ETL Development with SQL Server Integration Services

Overview

This purpose of this lab is to give you a clear picture of how ETL development is done using an actual ETL tool. The tool we will use is called SQL Server Integration Services or SSIS. The goal is to help you get a feel for how the concepts you've learned in class apply in a real-world scenario with actual ETL tooling, instead of SQL.

Disclaimer!

As such it should be noted, this lab does not demonstrate everything you need to know about the ETL process, nor does it serve as training tool for SSIS. Both of these activities are beyond the scope of this course. Yes, when you're finished with the lab you'll know your way around SSIS, but more importantly you'll understand how the ETL process gets implemented using modern tooling.

Requirements

To complete this lab you will need the following:

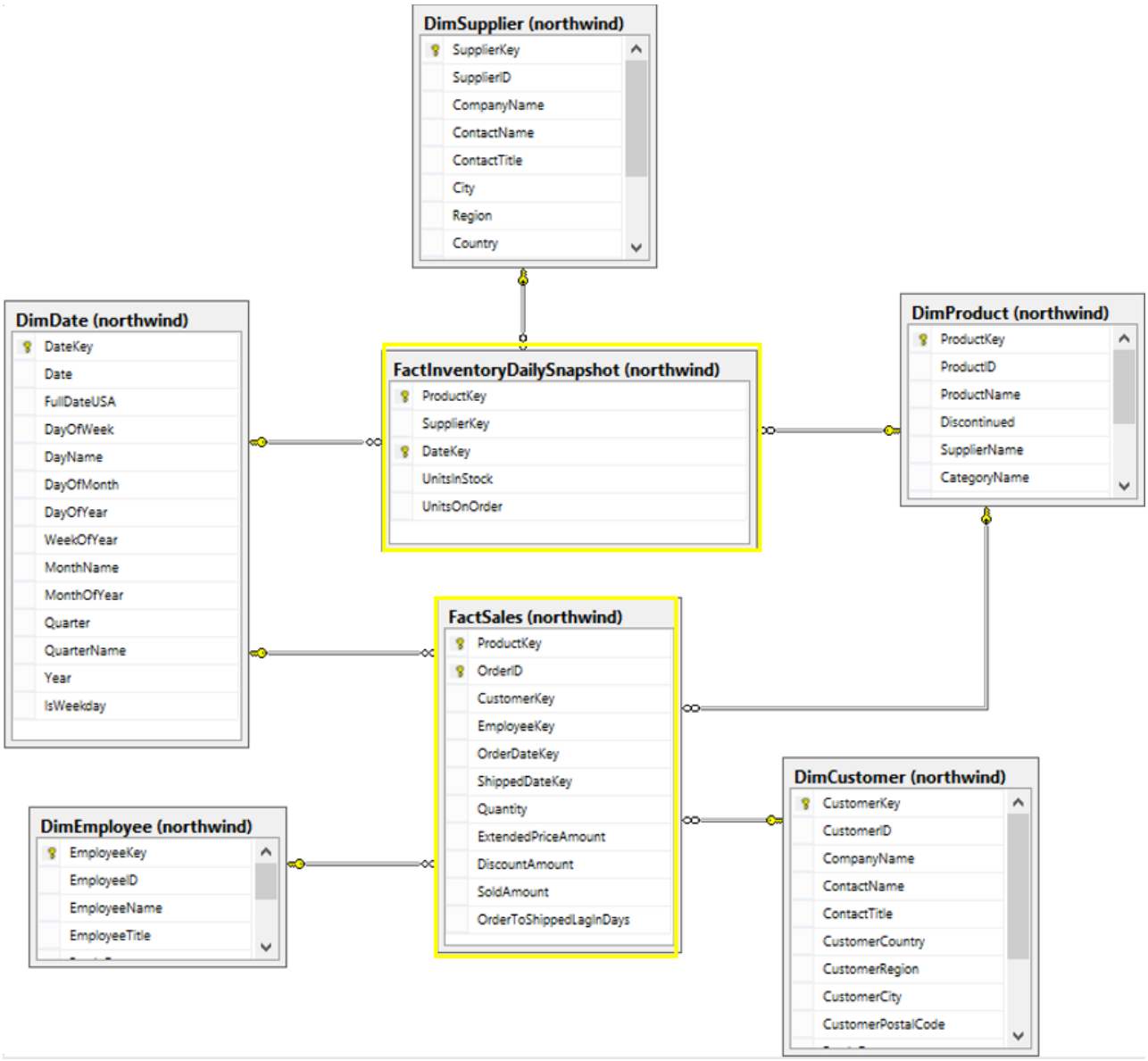
- A connection to the SQL Server 2014 using your IST722 Database Account. In SQL Server management studio 2014.
- Access to the SQL Server 2014 Data tools (used to author, SSIS packages).
- You should connect to your computer running your SQL server database before starting this lab.
- The **SSIS-Lab-Files.zip** file (in the same place you found this lab). There are two files you need from inside the zip file. Please copy them out of the zip file into some known location (eg. your home directory or desktop.) The two files:
    - **Northwind-ROLAP-Bus-Architecture.sql** file. This file will create the bus architecture of conformed dimensions across two business processes in Northwind: sales and inventory analysis. It will remove any other fact and dimension tables you have in the northwind schema.
    - **NorthwindDailyInventoryLevelsOneWeek.csv** file. This file simulates a weeks' worth of daily snapshots of changes to inventory levels.

The Northwind Data Warehouse

As you might recall, Northwind traders' management needs a data warehouse to track two business processes:

1. Daily Inventory Levels of Product
2. Product Sales Orders

In the file **Northwind-ROLAP-Bus-Architecture.sql** you will see SQL code to create the ROLAP star schemas for both of these business processes. You will also notice we are using a Kimball ***bus architecture*** as we are conforming (re-using) our dimensions across fact tables. The following figure give a clear picture of the target ROLAP schema:



Northwind Bus Architecture. Fact tables outlined in yellow.

Lab Breakdown

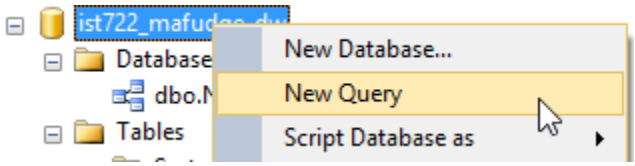
- In **Part 1** we get a feel for SSIS tooling, taking a quick, high-level tour of the product.
- In **Part 2** we will walk through the ETL load step-by-step for the first dimensional model, Daily Inventory Snapshots.
- In **Part 3** you will complete the ETL for Product Sales Orders on your own, with minimal guidance.

Before You Begin

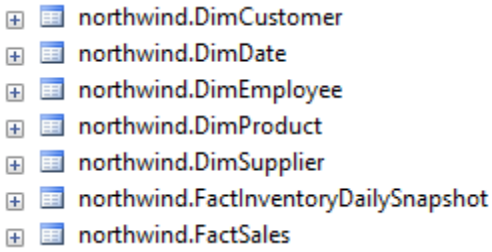
Before you begin the lab, you'll need to create the tables required for our schema:

1. Open SQL Server Management Studio 2014 and log-on to our data warehouse server.
2. Find your **_dw** database. For example, mine is **ist722_mafudge_dw**

3. Right-click on your database and select **New Query**… from the menu.



4. When the new query window is available, press **CTRL+O** to open a dialog window so you can browse for the **Northwind-ROLAP-Bus-Architecture.sql** file.
5. Once the file is loaded into the query window, execute it, by pressing **[F5]**
   IMPORTANT: The script is designed to automatically drop all the tables in your **northwind** schema, so if the script doesn't work, I suggest dropping all northwind.* tables manually.
6. You've completed this step correctly when you see ONLY these tables in the northwind schema of your **dw** database:



IMPORTANT: It is also advisable to drop all the tables in your **stage** database. This will need to be done manually.

Do not continue until you have completed these steps.

Part 1: SQL Server Integration Services (SSIS) Overview

In this part, we'll overview the SSIS tool, including how to launch the program and get around the basic user interface.

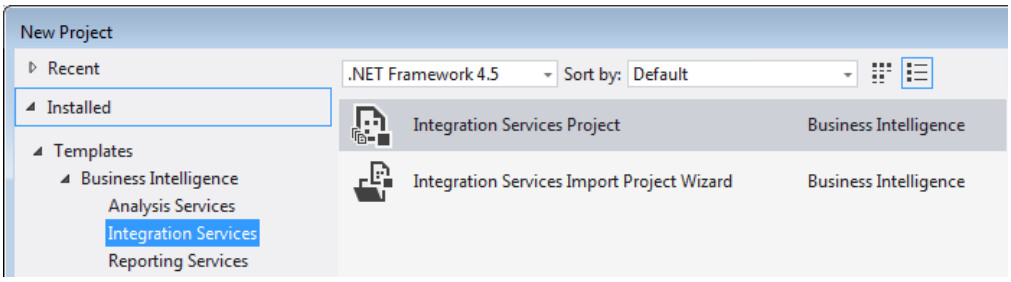
Launch SQL Data Tools

In order build our ETL solution with SSIS, first we will need to run SQL Data Tools. This is the main application for Integration and Analysis services. **DO THIS:**

1. Click the **Windows Start** button.
2. Click on the **SQL Server Data Tools for Visual Studio 2013** icon

3. After Visual Studio launches, from the menu, click **File → New → Project**

4. Select **Integration Services Project** from the New Project dialog. (Under Business Intelligence)



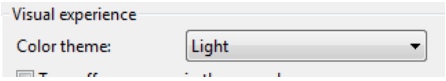
5. Don't bother to name the project, we're just exploring the SSIS features for now.
6. Click **OK**

The "Light" Theme

In the screenshots you'll see throughout this lab, I'm using the "Light" theme in Visual Studio. While I prefer the "Dark" theme, the light theme is more printer-friendly. All screenshot where taken with this theme configured so if you'd like to follow along, I suggest selecting the same theme.

To switch Visual Studio to the **Light** Theme:

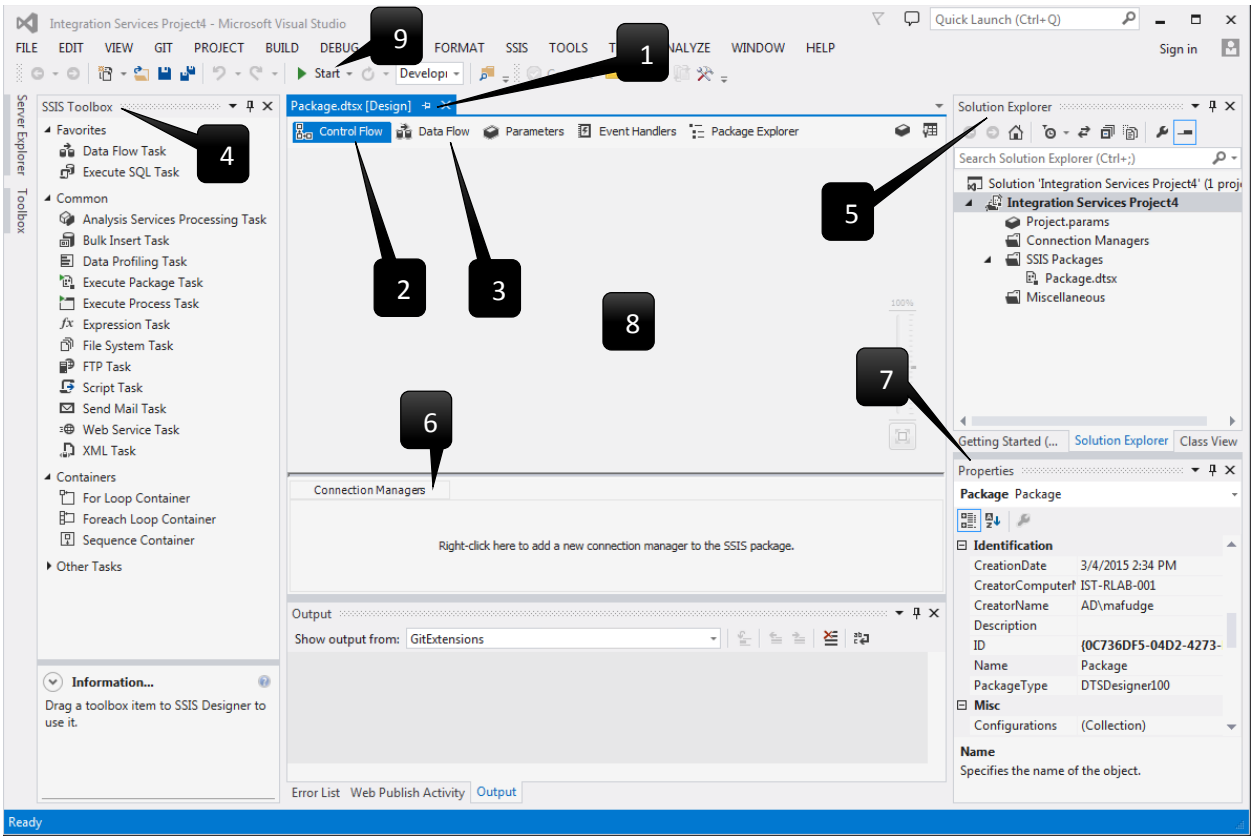
1. From the Menu, select **TOOLS → Options**
2. From the options dialog, select the **Light** color theme.



3. Click **OK** to save.

Getting To Know Your Way Around the SSIS User Interface.

Before we dive in and create our first package it would be helpful to give you a quick tour of the user interface. Each of the items we point to are explained below the screenshot.



1. SSIS consists of files called **Packages [1]** a package contains 1 **Control Flow [2]** with 1 or more **Data Flows [3]**. You create the package flows by dragging items from the **SSIS toolbox [4]** onto the main **package design surface [8]**.
2. The **Control Flow [2]** tab is your main workspace for control flows. **Control Flows** outline a set of tasks which should be carried out by the package. SSIS is as a visual programming language and you can setup tasks to run sequentially, or concurrently and based on various conditions.
3. The **Data Flow [3]** tab is your workspace for **Data Flows**. A **Data Flow** represents a visual transformation of data from a data **source** to a data **target**. Your data source can be just about anything, another DBMS, a text file, an XML file, a web service, etc. Your data target is typically a file or DBMS.
4. The **SSIS Toolbox [4]** contains different elements you can add to your design surface depending on whether you are in the Control Flow or Data Flow tab. These elements represent the ***steps*** in your SSIS program.
5. The **Solution Explorer [5]** is a collection of everything used as part of your project. It contains one solution file containing many of your **SSIS Packages [1]**, the **global connections [6]** used by the packages, and any **global variables** required to make the solution work.
6. The **Connection Managers [6]** tab displays the data source connections available to your package.
7. The **Properties [7]** window is where you can change the characteristics of the tasks you've selected in your design surface.
8. So In conclusion SSIS is a visual programming language. You main work consists of
    a. dragging tasks from the **SSIS Toolbox [4]** onto the **design surface [8],**

   b. double-clicking on the task to configure it and
   c. Connecting tasks together to execute in sequence.
9. When you want to execute your package, you click on the **Start [9]** button in the toolbar to place the package in run mode.

Part 2: Walk-Through ETL Solution for Inventory Daily Snapshot

Now that you have basic knowledge of the SSIS tooling, we'll walk through the creation of the ETL solution for populating the Inventory Analysis dimensional model. In this model we're taking a "daily snapshot" of inventory and order levels in order to track trends. Like most operational data, the Northwind database does not keep a history of inventory, but only current levels. This is why a dimensional model like this one is so desperately needed - to add time variance to our operational data!

Here's a high-level breakdown of the process:

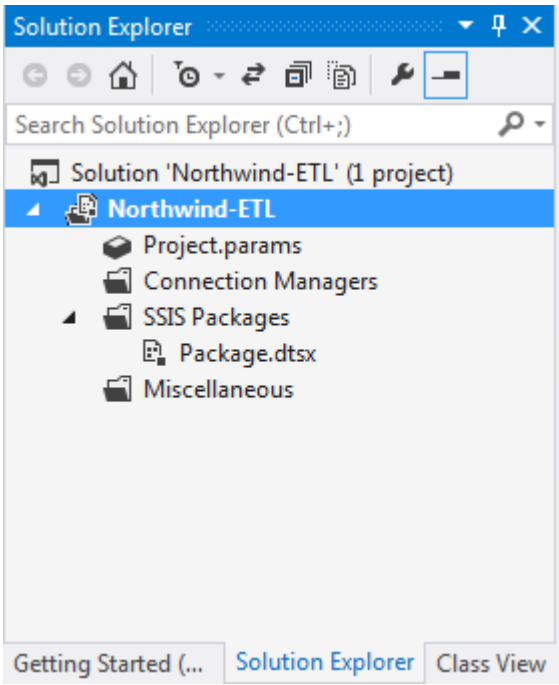
- In this section, we will establish our SSIS project and create the necessary connections.
- In Part 2.1, we take a nice, gentle introduction to SSIS by creating a package to populate the Date Dimension, creating the package: **DateDimensionImport.dtsx**
- In Part 2.2 will create a package called **Stage_InventoryLevels.dtsx** to stage our external world data into our stage database. We will stage the data "as-is" and use the "truncate and load" pattern.
- Part 2.3, we'll create a package called **DW_InventoryLevels.dtsx** to complete the ETL from our stage database into our dw database. Along the way we will perform the data transformations required by our target dimensions and fact tables to load the data properly. Also, we'll use the SCD (Slowly changing dimension) pattern to determine whether we need to "pay attention" to the dimension or fact and thereby avoid loading the same data more than once.

Create Our Solution and Project

Before you start, let's make an official **Northwind ETL** project and solution.
DO THIS:

1. From the Visual Studio menu, select **FILE → New → Project.**
2. From the dialog:
   a. Choose **Integration Services Project**.
   b. Name: **Northwind-ETL**
      NOTE: Pay attention to the path where you solution is saved. You'll need to get it later. Click the **Browse…** button to select a different folder.
3. Click **OK** when you're ready.

4. Your solution explorer should look like this:

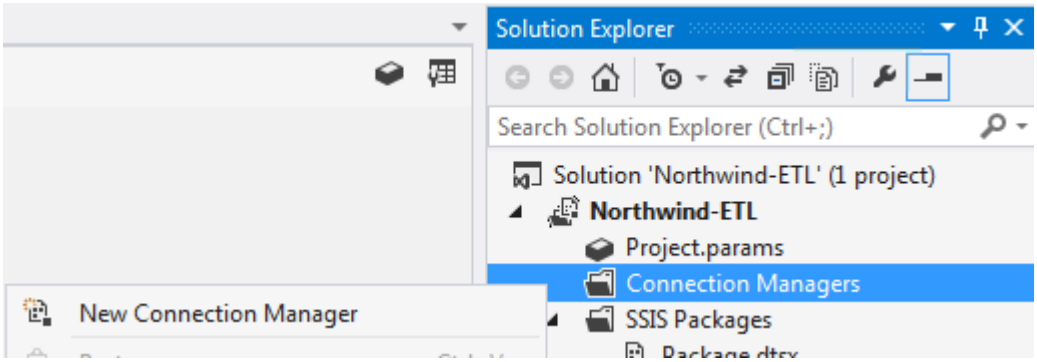


NOTE: if you do not see the solution explorer window. Press **[CTRL] + [ALT] +[L]** to bring it up.

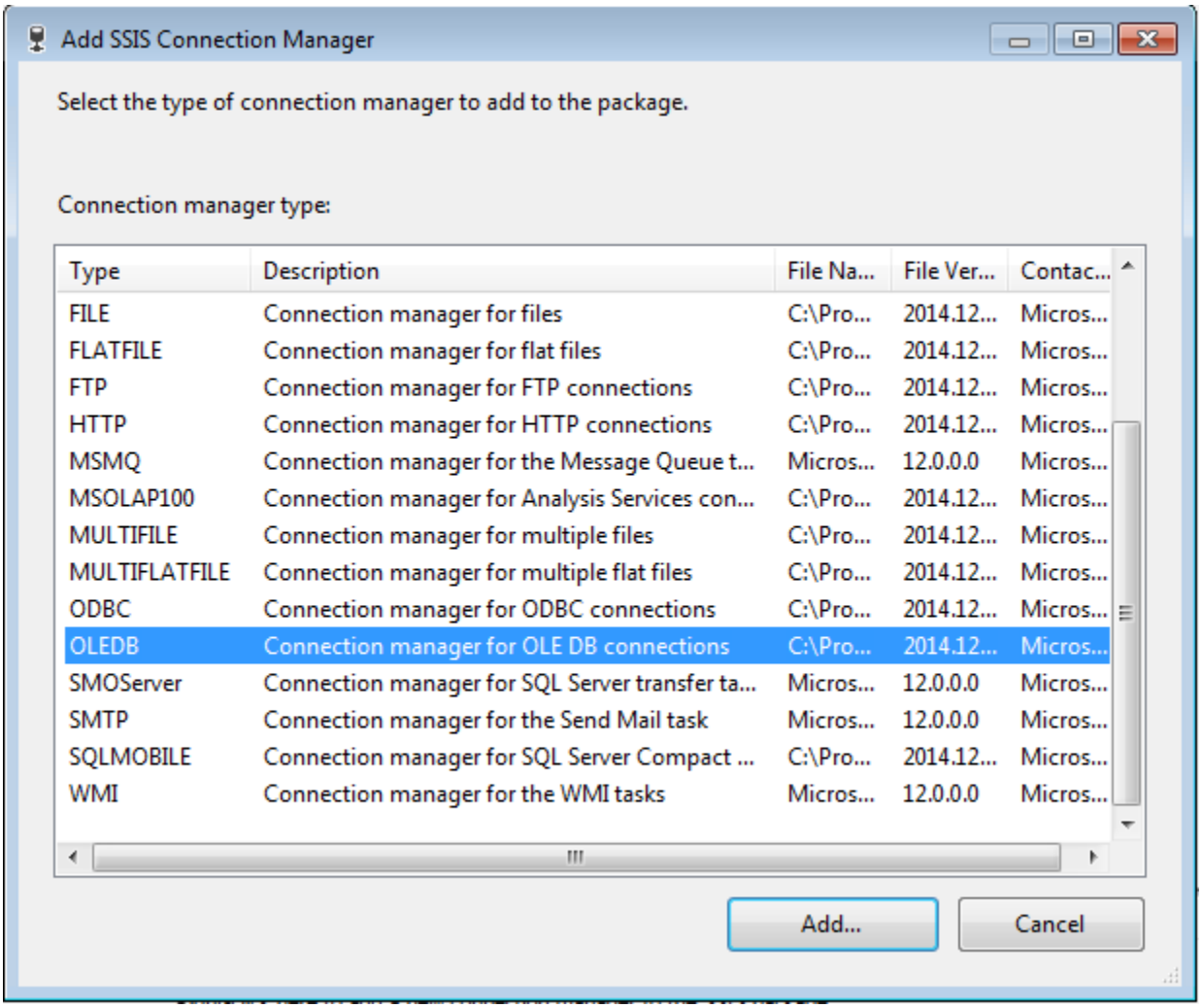
Establish Connections Used By the Project

In order to access external data for ETL we will need to establish connections to our sources and targets. While you could create these connection inside each package, the better approach is to create these connections globally inside the **Northwind-ETL** project, so they can be re-used by other packages. Let's do this now.

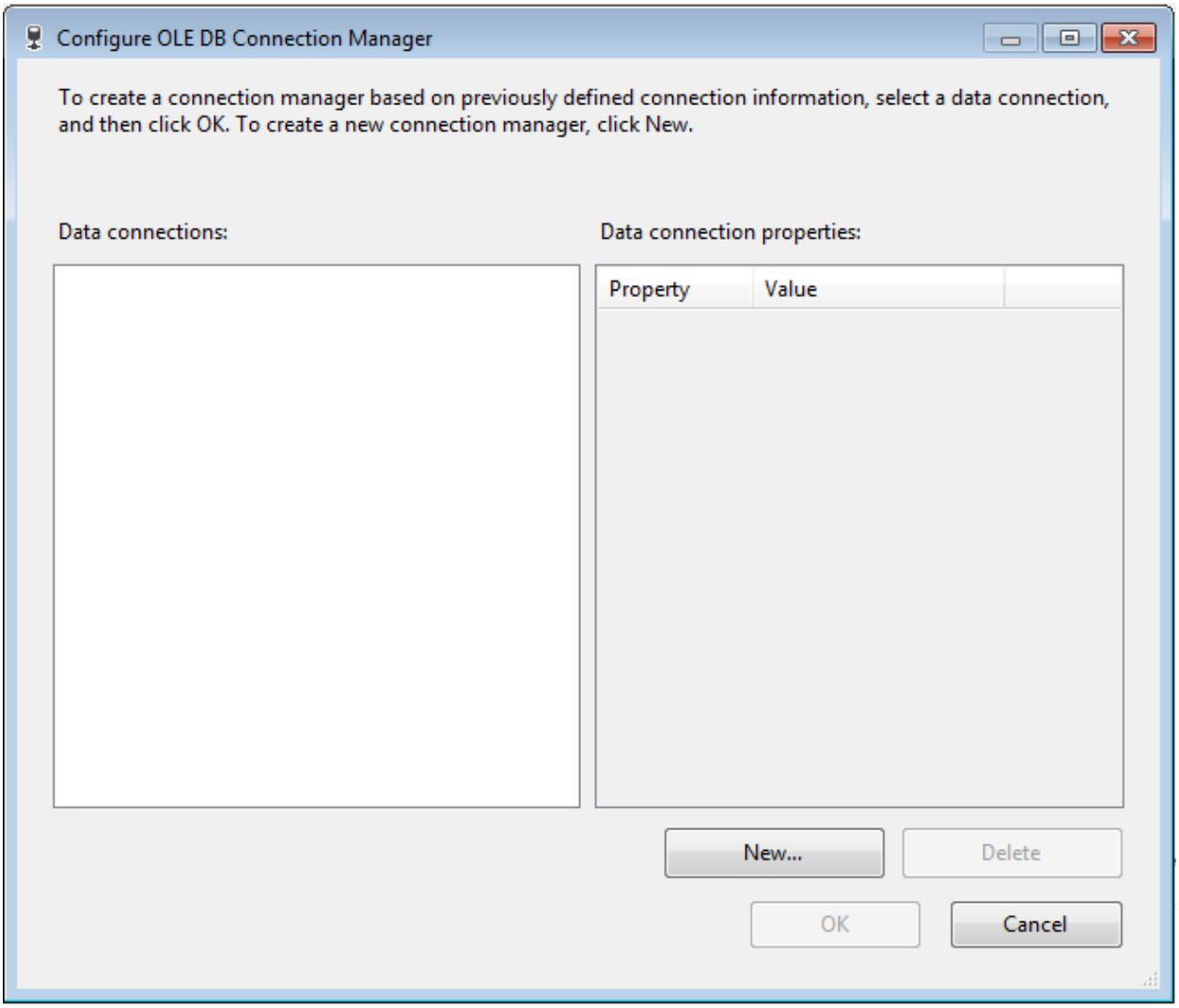
1. First, in solution explorer, right-click on **Connection Managers** and select **New Connection Manager** from the menu.



