Novedea Systems Inc, 1750 N Collins Blvd Suite 212 Richardson, TX- 75080 Ph: 972-918-0900 Fax: 972-918-0901

Confidential –Do not distribute.

1003 Application Processing Solution Document

ETL, Data ware & Analytics

6/1/2013

C:\Novedea\logo\logo.gif

Chaitanya Sagam

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Notes |
| 5/22/2013 | Initial | Chaitanya Sagam | Created initial document version |
| 6/01/2013 | V2 | Ram Kotamaraja | Added TOC and cover pages |
| 6/12/2013 | V3 | Chaitanya Sagam | Added Script task solution in story 2 Excel packages step 1 |
| 6/14/2013 | V4 | Chaitanya Sagam | Added Solution from Story 4 and 5. |

Table of Contents

[Revision History 1](#_Toc357859450)

[Solution Documentation 3](#_Toc357859451)

[Story 1: Gather 1003 Mortgage application input data 3](#_Toc357859452)

[Story 2: Read input data and Story 3: Insert data into staging tables 13](#_Toc357859453)

Story 4: Validating and Error handling Staging Database Story 5:Inserting Data Into ODS…………..43

# Solution Documentation

## Story 1: Gather 1003 Mortgage application input data

We need to create 3 types of Input Sources:

1. Excel Source
2. DB Source
3. XML Source
4. **Excel Source:-**

We need to create Excel Source based on the 1003 Application form of CMG. Here we need identify the required fields from the Application form, So that we can include the columns in the Excel document.

The Excel Source should contain the following fields:

|  |  |
| --- | --- |
| **Field** | **Datatype** |
| Borrower FirstName | Varchar |
| Borrower LastName | Varchar |
| Borrower Email | Varchar |
| Social Security number | Number |
| Home Phone | Number |
| Cell Phone | Number |
| Marital Status | Varchar |
| Date of Birth | Date |
| Current Street Address | Varchar |
| City | Varchar |
| State | Varchar |
| Zip | Number |
| Years at this Address | Number |
| Base Income(Monthly) | Int |
| Bonuses | Int |
| Commission | Int |
| Other Income | Int |
| Checking | Int |
| Savings | Int |
| Retirement Funds | Int |
| Mutual Funds | Int |
| Referral Name | Varchar |
| Real Estate Agent Name | Varchar |
| Real Estate Agent Phone | Number |
| Real Estate Agent Email | Varchar |
| Rent or Own | Varchar |
| Purpose of Loan | Varchar |
| Property Usage | Varchar |
| Loan Amount | Int |
| Purchase Price | Int |
| Number of Units | Number |
| Property Address:City | Varchar |
| Property State | Varchar |
| Property zip | Varchar |
| Sex | Varchar |
| Ethnicity | Varchar |
| Race | Varchar |

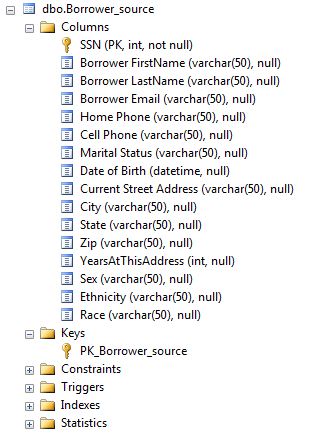
**Sample Data :- (** To get full data double click on the below table)

****

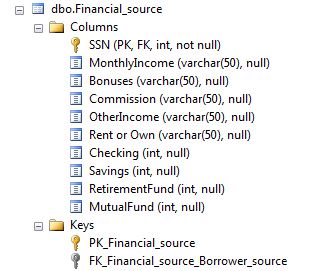
1. **DB Source :-**

We need to create the OLTP database, populate it with dummy data. The Database Structure for the input DB source is as follows:

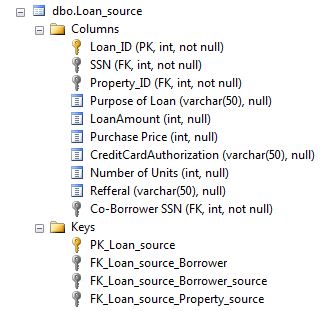
**Borrower Table:**

****

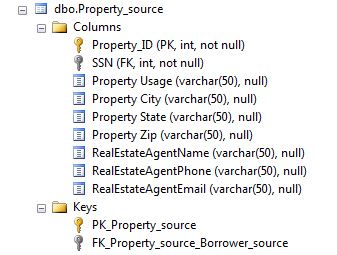
**Financial Table:**

****

**Loan Table:**

****

**Property Table:**

****

**SQL SCRIPT:-**

**Borrower:**

CREATE TABLE [dbo].[Borrower\_details](

[SSN] [int] NOT NULL,

[Borrower FirstName] [varchar](50) NULL,

[Borrower LastName] [varchar](50) NULL,

[Borrower Email] [varchar](50) NULL,

[Home Phone] [varchar](50) NULL,

[Cell Phone] [varchar](50) NULL,

[Marital Status] [varchar](50) NULL,

[Date of Birth] [datetime] NULL,

[Current Street Address] [varchar](50) NULL,

[City] [varchar](50) NULL,

[State] [varchar](50) NULL,

[Zip] [varchar](50) NULL,

[YearsAtThisAddress] [int] NULL,

[Sex] [varchar](50) NULL,

[Ethnicity] [varchar](50) NULL,

[Race] [varchar](50) NULL

)

**Financial:**

CREATE TABLE [dbo].[Financial\_details](

[SSN] [int] NOT NULL,

[MonthlyIncome] [varchar](50) NULL,

[Bonuses] [varchar](50) NULL,

[Commission] [varchar](50) NULL,

[OtherIncome] [varchar](50) NULL,

[Rent or Own] [varchar](50) NULL,

[Checking] [int] NULL,

[Savings] [int] NULL,

[RetirementFund] [int] NULL,

[MutualFund] [int] NULL

)

**Loan:**

CREATE TABLE [dbo].[Loan\_details](

[Loan\_ID] [int] NOT NULL,

[SSN] [int] NOT NULL,

[Property\_ID] [int] NOT NULL,

[Purpose of Loan] [varchar](50) NULL,

[LoanAmount] [int] NULL,

[Purchase Price] [int] NULL,

[CreditCardAuthorization] [varchar](50) NULL,

[Number of Units] [int] NULL,

[Refferal] [varchar](50) NULL,

[Co-Borrower SSN] [int] NOT NULL

)

**Property:**

CREATE TABLE [dbo].[Property\_details](

[Property\_ID] [int] NOT NULL,

[SSN] [int] NOT NULL,

[Property Usage] [varchar](50) NULL,

[Property City] [varchar](50) NULL,

[Property State] [varchar](50) NULL,

[Property Zip] [int] NULL,

[RealEstateAgentName] [varchar](50) NULL,

[RealEstateAgentPhone] [varchar](50) NULL,

[RealEstateAgentEmail] [varchar](50) NULL

)

**Constraints and Relationships:-**  After Creating the above tables, i.e Executing the above script add the following relations to your Database.

Borrower: Primary key on SSN

Financial: Primary Key on SSN and FK referencing Borrower table

Loan: PK on Loan\_ID, FK on SSN ref Borrower and Property ID ref Property table and Co borrower SSN ref borrower table.

Property: PK on Property ID and FK on SSN ref borrower.

**NOTE:-**

After creating the tables and adding relationships, we need to insert at least 10 rows in each table.

1. **XML Source:-**

We should create the input XML file with some dummy Data.

Sample XML File:

**<?xml version="1.0" encoding="UTF-8" ?>**

**<mortage>**

**<Borrowers>**

**<Borrower>**

**<SSN>67656473</SSN>**

**<BorrowerFirstName>Chaitu</BorrowerFirstName>**

**<BorrowerLastName>s</BorrowerLastName>**

**<BorrowerEmail>a@b.com</BorrowerEmail>**

**<HomePhone>2103654221</HomePhone>**

**<CellPhone>2145689563</CellPhone>**

**<MaritalStatus>Not Married</MaritalStatus>**

**<DateofBirth>03/06/91</DateofBirth>**

**<CurrentCityStreetAddress>MacallumBlvd</CurrentCityStreetAddress>**

**<city>Plano</city>**

**<State>TX</State>**

**<Zip>254123</Zip>**

**<YearsAtThisAdress>1</YearsAtThisAdress>**

**<Sex>Male</Sex>**

**<Ethnicity>Asian</Ethnicity>**

**<Race>Indian</Race>**

**</Borrower>**

**<CoBorrower>**

**<SSN>67656475</SSN>**

**<BorrowerFirstName>Sri</BorrowerFirstName>**

**<BorrowerLastName>G</BorrowerLastName>**

**<BorrowerEmail>c@b.com</BorrowerEmail>**

**<HomePhone>21036542267</HomePhone>**

**<CellPhone>21456895687</CellPhone>**

**<MaritalStatus>Not Married</MaritalStatus>**

**<DateofBirth>03/06/86</DateofBirth>**

**<CurrentCityStreetAddress>8970 Park</CurrentCityStreetAddress>**

**<State>TX</State>**

**<Zip>254127</Zip>**

**<YearsAtThisAdress>1</YearsAtThisAdress>**

**<Sex>Male</Sex>**

**<Ethnicity>Asian</Ethnicity>**

**<Race>Indian</Race>**

**</CoBorrower>**

**</Borrowers>**

**<financial\_detail>**

**<SSN>67656473</SSN>**

**<MonthlyIncome>10000</MonthlyIncome>**

**<Bonuses>500</Bonuses>**

**<Commission>100</Commission>**

**<OtherIncome>200</OtherIncome>**

**<RentorOwn>Rent</RentorOwn>**

**<Checking>5000</Checking>**

**<Savings>6000</Savings>**

**<RetirementFund>10000</RetirementFund>**

**<MutualFund>5000</MutualFund>**

**</financial\_detail>**

**<Loan\_detail>**

**<Loan\_ID>1239</Loan\_ID>**

**<SSN>67656473</SSN>**

**<property\_ID>333</property\_ID>**

**<PurposeofLoan>Education</PurposeofLoan>**

**<LoanAmount>10000</LoanAmount>**

**<PurchasePrice>10000</PurchasePrice>**

**<NumberofUnits>1</NumberofUnits>**

**<CreditCardAuthorization>Agree</CreditCardAuthorization>**

**<Refferal>NULL</Refferal>**

**<CoBorrowerSSN>67656473</CoBorrowerSSN>**

**</Loan\_detail>**

**<property\_detail>**

**<SSN>67656473</SSN>**

**<propertyID>333</propertyID>**

**<PropertyUsage>Land</PropertyUsage>**

**<PropertyCity>Ricardson</PropertyCity>**

**<PropertyState>TX</PropertyState>**

**<PropertyZip>257841</PropertyZip>**

**<RealEstateAgentName>NULL</RealEstateAgentName>**

**<RealEstateAgentPhone>NULL</RealEstateAgentPhone>**

**<RealEstateAgentEmail>NULL</RealEstateAgentEmail>**

**</property\_detail>**

**</mortage>**

## Story 2: Read input data and Story 3: Insert data into staging tables

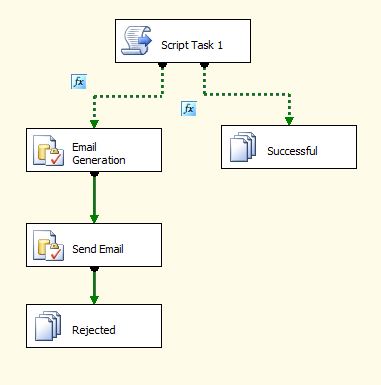
The End Result of these two stories is we need to create SSIS packages through which we can read the data from different types of input sources and Insert them in the staging tables.

**1) Excel packages :-** (Creating Excel packages)

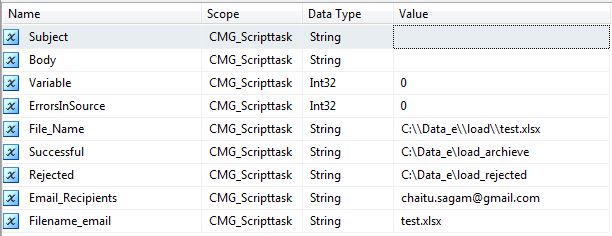
**Step 1:**

For Excel source, we use script task to validate excel to make sure all the required columns are there. As the data in excel is manually entered one there might be a chance to get bad data or Incomplete data.

Initially drag the following components in Control flow.

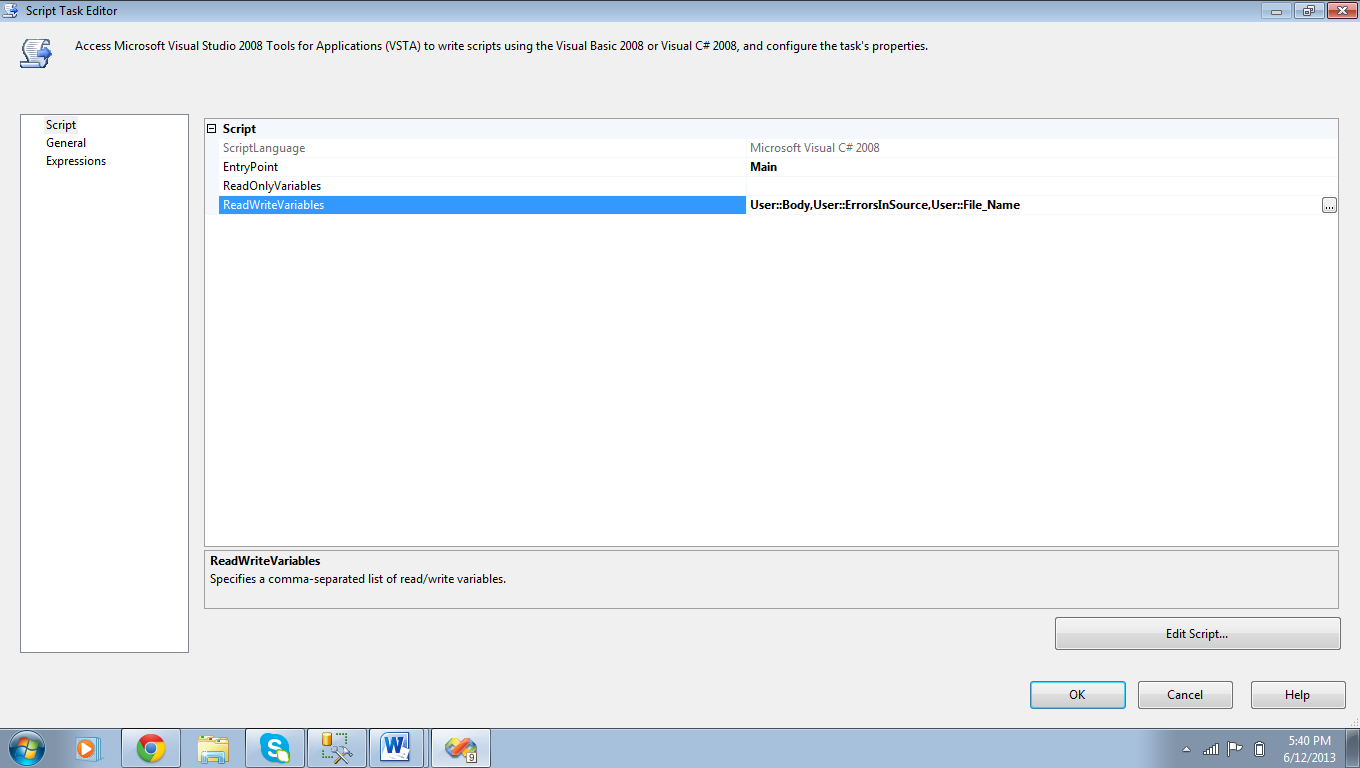


Before editing each component we need to configure the following variables. These variables are used in various contexts of the each component.



*Script Task:-*

In script task, initially add the following read-write variables also configure read-only variables if needed.



Then click on edit script and add the following code,

using System;

using System.Data;

using Microsoft.SqlServer.Dts.Runtime;

using System.Windows.Forms;

namespace ST\_fcd56ec20b8646dfb3261487b129ff89.csproj

{

[System.AddIn.AddIn("ScriptMain", Version = "1.0", Publisher = "", Description = "")]

public partial class ScriptMain : Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase

{

#region VSTA generated code

enum ScriptResults

{

Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,

Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure

};

#endregion

/\*

The execution engine calls this method when the task executes.

To access the object model, use the Dts property. Connections, variables, events,

and logging features are available as members of the Dts property as shown in the following examples.

To reference a variable, call Dts.Variables["MyCaseSensitiveVariableName"].Value;

To post a log entry, call Dts.Log("This is my log text", 999, null);

To fire an event, call Dts.Events.FireInformation(99, "test", "hit the help message", "", 0, true);

To use the connections collection use something like the following:

ConnectionManager cm = Dts.Connections.Add("OLEDB");

cm.ConnectionString = "Data Source=localhost;Initial Catalog=AdventureWorks;Provider=SQLNCLI10;Integrated Security=SSPI;Auto Translate=False;";

Before returning from this method, set the value of Dts.TaskResult to indicate success or failure.

To open Help, press F1.

