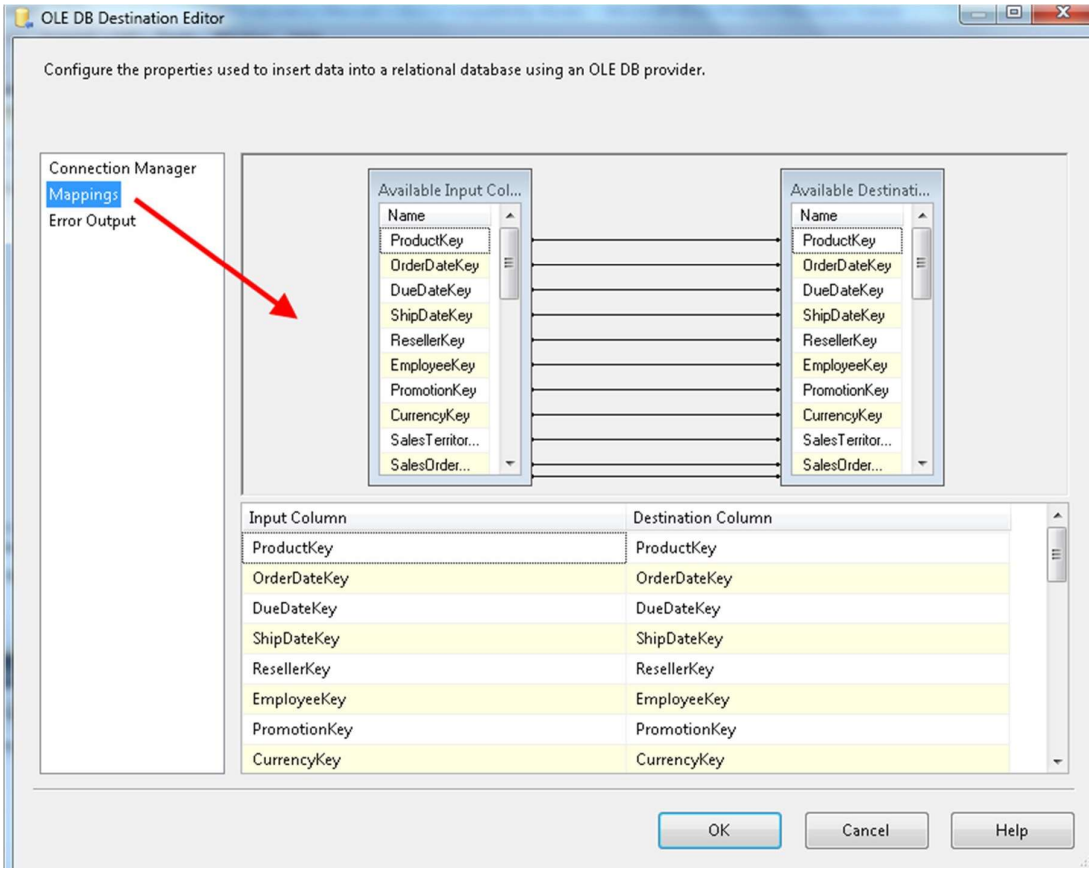
Double-click on the destination to provide a little bit more information, i.e. which *table* data to flow into. In this case that is *FactResellerSales*.



Click on ***mapping*** and observe that machine has automatically made some assumptions about mappings. And that's because the names of the fields in the

input are very similar to the names of the fields in the destination. So the machine assumes any field with the same name should be mapped to the field with the same name in the other.

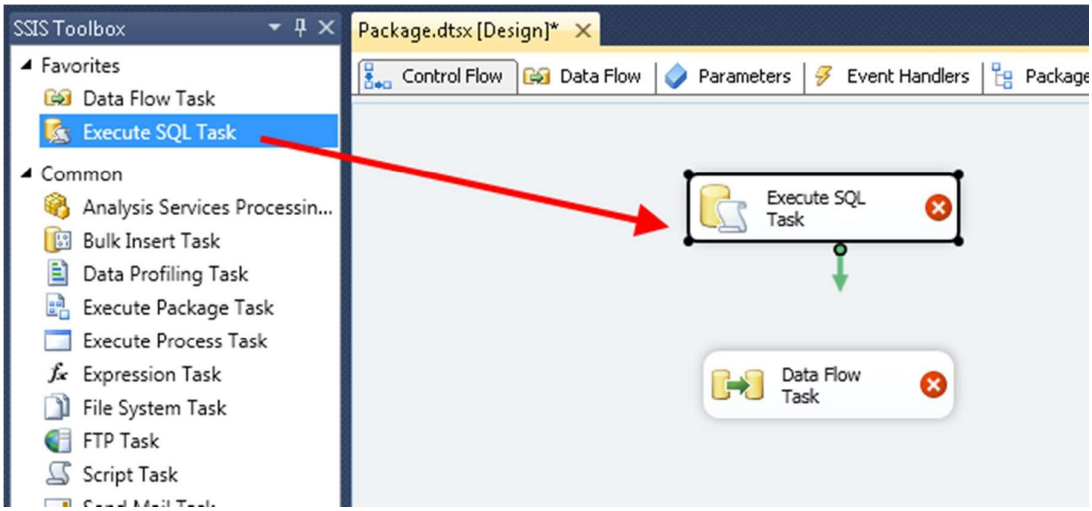
So dont need to make any changes just click OK.



With some ETL processes you can insert the new rows data that has changed. Other times, you're going to delete everything that was in the table, and completely reload the table from scratch.

In this case delete all the data that's currently in the table, and reload everything from the flat file. For tahtthere is need to add a command to delete the data before we run this data flow.

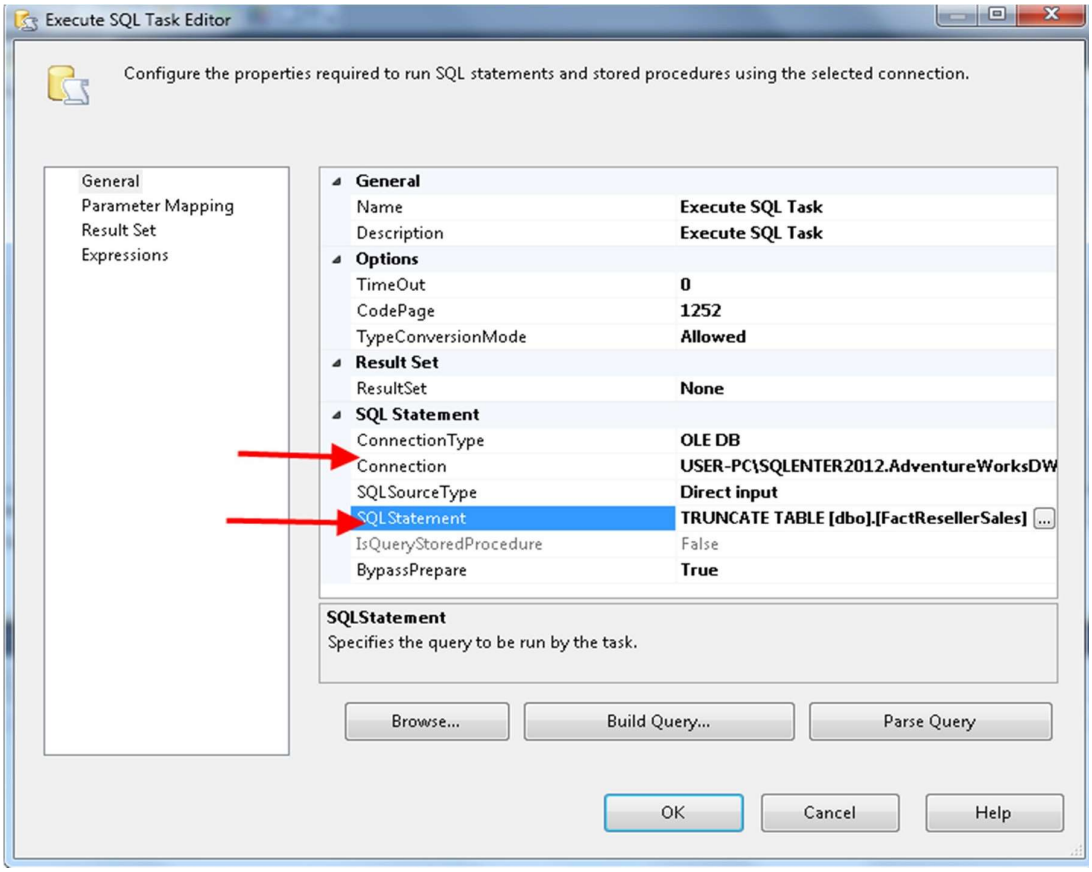
Go back to the Control Flow tab and add an ***Execute SQL Task***.



Double-click on it, to use an set ***connection*** to local server the one to AdventureWorksDW2012.

