**Hands on Training Lab**

**SQLServer Integration Services**

Table of Contents

[Lab 1-Data load from Flat File into Relational Table 3](#_Toc456962468)

[Lab 2-Derived Column transformation 8](#_Toc456962469)

[Lab 3- Aggregate Transformation 21](#_Toc456962470)

[Lab 4- Conditional split transformation 28](#_Toc456962471)

[Lab 5- Look Up Transformation 34](#_Toc456962472)

[Lab 6- Copy Column Transformation 48](#_Toc456962473)

[Lab 7- Merge Transformation 54](#_Toc456962474)

[Lab 8- Merge join Transformation 60](#_Toc456962475)

[Lab 9- Union All Transformation 63](#_Toc456962476)

[Lab 10-Slowly changing Dimension 65](#_Toc456962477)

[Lab 11- Extracting, Transforming and Loading Data 73](#_Toc456962478)

[Lab 12- Execute SQL Task & Script task 78](#_Toc456962479)

[Lab 13-File System Task 86](#_Toc456962480)

[Lab 14-For each loop container with Execute SQL task and Script task 93](#_Toc456962481)

[Lab 15- Execute Process Task 100](#_Toc456962482)

[Lab 16- Using Error Redirect Row 107](#_Toc456962483)

[Lab 17- Transaction support and Checkpoint 110](#_Toc456962484)

[Lab 18- Using Data Viewer 117](#_Toc456962485)

[Lab 19- Using Package Logging 119](#_Toc456962486)

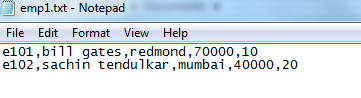
[Lab20-Project deployment 121](#_Toc456962487)

[Lab21- Package Execution using SQL Server Agent 130](#_Toc456962488)

Lab 1-Data load from Flat File into Relational Table

|  |  |
| --- | --- |
| **Objective** | How to extract data from flat file and load into a relational table. |
| **Lab Setup** | * SSDT tool * Login details for connecting to the database |

1. Create a separate folder “Project” for placing input data feed file
2. In that folder create a txt file emp1.txt and type this data



1. Connect to sql server using SSMS and create the following table

createtable **emp1**

(

**empno** varchar(10)primarykey,

**empname** varchar(50),

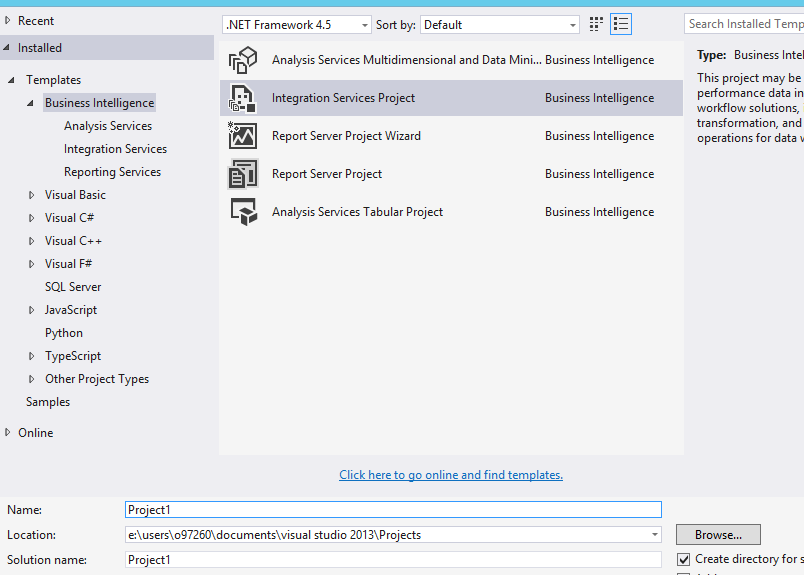
**city** varchar(50),

**sal** money,

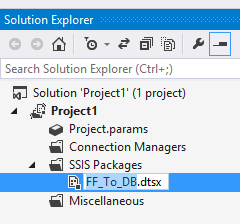
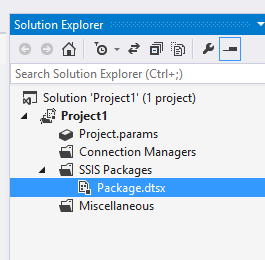
**deptno** int

)

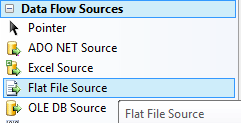
1. Start a new SSIS project. Start BIDS and click file => new project
   * Select integration services project among Business Intelligence Projects.
   * Rename it to “Project1”.



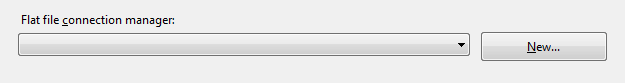
1. From Solution Explorer rename package.dtsx to FF\_To\_DB.dtsx



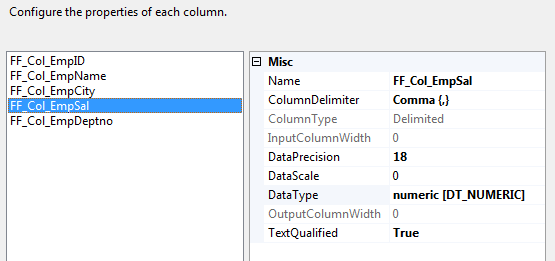
1. In the package designer, goto data flow tab and click the link in the middle to create a new data flow task. (this data flow task is automatically called from control flow).
2. In the data flow task designer drag “Flat File Source task” from toolbox
   * Rename it “Emp Flat File Data”



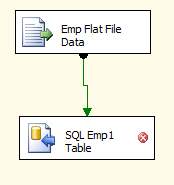
1. Double click the above dragged task to popup a dialog box. Click “New” Button to create a FF connection Manager.



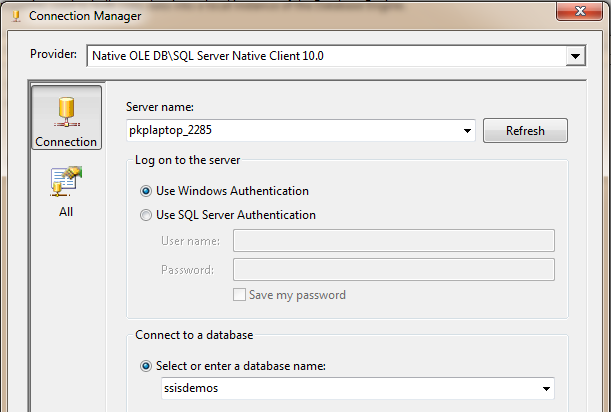
1. Give Connection Manager Name as: FF\_Emp\_CM
2. For filename, browse and locate emp1.txt.
3. Click Columns TAB to ensure columns identified properly
4. Click Advanced TAB and rename cols as below:
5. For Sal field set data type “currency [DT\_CY]”
6. For Deptno field set data type “four-byte signed integer [DT\_I4]”



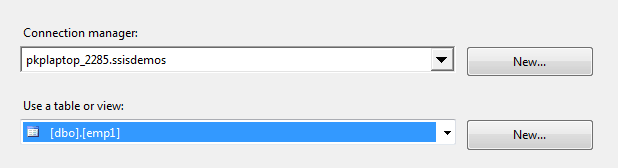
1. Click OK
2. In Toolbox , from Destinations section drag sql server destination
   * Rename it as SQL Emp1 Table
3. Connect the green connector of Flat File Source to SQL Emp1 Table destination.



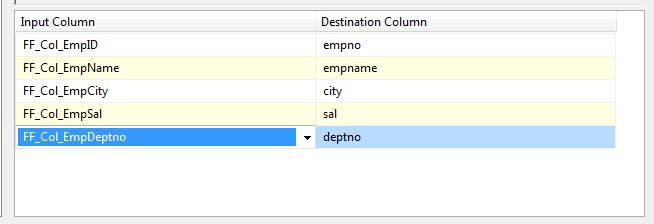
1. Double Click SQL Emp1 Table destination task.
   * Click New to create New Connection Manager, again click new
   * Give the connection details as appropriate



1. Test Connection, Click Ok twice.
2. Select the emp1 table created earlier in the dialog box as below:



1. Go to Mappings tab and map columns as below:

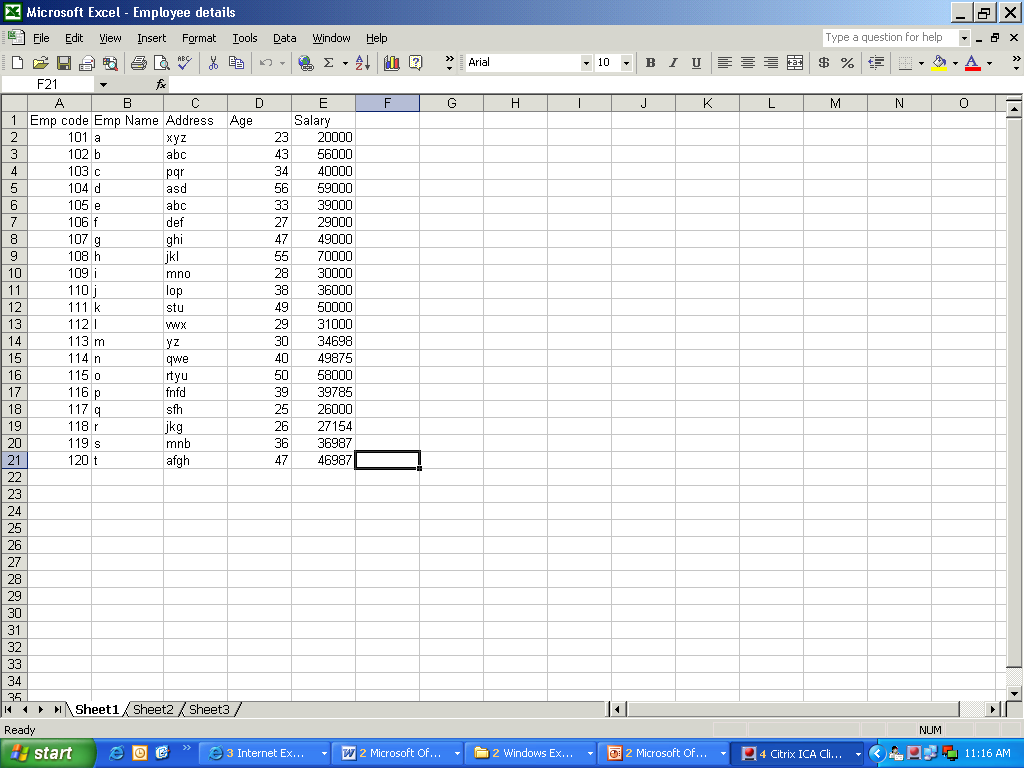


1. Click Ok to come to data flow designer surface
2. Click Run Button  on toolbar, the package should run successful. Click Stop Button explicitly on toolbar.
3. Goto SSMS and check the sql server table data using select\*from **emp1**

Lab 2-**Derived Column transformation**

|  |  |
| --- | --- |
| **Objective** | To load the data from excel sheet to OLE DB database. |
| **Lab Setup** | * SSDT tool * Existing SSIS Project and connection to database |

**Source:**

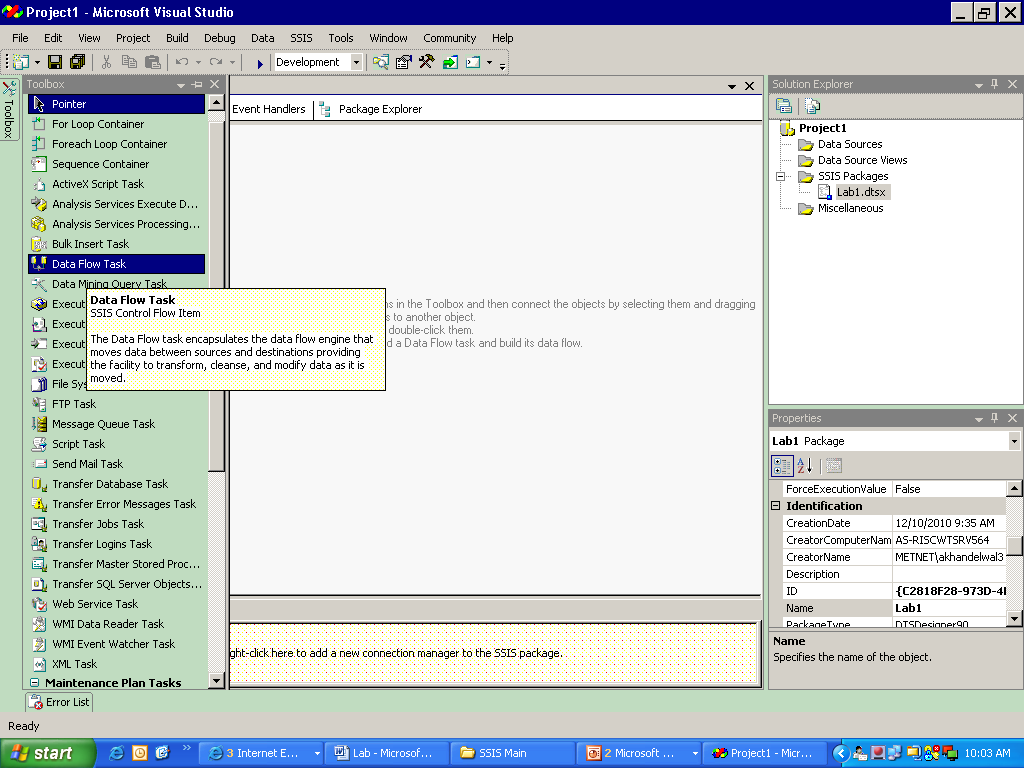


Source File attached here:

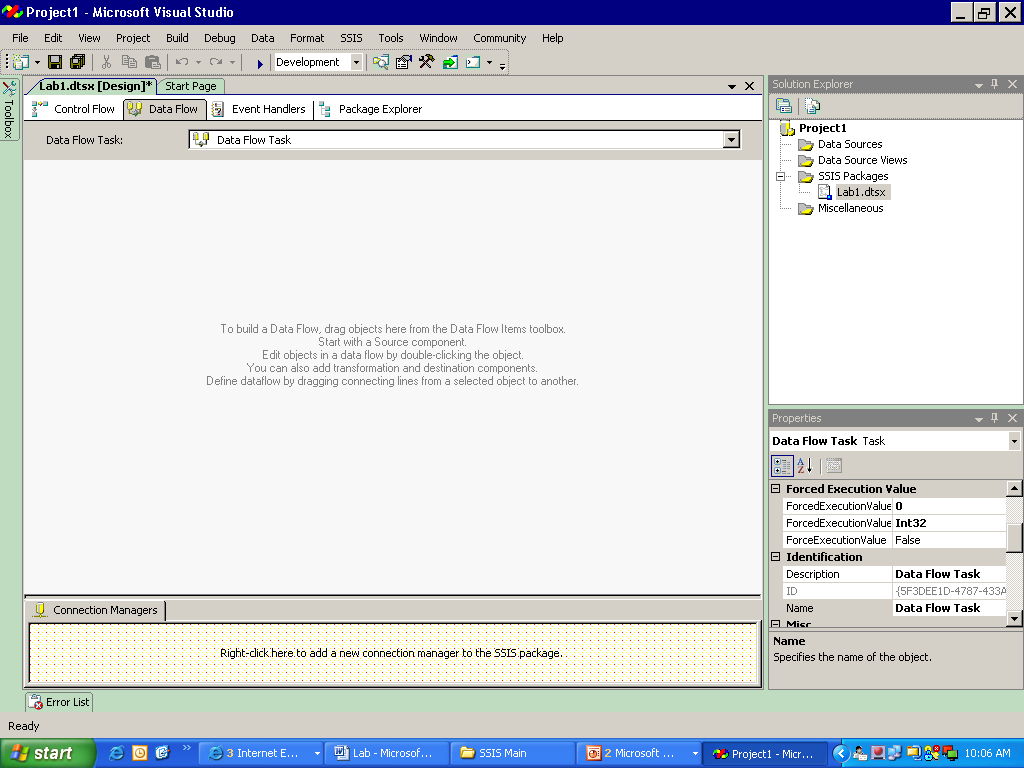
1)Create a new package named “Lab1” under the already created project.

2)Click on the toolbox and drag and drop the Data flow task on control flow task:

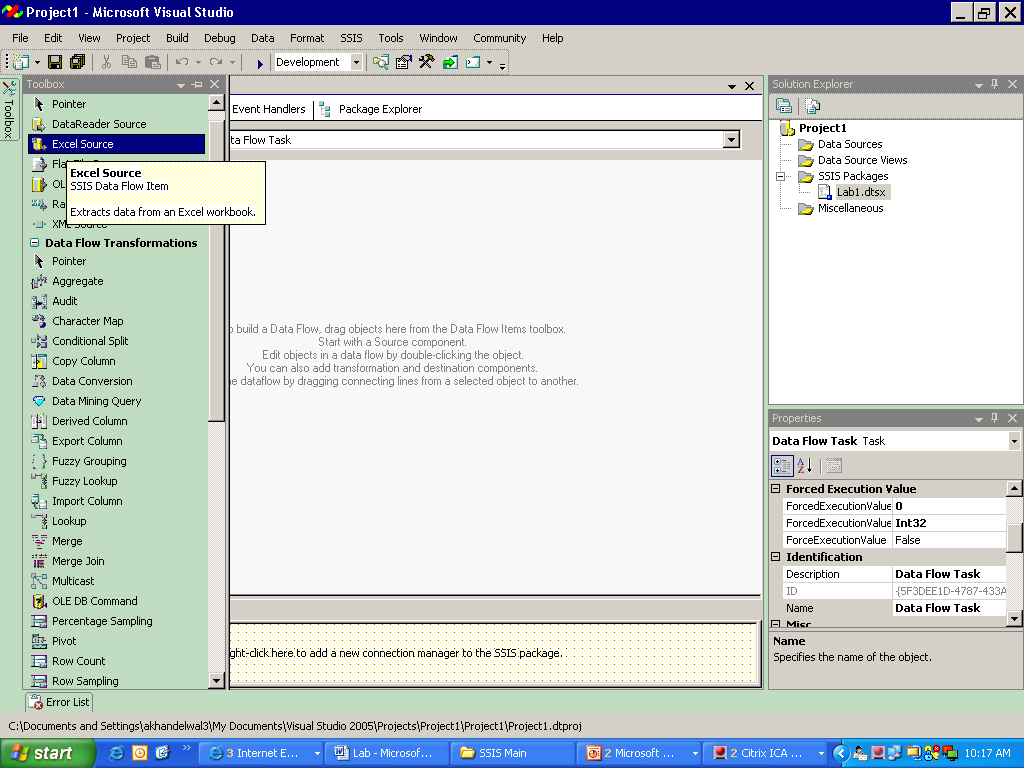
Change the package name



3) Double click the data flow task or click on the data flow task and select the data flow from the tab to begin the mapping:

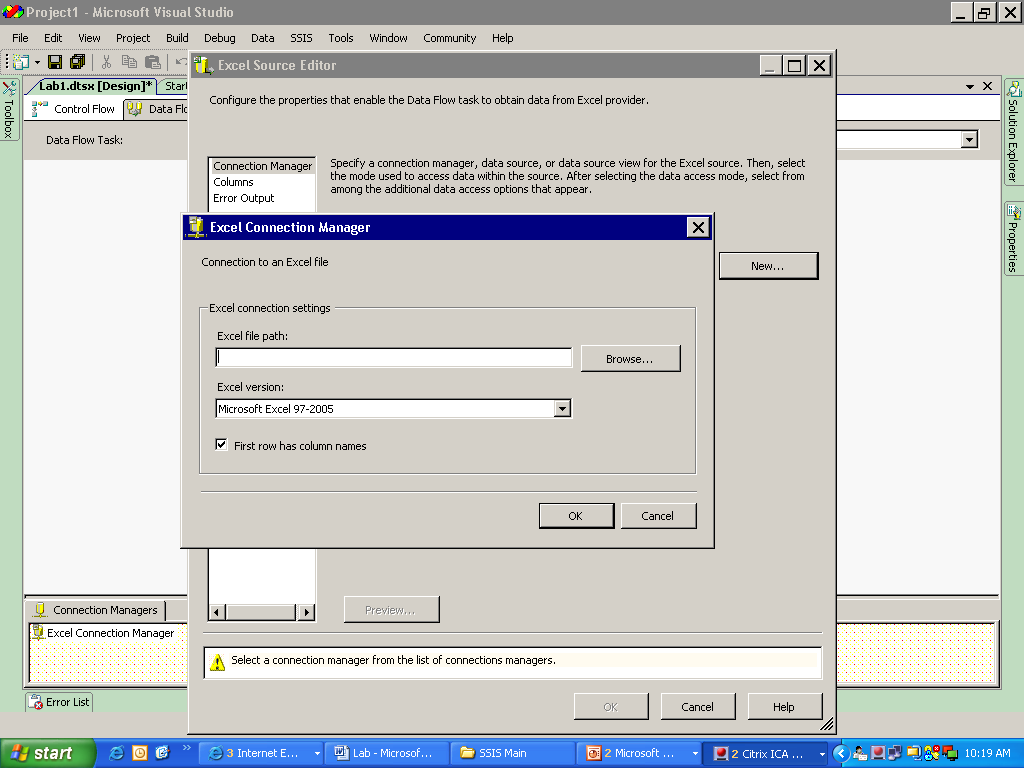


4) Select a source from the toolbox. Here we are taking the excel source:



5) Double click the source to open the editor:

5.1) Create a new connection manager:

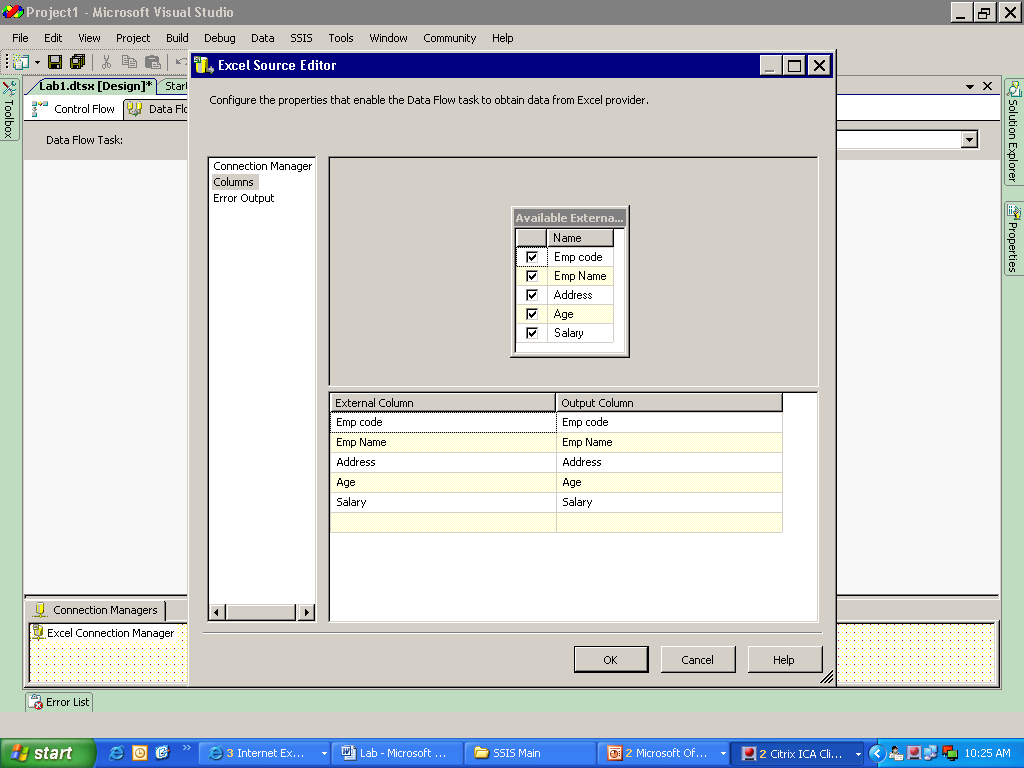


If the excel file contains columns then you must check the box for “First row has column names” check box.

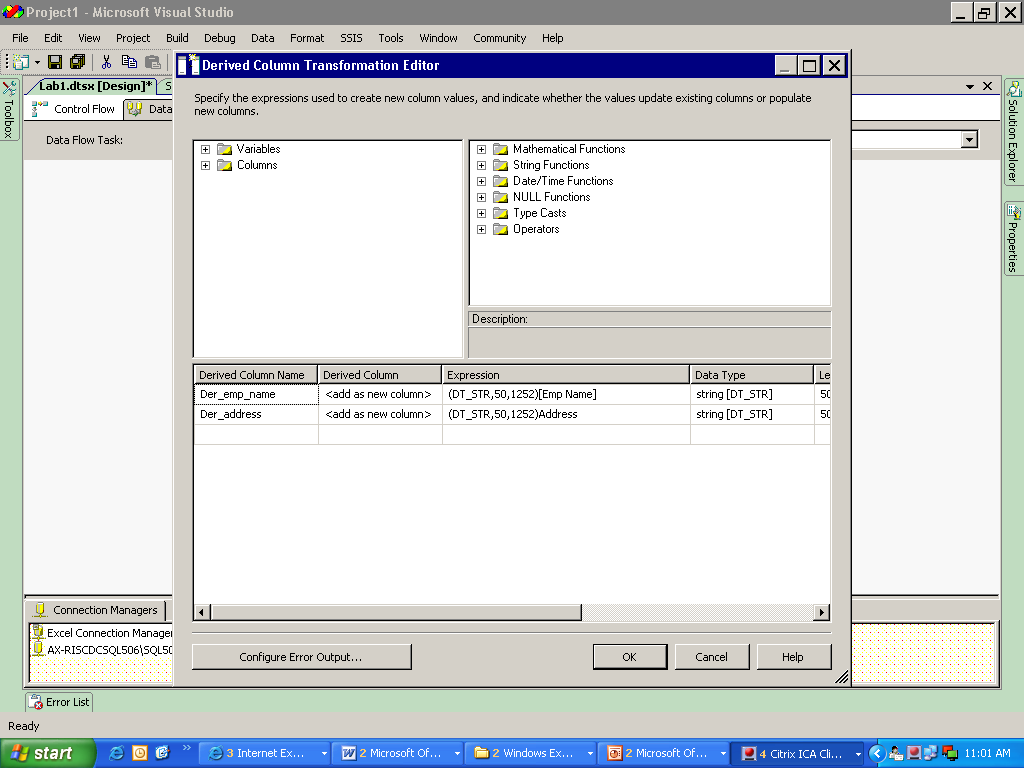
Click ‘OK’ after selecting the file and select Data access mode as – Table or view and name of the excel sheet as ‘sheet1’ if the data is in that sheet.

You can also click the “preview” button to view the data in the source table.

5.2) Click on columns to select the required columns in the target table:



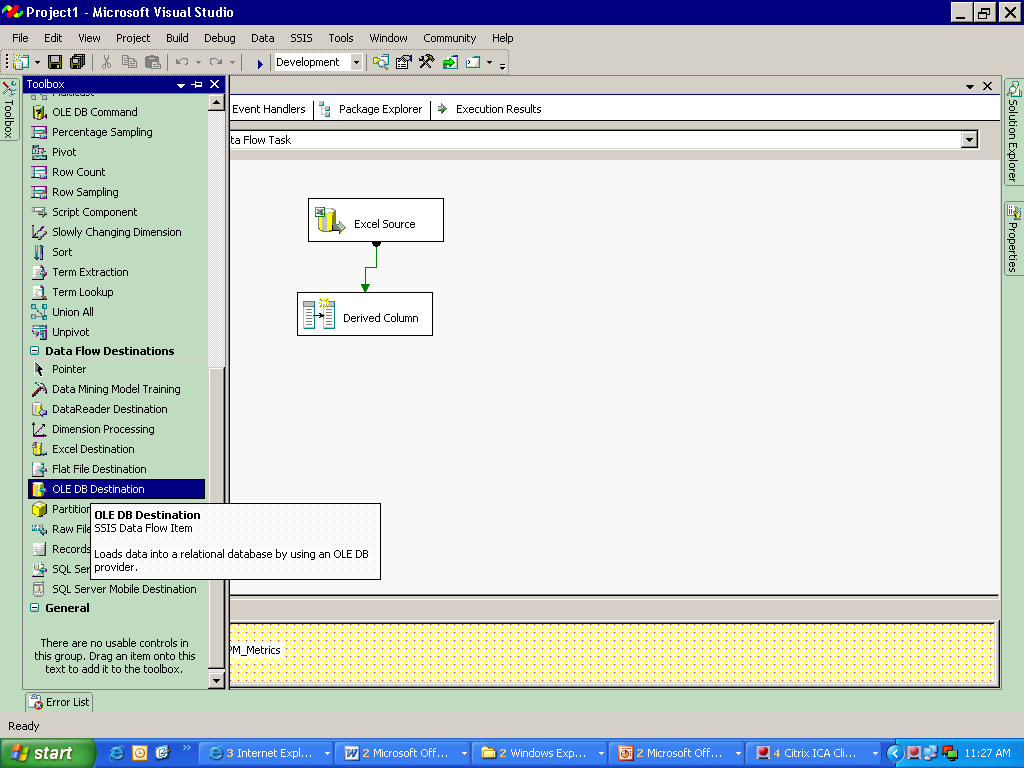
6) Select Derived column transformation from the toolbox and double click it to get the editor:



Change the data type of Emp\_name and address (Unicode in source) to string and click OK.

7) Create a target table in the database (SQL Server) if not present, to transform the data:

Select OLE BD Destination as target table:



Connect Derived column transformation and target using the GREEN connector:

7.1) Double click OLE DB destination to open the editor. Create a New Connection Manager:



Refer to Point 7.1

Refer to Point 7.2

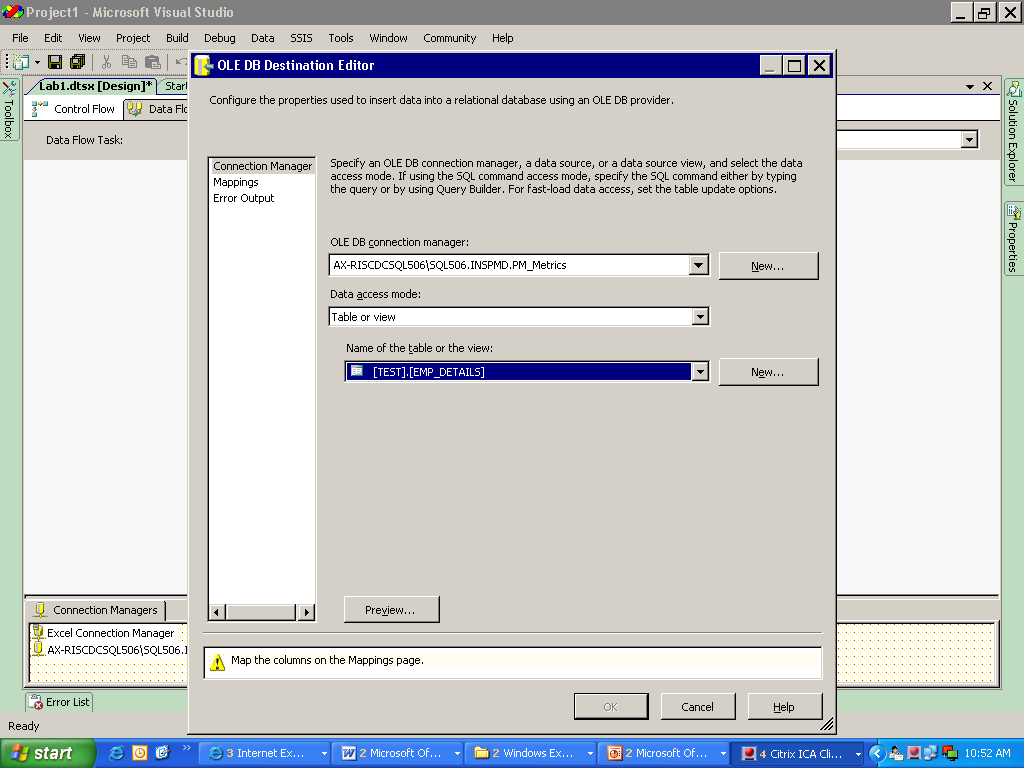
7.2) Select the server name, use SQL Server Authentication (use valid user name and password) and select a database name and

test the connection.

Click OK till you are able to view the editor with a connection manager.

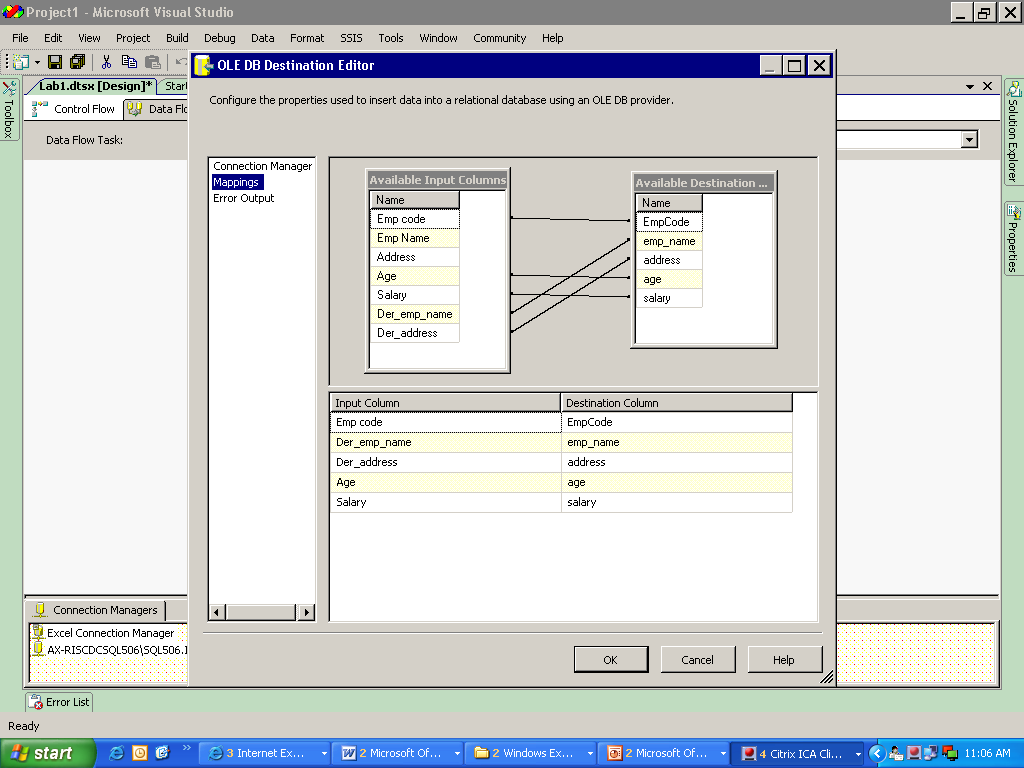
7.3) Select Data Access Mode as- Table or view.

7.4) And then select the name of the target table.

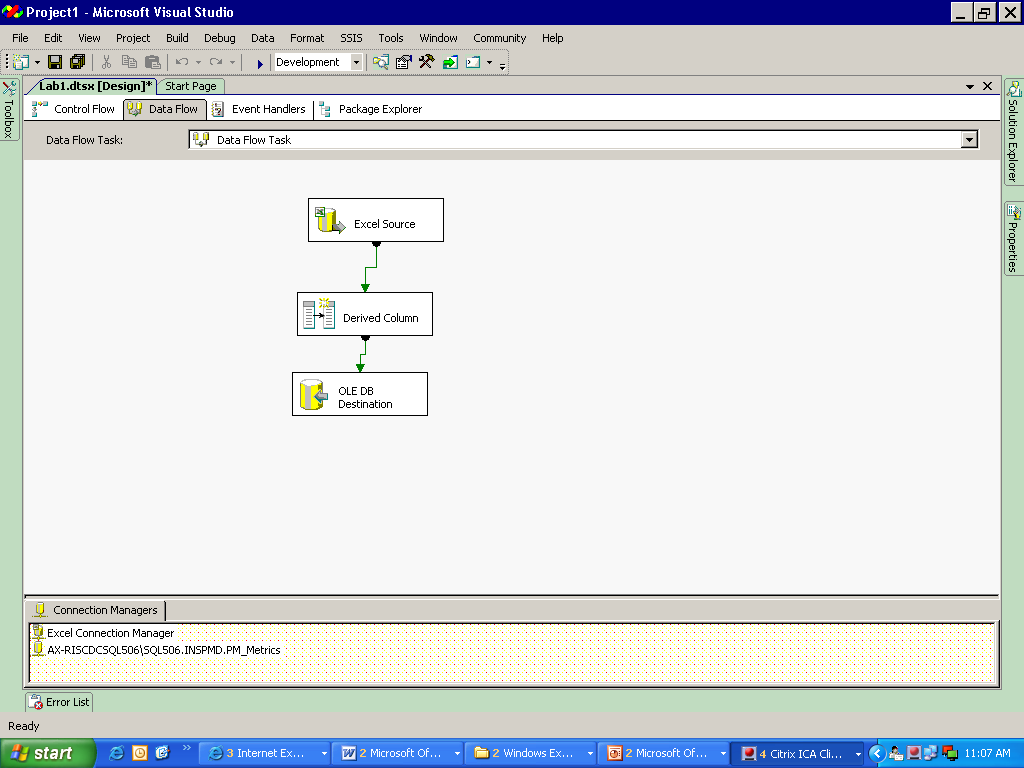


7.3 And 7.4

7.5) Click on Mappings to map the data from source to target.