\*/

public void Main()

{

// TODO: Add your code here

int error\_count = 0;

string error\_detail = "";

string excelFilePath = Dts.Variables["User::File\_Name"].Value.ToString();

string strCn = "Provider=Microsoft.ACE.OLEDB.12.0;" + "Data Source=" + excelFilePath + ";Excel 12.0;HDR=YES";

DataTable dtTables = new DataTable(); ;

System.Data.OleDb.OleDbConnection cn = new System.Data.OleDb.OleDbConnection(strCn);

cn.Open();

System.Data.OleDb.OleDbCommand command = new System.Data.OleDb.OleDbCommand();

command.Connection = cn;

command.CommandText = "SELECT \* FROM [Sheet1$]";

System.Data.OleDb.OleDbDataAdapter adapter = new System.Data.OleDb.OleDbDataAdapter();

adapter.SelectCommand = command;

adapter.Fill(dtTables);

string str = string.Empty;

if (dtTables.Columns.Count != 44)

{

error\_count += 1;

error\_detail = "Mis-match in the columns count" + "\r\n";

}

if (!dtTables.Columns.Contains("SSN"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name SSN is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Borrower FirstName"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Borrower FirstName is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Borrower LastName"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Borrower LastName is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Borrower Email"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Email is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Home Phone"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Home Phone is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Cell Phone"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Cell Phone is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Marital Status"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Marital Status is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Date of Birth"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Date of Birth is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Current Street Address"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Current Street Address is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("City"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name City is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("State"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name State is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Zip"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Zip is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("YearsAtThisAddress"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name YearsAtThisAddress is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("MonthlyIncome"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name MonthlyIncome is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Bonuses"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Bonuses is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Commission"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Commission is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("OtherIncome"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name OtherIncome is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Rent or Own"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Rent or Own is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Purpose of Loan"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Purpose of Loan is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Property Usage"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Property Usage is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("LoanAmount"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name LoanAmount is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Purchase Price"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Purchase Price is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Number of Units"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Number of Units is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Property City"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Property City is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Property State"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Property State is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Property Zip"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Property Zip is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Sex"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Sex is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Ethnicity"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Ethnicity is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Race"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Race is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("CreditCardAuthorization"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name CreditCardAuthorization is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Checking"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Checking is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Savings"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Savings is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("RetirementFund"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name RetirementFund is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("MutualFund"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name MutualFund is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Referral"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Referral is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("RealEstateAgentName"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name RealEstateAgentName is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("RealEstateAgentPhone"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name RealEstateAgentPhone is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("RealEstateAgentEmail"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name RealEstateAgentEmail is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Loan\_ID"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Loan\_ID is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Property\_ID"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Property\_ID is mis-spelled or not exists in the file. " + "\r\n";

}

if (!dtTables.Columns.Contains("Co-Borrower SSN"))

{

error\_count += 1;

error\_detail = error\_detail + " Column name Co-Borrower SSN is mis-spelled or not exists in the file. " + "\r\n";

}

Dts.Variables["User::ErrorsInSource"].Value = error\_count;

Dts.Variables["User::Body"].Value = error\_detail;

cn.Close();

}

// Dts.TaskResult = (int)ScriptResults.Success;

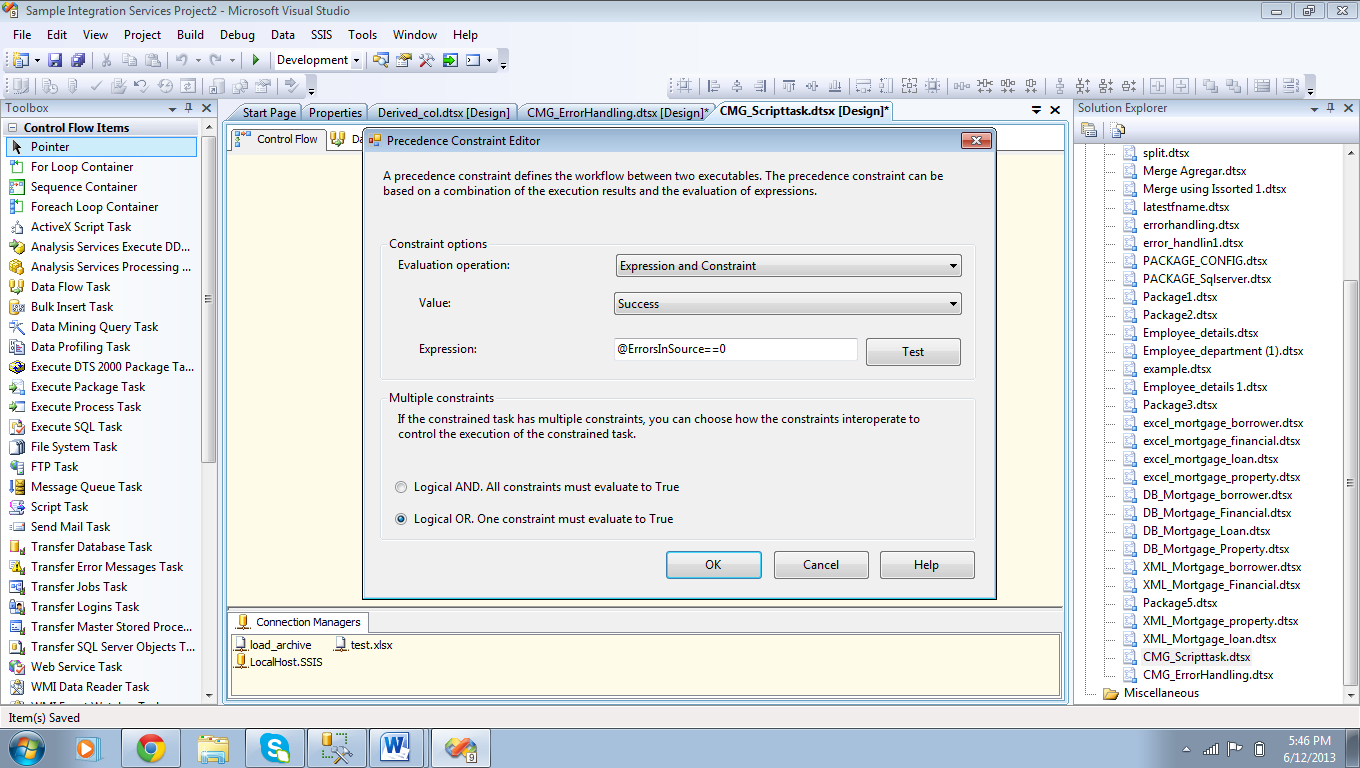
}

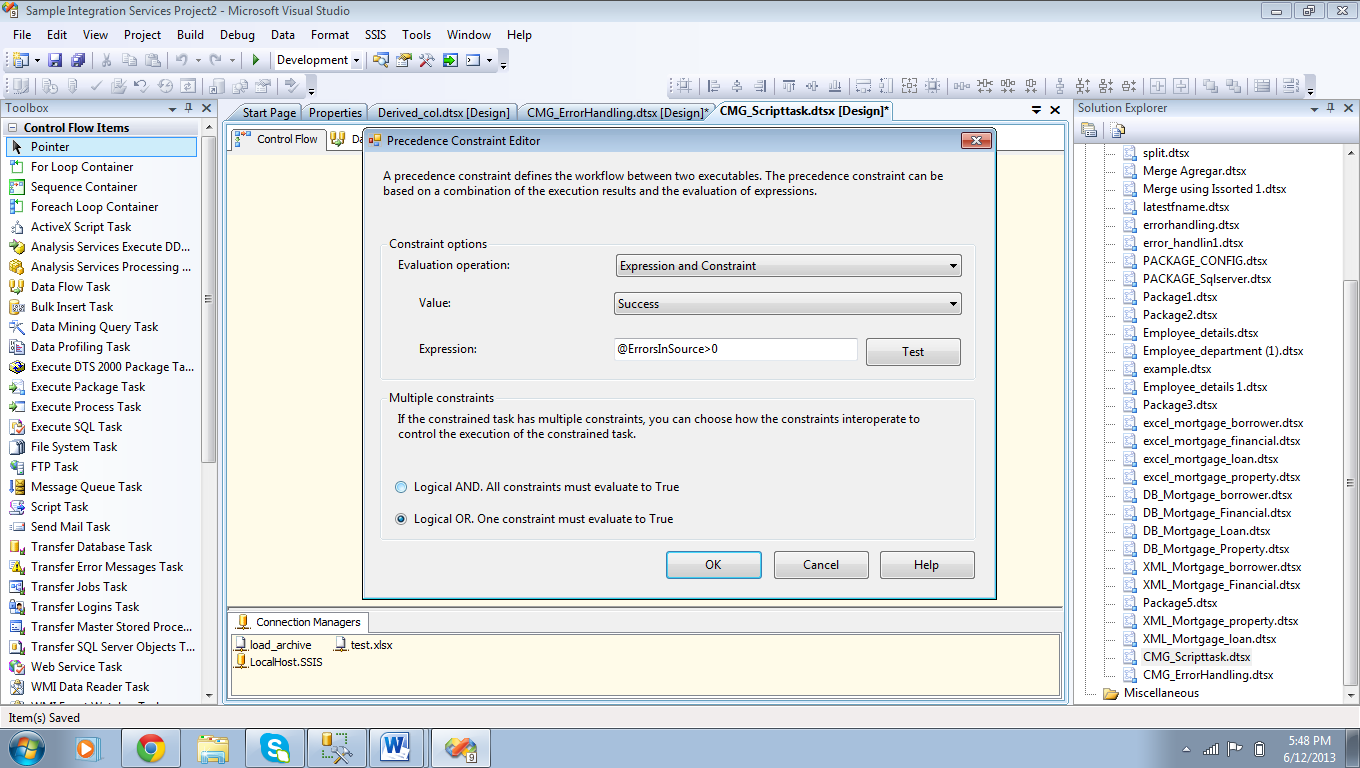
}

Next we need to give to precedent constraints,

1. If all the fields are present sent file as successful.
2. Else we need intimate the loan officer by sending email through send mail task

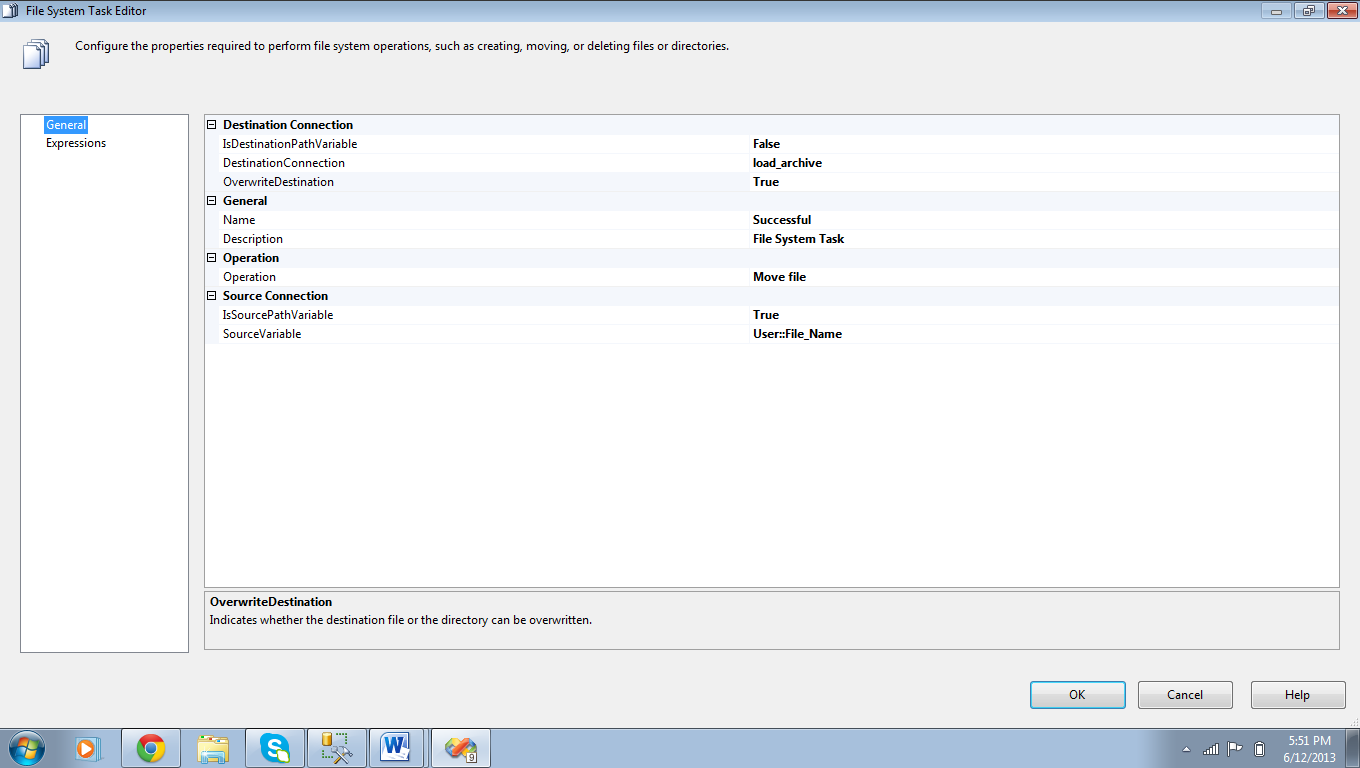
Adding the two precedent constraints as shown,





*File System Task on Successful:*

In this file system task, do the modifications to each tab as shown in following figure.



*Execute SQL Task for Email Generation:*

Next, go to execute SQL task of email generation and add the following SQL code in *SQL STATEMENT.*

declare @SourceFileName varchar(250)

declare @Body varchar(8000)

declare @Subject varchar (100)

set @SourceFileName = ?

set @Body=?

set @Body = '"' + @SourceFileName + '" is not in the correct format.Please check for the errors'

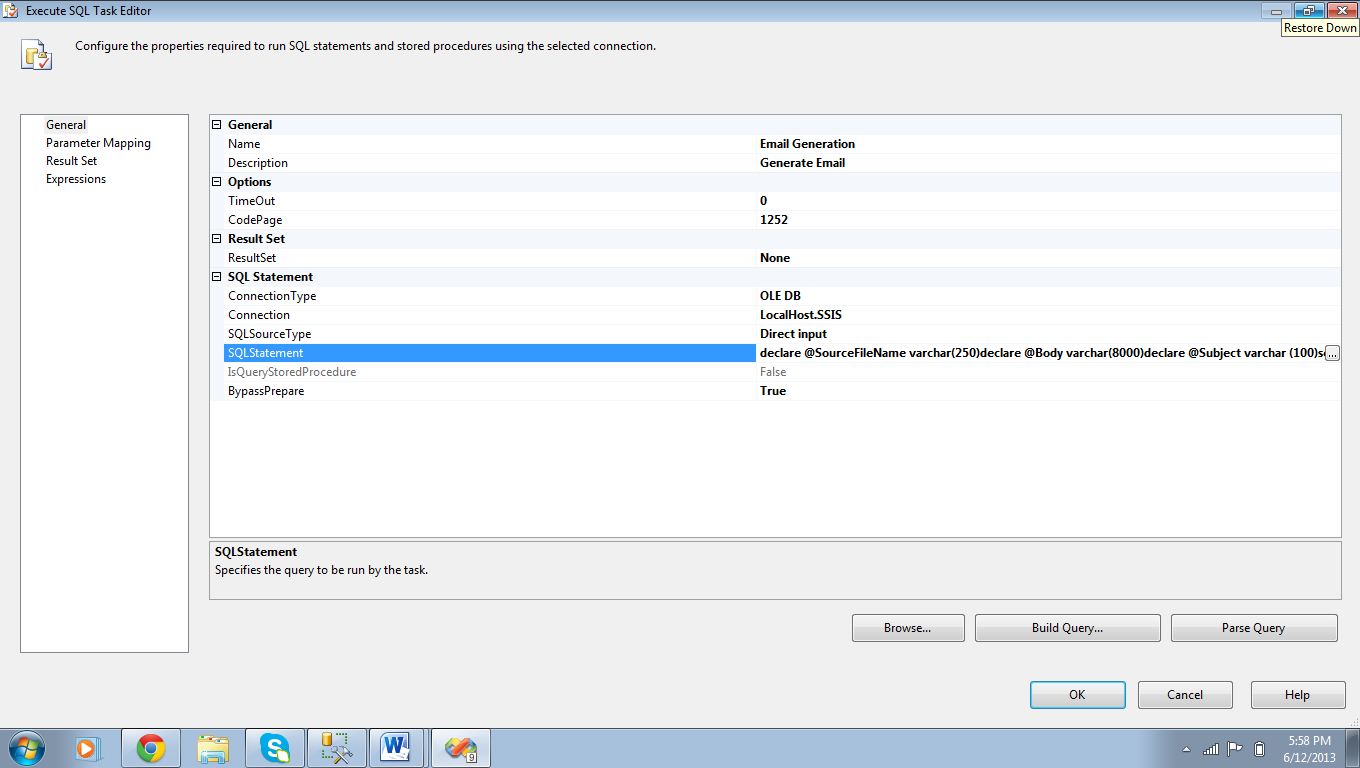
+ Char(13)+ Char(13)+

@Bodya

set @Subject = ' Check for errors in the file.'

*select @Body as Body , @Subject as Subject*

Enter the above code in the below highlighted tab.