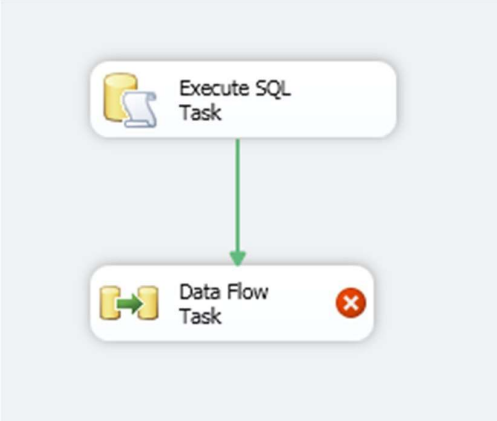
In ***SQL Statement*** Copy and paste following *SQL Statement*:

TRUNCATE TABLE [dbo].[FactResellerSales]

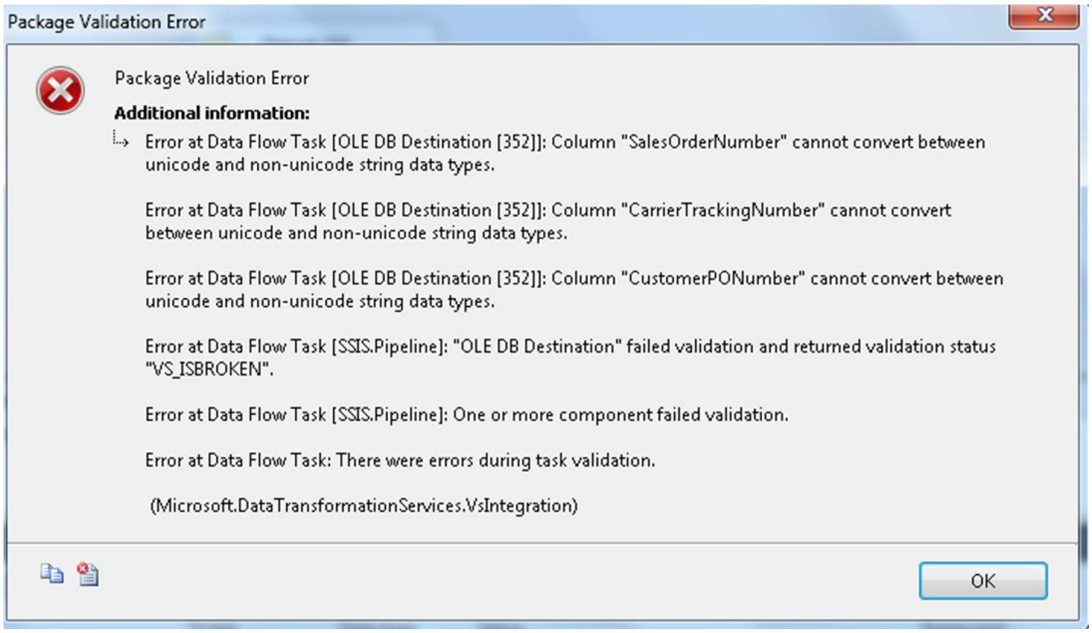


It's a simple code just says TRUNCATE TABLE dbo.FactResellerSales. So that will completely remove all the data from that particular table.

Connect green arrow for a success precedence constraint to the Data Flow Task.



Run this now and see an error, but it is a common error. We see that we're getting errors about cannot convert between unicode and non-unicode string data types.



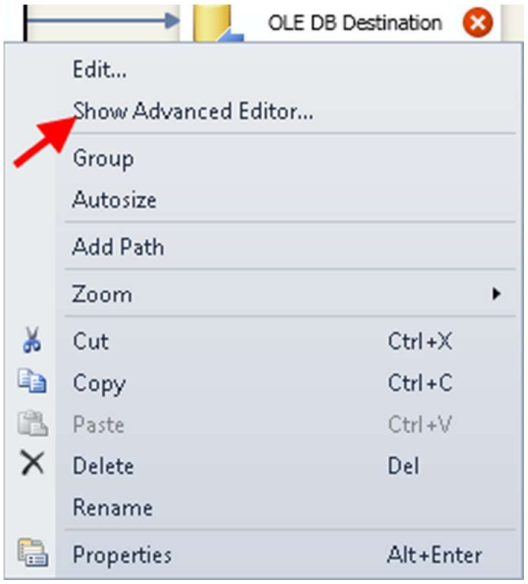
Remember Microsoft SQL Server recognizes two different types of strings. One is called a unicode string, which handles all of the letters of the English alphabet, plus several special characters, letters with accent marks and different characters used in international languages.

Then we also have non-unicode strings, which pretty much focuses just on the American alphabet.

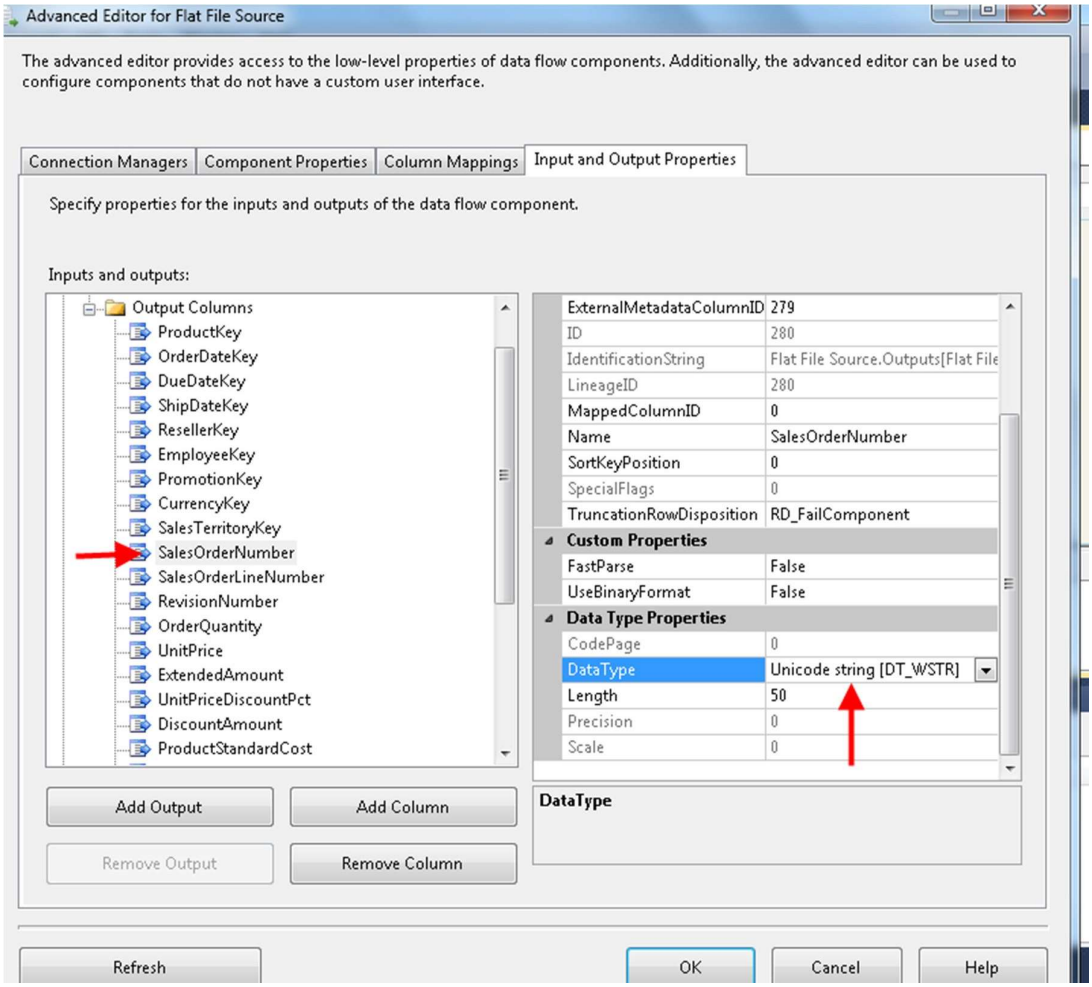
Resolve this issue by changing both source and destination to either both use unicode or both use non-unicode.

Go to Data Flow area, or double click on Data Flow task to go on ***Flat File data source***

Right click and go to the ***advanced editor.***



In there use tab *Input and Output Properties*, expand the Flat File *Source Output tree*, expand *Output Columns* and look at ***SalesOrderNumber*** and see that it is currently set to a string data type. Change that to a ***Unicode*** string.



And then do exactly the same thing for *CarrierTrackingNumber* and customerPONumber should become *a Unicode string*.

Click OK, and I again run this package. As the package is running, you can observe number of rows incrementing going up all the way to 60,000. We got a green check mark on the Flat File Source and OLE DB Destination.

Flip to the *Control Flow* tab, notice green check mark on the ***Execute SQL Task*** and ***Data Flow Task.***

So all of our data movement has been successful. Looking at a little more detail on the Data Flow Task. We just took data from one Flat File and moved it to a destination. We didn't do any transformations. The machine does have several pre-built transformations. We can Aggregate data, Split data, Convert data, etc. I'll drag one of these onto the screen, the **Data Conversion**. So now remove the existing data flow arrow, add a new one that now goes from the Flat File Source to the **Data Conversion.**

Then data will flow from the Data Conversion to the OLE DB Destination. In order to make this a little more clear, I'll rearrange some of these task. We can see data flowing downward through our application. I'll double click on the Data Conversion and we can see that for each input column. I have the option to change the Data Type, change the Length, the Precision or the Scale. Data Conversion tends to be the most common type of transformation. We often change things from a string to a date time field or from a string to an integer field.