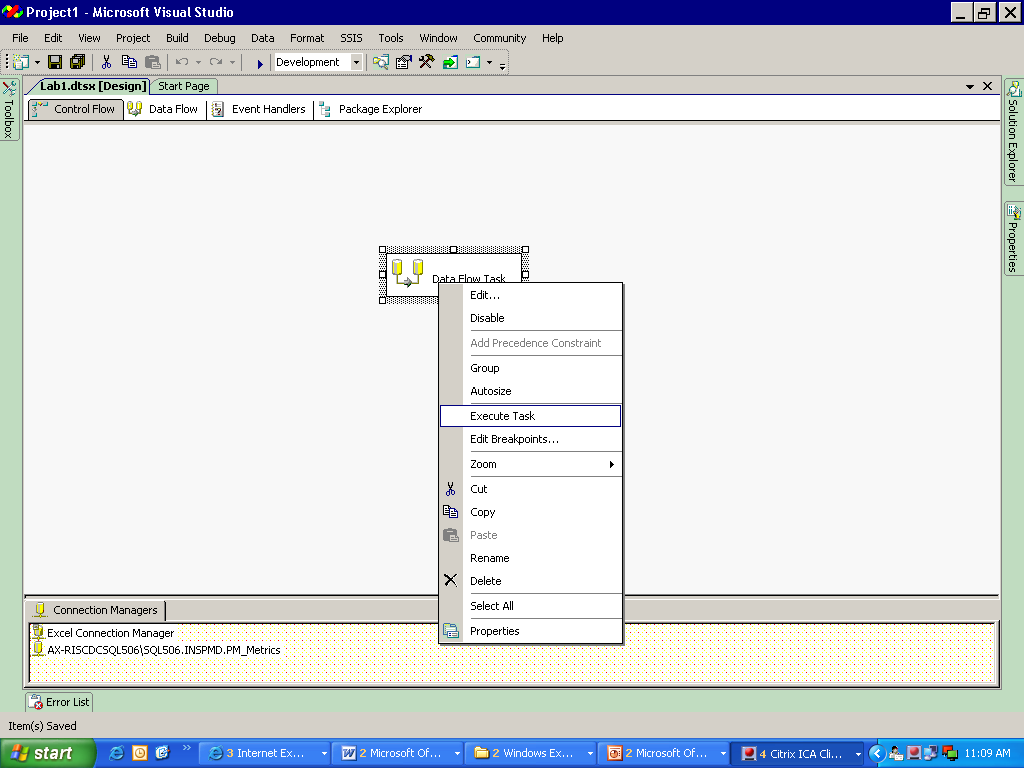


The mapping from source to target would be like:

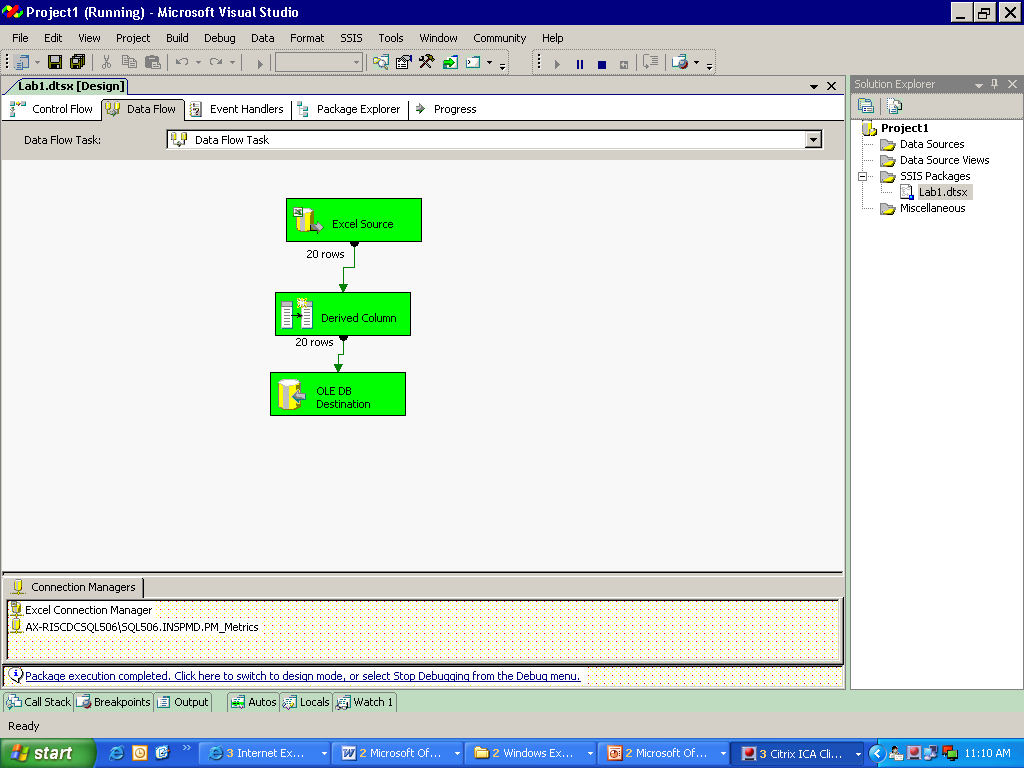


8) Save the project.

Switch to the control flow task and right click on data flow task. Click on Execute task.



9)After executing the data flow and on successful completion the Data Flow will be as follows:



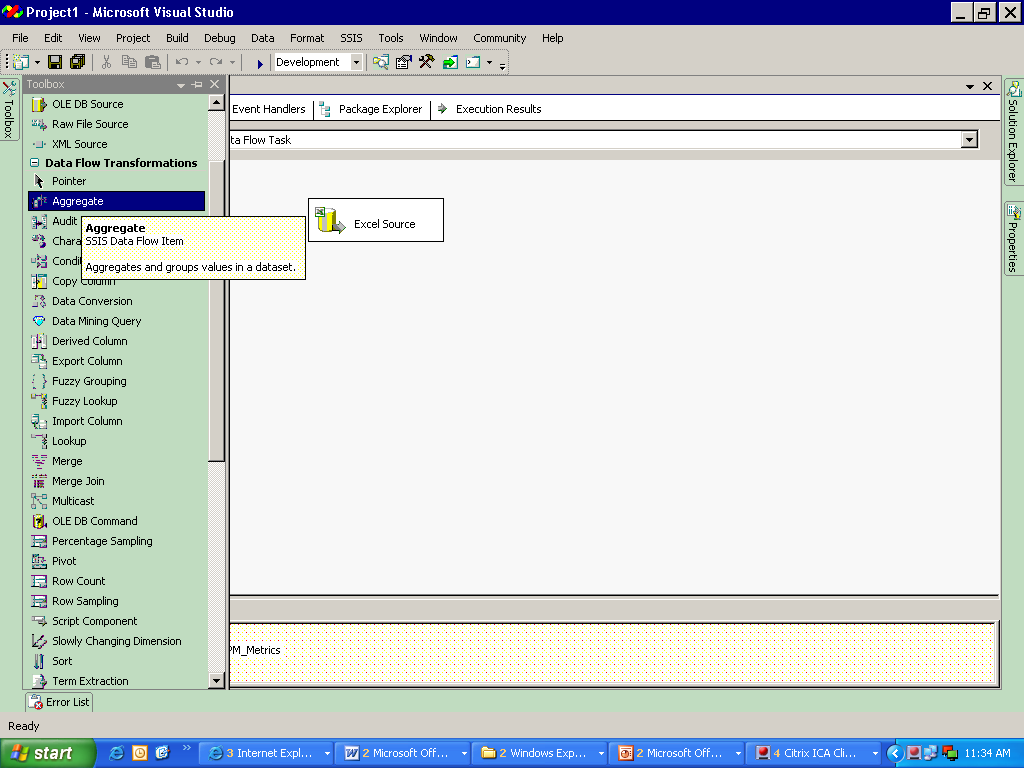
**You can check the data in the target table using SQL Server.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Emp\_code** | **emp\_name** | **Address** | **Age** | **salary** |
| 101 | a | xyz | 23 | 20000 |
| 102 | b | abc | 43 | 56000 |
| 103 | c | pqr | 34 | 40000 |
| 104 | d | asd | 56 | 59000 |
| 105 | e | abc | 33 | 39000 |
| 106 | f | def | 27 | 29000 |
| 107 | g | ghi | 47 | 49000 |
| 108 | h | jkl | 55 | 70000 |
| 109 | i | mno | 28 | 30000 |
| 110 | j | lop | 38 | 36000 |
| 111 | k | stu | 49 | 50000 |
| 112 | l | vwx | 29 | 31000 |
| 113 | m | yz | 30 | 34698 |
| 114 | n | qwe | 40 | 49875 |
| 115 | o | rtyu | 50 | 58000 |
| 116 | p | fnfd | 39 | 39785 |
| 117 | q | sfh | 25 | 26000 |
| 118 | r | jkg | 26 | 27154 |
| 119 | s | mnb | 36 | 36987 |
| 120 | t | afgh | 47 | 46987 |

Lab 3- **Aggregate Transformation**

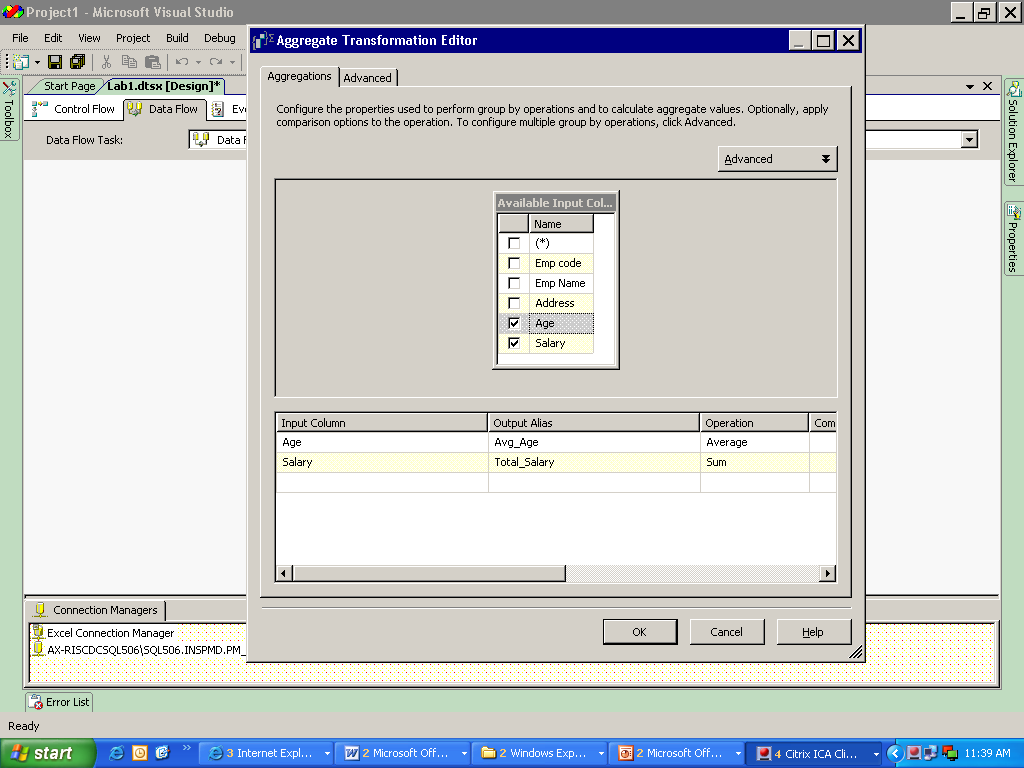
|  |  |
| --- | --- |
| **Objective** | To find the average age in the organization and the total salary paid |
| **Lab Setup** | * SSDT tool * Existing SSIS Project and Packgae |

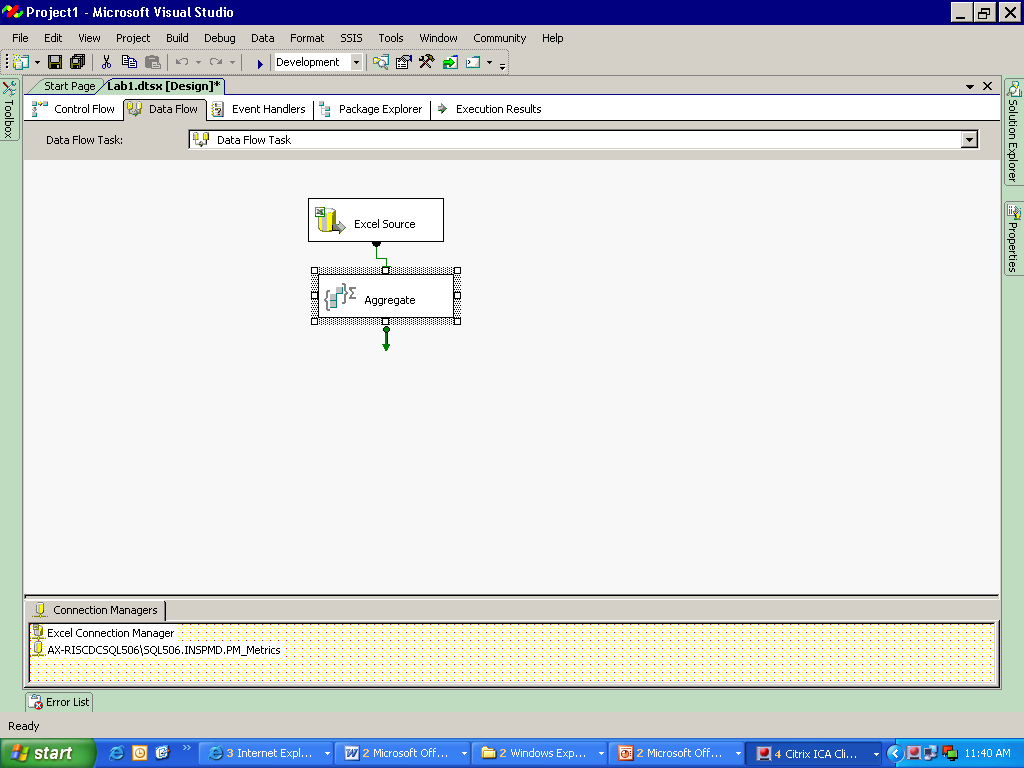
1) In the Data flow Task after selecting the source table, select aggregate transformation for the tool box.



2) Double click the transformation to open the editor.

* Select age and salary.
* Select the ‘Operation’ on AGE as average and on salary as SUM and then click on OK.

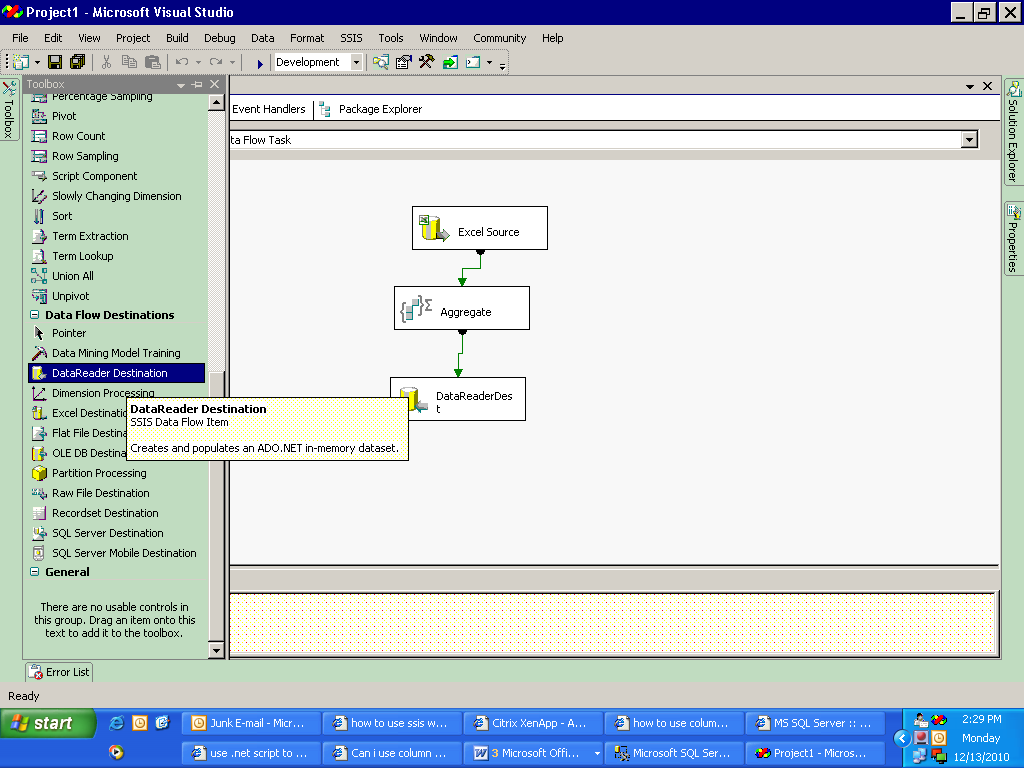




**Notes:**

You can use DATA READER Destination as dummy target table. This table is not save anywhere. It is only used to check the final data when we do not want to make any change or load the final data into the target table.

**This destination can be used without any configuration changes, in effect creating a ‘null’ destination. The data goes nowhere but we can trouble shoot any issue in our derived column upstream before**

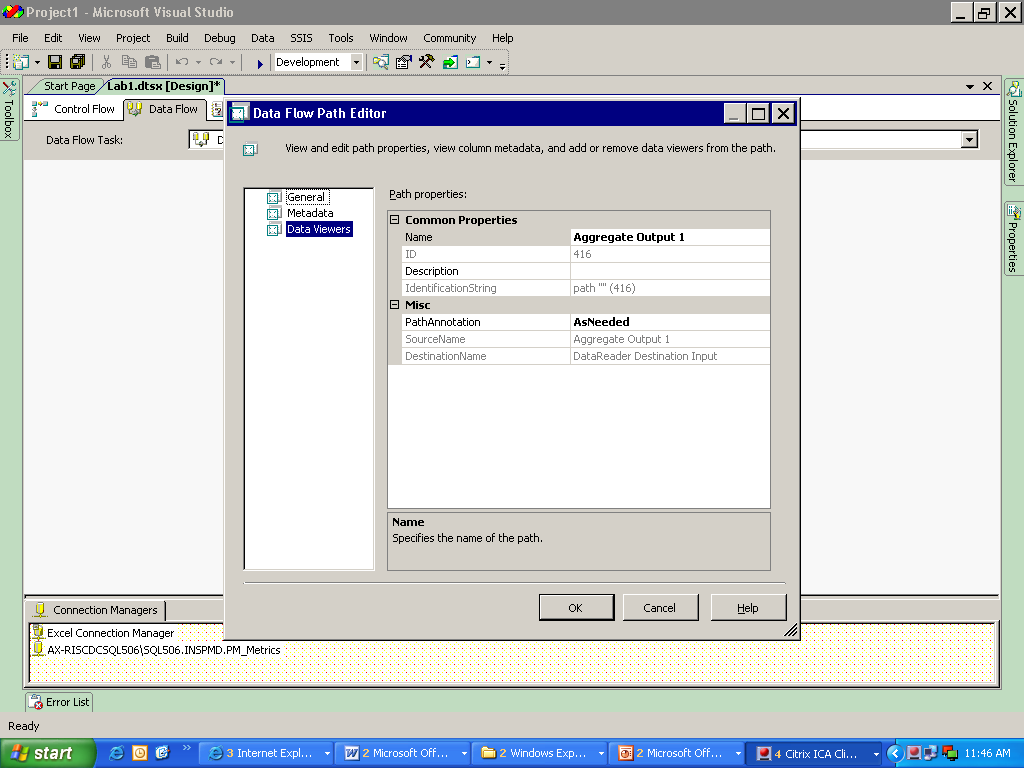


Click on the green link to generate a data viewer

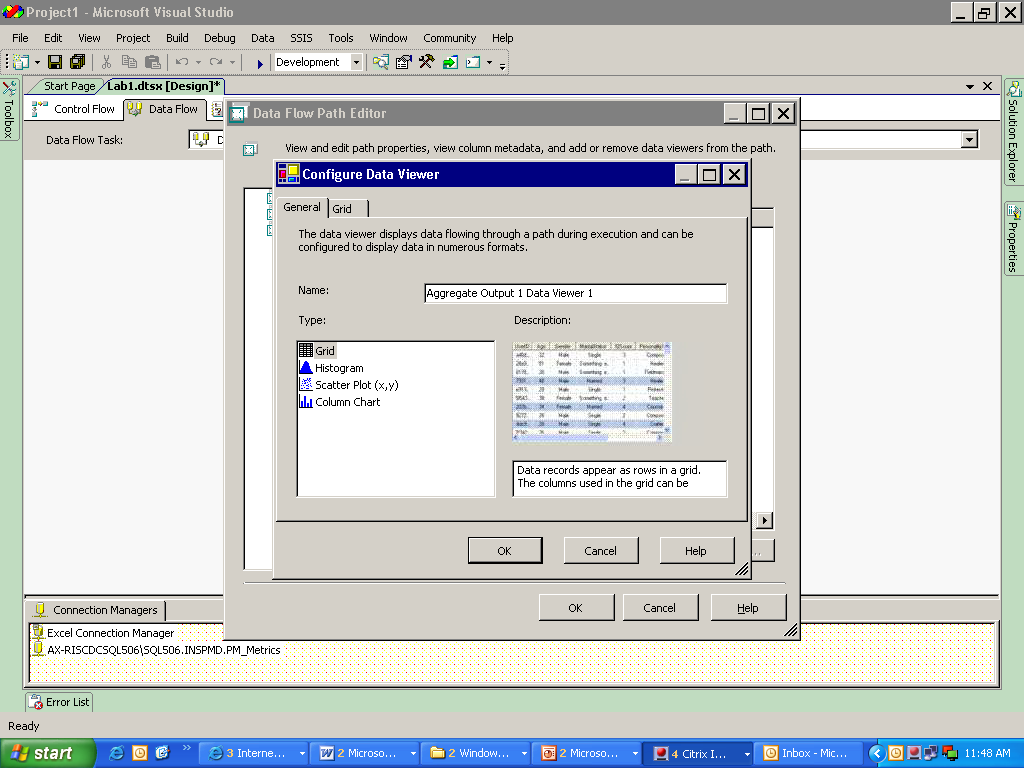
To view the data we have to use DATA VIEWER.

3) Select DATA READER Destination from the toolbox.

* Link the aggregate transformation to the DATA READER Destination.
* Double click the link to view the data flow editor and select Data Viewers.



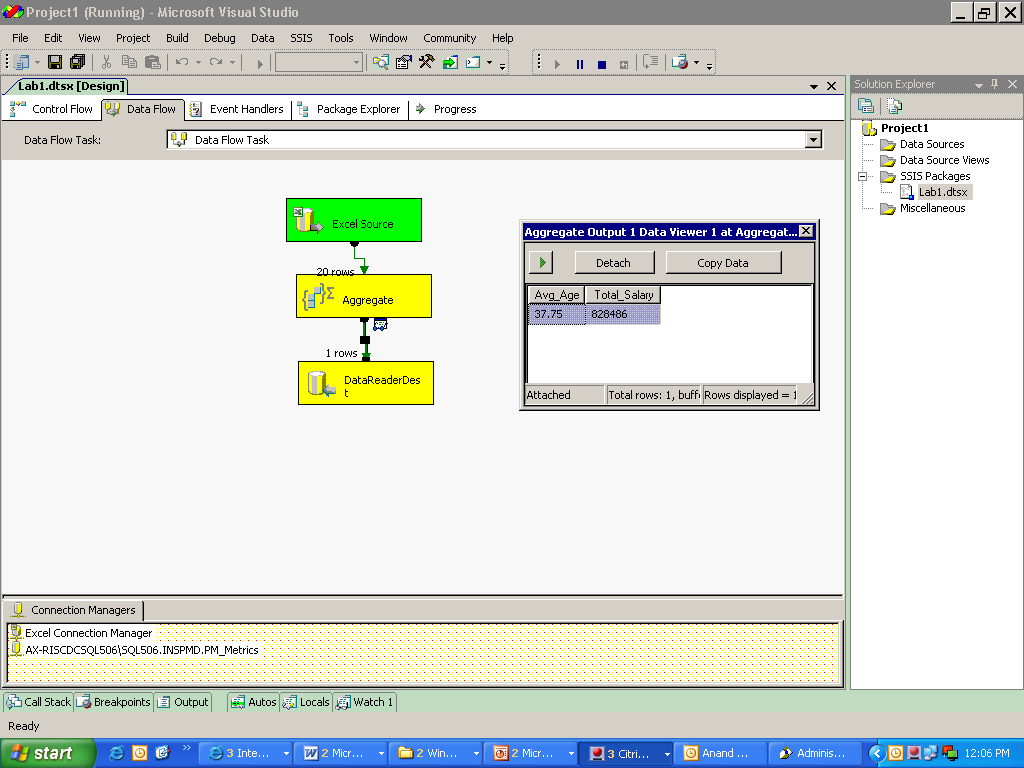
* Add a new data viewer and select Grid and click OK twice.



4)A small icon of GRID will be visible near the joining arrow.

5)Save and execute the DATA FLOW TASK.

6)On successful execution a grid with output will be shown as:



Grid Icon

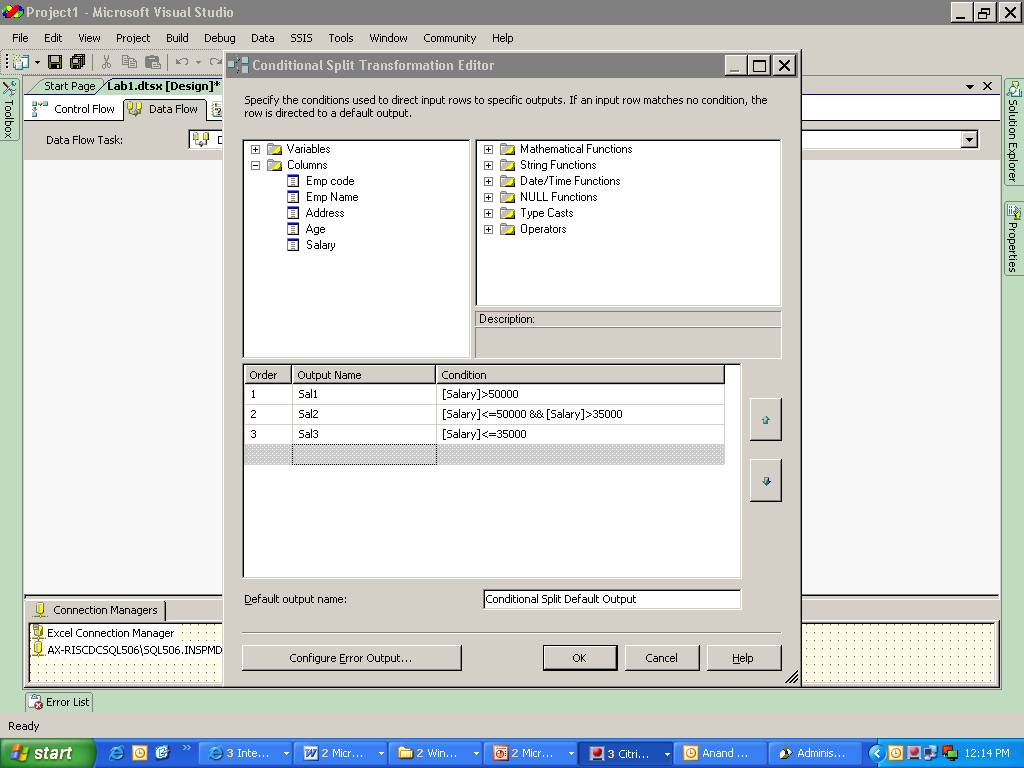
Output of data viewer

7)As soon as the data viewer is closed the execution gets completed.

Lab 4- **Conditional split transformation**

|  |  |
| --- | --- |
| **Objective** | To load different tables with specific salary range ( >50000, >35000 and <50000, <=35000) using conditional split and derived column transformation |
| **Lab Setup** | * SSDT tool * Existing SSIS project and Package |

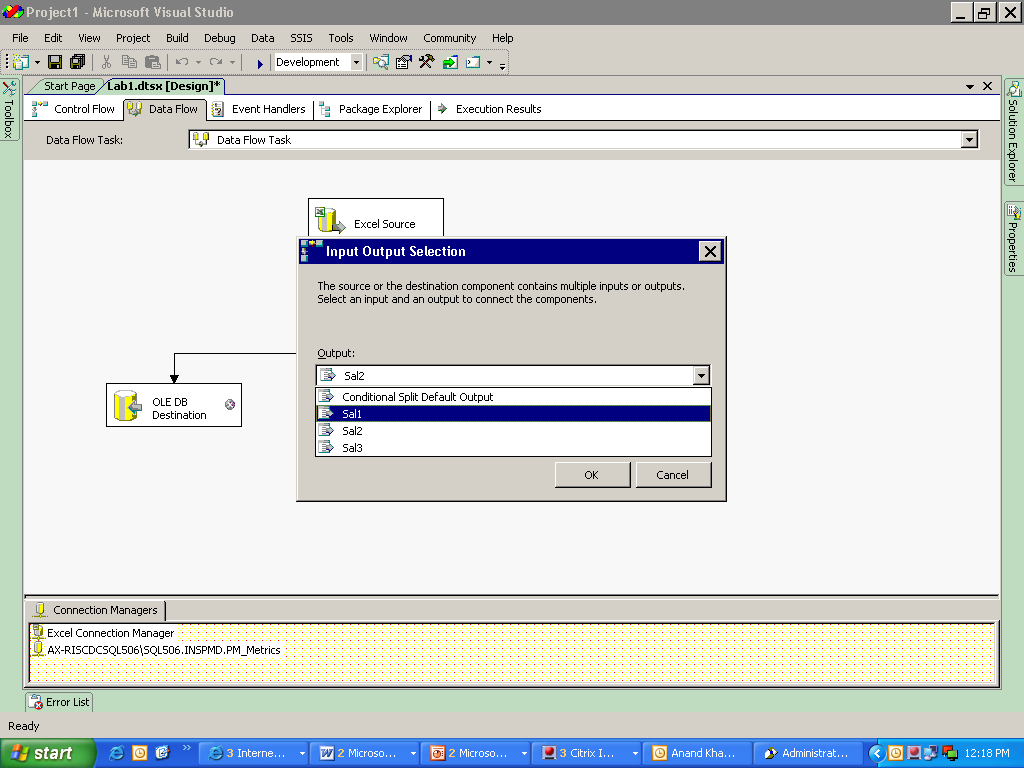
1)In the Data flow Task after selecting the source table, select conditional split transformation from the tool box and double click to get the editor.



2)In the editor use different conditions according to which you want to divide the data.

3)Insert a Derived column transformation between source and conditional split and change the data type to name and address.

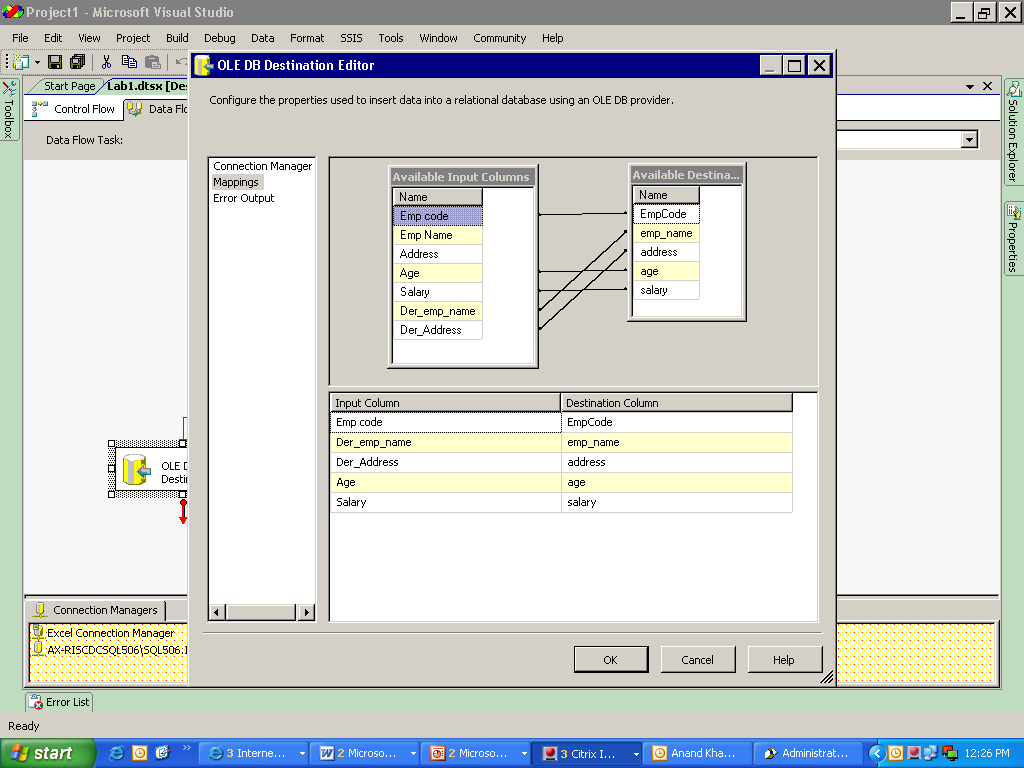
* + Select target tables (OLE DB) and join with the transformation (select valid output).



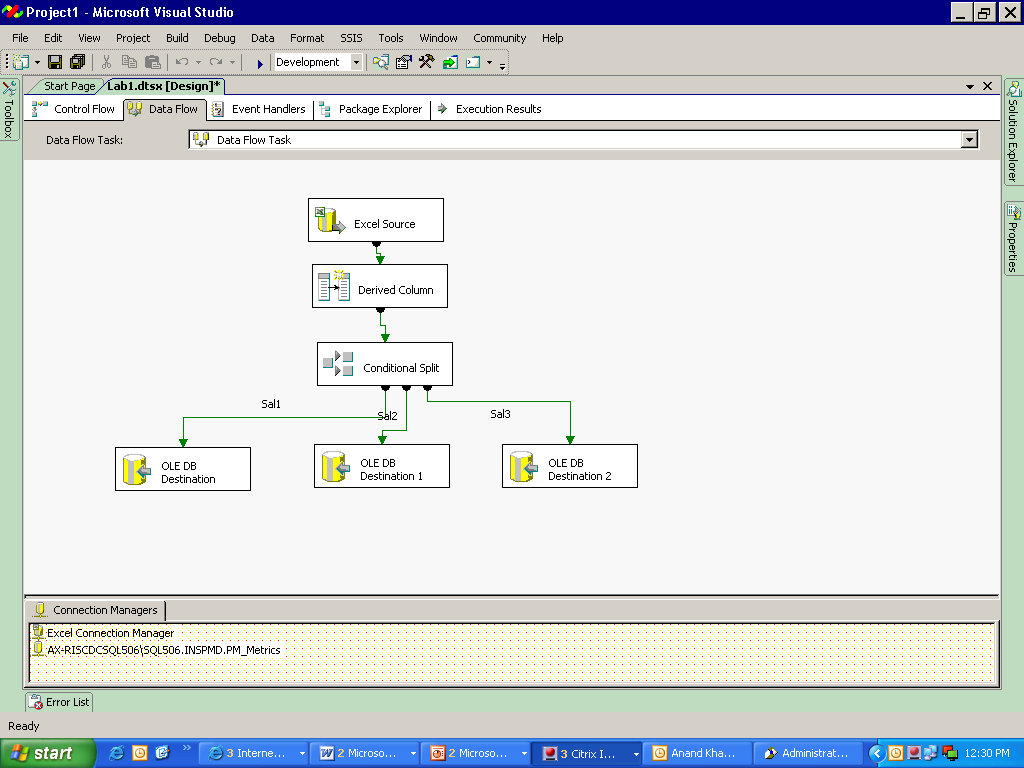
* You can use the existing connection manager or can create a new connection manager for

the target table.

* Use DATA ACCES MODE as –Table or view.
* Select appropriate mapping.



4)Repeat for all the target tables.



List of connection managers.

5) Save and execute.



6)Using “SELECT” statement in SQL server.

**Destination 1 :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Emp\_code** | **emp\_name** | **Address** | **Age** | **salary** |
| 102 | b | abc | 43 | 56000 |
| 104 | d | asd | 56 | 59000 |
| 108 | h | jkl | 55 | 70000 |
| 115 | o | rtyu | 50 | 58000 |

**Destination 2 :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Emp\_code** | **emp\_name** | **Address** | **Age** | **salary** |
| 103 | c | pqr | 34 | 40000 |
| 105 | e | abc | 33 | 39000 |
| 107 | g | ghi | 47 | 49000 |
| 110 | j | lop | 38 | 36000 |
| 111 | k | stu | 49 | 50000 |
| 114 | n | qwe | 40 | 49875 |
| 116 | p | fnfd | 39 | 39785 |
| 119 | s | mnb | 36 | 36987 |
| 120 | t | afgh | 47 | 46987 |

**Destination 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Emp\_code** | **emp\_name** | **Address** | **Age** | **salary** |
| 101 | a | xyz | 23 | 20000 |
| 106 | f | def | 27 | 29000 |
| 109 | i | mno | 28 | 30000 |
| 112 | l | vwx | 29 | 31000 |
| 113 | m | yz | 30 | 34698 |
| 117 | q | sfh | 25 | 26000 |
| 118 | r | jkg | 26 | 27154 |

Lab 5- **Look Up Transformation**

|  |  |
| --- | --- |
| **Objective** | To check in the database whether there is any change in the address of the employee or if any new employee details are added then load the data in the database. |
| **Lab Setup** | * SSDT tool * Existing SSIS project and Package |

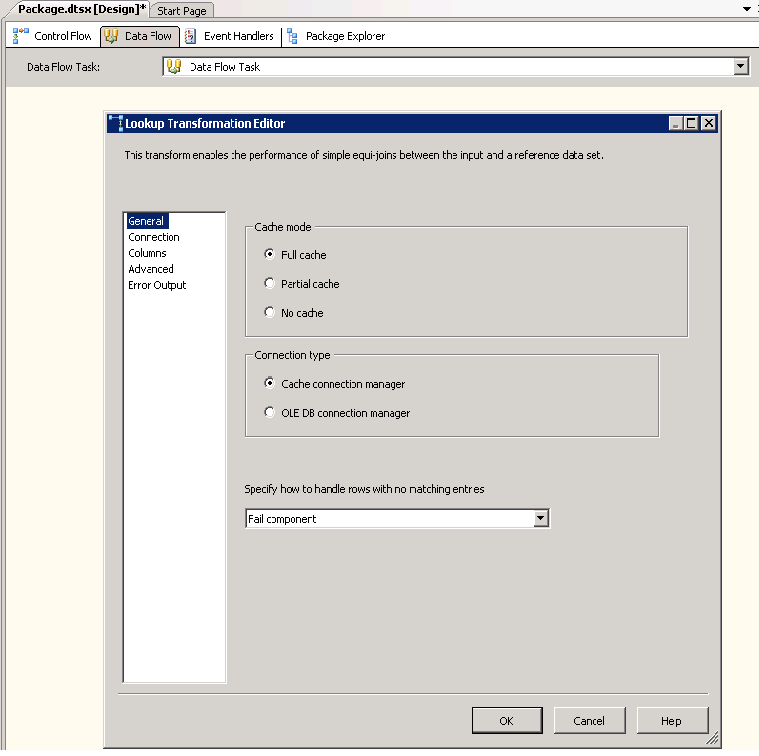
1)In the Data flow Task after selecting the source table, drag and drop Data conversion transformation and double click to view the editor.