*Execute SQL Task for SEND EMAIL:*

Next, go to execute SQL task of email generation and add the following SQL code in *SQL STATEMENT.*

DECLARE @recipients varchar(2000)

DECLARE @filename varchar(200)

DECLARE @subject varchar(2000)

DECLARE @body varchar(8000)

SET @recipients = ISNULL(?, ' ')

SET @subject = ISNULL(?, ' ')

SET @body = ISNULL(?, ' ')

SET @filename = ISNULL(?, ' ')

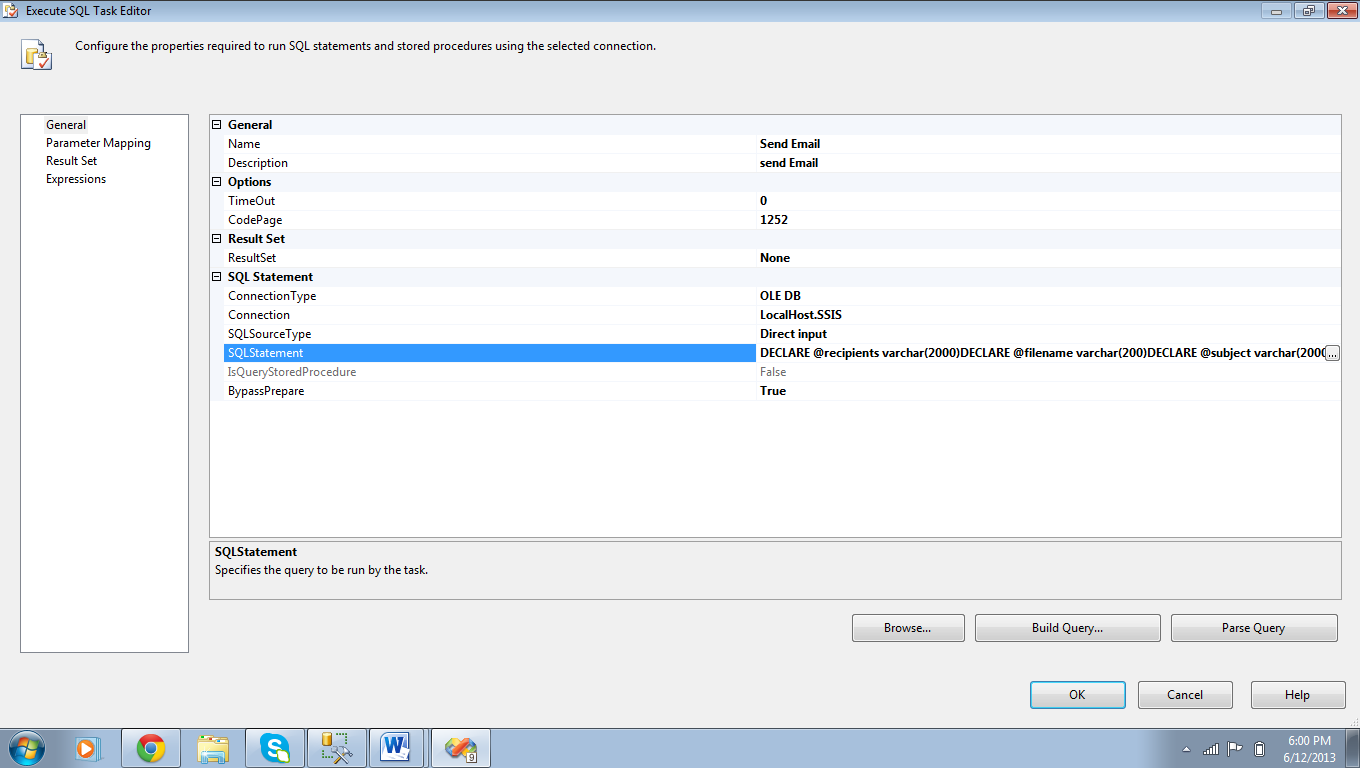
EXEC msdb.dbo.sp\_send\_dbmail

@recipients=@recipients,

@subject=@subject ,

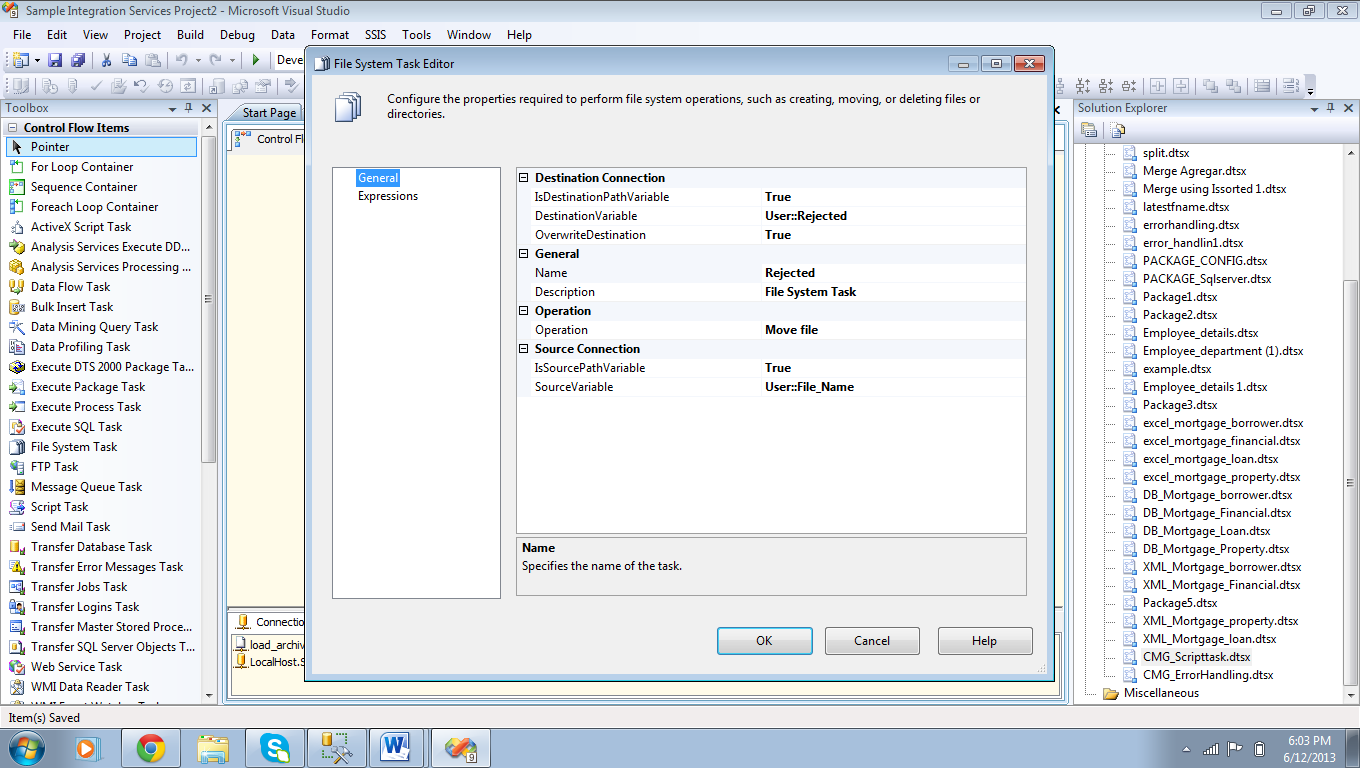
@body=@body

Enter the above code in the below highlighted tab.



*File System task for Rejected:*

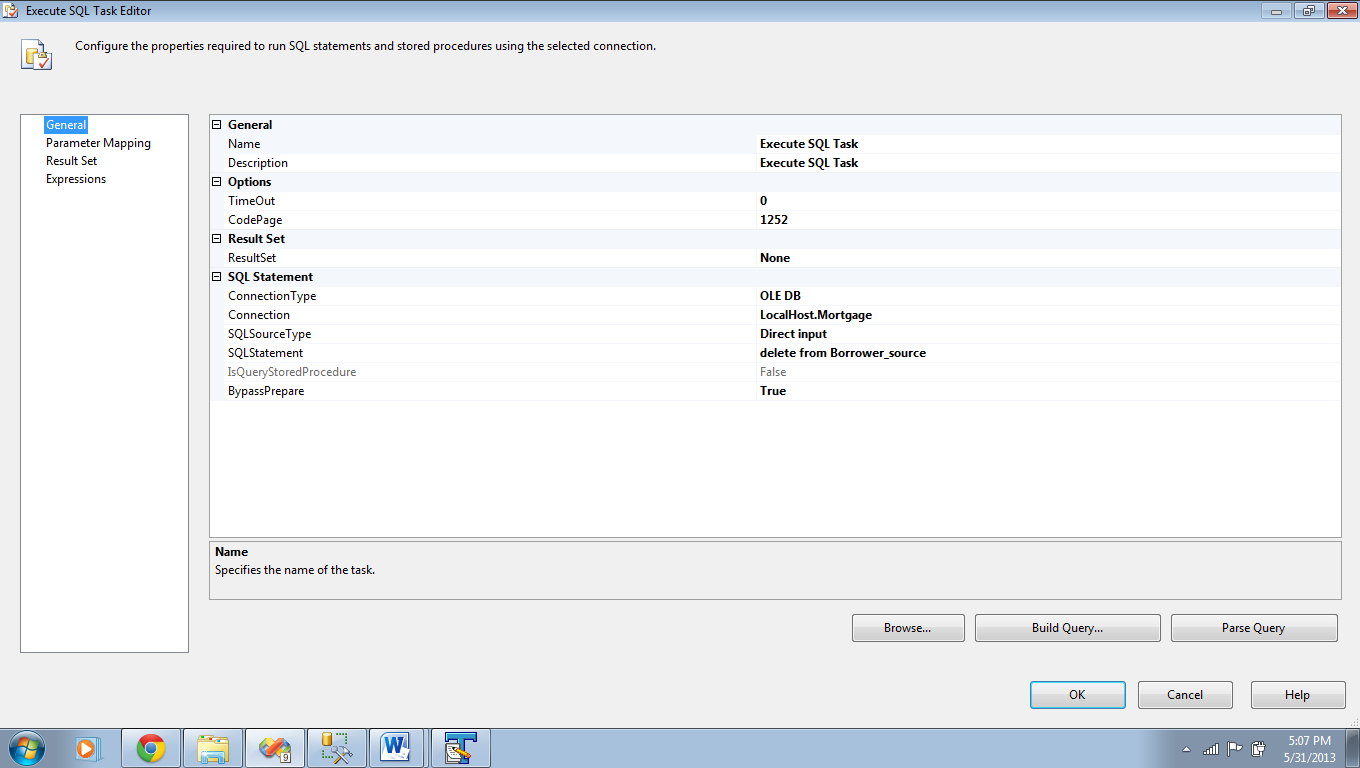
In this file system task, do the modifications to each tab as shown in following figure.



**Step 2:**

Go to Integration services Project and create a new package. Now, in the package drag Execute SQL task from the tool box.

Double click on it to edit



Go to general tab and do the following modifications.

Connection type: “OLE DB”

Connection: Select the local host database in which your staging table is located.

SQL Statement: “Delete from Borrower\_source(your table name)”

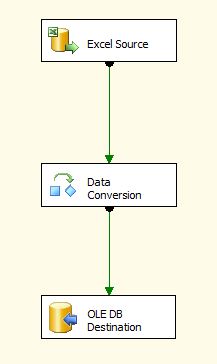
Click ok.

**Step 3:**

Now Drag the Data flow task and connect execute SQL task to Data flow task.

Double click on Data flow task.

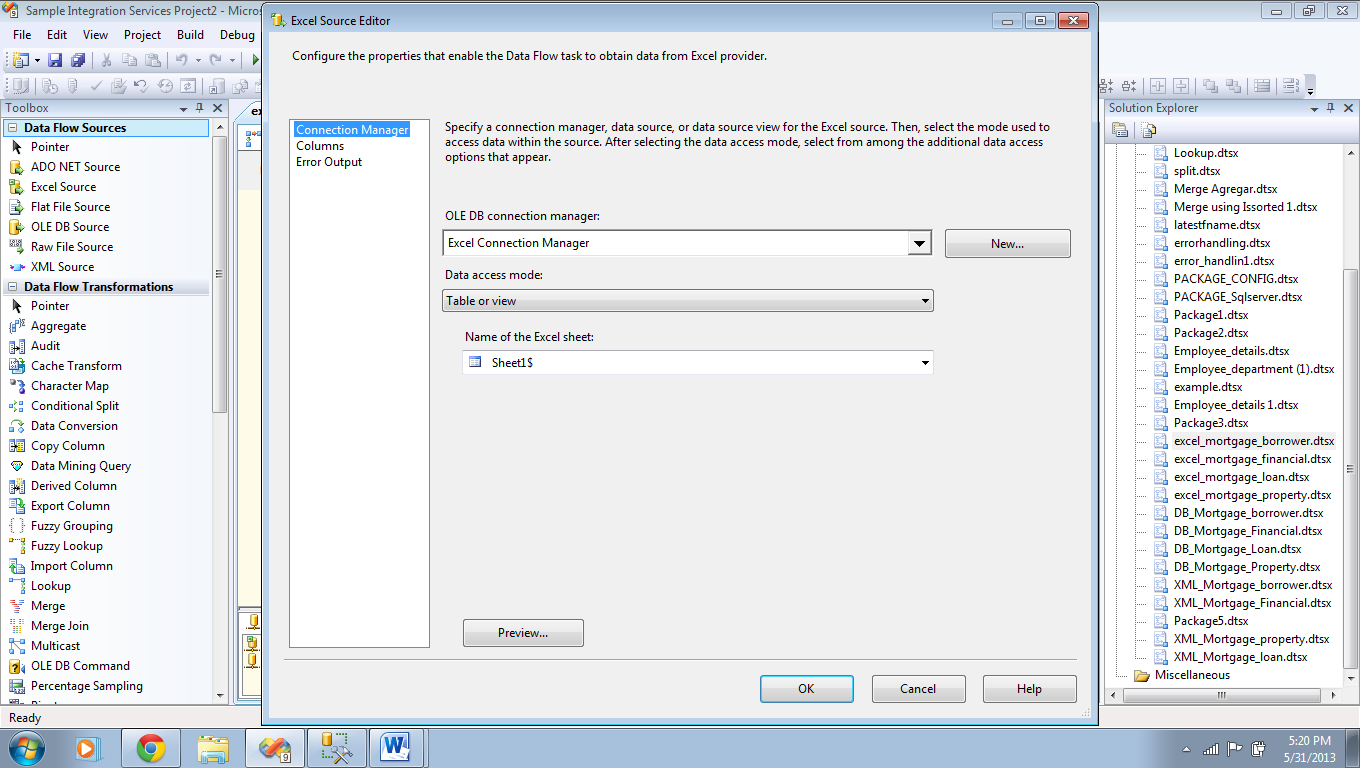
In Data flow task drag Excel source, Data conversion Transformation and OLE DB Destination.



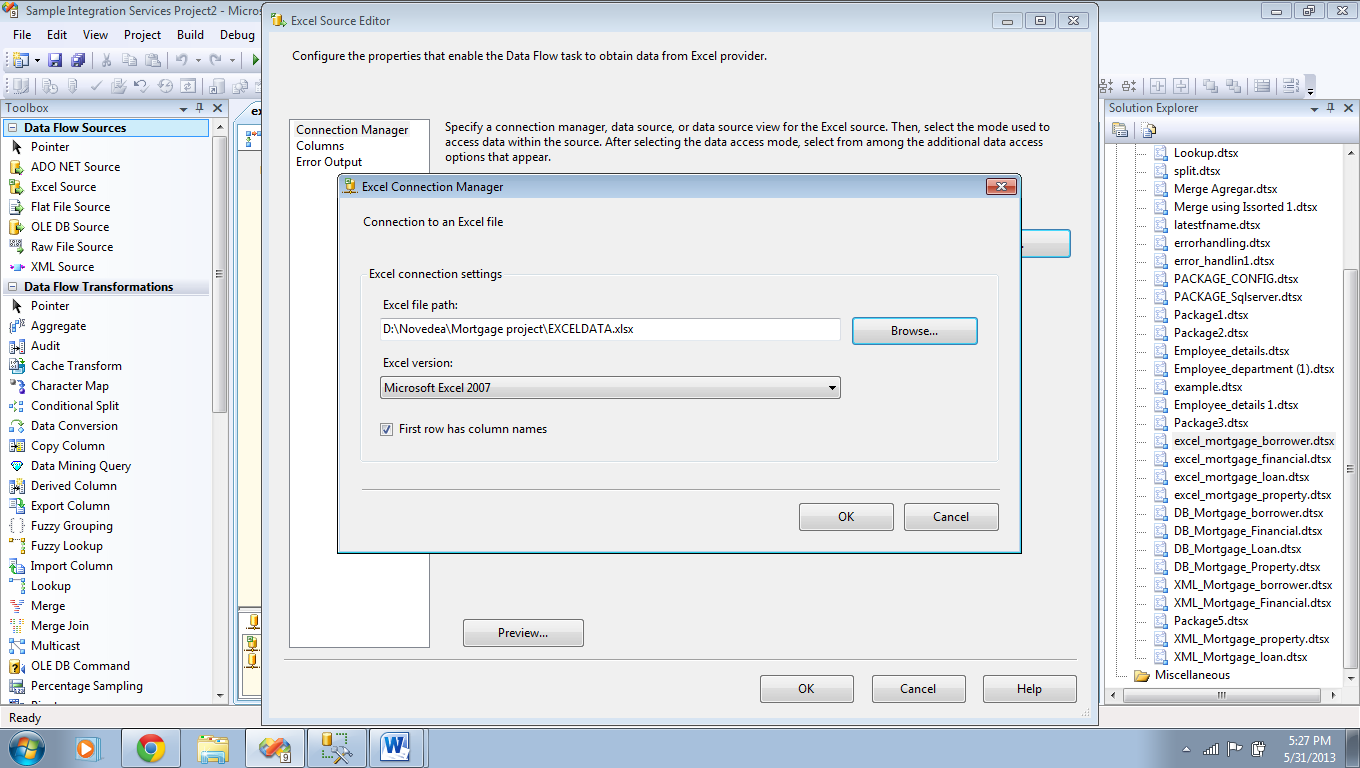
**Step 4: (Excel Source)**

In Excel Source, Under OLE DB Connection manager we need to select appropriate EXCEL connection manager (or) we need to create a new connection manager by clicking on NEW button.

The following figures illustrate the steps that had been described in the above paragraph.