SSIS makes this very easy for us. I don't have to write a whole lot of code. We can do most of it through this visual interface of data conversions, going from a data source to a data destination.

Deploying and Configuring SSIS Package

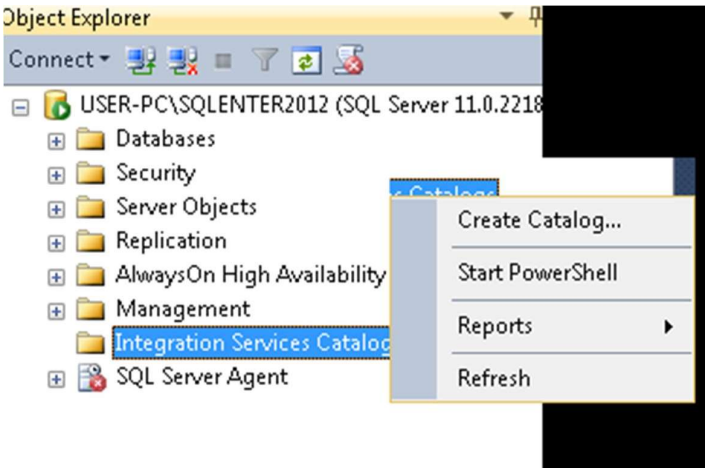
Creating SSIS Catalog

In SSIS 2012 we deploy a project, a project can contain multiple packages along with different support files and parameters. Those projects will be deployed to an Integration Service Catalog.

Open SQL Server Management Studio.

Go to Object Explorer -> Integration Services Catalogues.

At first it will be empty, Rght click -> Create Catalog.

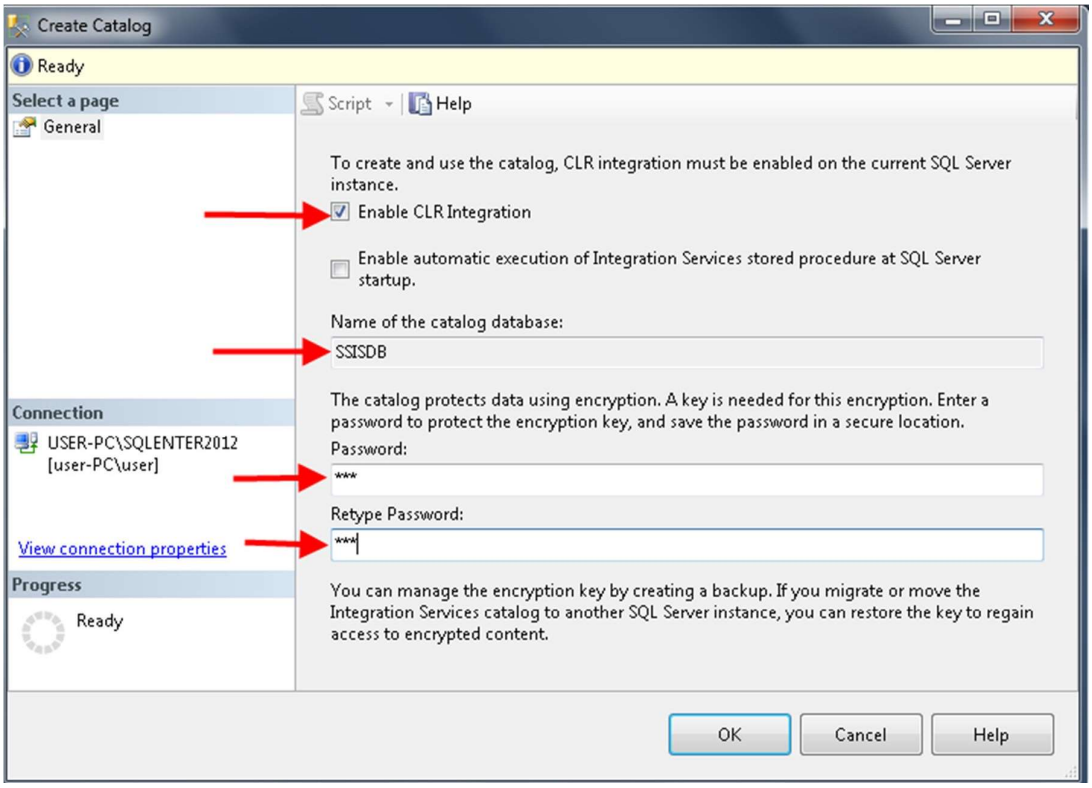


Create Catalog wizard will appear.

At the top, you'll see a check box for enable CLR integration. The catalog requires CLR integration. If that check box isn't checked, you're going to check it.

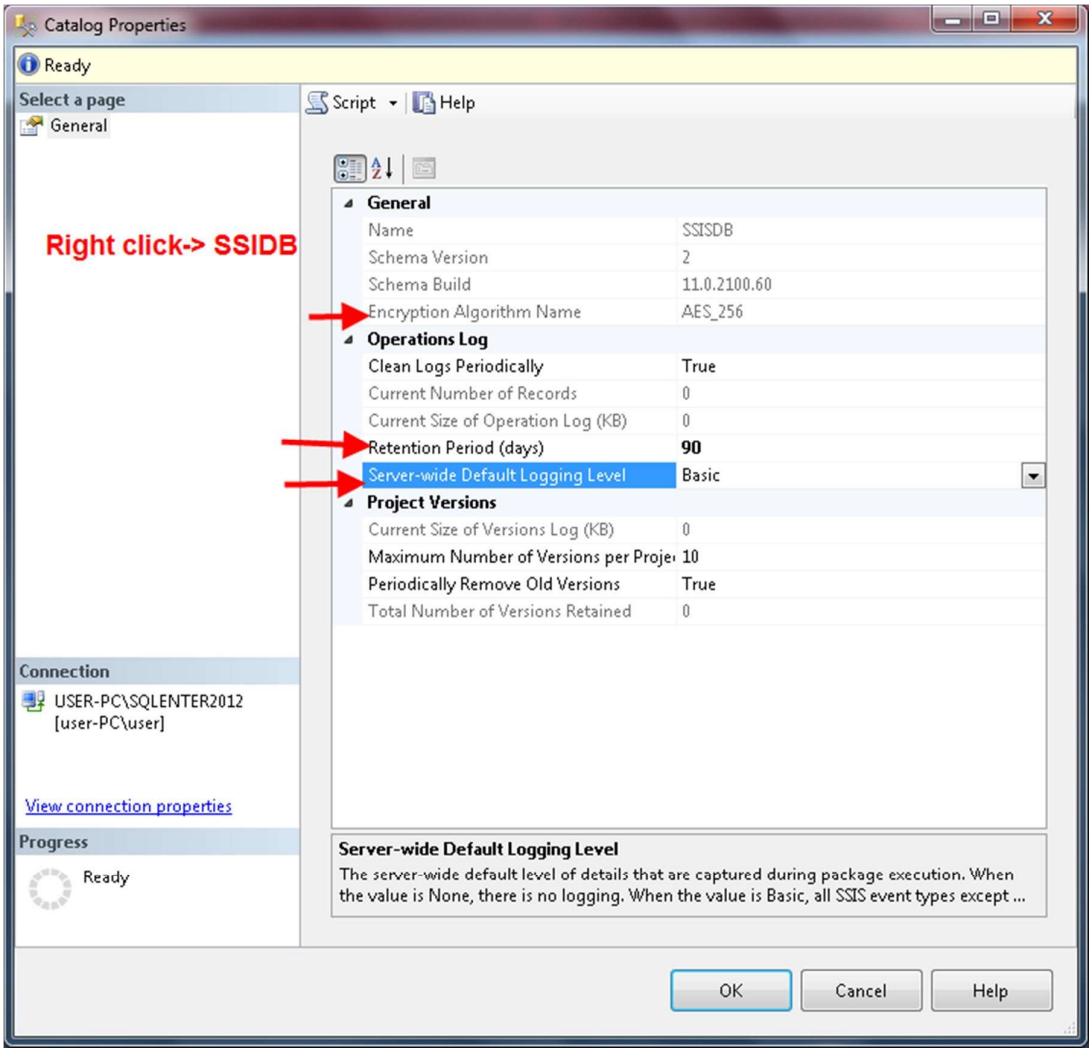
It' will ask for a name of the catalog database. It suggested the name SSISDB

Supply a password that will be used as a key for the encryption.



Click *OK* and now the Object Explore you can see that SSISDB is created and it is currently empty.

I right click on SSISDB, it check Properties.



Look for encryption algorithm name I am using *AES 256*. This can be changed but the database needs to be in single user mode to change it.

Option to clean up the logs periodically is ***Operation Log***. Typically in SSIS, we log a lot of things and those logs may be interesting to us for a short period of time. But we may not want to keep them forever therefore the default is to set to ***True***, meaning logs will be deleted on a periodic basis.

The Default ***retention period*** is 365 days, which is an entire year. If an ETL solution is running every night and a problem occurs, you might know about the problem in a lot less than 365 days. So logs that old to me are not interesting. If a problem occurred that long ago, you should solved it by now. Change this to 90.