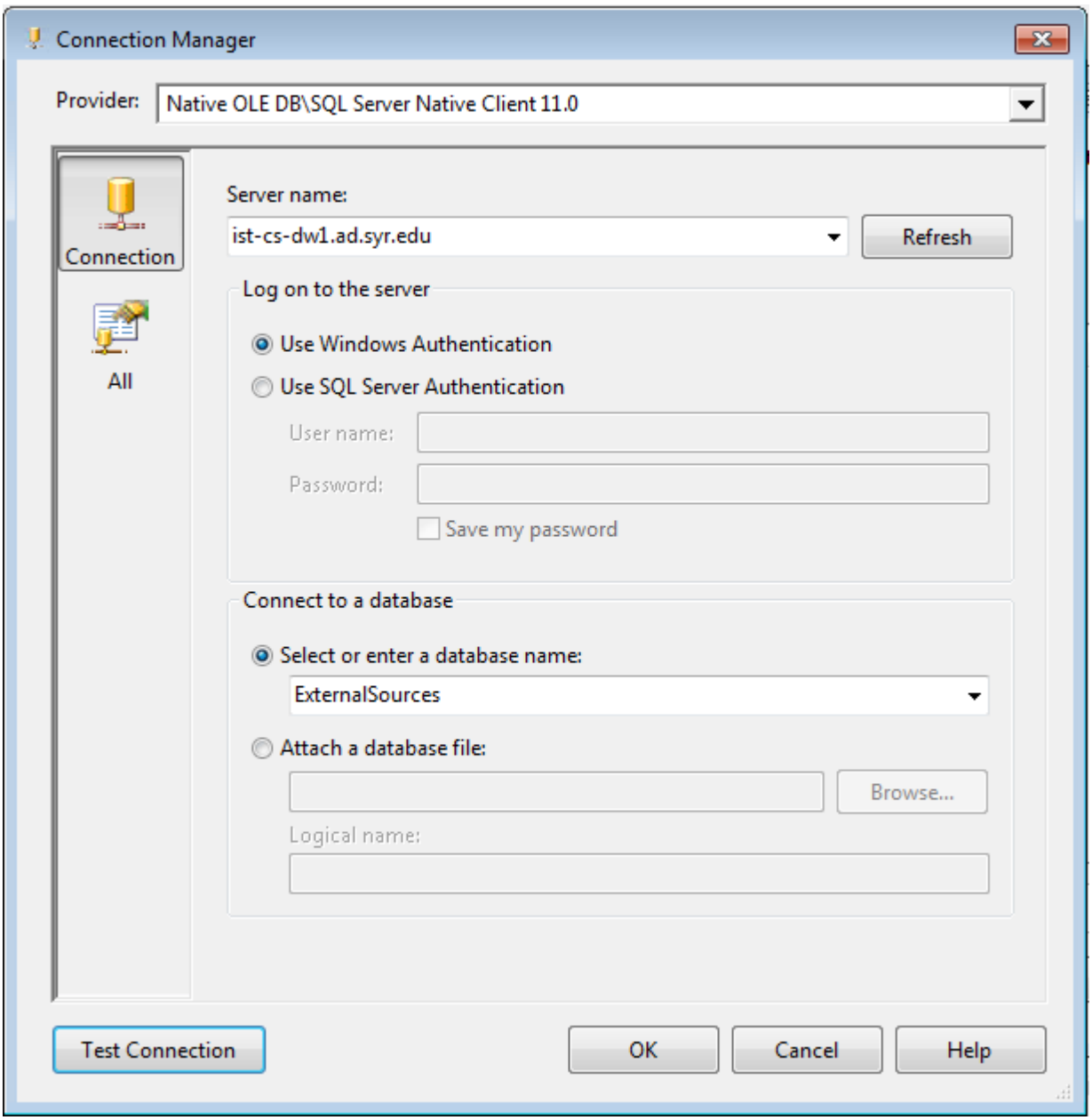
2. This brings up the **Add SSIS Connection Manager** dialog. There are many to choose from, and not all are for data connections. For most database-type connections we will use OLEDB and ODBC as a fallback when there is no OLEDB driver available. Choose **OLEDB** and click **Add…**


Add SSIS Connection Manager
Select the type of connection manager to add to the package.
Connection manager type:
Type
Description
File Na...
File Ver...
Contac...
FILE
Connection manager for files
C:\Pro...
2014.12...
Micros...
FLATFILE
Connection manager for flat files
C:\Pro...
2014.12...
Micros...
FTP
Connection manager for FTP connections
C:\Pro...
2014.12...
Micros...
HTTP
Connection manager for HTTP connections
C:\Pro...
2014.12...
Micros...
MSMQ
Connection manager for the Message Queue t...
Micros...
12.0.0.0
Micros...
MSOLAP100
Connection manager for Analysis Services con...
C:\Pro...
2014.12...
Micros...
MULTIFILE
Connection manager for multiple files
C:\Pro...
2014.12...
Micros...
MULTIFLATFILE
Connection manager for multiple flat files
C:\Pro...
2014.12...
Micros...
ODBC
Connection manager for ODBC connections
C:\Pro...
2014.12...
Micros...
OLEDB
Connection manager for OLE DB connections
C:\Pro...
2014.12...
Micros...
SMOServer
Connection manager for SQL Server transfer ta...
Micros...
12.0.0.0
Micros...
SMTP
Connection manager for the Send Mail task
Micros...
12.0.0.0
Micros...
SQLMOBILE
Connection manager for SQL Server Compact ...
C:\Pro...
2014.12...
Micros...
WMI
Connection manager for the WMI tasks
Micros...
12.0.0.0
Micros...
Add...
Cancel

3. Next you will see the **Configure OLE DB Connection Manager** dialog. This dialog will allow you to add connections to database servers. Click the **New…** button.

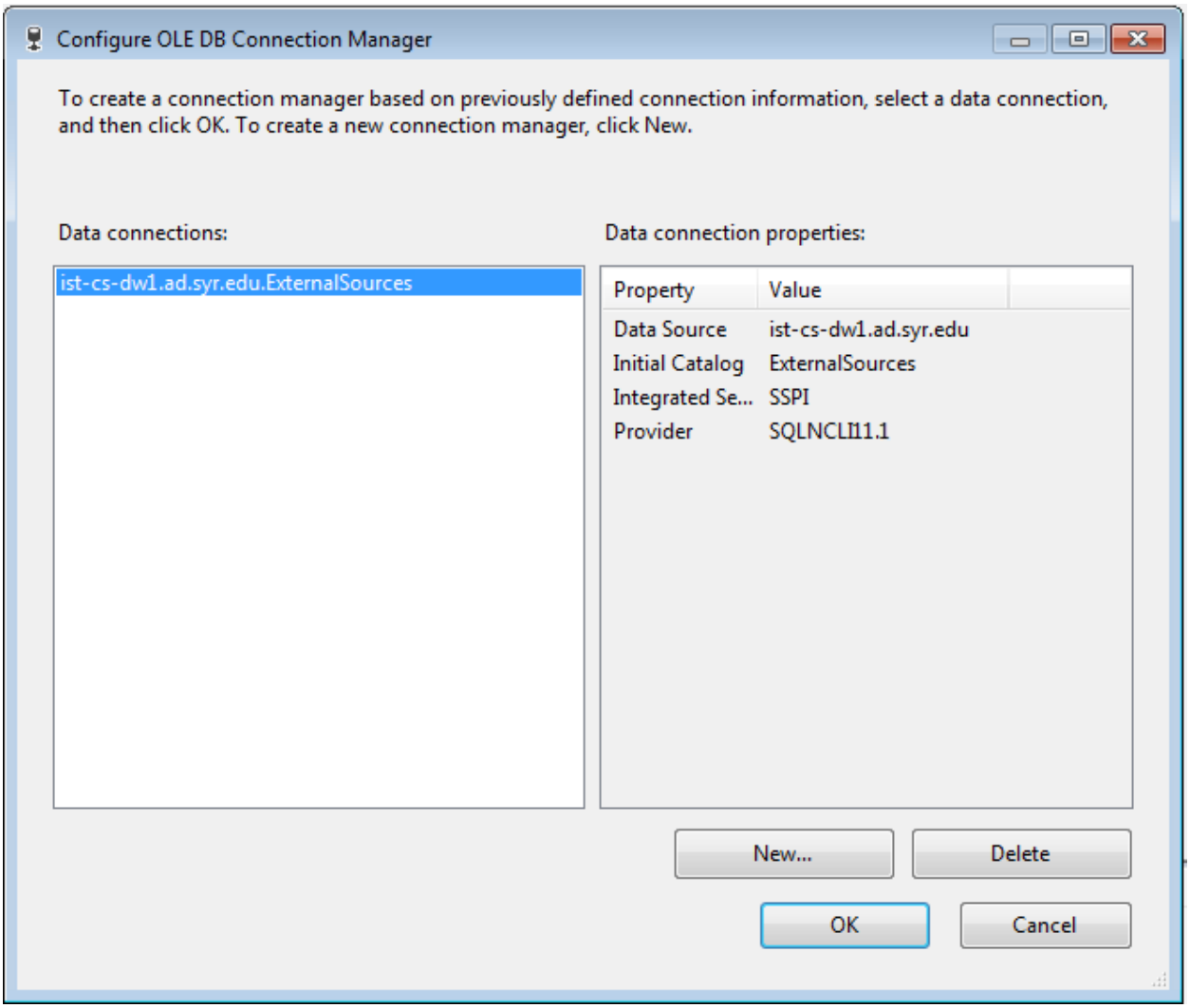


4. You will now see a **Connection Manager** dialog. In here, you need to choose the following options:
   Provider: **SQL Server Native Client**
   Server Name: **ist-cs-dw1.ad.syr.edu**
   Log on to the Server: **Use Windows Authentication**
   Connect to a database: **ExternalSources**

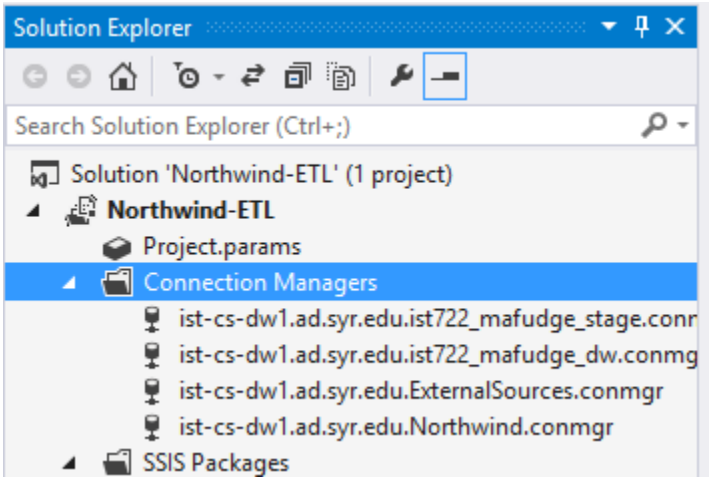


When you've got the information filled in, Click **Test Connection** first to verify the connection works. After you get the connection to work, click **OK** to save the connection.

5. You will now return to the **Configure OLE DB Connection Manager** screen, and should see the data connection to **ist-cs-dw1.ad.syr.edu.ExternalSources** that we just created:

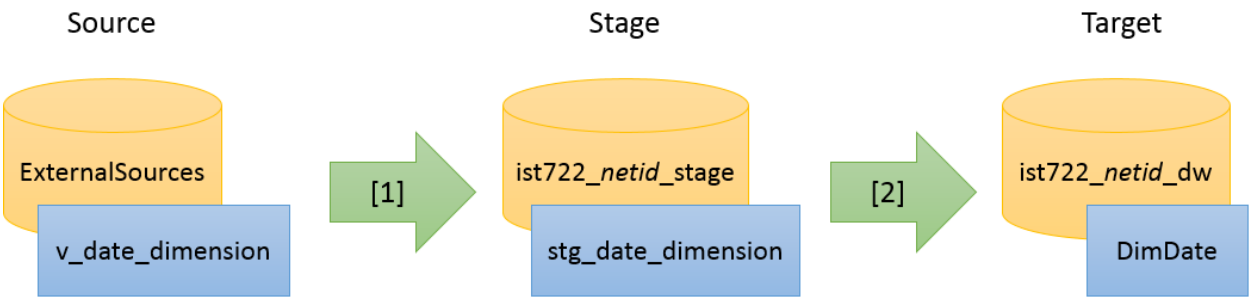


6. Select the ExternalSources connection and click **OK** to add the connection to the project.
7. Let's repeat this process (Steps 1 through 6) three more times for your **ist722_*netid*_dw** and **ist_*netid*_stage** databases and for the **Northwind** source database respectively. Here's an example showing all four connections you'll need in your **Connection Managers** folder.



Part 2.1: Importing the Date Dimension

First, let's get a feel for the power and capabilities of SSIS by populating our date dimension. As you may recall, the Date Dimension has been pre-determined, is not based on business data, and will only need to be populated into the data warehouse one time. It's really hard to do ETL without a plan, so for each step we will provide an outline of the steps in terms of a **source to target map.** Here's the source to target map for our date dimension:



In the diagram the green arrows represent SSIS tasks as data is transformed from source to target.

[1] First data is copied from the **v_date_dimension** in the **ExternalSources** database to a table called **stgDate** in your **stage** database. As you may recall from the previous lab we accomplished this with a SELECT INTO statement. Here we will use SSIS to create the destination table. We will also truncate the table before staging the data.

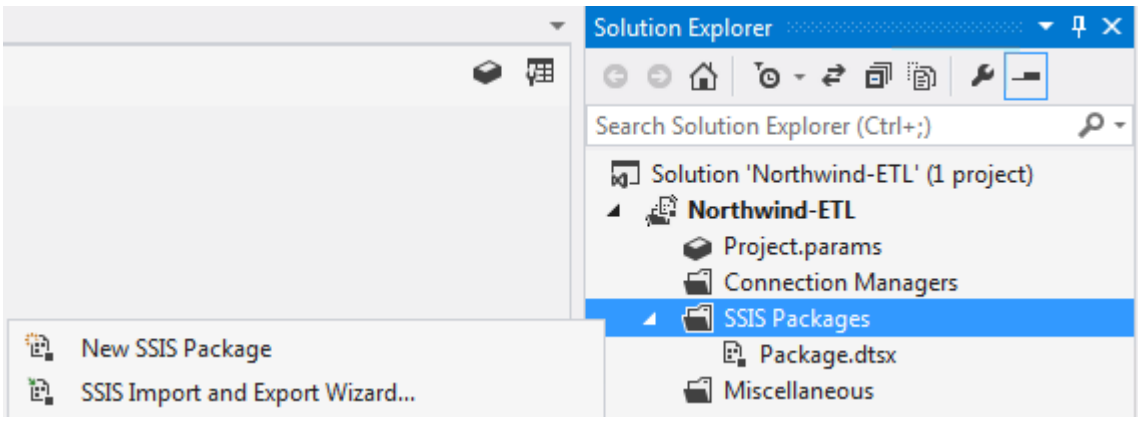
[2] Next after we complete the stage we will map columns from the **stgDate** source into the column names used by **DimDate** in the **dw** database. We'll apply type 1 SCD rules to ensure we do not re-import the same data more than once.

Let's get started!

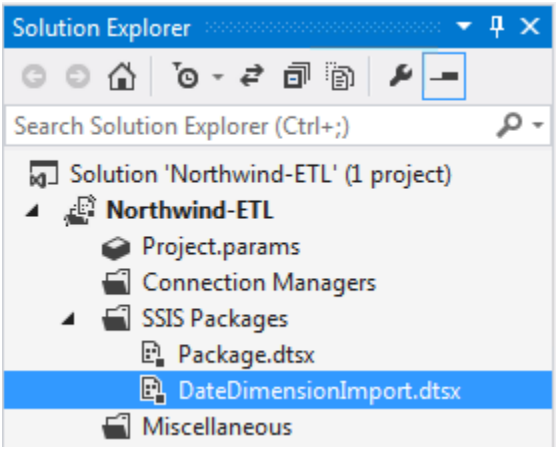
Step 2.1.1: Create the Package

First, let's create a new package, called **DateDimensionImport** to contain our ETL program.

1. In **Solution Explorer**, Right-click on **SSIS Packages** and select **New SSIS Package**
   (**NOTE**: If you do not see **Solution Explorer** open it from the **VIEW** menu.)



2. Right-Click on the package named **Package1.dstx** and choose **Rename** from the menu
3. Name the package **DateDimensionImport**
4. Verify you did this correctly. You should see two packages in your solution, **Package.dtsx** and **DateDimensionImport.dtsx**

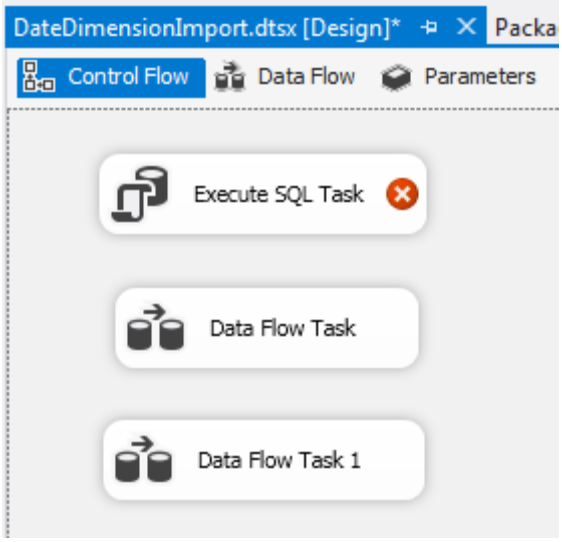


5. Finally, double-click on the **DateDimensionImport.dtsx** package to open it up so we can place tasks on the design surface.

Step 2.1.2: Setup Control Flow

Next, let's setup the control flow for this package.

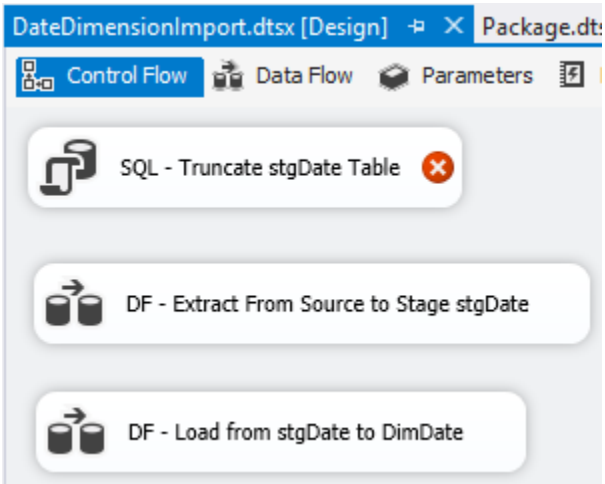
1. From the SSIS Toolbox drag and drop 1 **Execute SQL task** and 2 **Data flow tasks** on to the work surface. Like this:



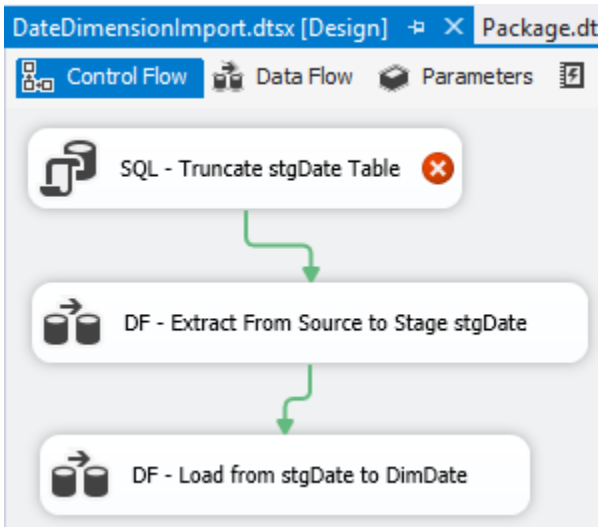
HELP!: Can't find the SSIS Toolbox? Try Menu → SSIS → SSIS Toolbox.

2. Next rename each task. To do this click on each task and press **[F2]** Name the tasks:
   a. Execute SQL Task → SQL - Truncate stgDate Table
   b. Data Flow Task → DF - Extract From Source to Stage stgDate

c. Data Flow Task 1 → DF - Load from stgDate to DimDate
d. Press **CTRL + S** to save your work. Saving often is a good thing!
Here's a screenshot:



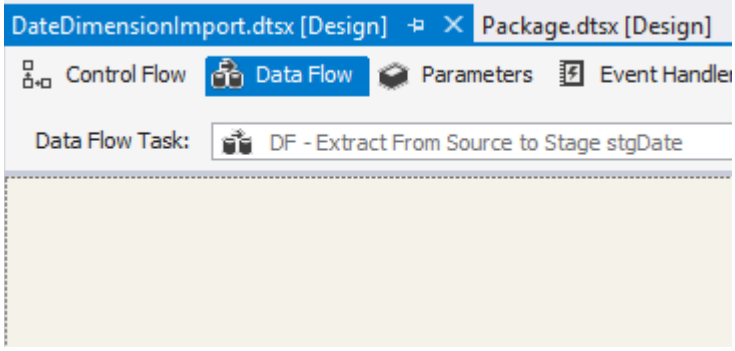
3. Next, Connect the tasks to that they run sequentially. Click on the **SQL - Truncate stgDate Table** task and a green arrow will appear. Drag the arrow and drop it on the **DF - Extract From Source To Stage stgDate** task. Repeat this process to connect the last two tasks so they are configured to execute sequentially, then **CTRL + S** to save.

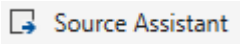


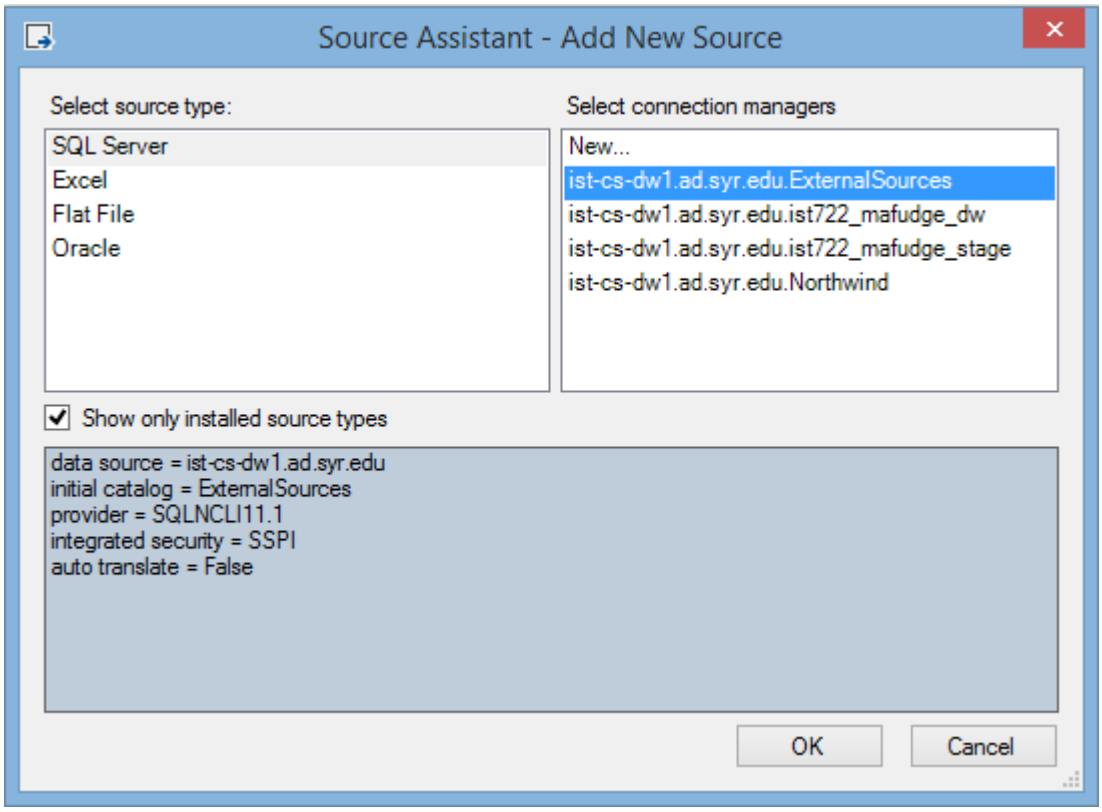
Step 2.1.3: Configure Source to Stage data flow

Now we need to configure each of these tasks. Let's start with the **DF - Extract From Source to Stage stgDate** task.

1. Double-click this task to open it in the Data Flow design surface:

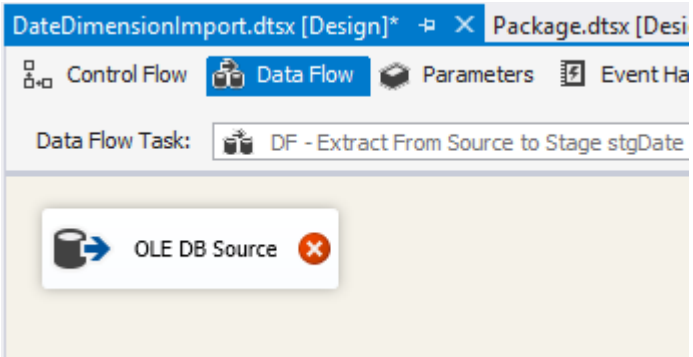


2. From the **SSIS Toolbox**, drag and drop the **Source Assistant**  to the design surface. This will open a dialog. Select source Type: **SQL Server** and the **ExternalSources** connection manager.

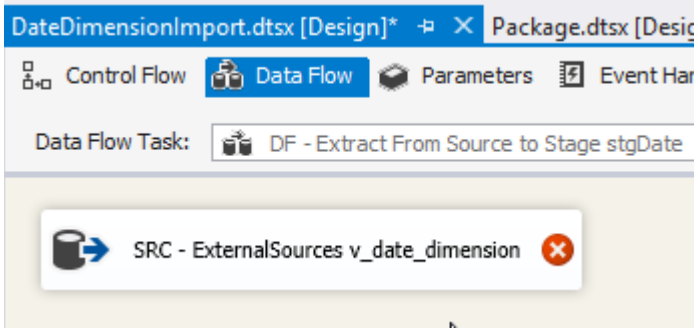


Click **OK** when ready.

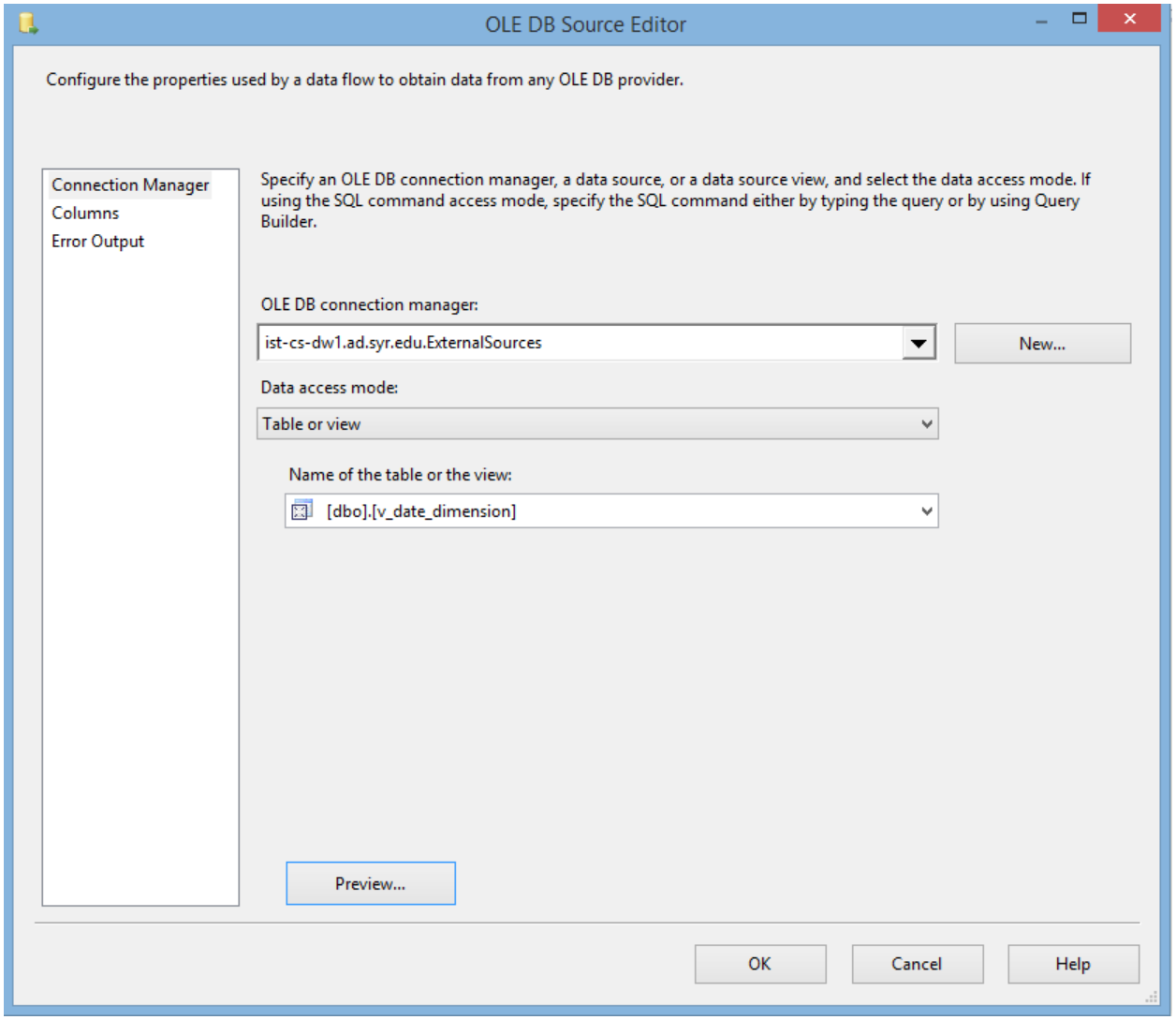
3. You should now have an unnamed and un-configured source on your design surface:



4. Click on the source and press **[F2]** to rename the source to **SRC - ExternalSources v_date_dimension**

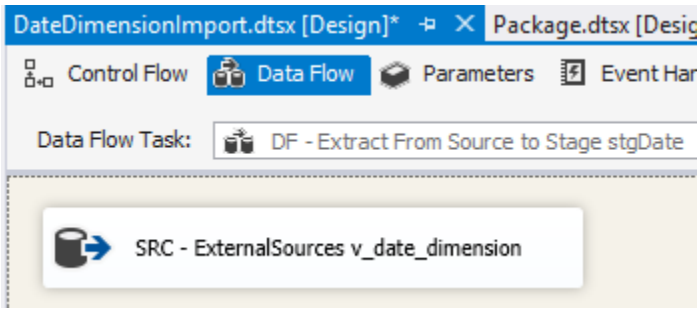


5. Double click on the source to configure it. This brings up the **OLE DB Source Editor**. From the drop-down for the **name of the table or view** select **[dbo].[v_date_dimension]** as the source. You can click **Preview…** to view the source data.