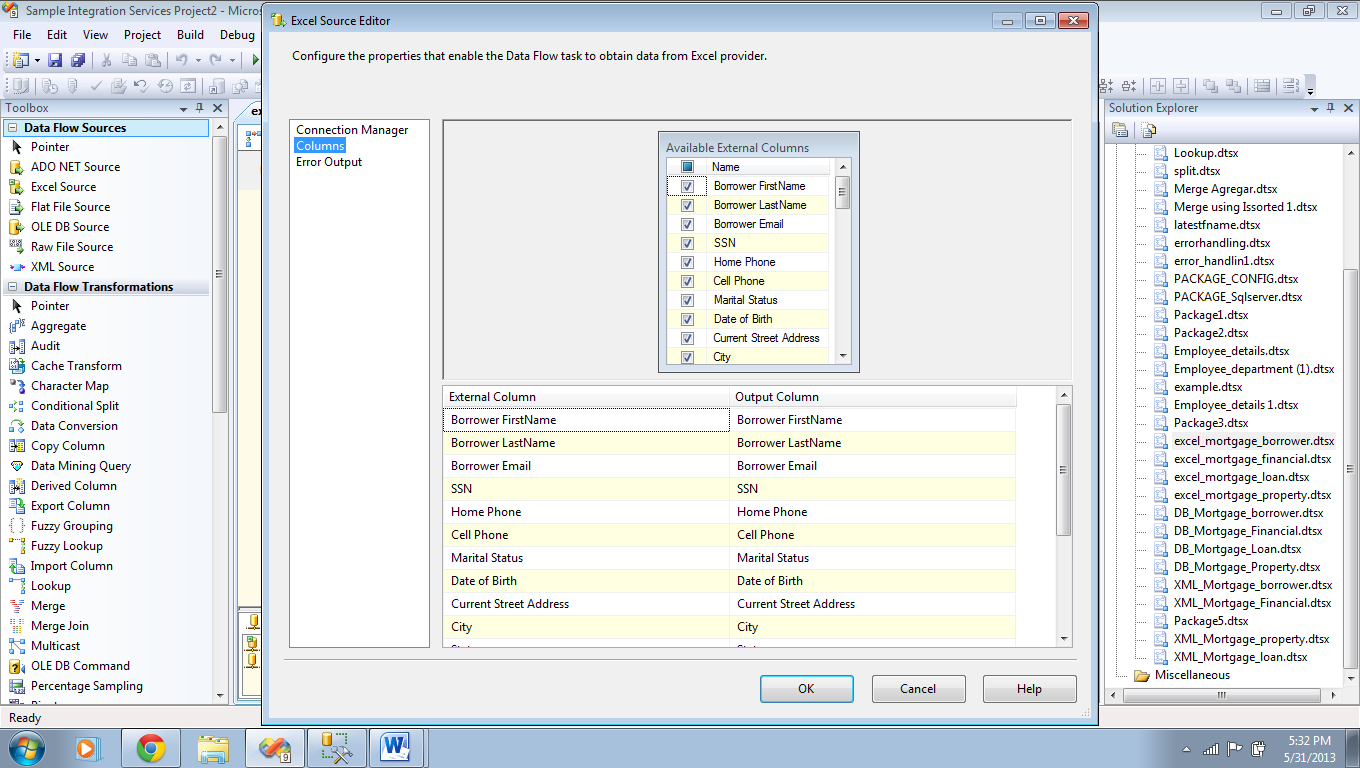


Browse the path and select the excel file



Click ok.

Now, Select the name of the Excel sheet and go to columns tab.

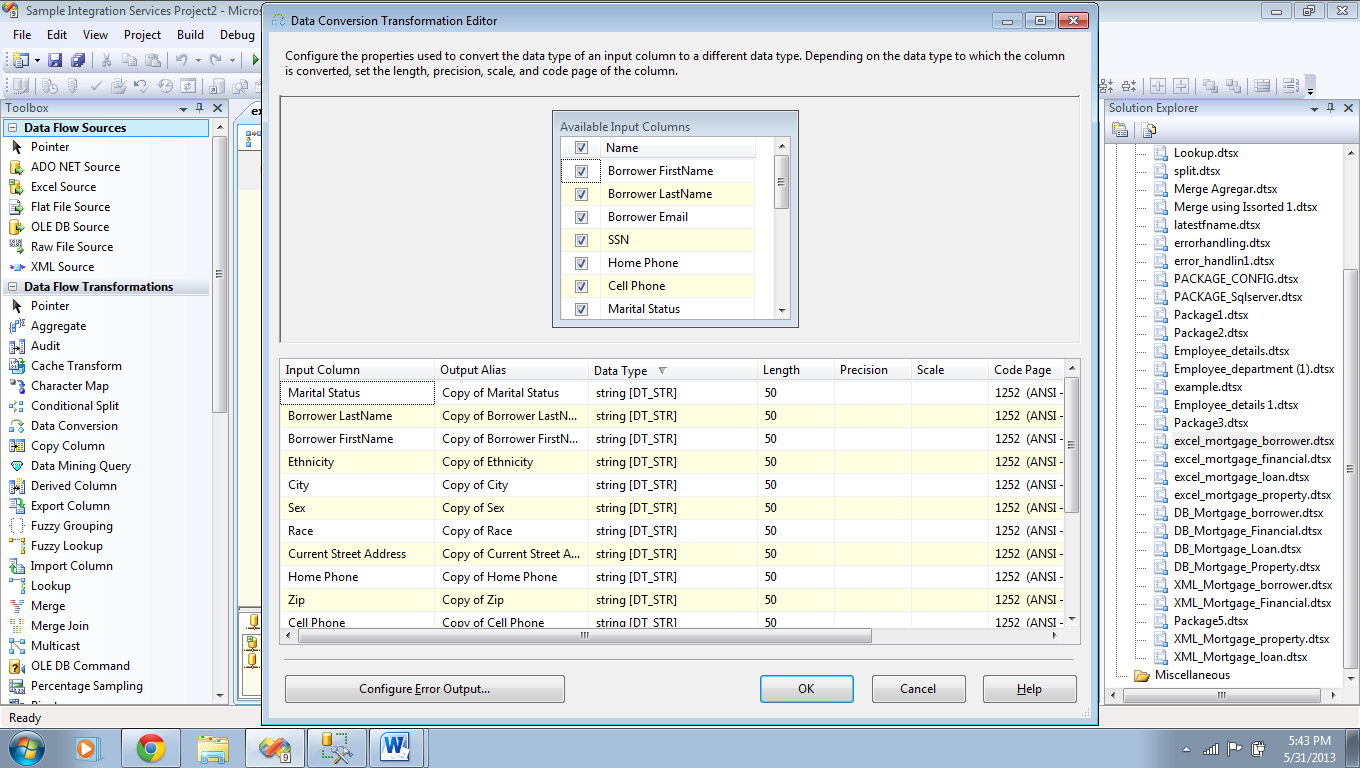


Select the columns that are to be inserted in the respective table and click ok.

**Step 5: (Data conversion Transformation)**

In Data conversion transformation we need to modify the data types of the output columns according to the data types of the OLE DB Destination. The following diagram depicts the SSIS to SQL server data types Translations.



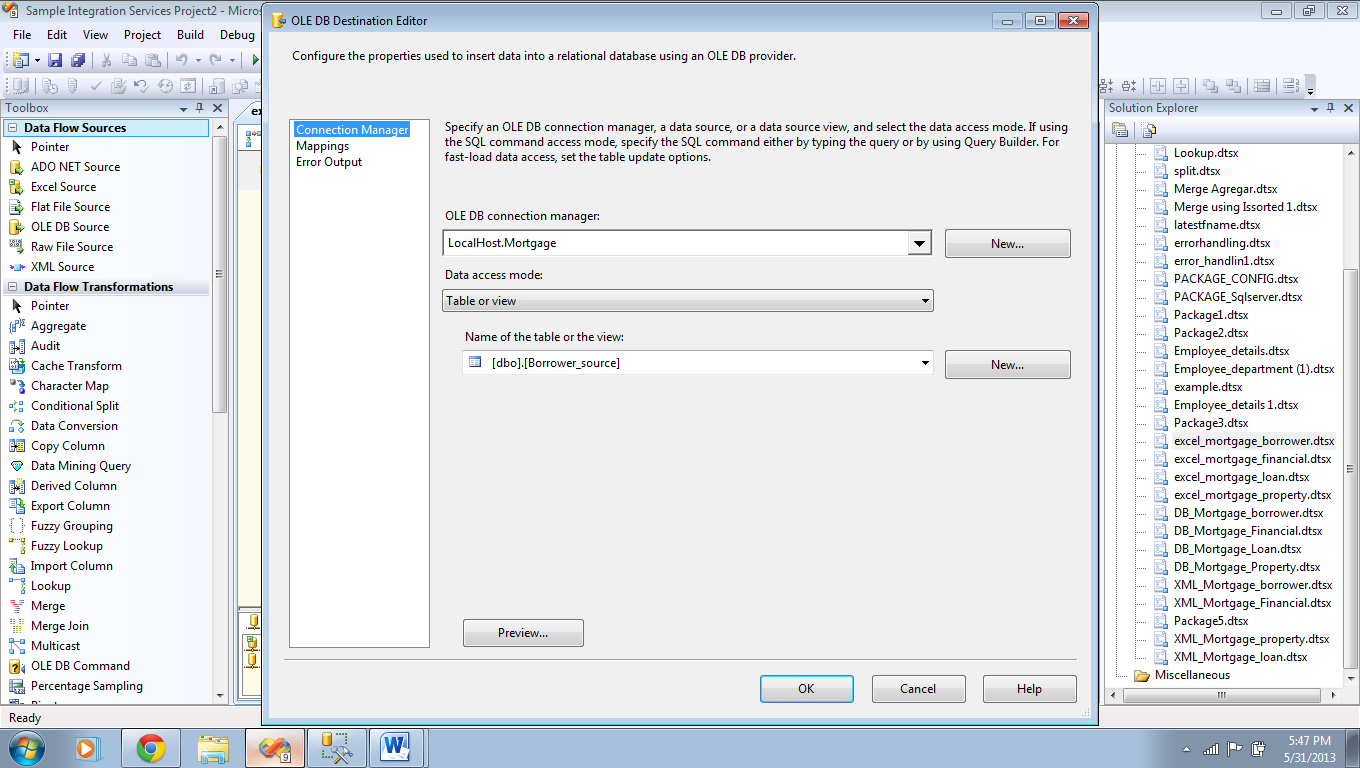


Click ok.

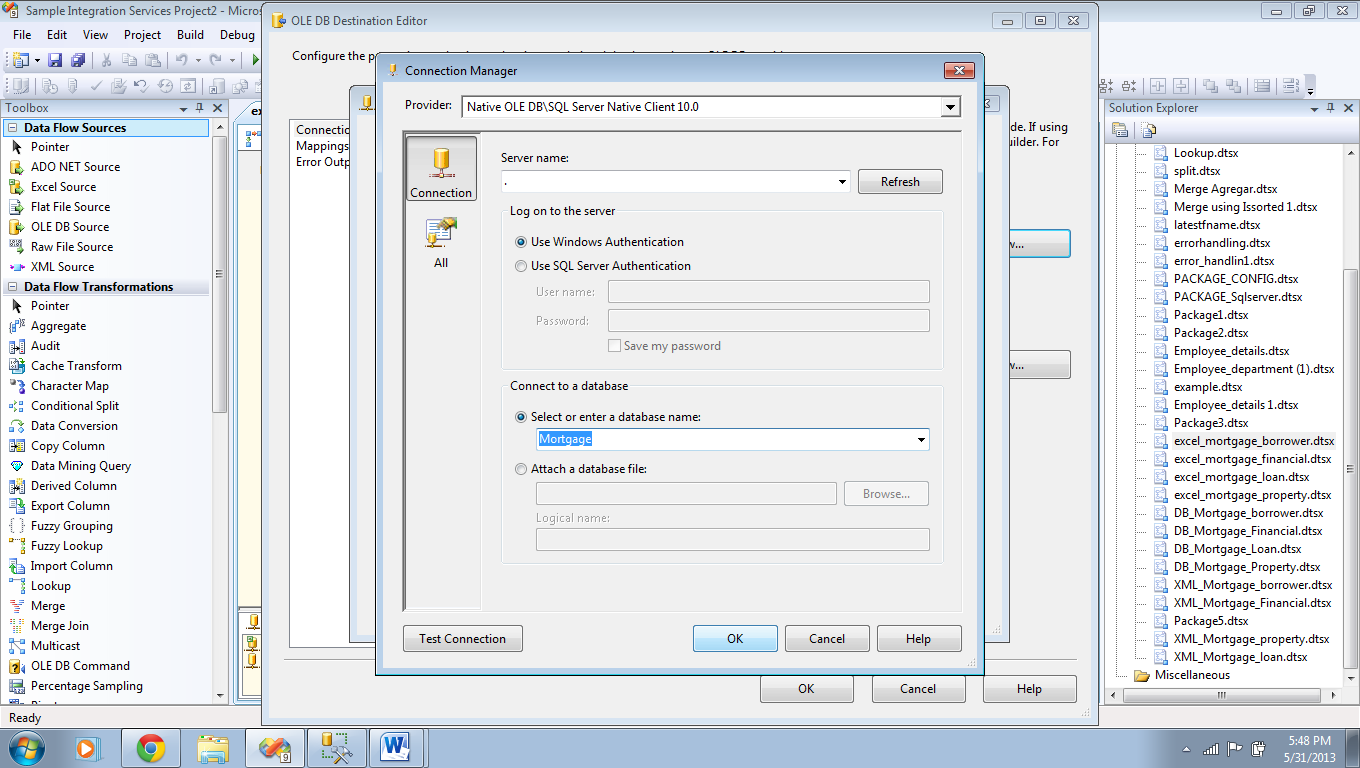
**Step 6: (OLE DB Destination)**

In OLE DB Destination, Under OLE DB Connection manager we need to select appropriate DB connection manager (or) we need to create a new connection manager by clicking on NEW button.

The following figures illustrate the steps that had been described in the above paragraph.



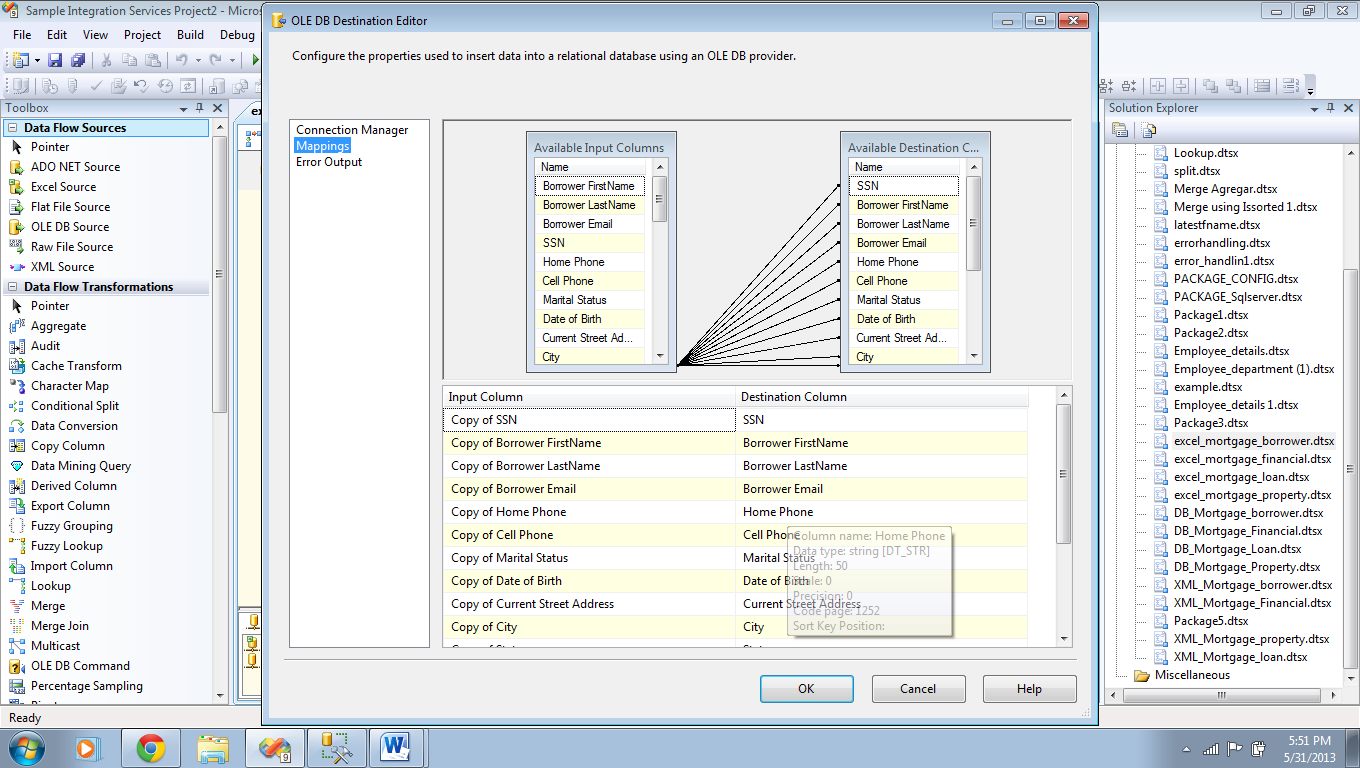
Click New button to create new Connection manager.



Select Server name and Database name, then check test connection and click ok.

Now, select the Name of the Table or view from Drop down list and go to Mapping’s tab.

In Mapping’s tab, select each and every column with appropriate output alias name provided in the Data conversion for each destination column.



Click Ok to continue.

Finally Save the package with proper naming convention and execute the package to load the data into the respective Staging table of the OLE DB destination.