* Convert the data type of the fields like emp\_name, emp\_code, address and salary compatible

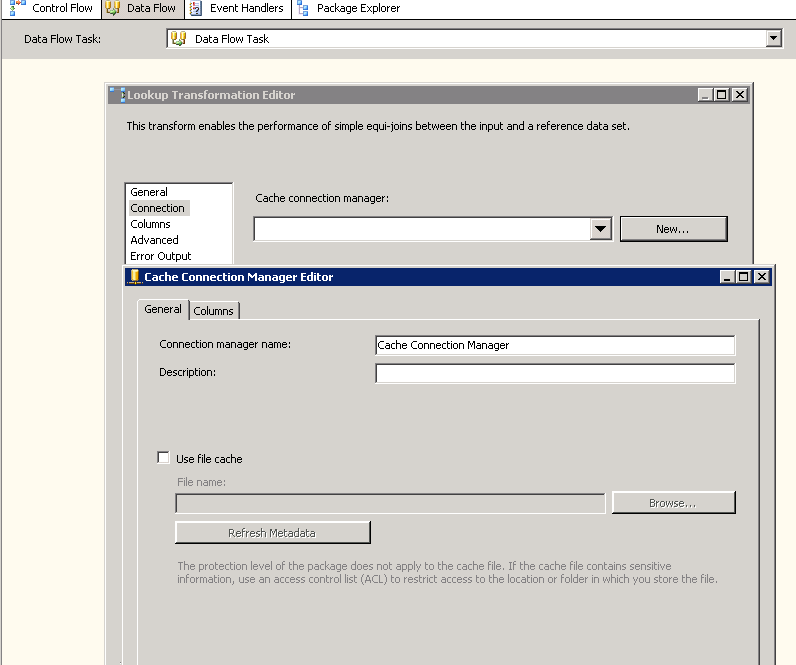
with the target table.

2)Drag and drop the Look up transformation and then double click to view the editor.

3) In the General tab, you can select the Cache Mode and the Connection Type:



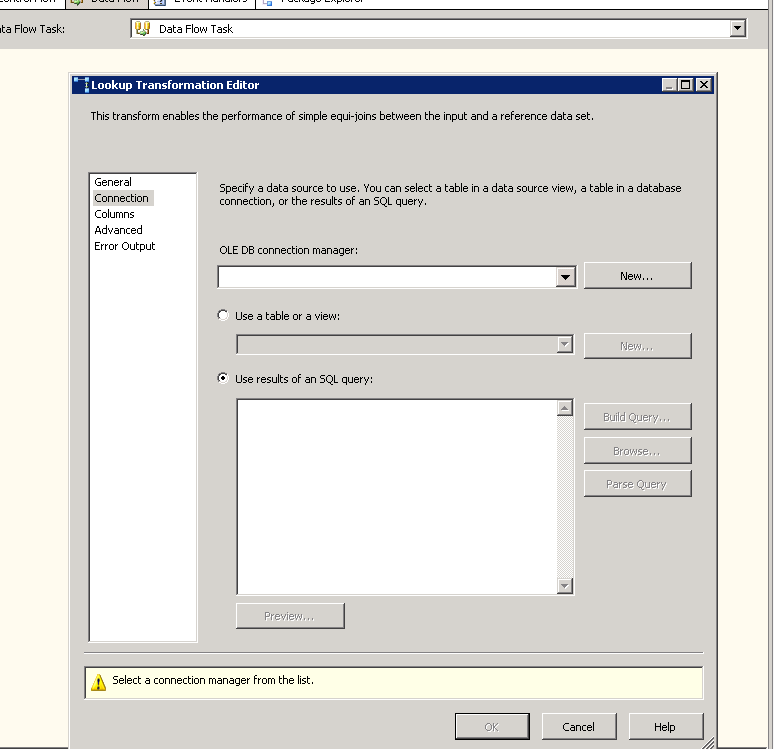
4)After selecting Cache Connection Manager in the Connection Type,You can select the Cache Connection Manager which you have created in the Cache Transformation created before the Lookup Transformation.



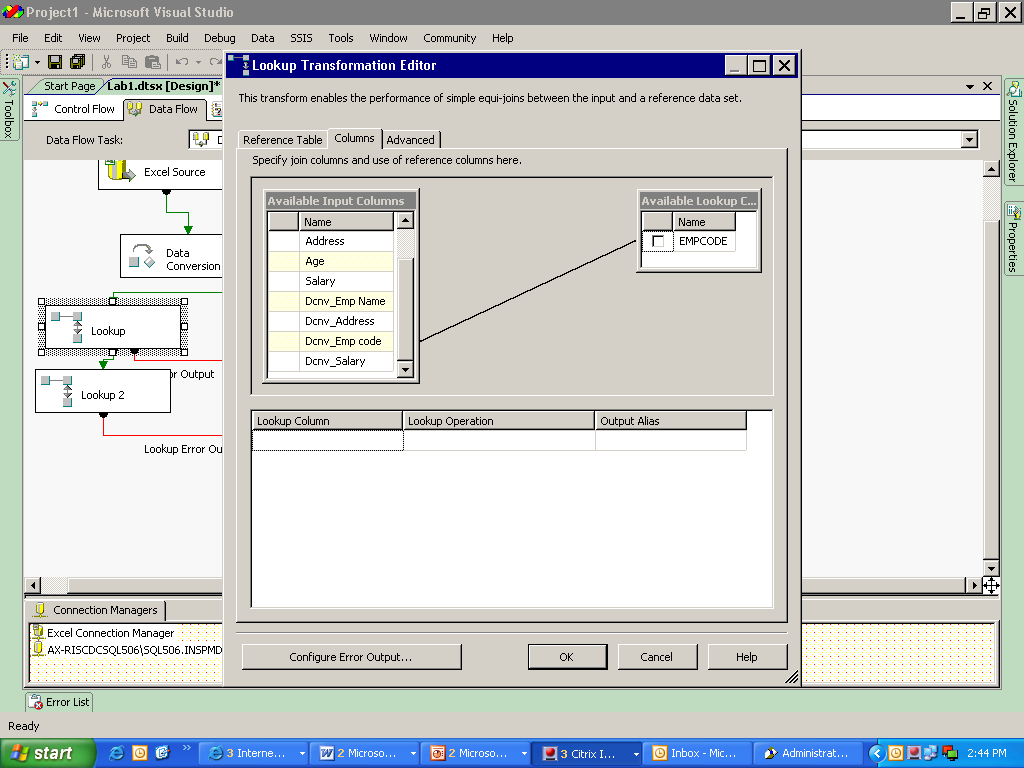
5) After selecting OLE DB Connection Manager in the Connection Type,In the editor you can either use a table or use SQL statement to look up the data.

6)Select the connection manager and use a SQL statement to extract empcode.

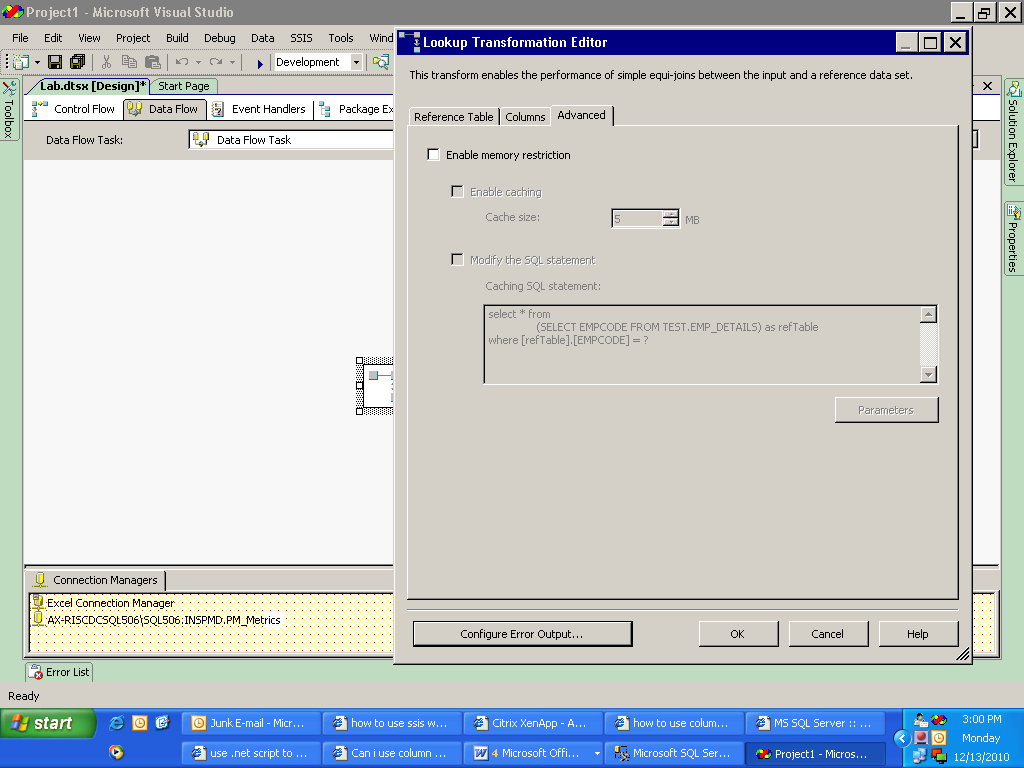
**(Select empcode from table name)**



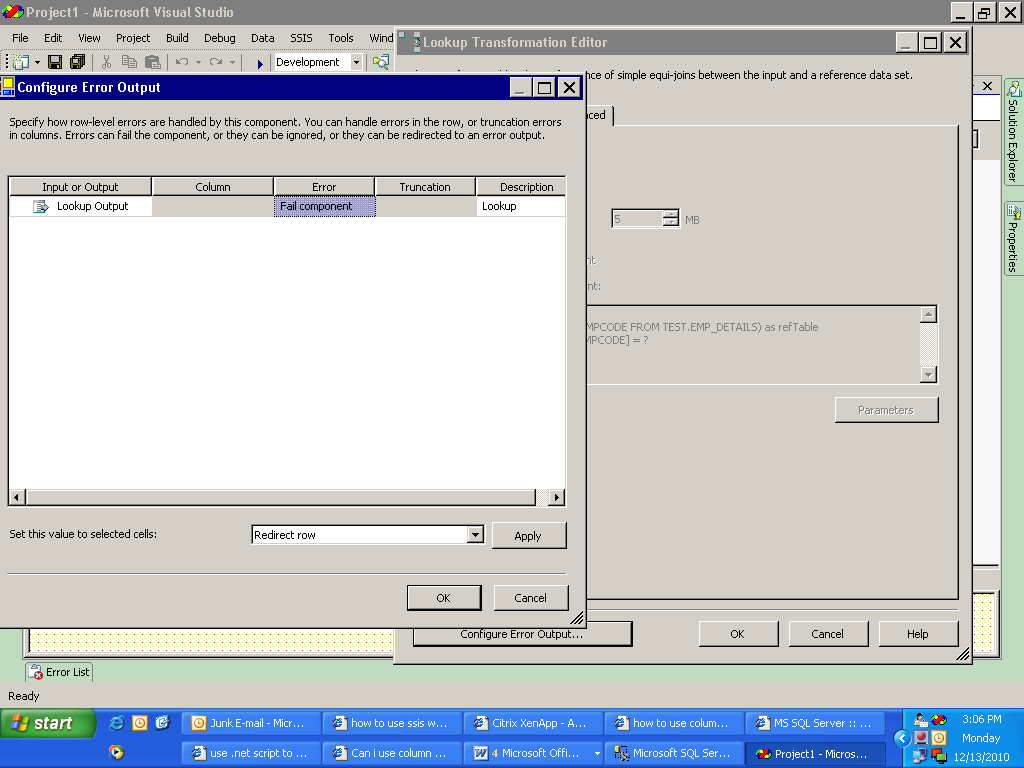
7) On the column tab join the empcode to find a match for the same.



8) If the lookup finds any new data then it redirects to the target table.

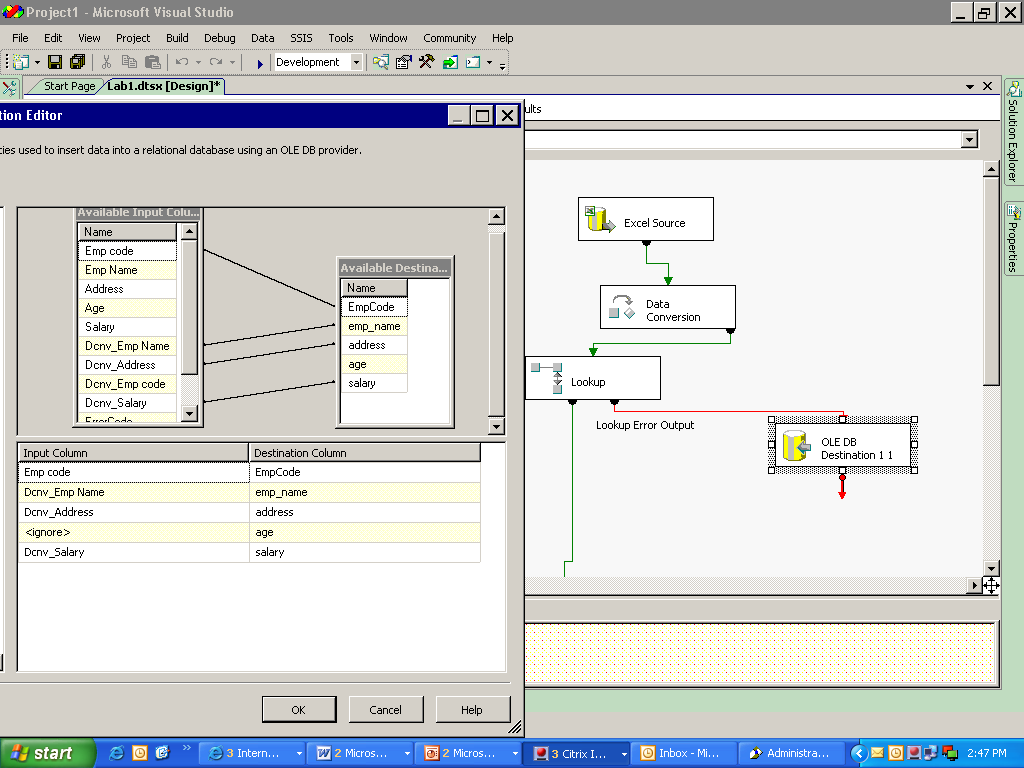


Use this tab to redirect the row to the target table



Change to “Redirect Row” and click “Apply” then click OK

9)Use one OLE DB destination to redirect the new row to target table.In the OLE DB destination set the connection properties and map the columns.



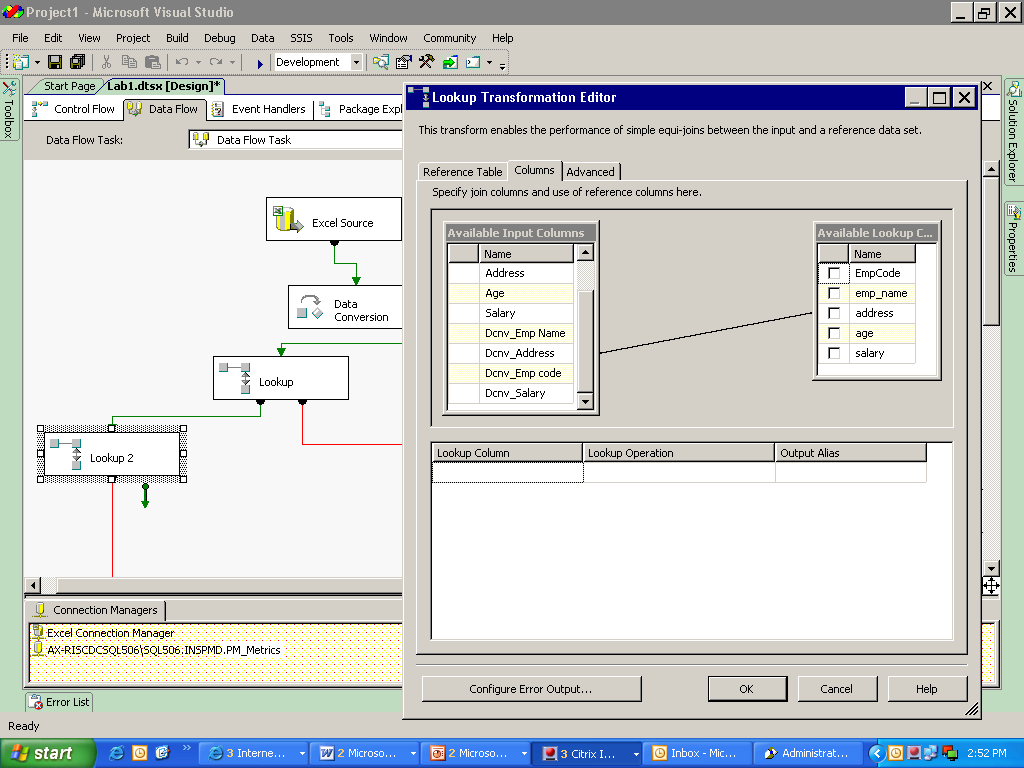
Redirected row.

10)Use another look up to check if there is any change in address of the data that is already present in the target table.

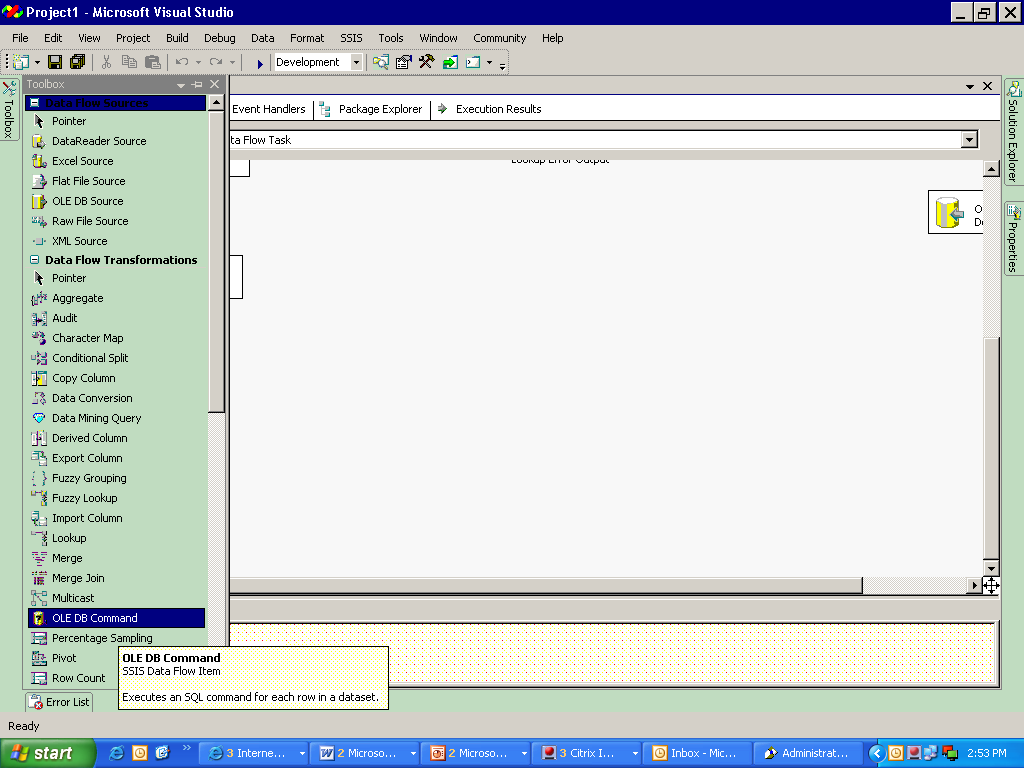
* To use the result of an SQL query use the following query:

SELECT \* FROM TEST.EMP\_DETAILS

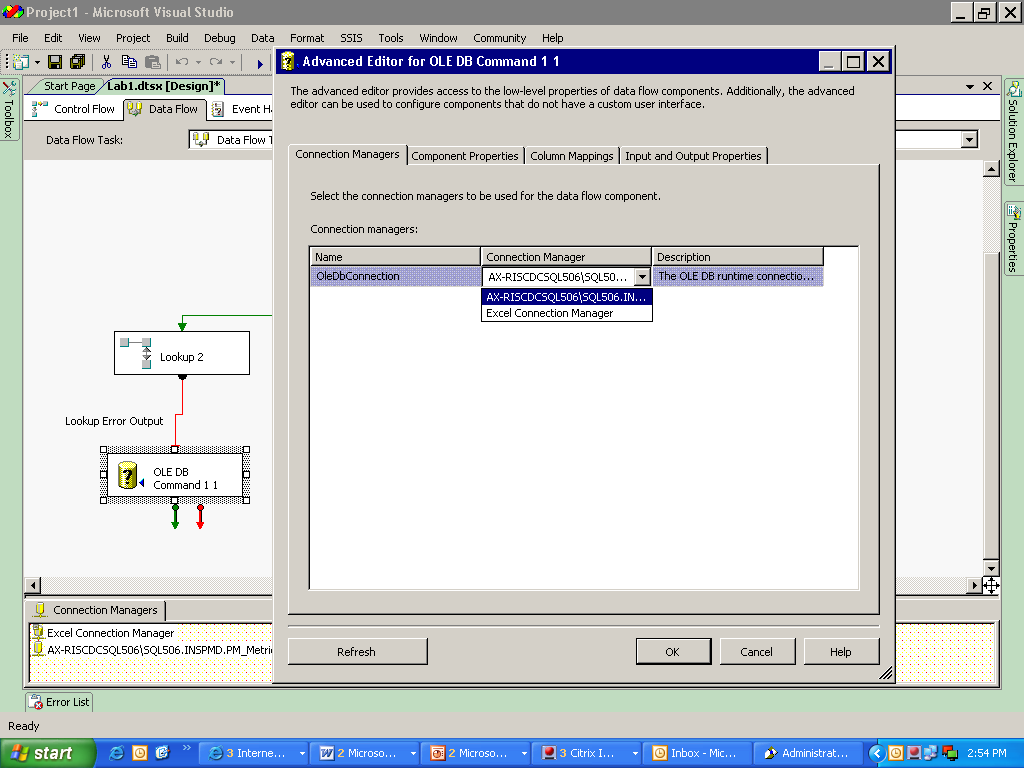
11)In the Columns tab match the address columns to check.



12) Now select OLE DB command from the toolbox.



13)Set the connection as available:



This connection is shown because it was already present

* In the component properties tab select the SqlCommand and write the query as:

**UPDATE TEST.EMP\_DETAILS**

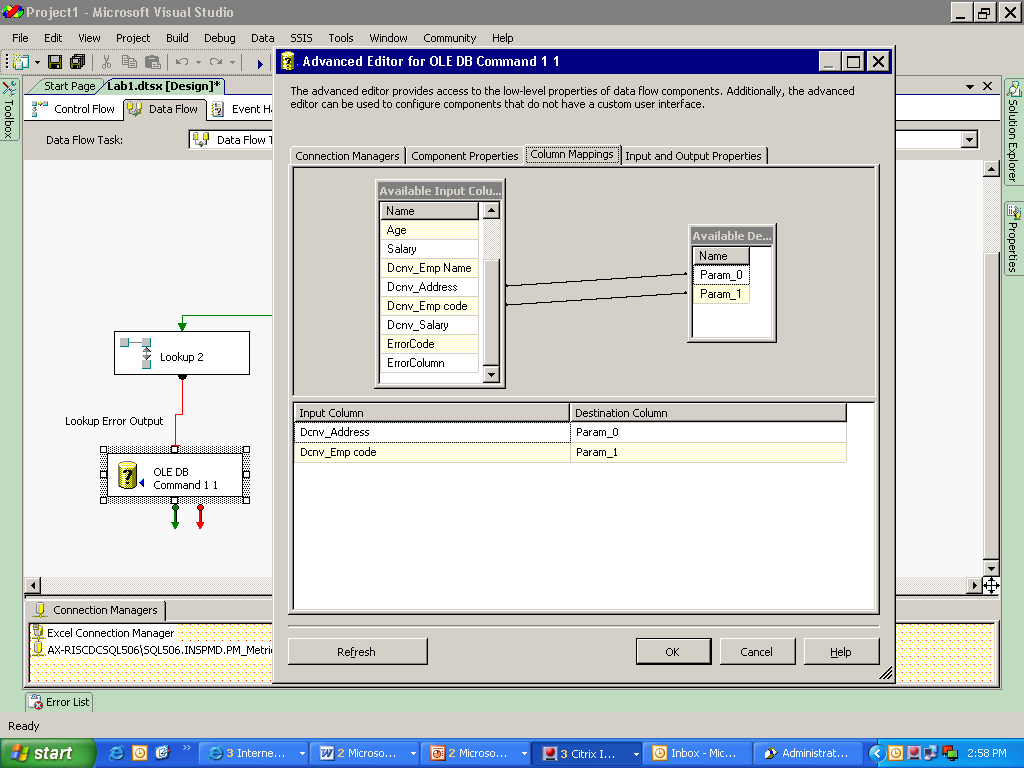
**SET ADDRESS=?**

**WHERE EMPCODE=?**

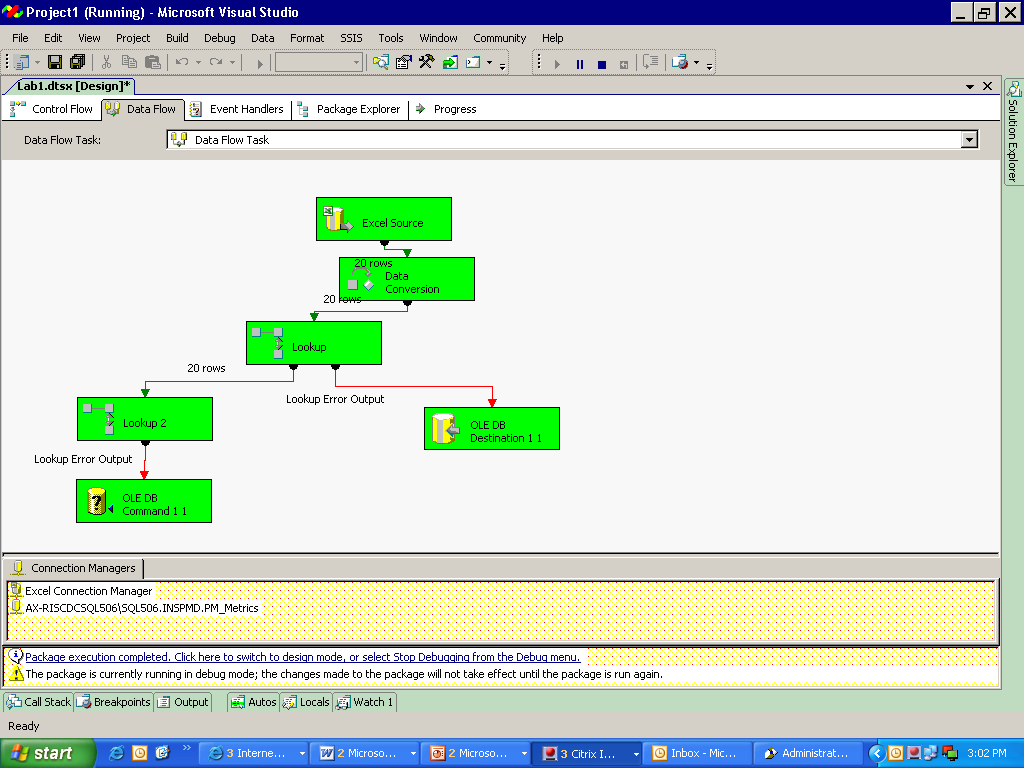
[The two question marks create parameters which are used to check the data (Old and updated).

The first question mark refer to first and second refer to the second paramet]

14)In the column Mappings tab map the columns with the parameters:



15)Now save and execute the Data flow.



**Original data:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Emp\_code** | **emp\_name** | **Address** | **Age** | **salary** |
| 101 | a | xyz | 23 | 20000 |
| 102 | b | abc | 43 | 56000 |
| 103 | c | pqr | 34 | 40000 |
| 104 | d | asd | 56 | 59000 |
| 105 | e | abc | 33 | 39000 |
| 106 | f | def | 27 | 36254 |
| 107 | g | ghi | 47 | 49000 |
| 108 | h | jkl | 55 | 70000 |
| 109 | i | mno | 28 | 30000 |
| 110 | j | lop | 38 | 36000 |
| 111 | k | stu | 49 | 50000 |
| 112 | l | vwx | 29 | 31000 |
| 113 | m | yz | 30 | 32000 |
| 114 | n | qwe | 40 | 49875 |
| 115 | o | rtyu | 50 | 58000 |
| 116 | p | fnfd | 39 | 39785 |
| 117 | q | sfh | 25 | 26000 |
| 118 | r | jkg | 26 | 27154 |
| 119 | s | mnb | 36 | 35647 |
| 120 | t | afgh | 47 | 46987 |

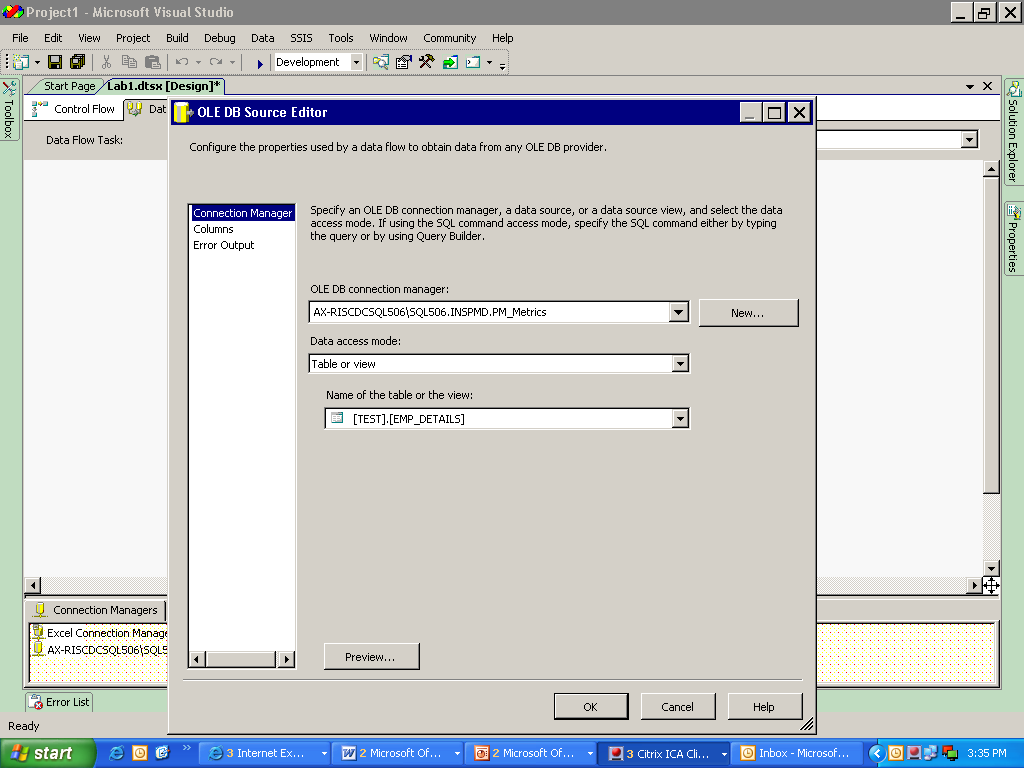
**Final data:**

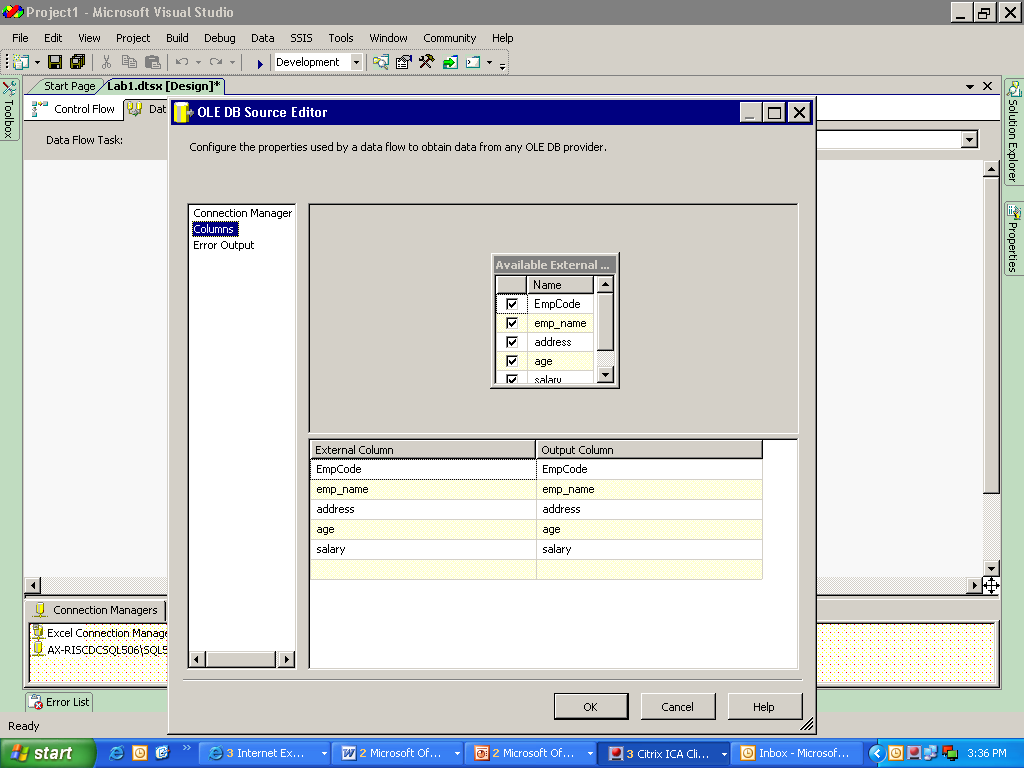
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Emp\_code** | **emp\_name** | **Address** | **Age** | **salary** |
| 101 | a | **klasfnx** | 23 | 20000 |
| 102 | b | abc | 43 | 56000 |
| 103 | c | pqr | 34 | 40000 |
| 104 | d | asd | 56 | 59000 |
| 105 | e | abc | 33 | 39000 |
| 106 | f | **perghjsd** | 27 | 36254 |
| 107 | g | ghi | 47 | 49000 |
| 108 | h | jkl | 55 | 70000 |
| 109 | i | mno | 28 | 30000 |
| 110 | j | lop | 38 | 36000 |
| 111 | k | **jshfkag** | 49 | 50000 |
| 112 | l | vwx | 29 | 31000 |
| 113 | m | yz | 30 | 32000 |
| 114 | n | qwe | 40 | 49875 |
| 115 | o | rtyu | 50 | 58000 |
| 116 | p | fnfd | 39 | 39785 |
| 117 | q | **sdfjfgosf** | 25 | 26000 |
| 118 | r | jkg | 26 | 27154 |
| 119 | s | mnb | 36 | 35647 |
| 120 | t | afgh | 47 | 46987 |

Lab 6- **Copy Column Transformation**

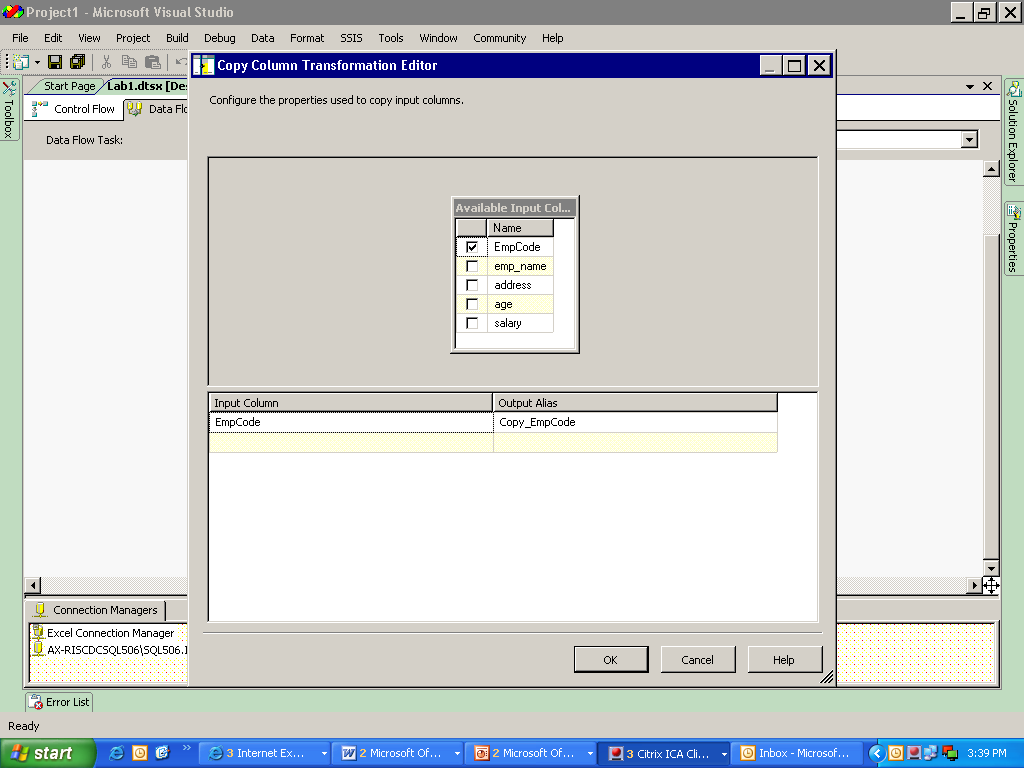
|  |  |
| --- | --- |
| **Objective** | To create a copy of a column. |
| **Lab Setup** | * SSDT tool * Existing SSIS project and Package |

1)Select OLE DB source and double click to view the editor, set the connection manager and select the required columns.