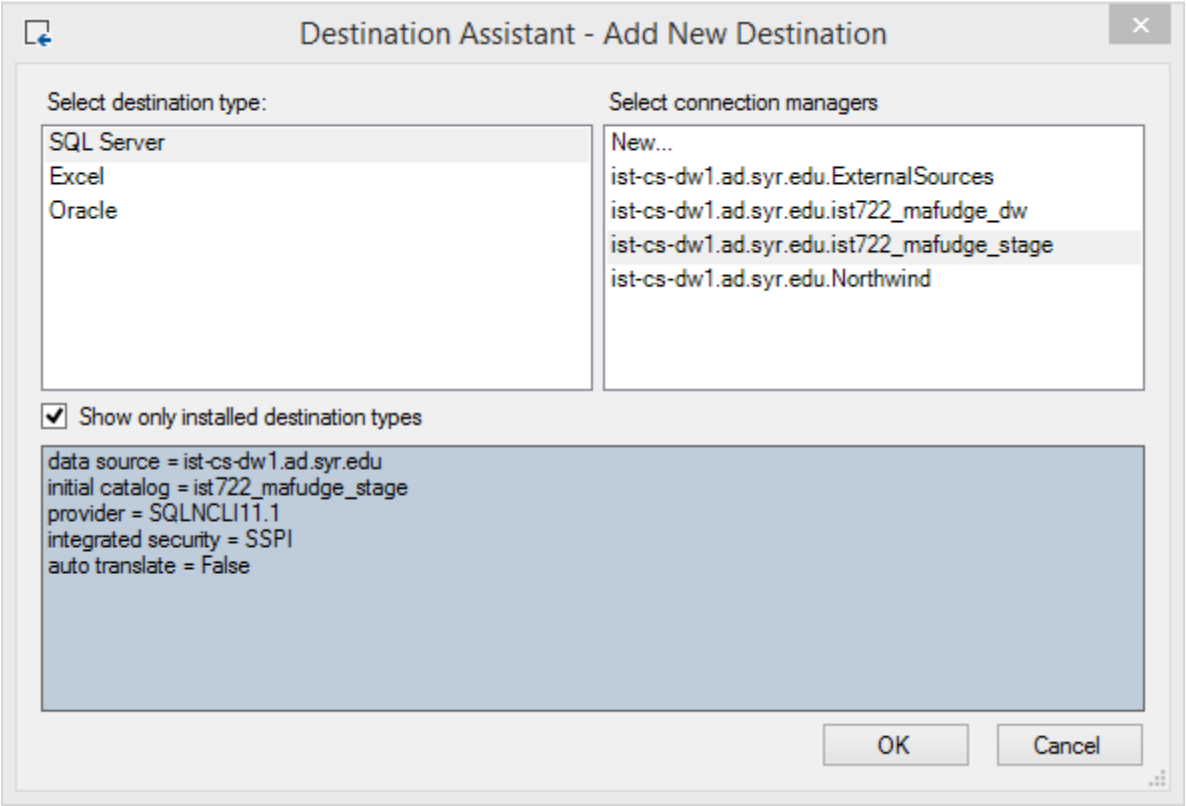


When you're finished click **OK** to save the source.

6. Your source is now configured. You can see this because there is no longer a red [x] error icon next to the source:

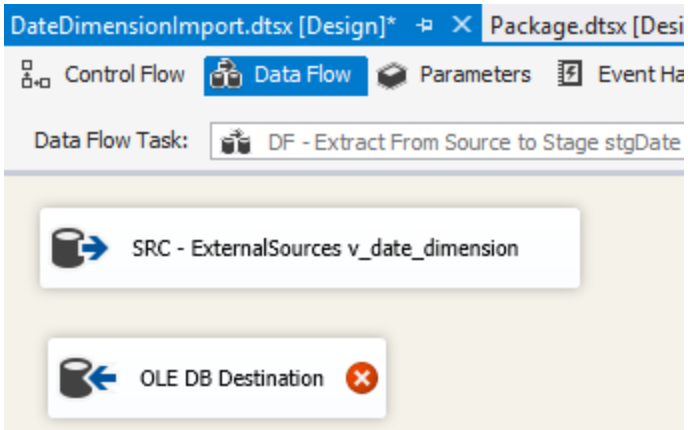


7. Next, we need to configure the destination. From the **SSIS Toolbox**, drag and drop the **Destination Assistant** to the design surface. This will open a dialog. Select source Type: **SQL Server** and the **ist722_*yournetid*_stage** connection manager.

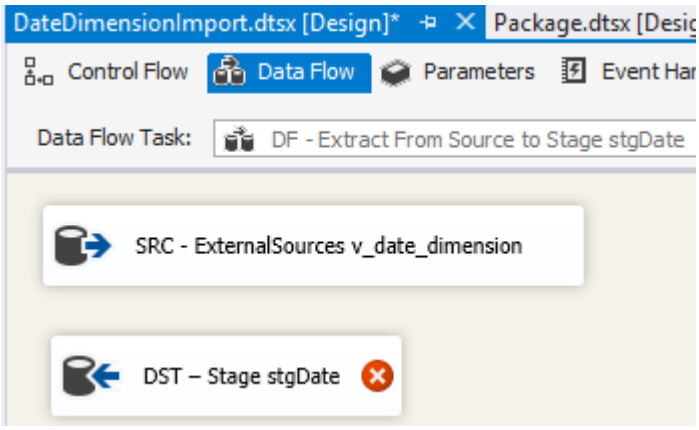


Click **OK** when you're ready.

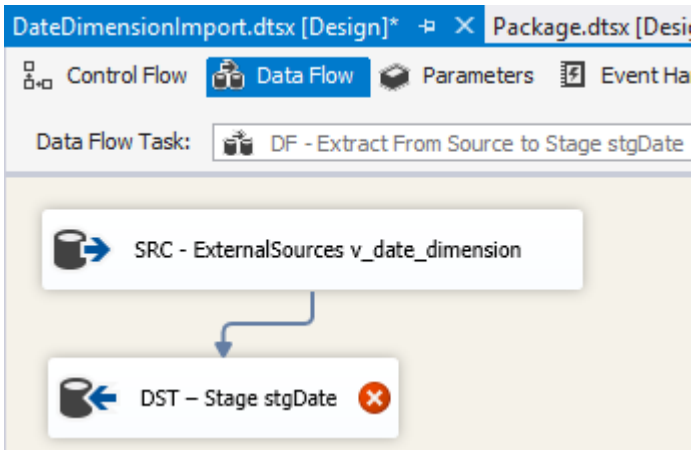
8. We now have an unnamed and un-configured *destination* on your design surface:



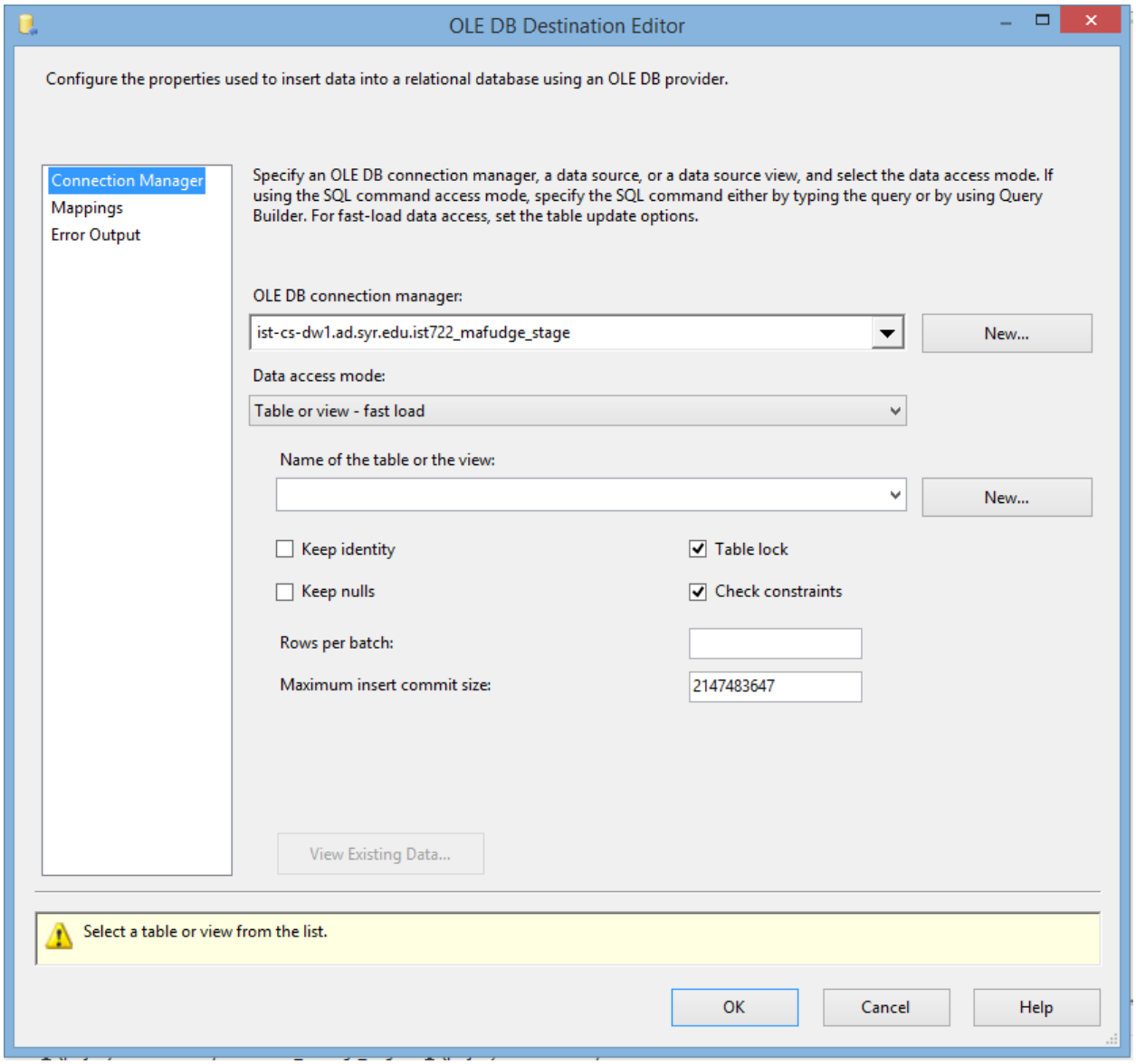
9. Click on the destination and press **[F2]** to rename it to **DST – Stage stgDate**



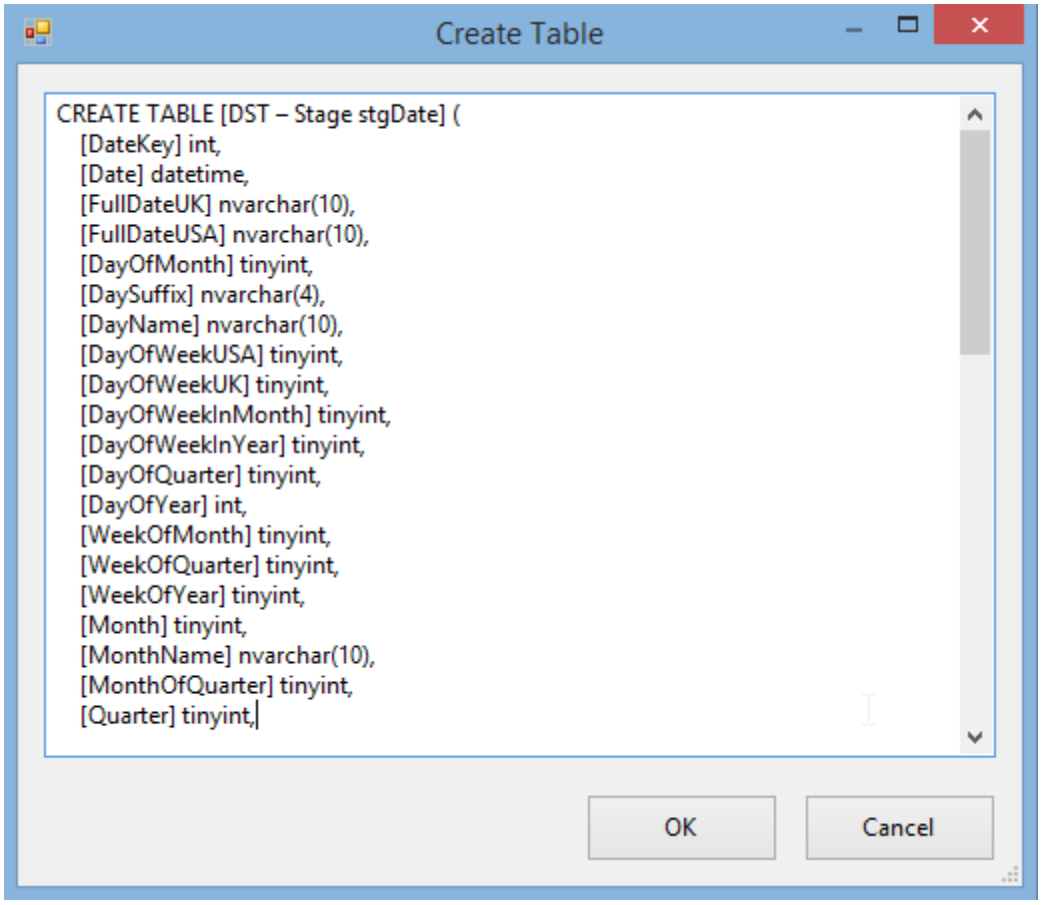
10. You probably haven't saved in a while. Press **CTRL + S** to save.
11. Now we need to take the ***output*** of the source and connect to the ***input*** of the destination. To do that we simply click on the source then drag and drop the blue arrow onto the destination.



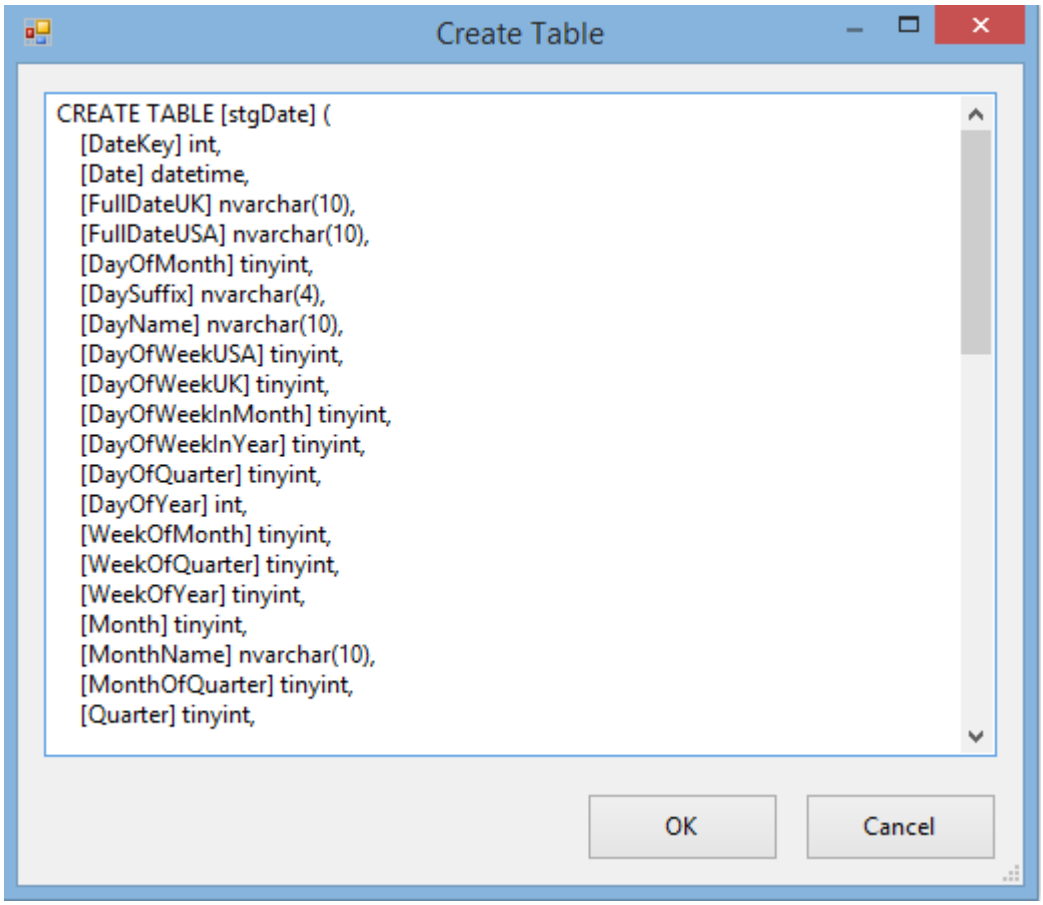
12. Finally, double click on the destination to configure it. This brings up the **OLE DB Destination Editor**.



13. Our destination table does not exist. What do we do? The good news is SSIS can create the table for us. Where does it get the information to create this table? SSIS looks at the schema of the source data and automatically generates a create table statement from it! Click the **New…** button to create the table in the stage database. You will see an SQL CREATE TABLE statement in a window titled **Create Table**.



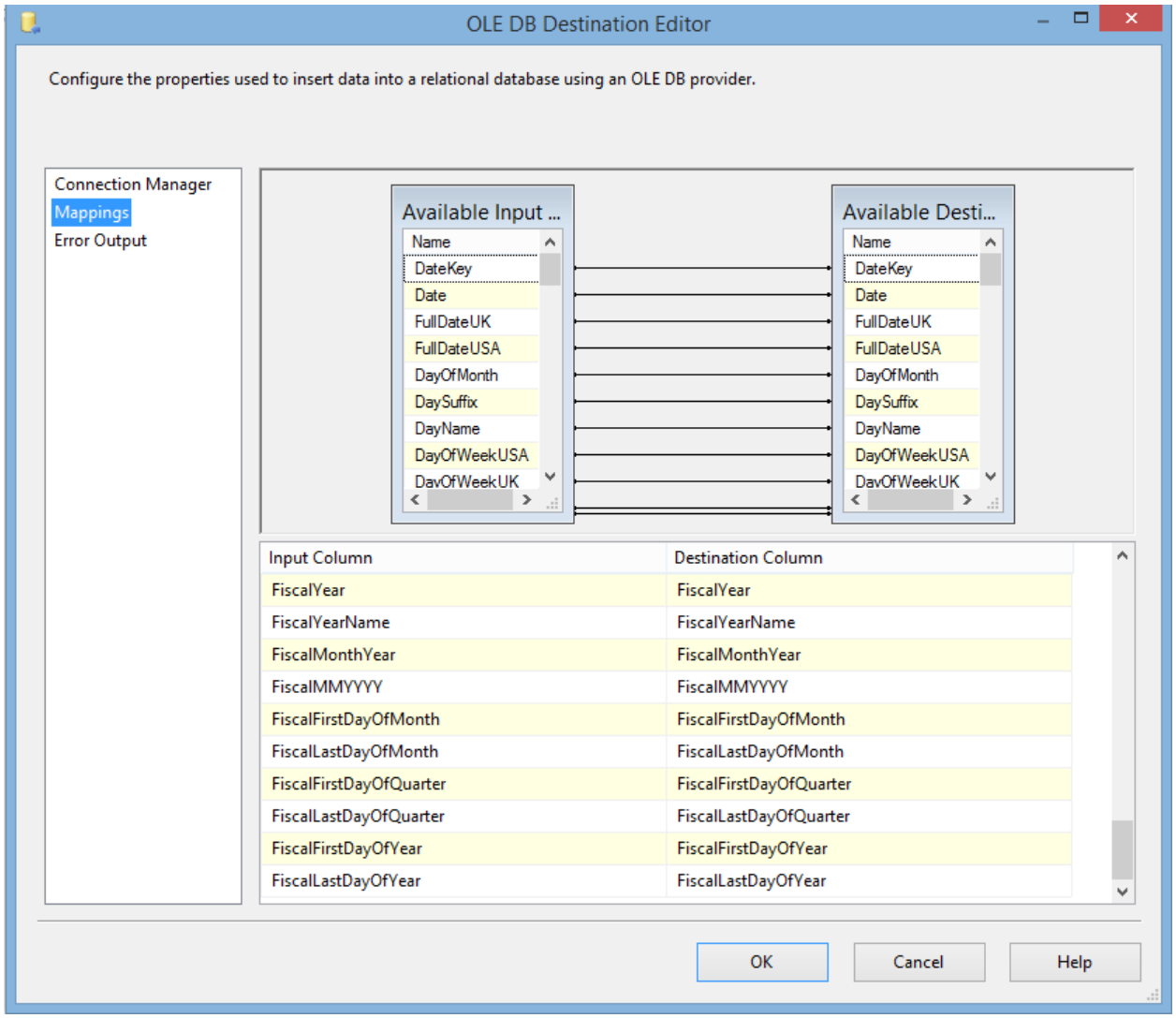
14. The only problem is the name of the table is wrong. Please change the first line of the statement to read:
    CREATE TABLE [stgDate] (



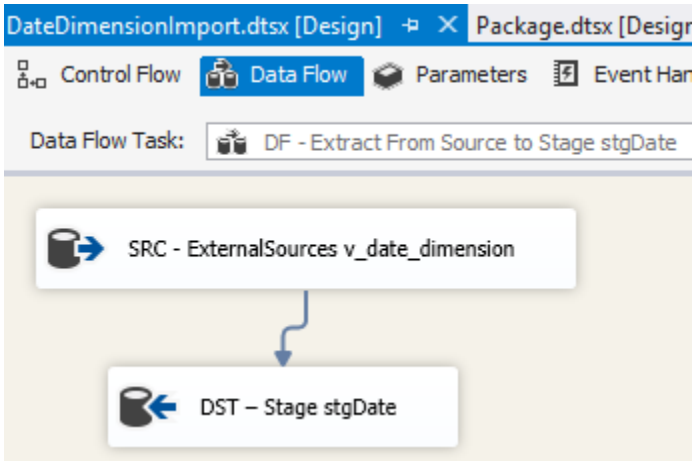
Then click **OK** to create the table

15. Back at the **OLEDB Destination Editor** the **Name of the table or view** should now be **[stgDate]**
16. There's just one step remaining. We need to configure the source to destination mapping. SSIS will perform this automatically when the column names match. In the left-hand menu, click on

Mappings to display the generated column mappings:



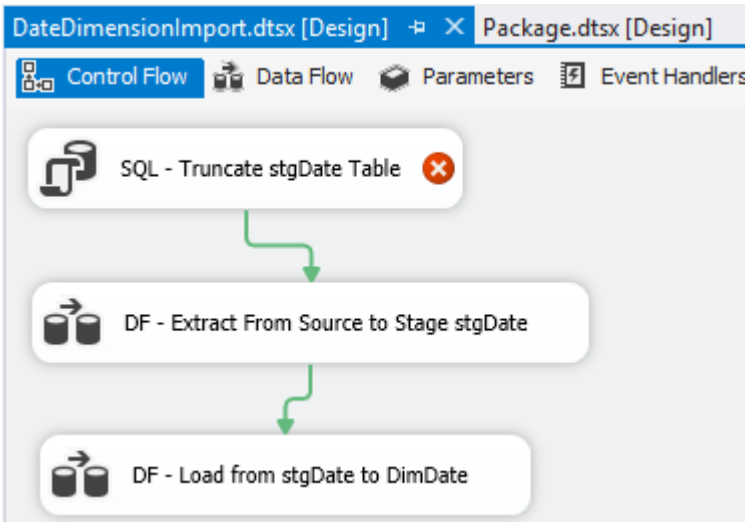
17. Click **OK** to complete the configuration of your destination. Your data flow design surface configuration should be complete.



a. Click back on the **Control Flow** tab. Press **CTRL +S** to save.

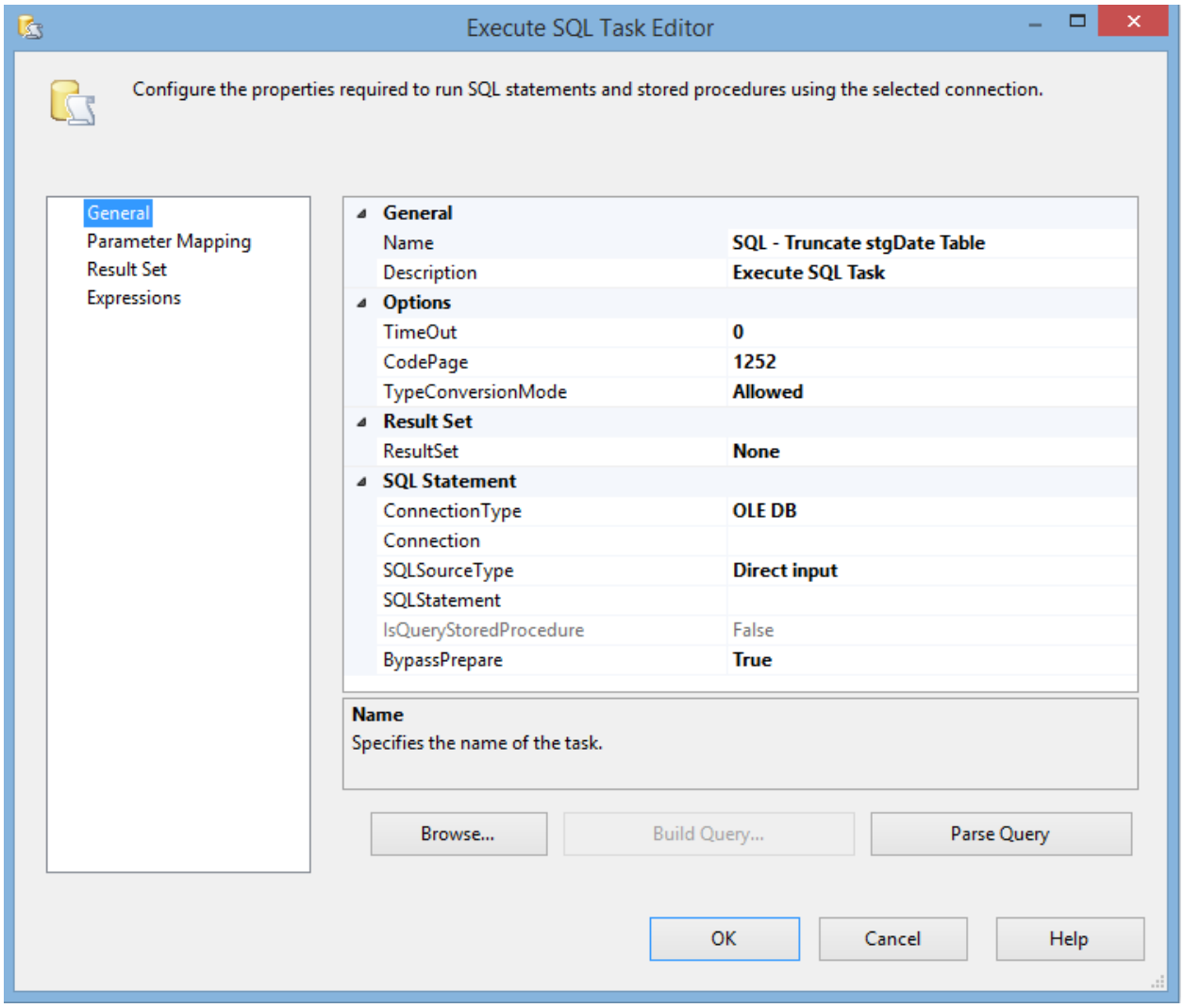
Step 2.1.4: Setup the truncate table SQL task

At this point we have completed ½ of the truncate and load pattern for stgDate.