**NOTE:-**

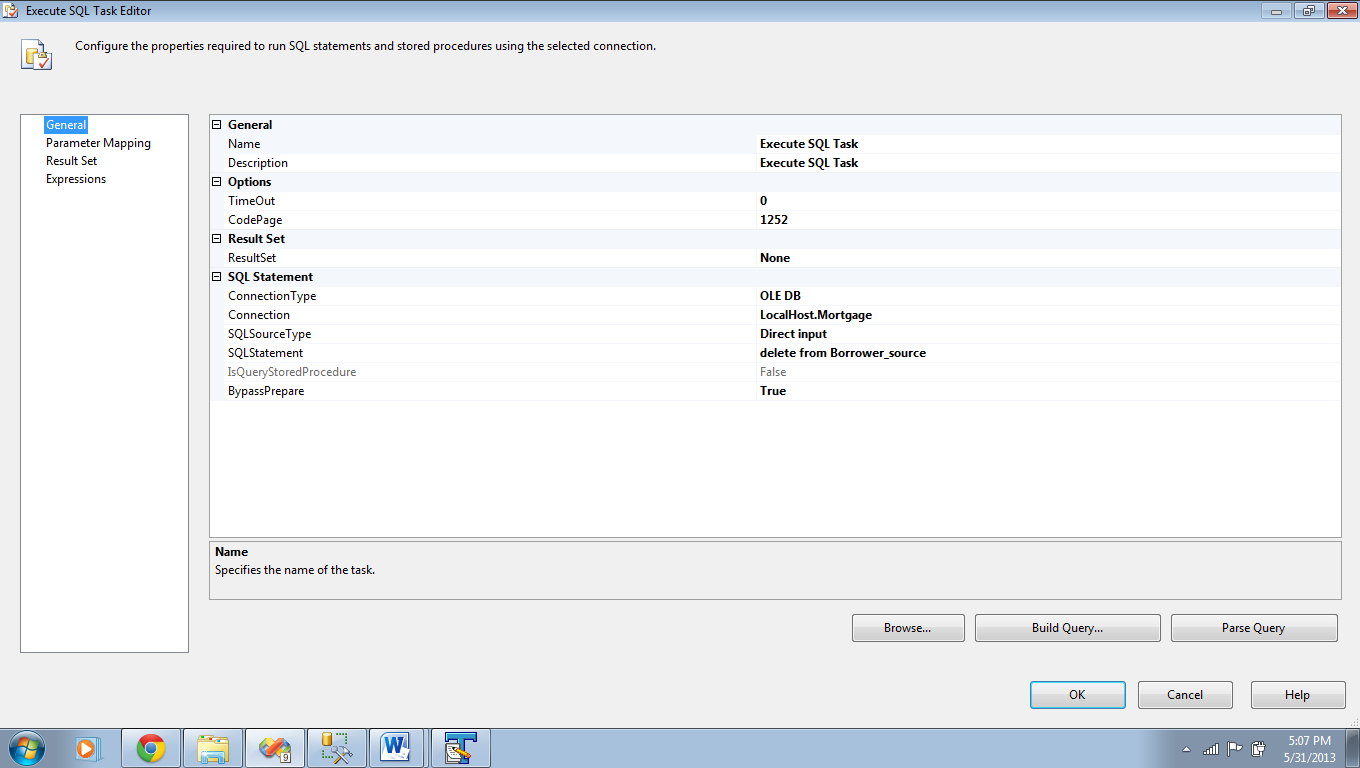
*Repeat the above 5 steps for each and every staging table.*

**2) DB packages:-** (Creating DB packages)

**Step 1:**

Go to Integration services Project and create a new package. Now, In the package drag Execute SQL task from the tool box.

Double click on it to edit..



Go to general tab and do the following modifications.

Connection type: “OLE DB”

Connection: Select the local host database in which your staging table is located.

SQL Statement: “Delete from Borrower\_source(your table name)”

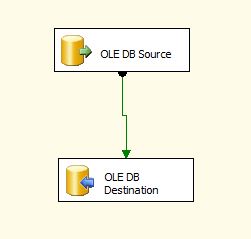
Click ok.

**Step 2:**

Now Drag the Data flow task and connect execute SQL task to Data flow task.

Double click on Data flow task.

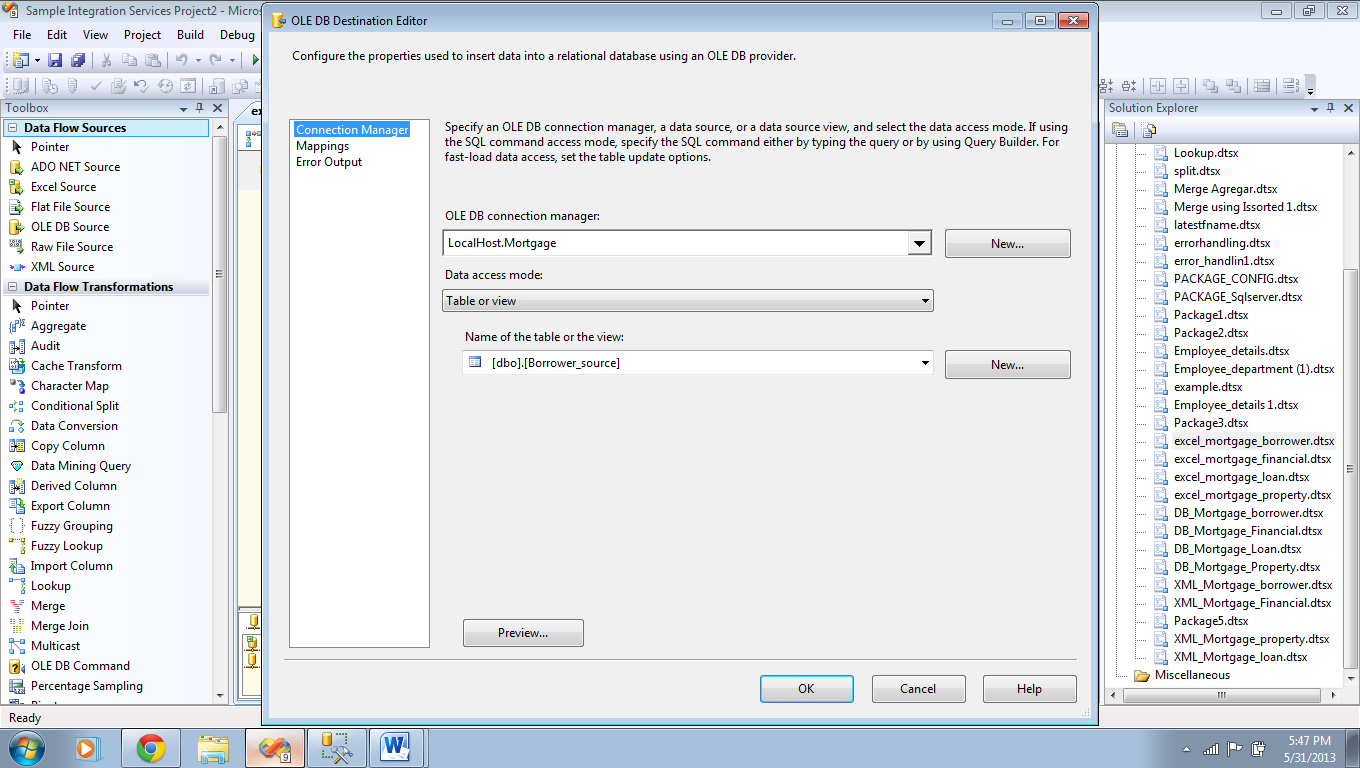
In Data flow task drag OLE DB source and OLE DB Destination.



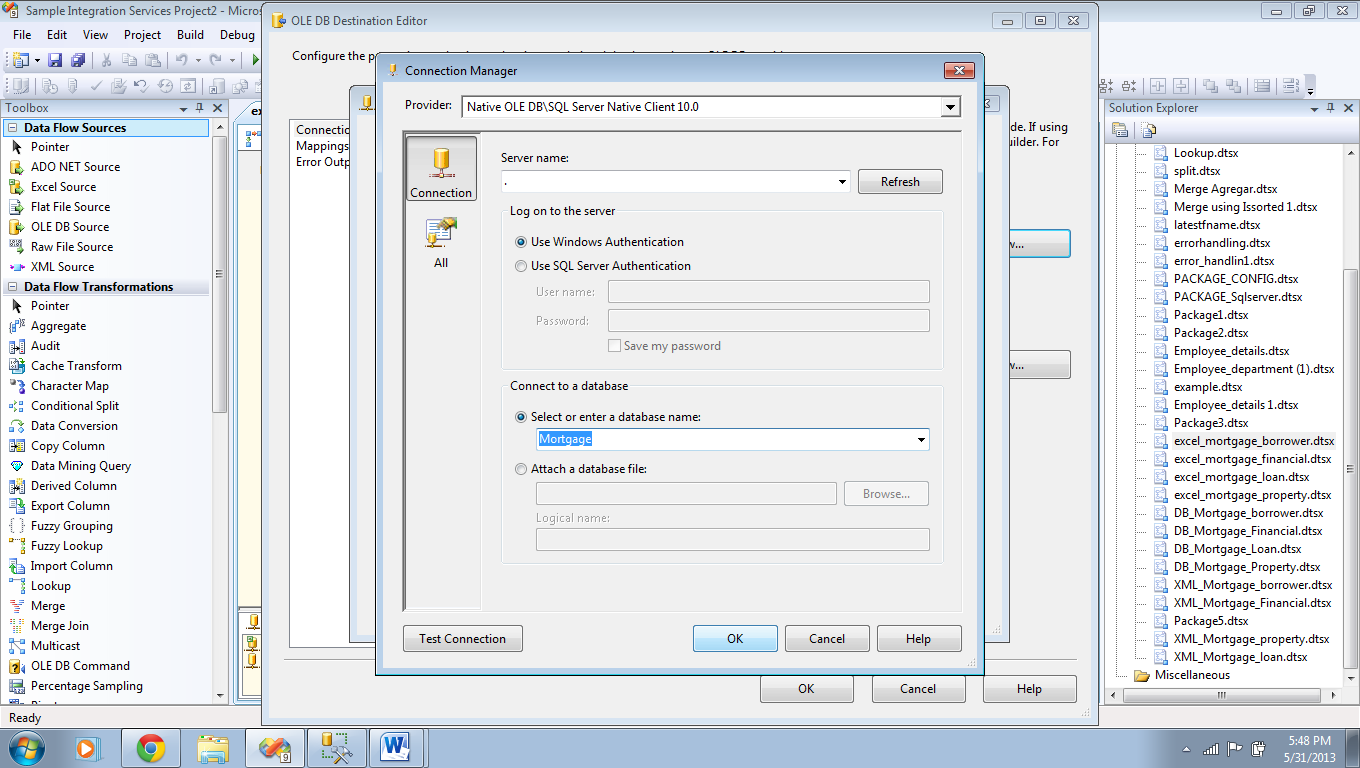
**Step 3: (OLE DB Source)**

In OLE DB Source, Under OLE DB Connection manager we need to select appropriate DB connection manager (or) we need to create a new connection manager by clicking on NEW button.

The following figures illustrate the steps that had been described in the above paragraph.



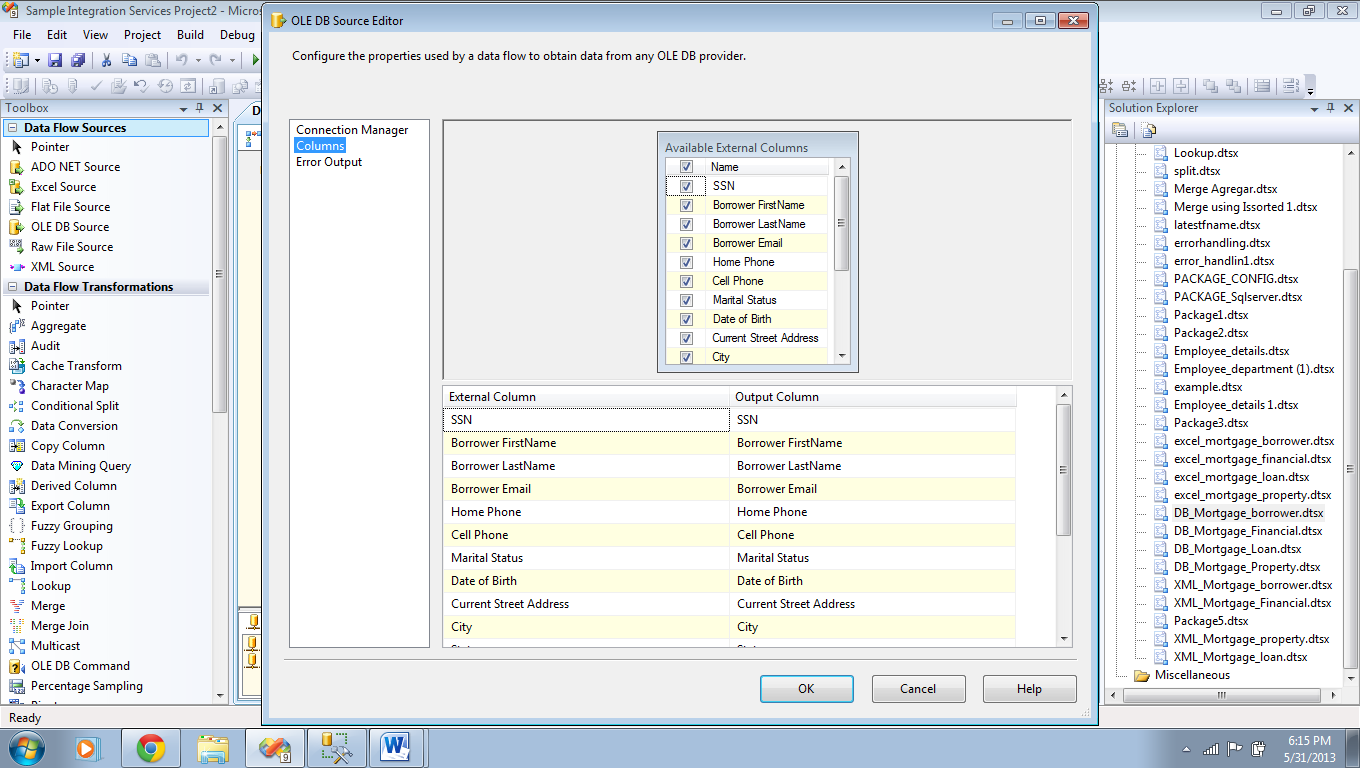
Click New button to create new Connection manager.



Select Server name and Database name, then check test connection and click ok.

Now, select the Name of the Table or view from Drop down list and go to columns tab.

In columns tab, select the columns that are to be inserted in the Destination table.

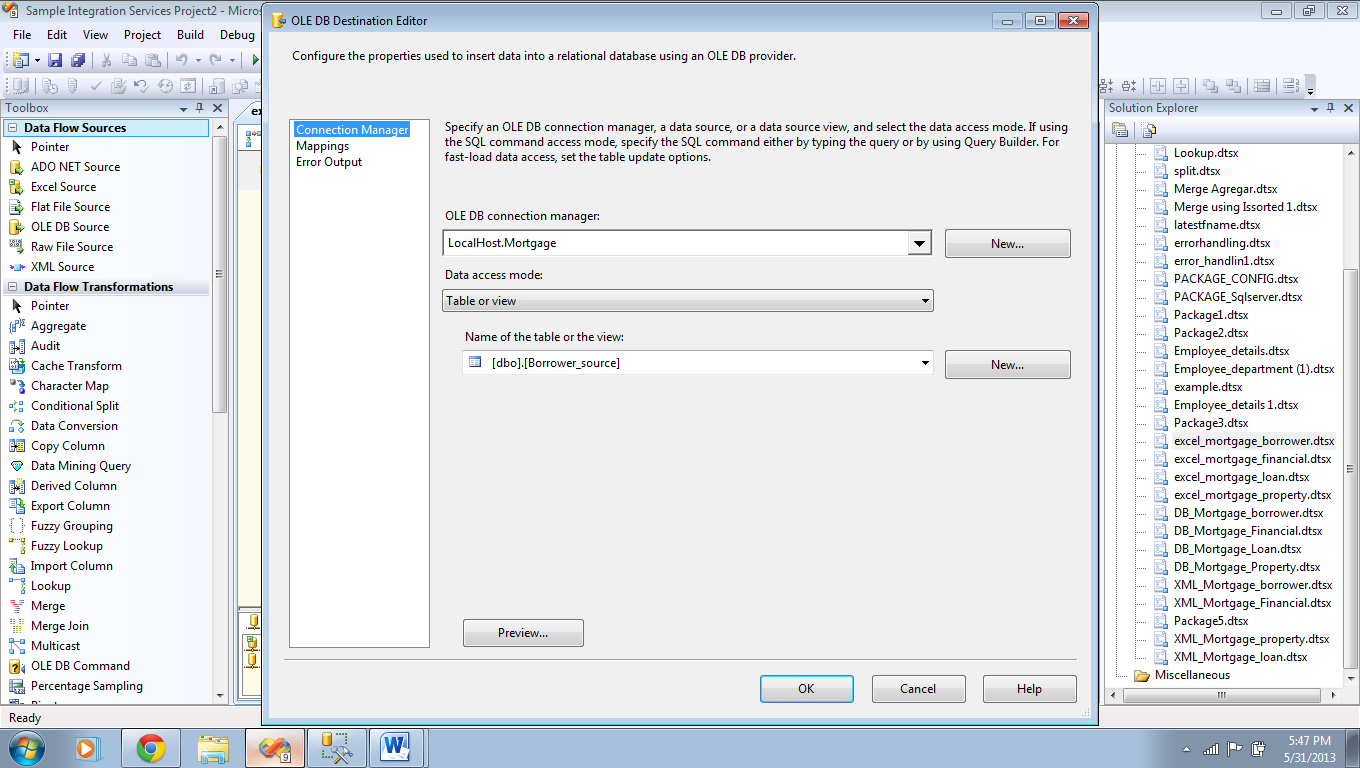


Click ok.