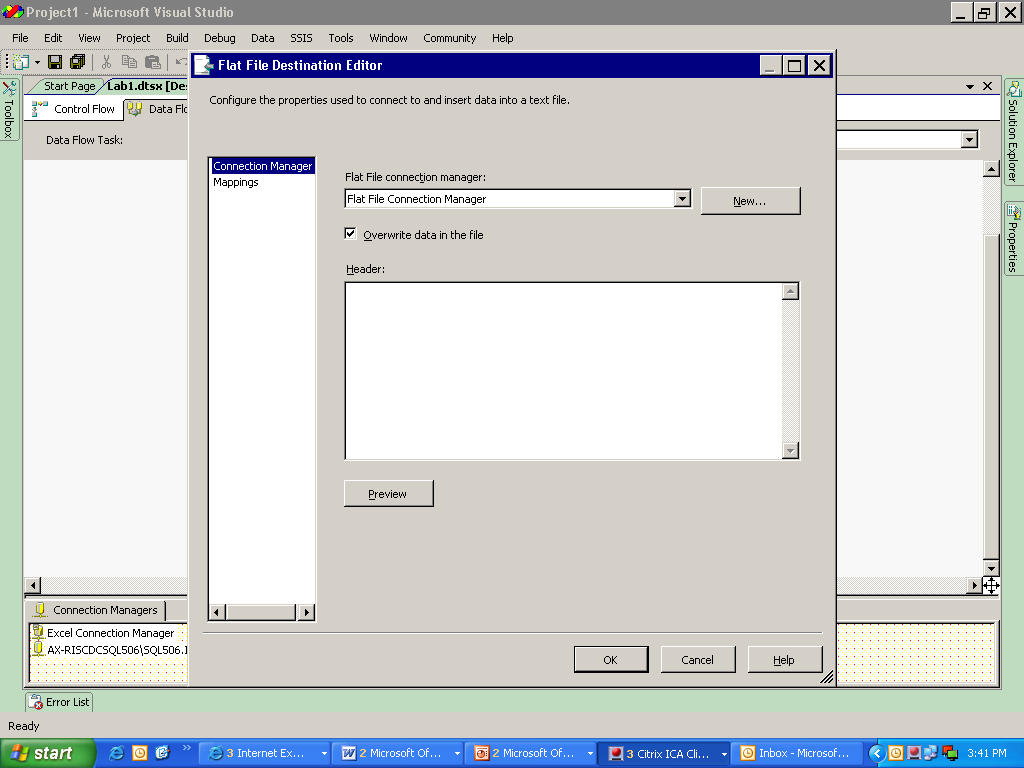


2) Select copy column transformation and double click to view the editor.



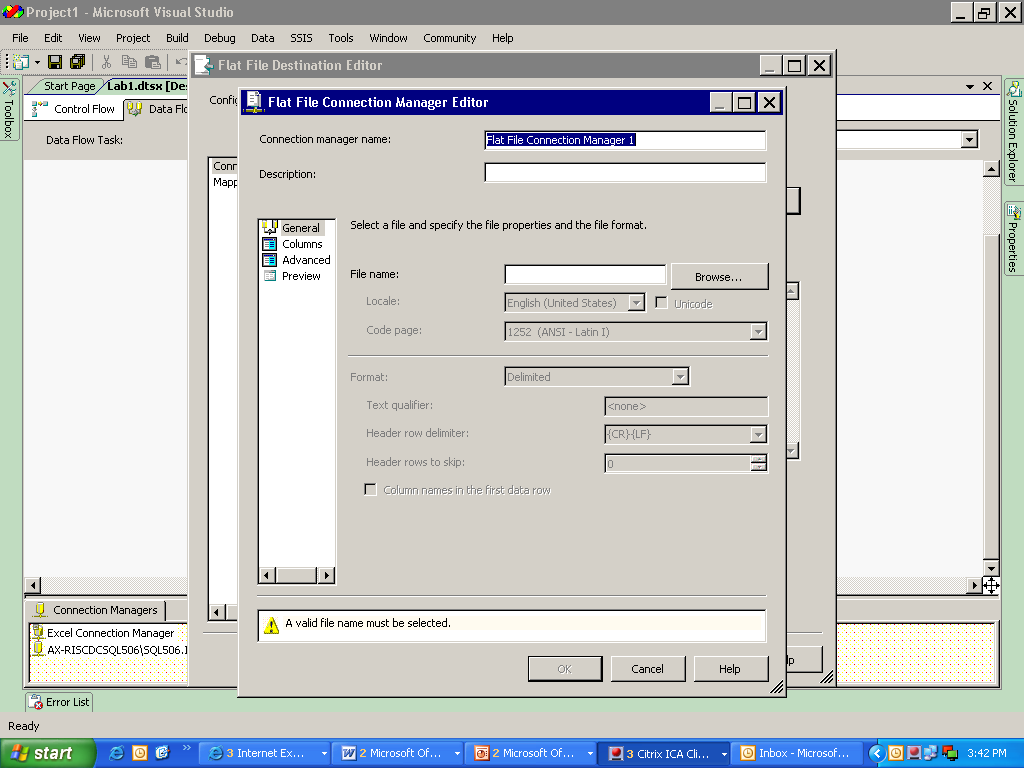
3)Select the column for which you need a copy and click OK.

4)Select a Flat file destination and double click to view editor.



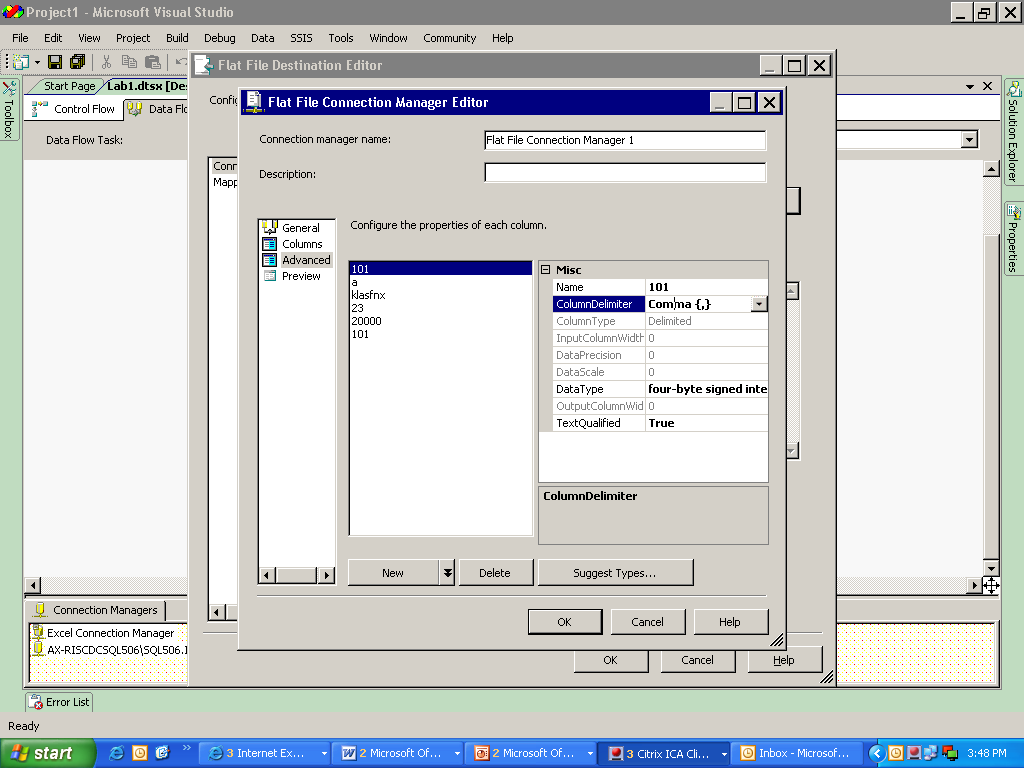
5)Click on NEW to get the format.

6)Select Delimited.

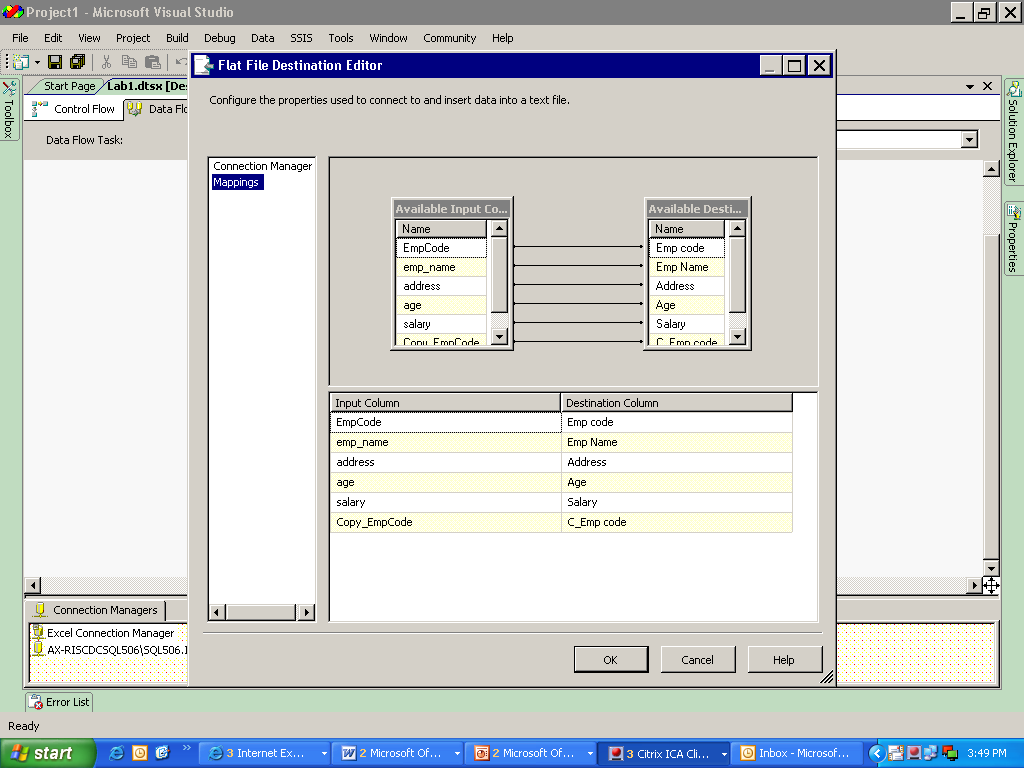


7)Browse for a file where you want to load the data.

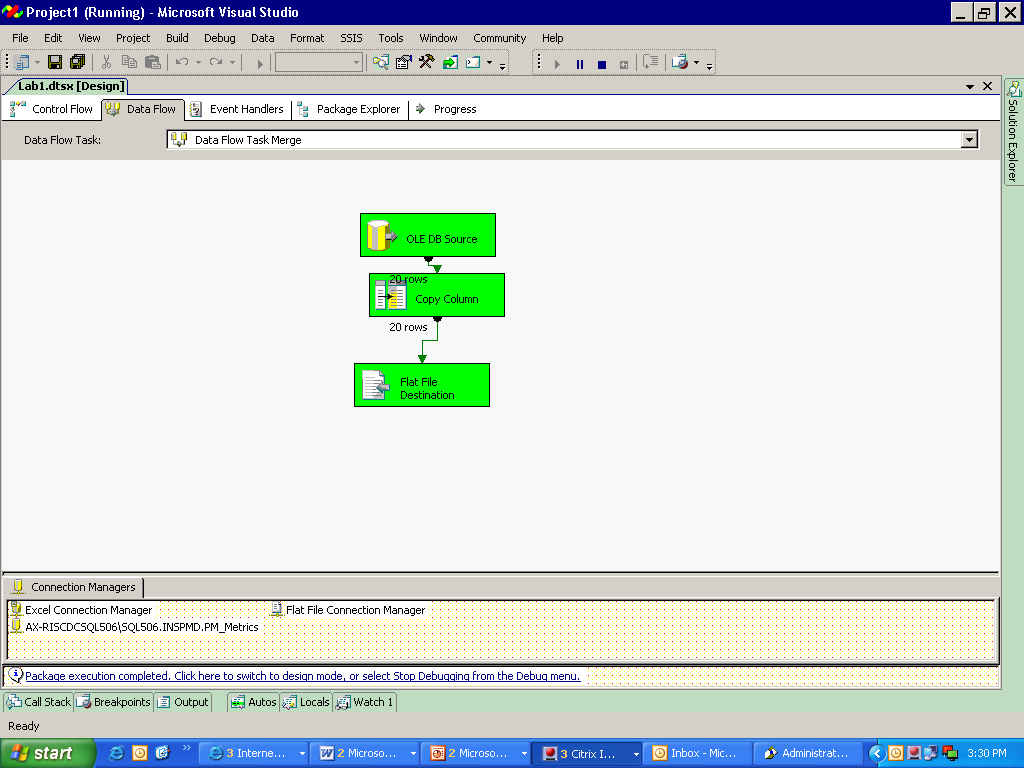
* Usually the column delimiter is Comma (,) which can be changed in Columns or advanced option.



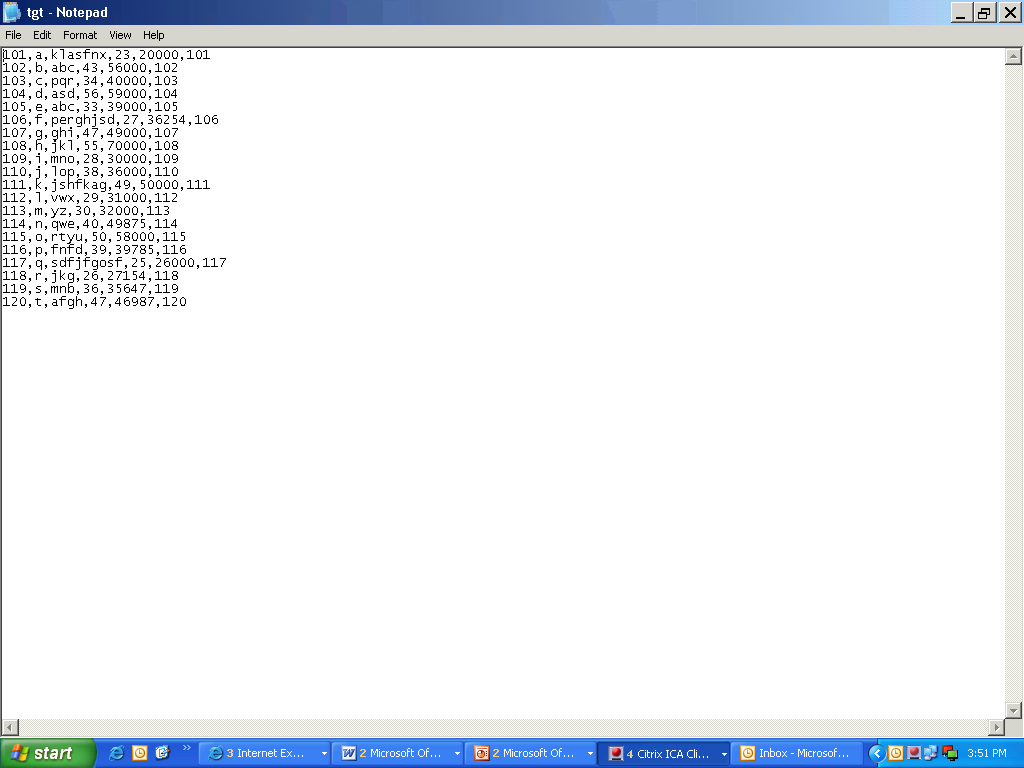
8)Select mapping and map the corresponding columns.



9)Save and execute the data flow.



**Output:**

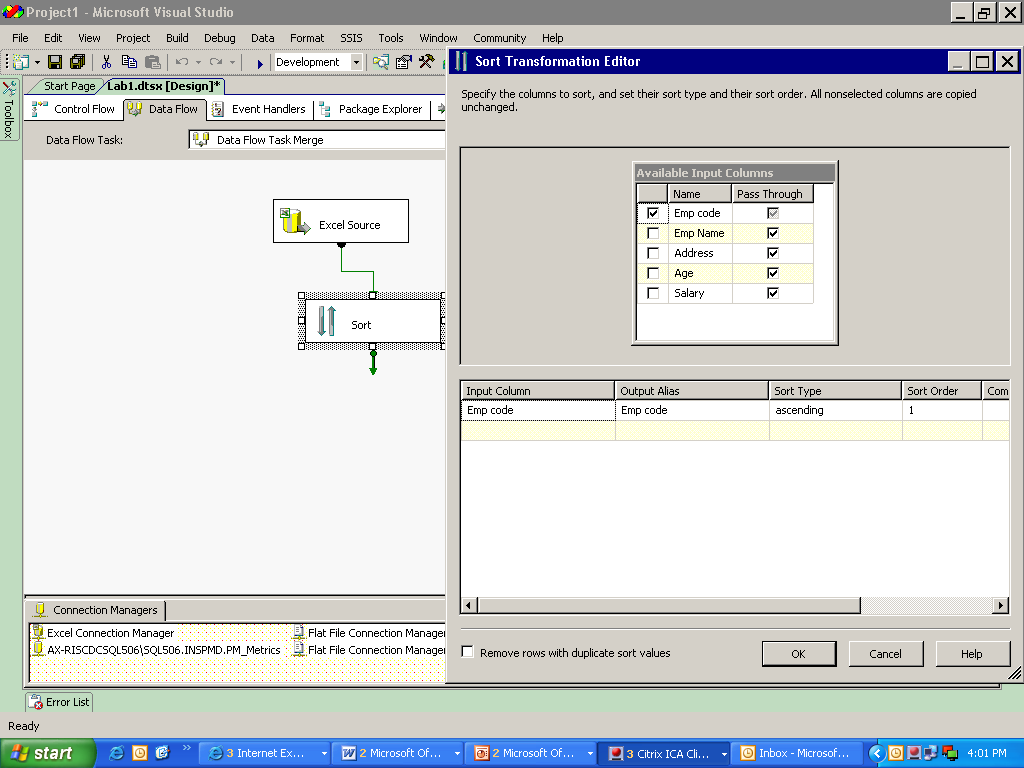


Lab 7- Merge Transformation

|  |  |
| --- | --- |
| **Objective** | To merge data from two different sources into one target. |
| **Lab Setup** | * SSDT tool * Existing SSIS project and Package |

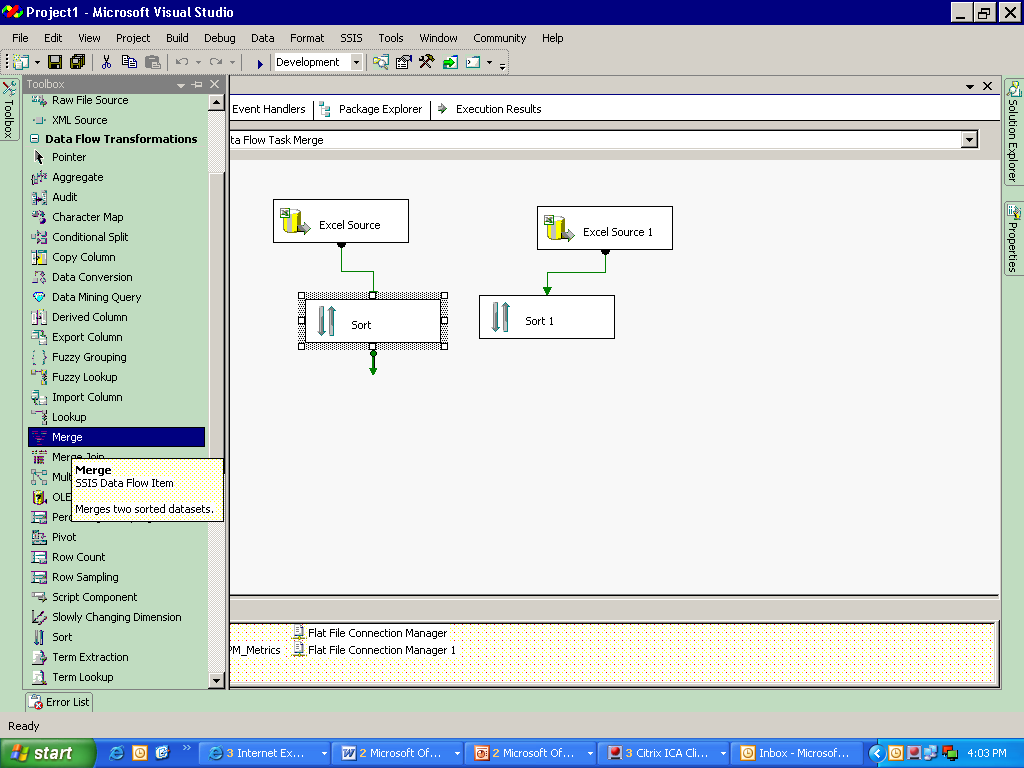
1)Select two excel file as source and edit the connection manager.

2)Select two sort transformations and edit the settings.

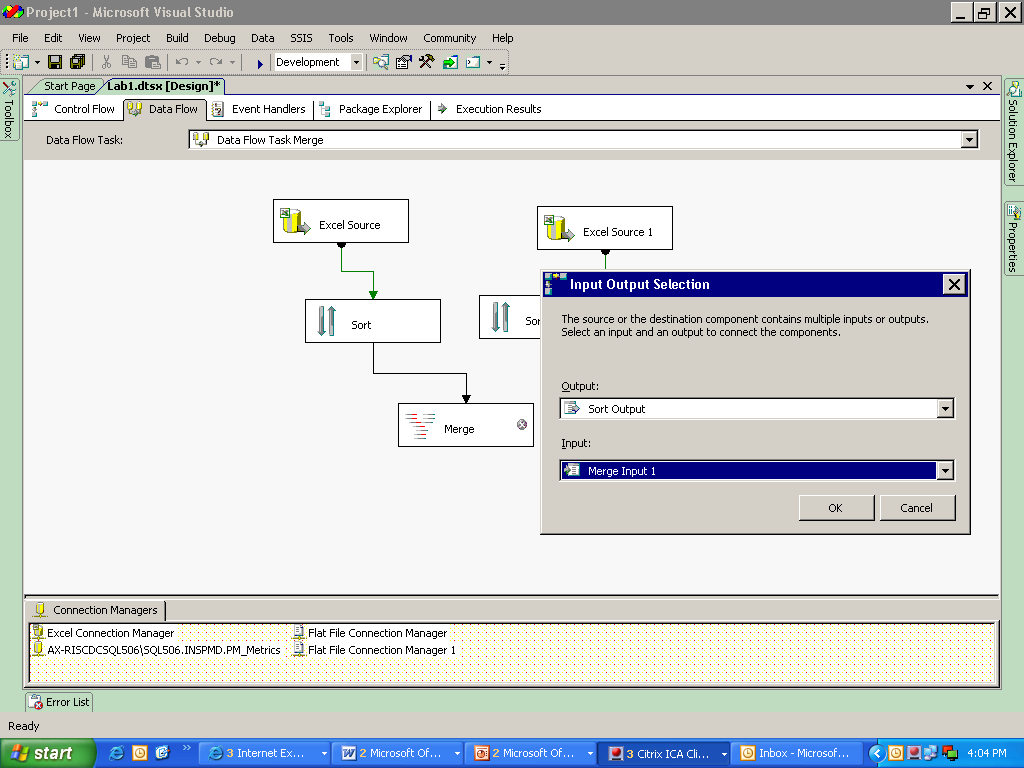


3)Sort both the source on the same column.

4)Then select merge transformation and join the sorted source to it.



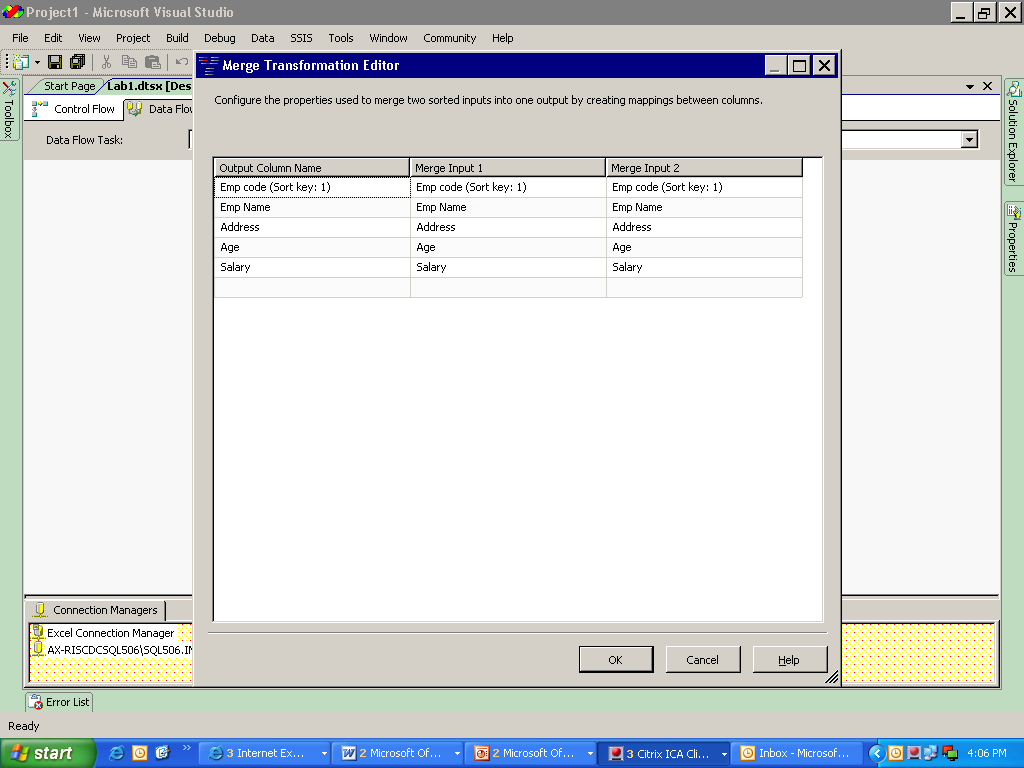
5)After joining the first SORT to the merge transformation, the window opens as:



6)Select Sort Output and Merge Input 1

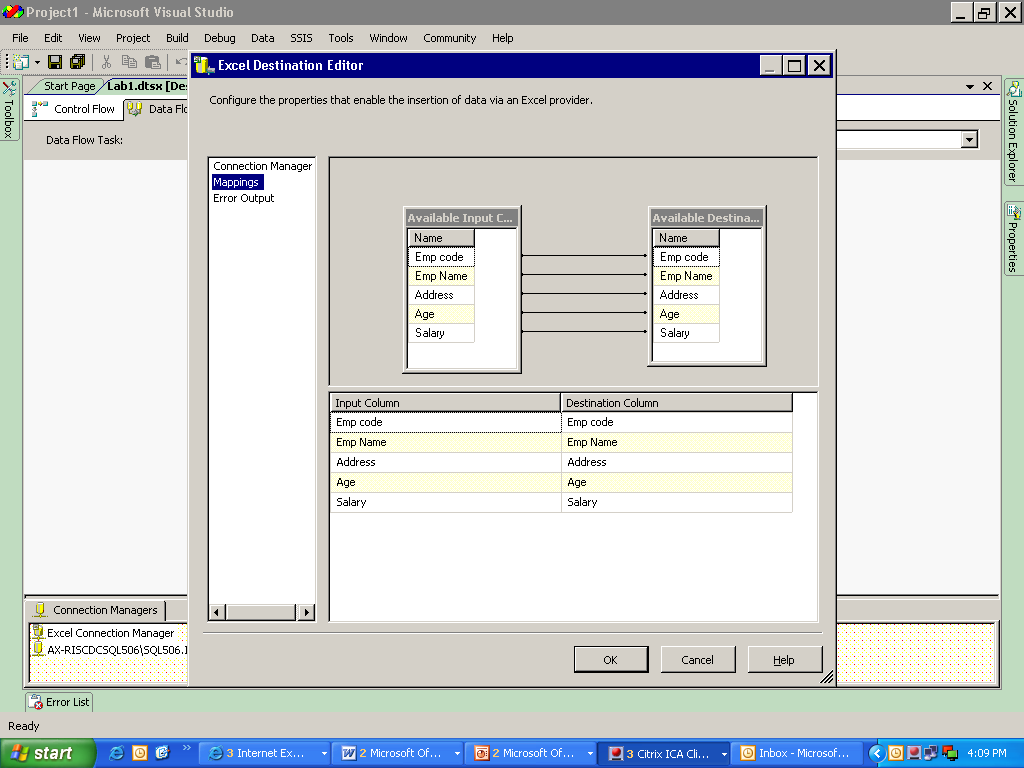
7)And then join the second sort to the Merge transformation.

8) Double click the merge transformation to view the editor.

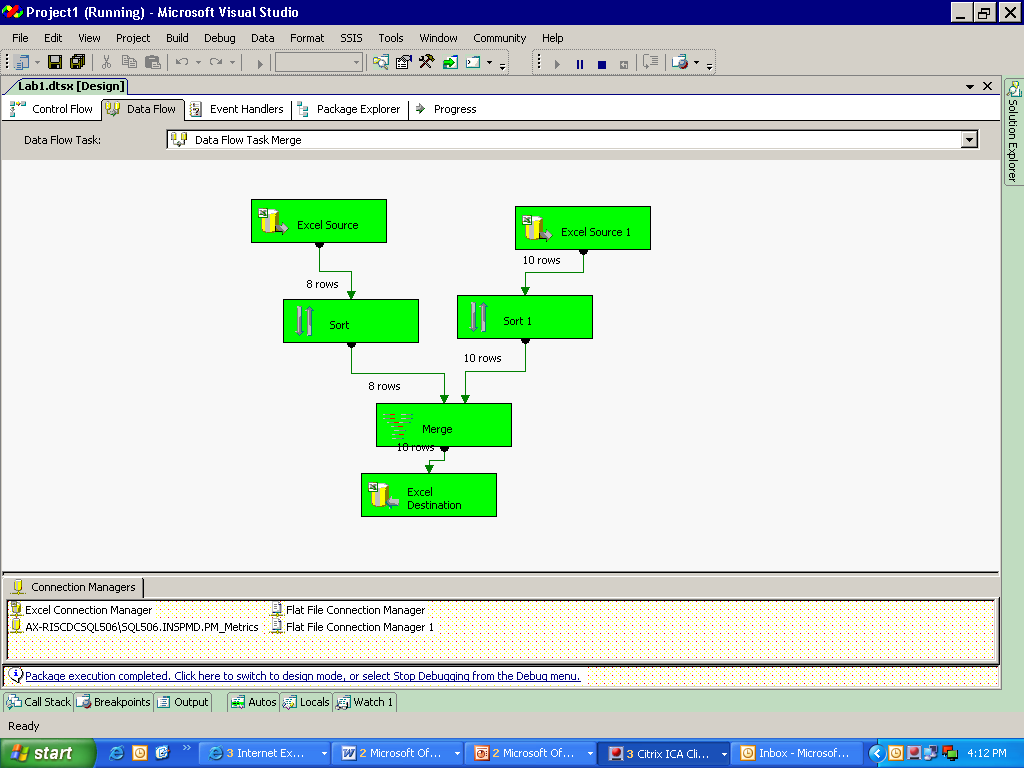


9)Select a destination file (here we have taken excel as target)

* Set the connection manager and map the columns in the Columns tab.



10)Save and execute the data flow task.



**Data 1:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Emp code** | **Emp Name** | **Address** | **Age** | **Salary** |
| 112 | l | vwx | 29 | 31000 |
| 113 | m | yz | 30 | 32000 |
| 114 | n | qwe | 40 | 49875 |
| 101 | a | klasfnx | 23 | 20000 |
| 102 | b | abc | 43 | 56000 |
| 103 | c | pqr | 34 | 40000 |
| 104 | d | asd | 56 | 59000 |
| 105 | e | abc | 33 | 39000 |

**Data 2:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Emp code** | **Emp Name** | **Address** | **Age** | **Salary** |
| 101 | a | klasfnx | 23 | 20000 |
| 102 | b | abc | 43 | 56000 |
| 103 | c | pqr | 34 | 40000 |
| 104 | d | asd | 56 | 59000 |
| 105 | e | abc | 33 | 39000 |
| 119 | s | mnb | 36 | 35647 |
| 120 | t | afgh | 47 | 46987 |
| 112 | l | vwx | 29 | 31000 |
| 113 | m | yz | 30 | 32000 |
| 114 | n | qwe | 40 | 49875 |

**Output data:**

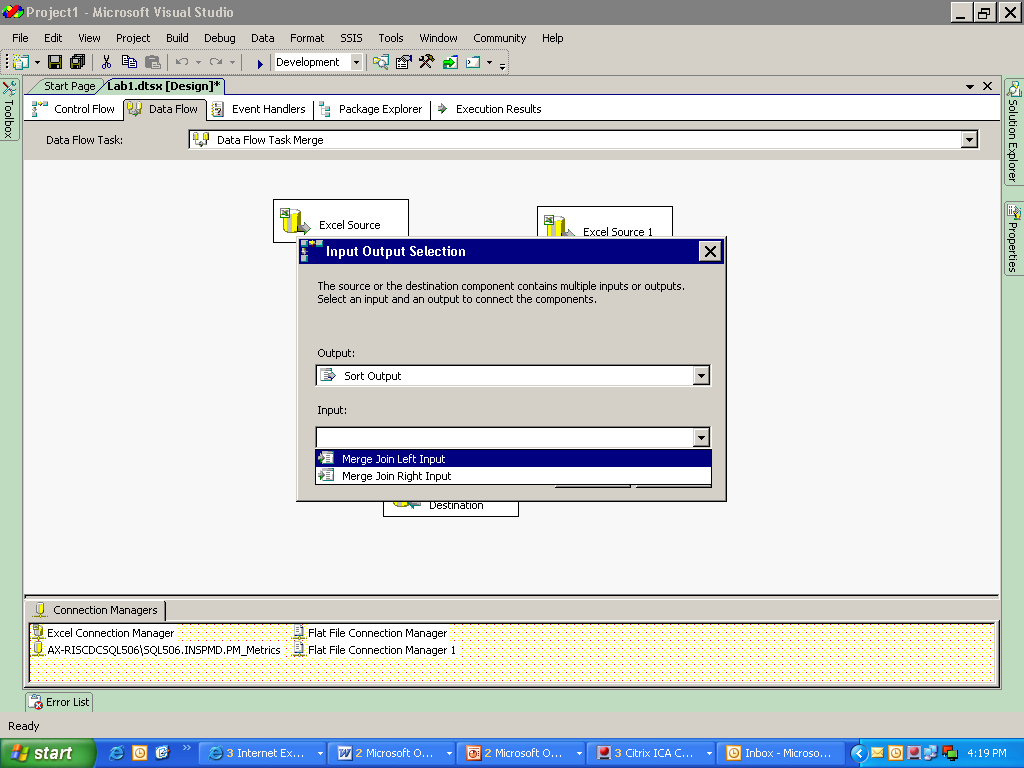
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Emp code** | **Emp Name** | **Address** | **Age** | **Salary** |
| 101 | a | klasfnx | 23 | 20000 |
| 101 | a | klasfnx | 23 | 20000 |
| 102 | b | abc | 43 | 56000 |
| 102 | b | abc | 43 | 56000 |
| 103 | c | pqr | 34 | 40000 |
| 103 | c | pqr | 34 | 40000 |
| 104 | d | asd | 56 | 59000 |
| 104 | d | asd | 56 | 59000 |
| 105 | e | abc | 33 | 39000 |
| 105 | e | abc | 33 | 39000 |
| 112 | l | vwx | 29 | 31000 |
| 112 | l | vwx | 29 | 31000 |
| 113 | m | yz | 30 | 32000 |
| 113 | m | yz | 30 | 32000 |
| 114 | n | qwe | 40 | 49875 |
| 114 | n | qwe | 40 | 49875 |
| 119 | s | mnb | 36 | 35647 |
| 120 | t | afgh | 47 | 46987 |

Lab 8- Merge join Transformation

|  |  |
| --- | --- |
| **Objective** | To merge data from two data sources and load into a target table using join condition. |
| **Lab Setup** | * SSDT tool * Existing SSIS project and Package |

1)Repeat the steps from selecting the source till sorting of data and then drag and drop Merge Join Transformation.

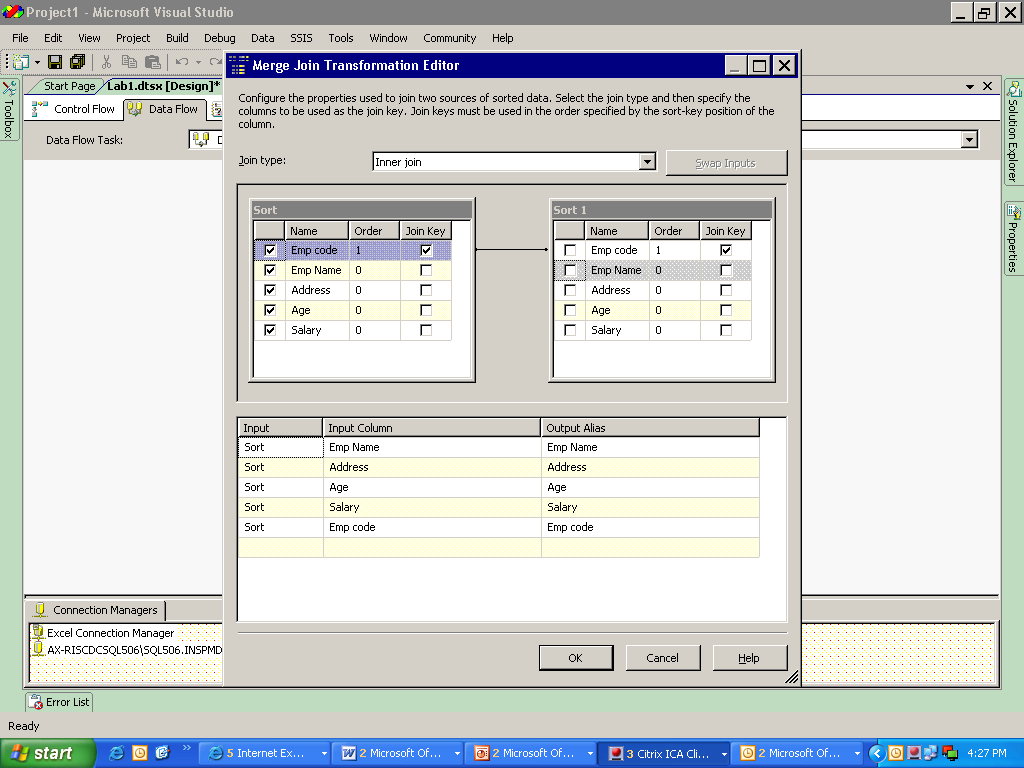
* Join the sort output to the transformation.
* The window opens as:



* Select the valid input type.
* Connect the second input to the transformation.