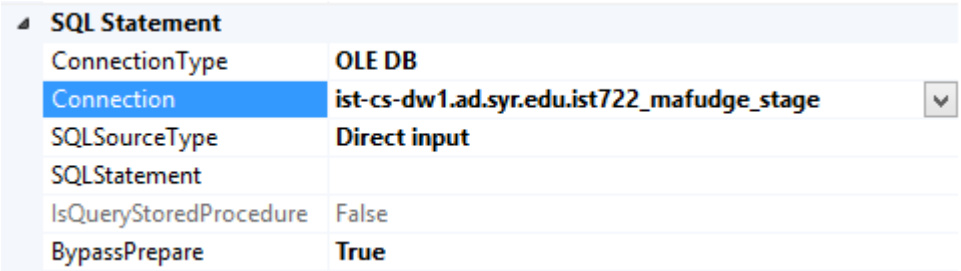


We still need to configure the SQL task to truncate the stgDate table before the extract.

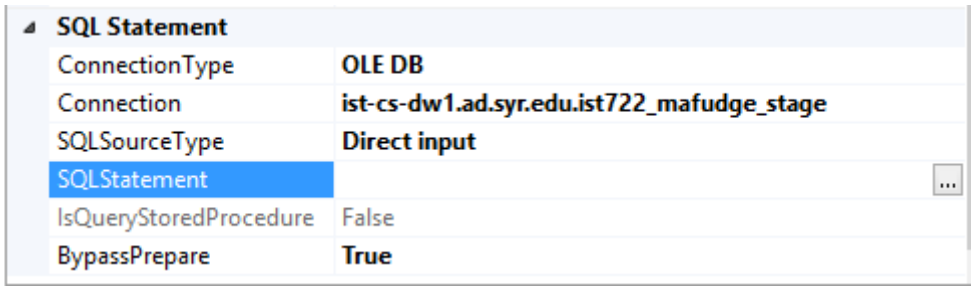
1. Double-click on the **SQL - Truncate stgDate Table** task to configure it. This will open the **Execute SQL Task Editor**



2. Under the **SQL Statement** heading you will find a **Connection** setting. When you click it a drop-down will appear. Select **ist722_*yournetid*_stage** from the drop down.

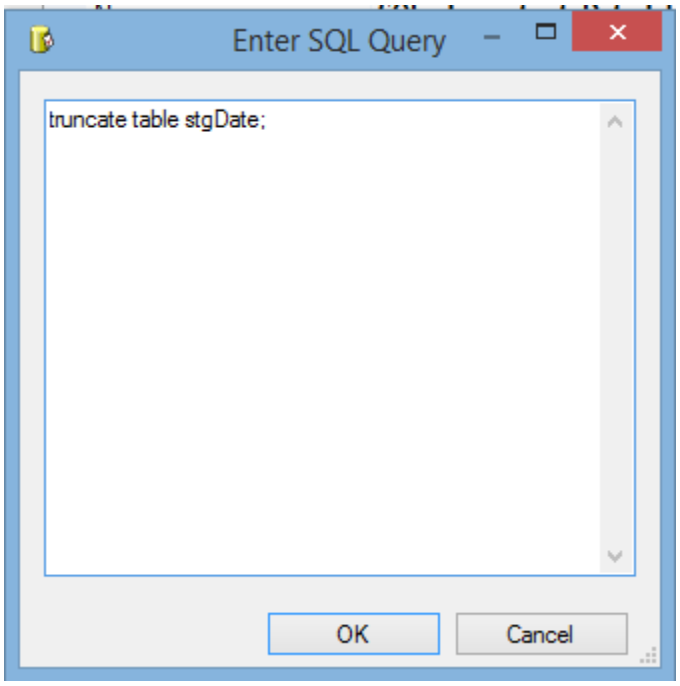


3. In the same section, under **SQL Statement** you will find a **SQLStatement** setting. When you click on it you will see a button with three dots **[…]** this is called a builder button.



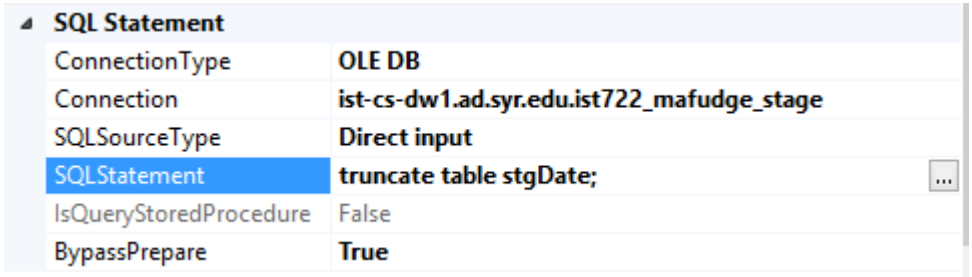
4. Click the builder button to open a dialog where you can enter an SQL Query. In the window, type this SQL command:

```
truncate table stgDdate;
```

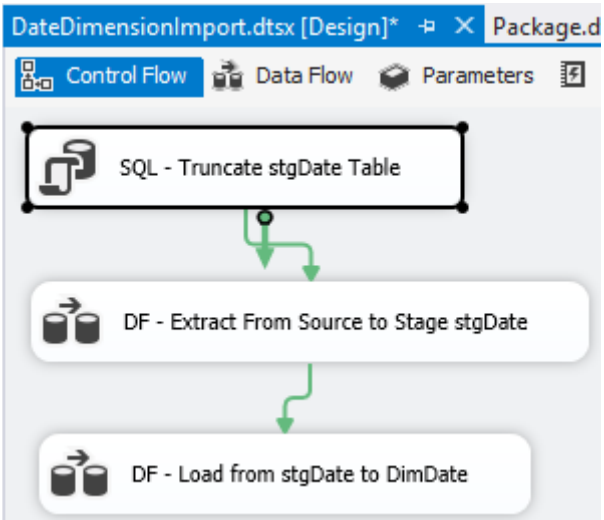


IMPORTANT: It should be noted that while you can type any SQL you want into the box, but the probability that you will type *valid* and *correct* SQL code is rare. You should always type your SQL code in SQL server management studio first, then copy / paste the code into this box.

5. Click **OK** to save your SQL Statement.



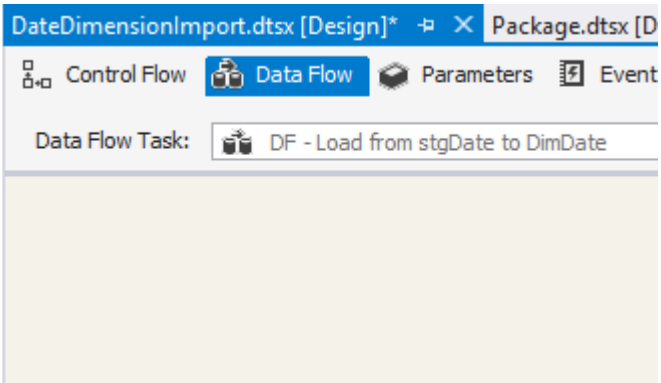
6. Click **OK** to close the **Execute SQL Task Editor**, then press **CTRL + S** to save. You should now have two steps in the control flow complete:



Step 2.1.5: Configure the Stage to Target data flow

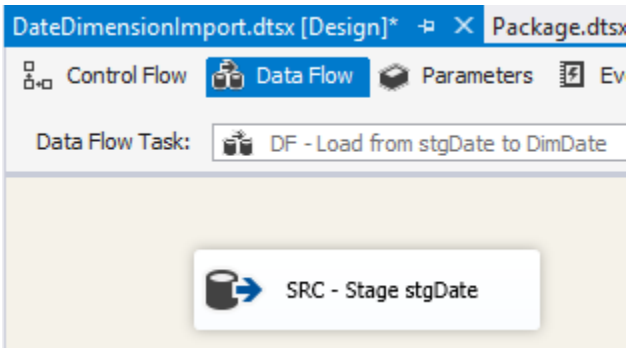
The final step is to configure the stage to target dimension data flow. In this data flow we will use a Type 1 Slowly Changing Dimension to ensure that we do not unintentionally introduce the same business key more than once to the dimension.

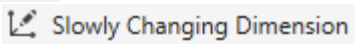
1. Start by double-clicking on the **DF - Load from stgDate to DimDate** task to bring up the data flow design surface.



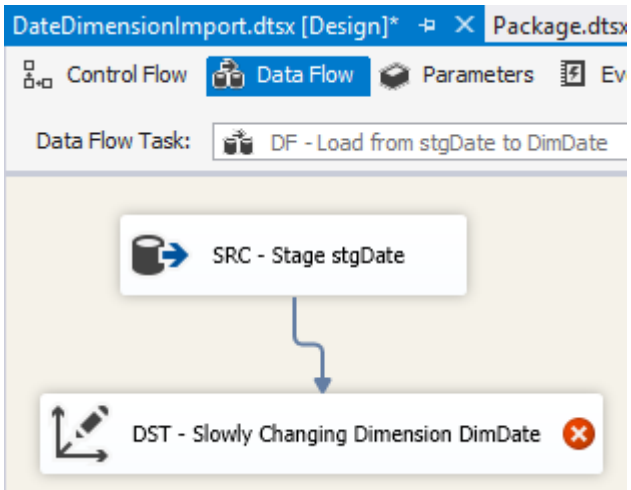
   Since you've done a data flow task once before you shouldn't need the same amount of hand holding.
2. Drag and drop the **Source Assistant** onto the design surface. Select **SQL Server** as the **source type** and **ist722_*yournetid*_stage** as the connection manager.
3. Rename the source to **SRC - Stage stgDate**, then double-click on it to configure.

4. From the **OLE DB Source Editor** select the **[dbo].[stgDate]** table for the **name of the table or view**. Click OK to finish the configuration, then **save your work**.



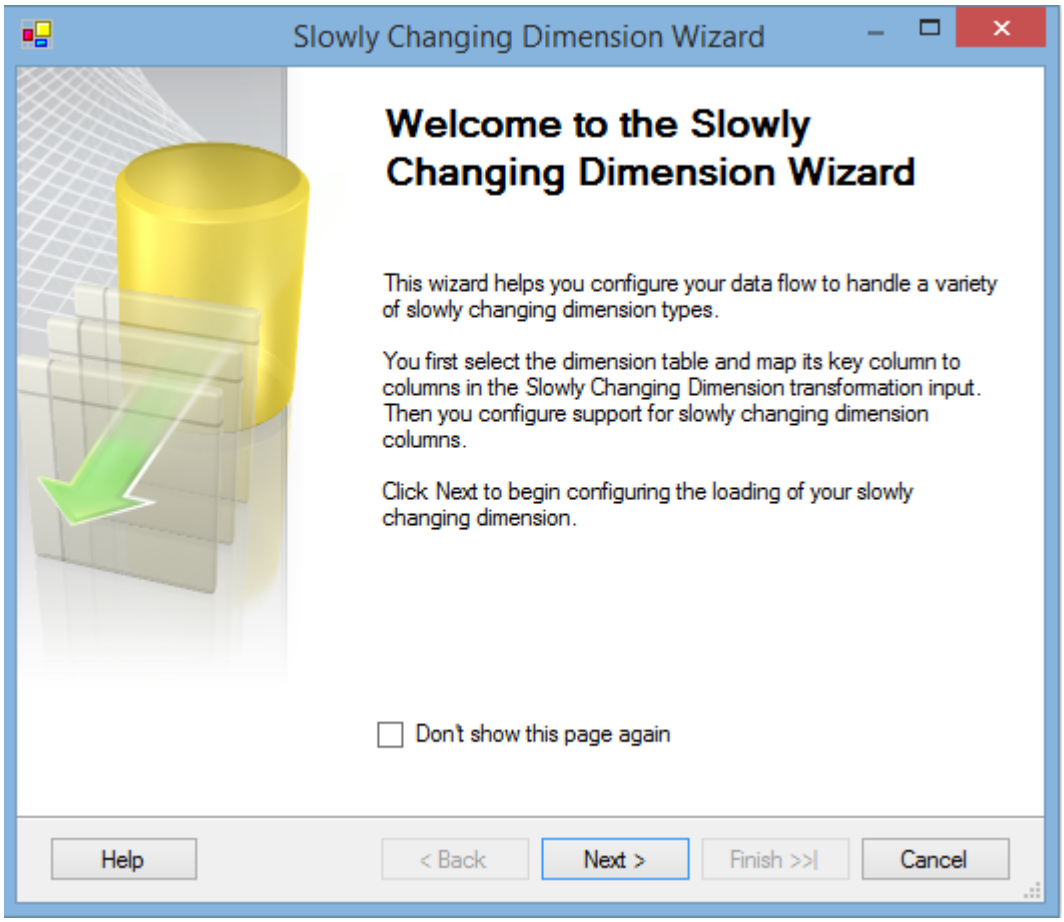
5. Drag and drop the **Slowly Changing Dimension** 

onto the design surface. Rename the destination **DST - Slowly Changing Dimension DimDate** and connect the blue arrow from SRC to DST.



6. The final step in this process is to configure the destination and type 1 workflow. Double click on the **DST - Slowly Changing Dimension DimDate** to begin.

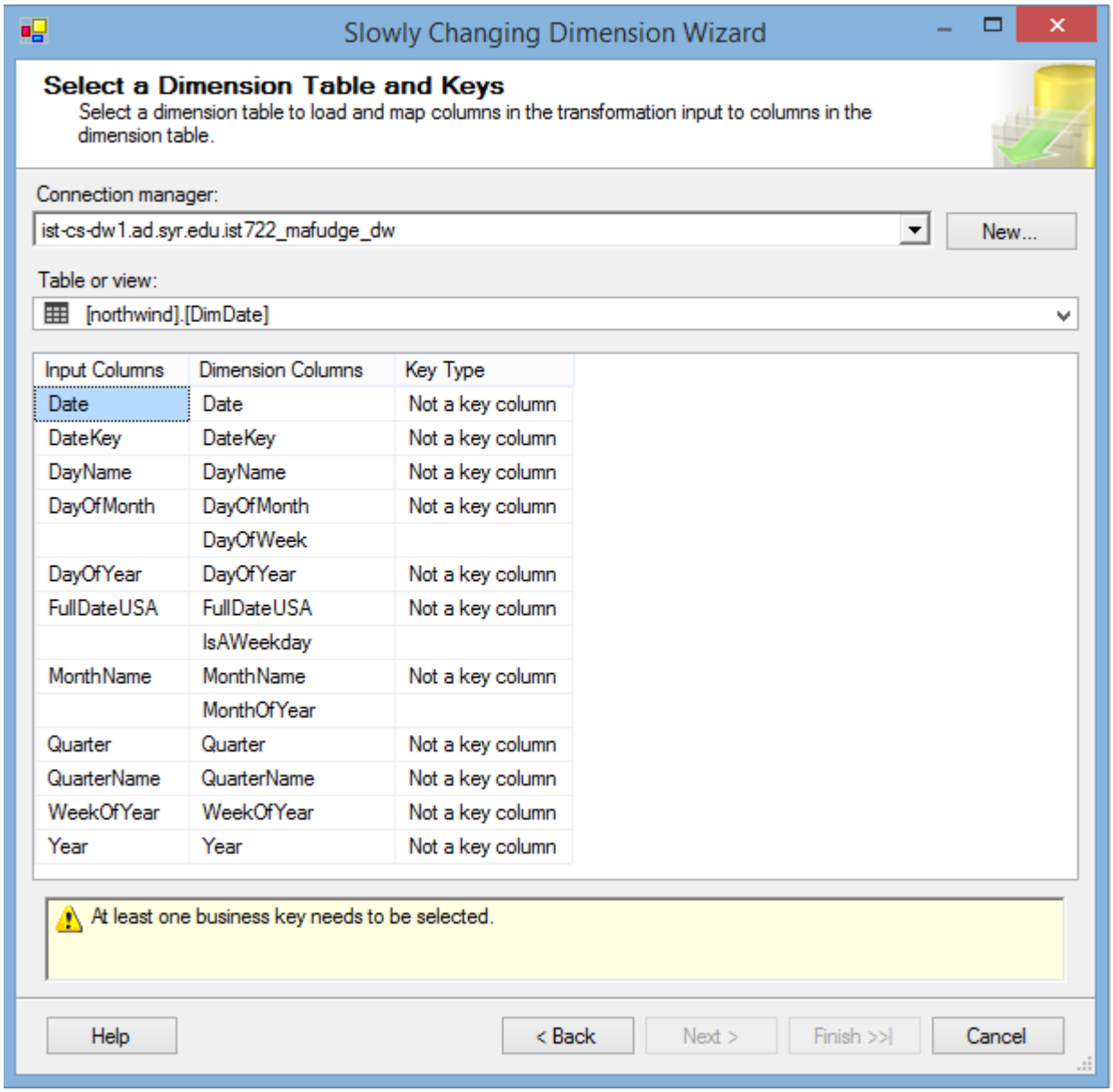
7. First you'll see the welcome screen for the **Slowly Changing Dimension Wizard**.



Read the information and then click **Next >**

8. The first step in this process is to select the Destination for the dimension. This should be the **northwind.DimDate** table in your **ist722_*yournetid*_dw** database. Under **connection manager** select your **ist722_*yournetid*_dw** database.

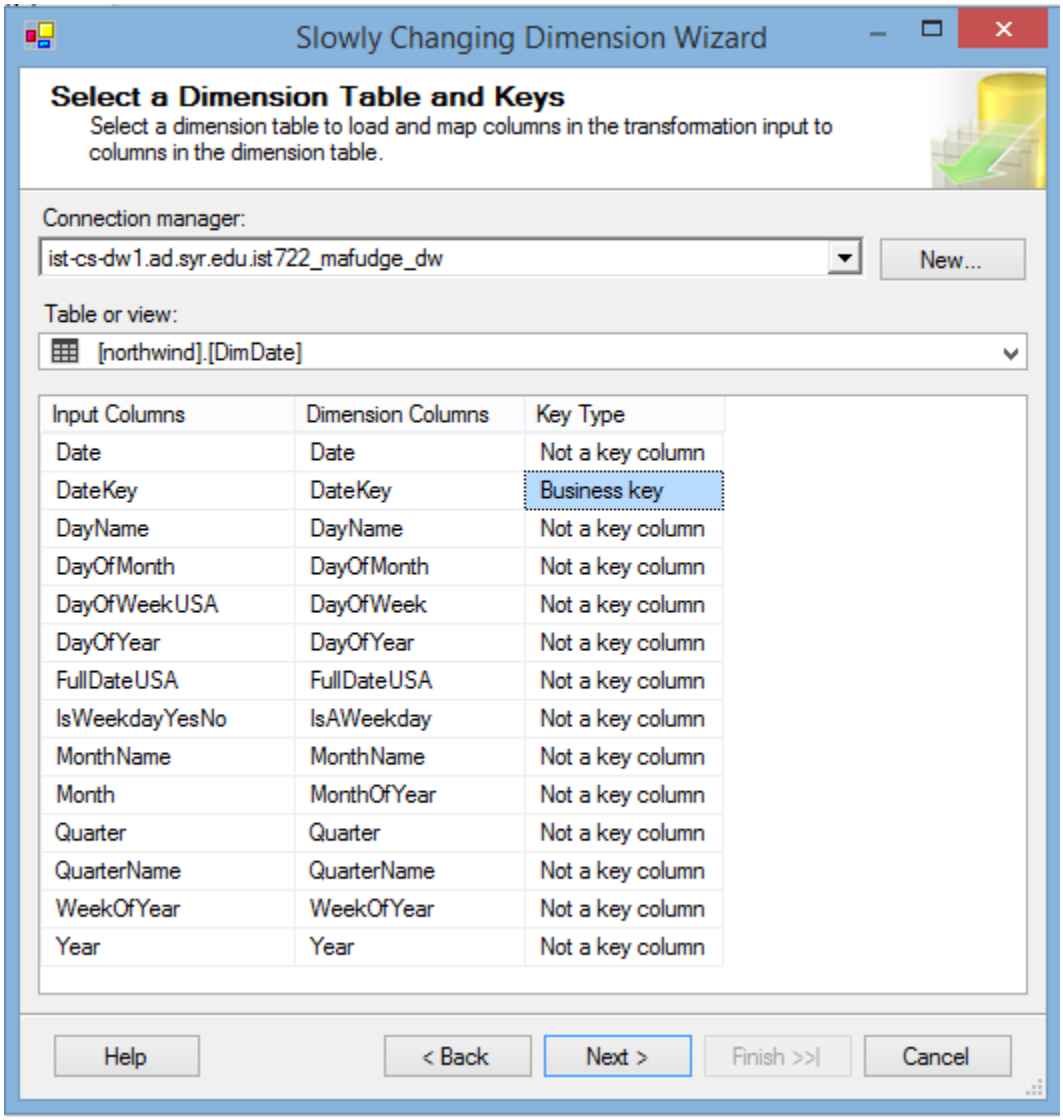
9. Under **Table or view** select the **[northwind].[DimDate]** table.



Now you need to configure the source to target mapping of the columns, and select the column(s) which act as business key. Every dimension column must have an input column. The SCD processing uses the business key to track changes in the following manner:
- Business key not in Dimension? Add.
- Business key in Dimension but one of the non-keys is different? Update (Type 1 or Type2).
- Business key in Dimension but all non-keys are the same? Ignore. Data exists already.

10. Map the remaining dimension columns, from **Input Columns** to **Dimension Columns**:
    a. DayOfWeekUSA → DayOfWeek
    b. IsWeekdayYesNo → IsAWeekday
    c. Month → MonthOfYear

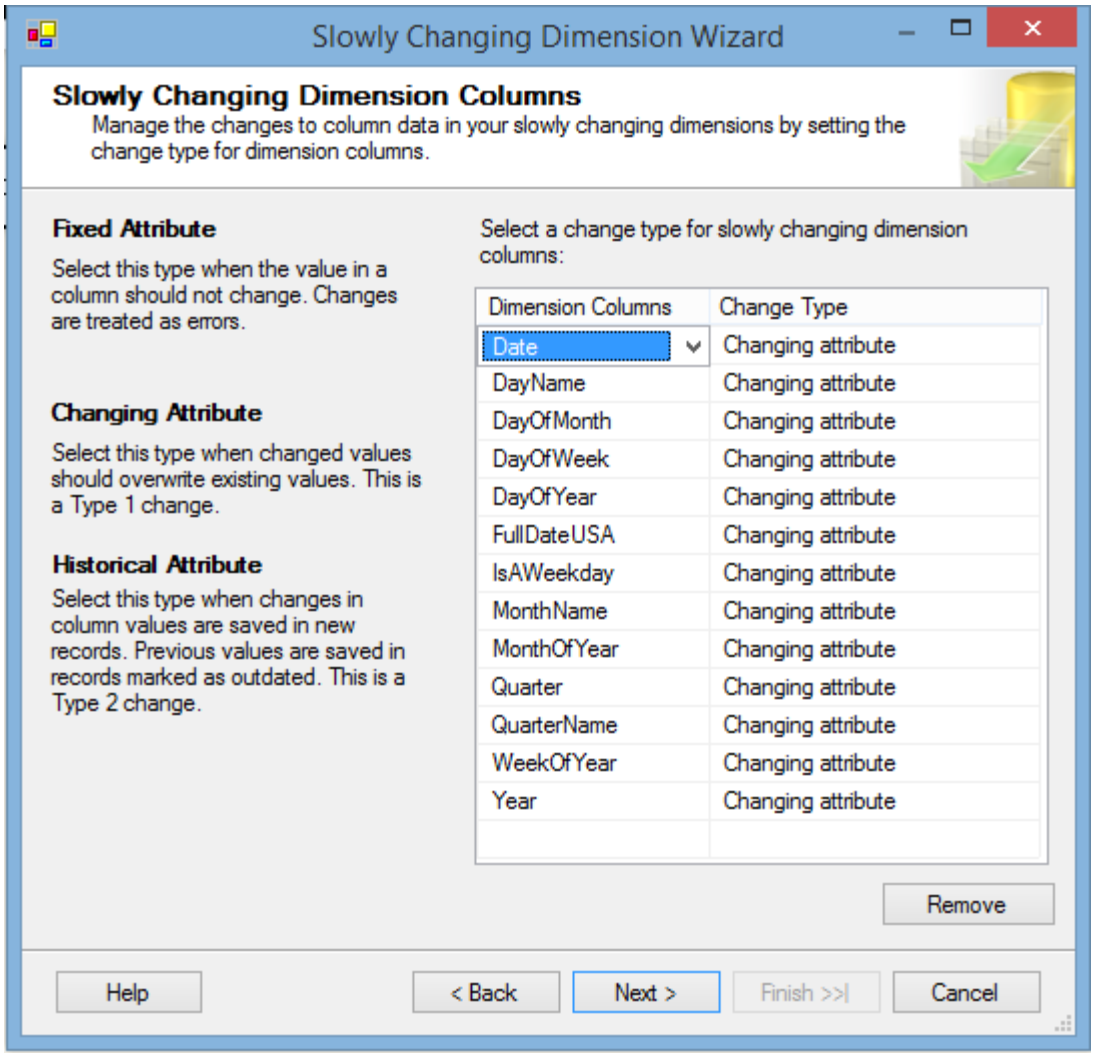
11. Select **DateKey** as the business key. All other columns should say "Not a key column" **NOTE:** This is not the typical case – it's only this way because it's a date dimension.



Click **Next >** when you're ready.

12. In the next step you'll configure the SCD type for the dimension. Your choices are:
    - Fixed → No changes tracked
    - Changing → Type 1 (Update)
    - Historical → Type 2 (Add new row, Make old row obsolete.)