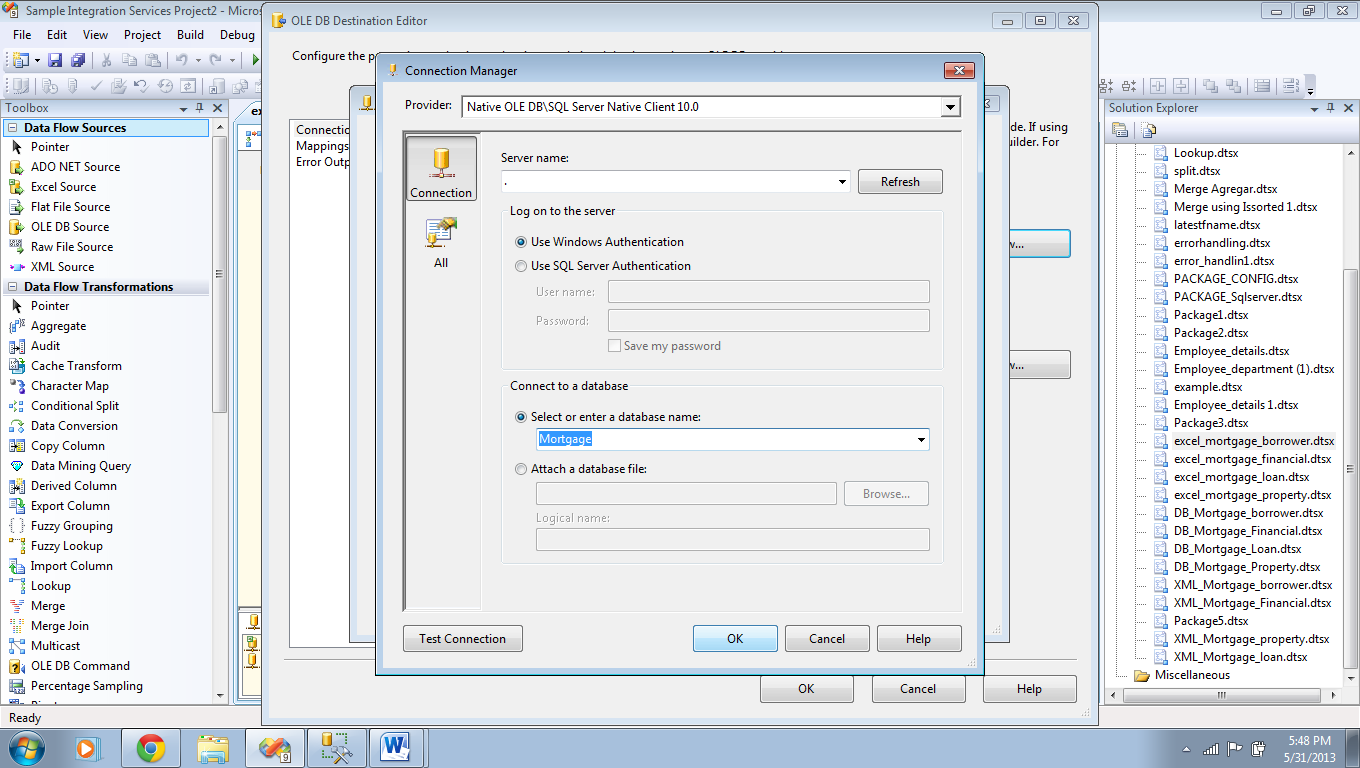
**Step 4:**

In OLE DB Destination, Under OLE DB Connection manager we need to select appropriate DB connection manager (or) we need to create a new connection manager by clicking on NEW button.

The following figures illustrate the steps that had been described in the above paragraph.



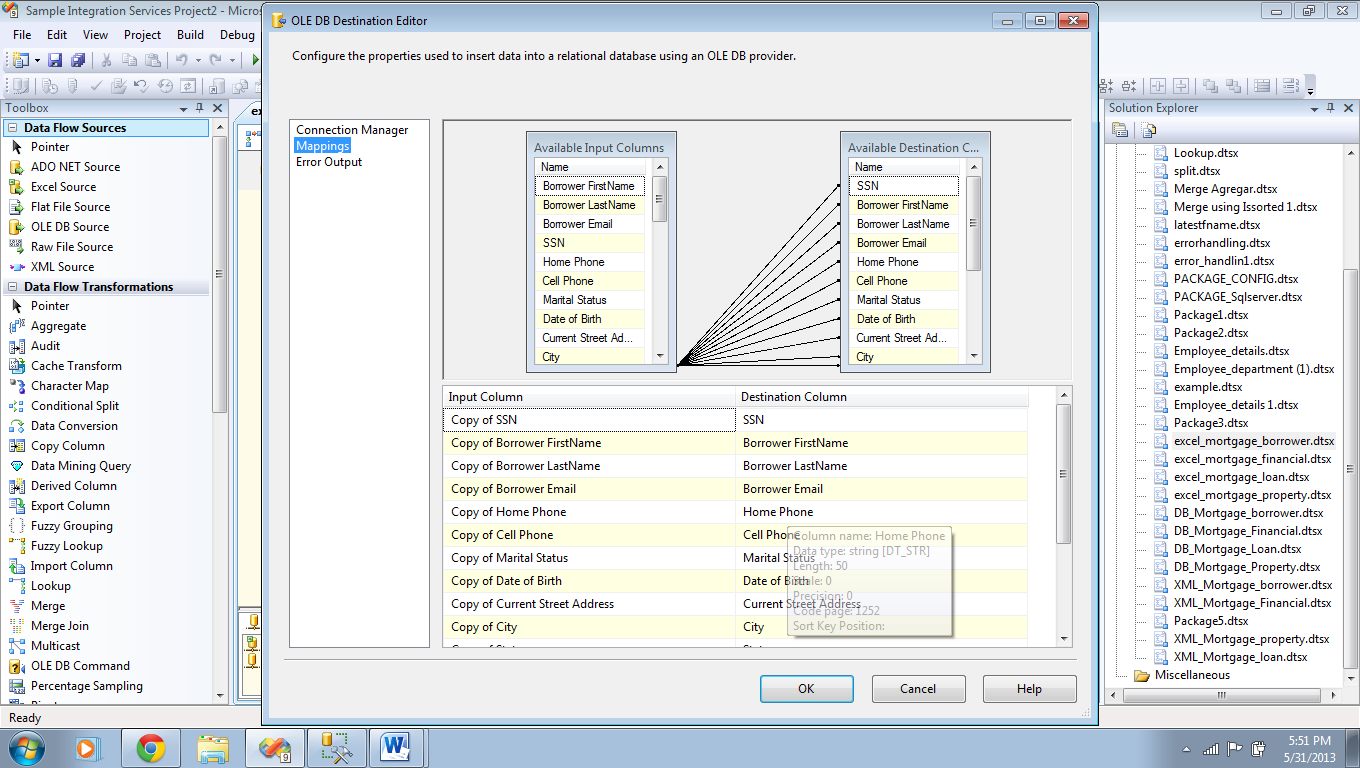
Click New button to create new Connection manager.



Select Server name and Database name, then check test connection and click ok.

Now, select the Name of the Table or view from Drop down list and go to Mapping’s tab.

In Mapping’s tab, select each and every column with appropriate output alias name provided in the Data conversion for each destination column.



Click Ok to continue.

**NOTE:-**

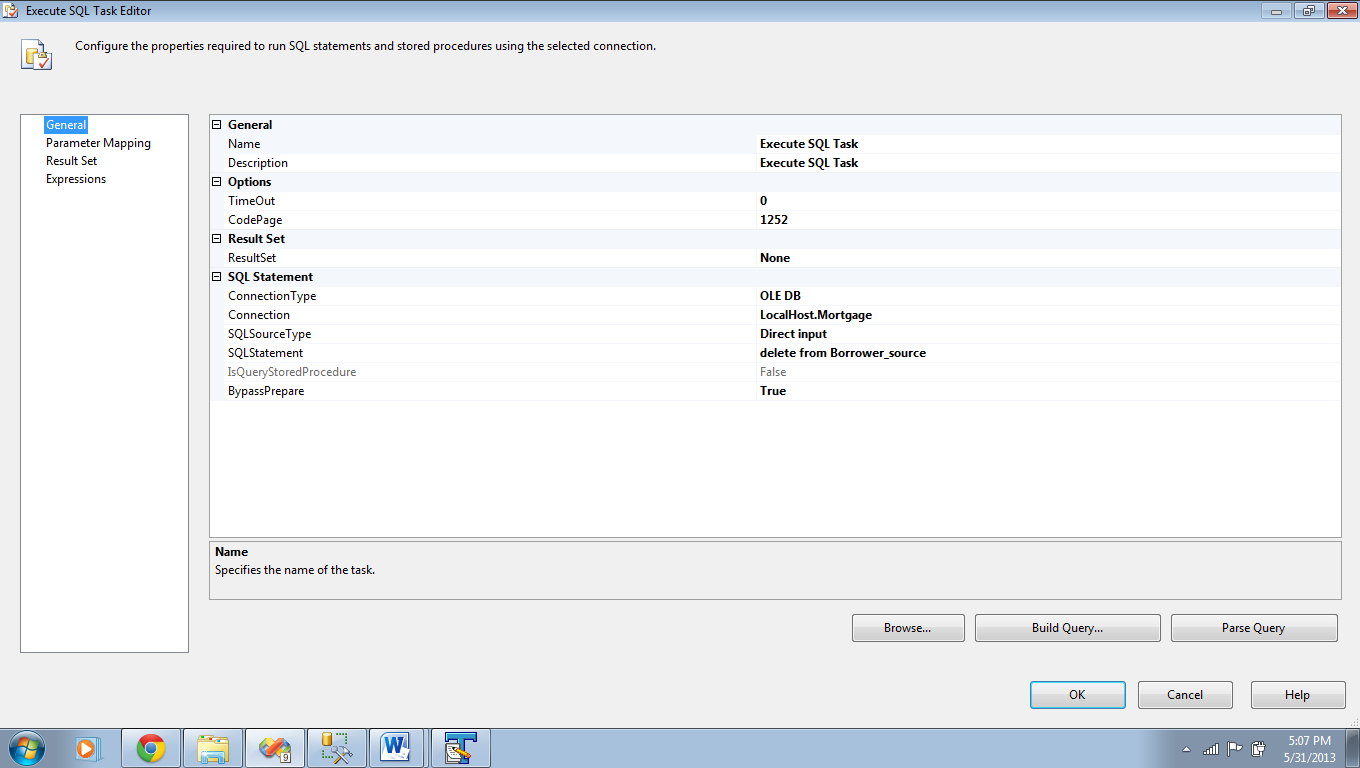
*Repeat the above 5 steps for each and every staging table.*

**3) XML packages:-** (Creating XML packages)

**Step 1:**

Go to Integration services Project and create a new package. Now, In the package drag Execute SQL task from the tool box.

Double click on it to edit..



Go to general tab and do the following modifications.

Connection type: “OLE DB”

Connection: Select the local host database in which your staging table is located.

SQL Statement: “Delete from Borrower\_source(your table name)”

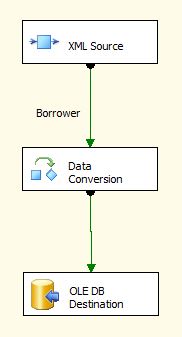
Click ok.

**Step 2:**

Now Drag the Data flow task and connect execute SQL task to Data flow task.

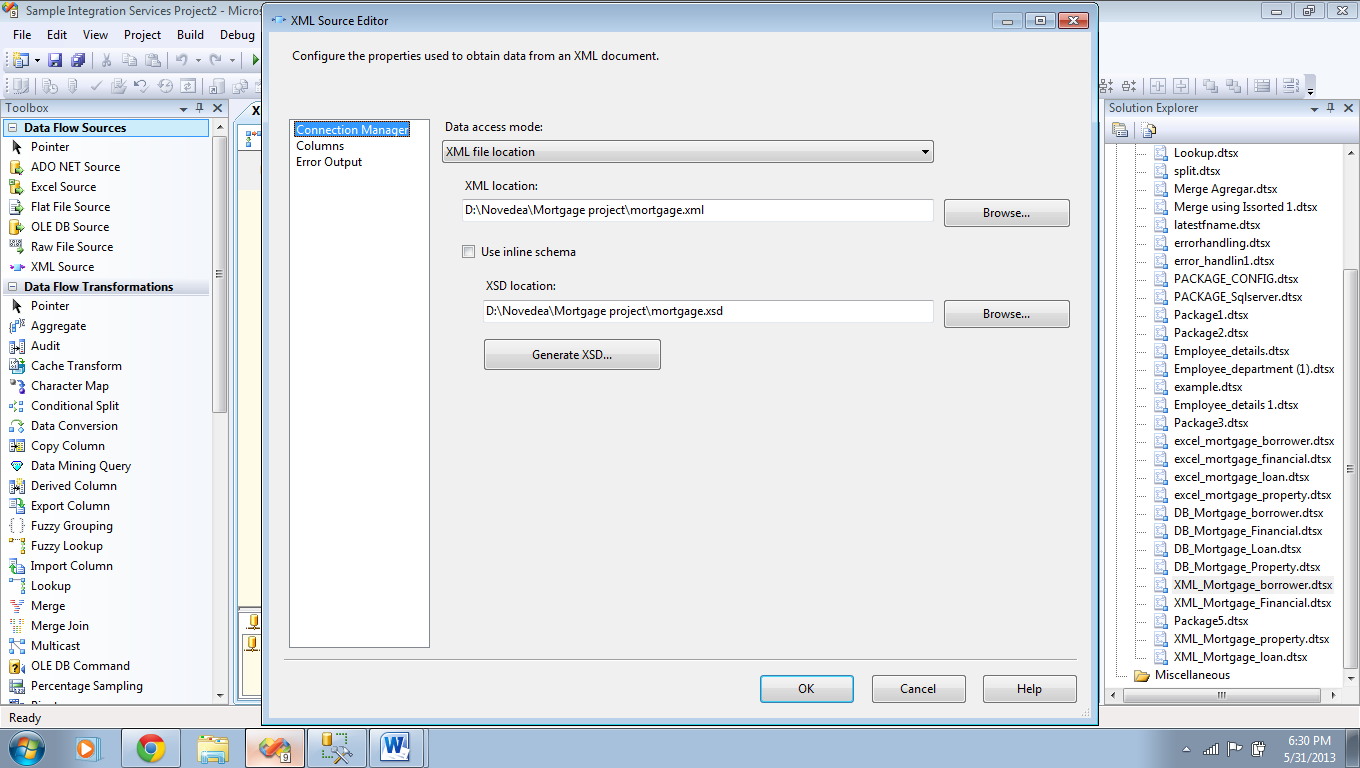
Double click on Data flow task.

In Data flow task drag XML source, Data conversion Transformation and OLE DB Destination.



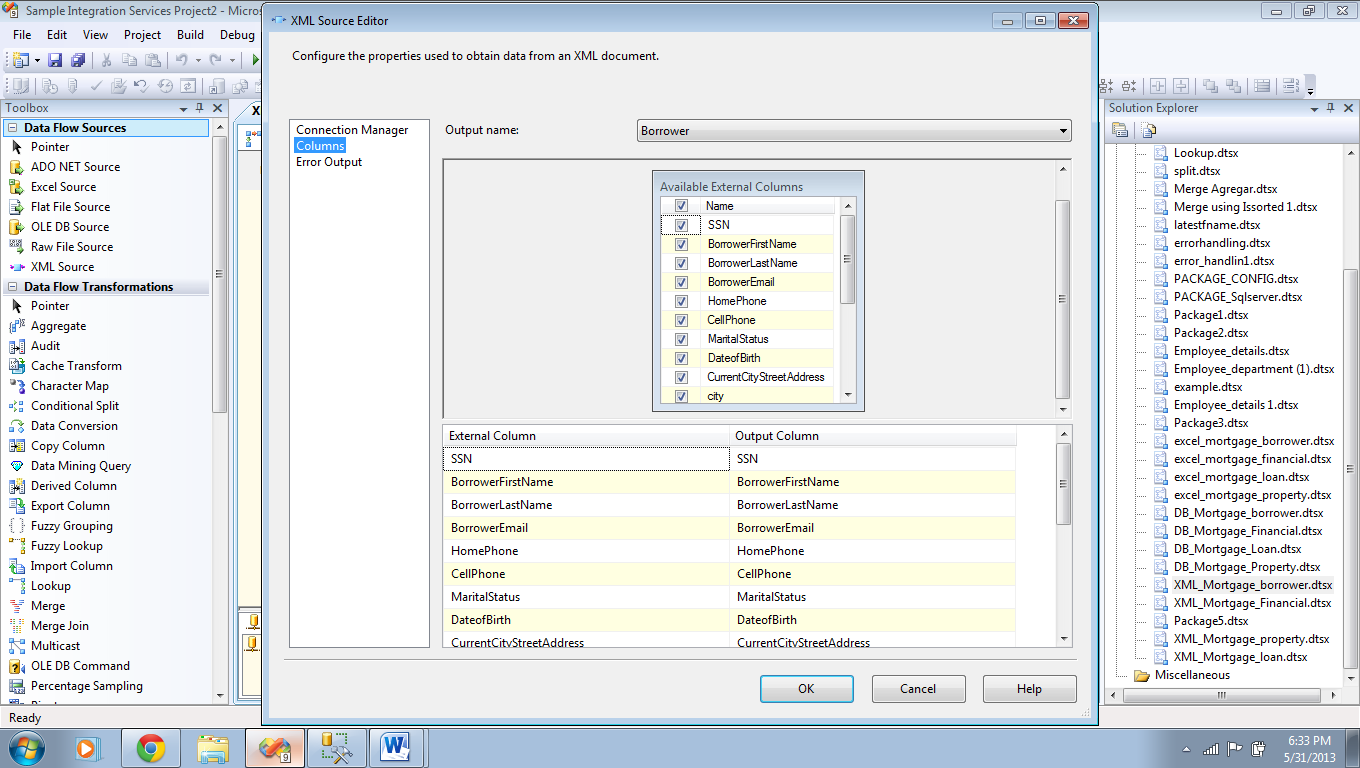
**Step 3: (XML Source)**

Drag XML source and double click on it. Now, It will pop up window similer to the following window



Select XML file location by clicking browse button and then click Generate XSD button to create XML schema definition file and save it.

Now, Go to Columns tab.