2)Double click merge join transformation to view the editor.

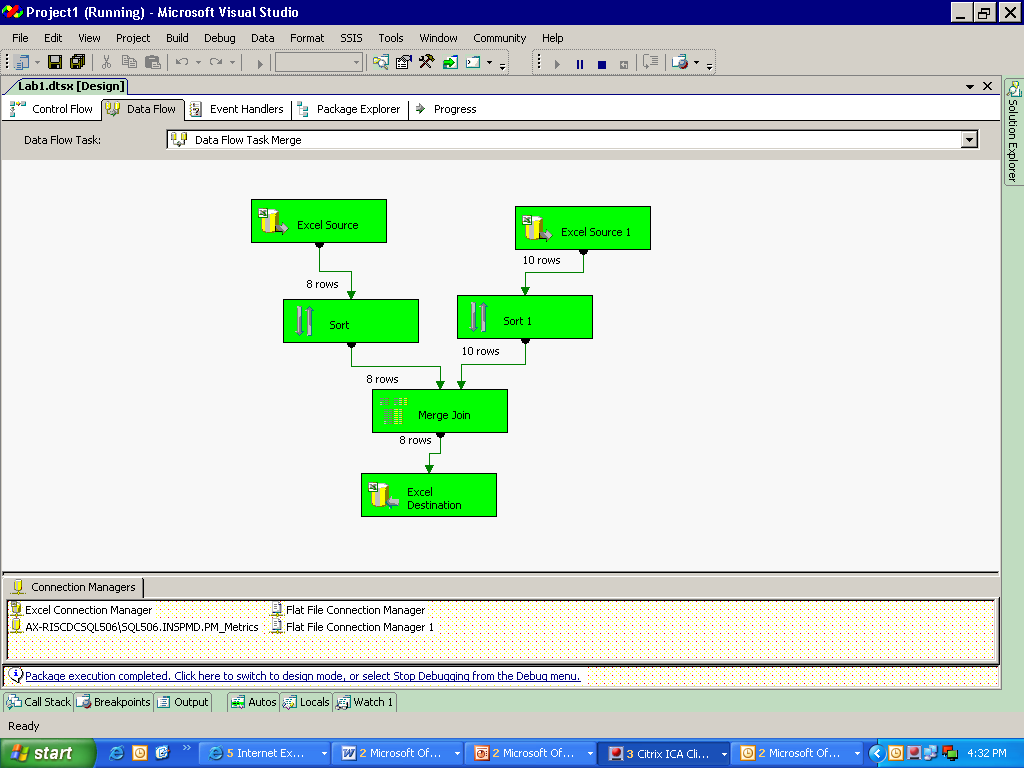
3)Select the required join type and the columns.



4)Connect the transformation to destination.

5)Edit the connection manager with valid connections and map the columns.

6)Save and execute the data flow.



**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Emp Name** | **Address** | **Age** | **Salary** | **Emp code** |
| a | klasfnx | 23 | 20000 | 101 |
| b | abc | 43 | 56000 | 102 |
| c | pqr | 34 | 40000 | 103 |
| d | asd | 56 | 59000 | 104 |
| e | abc | 33 | 39000 | 105 |
| l | vwx | 29 | 31000 | 112 |
| m | yz | 30 | 32000 | 113 |
| n | qwe | 40 | 49875 | 114 |

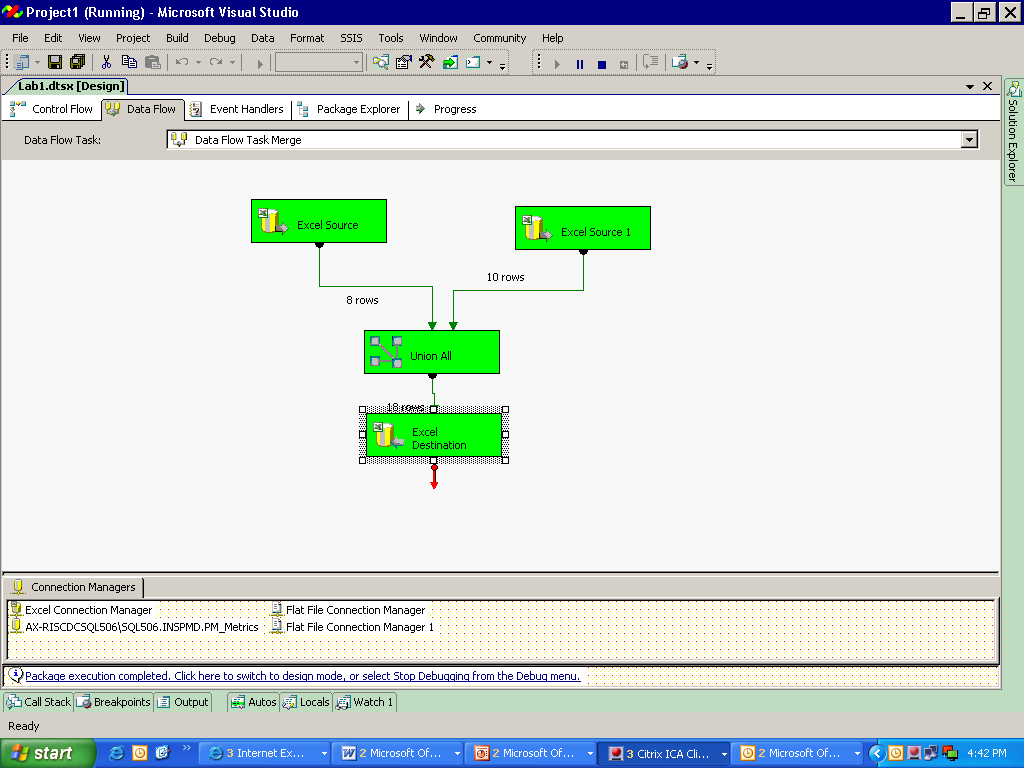
Lab 9- Union All Transformation

|  |  |
| --- | --- |
| **Objective** | To merge two data sources into one target table. |
| **Lab Setup** | * SSDT tool * Existing SSIS project and Package |

1)Select union all from the toolbox and connect it to the sources.

2)Select destination and join the union transformation to it. Edit the connection manager mapping.

3)Save and execute the data flow.



**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Emp code** | **Emp Name** | **Address** | **Age** | **Salary** |
| 112 | l | vwx | 29 | 31000 |
| 113 | m | yz | 30 | 32000 |
| 114 | n | qwe | 40 | 49875 |
| 101 | a | klasfnx | 23 | 20000 |
| 102 | b | abc | 43 | 56000 |
| 103 | c | pqr | 34 | 40000 |
| 104 | d | asd | 56 | 59000 |
| 105 | e | abc | 33 | 39000 |
| 101 | a | klasfnx | 23 | 20000 |
| 102 | b | abc | 43 | 56000 |
| 103 | c | pqr | 34 | 40000 |
| 104 | d | asd | 56 | 59000 |
| 105 | e | abc | 33 | 39000 |
| 119 | s | mnb | 36 | 35647 |
| 120 | t | afgh | 47 | 46987 |
| 112 | l | vwx | 29 | 31000 |
| 113 | m | yz | 30 | 32000 |
| 114 | n | qwe | 40 | 49875 |

**Notes:**

The Merge Join transformation provides an output that is generated by joining two sorted datasets using a FULL, LEFT, or INNER join.

The Merge transformation combines two sorted datasets into a single dataset. The rows from each dataset are inserted into the output based on values in their key columns.

The Merge transformation is similar to the Union All transformations. Use the Union All transformation instead of the Merge transformation in the following situations:

* The transformation inputs are not sorted.
* The combined output does not need to be sorted.
* The transformation has more than two inputs.

Lab 10-**Slowly changing Dimension**

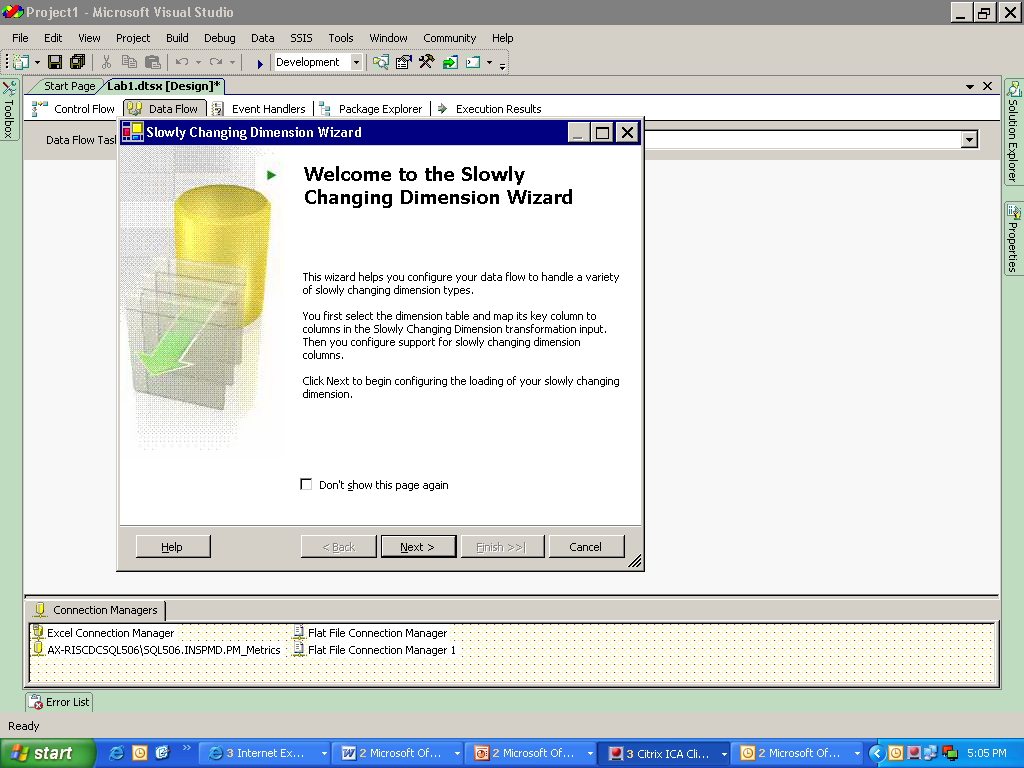
|  |  |
| --- | --- |
| **Objective** | To maintain the database in such a way that if any change occurs in the old data then the previous data is also shown along with the updated new data (Using Wizard) |
| **Lab Setup** | * SSDT tool * Existing SSIS project and Package |

This transformation is used to maintain historical data.

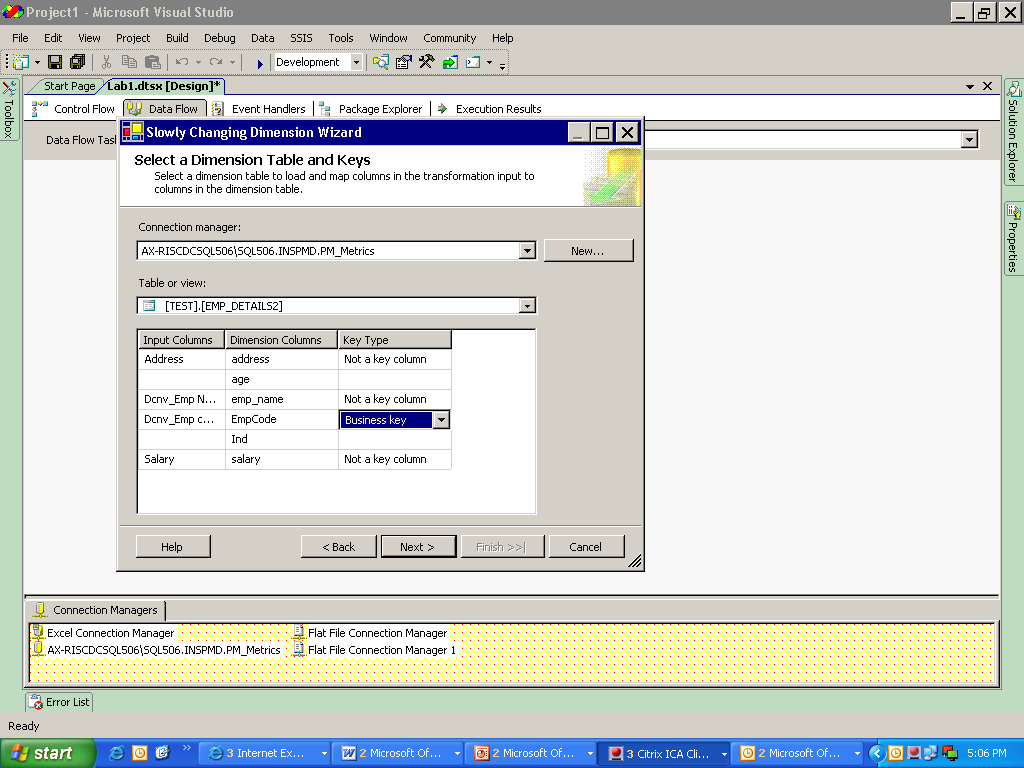
1)Select a source (here we have used OLE DB)

2)Select Slowly Changing dimension transformation, Connect the source to it.

* Double click the transformation to start the wizard.



* Click on NEXT .
* Select a target table and then select input columns and use emp code as business key.



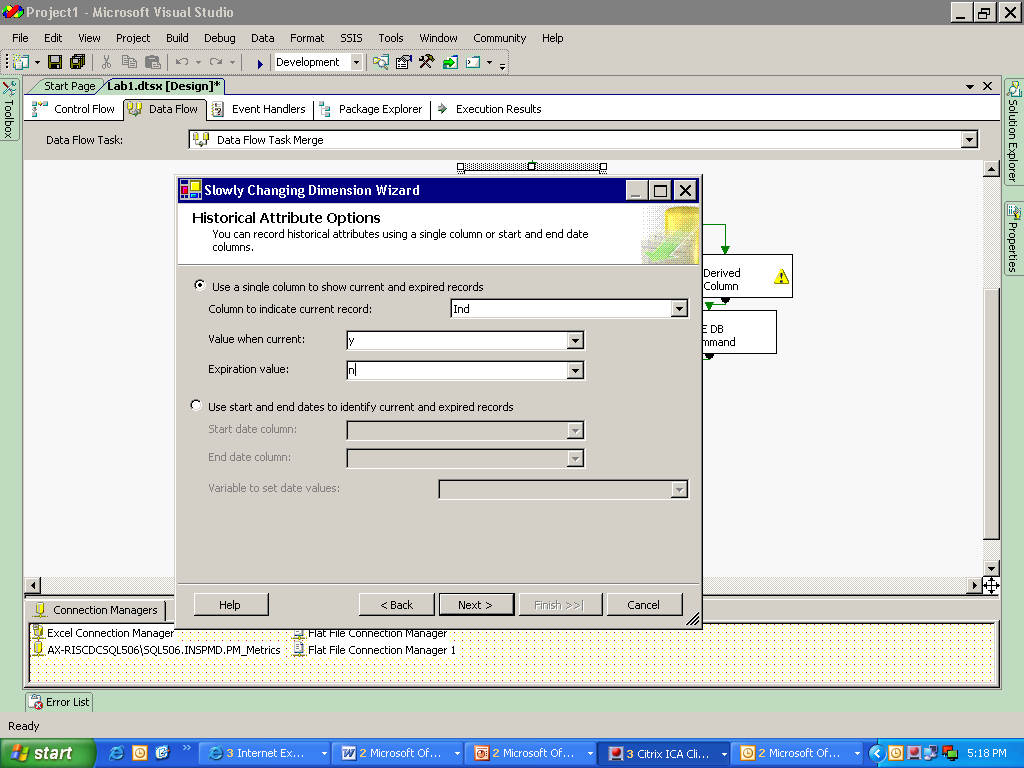
Target table

* Now click on NEXT, select the column where the data changes and change type to **historical attribute.**

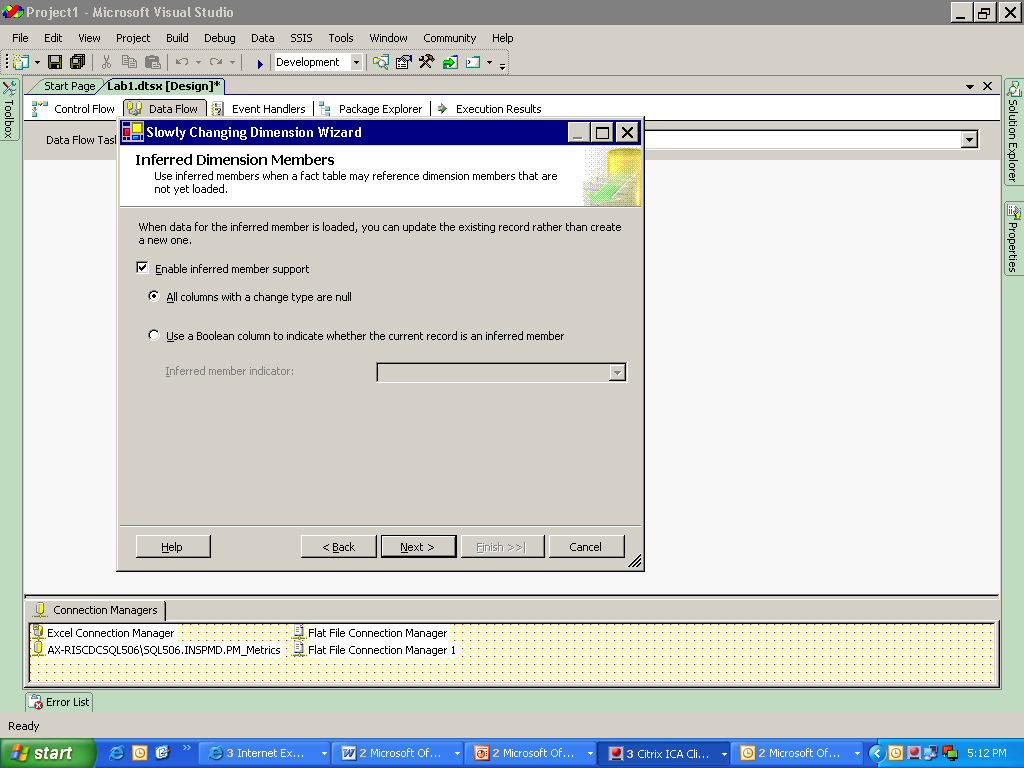


* Click NEXT.
* Use a column to differentiate new and old values.

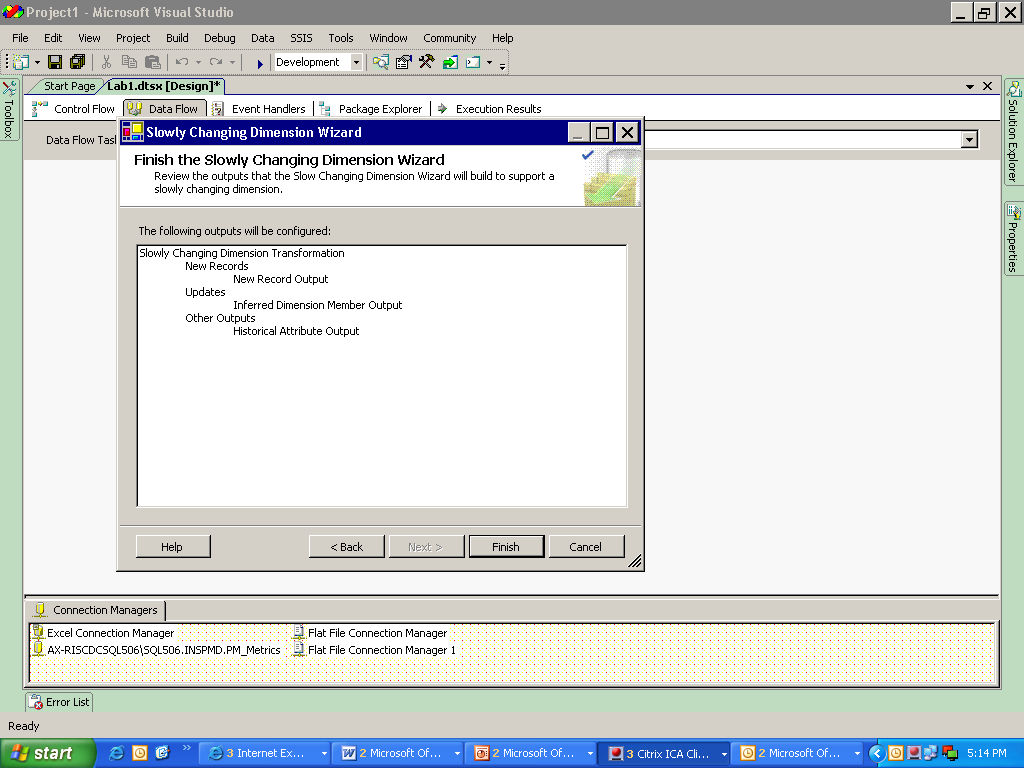
Set the required fields and then click NEXT.



Updating the values

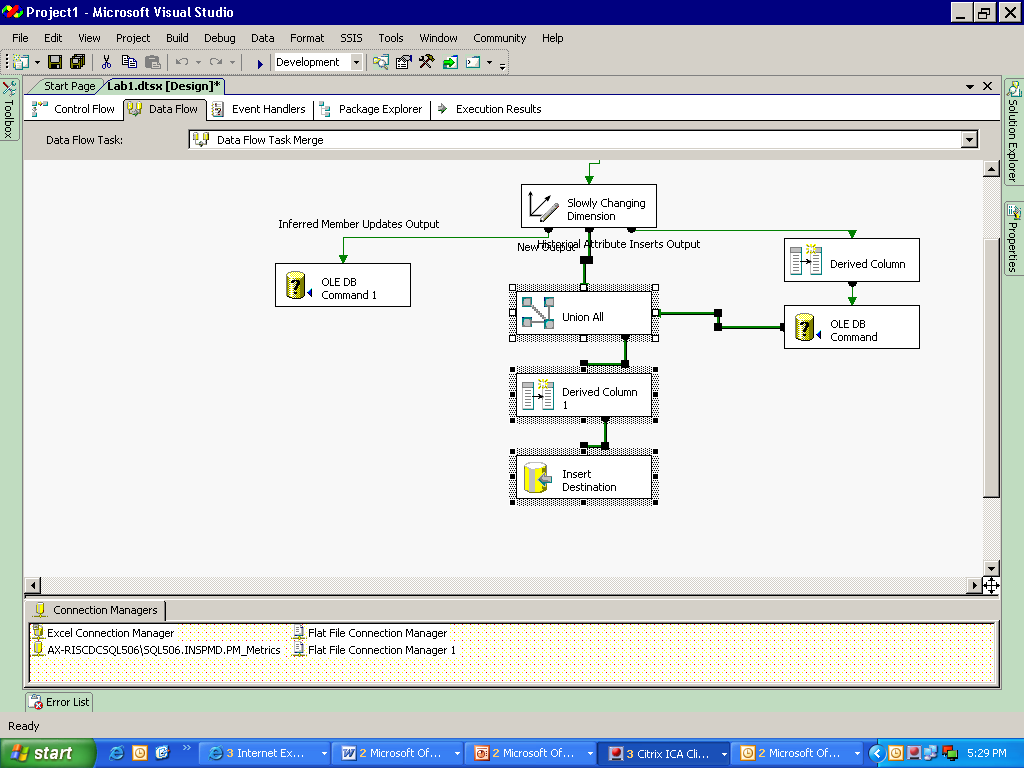


* Click NEXT



* Click FINISH.

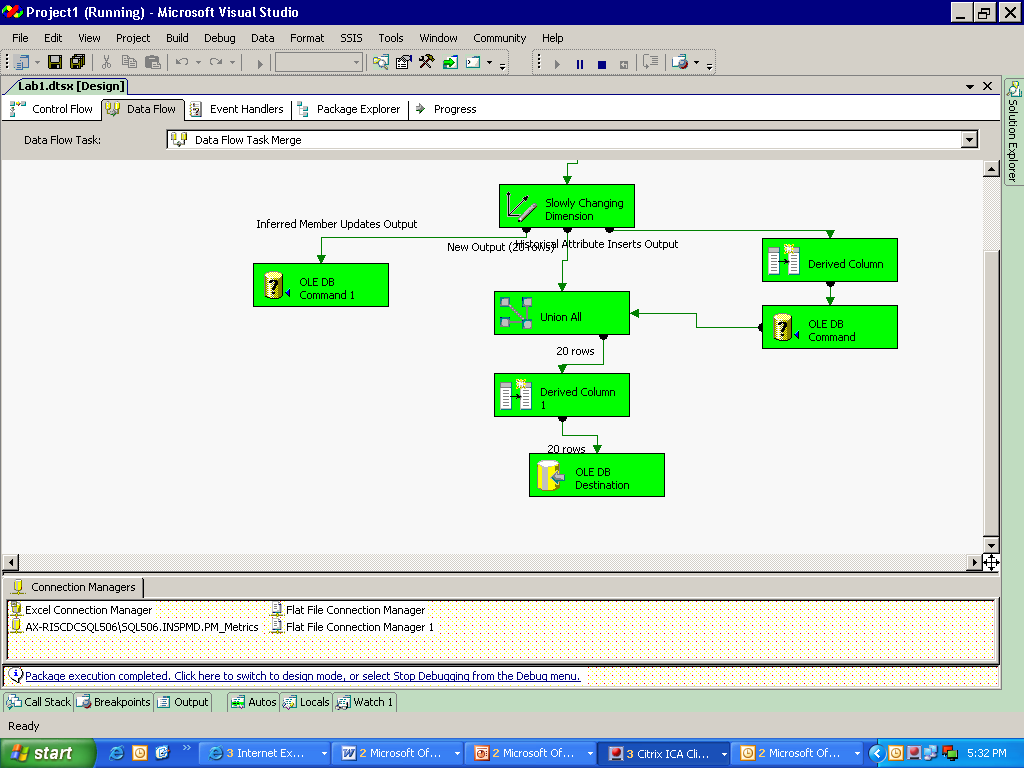
The output for the wizard will be like:



Replace with a destination table.

3) Update the destination with the target table.

4)Save and run the data flow.



**Initial Load will show the data as:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Emp\_code** | **emp\_name** | **Address** | **Age** | **Salary** | **IND** |
| 101 | a | klasfnx | 23 | 20000 | y |
| 102 | b | abc | 43 | 56000 | y |
| 103 | c | pqr | 34 | 40000 | y |
| 104 | d | asd | 56 | 59000 | y |
| 105 | e | abc | 33 | 39000 | y |
| 106 | f | perghjsd | 27 | 36254 | y |
| 107 | g | ghi | 47 | 49000 | y |
| 108 | h | jkl | 55 | 70000 | y |
| 109 | i | mno | 28 | 30000 | y |
| 110 | j | lop | 38 | 36000 | y |
| 111 | k | jshfkag | 49 | 50000 | y |
| 112 | l | vwx | 29 | 31000 | y |
| 113 | m | yz | 30 | 32000 | y |
| 114 | n | qwe | 40 | 49875 | y |
| 115 | o | rtyu | 50 | 58000 | y |
| 116 | p | fnfd | 39 | 39785 | y |
| 117 | q | sdfjfgosf | 25 | 26000 | y |
| 118 | r | jkg | 26 | 27154 | y |
| 119 | s | mnb | 36 | 35647 | y |
| 120 | t | afgh | 47 | 46987 | y |

**Updating the rows:**

update test.EMP\_DETAILS

SET salary=23465 where EmpCode=101

update test.EMP\_DETAILS

SET salary=50200 where EmpCode=107

update test.EMP\_DETAILS

SET salary=26987 where EmpCode=117

**After changes in salary:**

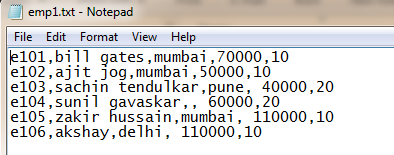
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Emp\_code | emp\_name | Address | Age | Salary | IND |
| **101** | **a** | **klasfnx** | **23** | **23564** | **n** |
| 102 | b | abc | 43 | 56000 | y |
| 103 | c | pqr | 34 | 40000 | y |
| 104 | d | asd | 56 | 59000 | y |
| 105 | e | abc | 33 | 39000 | y |
| 106 | f | perghjsd | 27 | 36254 | y |
| **107** | **g** | **ghi** | **47** | **50000** | **n** |
| 108 | h | jkl | 55 | 70000 | y |
| 109 | i | mno | 28 | 30000 | y |
| 110 | j | lop | 38 | 36000 | y |
| 111 | k | jshfkag | 49 | 50000 | y |
| 112 | l | vwx | 29 | 31000 | y |
| 113 | m | yz | 30 | 32000 | y |
| 114 | n | qwe | 40 | 49875 | y |
| 115 | o | rtyu | 50 | 58000 | y |
| 116 | p | fnfd | 39 | 39785 | y |
| **117** | **q** | **sdfjfgosf** | **25** | **27564** | **n** |
| 118 | r | jkg | 26 | 27154 | y |
| 119 | s | mnb | 36 | 35647 | y |
| 120 | t | afgh | 47 | 46987 | y |
| **101** | **a** | **klasfnx** | **23** | **23465** | **y** |
| **107** | **g** | **ghi** | **47** | **50200** | **y** |
| **117** | **q** | **sdfjfgosf** | **25** | **26987** | **y** |

Lab 11- Extracting, Transforming and Loading Data

|  |  |
| --- | --- |
| **Objective** | To learn   1. How to extract data from flat file 2. How to Transform Data    1. Add new Derived Column    2. Change Data Types    3. Perform a Data Lookup    4. Conditionally route and load data into relational table 3. Load data into relational table |
| **Lab Setup** | * SSDT tool * Existing SSIS project and connection to database |

1)Create a separate folder “Demo2” for placing input data feed files

2)Place a emp1.txt data file in this folder with following data



**IN SSMS:**

3)Create “mumbaiemp” table in sql server database with following structure

CREATETABLE **[dbo]**.[mumbaiemp](

**[empno] [varchar]**(7)PRIMARYKEY,

**[empname] [varchar]**(20)NULL,

**[city] [varchar]**(20)NULL,

**[sal] [money]** NULL,

**[da] [money]** NULL,

**[hra] [money]** NULL,

**[deptno] [int]** NULL,

**dname** varchar(20)

)

4)Create 2 more tables “puneemp” and “otheremp” with exact same structure as above.

5)Create dept table as below:

createtable **dept**

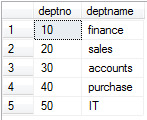
(

deptno intprimarykey,

**deptname** varchar(20)

)

6)Add the following records to the dept table:



**IN SSIS:**

1)In SSDT ,create New SSIS Package

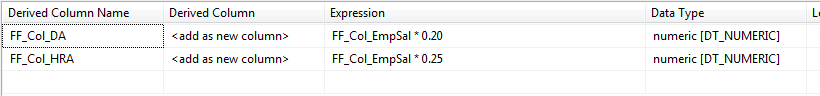
* + Rename it as “FF\_To\_DB\_With\_Transformation.dtsx”

2)In the package designer, goto data flow tab and click the link in the middle to create a new data flow task.

3)Same as was done in Lab 1, drag a flat file source task from toolbox onto designer surface and make it point to emp1.txt file placed in Demo2 folder.

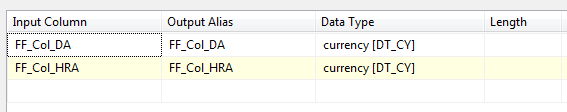
4)Next, drag derived column task from transformation section of toolbox

* + Connect the flat file task green connector to this task
  + Double click this task and configure the additional derived columns as shown below:



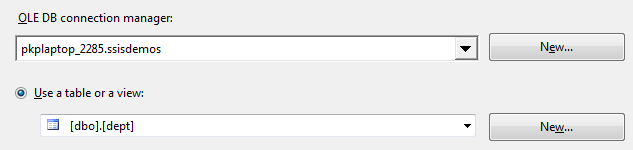
5)Drag a Data Conversion task and connect the Derived Column task green connector to this task.

* + Double click it and change the data type of the following to columns from DT\_Numeric to DT\_CY as shown below:



6)Drag a lookup transform task and connect the data conversion task to it.

* + Double click lookup task, go to connection tab, create a new connection manager which will point to the sql server database which has “Dept” table created earlier in this lab.

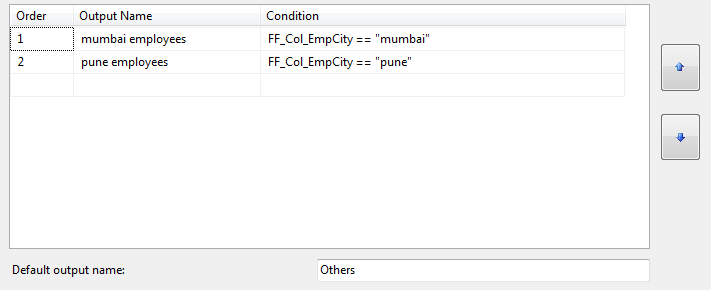


* Goto columns tab, and configure this tab as shown below. (connect FF\_Col\_EmpDeptno to Deptno in Dept Table, select deptname column)



7)Drag a conditional split task onto surface rename it as “Split Employee records based on city”, connect the green connector of lookup task to this task a dialog will popup, from drop down select “lookup match output”.

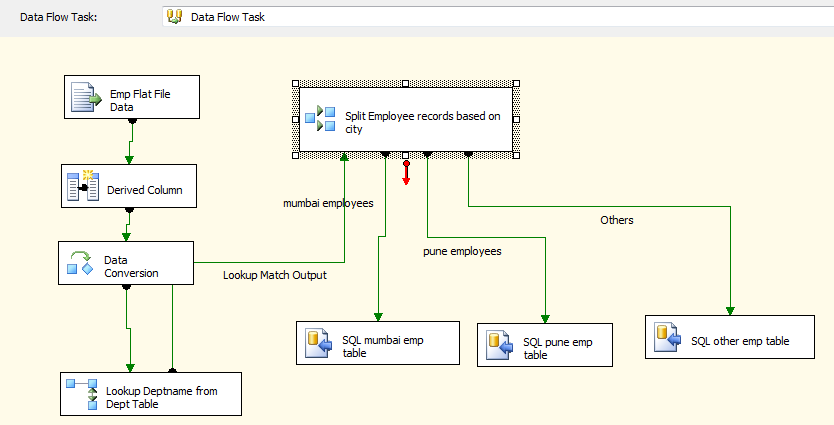
* Double click conditional split task and configure as below: (there are 2 named outputs which will send only those rows that satisfy the specified condition, and there is default output as well called “others”)



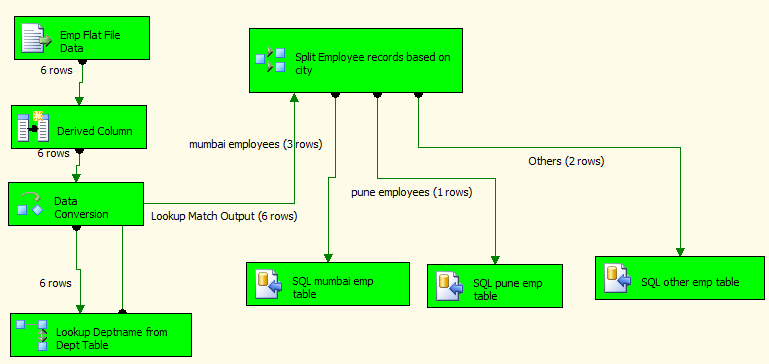
8)Drag 3 SQL Server Destinations tasks from data flow destinations section of toolbox. Rename them as “SQL Mumbai emp table”, SQL Pune Emp Table, SQL Other emp table.

9)Connect conditional split task to the first “SQL Mumbai emp table” sql server destination task. A dialog box will popup select the output name to connect as “mumbai employees”. Similarly connect the remaining outputs to the corresponding tasks.

10)Configure each of the these tasks by double clicking them so that they point to appropriate tables in the sql server. The complete SSIS package is shown below:

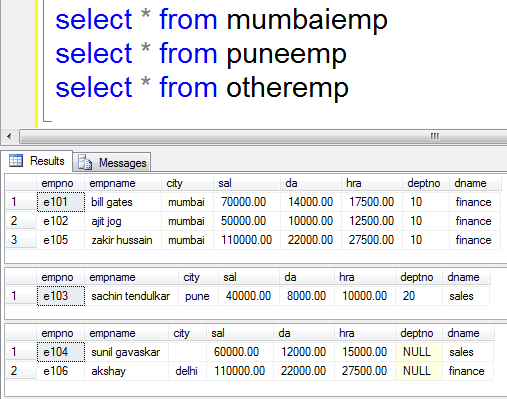


11)Run the package, when the entire package executes successfully it will look as shown below:



12)Stop the Package. 