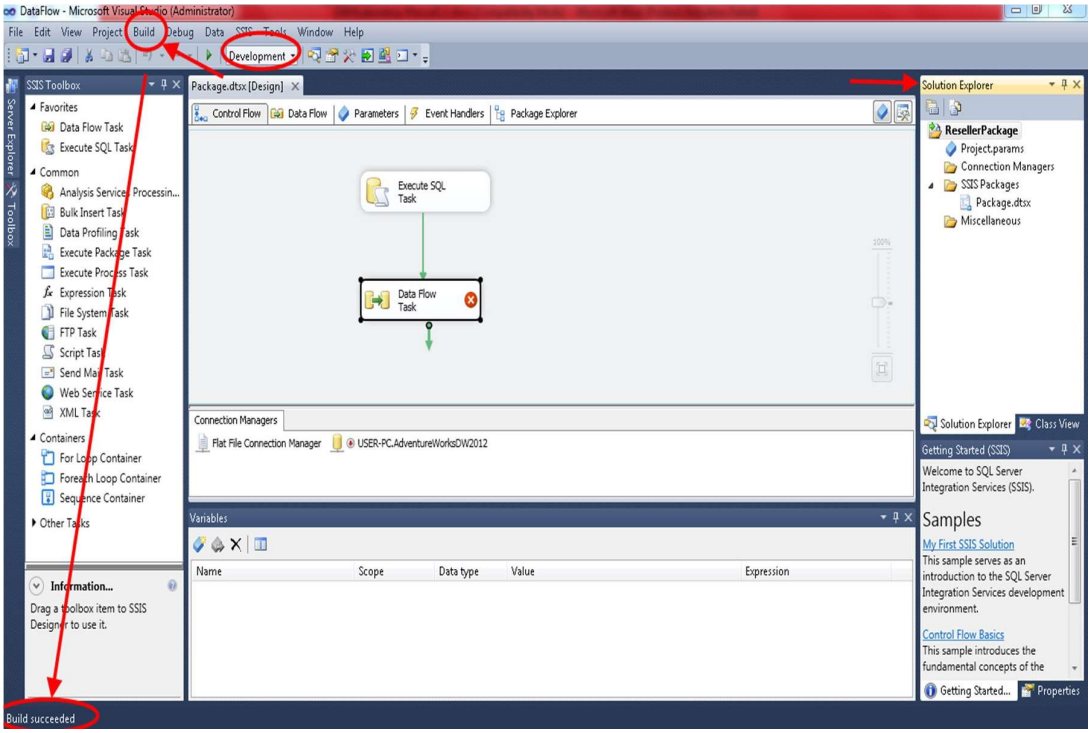
The other option is ***server-wide default logging*** level. Each package can specify its own logging rules, but by default a package doesn't specify any logging

information and will take on the characteristic of the default server behavior. Here the default server is set to basic.

You will observe total of four options, *Basic, None, Performance, and Verbose*.
The default logging level of verbose will log everything. Literally every event, the start and stop of it. Everything the machine can possibly log about an SSIS package will be logged. This is typically not appropriate for your production environment. You might want to do this in testing and development but in production, your logs would get way too big.
A slightly less aggressive option and the default option is *Basic* which still logs a large amount of things. It logs everything except the on progress event and the on custom event.
The option ***Performance is*** only logs the OnError event.
And the last option is None, which is self-explanatory. It logs absolutely nothing.

Deploying SSIS solution

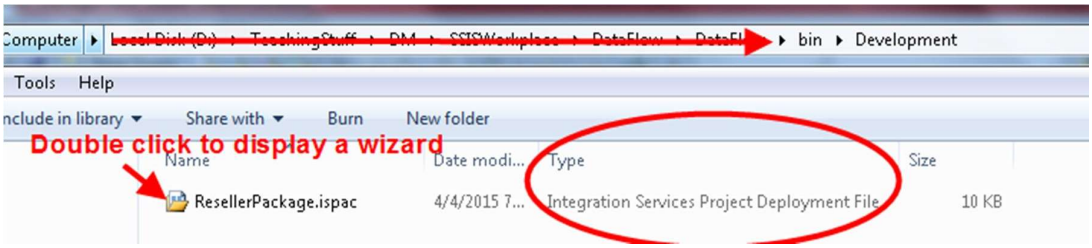
Now we have our Dataflow package (created during implementation of Data flow with name ResellerPackage) and SSIDB as we learned in previous two topics.
Next task is how to deploy SSIS package on SSIDB.
Open previous Data Flow package (ResellerPackage).
Run Package in Development Mode, Which is available right after green arrow run button. You can configure solution according to your users requirement by selecting “Configuration manager” “right below the Development”
From **“Build”** build “ResellerPackage”. You may notice in very lower left corner phrase
“ Build Successful”



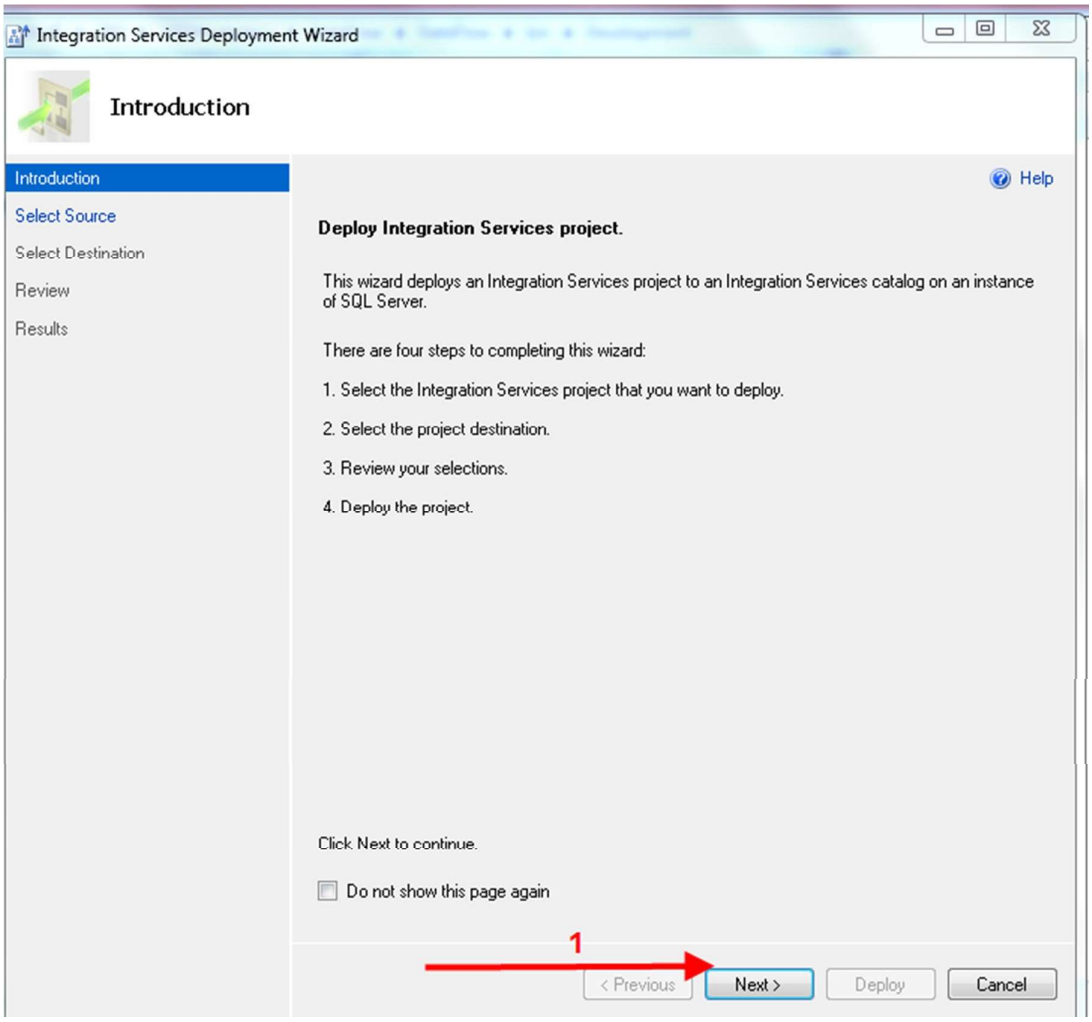
Now you can locate the folder where your project are stored and look for where Build is stored we just ran in Development mode