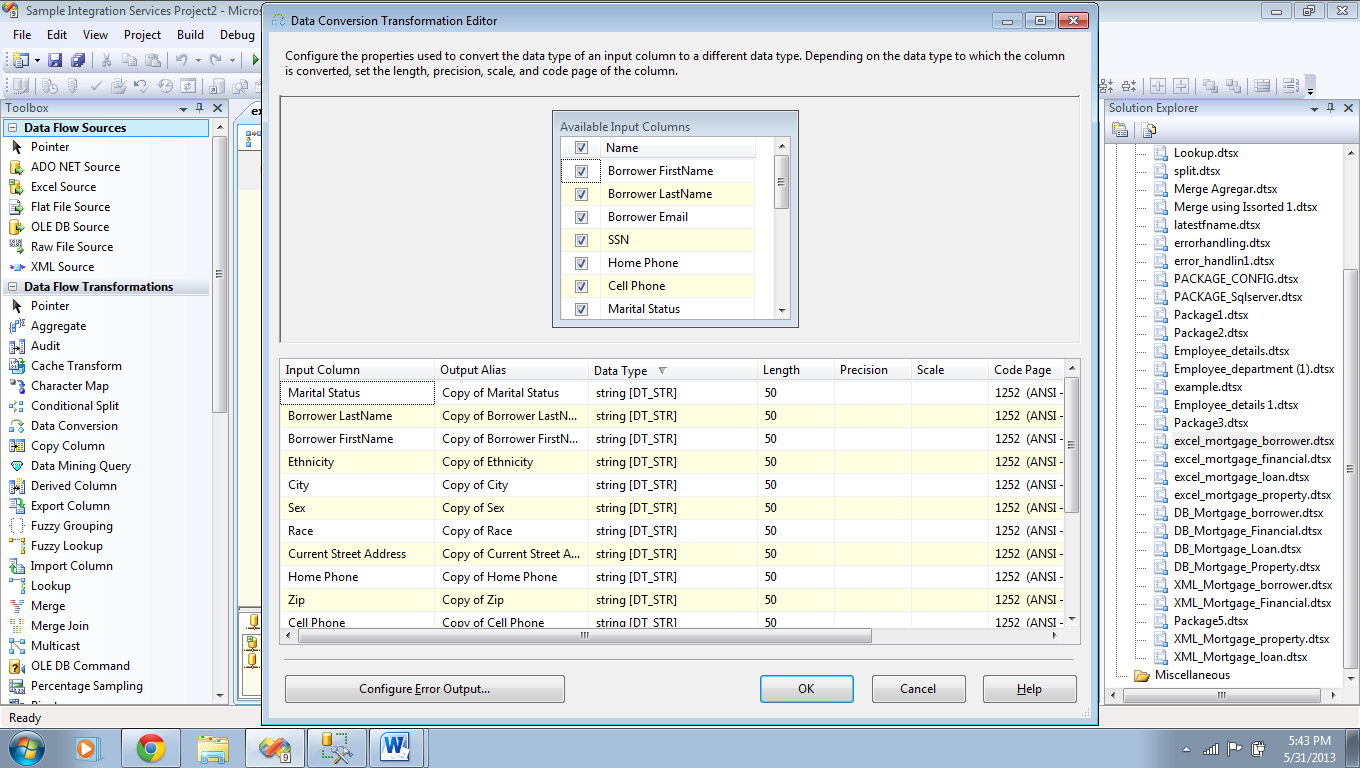
In the above window, initially select “Output name” drop down list choose appropriate entity to load the data into the staging table.

Now, select the list of columns that are needed be inserted and then Click Ok.

**Step 4: (Data conversion Transformation)**

In Data conversion transformation we need to modify the data types of the output columns according to the data types of the OLE DB Destination. The following diagram depicts the SSIS to SQL server data types Translations.



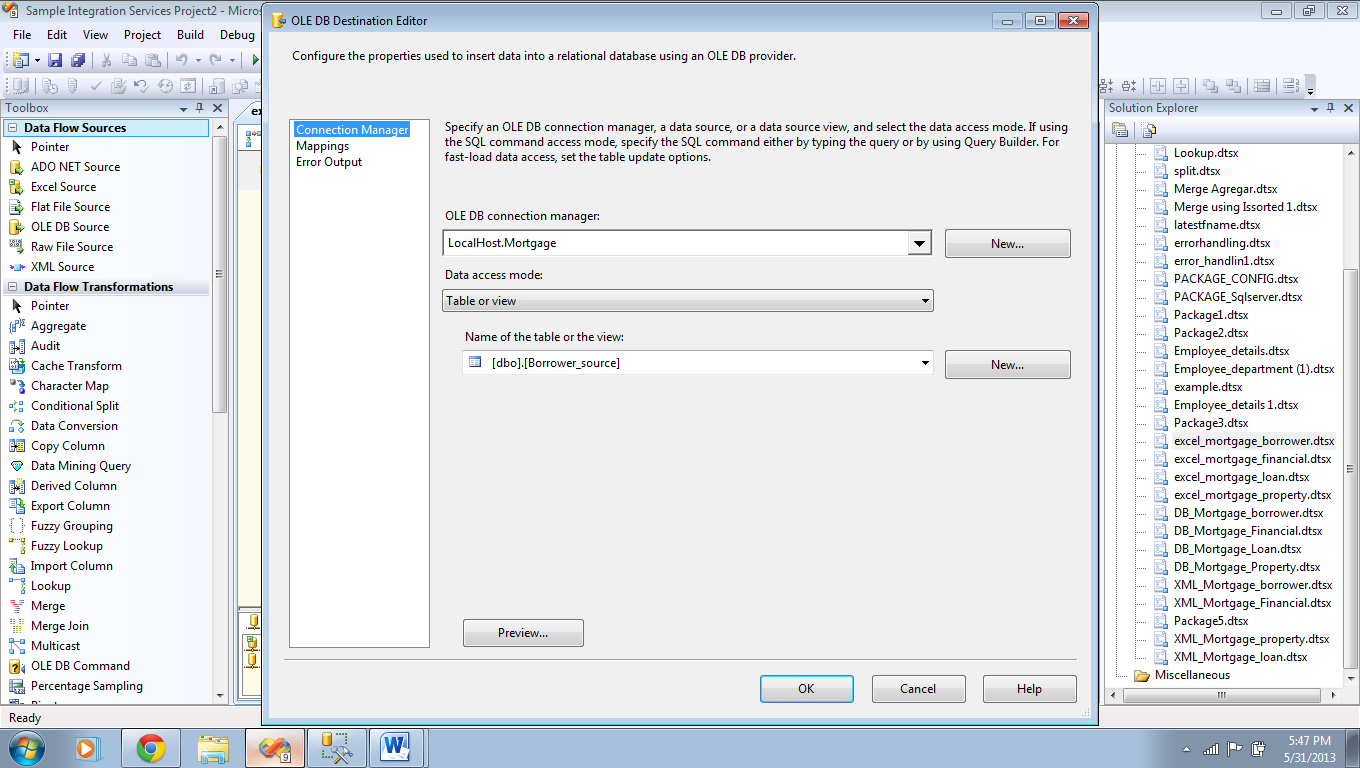


Click ok.

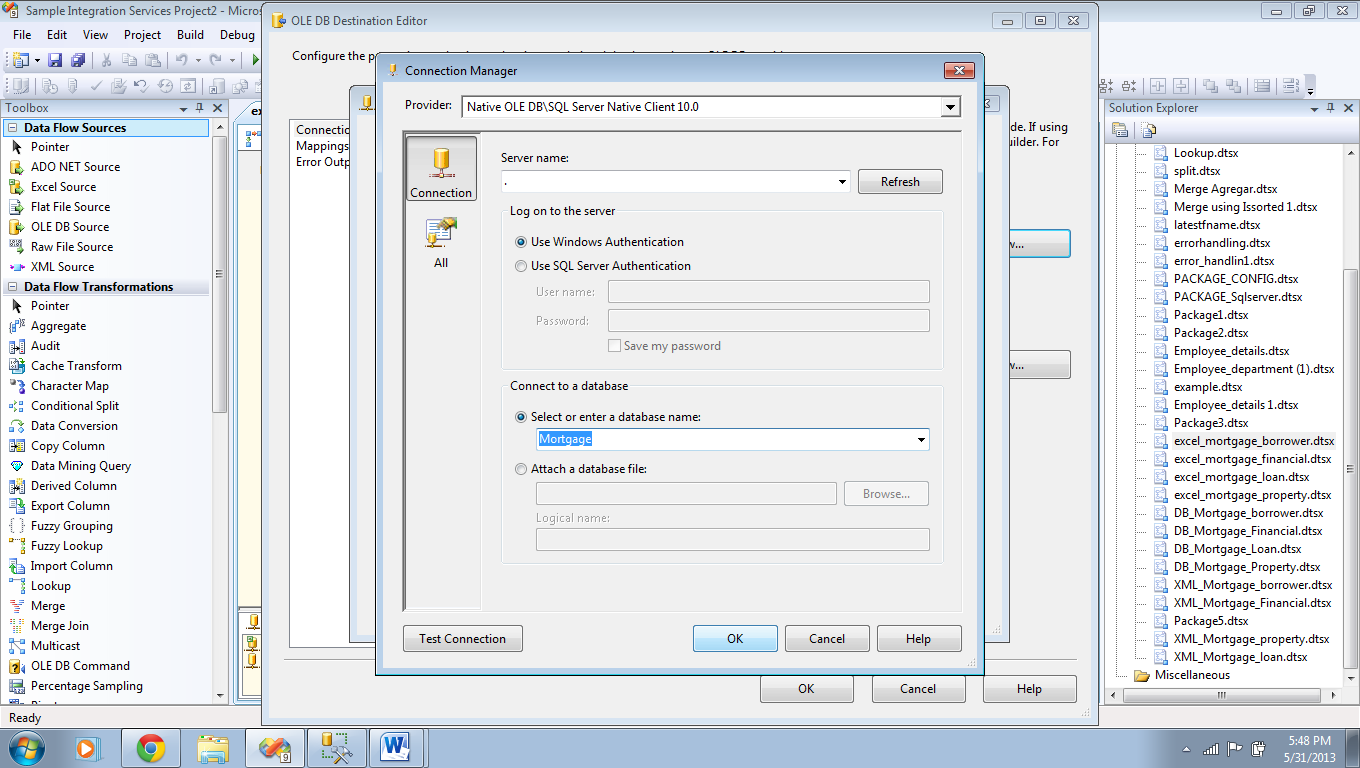
**Step 5: (OLE DB Destination)**

In OLE DB Destination, Under OLE DB Connection manager we need to select appropriate DB connection manager (or) we need to create a new connection manager by clicking on NEW button.

The following figures illustrate the steps that had been described in the above paragraph.



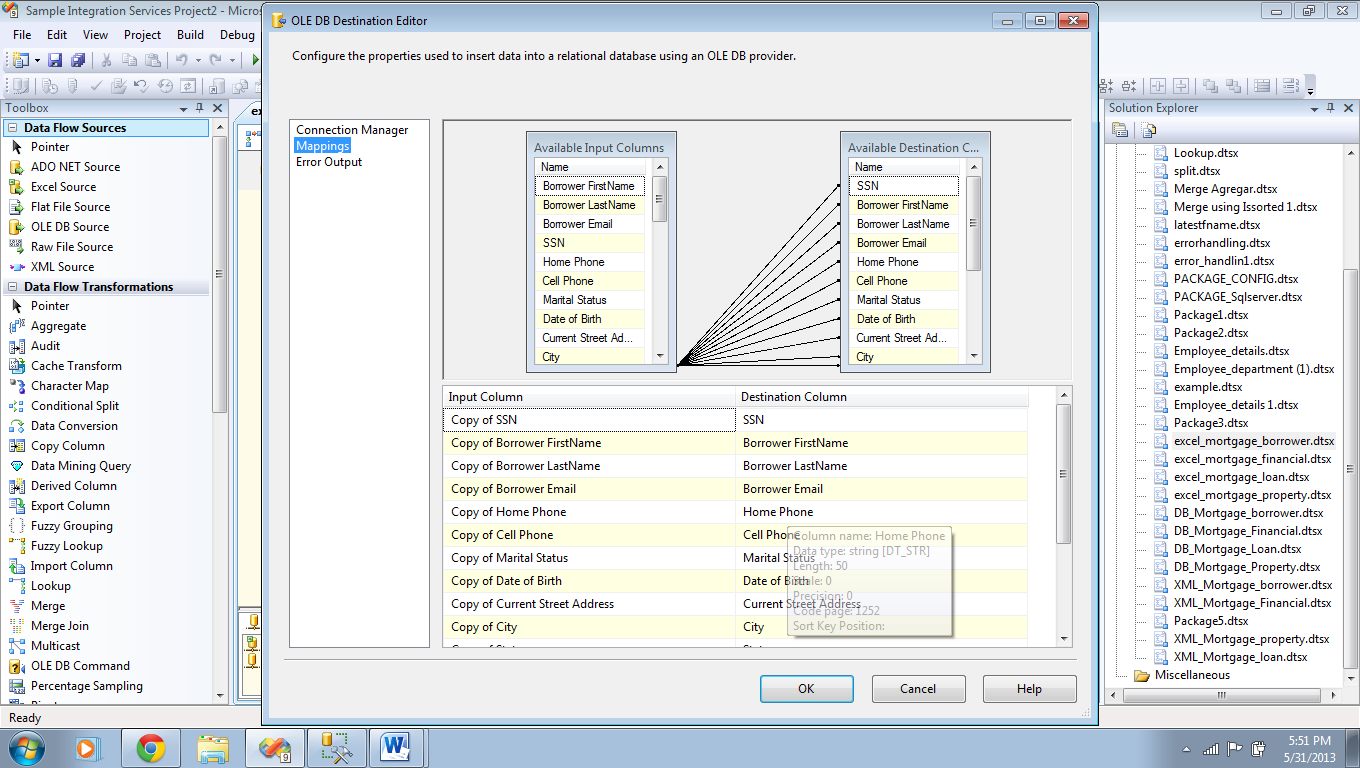
Click New button to create new Connection manager.



Select Server name and Database name, then check test connection and click ok.

Now, select the Name of the Table or view from Drop down list and go to Mapping’s tab.

In Mapping’s tab, select each and every column with appropriate output alias name provided in the Data conversion for each destination column.



Click Ok to continue.

Finally Save the package with proper naming convention and execute the package to load the data into the respective Staging table of the OLE DB destination.

**NOTE:-**

*Repeat the above 5 steps for each and every staging table.*

## Story 4: Validate & Error Handling staging data

## Story 5: Insert data into ODS database

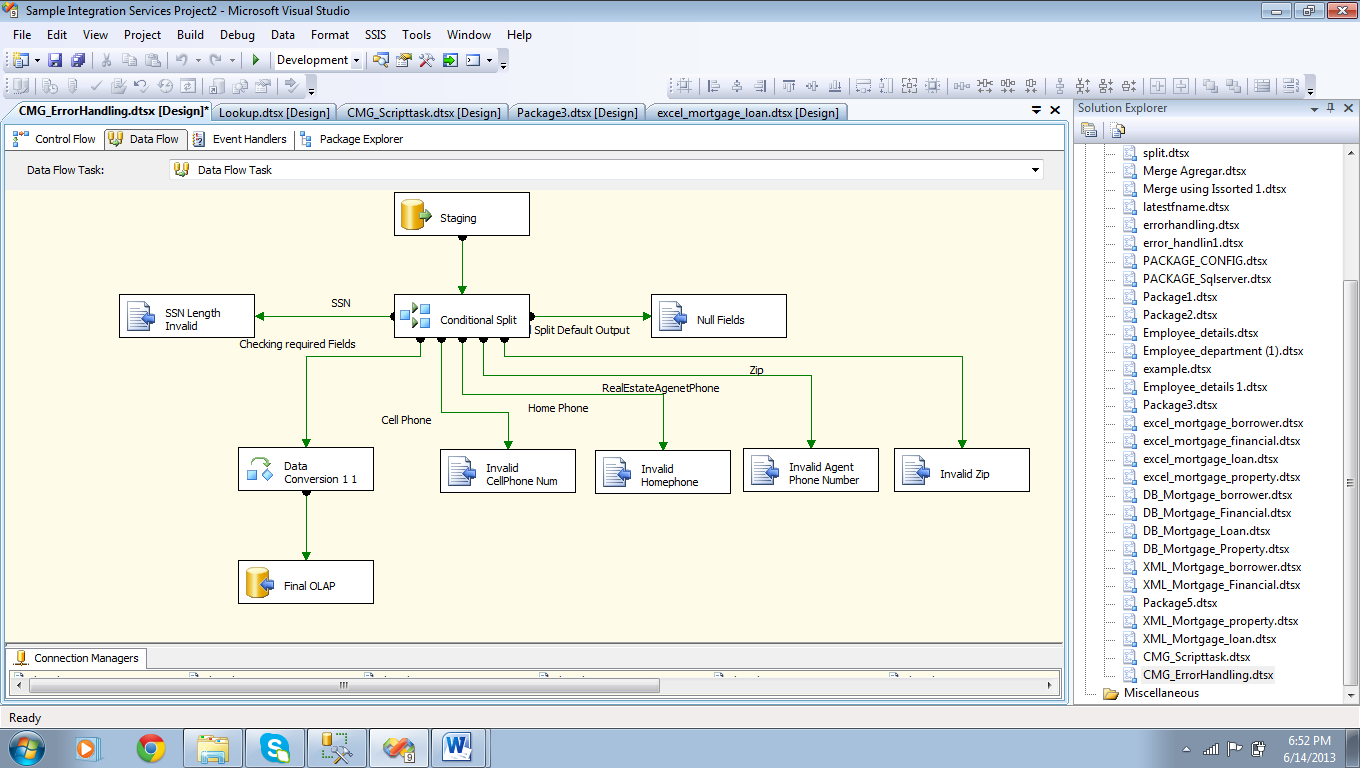
In Story 4 and 5, we will validate the 1003 mortgage application data that is received to make sure that we have all required information in the format that is acceptable So that we can load the data into OLAP database.