13)Connect to sql server using SSMS and check the table data



Lab 12- Execute SQL Task& Script task

|  |  |
| --- | --- |
| **Objective** | Execute a stored procedure from SSIS to get employee salary and display output in message box |
| **Lab Setup** | * SSDT tool * Existing SSIS project and connection to databse |

1)As per below create a stored procedure to get employee salary as outcome when passing salary name as input parameter.

createprocedure [dbo].[emp\_name9](

@emp\_name varchar(20),

@salary intoutput

)

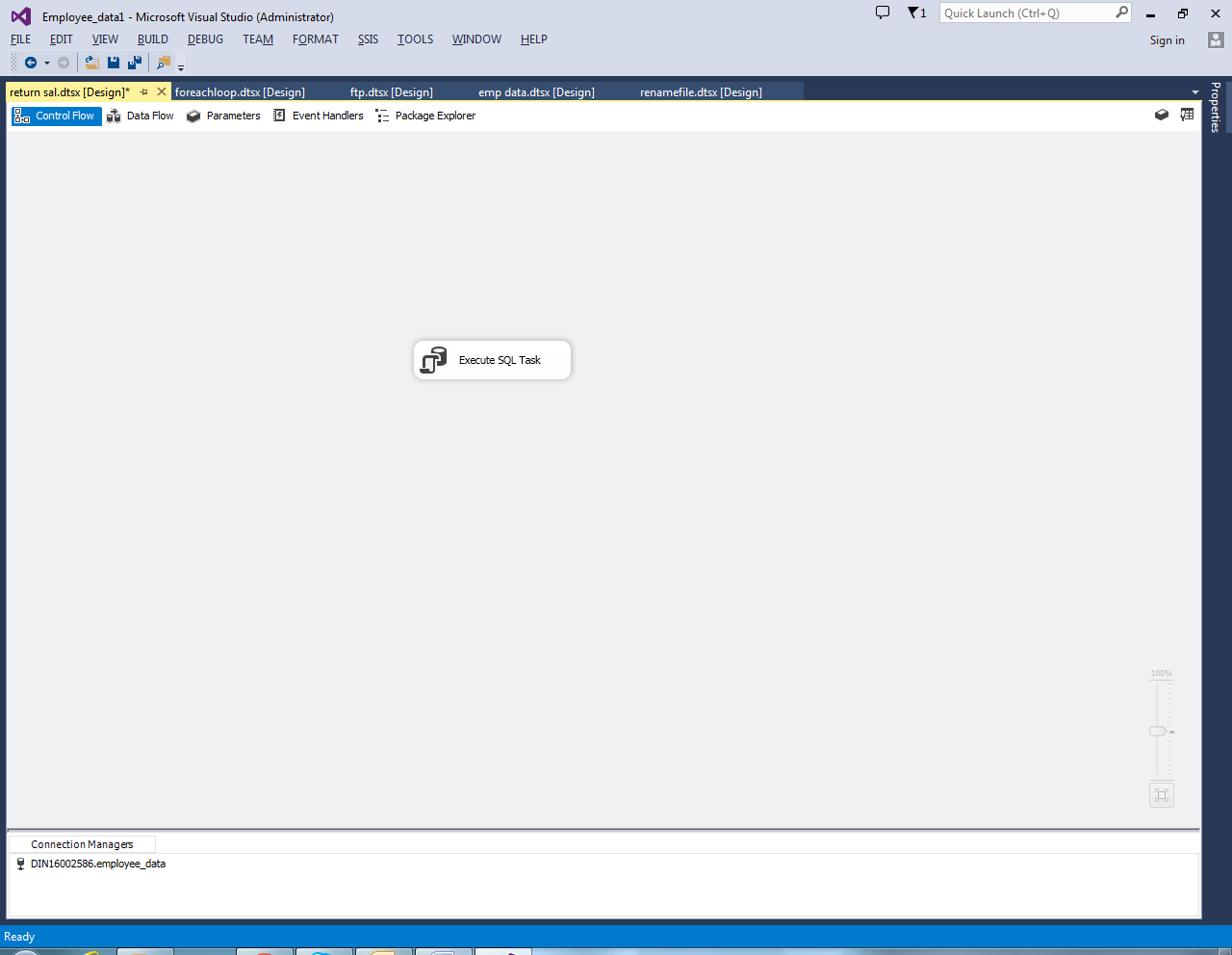
as

begin

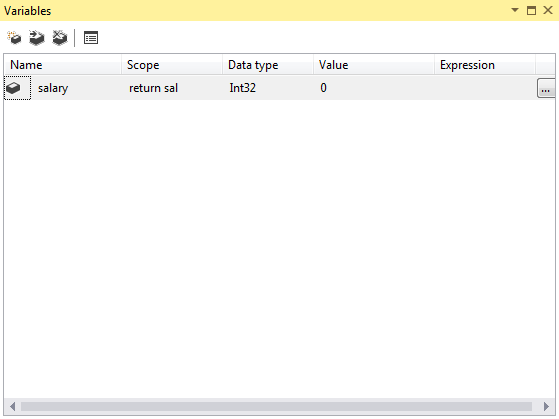
select @salary=[salary] from [dbo].[empdata] where [emp\_name]=@emp\_name

end

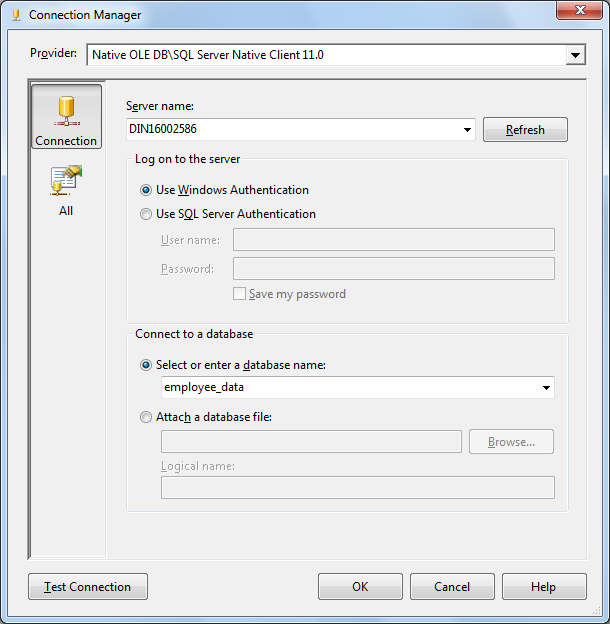
2)In SSDT create a new package and drag the execute sql task to the control flow.



3)Create a variable as per below. You can create variables by right clicking on the control flow or from the SSIS menu above the control flow.



4)Create a new connection string as per below. If connection string already exists no need to create new connection string, you can use the same connection string.



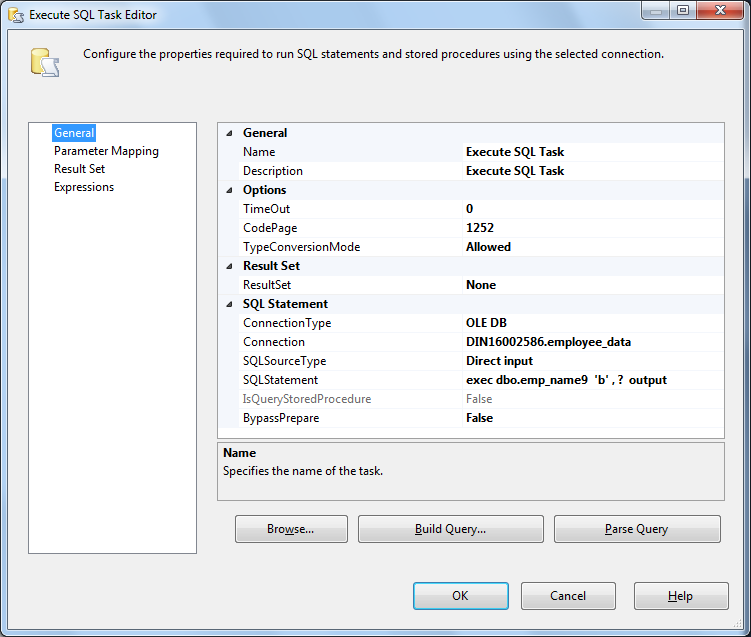
5)Choose the provider as SQL Server Native Client 11.0.

6)Choose your server name and database which holds your employee\_data table.

7)Click the Test Connection tab and test the connection.



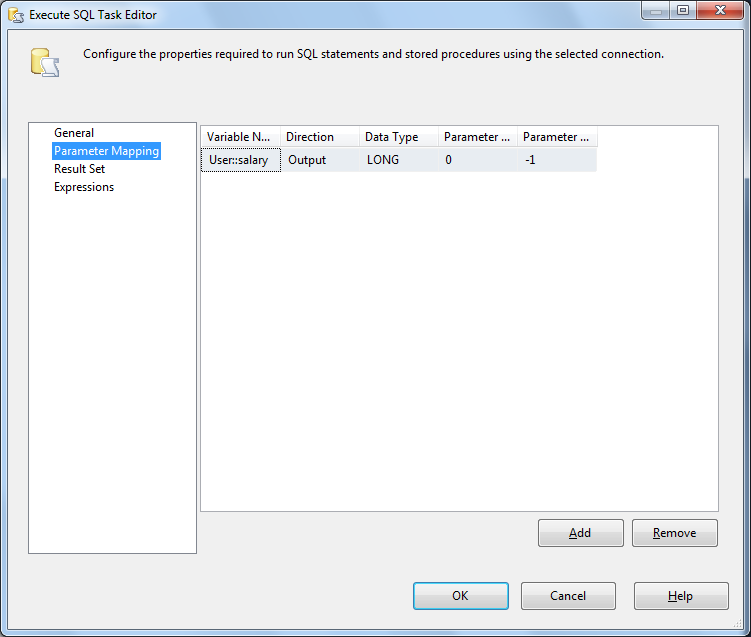
8)Edit the Execute SQL task and choose the connection string which you have created and SQL Source type as Direct Input as per below.



5)Provide the execute statement in SQLStatement field as per below.

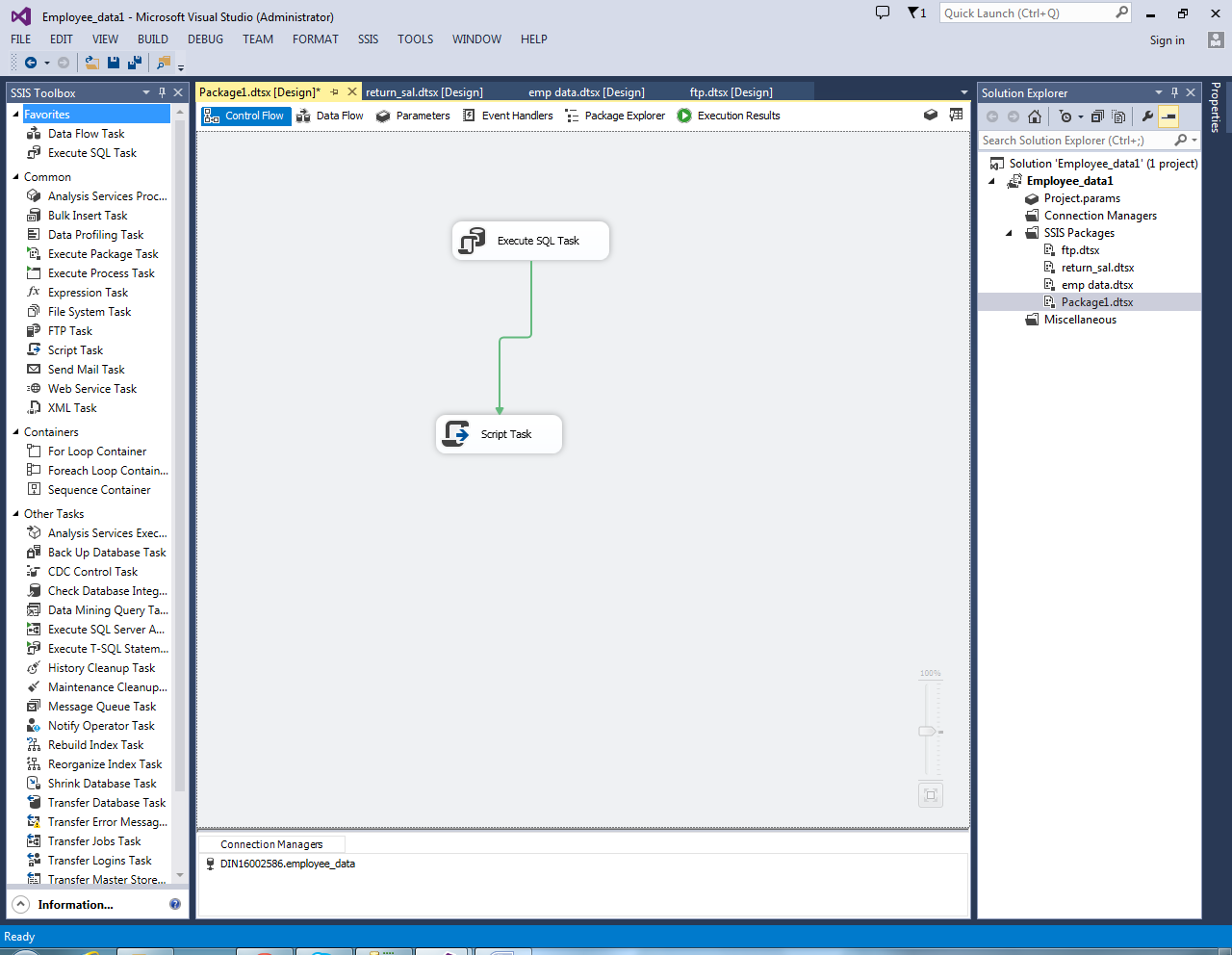
**SQL Statement:**exec dbo.emp\_name9 'p', ? output

6)In parameter mapping choose the parameter you have created and give the direction as output and parameter name as 0.

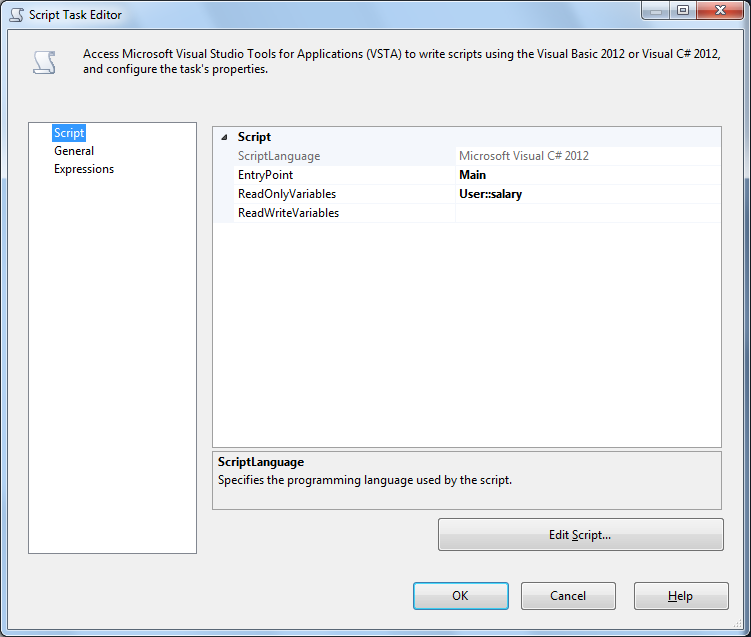


7)Now will use Script task to display salary of the Employee that we have stored in output variable.

8)Drag a Script task to the control flow and connect the green arrow from execute sql task to the script task.



9)Now edit the script task and in the read only variable choose the output parameter User::salary

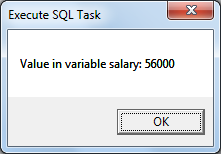


10)In Edit Script and add the below code in main function to display the salary of the employee.

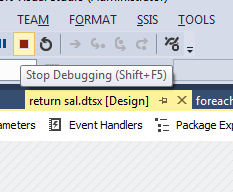
MessageBox.Show(String.Format("Value in variable salary: {0}",

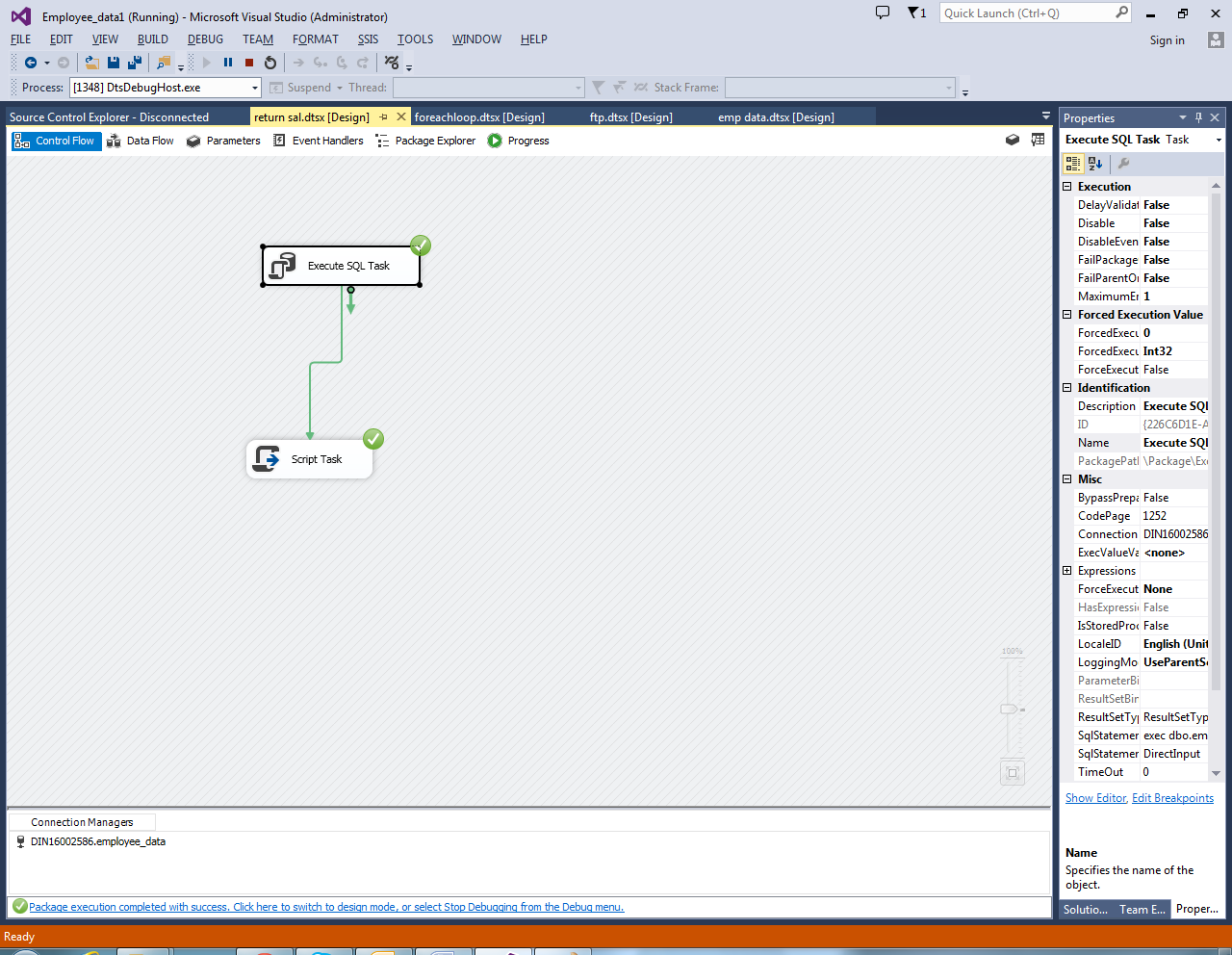
Dts.Variables["User::salary"].Value),"Execute SQL Task ");

11)Now execute the package by right clicking the project name in the solution explorer and you can see the salary of the employee displayed in the message box.



12)After executing stop debugging to return to the design mode.

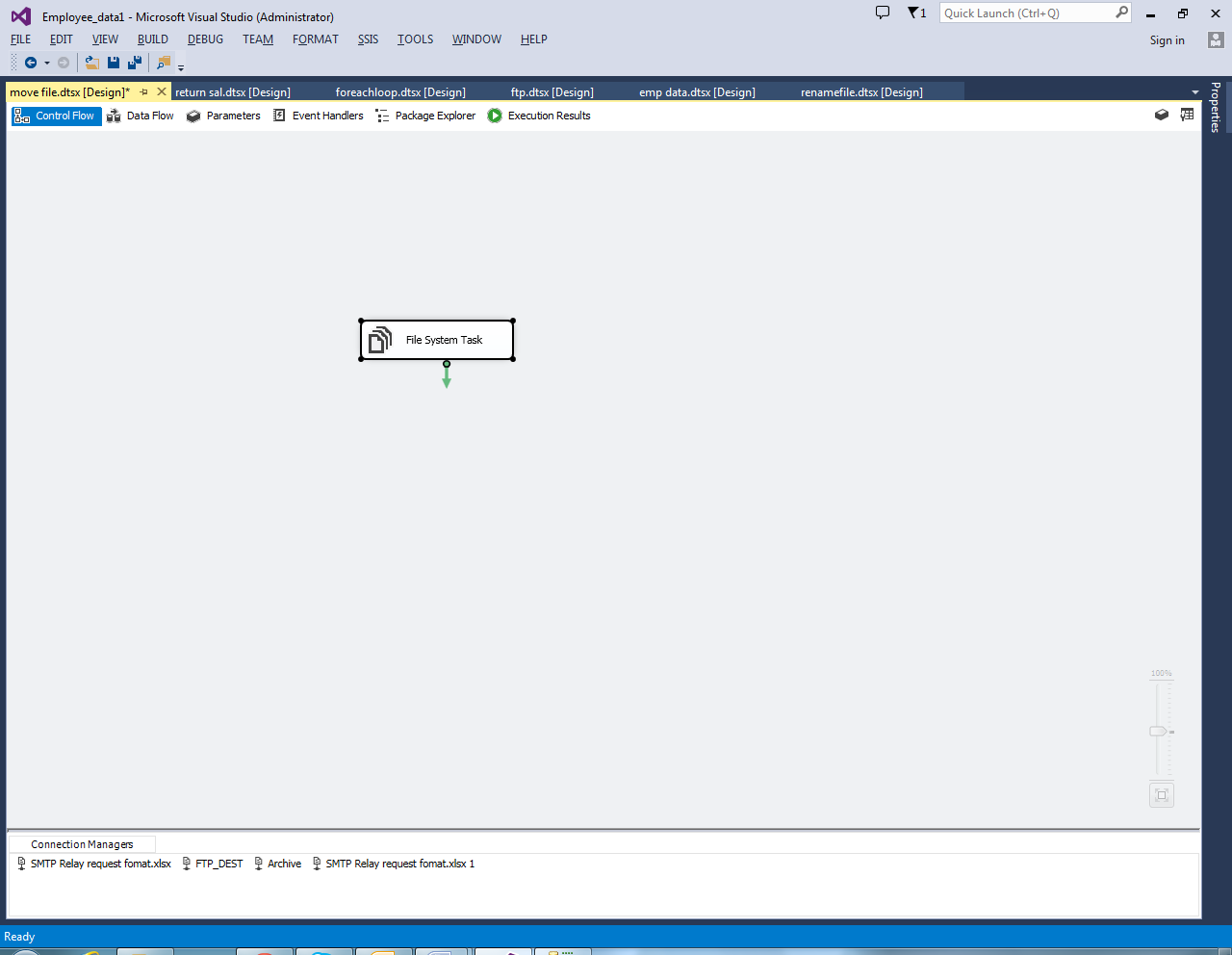




Lab 13-File System Task

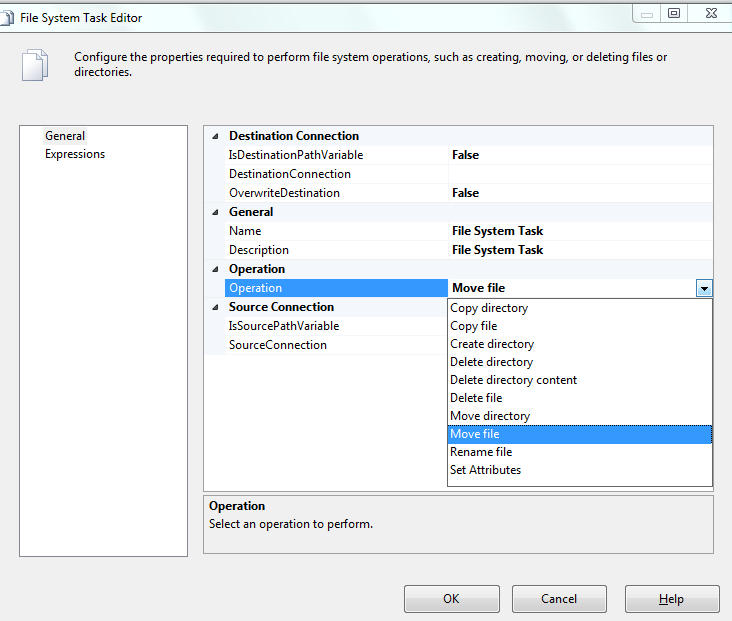
|  |  |
| --- | --- |
| **Objective** | To learn  1.How to move file from one location to another location. |
| **Lab Setup** | * SSDT tool * Create package |

1)In order to move file using file system task , drag and drop the File System Task into the Control Flow region.



2) Double click on it will open the File System Task Editor to configure it.

3)In this example, We are Moving single file so, Please change the **operation** property to **Move File**as shown in the below screenshot.

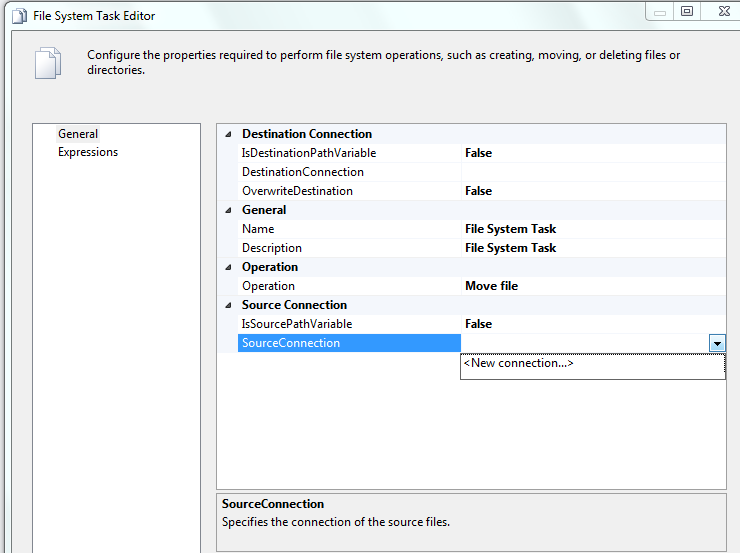


4)Let us configure the Source Connection by selecting the **Source Connection** property.

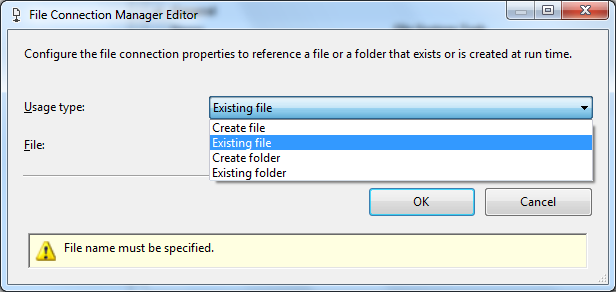
5)If you already created the File Connection Manager before then select the created one or If you stored the Source Connection in the Variable then, please change the **IsSourcePathVariable**property to TRUE and select the Variable Name.

6) Here, We haven’t created any connection Manager before so, We are selecting

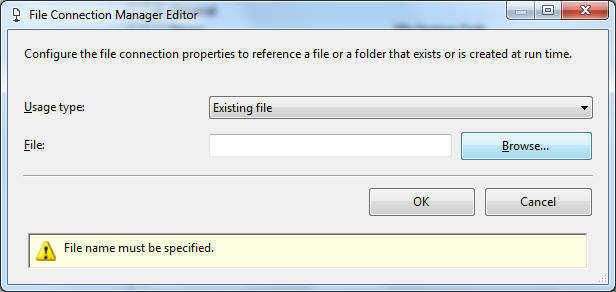
**<New Connection…>**

****

7)Once you click on the **<New Connection..>**option, File Connection Manager Editor will be opened to configure it. In this example we are Moving existing file so we are selecting Existing File option from the**Usage Type.**



8)Click on the Browse button to select the Existing File from the file system.

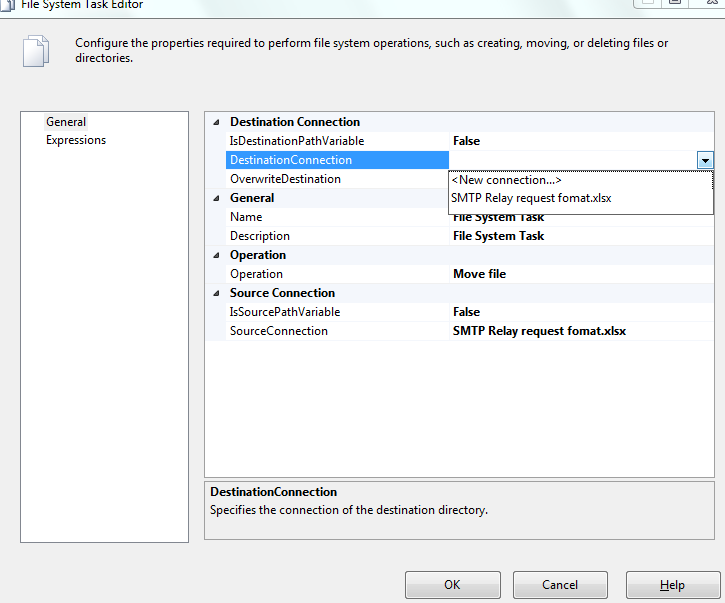


9) From the above screenshot you can observe that, We selected the SMTP Relay request format.xlsx file inside the File System Task Folder.

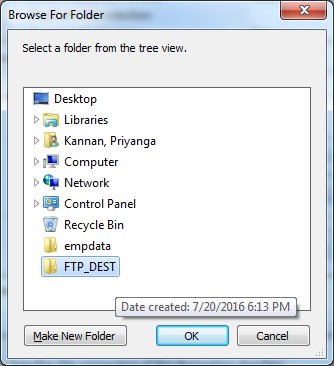


10)Now we have to configure the Destination Connection so, Please select the **DestinationConnection**property.

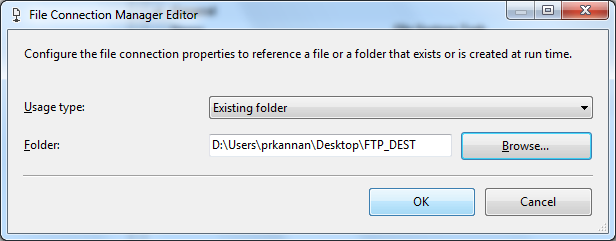
11)We haven’t created any connection Manager before so, We are selecting **<New Connection..>**



12)In this example we are Moving existing file to the already existing folder so, we are selecting Existing Folder option from the**Usage Type.**



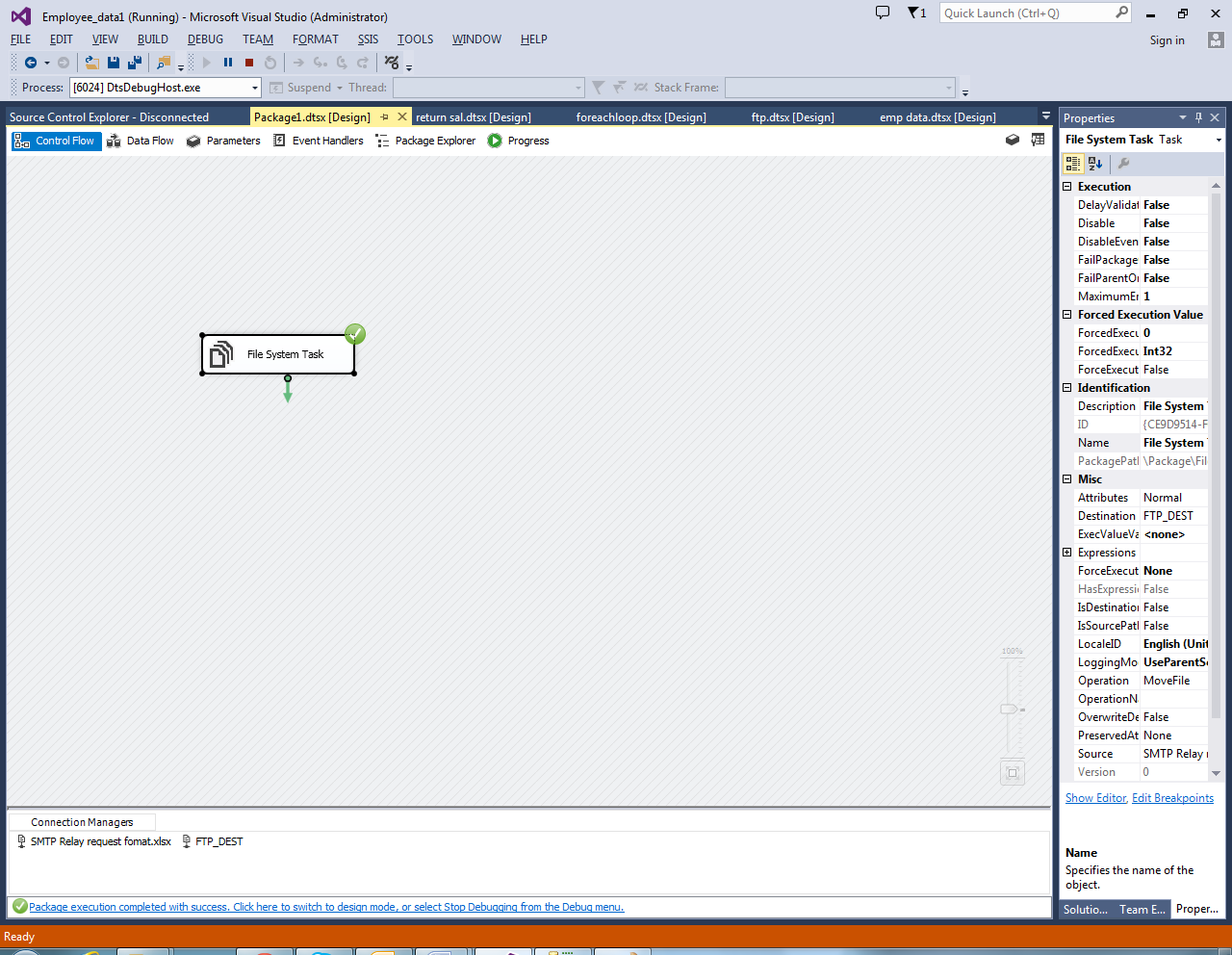
13) From the above the above screenshot you can observe that, We selected the Copied FTP\_DEST as the destination folder. Click Ok button to select it.



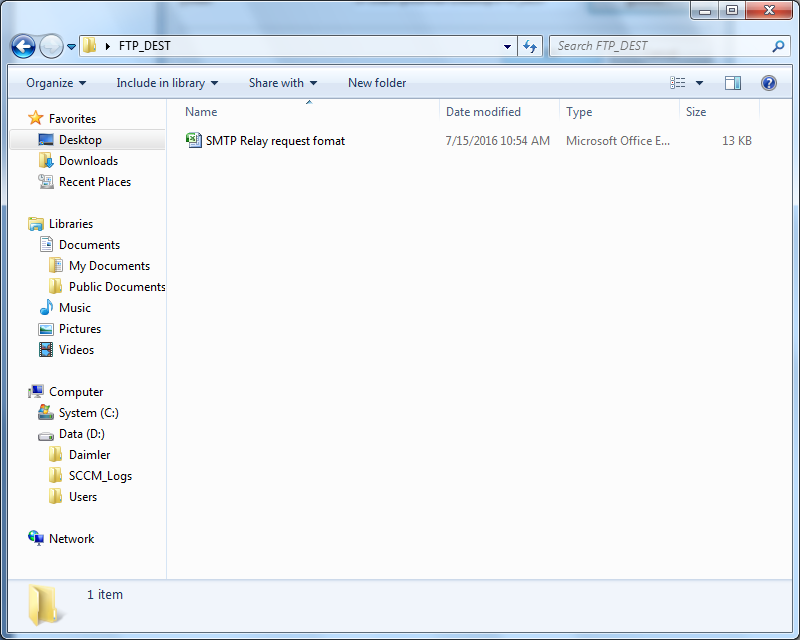
14)Click Ok to finish configuring the File Connection Manager for the destination.

15)Click Ok to finish configuring Move File using File System Task in SSIS package.

16)Click Ok to finish configuring Move File using File System Task in SSIS package. Let’s run and see whether we successfully Moved the Customers.txt file using the File System Task or Not.



17) We successfully Moved the SMTP Relay request format.xlsx file present in the File System Task Folder to the Copied Folder.



Lab 14-For each loop container with Execute SQL task and Script task

|  |  |
| --- | --- |
| **Objective** | To learn   1. How to make use of control flow tasks 2. Use of For Each Loop container task as result set iterator 3. Calling Stored Proc 4. Using a Custom Script Task Component 5. Using DTS Variables |
| **Lab Setup** | * SSDT tool * Existing project and connection to database |

1.Create a separate folder “Demo3”. Create a script CreateTab.sql and put the following T-SQL code in it:

2.Connect to SQL server using SSMS and create the following tables

CREATETABLE **[dbo]**.[Department](

**[DepartmentID] [int]** PRIMARYKEY,

**[Name] [nvarchar]**(50)NOTNULL,

**[Budget] [money]** NOTNULL

)

CREATETABLE **[dbo]**.[Course](

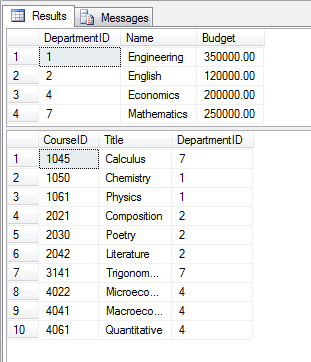
**[CourseID] [int]** PRIMARYKEY,

**[Title] [nvarchar]**(100)NOTNULL,

**[DepartmentID] [int]** NOTNULL

)

3.Add the following records into the above tables:



4.Create a folder “Demo3”, and create a script “createproc.sql” in this folder. Put the following code in the script

ifOBJECT\_ID('getcourses')isnotnull

dropprocedure **GetCourses**

go

createprocedure **GetCourses**

( **@dno** int,

**@coursecnt** intoutput

)

as

select **@coursecnt** =count(\*)

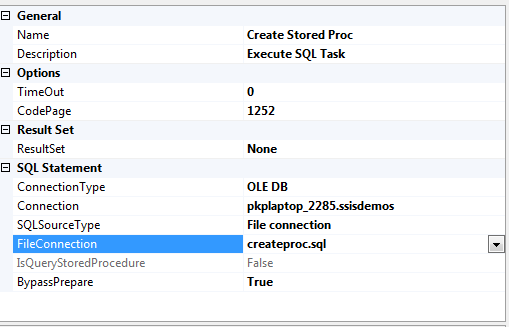
from **course** where **departmentid** = **@dno**

go

5.Add a new ssis package, Project => New SSIS package, name it “ExecuteSQLParametersResultSets.dtsx”.