Open up “Resellerpackage-> Resellerpackage ->Bin->Development

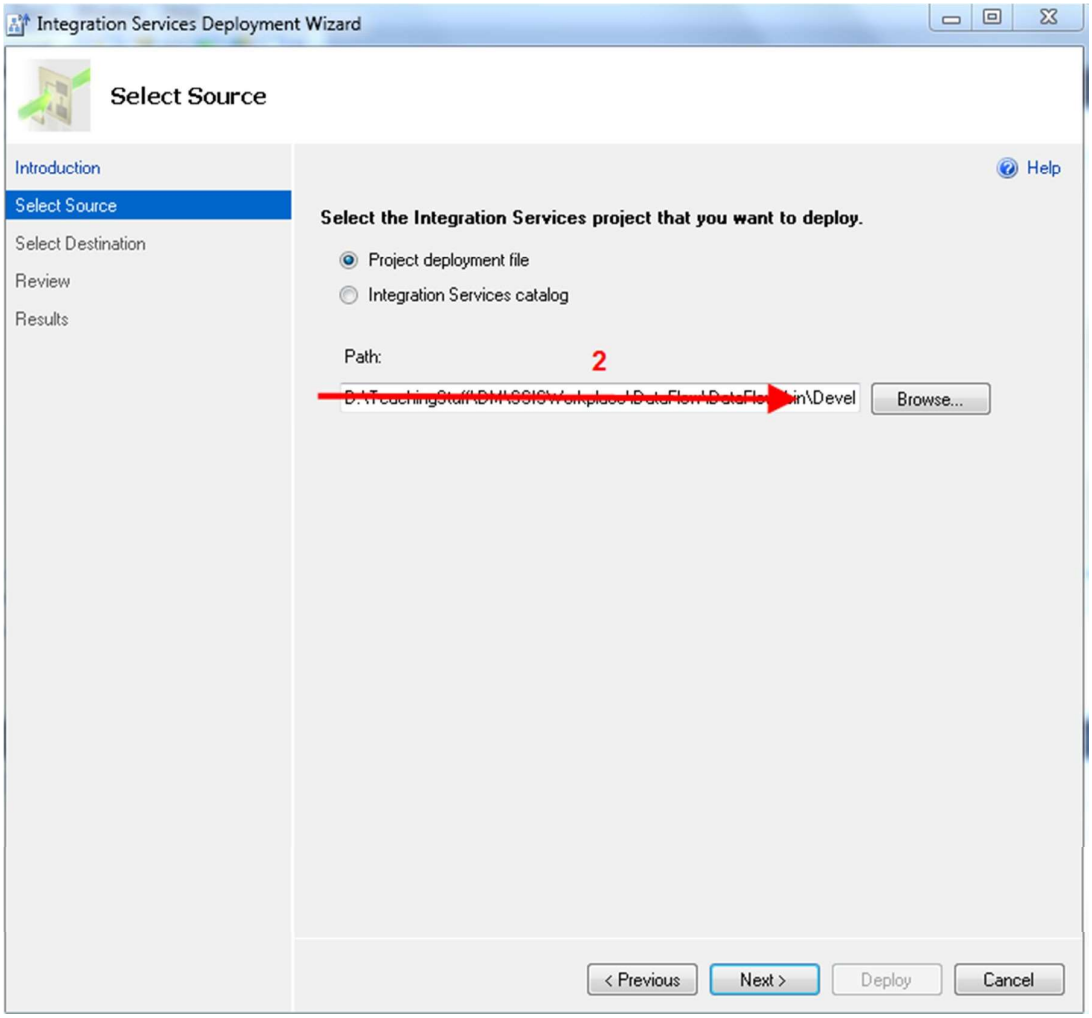
You will see only one file as following: Double click ResellerPackage to run a wizard to deploy this SSIS project.



Follow following steps:

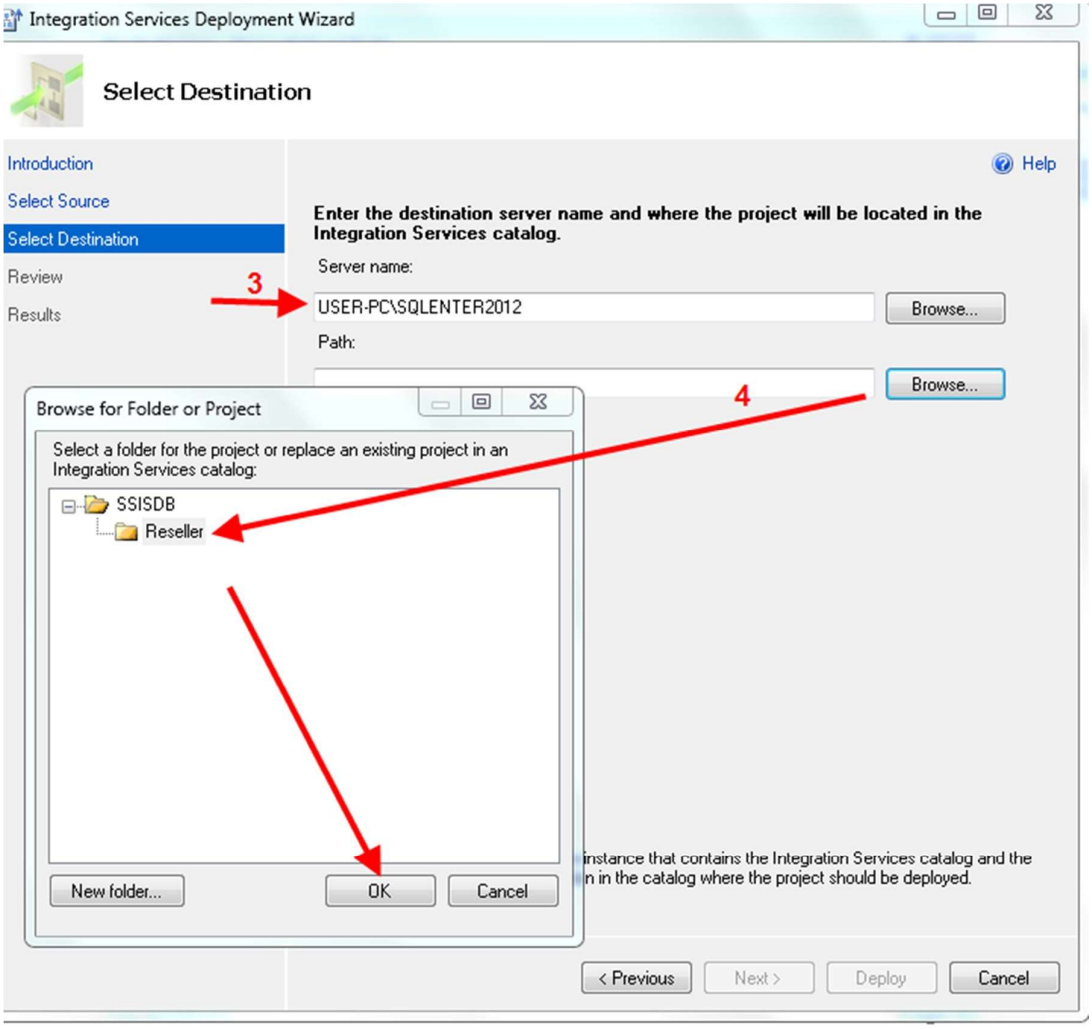


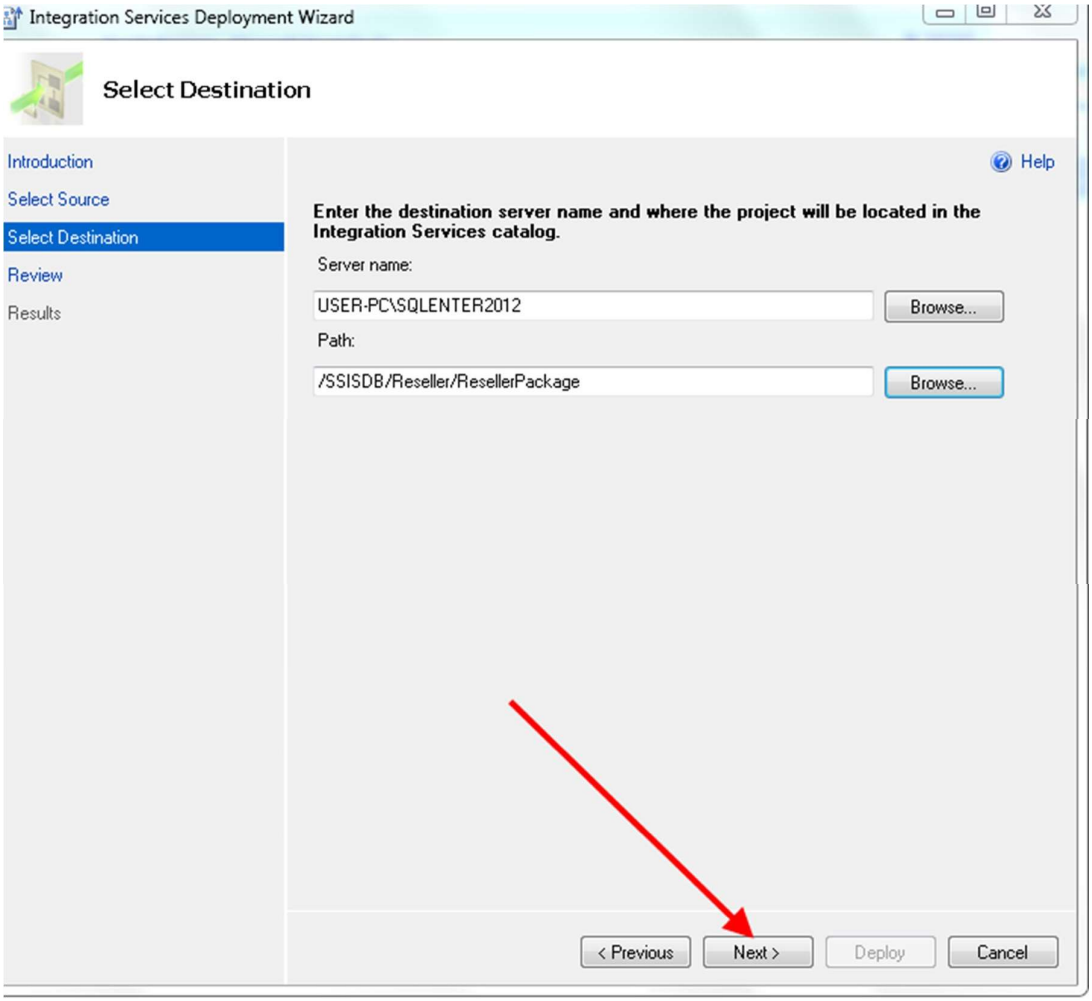
Provide source file path, in this case "ResellerProject" file already selected since we are going to deploy this file