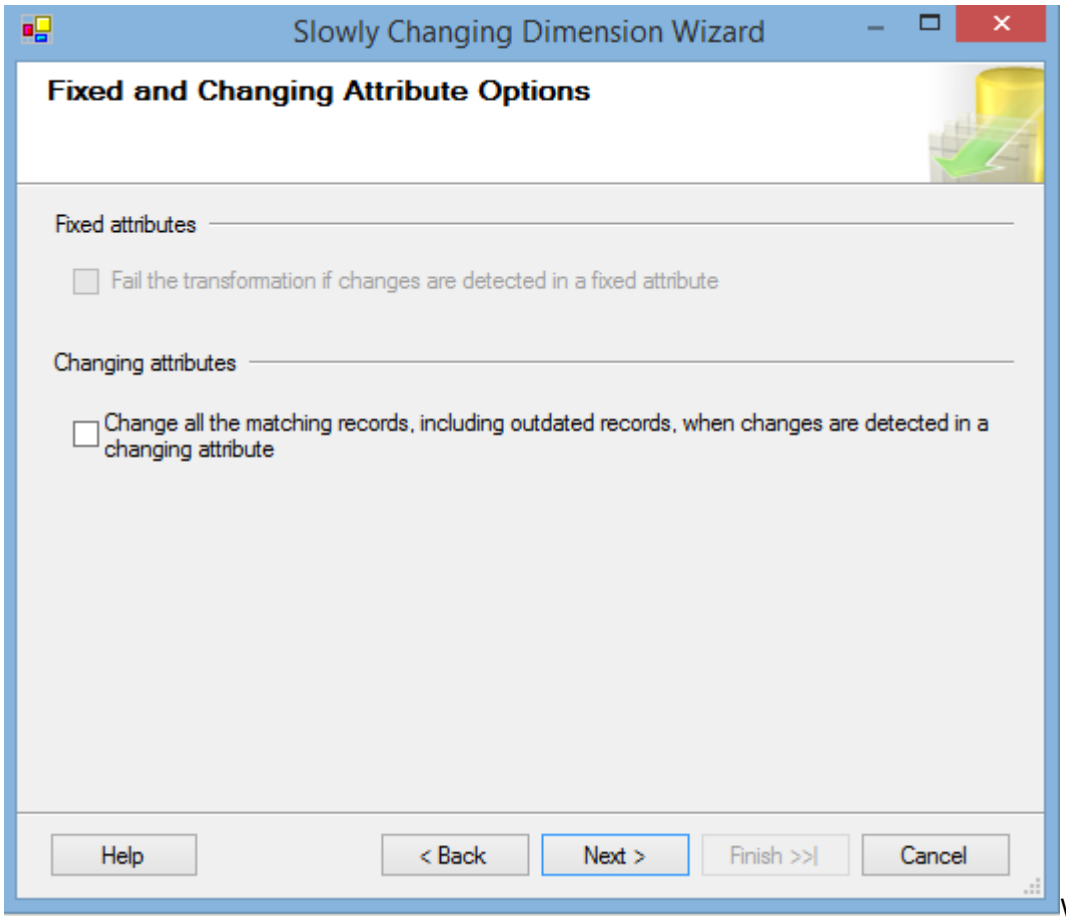
    As a rule of thumb you should apply the same SCD type to the entire dimension, although this is not a requirement of the tooling.



Configure all of the non-business key columns to use a changing attribute.

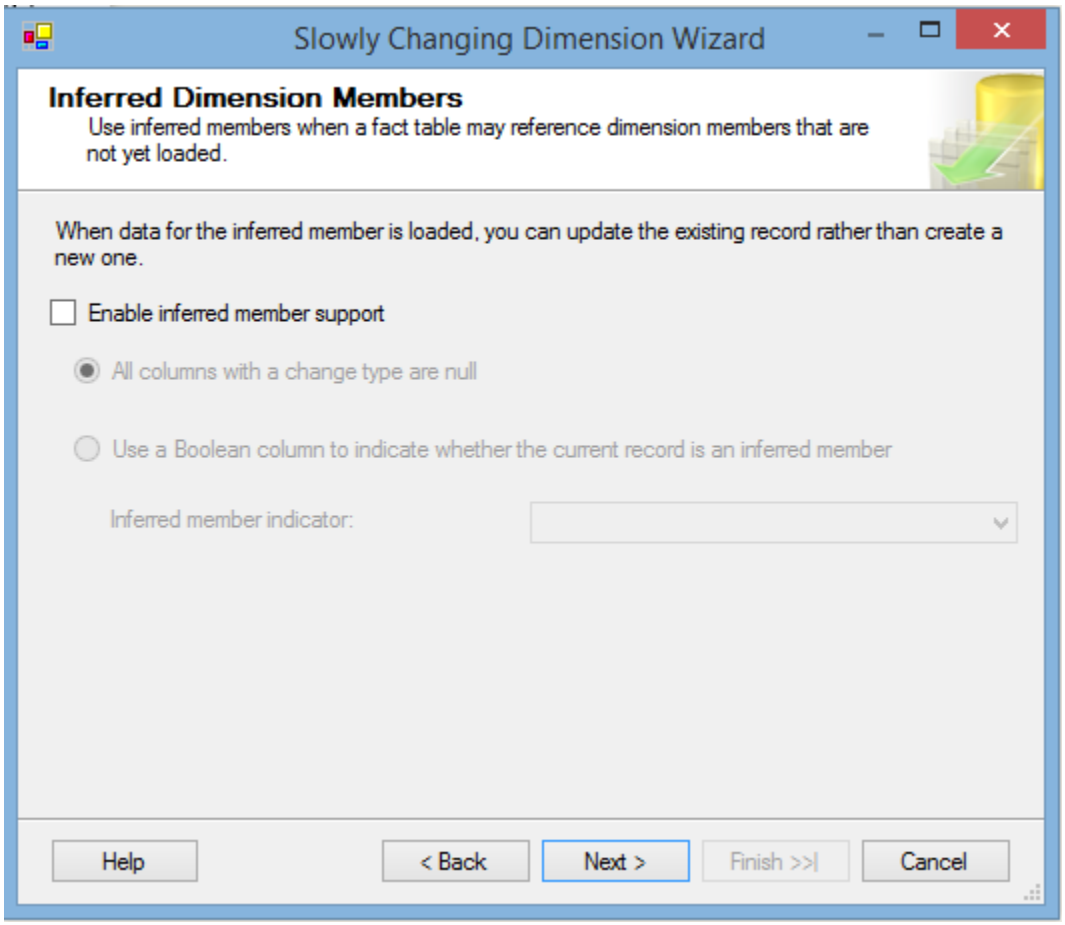
The user interface for this is a little wonky, using "magic dropdowns" which only appear when you click in the grid. Just remember the **[tab]** key is your friend here.
When you've completed this step, click **Next >**

13. Next up is the **Fixed and Changing Attrribute Options** screen. For the Date Dimension, we want **both of these check-boxes cleared**, since an update here will only change 1 row.



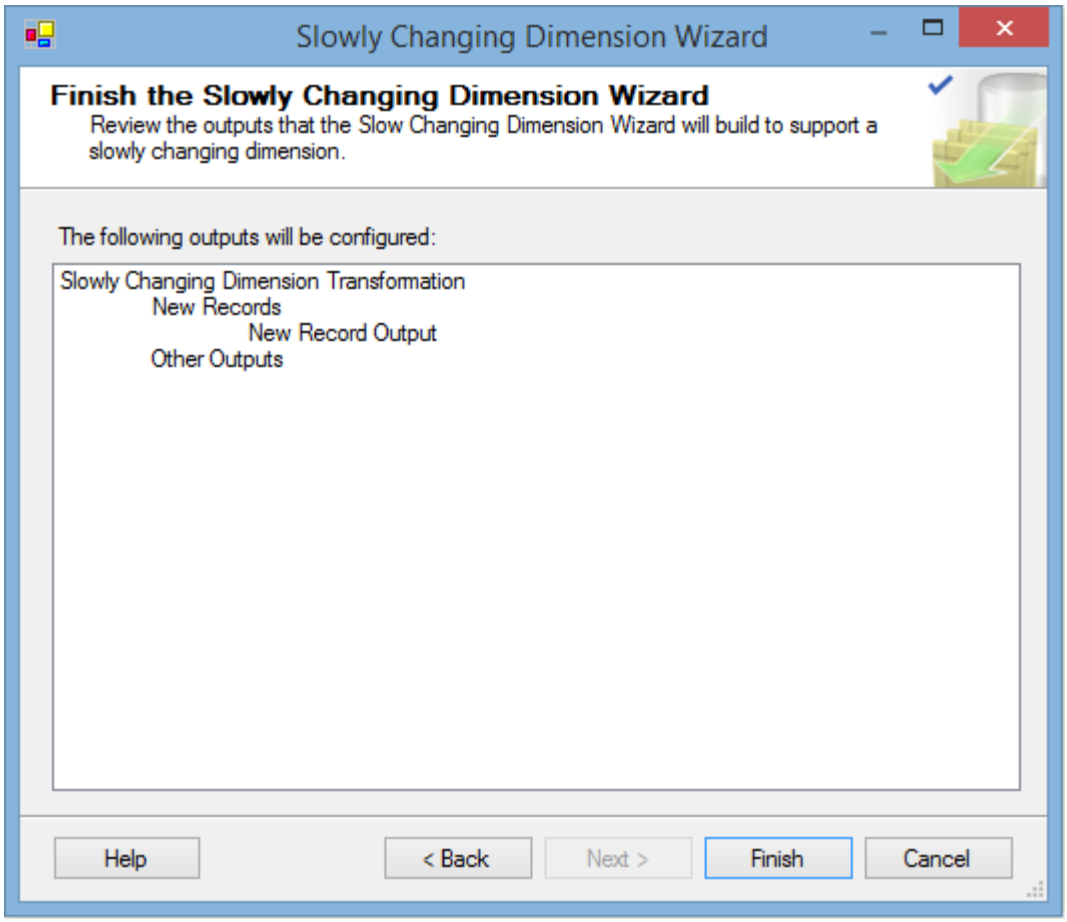
When you're ready, click **Next >**

14. Now you will see the **Inferred Dimension Members** screen. This is used for late-arriving facts where you do not have all of the dimension values. Since this case does not apply here, **clear the check box**.

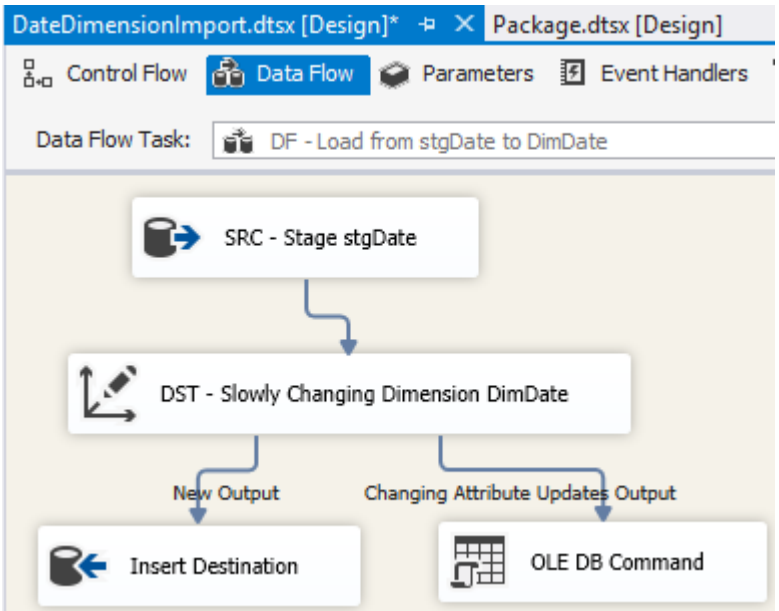


When you're ready, click **Next >**

15. Finally, the finish Screen. This screen will inform you of the script SSIS will generate to complete the processing.



Click **Finish** to generate the script. You will now have a completed data flow.



a. Save your work. Switch back to the control flow tab.

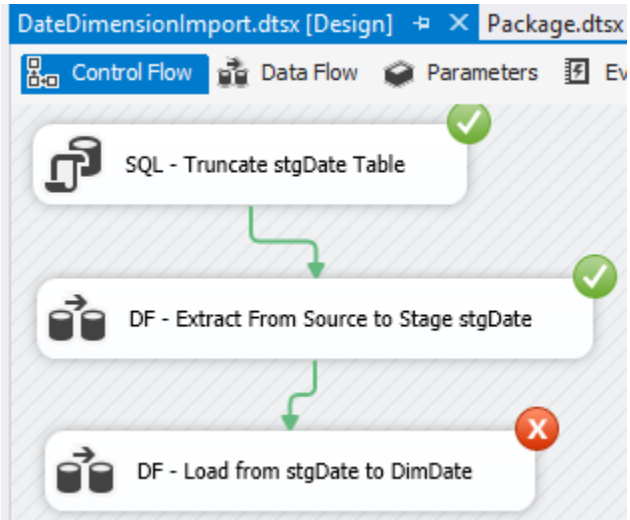
Step 2.1.6: Executing and Troubleshooting Your SSIS Package

Now that your package is complete, and there are no syntax / configuration errors, it's time to execute it. There could, however, still be some runtime errors. Runtime errors occur most often when adding or updating data into the destination tables. Under these conditions, one of the following occurs:

- A constraint (PK, FK, unique or check) fails,
- The data types from source to target do not match, or
- The source data has null, but the target does not allow it.

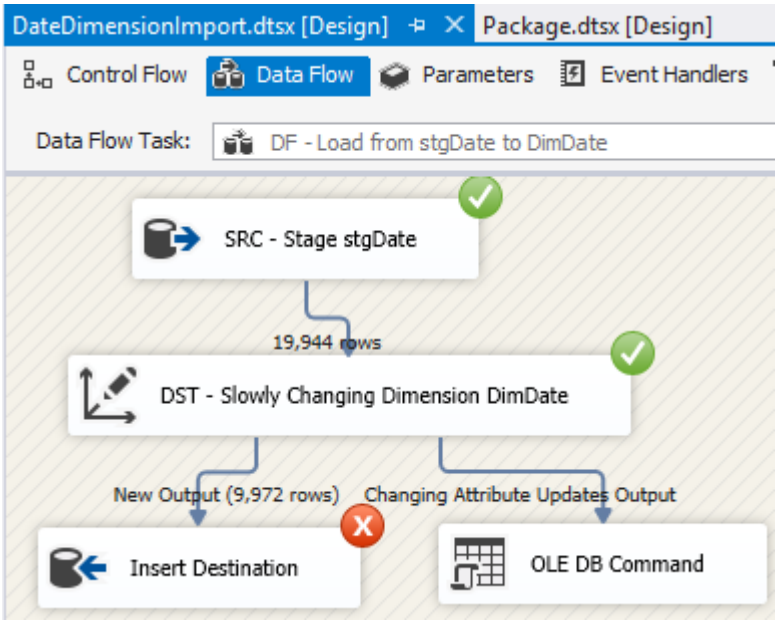
As we'll see in a minute, the SSIS errors are rather cryptic. Troubleshooting SSIS errors is a lot of common sense. You need to think about the operation you're doing and brainstorm the potential issues which may occur.

1. First let's try to execute the package. Press **[F5]** to execute. The background will now contain lines to indicate the package is in execution mode, and after a few seconds your package will execute with an error.



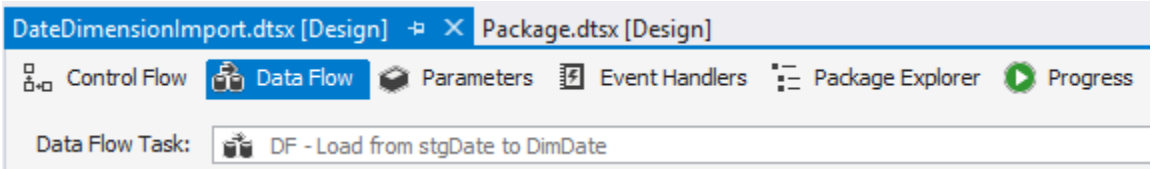
   NOTE: If you haven't figured it out yet, this was by design. I intentionally added an error so you can learn how to troubleshoot.

2. Let's try to figure out what went wrong. We know the **DF – Load from StgDate to DimDate** failed, but why? Double click on the task to bring up the data flow:

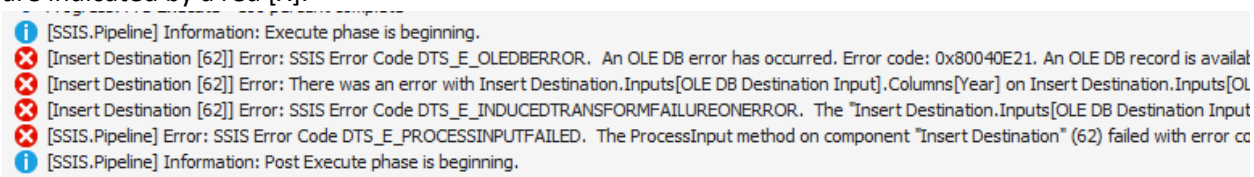


In the data flow you can see that the task failed when trying to insert data into the destination table **northiwind.DimDate**.

3. Let's check the error message. At the top of the **design surface** you will see a **progress tab**.

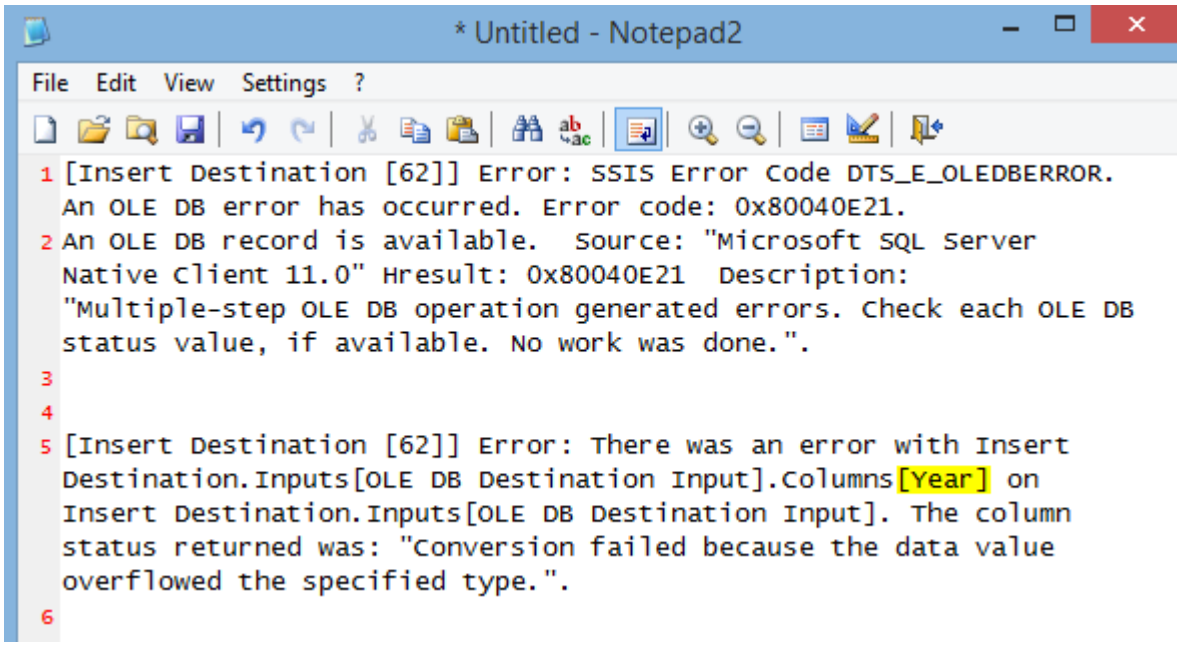


Click on the **progress tab,** then scroll down through the progress until you see the first error. Errors are indicated by a red [X].



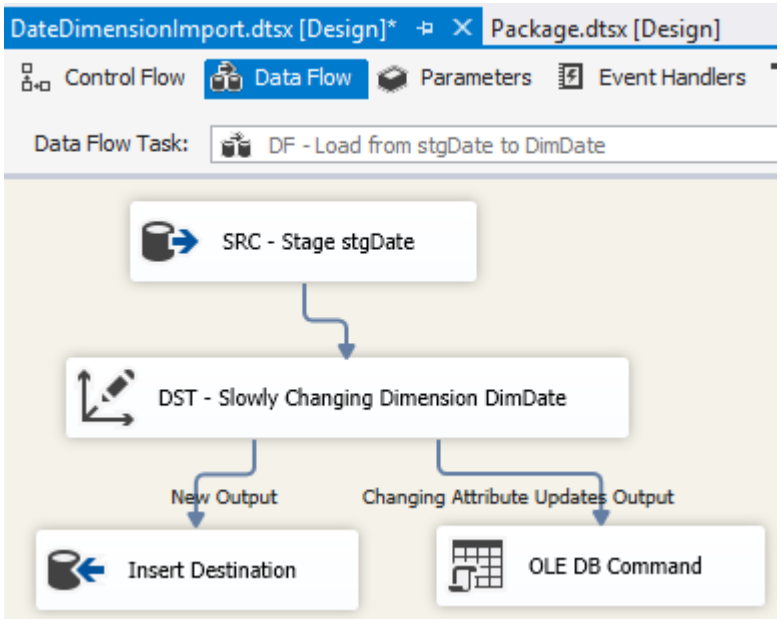
4. Here's where things get really ugly. The error is so long, you cannot view it in the progress window! Ugh. Your best bet is to **right-click on each error** then select **Copy Message Text** from the menu to copy the error message, and then paste it into a text editor like Word or Notepad. You should do

this for each error until you find one that gives an actual hint as to what the problem might be.



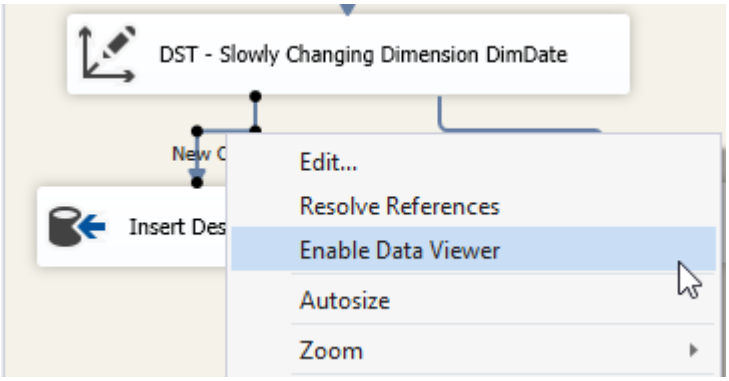
In the screenshot above I've highlighted the column giving us trouble – the **[Year]** column in the destination **northwind.DimDate**. So for some reason it cannot insert the values from the source. Furthermore the message specifies an "overflow" which typically means the source data type is too large for the target data type.

5. Before we dive right in and fix the **[Year]** column in the **northwind.DimDate** table, let's first learn about a really neat troubleshooting feature in SSIS, called the **data viewer**. As the name implies, the data viewer allows you to observe what the data flow looks like at any given step. This is a great way to see what SSIS transformations are actually doing to your data. First stop package execution by pressing **[Shift] + [F5]**, and then switch back to the **Data Flow** tab.

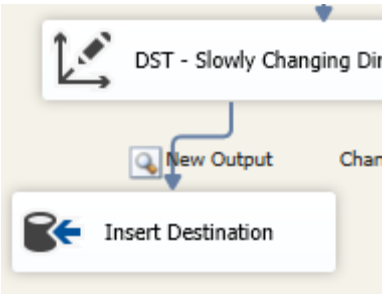


NOTE: You'll notice there's no more diagonal lines in the background because the package is no longer in execution mode.

6. To enable data viewer we must first select the input / output data flow (blue arrow) for which we would like to view data. You can select more than one arrow, but in our case we want the arrow going into the **Insert Destination** task. Click on this arrow to select it, then right-click and select **Enable Data Viewer** from the menu.



You know you've done it correctly when you see a **magnifying glass icon** next to the arrow.



7. Next, let's re-run our package, by pressing **[F5]** again. When the data flow progress reaches the data viewer, a window with the data will pop up!

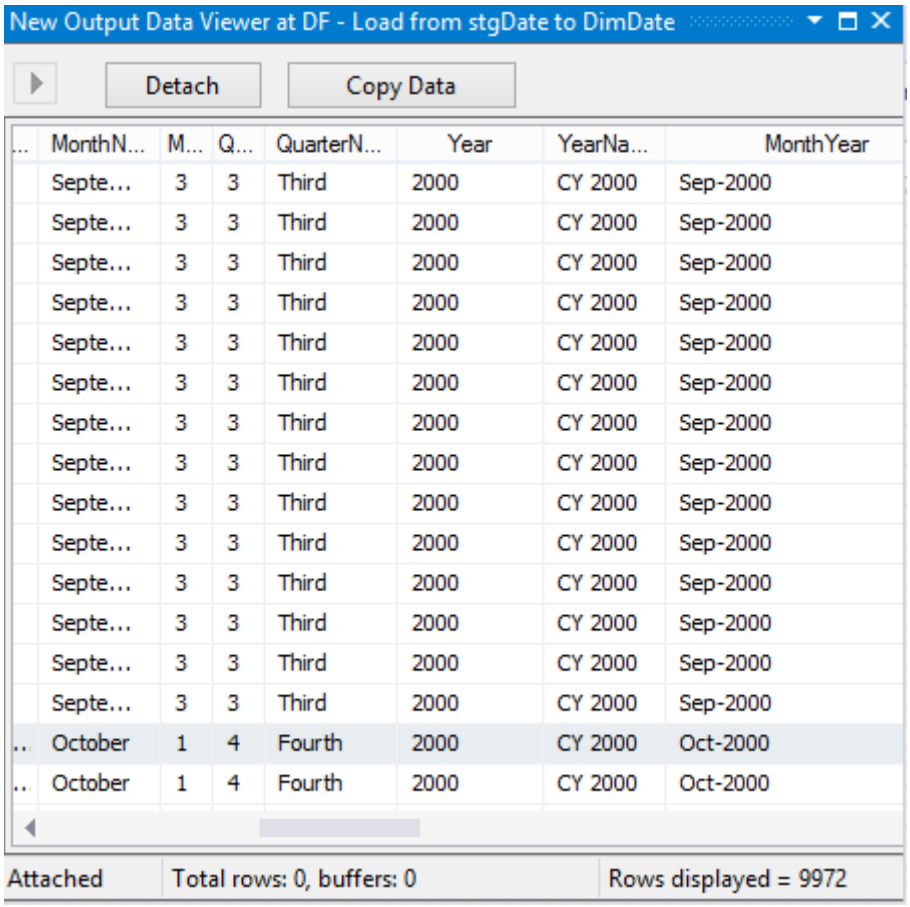
New Output Data Viewer at DF - Load from stgDate to DimDate

Detach | Copy Data

Dat...	Date	FullDate...	FullDate...	D...	DaySuf...	DayName	D...	D...	D...
20...	2000-0...	17/09/...	09/17/2...	1...	17th	Sunday	1	7	3
20...	2000-0...	18/09/...	09/18/2...	1...	18th	Monday	2	1	3
20...	2000-0...	19/09/...	09/19/2...	1...	19th	Tuesday	3	2	3
20...	2000-0...	20/09/...	09/20/2...	2...	20th	Wedne...	4	3	3
20...	2000-0...	21/09/...	09/21/2...	2...	21st	Thursday	5	4	3
20...	2000-0...	22/09/...	09/22/2...	2...	22nd	Friday	6	5	4
20...	2000-0...	23/09/...	09/23/2...	2...	23rd	Saturday	7	6	4
20...	2000-0...	24/09/...	09/24/2...	2...	24th	Sunday	1	7	4
20...	2000-0...	25/09/...	09/25/2...	2...	25th	Monday	2	1	4
20...	2000-0...	26/09/...	09/26/2...	2...	26th	Tuesday	3	2	4
20...	2000-0...	27/09/...	09/27/2...	2...	27th	Wedne...	4	3	4
20...	2000-0...	28/09/...	09/28/2...	2...	28th	Thursday	5	4	4
20...	2000-0...	29/09/...	09/29/2...	2...	29th	Friday	6	5	5
20...	2000-0...	30/09/...	09/30/2...	3...	30th	Saturday	7	6	5
20...	2000-1...	01/10/...	10/01/2...	1	1st	Sunday	1	7	1
20...	2000-1...	02/10/...	10/02/2...	2	2nd	Monday	2	1	1

Attached | Total rows: 0, buffers: 0 | Rows displayed = 9972

If you scroll over to the **Year** column, you will see 4 digit years.



...	MonthN...	M...	Q...	QuarterN...	Year	YearNa...	MonthYear
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
	Septe...	3	3	Third	2000	CY 2000	Sep-2000
..	October	1	4	Fourth	2000	CY 2000	Oct-2000
..	October	1	4	Fourth	2000	CY 2000	Oct-2000

```
Attached    Total rows: 0, buffers: 0    Rows displayed = 9972
```

Very helpful. At this time, please turn off the data viewer by right clicking on the **magnifying glass icon** and selecting **Disable Data Viewer**.