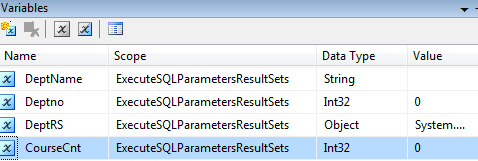
6.In the control flow tab designer, drag a “Execute SQL Task”. Rename it as “Create Stored Proc”

* 1. double click task, set the following properties:
     1. connection type: oledb
     2. sql source type: file connection.
     3. connection: create new connection and connect to your sql server.
     4. file connection: create a new connection which points to the “createproc.sql” script file created in step (4).



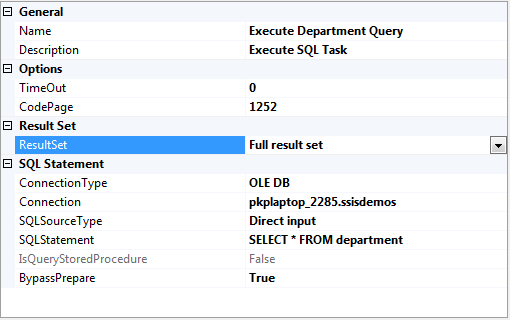
7.Select the package, by single clicking the package designer surface in free area, Open Variable Window, View => Other Windows => Variables

* 1. Create a variable “DeptRS” clicking top left small button in the variables window, set datatype to “Object”
  2. Create 3 more variables Deptno Datatype Int32, DeptName datatype: string, CourseCnt datatype int32

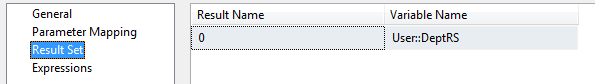


8.Drag another “Execute SQL Task”. Rename it as “Execute Department Query”

* 1. Double click task, set the following properties:
     1. ResultSet: Full Result Set
     2. connection type: oledb
     3. sql source type: direct input
     4. SQL Statement: SELECT \* FROM department

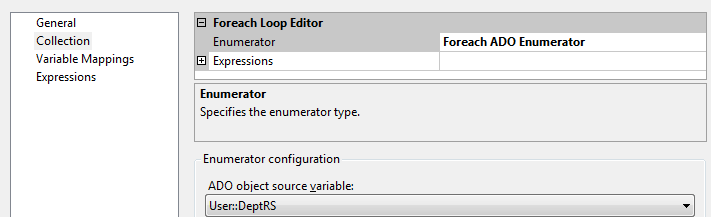


9.Goto ResultSet TAB, click “Add” button and configure as below and click ok

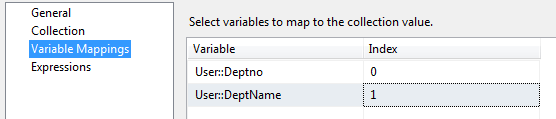


10.Drag a For Each Loop Container Task. Rename it “Iterate the Department Rows”

* 1. Double click the task, goto collection tab, and set enumerator property to “ForEach Ado Enumerator” , set ado object source variable to User::DeptRS

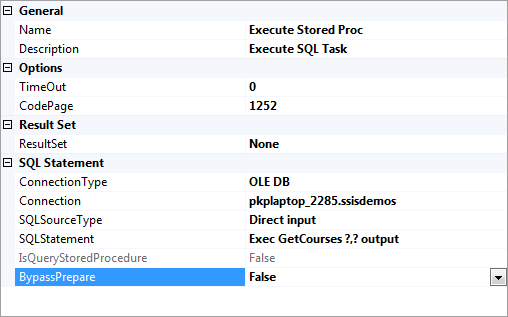


11.Goto Variable Mappings tab and configure as below and then click Ok



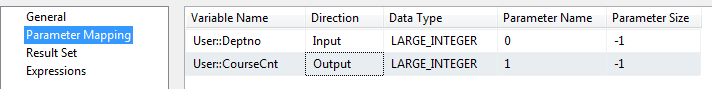
12.Drag a Execute SQL Task inside the For Each Loop Container, Rename the task as “Execute Stored Proc”

* 1. Double click task, set the following properties:
     1. ResultSet: None
     2. connection type: oledb
     3. connection: specify a connection to your sql server
     4. sql source type: direct input
     5. SQL Statement: Exec GetCourses ?,? output
     6. ByPass Prepare: false

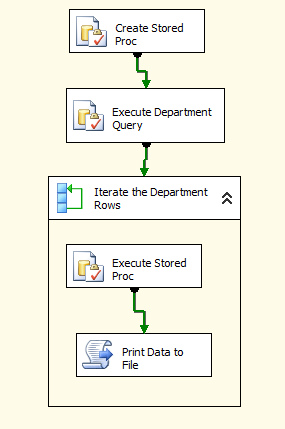


1. Goto Parameter Mapping Tab and configure it as below and click Ok

Note: Click Add button to add variable entries as parameters



1. Drag a script task inside the For Each Loop container task and rename it as “Print Data to File”
2. Connect the various tasks that are dragged sequentially as shown below:



1. In the package designer right click the “Connection Managers” section at the bottom in free space and select “new file connection”
   1. Set Usage Type: Create File and
   2. Browse and specify filename as Results.txt in Demo3 folder created at the start of this lab. Click ok
2. Double the script task and in the ellipses for readonly variables property, select User::CourseCnt, User::DeptName, User::Deptno Variables.
3. Click Edit Script button
4. Define the following code in the main function

publicvoid Main()

{

int? dno,coursecnt;

string dname;

string FileName = (string)Dts.Connections["Results.txt"].AcquireConnection(null);

dno = (int)Dts.Variables["Deptno"].Value;

if (Dts.Variables["CourseCnt"].Value == null)

coursecnt = 0;

else

coursecnt = (int)Dts.Variables["CourseCnt"].Value;

dname = (string)Dts.Variables["DeptName"].Value;

using (StreamWriter sw = File.AppendText(FileName))

{

sw.WriteLine("The No Of Courses for Department " + dname + "(" + dno.ToString() +") is " + coursecnt.ToString());

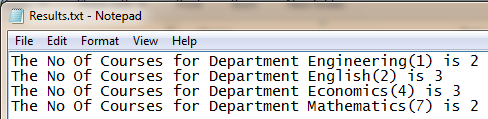
sw.Close();

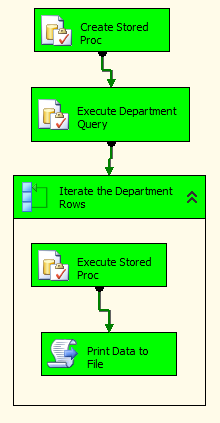
}

Dts.TaskResult = (int)ScriptResults.Success;

}

1. Build the script Ctrl+Shift+B. Once succeeds exit the script editor. Click ok on the script task dialog box to go to the package designer.
2. Run the package and open the Results.txt file as shown below and then stop the package





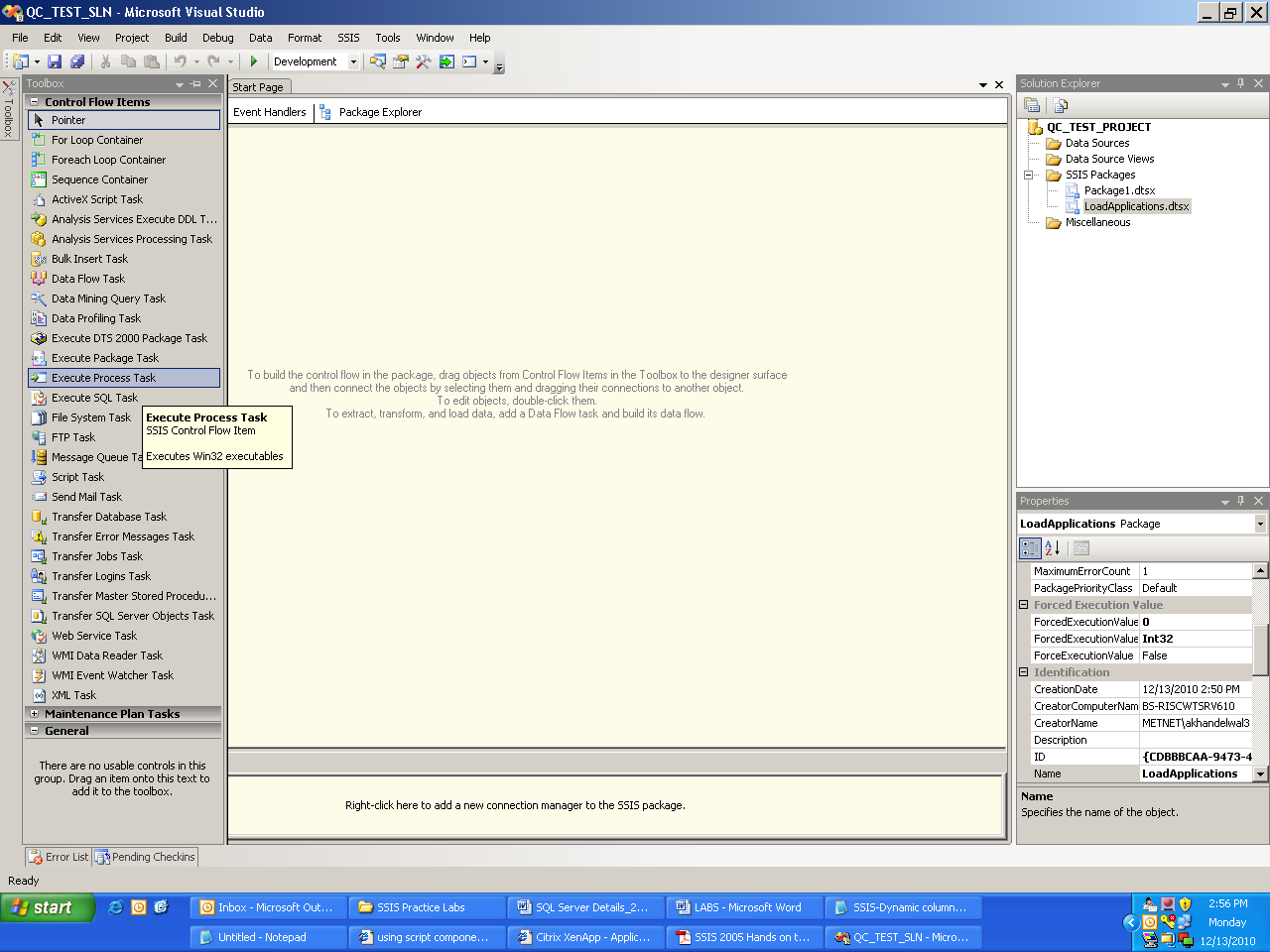
Open a different application based on day of week

Lab 15- Execute Process Task

|  |  |
| --- | --- |
| **Objective** | The purpose of this Lab is to show an example of property expressions in a fringe operation, using the Execute Process task to open other applications. |
| **Lab Setup** | * SSDT tool * Create package. |

1)Save and Close all other open packages and create a new package as shown below and then rename it:

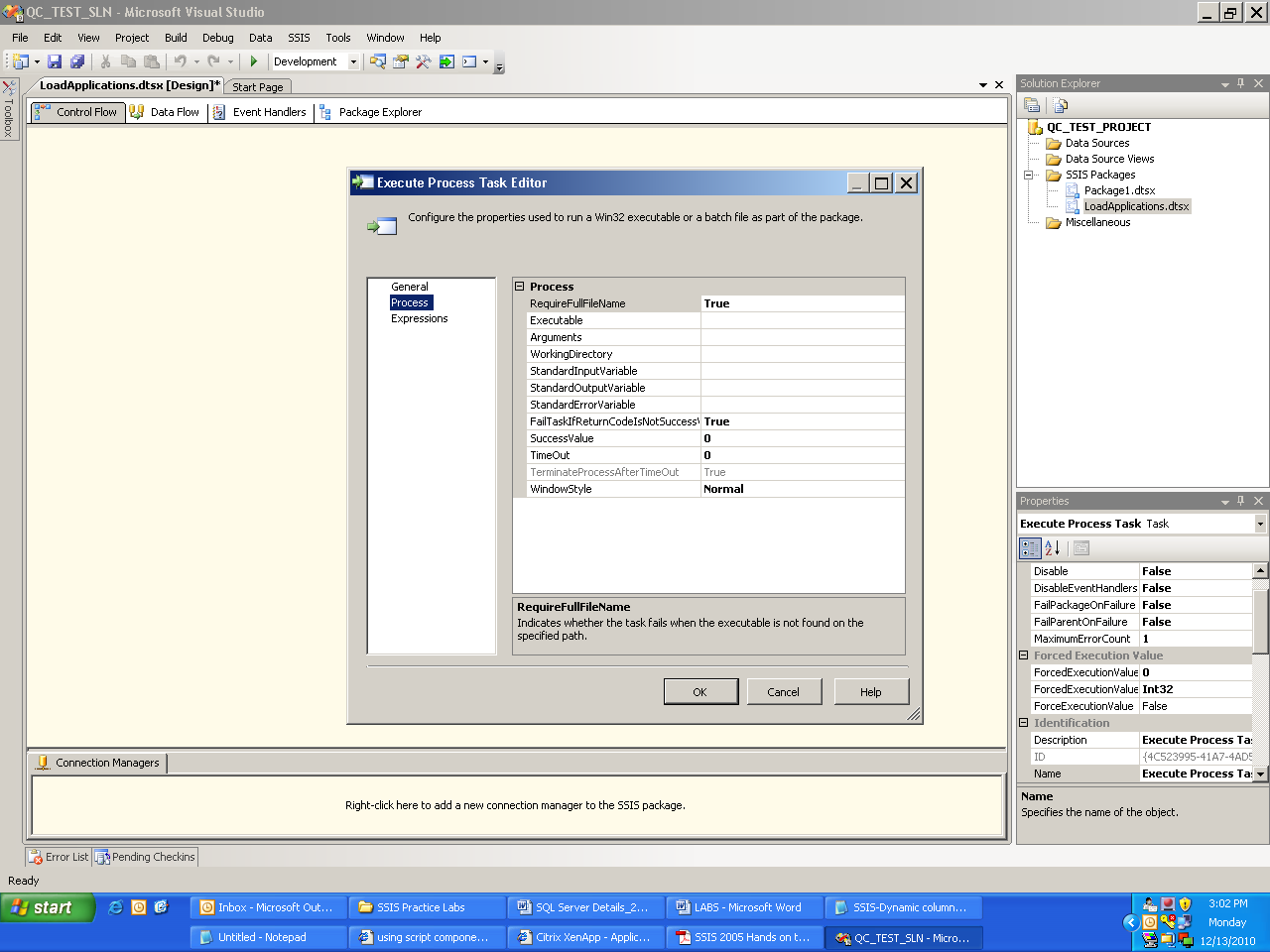
2)Add an Execute Process task to the control flow of this new package:



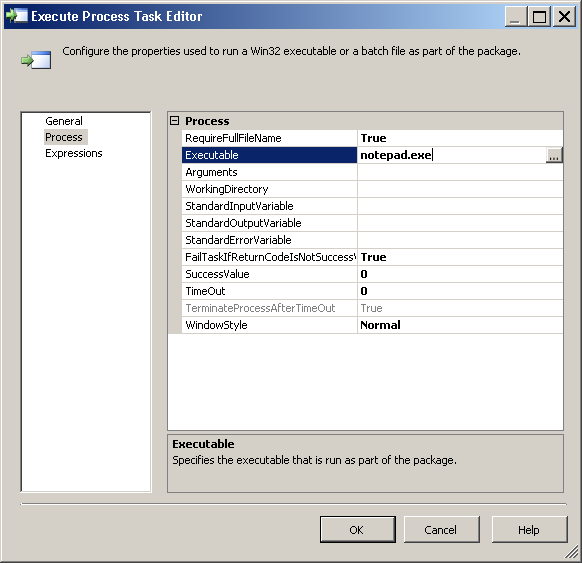
**Note:**

[Using property expressions we can configure the single task to open a different application based on the day of the week. Open either notepad.exe or mspaint.exe depending on day of week.]

3)Now double click on the Execute Process task to go to editor and click on the Process Page:

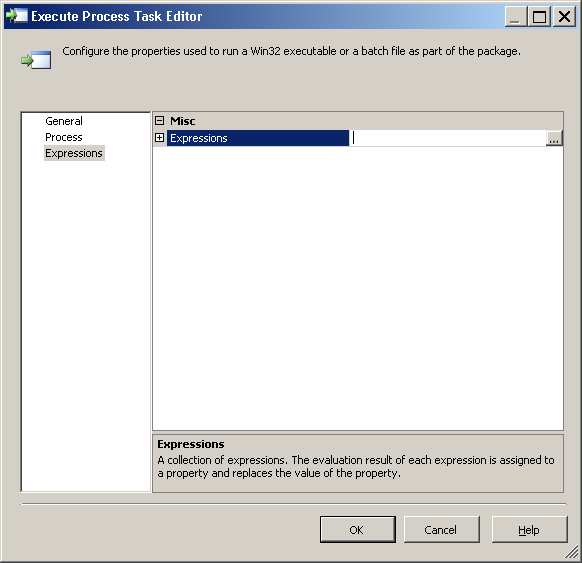


4)Now enter “notepad.exe” for the ‘Executable’ property.



5)Now click the “Expressions” page. In the right pane click in the empty row for “Expressions” and then press the ellipse button which will take you to the Property Expressions Editor.

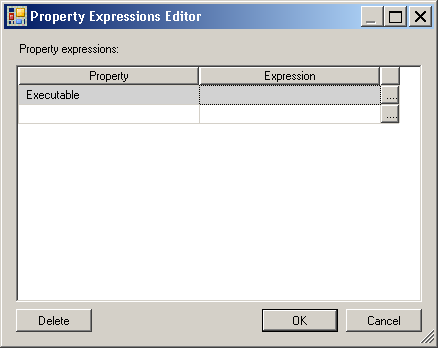
Press this ellipse button



6)Now choose the “Executable” property and either copy/paste the following expression or press the other ellipse button to go into the expression builder and build this yourself and then evaluate the expression:

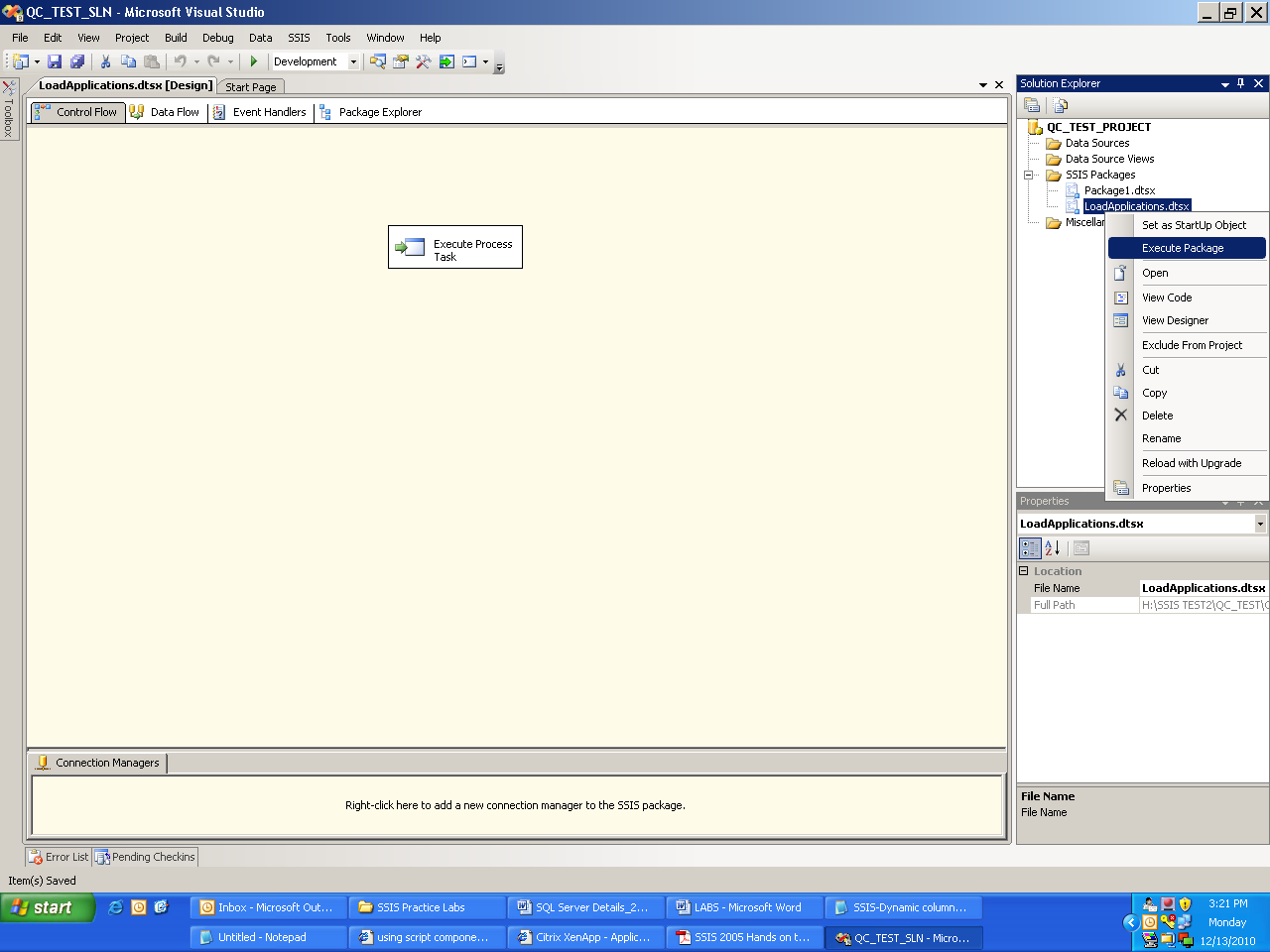
**Expression:**

DATEPART("weekday", GETDATE()) ==5?"notepad.exe":"mspaint.exe"

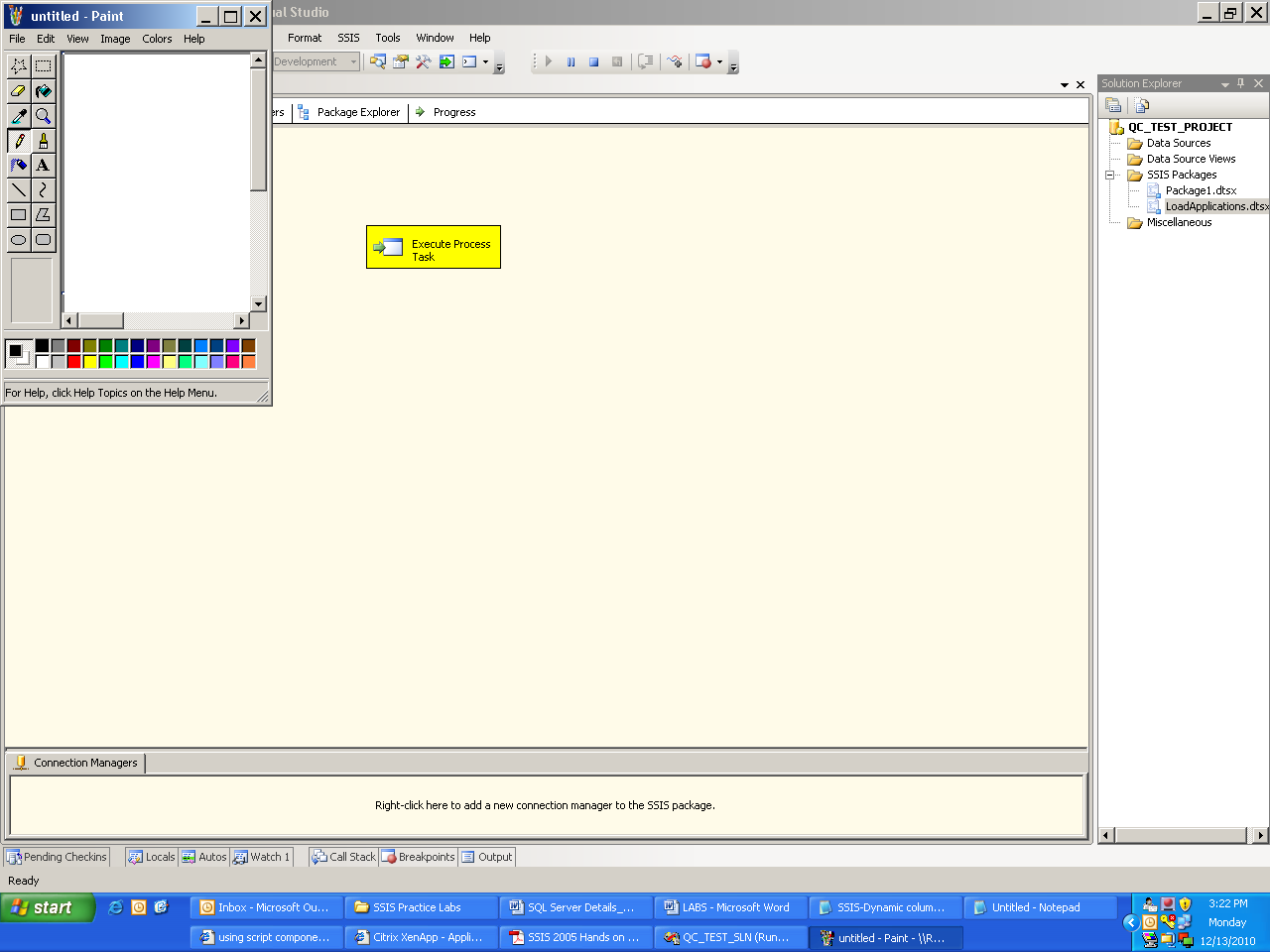


Click this ellipse button to go to the Expression Builder to build expression.

7)Save the package. In the solution explorer select this particular package and select “Execute Package”.

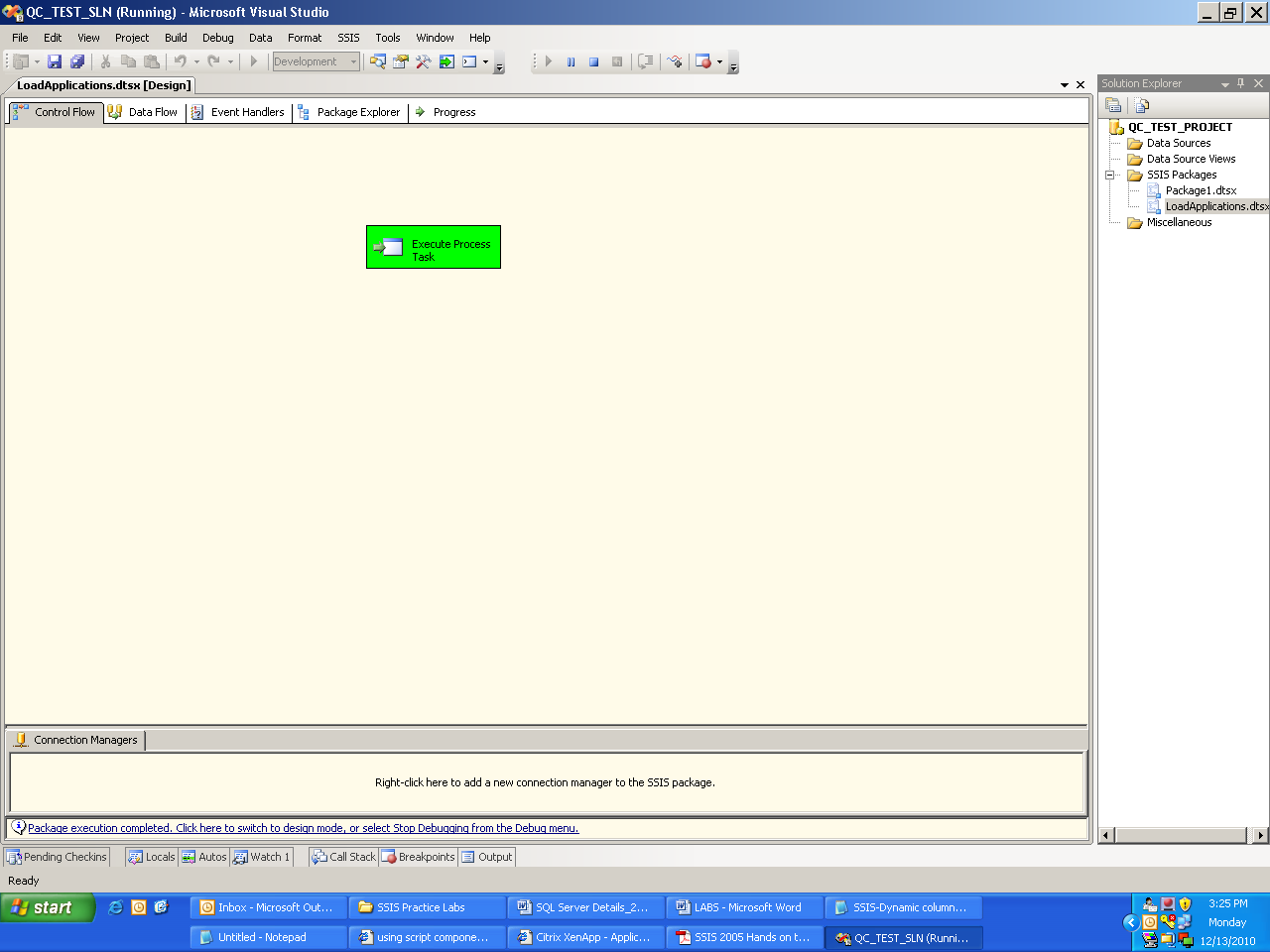


8)IF the current day is Thursday (5) then Windows Notepad will open, otherwise Paint will open as shown below:



|  |
| --- |
| **Note:** The Execute Process task is still yellow and the package is still considered ‘running’ until you close the application that opened, then the task will turn green. |

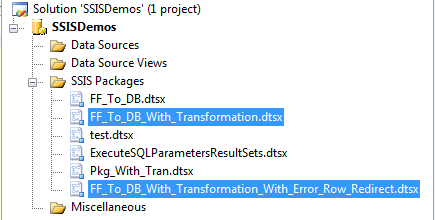
9)After closing the Paint application this task turned green.



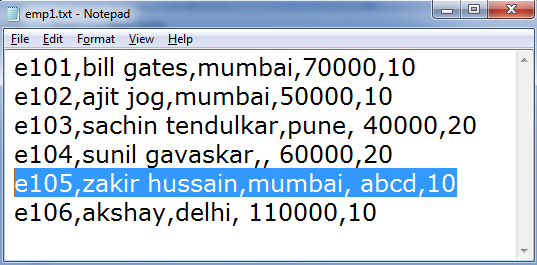
Lab 16- Using Error Redirect Row

|  |  |
| --- | --- |
| **Objective** | To learn   1. How to use error redirect output to identify the rows having processing errors and prevent the entire ELT process from halting |
| **Lab Setup** | * SSDT tool * Login details for connecting to the database and Create package. |

1)In Solution Explorer copy and paste the “**FF\_To\_DB\_With\_Transformation.dtsx**”  
and rename as “**FF\_To\_DB\_With\_Transformation\_With\_Error\_Row\_Redirect.dtsx**”. Click “Yes” to the dialog box that will come up to complete renaming.



2)Goto Demo2 folder created for Lab 2 open “**emp1.txt**” file and change the salary value of one the employee to character as shown and save changes



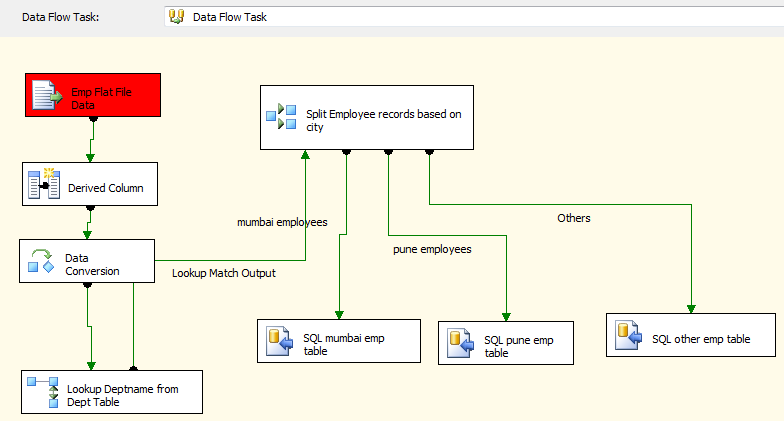
3)Prior delete all the records from the tables

deletefrom **mumbaiemp**

deletefrom **puneemp**

deletefrom **otheremp**

4)Open “**FF\_To\_DB\_With\_Transformation.dtsx**” package and execute the package. There will be error in the package execution as shown:



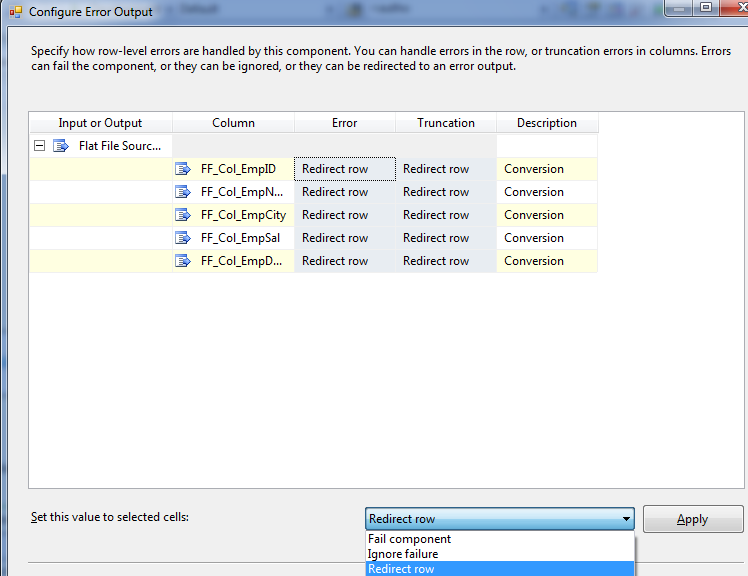
5)Stop the Package execution from VS Studio.

6)Check the SQL tables, no data would be transferred.

7)Open “**FF\_To\_DB\_With\_Transformation\_With\_Error\_Row\_Redirect.dtsx**” goto data flow tab, from toolbox drag “**Flat File Destination**” task.

8)Join the Red Error Output Connector of the “**Emp Flat File Data**” Flat File Source Task to it.

9)In the dialog box select all the error and truncation column entries. (Use: “shift” + click) select “redirect row” from dropdown below and click appl. Click ok

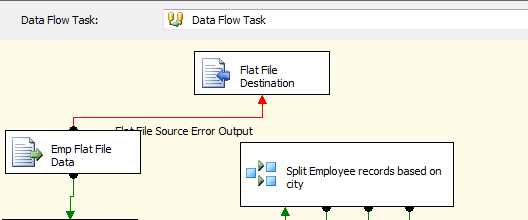


10)Double Click the “**Flat File Destination**” task click new “New” button to create a new Connection Manager. Select “Delimited” in the dialog box that pops. Name the connection manager as “FFCM\_Error\_Output\_File”.

11)In filename specify the file as “emperr.txt” in the same directory Demo2

12) For eg: C:\Users\jogajitm\Desktop\Project 1\Folders\Demo2\emperr.txt

13)Click Ok.



14)Save Changes to package. Run it.

15)The package will succeed. Stop it. The error row will come into the emperr.txt file rest of the rows will be successfully transferred to the SQL tables.

Lab 17- Transaction support and Checkpoint

|  |  |
| --- | --- |
| **Objective** | To learn   1. How to use Transaction support available for atomicity 2. How to use Checkpoint feature to resume the package from the point where it was abandoned |
| **Lab Setup** | * SSDT tool * Login details for connecting to the database and Create package. |

1)Create Demo5 folder for this lab. Create “In”, “Out” and “Checkpoint” 3 folders under it.

2)Connect to sql server and create 2 tables as below:

CREATETABLE **staginginvoices**

(

**invno** intPRIMARYKEY,

**invamt** numeric(18, **0**)NULL,

**invdate** smalldatetimeNULL,

)

CREATETABLE **invoices**

(

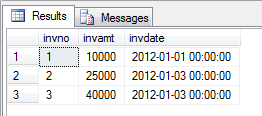
**invno** intPRIMARYKEY,

**invamt** numeric(18, **0**)NULL,

**invdate** smalldatetimeNULL,

)

Add few records into staging invoices table for eg as below:



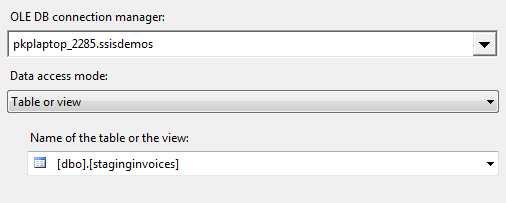
3)Add a new ssis package and name it as “**Pkg\_with\_Tran\_and\_CheckPoint\_Demo.dtsx**”.

4)In the control flow, drag a sequence container task. Rename it “Transfer Invoices to Table and File”.

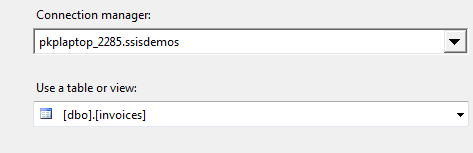
5)Drag a data flow task inside this sequence container. Rename data flow task to “Transfer Invoices to SQL Table”.

6)double click the above data flow task.

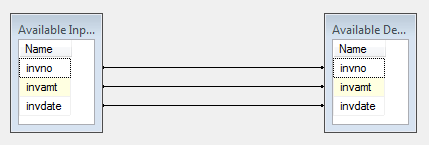
7)Drag a oledb data source task from data flow sources section. Rename as “Staging Table” Double click and point it to the sql server database where staginginvoices table is created as shown below. Click ok



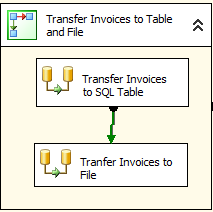
8)Drag a Sql server destination task. Rename it “Invoices Table”. Connect the oledb datasource task to the destination task. Double click destination task and point it to invoices table created above.



9)Go to mappings tab and map the corresponding columns. Click ok



10) In control flow, drag another data flow task in the sequence container, below the earlier data flow task and connect them. Rename this task as “Transfer Invoices to File”.



11)Double click above data flow task to invoke data flow designer,

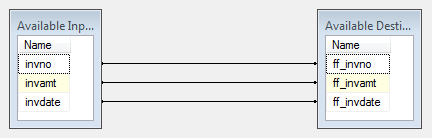
12)Drag a oledb data source and point it to staginginvoices.