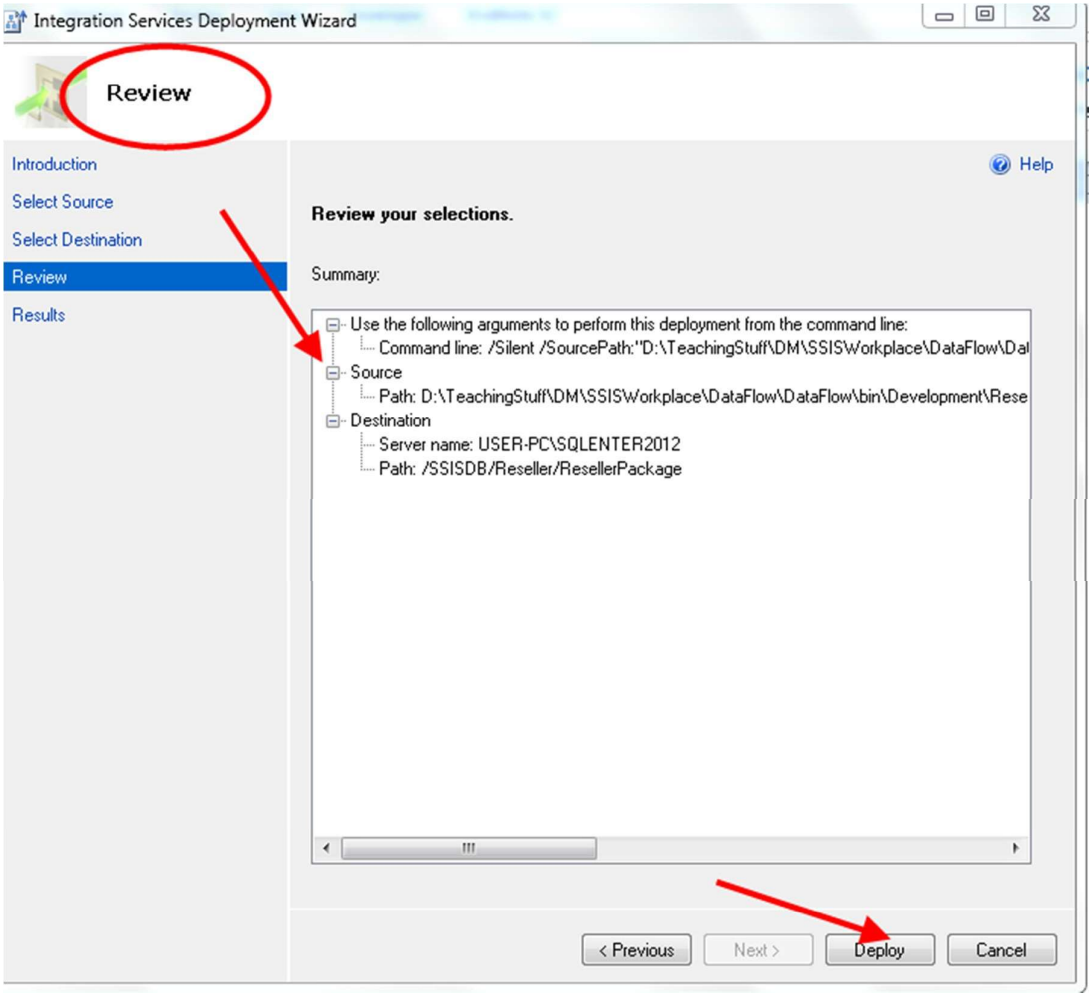


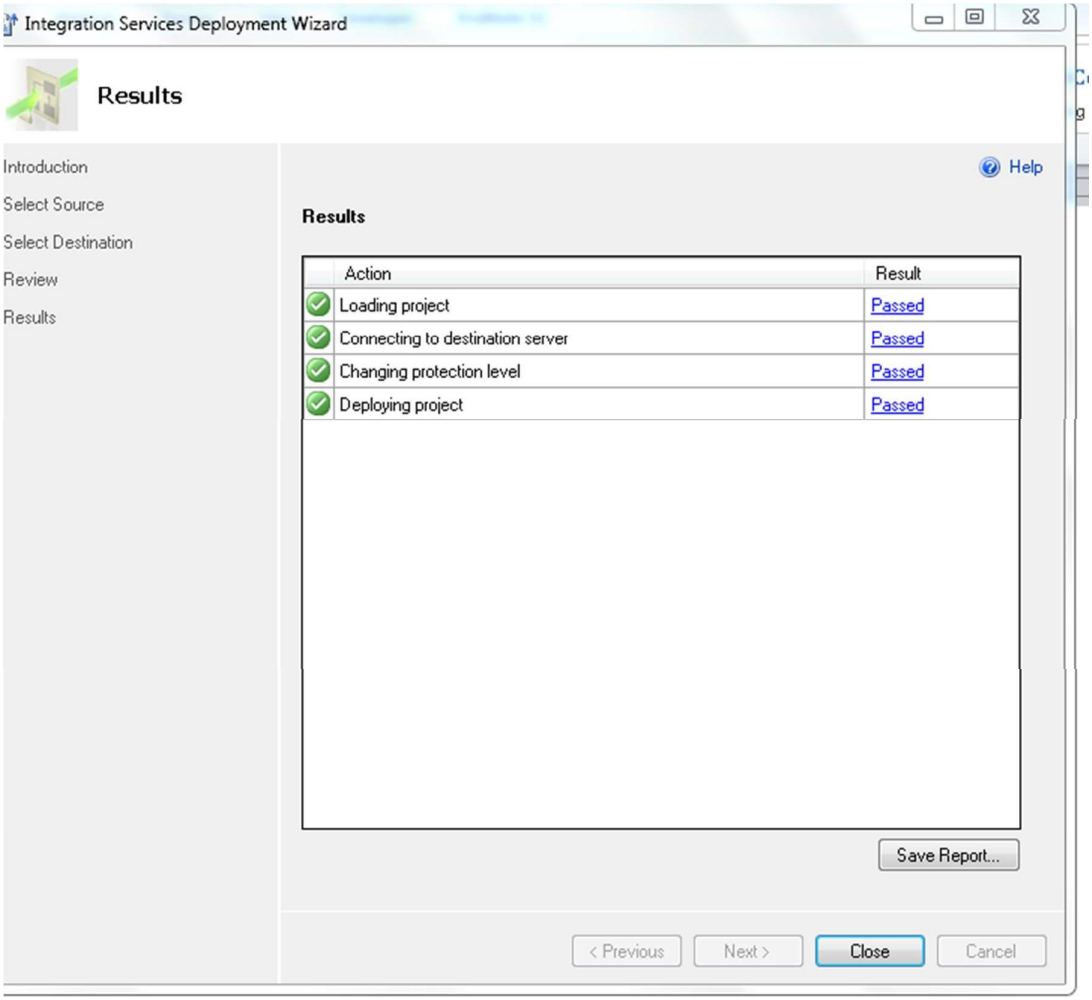
Provide destination which is your local server

In path Select SSISDB which created in previous exercise in Integration Service Catalog. Create a new subfolder by clicking on "New Folder" where this project need to be placed


Integration Services Deployment Wizard
Select Destination
Introduction
Select Source
Select Destination
Review
Results
Help
Enter the destination server name and where the project will be located in the Integration Services catalog.
Server name:
3
USER-PC\SQLENTER2012
Browse...
Path:
Browse...
4
Browse for Folder or Project
Select a folder for the project or replace an existing project in an Integration Services catalog:
SSISDB
Reseller
New folder...
OK
Cancel
instance that contains the Integration Services catalog and the
n in the catalog where the project should be deployed.
< Previous
Next >
Deploy
Cancel