8. Let's get to the source of the problem. Check the SQL code in **Northwind-ROLAP-Bus-Architecture.sql** we used to create the schema. Odds are pretty good that the data type I chose for the **[Year]** column in the table **northwind.DimDate** is too small to support a 4 digit year. Switch back to **SQL Server Management studio** and open the file if it is not open already. Scroll down to **line 152**

and you'll see the problem.

```
CREATE TABLE [northwind].[DimDate](
    [DateKey] [int] NOT NULL,
    [Date] [datetime] NULL,
    [FullDateUSA] [nchar](11) NOT NULL,
    [DayOfWeek] [tinyint] NOT NULL,
    [DayName] [nchar](10) NOT NULL,
    [DayOfMonth] [tinyint] NOT NULL,
    [DayOfYear] [int] NOT NULL,
    [WeekOfYear] [tinyint] NOT NULL,
    [MonthName] [nchar](10) NOT NULL,
    [MonthOfYear] [tinyint] NOT NULL,
    [Quarter] [tinyint] NOT NULL,
    [QuarterName] [nchar](10) NOT NULL,
    [Year] [tinyint] NOT NULL,
    [IsAWeekday] varchar(1) NOT NULL DEFAULT (('N')),
    constraint pkNorthwindDimDate PRIMARY KEY ([DateKey])
)
```

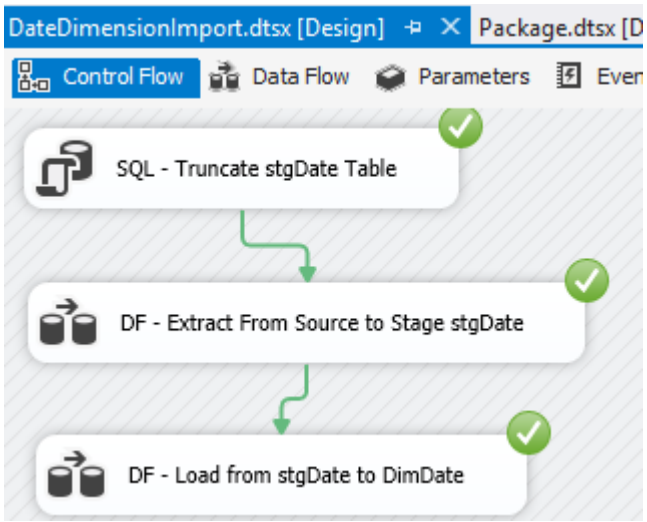
The **tinyint** data type is too small (its only 1 byte in size), and can only represent numbers from -128 to +127. 4 digit years are outside this range.

9. Edit the code, changing the data type from **tinyint** to **smallint**. Then press **[F5]** to execute the SQL script and re-create the tables. Congratulations, you've "fixed" my mistake!

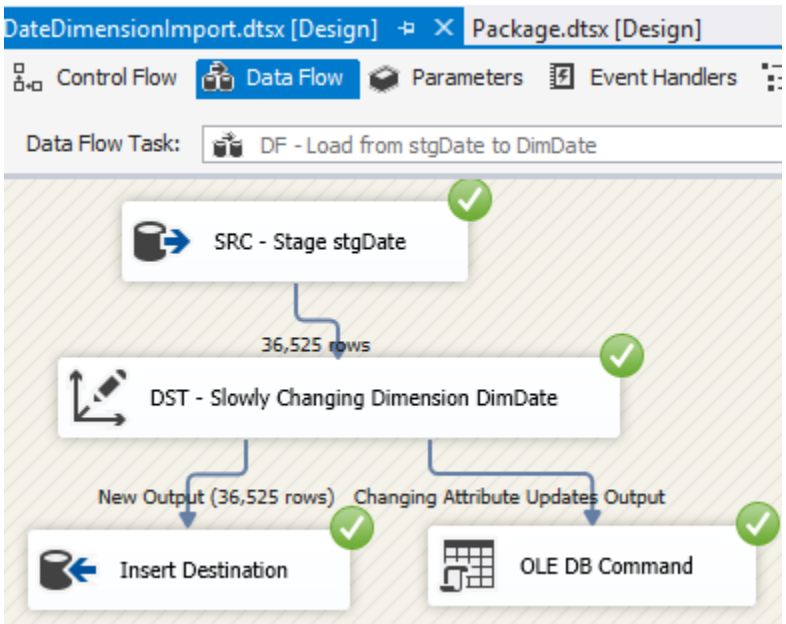
```
CREATE TABLE [northwind].[DimDate](
    [DateKey] [int] NOT NULL,
    [Date] [datetime] NULL,
    [FullDateUSA] [nchar](11) NOT NULL,
    [DayOfWeek] [tinyint] NOT NULL,
    [DayName] [nchar](10) NOT NULL,
    [DayOfMonth] [tinyint] NOT NULL,
    [DayOfYear] [int] NOT NULL,
    [WeekOfYear] [tinyint] NOT NULL,
    [MonthName] [nchar](10) NOT NULL,
    [MonthOfYear] [tinyint] NOT NULL,
    [Quarter] [tinyint] NOT NULL,
    [QuarterName] [nchar](10) NOT NULL,
    [Year] [smallint] NOT NULL,
    [IsAWeekday] varchar(1) NOT NULL DEFAULT (('N')),
    constraint pkNorthwindDimDate PRIMARY KEY ([DateKey])
)
```

NOTE: A **smallint** is 2 bytes in size and represents numbers in the range -32768 to + 32767. You might be asking yourself, why not just use an **int**? Well, an int is 4 bytes, 2 more than we need, which means my table would be 73KB (2 * 36,525 rows) larger than it has to be! While this doesn't seem like a huge waste, believe it or not every byte counts in the data **warehouse especially in fact tables** where there are billions of rows. If there were 1 billon rows, we'd be "wasting" 2GB of space!

10. Let's go back to SSIS and press **[F5]** to execute the package. When the package is complete you will see green **checkmarks next** to each task.



11. If you observe the **DF - Load From stgDate To DimDate** task.
You will see the data flow is now working and includes rows affected.



12. Press **[Shift] + [F5]** to stop package execution. This returns you to design mode. Press **CTRL + S** to save your work.

Congratulations! You just created and executed and debugged your first SSIS package. You now have a good foundation to build upon. In the next step we'll stage the dimensions and facts for Inventory Levels.

Step 2.1.7: Did it work?

At this point you might still have trouble "seeing the forest through the trees." What did we just do? I know it "works" but how do I know it really did what was expected?

The best way to address this is to simply inspect the target table. Is there data in the table? 36,525 rows, to be exact?

You can verify this with some simple SQL queries.

1. Open **SQL Server Management studio.**
2. Switch to your **ist722_*yournetid*_dw** database
3. Press **CTRL + N** to open a new query window.
4. Execute the following query:
   ```
   SELECT * FROM [northwind].[DimDate]
   ```
5. Does it return output? How many rows?

It's important to verify that you SSIS package are indeed accomplishing the work we originally intended them to do. The best way to verify that is to inspect data in the target tables!

Part 2.2: Staging the Inventory Levels Dimensional Model

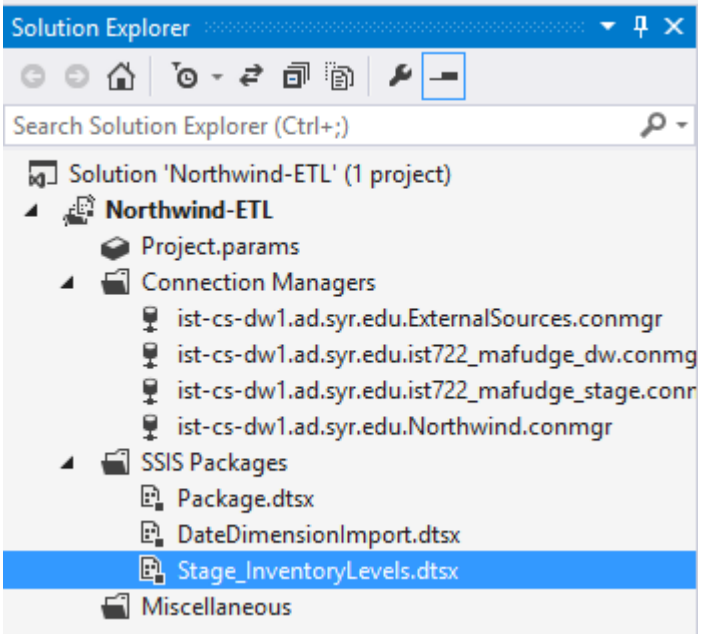
In this next part, we will create a package which performs ½ of the ETL process, collecting data from our sources and staging the data "as-is." When I say stage "as-is" I mean we will apply no transformations to the data. We will simply copy data from source into our staging database. Again we will use the truncate and load pattern so that each time the package executes the stage tables are emptied.

Step 2.2.1: Create the Package

First we need to create the **Stage_InventoryLevels.dtsx** package to contain our SSIS logic.

1. In the **Solution Explorer** window, right-click on **SSIS Packages** and select **New SSIS Package** from the menu.

2. Rename the package **Stage_InventoryLevels.dtsx**

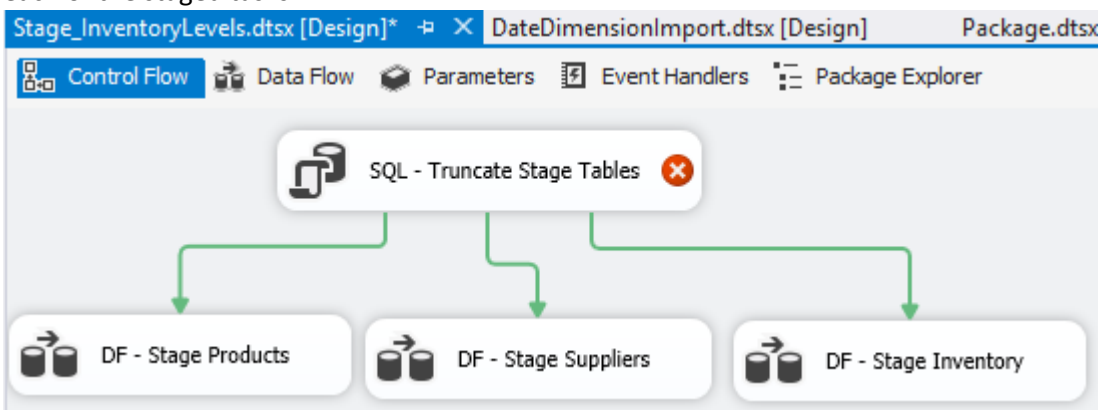


3. Double-click on the **Stage_InventoryLevels.dtsx** package to open it in the **design surface**.

Step 2.2.2: Setup the Control Flow

Next we setup the control flow. Again we will use the truncate and load pattern for all three sources.

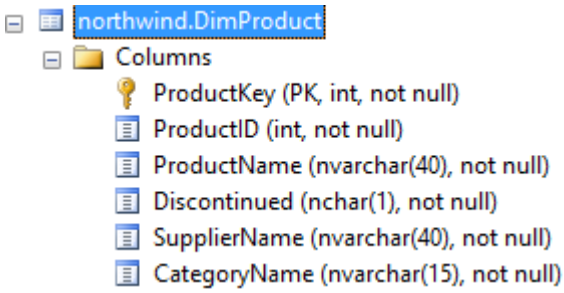
1. Add **one Execute SQL Task** and **Three Data Flow Tasks** to your design surface.
2. Rename the **Execute SQL Task → SQL – Truncate Stage Tables**
3. Rename the 3 Data flow Tasks:
   a. **DF – Stage Products**
   b. **DF – Stage Suppliers**
   c. **DF – Stage Inventory**
4. Connect the SQL task to each of the data flow tasks, so that the SQL task executes first, then each of the staged tasks.



Step 2.2.3: Staging Products

First we will work on staging Products. Unlike the previous example where we build the entire package and then executed it, this time we will build, execute, and verify before staging the next source.

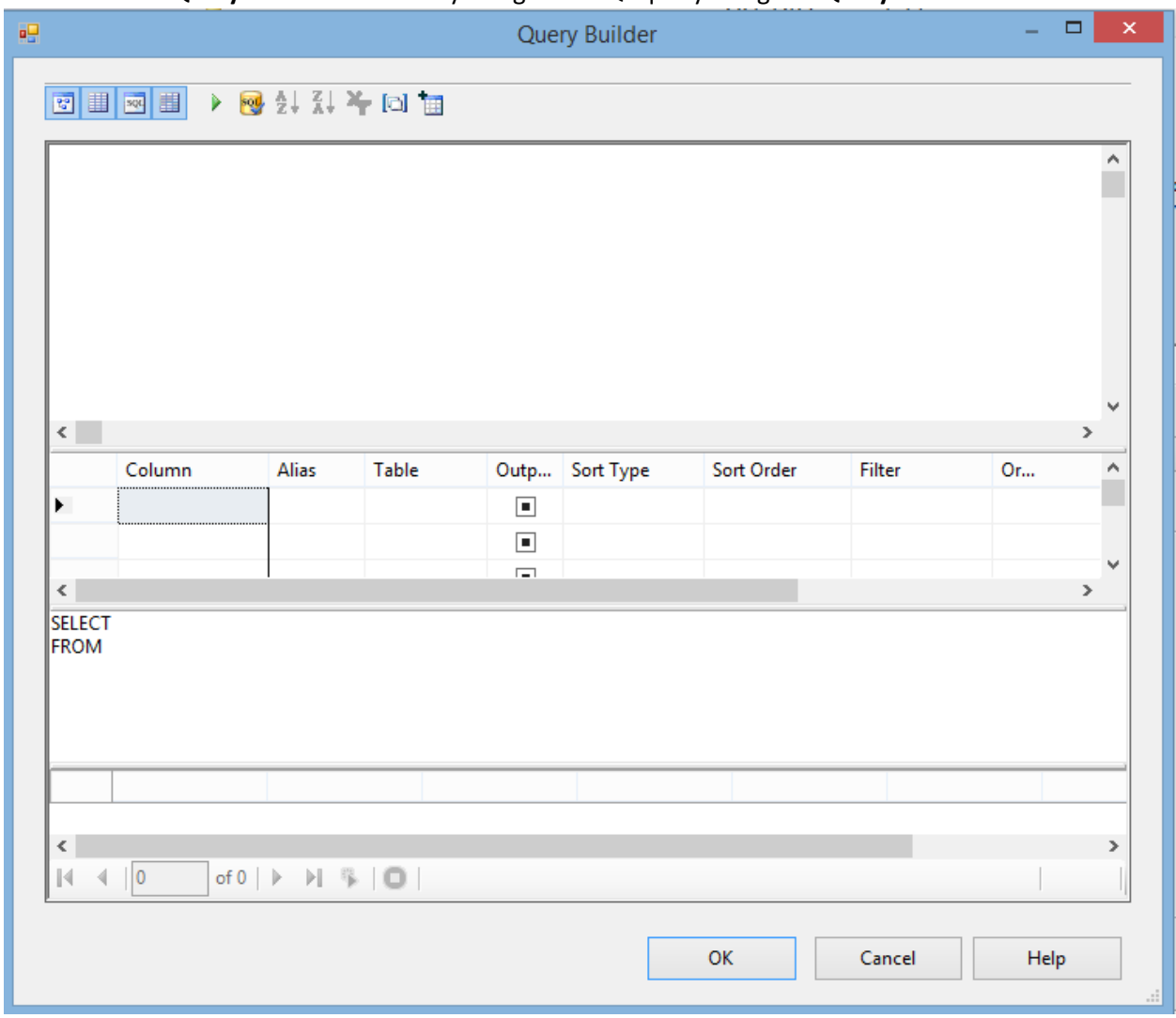
1. Double click on **DF - Stage Products** to open the data flow.
2. Use the **Source Assistant** to create the data source:
    a. Type: **SQL Server**
    b. Connection Manager → **Northwind**
    c. Rename to → **SRC - Northwind Products**
3. Double-click on the data source to configure it. How do we know what we will need from the source? We look at our detailed dimensional modeling documentation! Since that was not provided for this lab in this case we look at the target Dimension:



We need ProductID, ProductName, Discontinued, SupplierName and CategoryName. These do not come from the same table, so we'll need to write our own query:

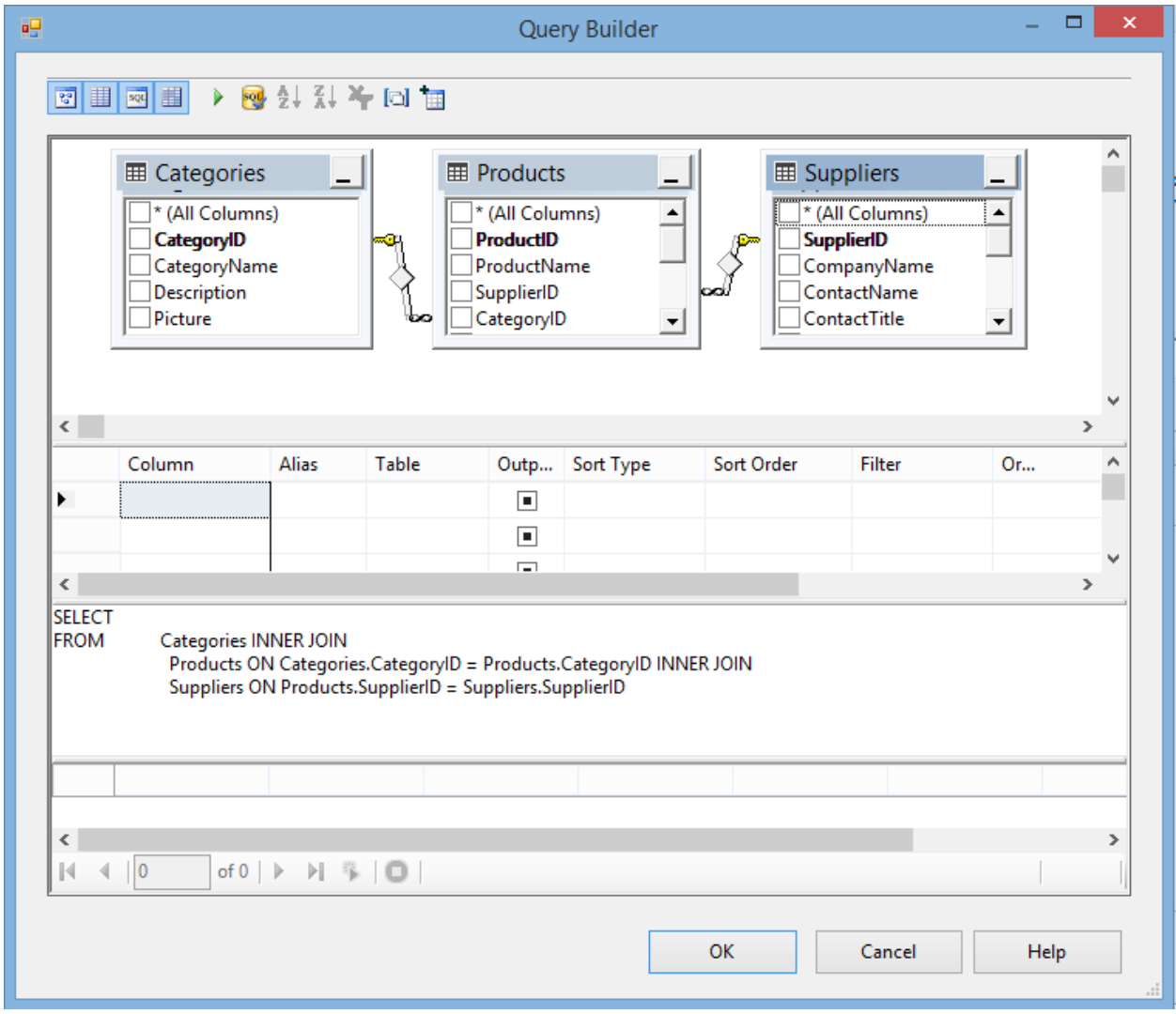
    a. In the **OLEDB Source Editor** change the **Data access mode** to **SQL Command**

b. Click the **Build Query…** button to visually design our SQL query using the **Query Builder!**



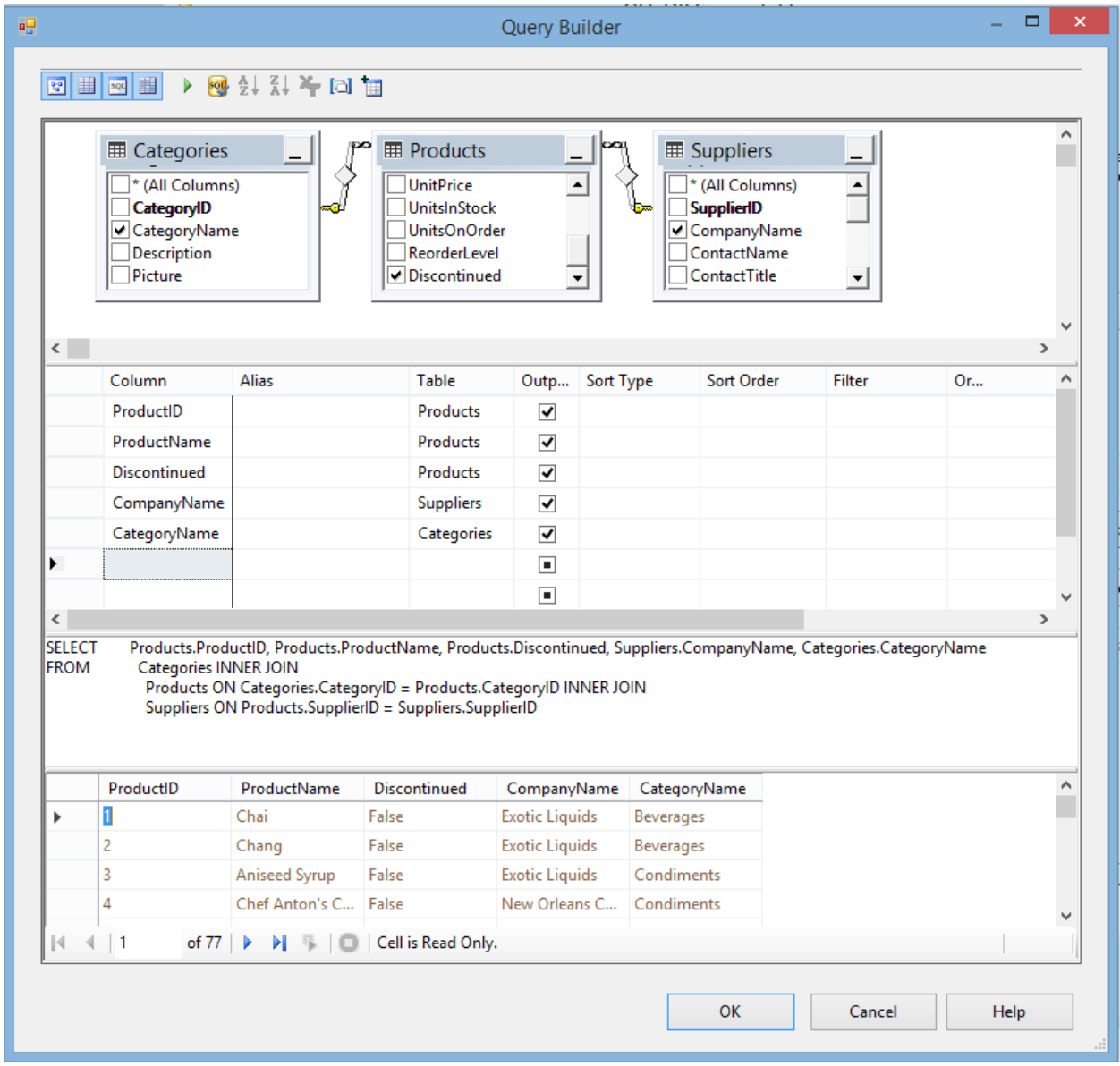
c. Click the **Add Table** button in the **toolbar,** then add the **Categories, Products,** and **Suppliers** tables and click close. The query builder is starting to create an SQL SELECT statement for you. Thanks to foreign keys, the join statements have been constructed

for us!



d. Next we add the columns we need. Simply click on the Column name from the tables at the top to add them:
From **Products**, add **ProductID, ProductName, Discontinued**.
From **Suppliers**, add **CompanyName** (this will be the Supplier Name)
From **Categories**, add **CategoryName**

e. Press the **Run** button in the **toolbar** to execute the query and see the output data.

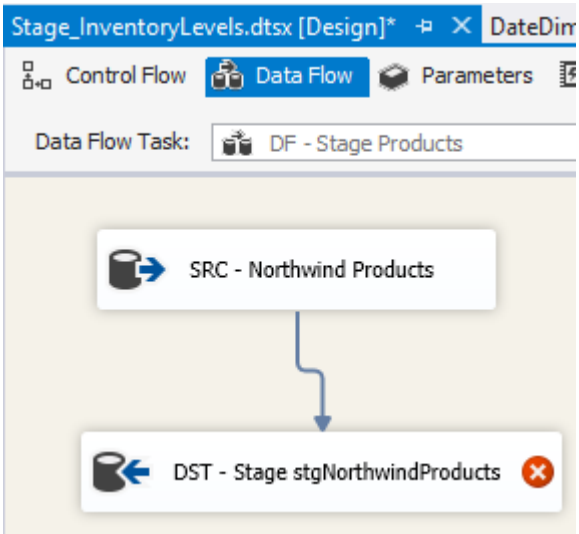


f. Press **OK** to save your query as the source for the **OLE DB Source Editor**.

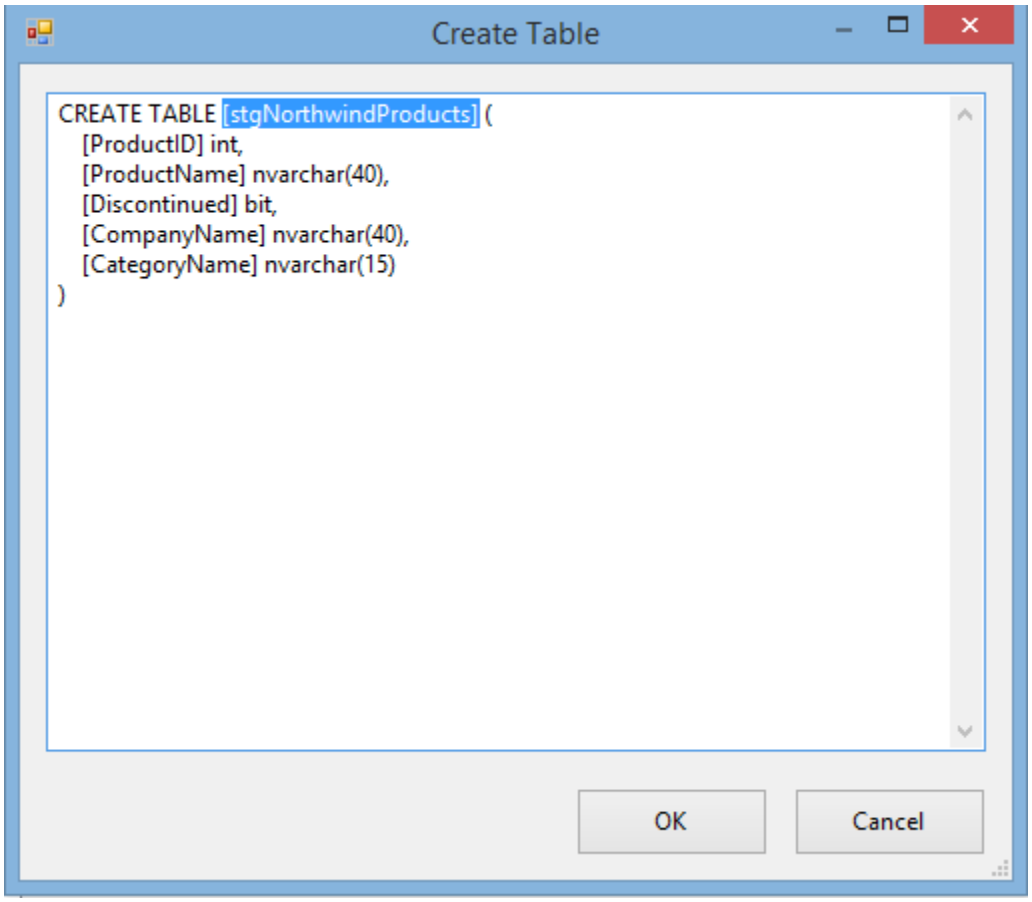
g. Press **OK** to close the source editor

4. Save your work.
5. Use the **Destination Assistant** to create the data target:
   a. Type: **SQL Server**
   b. Connection Manager → **ist722_*yournetid*_stage**
   c. Rename to → **DST - Stage stgNorthwindProducts**

6. Connect the output of the source to the input of the destination by dragging the blue arrow from source and dropping it on destination.