13)Drag a flat file destination, rename it “Invoices Flat File”.

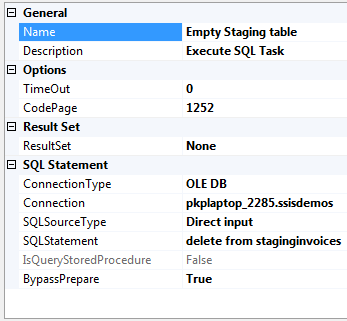
14)Connect the oledb source of step 12 to it. double click flat file destination and through a new flat file connection point it to a file in “In” folder of step 1.

**For eg: C:\Users\jogajitm\Desktop\Project 1\Folders\Demo5\In\invoices.txt**

15) Set the mappings as below:



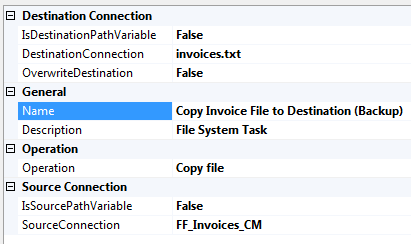
16) Back to control flow, drag a Execute SQL Task outside sequence container, rename as “Empty Staging table”. Double click it and configure it as below:



**Note: Connection: is the connection manager which points to staginginvoices sql server database.**

17)Drag a File System task on to control flow and rename as “Copy Invoice File to Destination (Backup)”.

18)Double click the above task and configure as below:



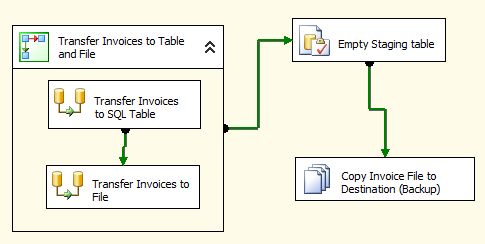
**Note:**

* **Destination connection** is a flat file connection manager which points to some file in “Out” folder created in step 1.

**For eg: C:\Users\jogajitm\Desktop\Project 1\Folders\Demo5\Out\invoices.txt**

* **Source Connection** is the flat file manager created above in step 14 which points to the file in “In” folder
* The **operation property** is set to Copy File.

19)Connect the control flow tasks as shown below



**Configure Transaction Support:**

20)Select the sequence container task, and set the transaction option of the task to “required”.

21)Set the transaction option of both the data flow tasks within sequence container to “supported”

**Configure Checkpoint Support:**

22)Select the control flow designer , by clicking it on free surface.

23) Set the following properties

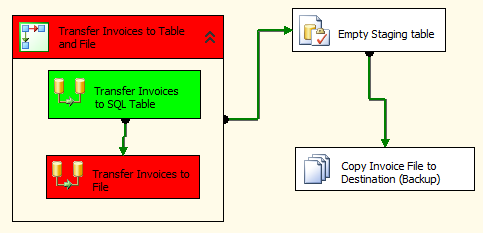
|  |  |
| --- | --- |
| **Property Name** | **Value** |
| Checkpoint File Name | To a file name in checkpoint folder created in step 1  For example something like: C:\Users\jogajitm\Desktop\Project 1\Folders\Demo5\CheckPoint\ChkPnt.txt |
| CheckPoint Usage | If Exists |
| Save Checkpoints | True |

24)Make use that for all tasks in the control flow viz: sequence container, sub tasks of sequence container, execute sql task, file system task, the properties “FailPackageOnFailure” and “FailParentOnFailure” properties are true.

* Now we ready to run the package and test the features

25) Before we run delete the “In” folder for time being or rename it to something else.

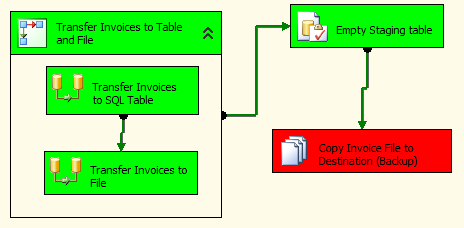
26)Run the Package, the package will fail in the sequence container because “In” folder is not found while transferring invoices details to flat file as below:



27)Go to SSMS and check that the work of first data flow task is rolled back. The data is not successfully transferred to Invoices online table from staging table. Stop the package

28)Now recreate the “In” folder, but now delete the “Out” folder from Demo5 folder for the time being.

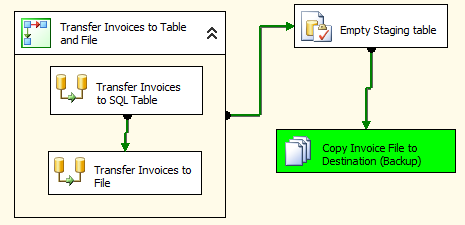
29)Now run the package, the package fails at the File System Task “Copy Invoice File to Destination (Backup)” but the earlier tasks are completed as below:



30)Stop the package, and check that:

* + Rows have been copied to invoices table from staging table.
  + The flat file is created in “In” folder
  + A checkpoint has been created in checkpoint folder

31)Stop the package and restore or recreate the “Out” folder and rerun the package.



32)The package will run directly run the last file system task and the checkpoint file will be automatically removed from its location and the file will be copied to the “Out” folder.

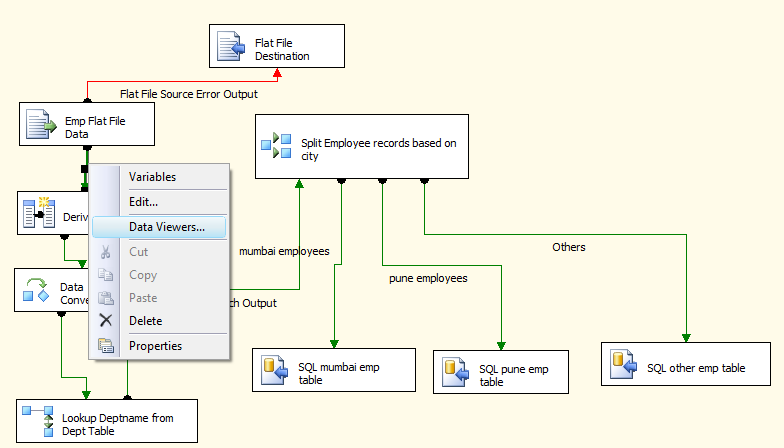
33)Stop the package.

Lab 18- Using Data Viewer

|  |  |
| --- | --- |
| **Objective** | To learn   1. How to use data viewer |
| **Lab Setup** | * SSDT tool * Create package. |

1)Open the Package “FF\_To\_DB\_With\_Transformation\_With\_Error\_Row\_Redirect.dtsx” created in Lab 4

2)Right Click the data flow path between the “Emp Flat File Data” and “Derived Column” data flow tasks.



3)Click Data Viewers

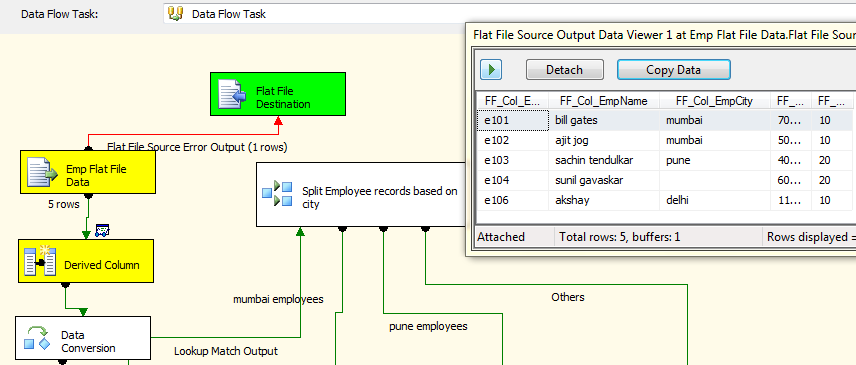
4)In the Data Viewer dialog box click “Add” button.

5)Select Grid from options. Click ok. Again click Ok.

6)delete the data from mumbaiemp, puneemp and otheremp tables.

7)Execute the package.

8)A data viewer window will automatically popup and will show the records retrieved from the flat data file. The package execution will also pause for a moment.



9)Resume the package execution click  button on data viewer window and complete the execution.

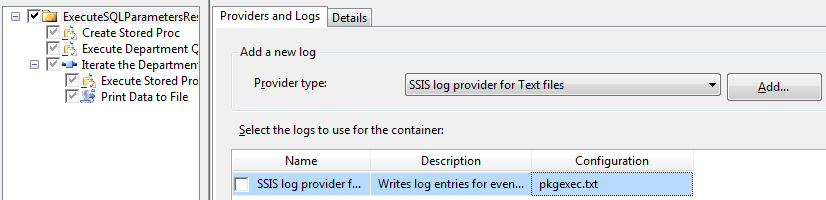
Lab 19- Using Package Logging

|  |  |
| --- | --- |
| **Objective** | To learn   1. How to enable package logging |
| **Lab Setup** | * SSDT tool * Create project. |

1)Open the Package “ExecuteSQLParametersResultSets.dtsx” of Lab 3.

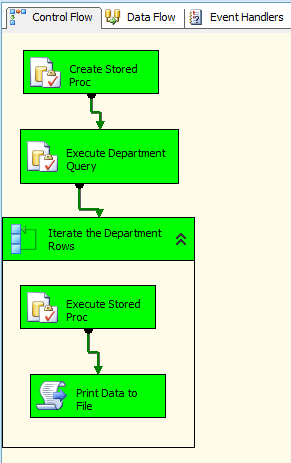
2)To enable logging for this package go to SSIS => Logging Menu option.

* + In the logging dialog box, select the root tree node of the package in left pane.
  + In right pane, select “SSIS log provider for text file” and click add. A row will be added.
  + Go to configuration column of the row, Click new connection.
  + Select Create File, and give a file path, where you want your package execution log file to be created.
  + Click ok



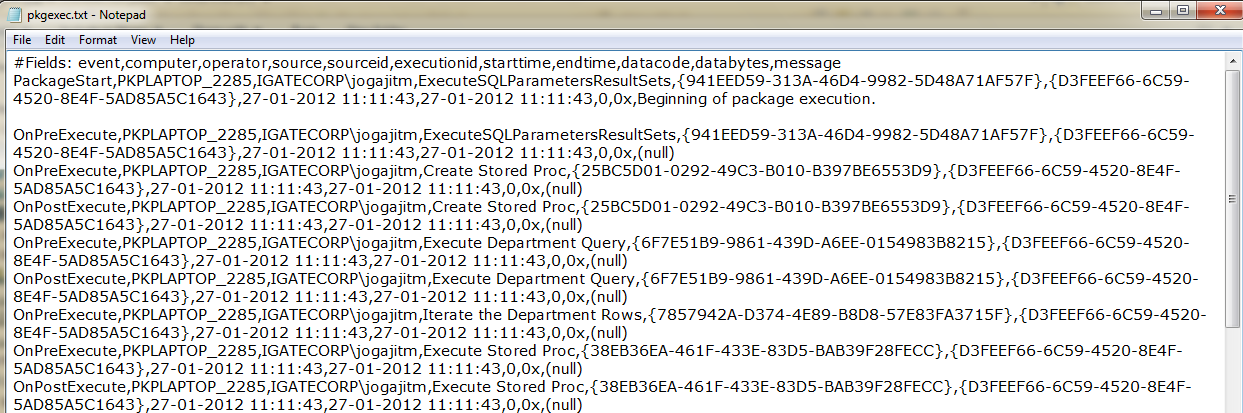
* 1. Now go to details tab, select “onpreexecute” ,“onpostexecute” and “ontaskfailed ”events.
  2. Click ok.

3)Execute the package.



4)Stop the execution on the package.

5)Open the package log file. The package file contents will look as below:



Lab20-Project deployment

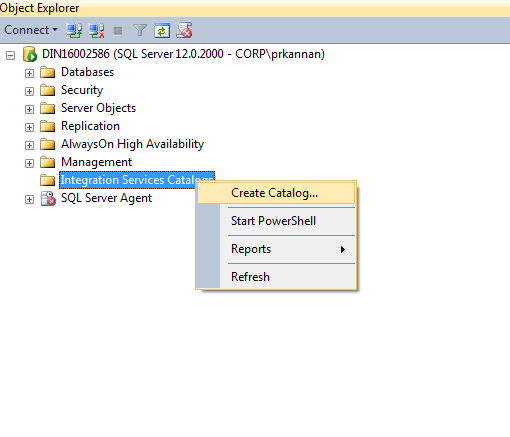
|  |  |
| --- | --- |
| **Objective** | To learn  1.How to deploy a project. |
| **Lab Setup** | * SSDT tool * Login details for connecting to the database |

**IN SSMS:**

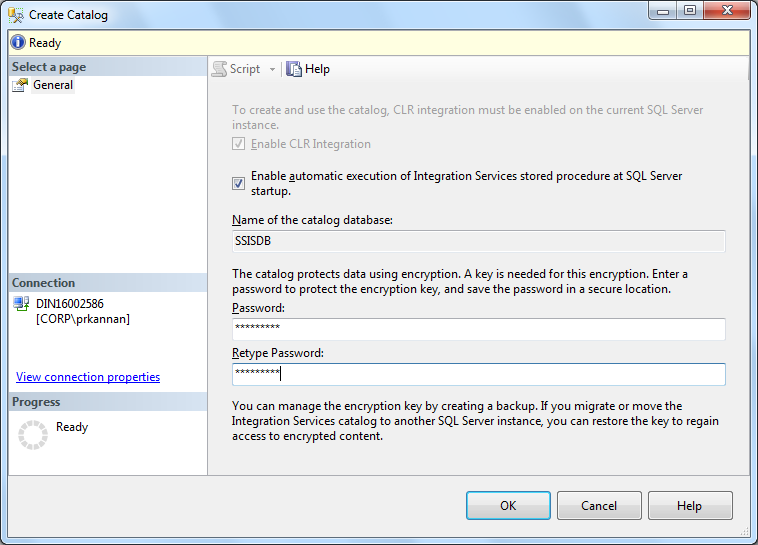
1)Before we can deploy any projects to a SQL Server instance, we first need to create the Integration Services Catalog. Think of this catalog as the container for all deployed projects, their settings and historical versions.

2)There is only one Catalog per SQL Server instance; it is represented by a separate SQL Server database called SSISDB, which contains the deployment’s versioning, settings, and even statistical performance data.

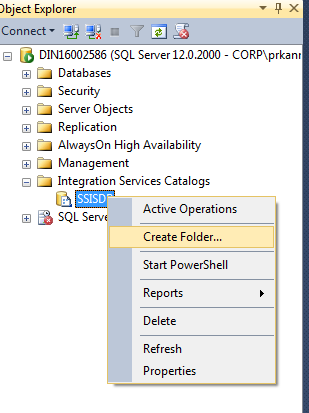
3) In order to create the catalog, we need to connect to the SQL Server instance and right-click on the ‘Integration Services Catalogs’ and click ‘Create Catalog’.



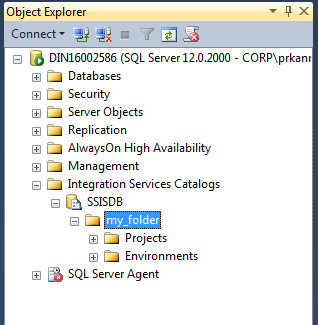
4) In the next screen we are asked for the encryption password and whether we would like to run the **Catalog. Startup**procedure every time the SQL Server instance starts (the **Catalog.startup** stored procedure fixes the status of the packages in the catalog if there were packages running when the SQL Server instance went down):



5)After we have created the Catalog, we can create a new folder under it, which will contain our first project. To create the folder, simply right-click on the SSISDB catalog and click ‘Create Folder’:

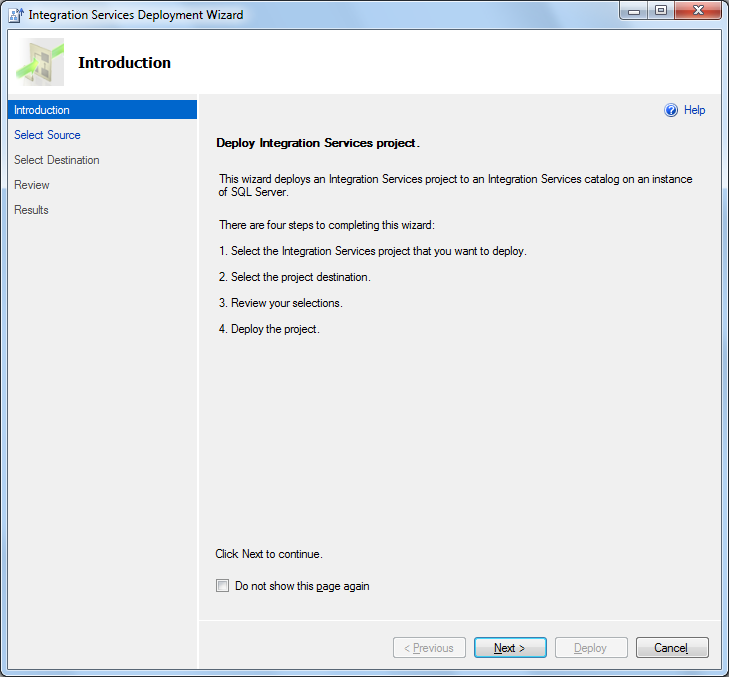


6)Under every newly-created folder we have two sub-folders: ‘Projects’ and ‘Environments’.

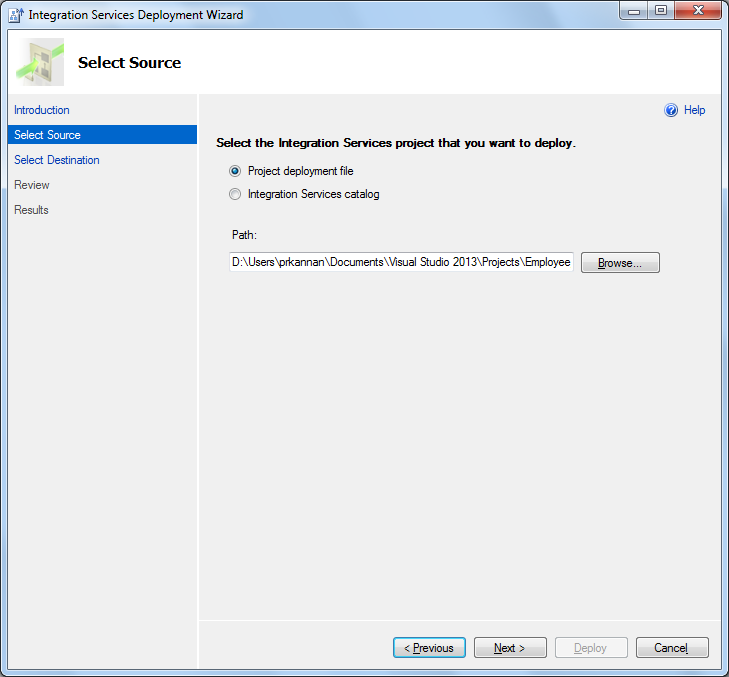


**IN SSDT:**

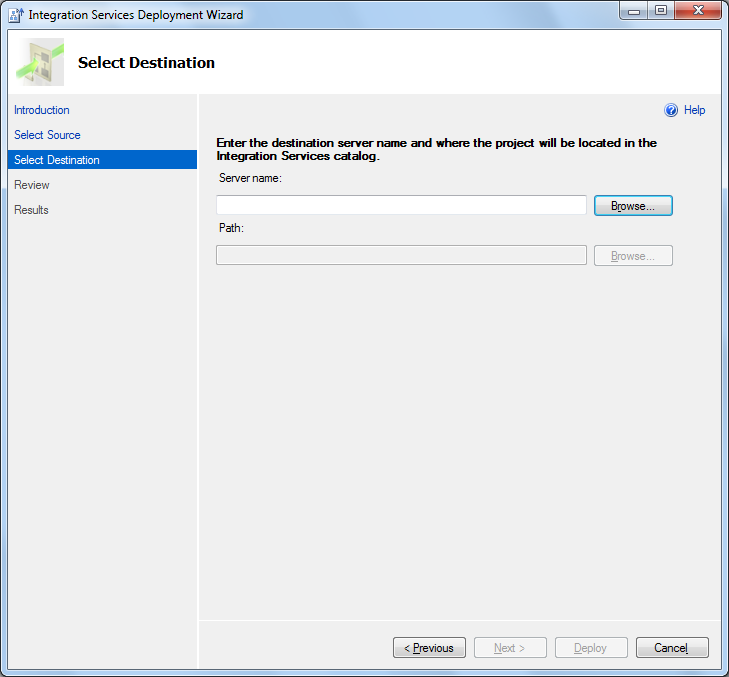
7) Now let’s deploy the project to our SQL Server instance. By right-clicking on the project solution in SSDT and clicking Deploy, we get to the following wizard:



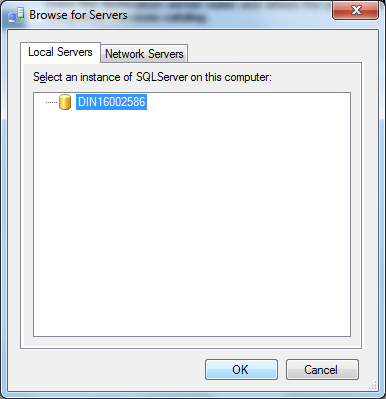
8)Select source page will appears like this:



9)Select Destination server looks like the below screenshot:

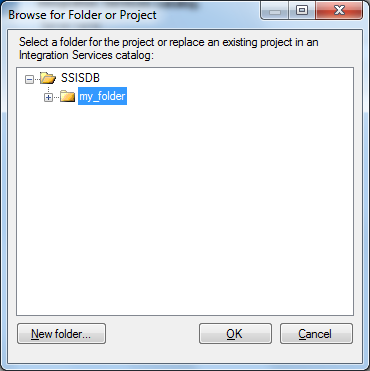


10)Click browse button to choose your destination server name like below screenshot and click ok.



11)Now browse path where the project should be deployed as given below.

12)You will get the below screen when you click the browse button.



13)Choose the folder which you have created under Integration services catalog and click ok and give next.

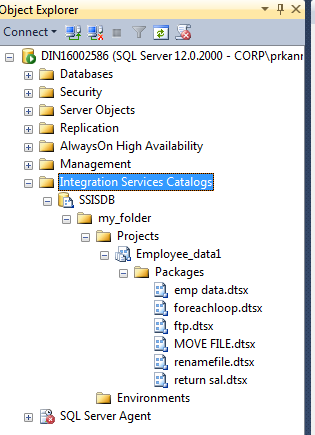
14)Check once for your source and destination and deploy.

15)If the deployment is successful you get like the below screenshot.



16)If you need, you can save the report.

17)The deployed project now looks like this:

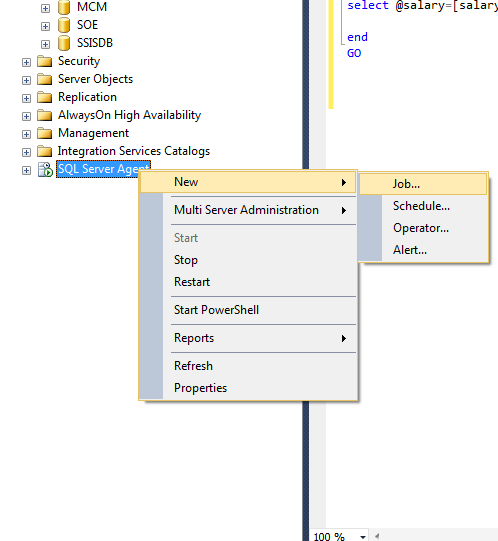


18)From this point on, we can execute the deployed package by scheduling an SQL Server Agent Job and using the package in it.

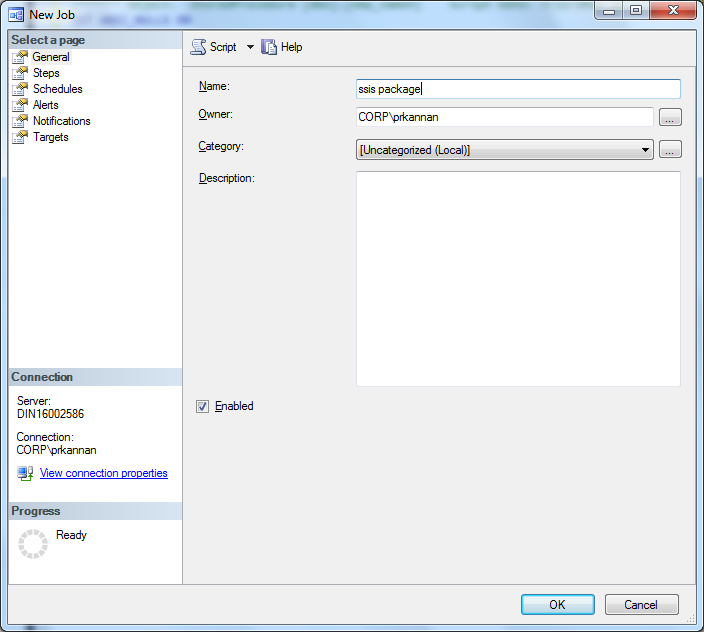
Lab21-Package Execution using SQL Server Agent

|  |  |
| --- | --- |
| **Objective** | Package execution from SQL Server Agent |
| **Lab Setup** | Login details for connecting to the database |

1)To execute a package from SSMS , right click on the SQL Server Agent which will be under your server as per below.

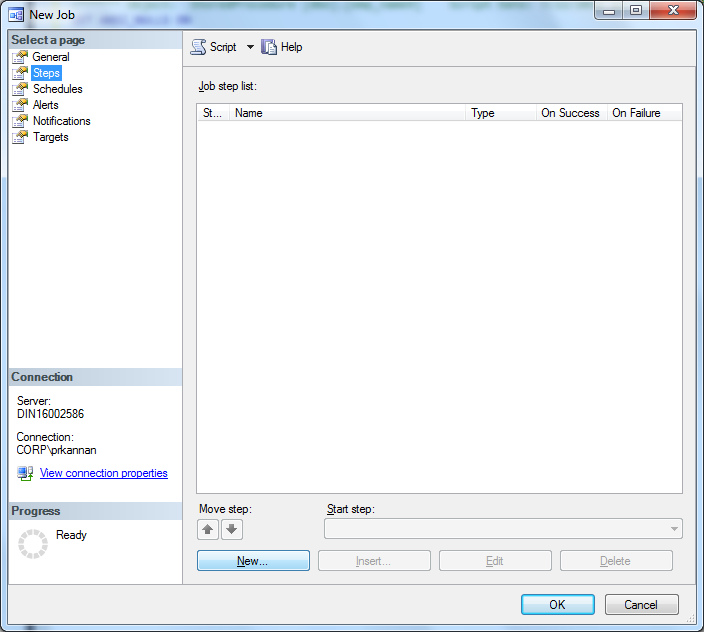


2)Click on new job you will get the below screen. Give name for your job.

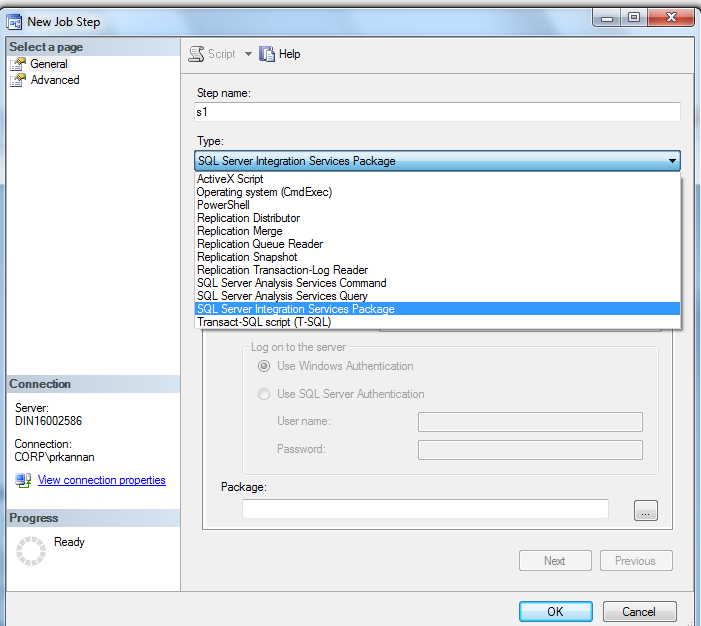


3)Click on the “Steps” under Select a page. You will get a screen like below where you should choose

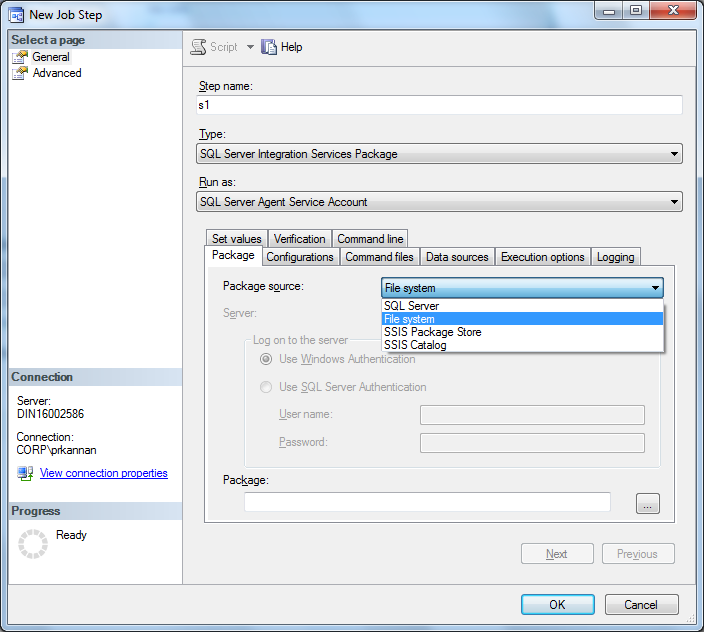
the  option.



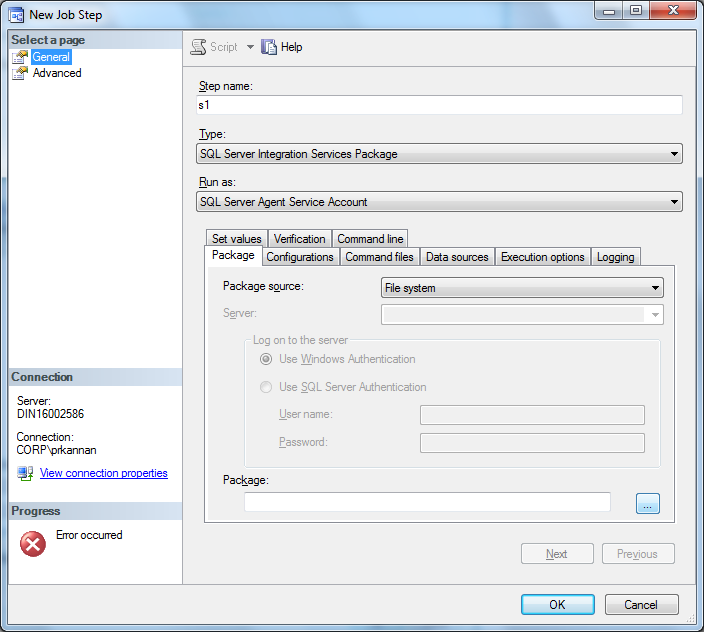
4) After selecting new , you will be prompted to this page. Give a name to your step and choose SQL Server Integration Services under type.

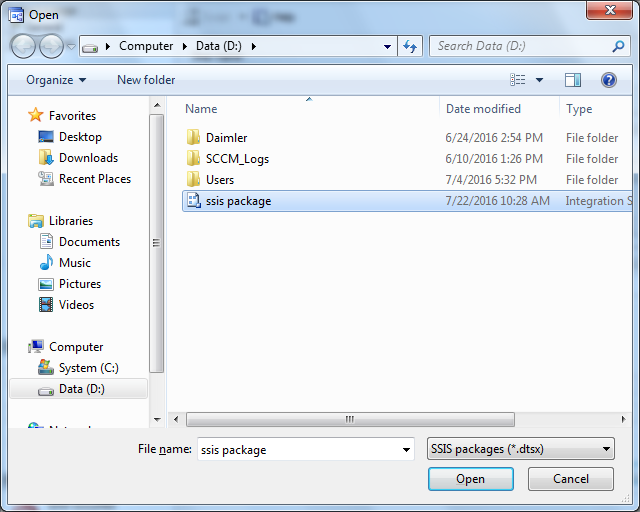


5)Select the package source as File System.

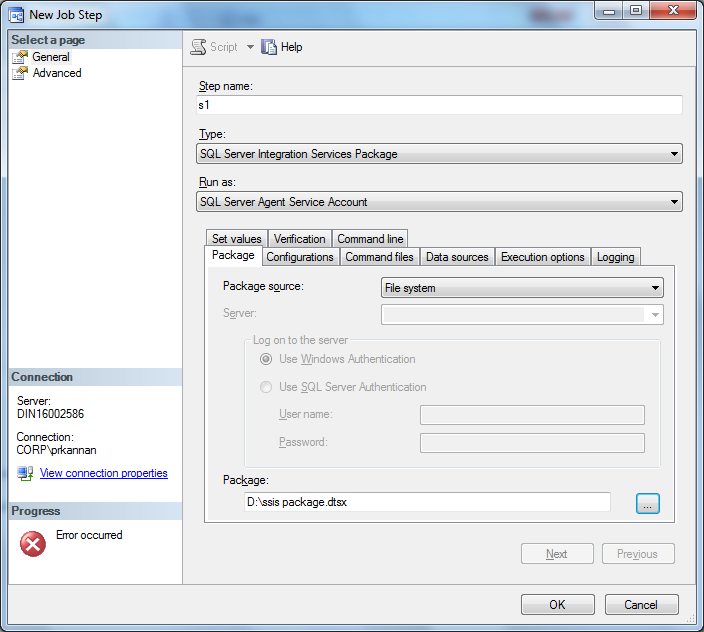


6)Select the package which you want to execute using the browse button as per below.



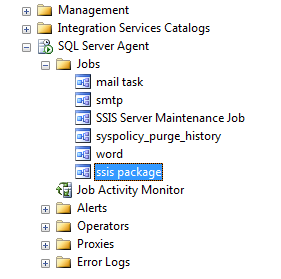


7)After selecting the package click open.

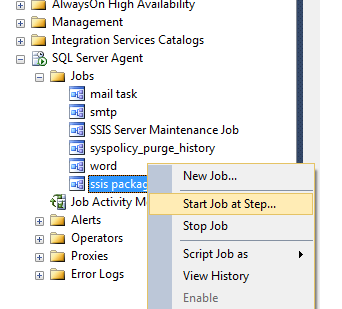


8)Then click ok in the new job step page and new job page.

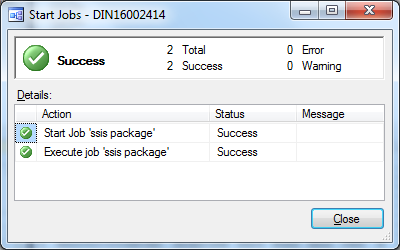
9)Now expand the SQL Server Agent and now you can see your job under server agent.



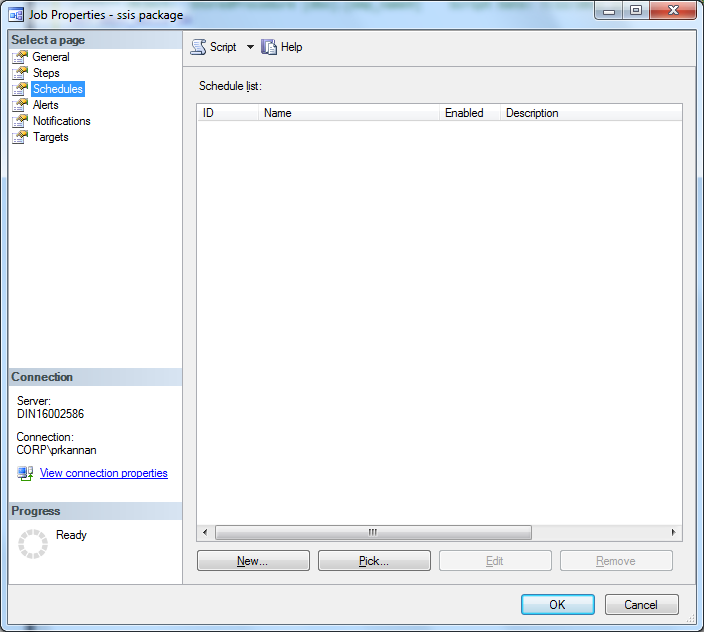
10)Now right click on the package on click Start job at Step.



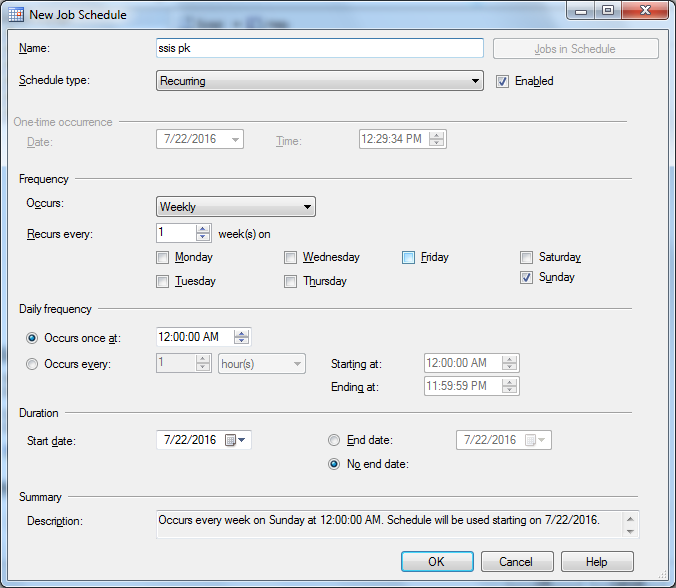
11)After package execution successfully you will like the below screenshot.



12)After executing your package by sql server agent you can schedule your package to run daily on at a frequency you wish. To schedule the job ,double click the package under sql server agent you will get the below screen.



13)Click new at present at the bottom of the screen. You will get the below screen.



14)Here you can specify the schedule type, frequency and daily frequency for executing your job.