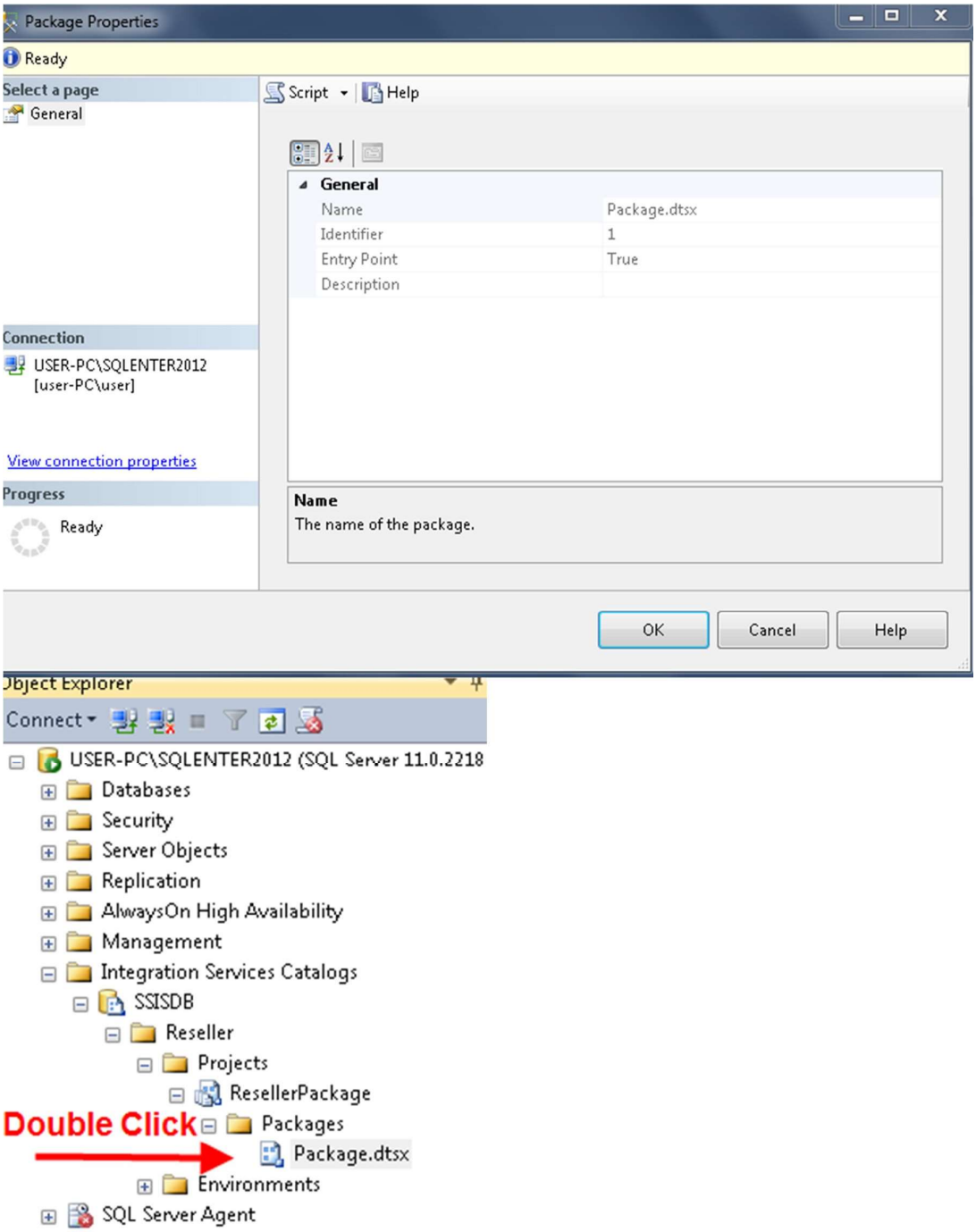

Integration Services Deployment Wizard
Select Destination
Introduction
Select Source
Select Destination
Review
Results
Help
Enter the destination server name and where the project will be located in the Integration Services catalog.
Server name:
USER-PC\SQLENTER2012
Browse...
Path:
/SSISDB/Reseller/ResellerPackage
Browse...
< Previous
Next >
Deploy
Cancel


Integration Services Deployment Wizard
Review
Introduction
Select Source
Select Destination
Review
Results
Help
Review your selections.
Summary:
Use the following arguments to perform this deployment from the command line:
Command line: /Silent /SourcePath:"D:\TeachingStuff\DM\SSISWorkplace\DataFlow\Dat
Source
Path: D:\TeachingStuff\DM\SSISWorkplace\DataFlow\DataFlow\bin\Development\Rese
Destination
Server name: USER-PC\SQLENTER2012
Path: /SSISDB/Reseller/ResellerPackage
< Previous
Next >
Deploy
Cancel

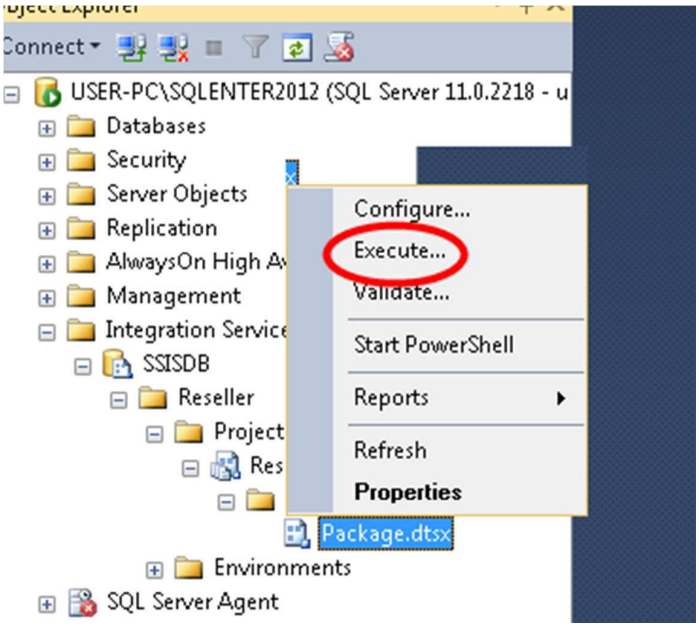


Now go to SQL Server Management Studio and see what happen there:

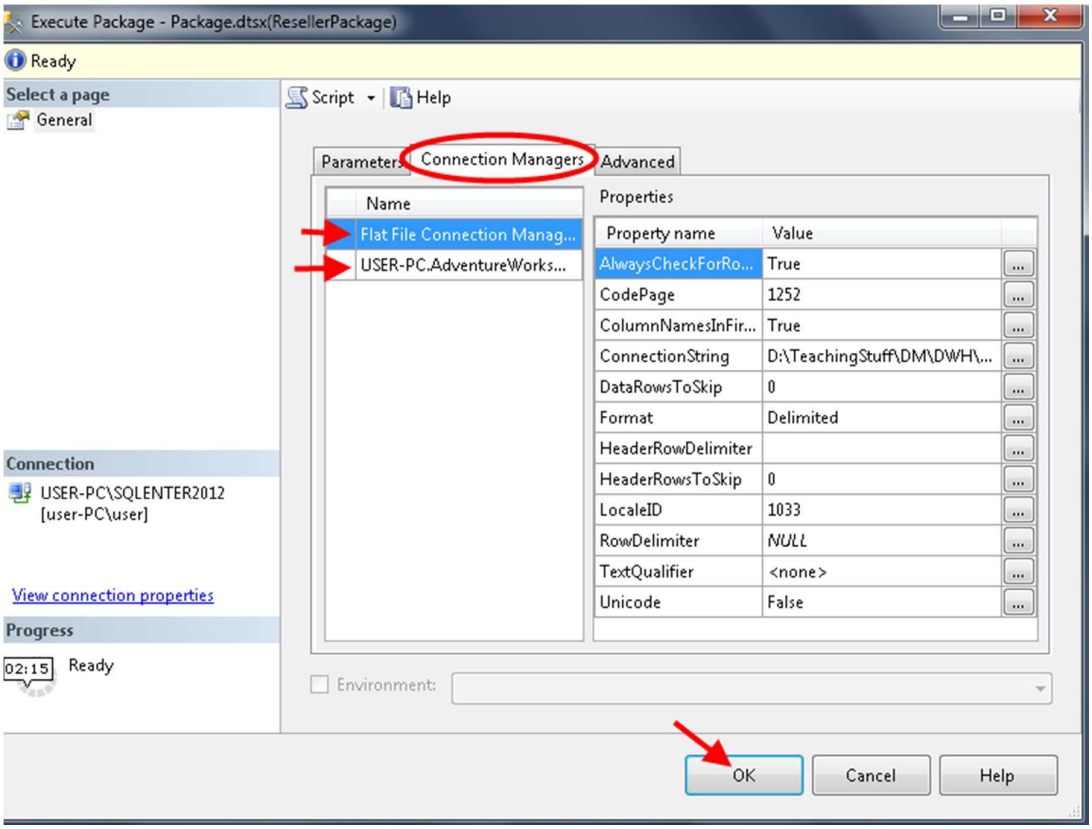
Elect “SSISDB”->Rightclick->Refresh and check it’s a valid package