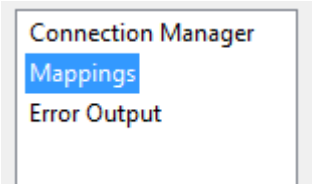
7. Double-click on the destination to configure it. This will bring up the **OLE DB Destination Editor**
    a. Click the **New…** button to create the target table. Remember there is no table in the stage database.
    b. You will see a create table statement which was automatically generated from the schema of the source query. Edit the first line of this code to create the table **[stgNorthwindProducts]**.



Click **OK** to create the table.

c. You should now see the **Name of the table or view** configured to be **[stgNorthwindProducts]**.

d. Our last step is to configure the mappings. Click on **mappings** on the left hand side of the window.



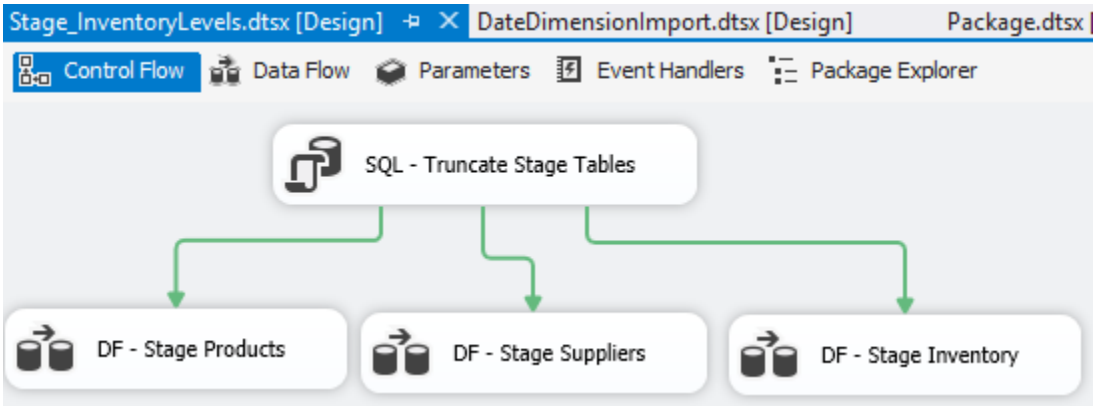
NOTE: This step is automatic because the source and destination have the same columns!

e. Click **OK** to complete the configuration of the destination.

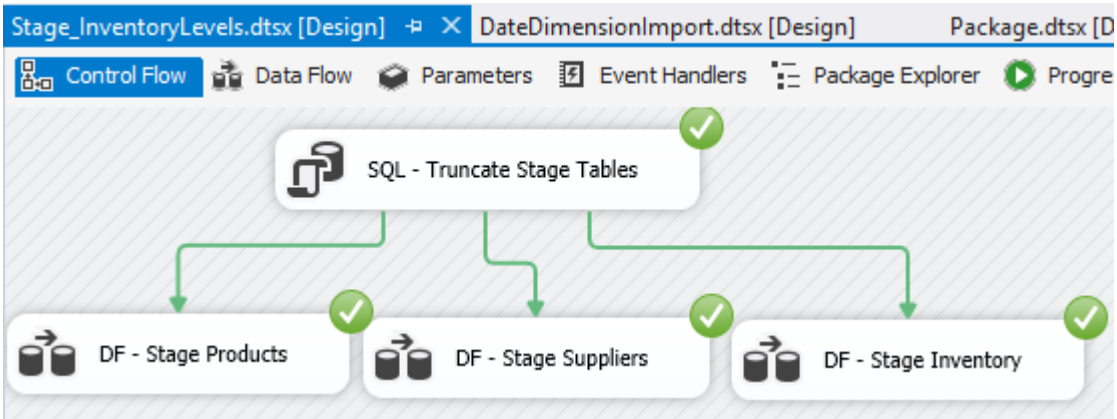
8. Save your work. Return to the **control flow** tab.
9. We need to truncate the table before the data flow occurs. Double click on the **SQL - Truncate Stage Tables** task.
   a. Set the **Connection** to **ist722_*yournetid*_stage**

b. Set the **SQL Statement** to:
```
truncate table stgNorthwindProducts;
```
c. Click **OK** to store the configuration.

10. Save your work. You have done enough configuration to test your package.



11. Press **[F5]** to execute your package. If everything goes to plan, it should execute without error:



IMPORTANT: If you get an error check the **Progress** and try to troubleshoot the issue before moving on. (We covered how to do this in the previous section).

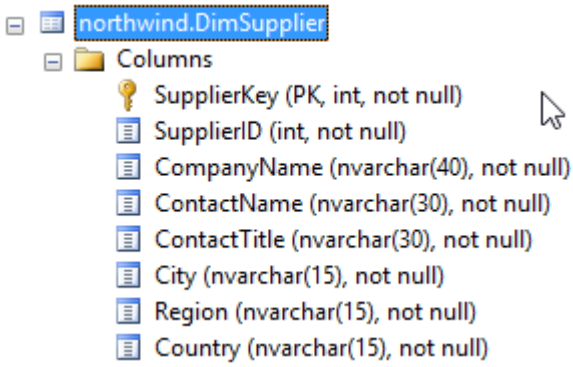
12. Stop package execution, by pressing [**Shift]+[F5]**
13. Verify there is data in the **stgNorthwindProducts** table in your **stage** database by running a simple SQL SELECT statement on the table. There should be **77 rows.**

Step 2.2.4: Staging Suppliers

Next we repeat the steps of the previous section, but instead stage Northwind Supplier data.

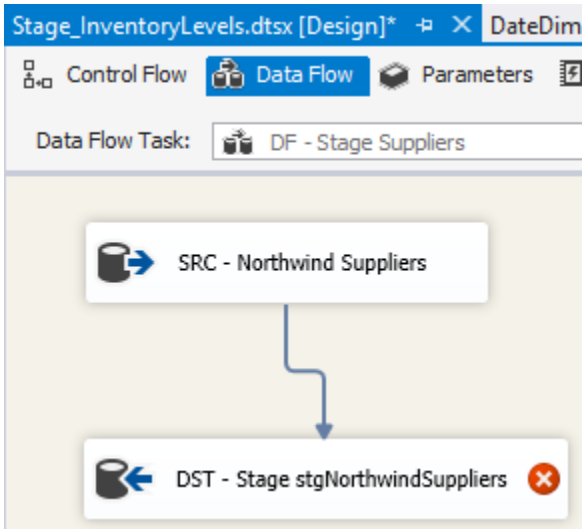
1. Double click on **DF - Stage Suppliers** to open the data flow.
2. Use the **Source Assistant** to create the data source:
   a. Type: **SQL Server**
   b. Connection Manager → **Northwind**
   c. Rename to → **SRC - Northwind Suppliers**

3. Double-click on the data source to configure it. This opens the **OLEDB Source Editor**. Once again we need to build a query to match the target data of the dimension:



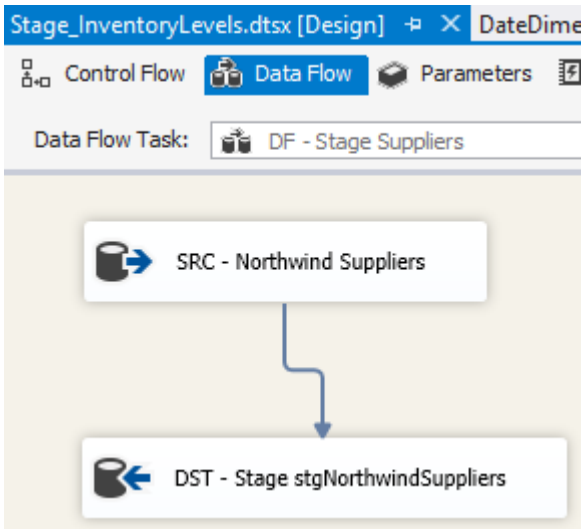
We will need SupplierID, CompanyName, ContactName, ContactTitle, City, Region, and Country

   a. For Data Access Mode select **SQL Command**
   b. Use the **Build Query…** button to create the query. Here's the SQL if you're lazy:
      `SELECT SupplierID, CompanyName, ContactName, ContactTitle, City, Region, Country FROM Suppliers`
   c. Click **OK** to store your source configuration.
4. Save your work.
5. Use the **Destination Assistant** to create the data target:
   a. Type: **SQL Server**
   b. Connection Manager → **ist722_*yournetid*_stage**
   c. Rename to → **DST - Stage stgNorthwindSuppliers**
6. Connect the output of the source to the input of the destination by dragging the blue arrow from source and dropping it on destination.

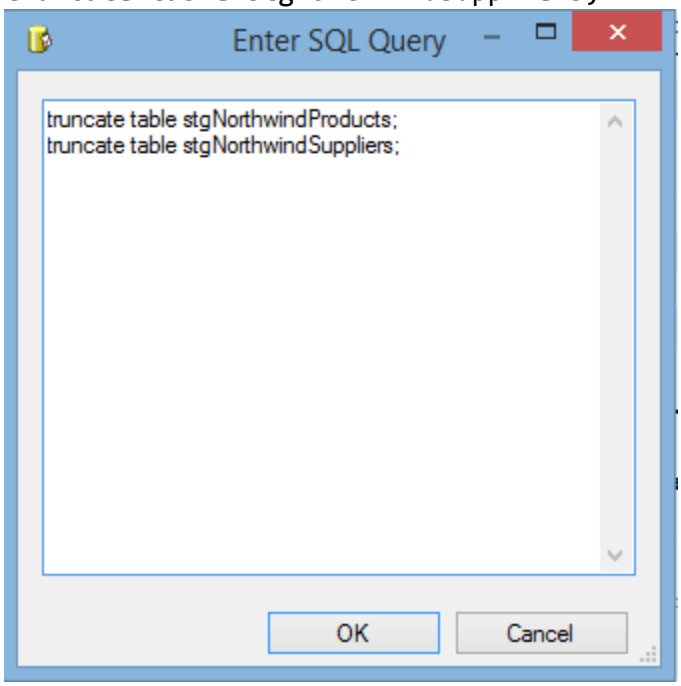


7. Double-click on the destination to configure it. This will bring up the **OLE DB Destination Editor**
   a. Create a new target table, named **[stgNorthwindSuppliers]**. Again there's no table in the **stage** database right now.
   b. Click on **mappings** to automatically configure the mappings.

c. Click **OK** to store the configuration and close the **OLE DB Destination Editor.**



8. Save your work. Click on the **Control Flow** tab.
9. Now we need to add the truncate table statement to the **SQL - Truncate Stage Tables** task. Double click on it to bring up the **Execute SQL Task Editor**.
   a. Under **SQL Statement** click the **Builder Button [...]** to configure the SQL query.
   b. Add the truncate table statement under the existing statement.
   `truncate table stgNorthwindSuppliers;`



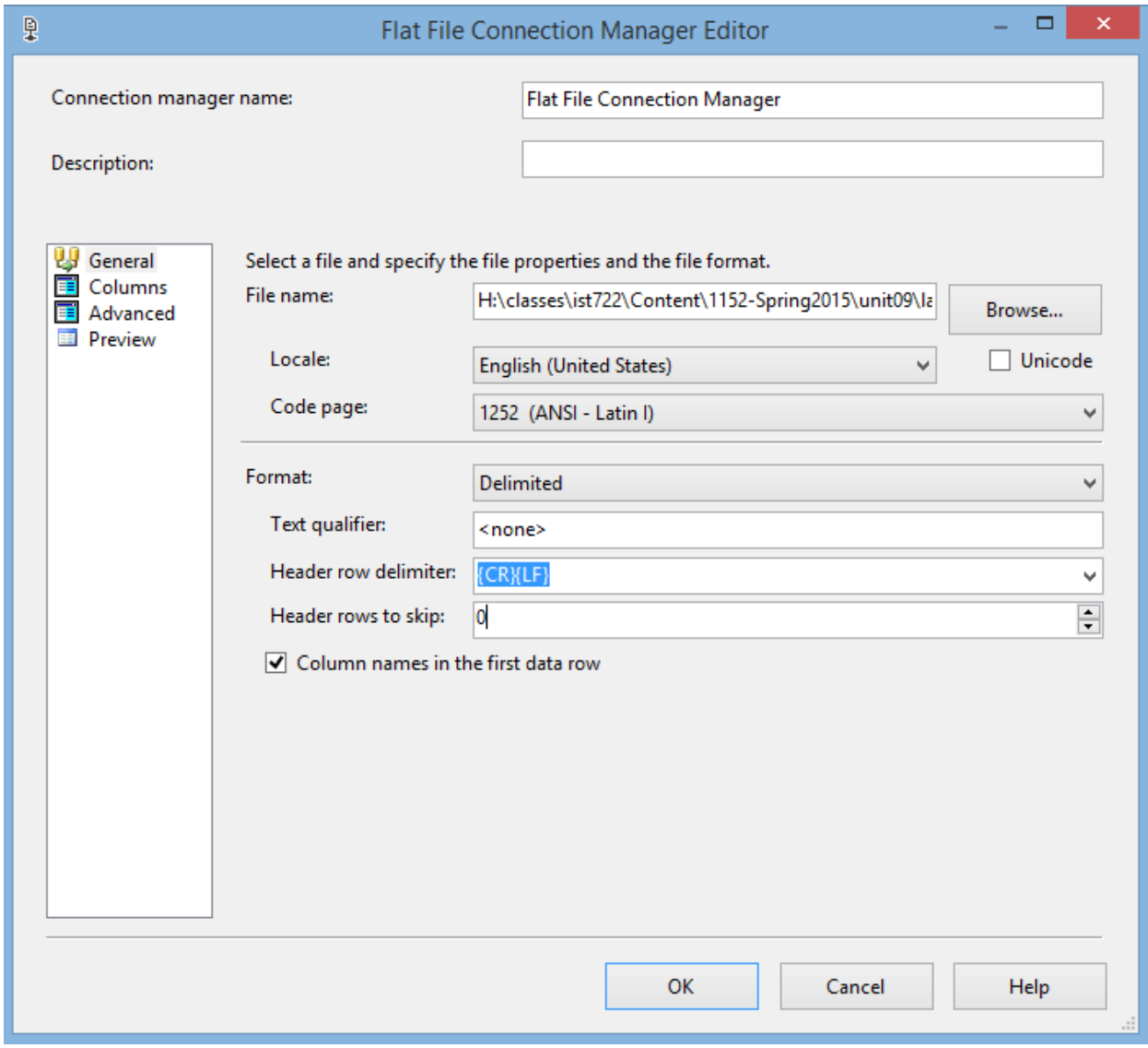
   Click **OK** to store the SQL statement.
   c. Click **OK** to close the **Execute SQL Task Editor**.
10. Save your work. You have done enough configuration to test your package!
11. Execute the package by pressing **[F5]**. Troubleshoot any issues that may arise. Stop package execution with **[Shift]+[F5]**.

12. Verify there is data in the **stgNorthwindSuppliers** table in your **stage** database by running a simple SQL SELECT statement on the table. There should be **29 rows.**

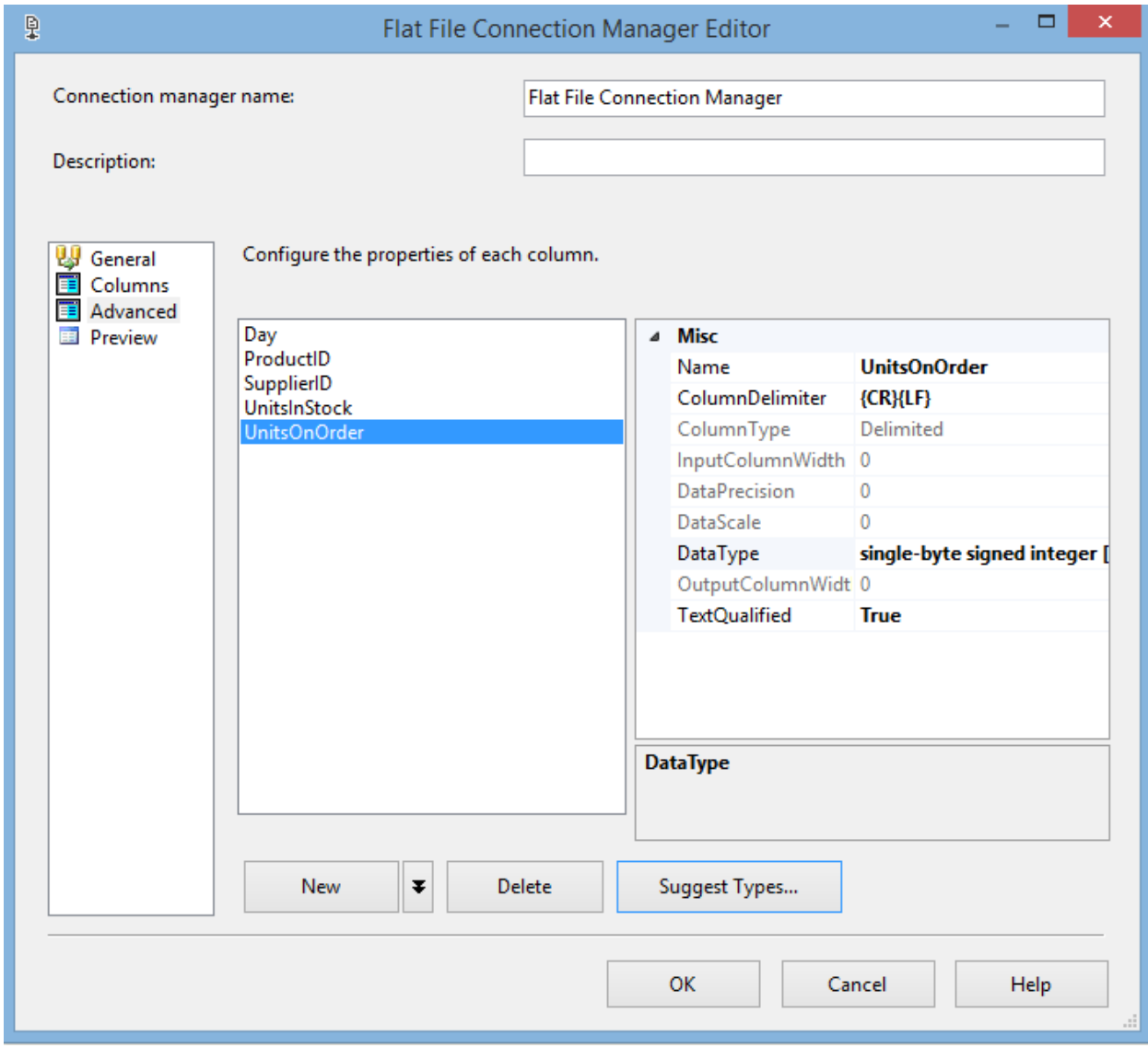
Step 2.2.5: Staging Inventory

The last data flow is next - staging Northwind Inventory data. There's a slight wrinkle in this data flow as the source is a text file.

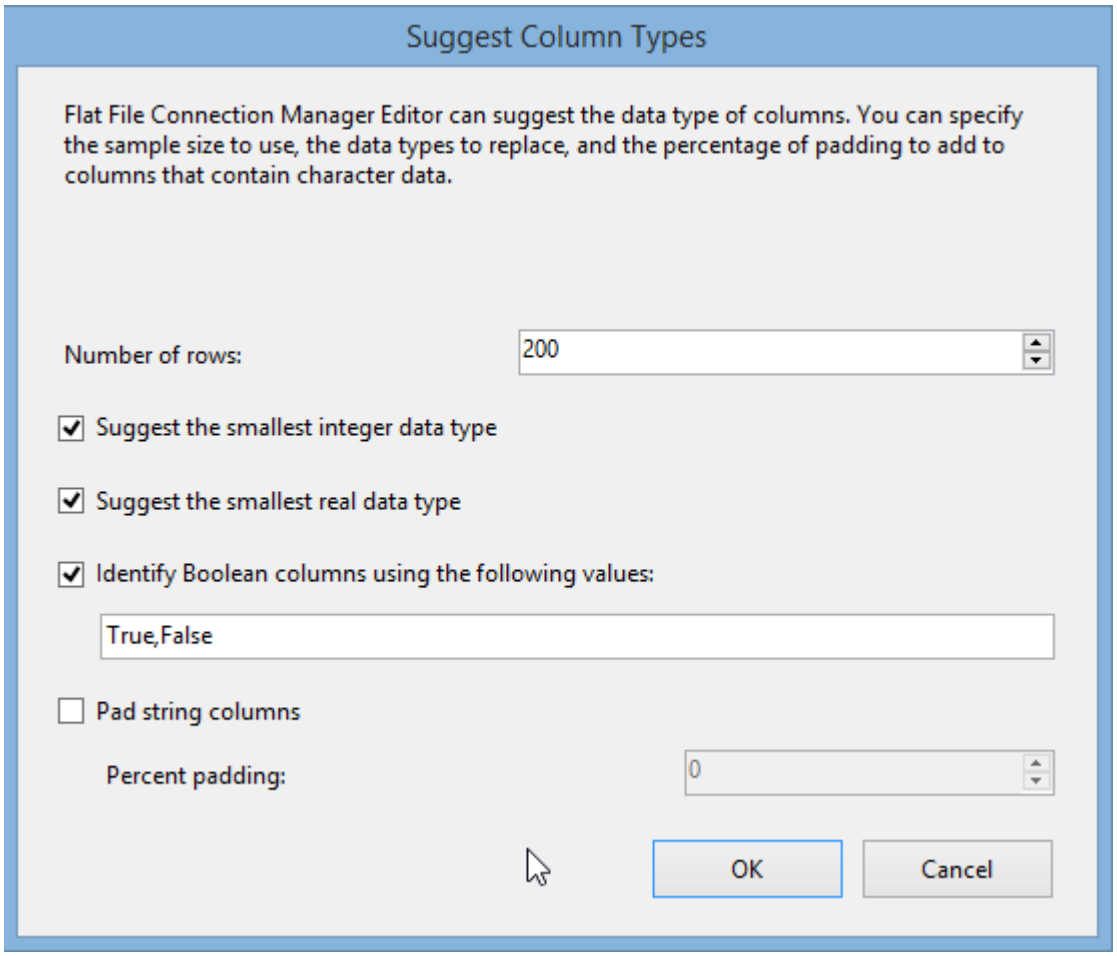
1. Double click on **DF - Stage Inventory** to open the data flow.
2. Use the **Source Assistant** to create the data source:
    a. Type: **Flat File**
    b. Connection Manager → **New…** (since we have no connection for this)
    c. The **Flat File Connection Manager Editor** dialog appears. Click the **Browse…** button to locate the **NorthwindDailyInventoryLevelsOneWeek.csv** file



d. By default, all columns are treated as text. We need to change that. Click on the **Advanced** setting in the left hand side of the dialog.



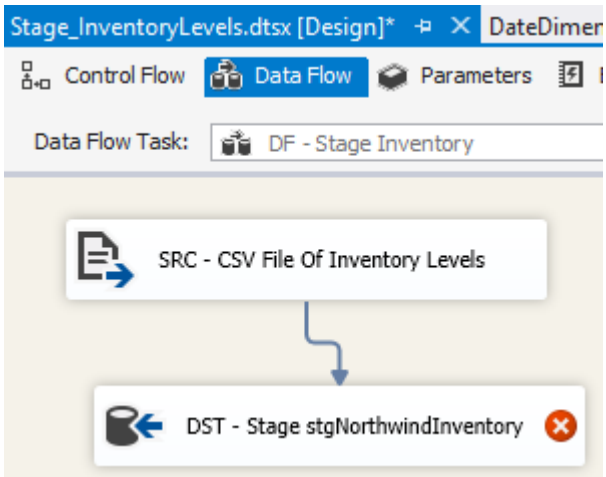
e. Click the **Suggest Types…** button to determine the data type of the columns.



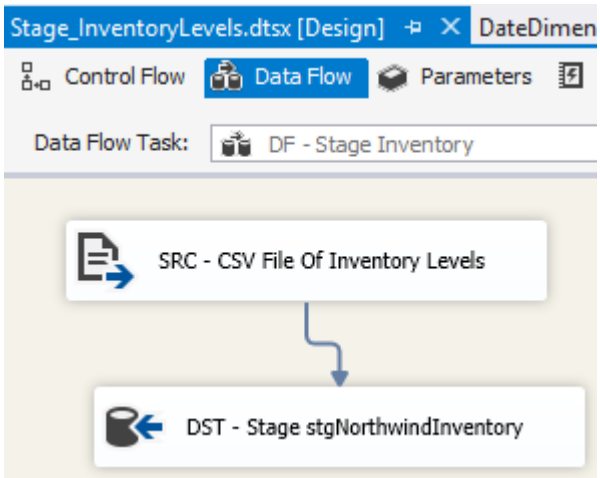
f. Click **OK** to detect the data types.
g. Click **OK** to close the connection Manager
h. Rename to → **SRC - CSV File Of Inventory Levels**

3. Save your work.
4. Use the **Destination Assistant** to create the data target:
   a. Type: **SQL Server**
   b. Connection Manager → **ist722_*yournetid*_stage**
   c. Rename to → **DST - Stage stgNorthwindInventory**

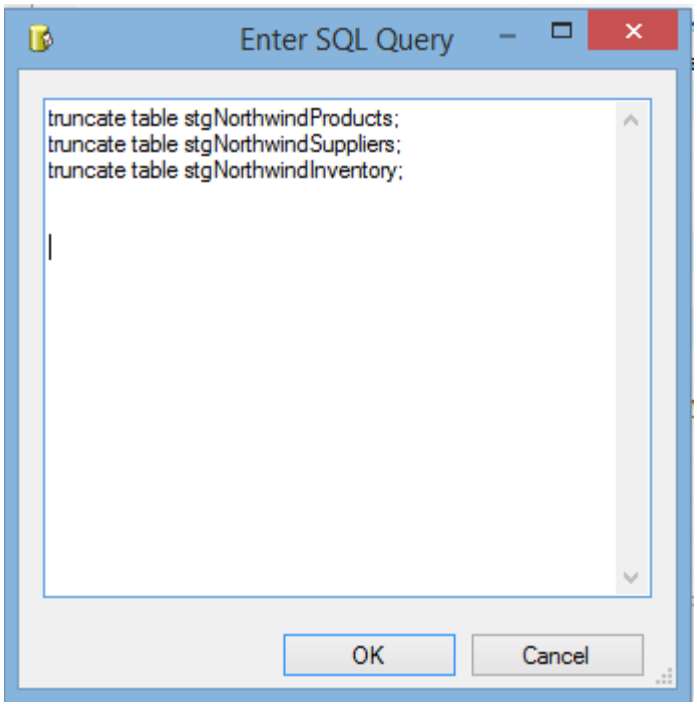
5. Connect the output of the source to the input of the destination by dragging the blue arrow from source and dropping it on destination



6. Double-click on the destination to configure it. This will bring up the **OLE DB Destination Editor**
   a. Create a new target table, named **[stgNorthwindInventory]**. Again there's no table in the **stage** database right now.
   b. Click on **mappings** to automatically configure the mappings.
7. Click **OK** to store the configuration and close the **OLE DB Destination Editor.**



8. Save your work. Click on the **Control Flow** tab.
9. Now add the truncate table statement to the **SQL - Truncate Stage Tables** task. Double click on it to bring up the **Execute SQL Task Editor**.
   d. Under **SQL Statement** click the **Builder Button [...]** to configure the SQL query.
   e. Add the truncate table statement under the existing statement.
   `truncate table stgNorthwindInventory;`



Click **OK** to store the SQL statement.

f. Click **OK** to close the **Execute SQL Task Editor**.