The Data validations that we do in this story are:

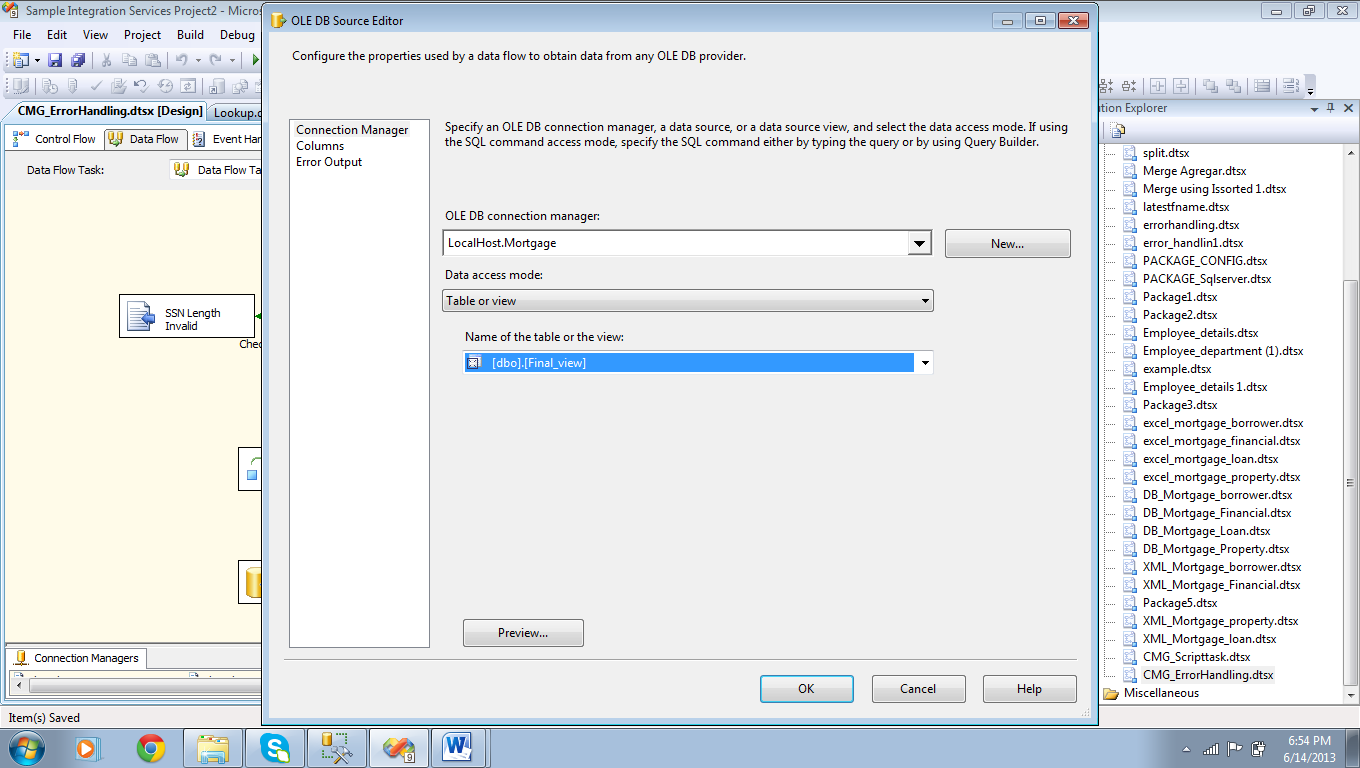
1. To Check all the required fields are present are not.
2. Verifying length of specified fields like Phone numbers, SSN and Zip are according to Appendix.

*Creating Error Handling Package:-*

This package will contain following transformations in its Data flow task.



Initially configure OLE DB source as shown,



Do the following steps:

1. Create appropriate OLE DB Connection Manager.
2. Select the view which needs to be created before creating the connection manager. The view must contain all the columns information so that we can load the data into De-normalized OLAP Database using this view.

SQL Code for creating the view:

create view [dbo].[Mortgage\_view] as

select a.\*,

b.MonthlyIncome,b.Bonuses,b.Commission,b.OtherIncome,b.[Rent or Own],b.Checking,b.Savings,b.RetirementFund,b.MutualFund,

c.[Property City],c.[Property State],c.[Property Usage],c.[Property Zip],c.Property\_ID,c.RealEstateAgentEmail,c.RealEstateAgentName,c.RealEstateAgentPhone,

d.Loan\_ID,d.[Purpose of Loan],d.LoanAmount,d.[Purchase Price],d.CreditCardAuthorization,d.[Number of Units],d.Refferal,d.[Co-Borrower SSN]

from Borrower\_source a

Inner join Financial\_source b

Inner join Property\_source c

Inner join Loan\_source d

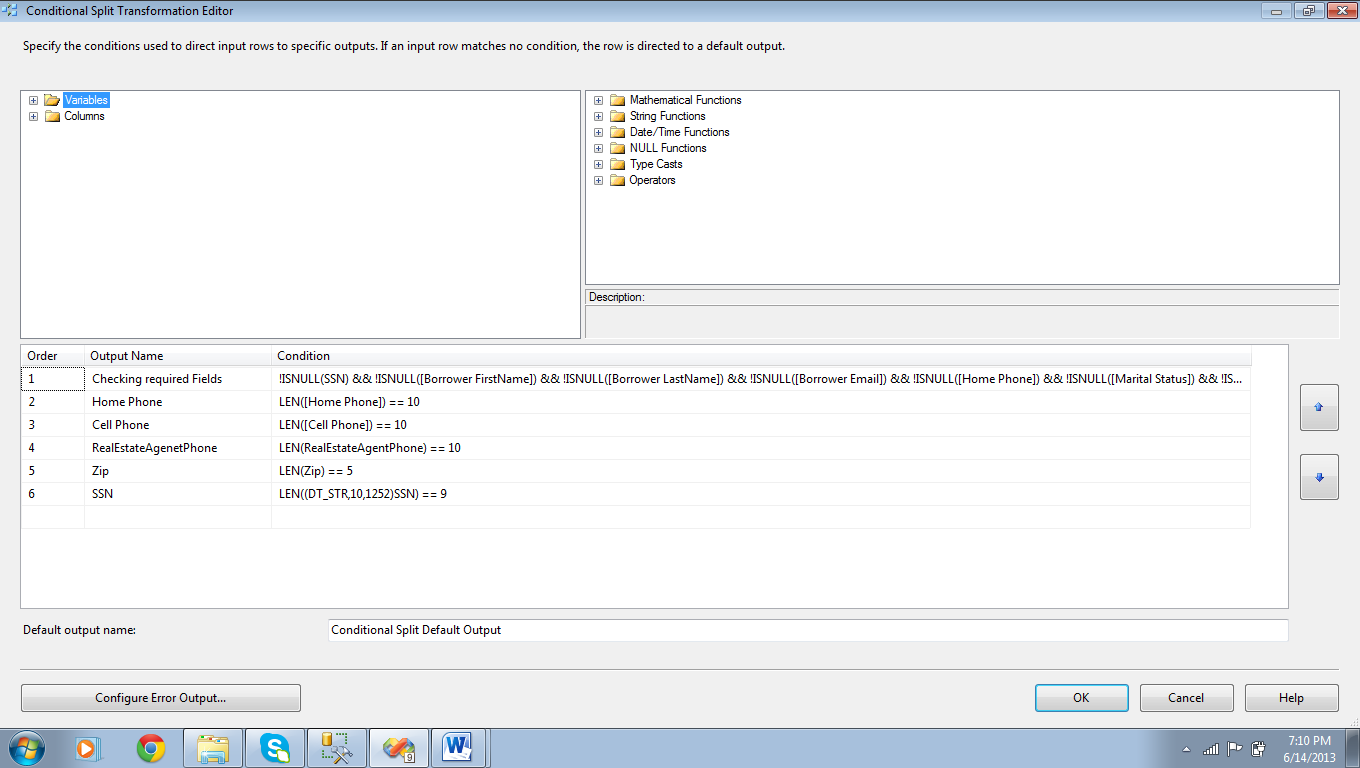
on d.SSN=c.SSN

on c.SSN=b.SSN

on b.SSN=a.SSN

1. Check preview and click ok.

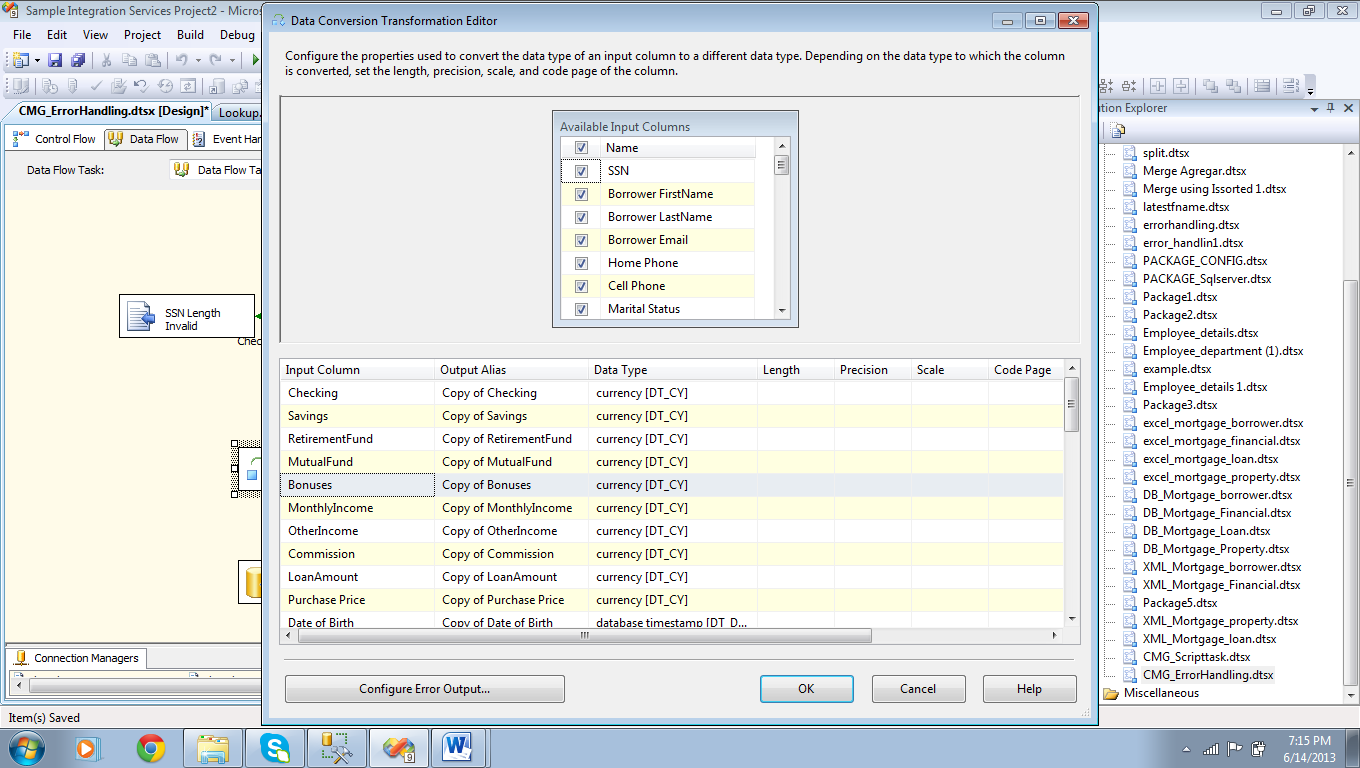
Now, Configure the Conditional split as shown in following figure:-



Add the above 6 conditions to do data validations.

Attach the precedent constraint of checking required fields to Data Conversion transformation and edit it.

Select all the available columns by click ok Name check box.

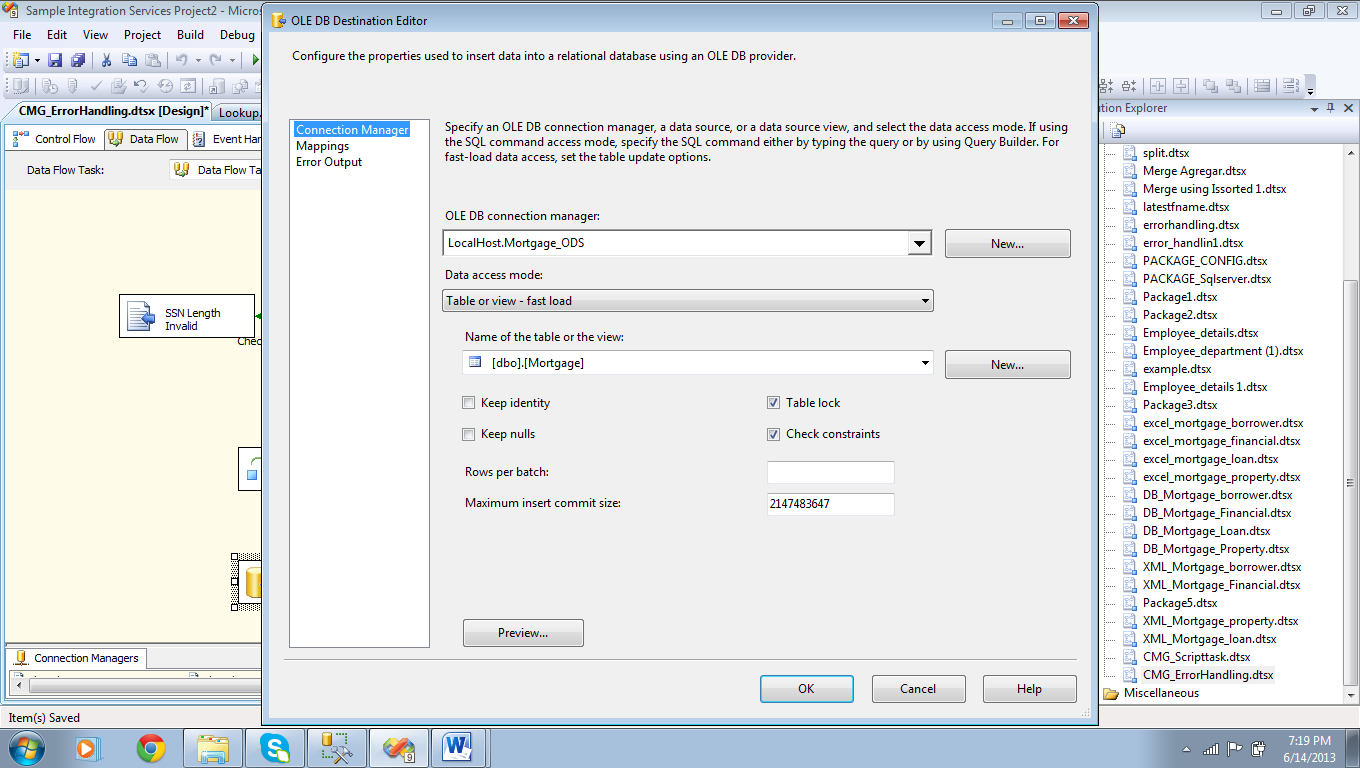


For each and every Input column configure output Alias name and Data type.

Now, click ok and save it.

Now, Load the data into final De-Normalized OLAP data base by using OLE DB Destination.

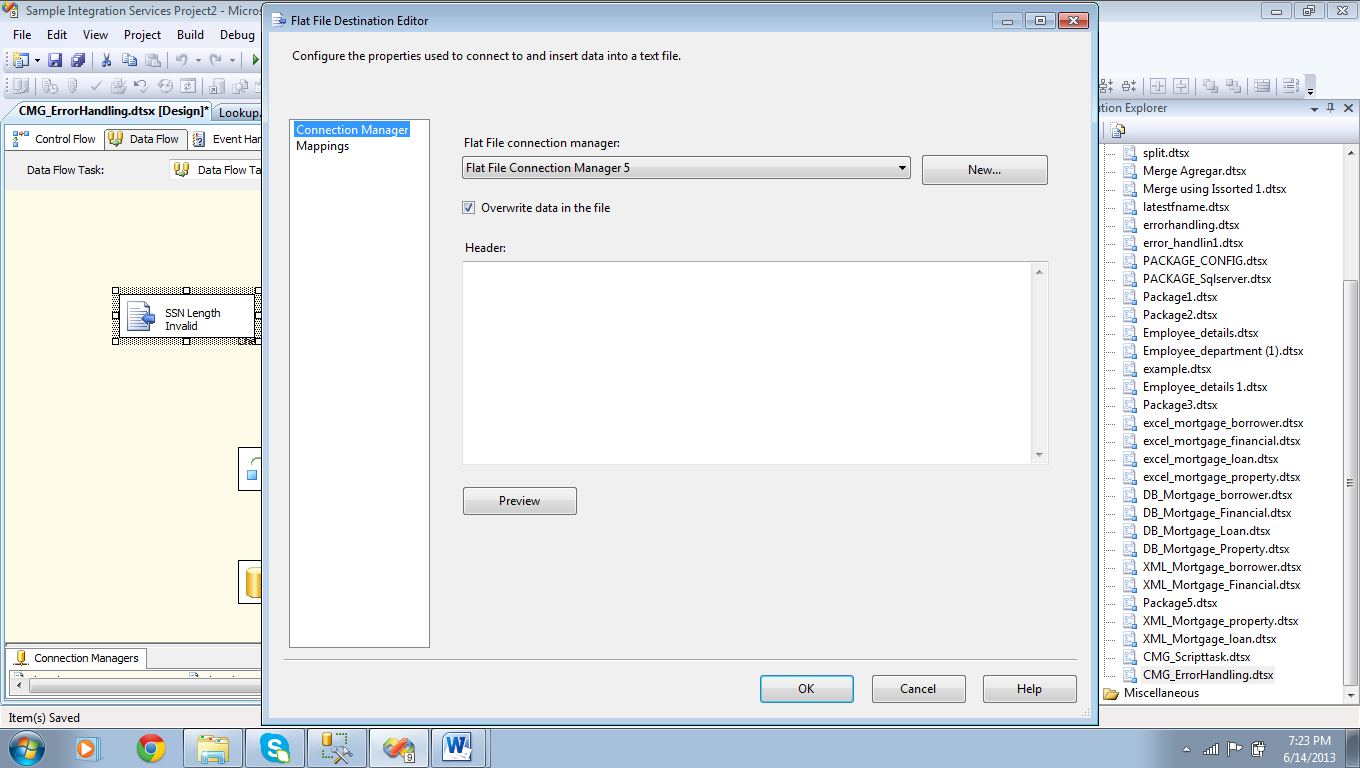
Configure the OLE DB Destination as shown,



Select the appropriate Database by creating a connection manager and select the table from the drop down list of “Name of the Table or the View” and click ok.

Finally configure all Error outputs to flat file destinations using appropriate precedent constraint to each of the Error case.

Configuring the FLAT FILE DESTINATION is depicted in the following figure:



Steps:

1. In Connection manager tab, create a new Flat file connection manager by browsing the file location using NEW button.
2. Attach appropriate Precedent from Conditional split.
3. Click ok and save it.