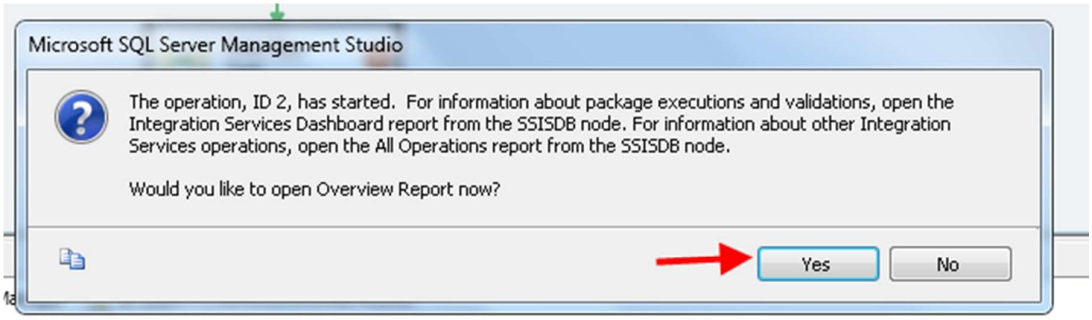


Again Right click-> Execute

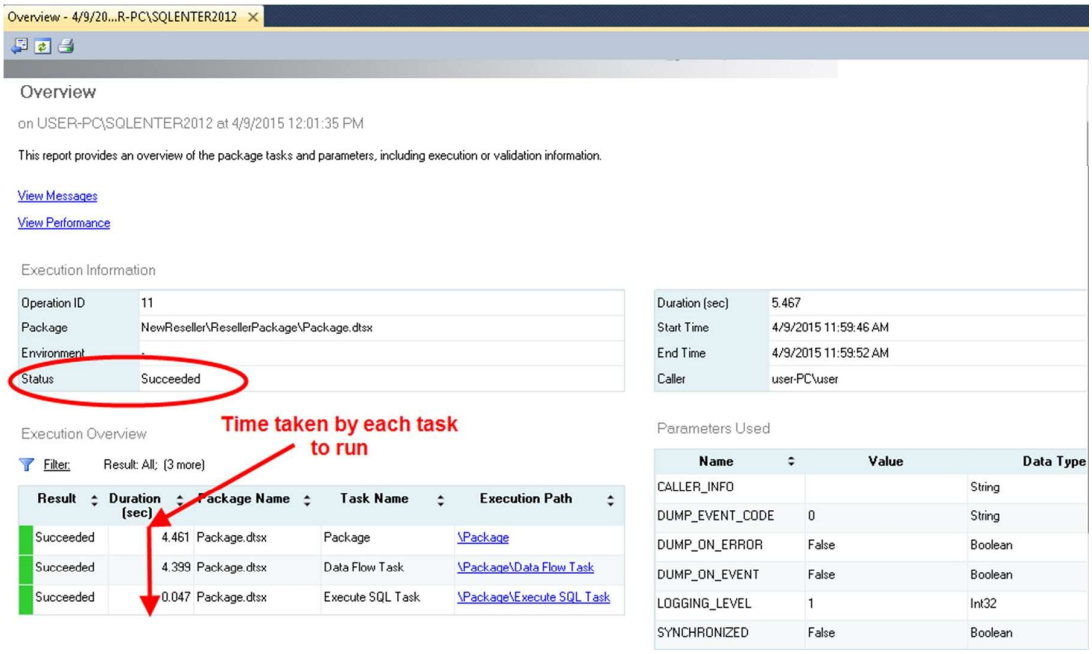


Check for connection files and keep them default then click OK

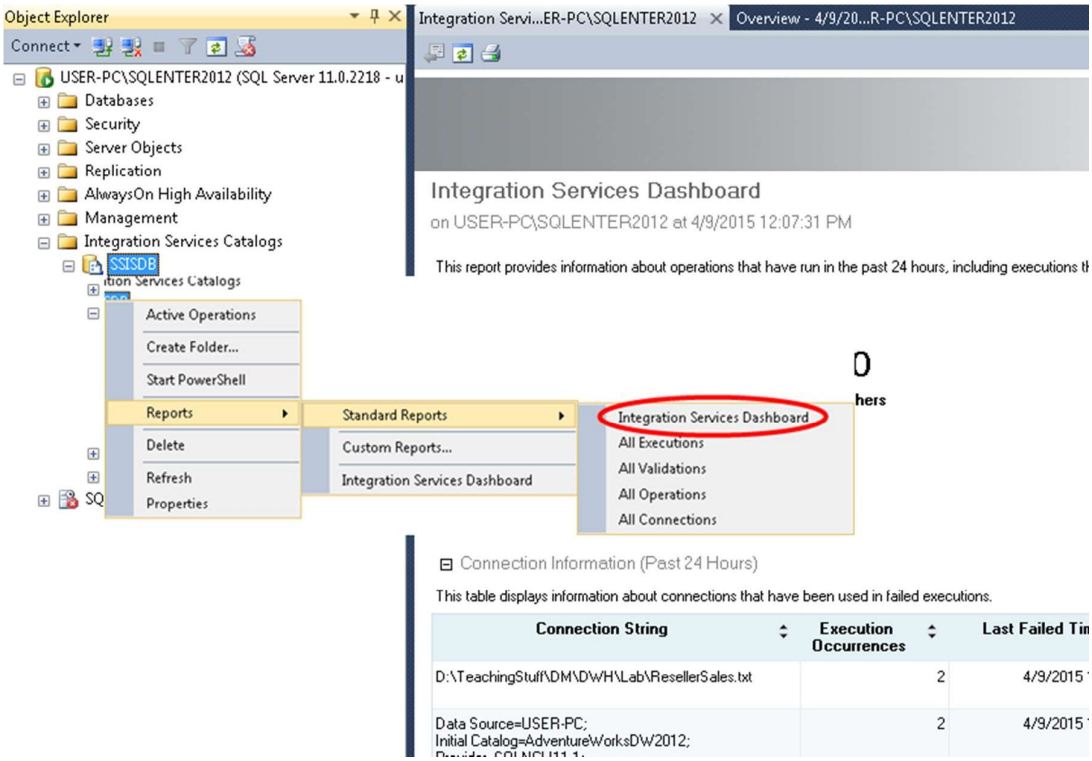


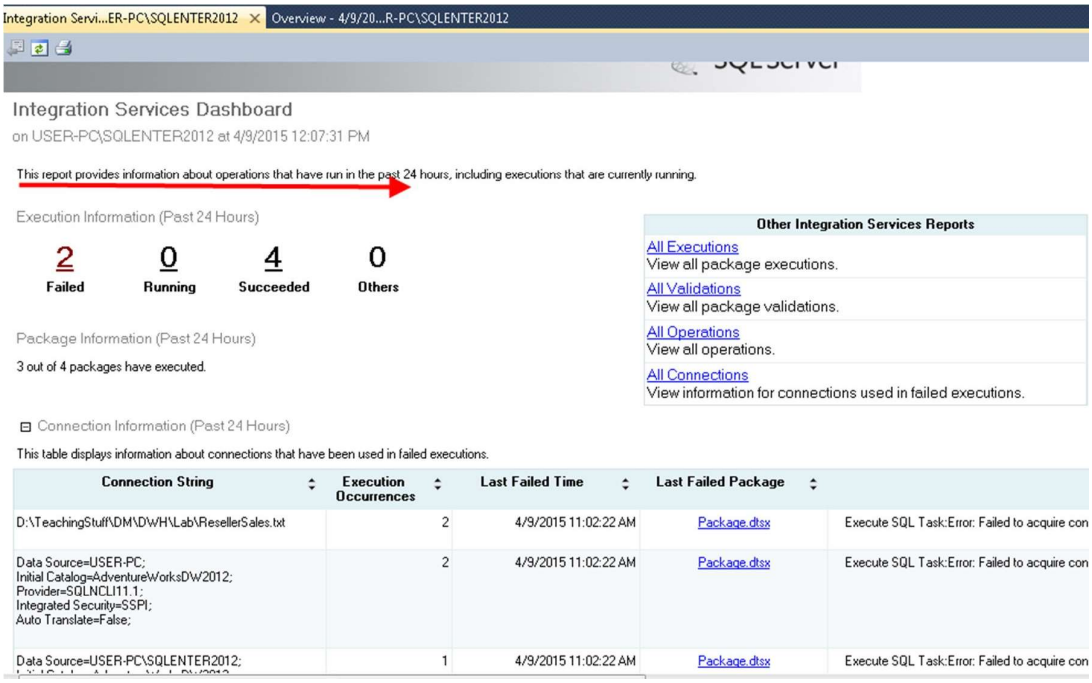


Following overview report will display.



You can view more reports by right clicking on SSISDB




Integration Servi...ER-PC\SQLENTER2012
Overview - 4/9/20...R-PC\SQLENTER2012
Integration Services Dashboard
on USER-PC\SQLENTER2012 at 4/9/2015 12:07:31 PM
This report provides information about operations that have run in the past 24 hours, including executions that are currently running.
Execution Information (Past 24 Hours)
2 Failed
0 Running
4 Succeeded
0 Others
Other Integration Services Reports
All Executions
View all package executions.
All Validations
View all package validations.
All Operations
View all operations.
All Connections
View information for connections used in failed executions.
Package Information (Past 24 Hours)
3 out of 4 packages have executed.
Connection Information (Past 24 Hours)
This table displays information about connections that have been used in failed executions.
Connection String
Execution Occurrences
Last Failed Time
Last Failed Package
D:\TeachingStuff\DM\DWH\Lab\ResellerSales.txt
2
4/9/2015 11:02:22 AM
Package.dtsx
Execute SQL Task:Error: Failed to acquire con
Data Source=USER-PC;
Initial Catalog=AdventureWorksDW2012;
Provider=SQLNCLI11.1;
Integrated Security=SSPI;
Auto Translate=False;
2
4/9/2015 11:02:22 AM
Package.dtsx
Execute SQL Task:Error: Failed to acquire con
Data Source=USER-PC\SQLENTER2012;
1
4/9/2015 11:02:22 AM
Package.dtsx
Execute SQL Task:Error: Failed to acquire con