10. Save your work. Time to test!
11. Execute the package by pressing **[F5]**. Troubleshoot any issues that may arise. Stop package execution with **[Shift]+[F5]**.
12. Verify there is data in the **stgNorthwindInventory** table in your **stage** database by running a simple SQL SELECT statement on the table. There should be **616 rows.**

That's it! You've completed the entire **Stage_InventoryLevels.dtsx** package.

Part 2.3: Loading the Inventory Levels Dimensional Model

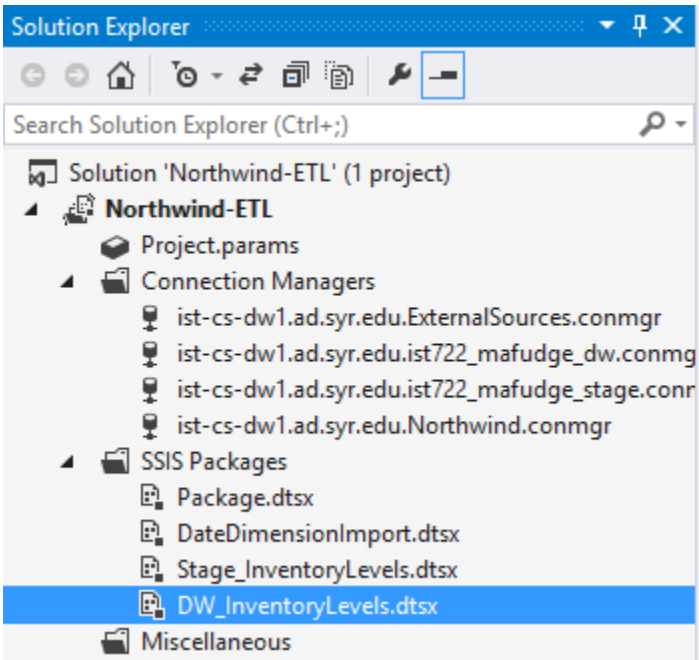
In this final part we will load from the stage tables into the Fact and Dimension table of our data warehouse. Along the way we will need to transform our data to the requirements of the source tables. Let's get started!

Step 2.3.1: Create the Package

Let's start by creating the **DW_InventoryLevels.dtsx** package to contain our SSIS logic.

1. In the **Solution Explorer** window, right-click on **SSIS Packages** and select **New SSIS Package** from the menu.

2. Rename the package **DW_InventoryLevels.dtsx**

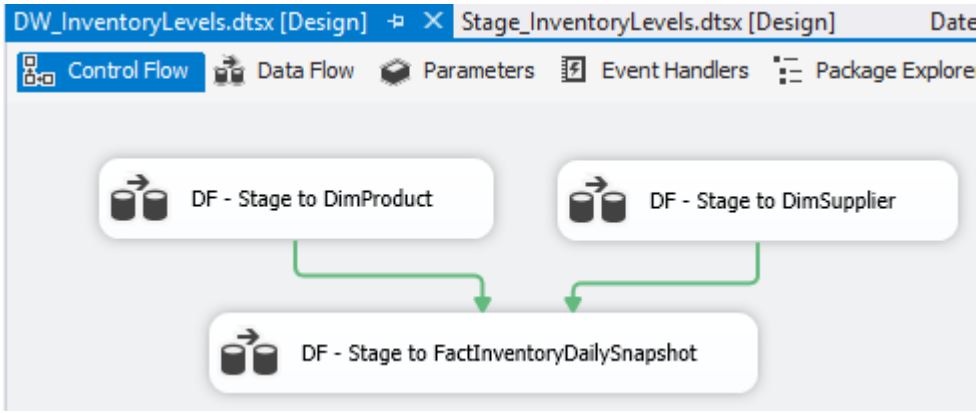


3. Double-click on the **DW_InventoryLevels.dtsx** package to open it in the **design surface**.

Step 2.3.2: Setup the Control Flow

Next need to setup the base control flow for the package. We will process the dimensions concurrently and their success will depend on the fact table processing.

1. Add **three Data Flow Tasks** to your design surface.
2. Rename the 3 Data flow Tasks:
   a. **DF – Stage to DimProduct**
   b. **DF – Stage to DimSupplier**
   c. **DF - Stage to FactInventoryDailySnapshot**
3. Connect the green arrows from each dimension to the fact table.



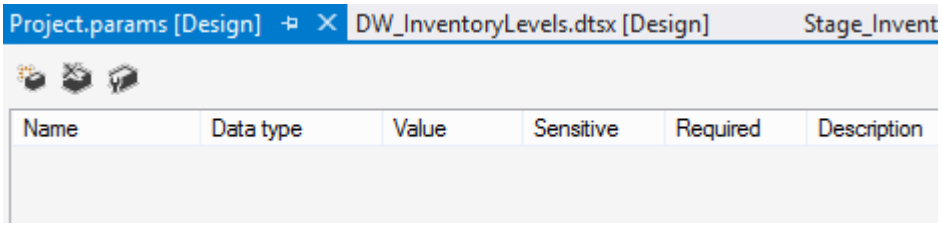
4. Save your work.

Step 2.3.3: Setup InitialDate Project Parameter

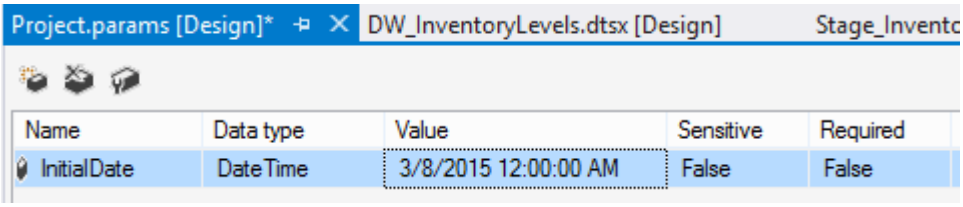
Since the fact table is a daily periodic snapshot, we need some method of determining the date that the ETL job was executed and therefore the inventory levels added. Normally this would be the current

date, but since this is a simulated lab activity I prefer us to all use the same date. Therefore we will set a parameter to "simulate" the current date.

1. In **Solution Explorer** double-click on **Project.params** this will open the **Project.params** tab in the designer.



2. Let's create a parameter, in the toolbar, click the **Add Parameter** button.
3. Configure the parameter as follows:
   a. Name → **InitialDate**
   b. Data Type → **DateTime**
   c. Value → **3/8/2015 12:00:00AM**

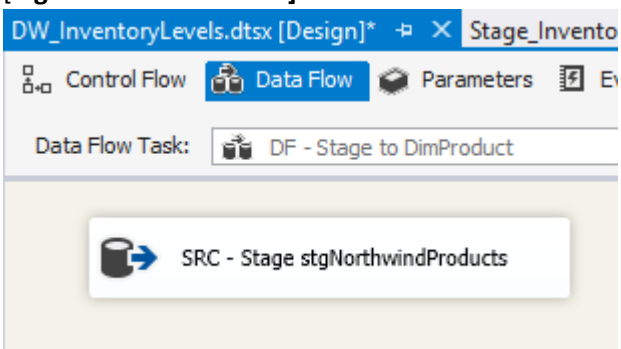


4. Save your work and close the Project.params tab.

Step 2.3.4: Data Flow Stage to DimProduct

It's time to work through the logic to import data from **stgNorthwindProducts** into **northwind.DimProduct**. We will need to transform data along the way as well as track type 2 SCD changes. Here's the procedure.

1. Double click on **DF - Stage to DimProduct** to open the data flow.
2. Use the **Source Assistant** to create the data source:
   a. Type: **SQL Server**
   b. Connection Manager → **ist722_*yournetid*_stage**
   c. Rename to → **SRC - Stage stgNorthwindProducts**
3. Double-click on the data source to configure it. The source we will use is the **[stgNorthwindProducts]** table. Select it for the **Name of the table or the view**. Click **OK** to close.

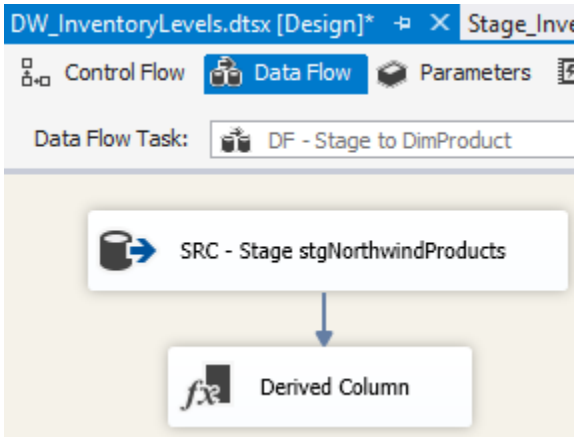


4. Here comes the challenging part. Our staged data does not match the schema of our dimension. Observe:

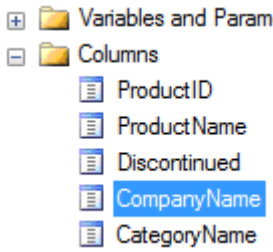
Stage Database	DW Database
dbo.stgNorthwindProducts	northwind.DimProduct
Columns	Columns
ProductID (int, null)	ProductKey (PK, int, not null)
ProductName (nvarchar(40), null)	ProductID (int, not null)
Discontinued (bit, null)	ProductName (nvarchar(40), not null)
CompanyName (nvarchar(40), null)	Discontinued (nchar(1), not null)
CategoryName (nvarchar(15), null)	SupplierName (nvarchar(40), not null)
	CategoryName (nvarchar(15), not null)

We need to convert the bit column Discontinued into a Y/N column in the Dimension.
We also need to change CompanyName to SupplierName in the Dimension

a. Drag and drop the **Derived Column** Derived Column tool onto the design surface.
b. Connect the output from **SRC - Stage stgNorthwindProducts** to the input of the **Dervied Column**.



c. Rename the derived column **DC - Transform Discontinued and SupplierName**.
d. Double-click on it to configure. This opens the **Derived Column Transformation Editor** dialog.
e. Under **Columns** drag **CompanyName**

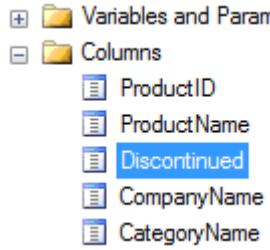


to the area under expression and drop.

Derived Column Name	Derived Column	Expression	Data Type
Derived Column 1	<add as new column>	[CompanyName]	Unicode string [DT_WSTR]

then change the name from **Derived Column 1** to **SupplierName**

f. Under **Columns** drag **Discontinued**



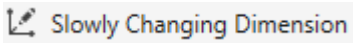
to the area under [Company Name] and drop. Edit the expression to say:
[Discontinued] ? "Y" : "N"

Derived Column Name	Derived Column	Expression	Data Type
SupplierName	<add as new column>	CompanyName	Unicode string [DT_WSTR]
DiscontinuedYN	<add as new column>	Discontinued ? "Y" : "N"	Unicode string [DT_WSTR]

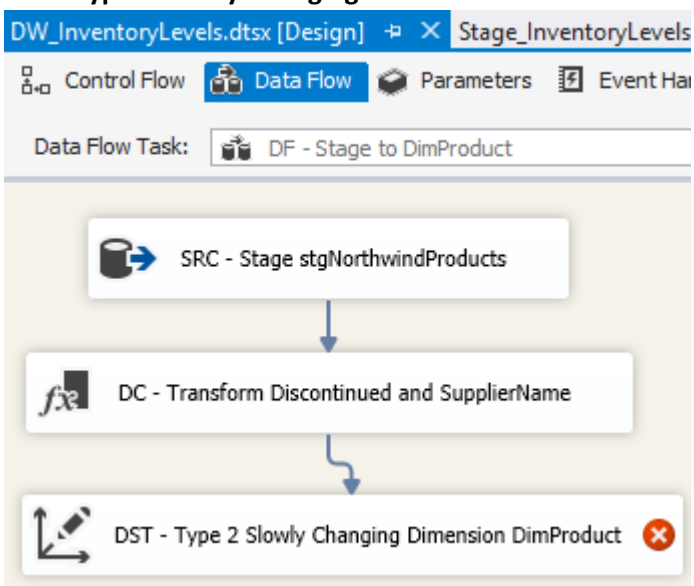
then change the name from **Derived Column 1** to **DiscontinuedYN**

NOTE: The ? : is an "inline if" operator. When Discontinued == true "Y" is returned otherwise "N" is returned.

g. When you've completed the transformations, click **OK** to close.

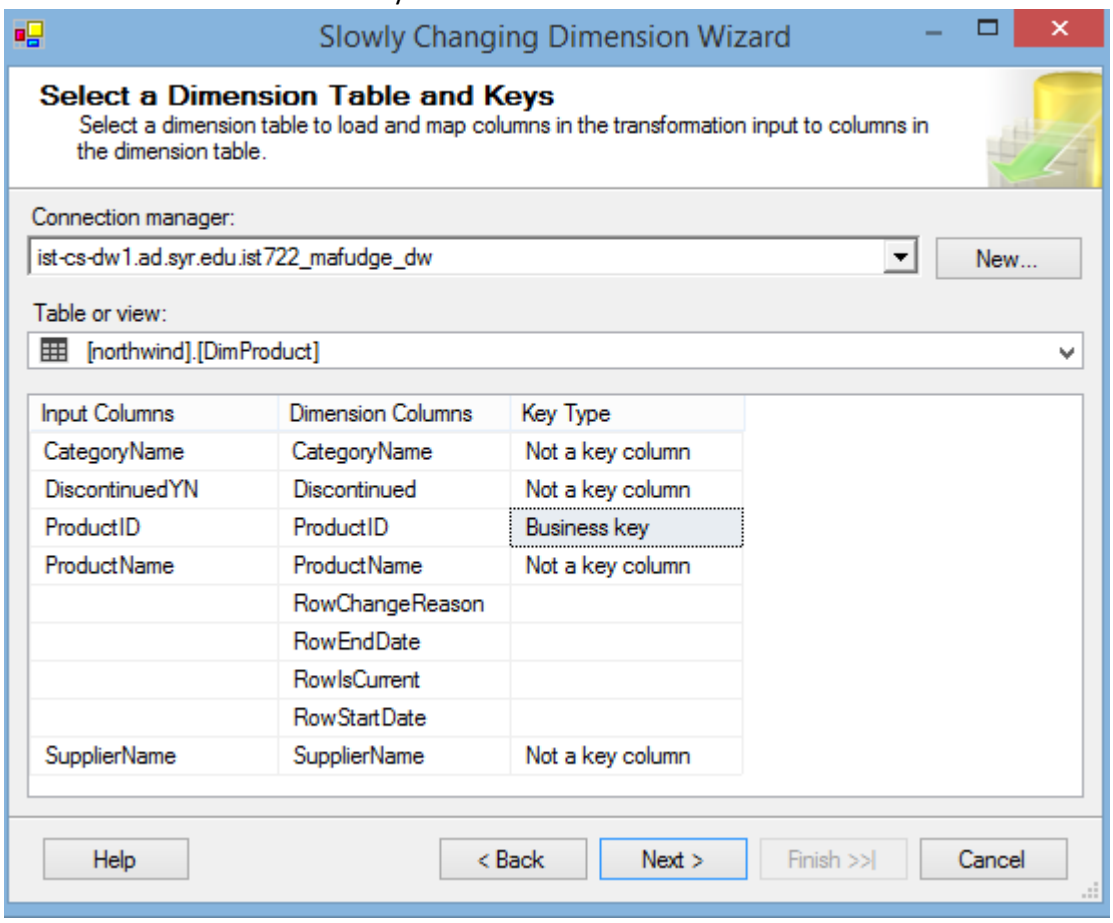
5. Save your work.
6. Now that we have everything we need to populate the dimension it is time to use the **Slowly Changing Dimension** 

destination to populate our Type 2 dimension.
   a. Drag and drop the **Slowly Changing Dimension** destination on the design surface.
   b. Rename it → **DST - Type 2 Slowly Changing Dimension DimProduct**
   c. Connect the output of **DC - Transform Discontinued and SupplierName** to the input of **DST - Type 2 Slowly Changing Dimension DimProduct**



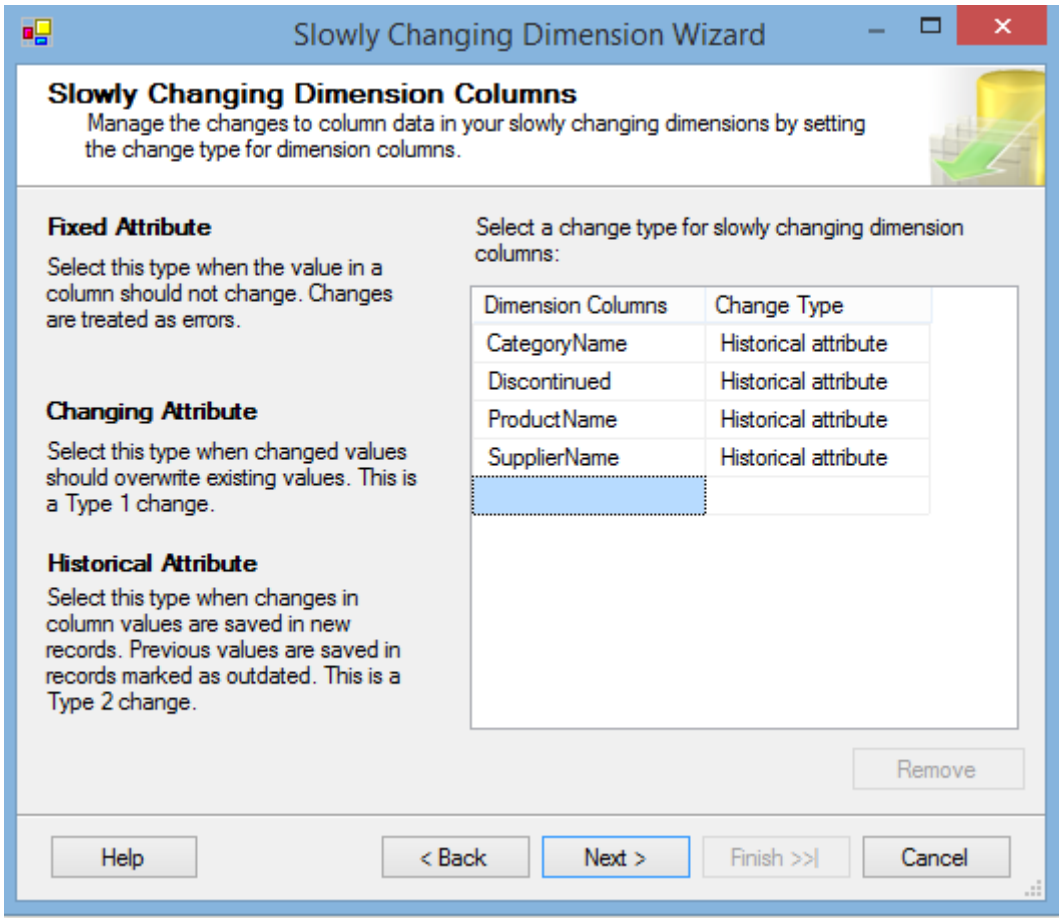
7. Double click on **DST - Type 2 Slowly Changing Dimension DimProduct** to configure it.
   a. Click **Next >** past the first page of the wizard.
   b. Use **ist722_*yournetid*_dw** as the connection.

c. Select **northwind.DimProduct** as the dimension.
d. Configure the input Column: DiscontinuedYN → Discontinued
e. Set **ProductID** as the business key



f. Click **Next >**

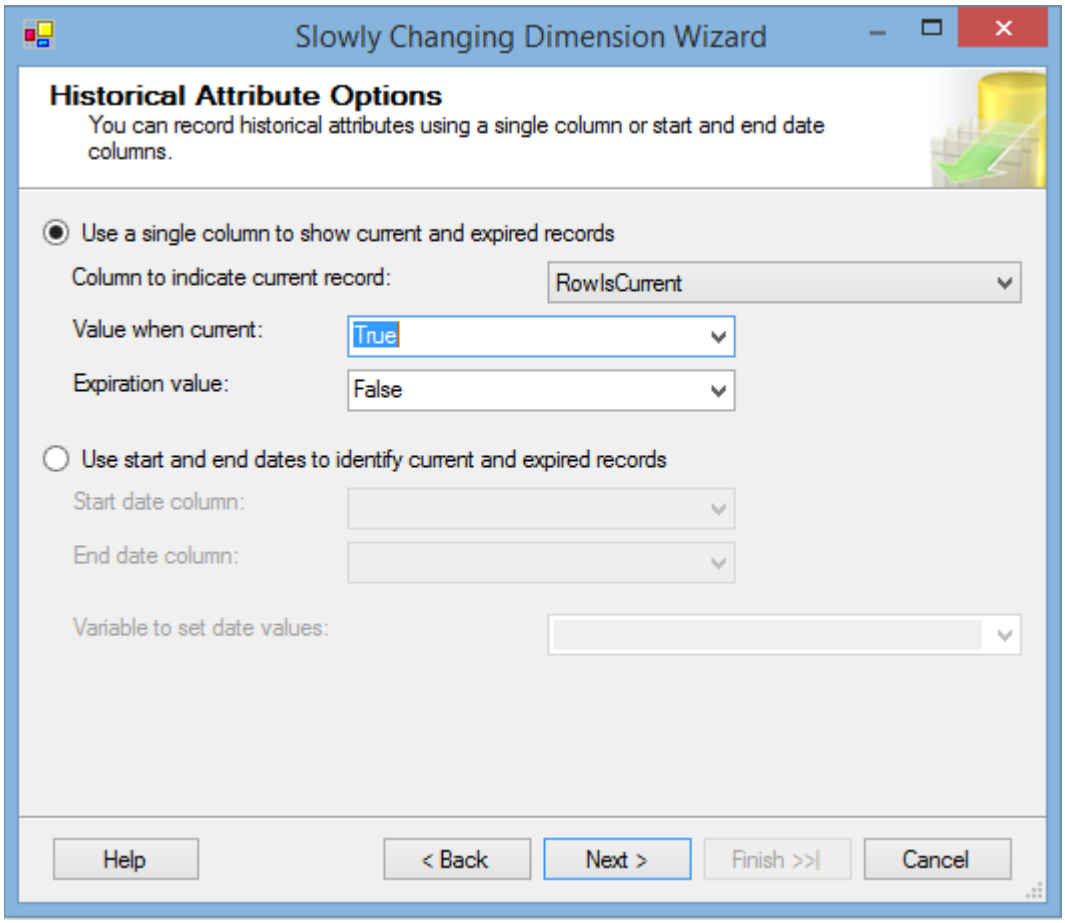
g. Add CategoryName, Discontinued, ProductName and SupplierName as **Historical Attributes**



h. Click **Next >**

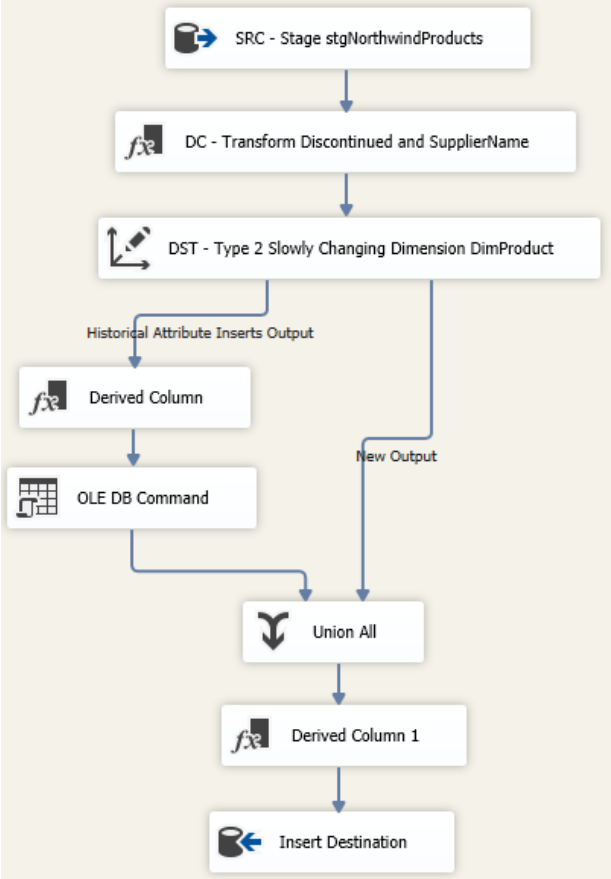
i. Now we must configure the type 2 SCD metadata columns.
Select **RowIsCurrent** as the column to indicate current record. And value when current

to **True**.



j. Click **Next >**
k. **Turn off** inferred member support, by **clearing the checkbox.** Click **Next >**
l. Click **Finish** to generate the Type 2 SCD logic.

8. Save your work.



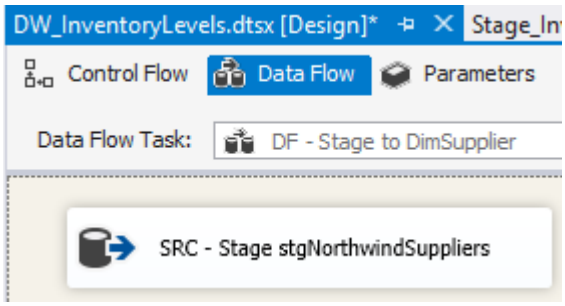
9. Execute the package. Make sure it works. Troubleshoot it if you have problems.
10. Verify there are **78 rows** in the **northwind.DimProduct** dimension. (77 ETL rows + the unknown member)

Step 2.3.5: Data Flow Stage to DimSupplier

Now we're going to repeat this process for **northwind.DimSupplier**:

1. Double click on **DF - Stage to DimSupplier** to open the data flow.
2. Use the **Source Assistant** to create the data source:
    a. Type: **SQL Server**
    b. Connection Manager → **ist722_*yournetid*_stage**
    c. Rename to → **SRC - Stage stgNorthwindSuppliers**
3. Double-click on the data source to configure it. The source we will use is the **[stgNorthwindSuppliers]** table. Select it for the **Name of the table or the view**. Click **OK** to

close.



4. Of course our staged data does not match the schema of our dimension. Observe:

Stage Database	DW Database
dbo.stgNorthwindSuppliers	northwind.DimSupplier
Columns	Columns
SupplierID (int, null)	SupplierKey (PK, int, not null)
CompanyName (nvarchar(40), null)	SupplierID (int, not null)
ContactName (nvarchar(30), null)	CompanyName (nvarchar(40), not null)
ContactTitle (nvarchar(30), null)	ContactName (nvarchar(30), not null)
City (nvarchar(15), null)	ContactTitle (nvarchar(30), not null)
Region (nvarchar(15), null)	City (nvarchar(15), not null)
Country (nvarchar(15), null)	Region (nvarchar(15), not null)
	Country (nvarchar(15), not null)

This time our transformation is a little more subtle. There are nulls in the source of **Region** which must be replaced with **"N/A"** in the Dimension. Nulls in the dimension are bad because when a user is exploring a dimension model the information is just blank and therefore confusing.

a. Drag and drop the **Derived Column**  tool onto the design surface.
b. Connect the output from **SRC - Stage stgNorthwindProducts** to the input of the **Dervied Column**.
   Rename the derived column **DC - Replace Nulls with Defaults**
c. Double-click on it to configure. This opens the **Derived Column Transformation Editor** dialog.
d. Under **Columns** drag **Region** to the area under **Expression** and drop.

Derived Column Name	Derived Column	Expression	Data Type
Derived Column 1	<add as new column>	[Region]	Unicode string [DT_WSTR]

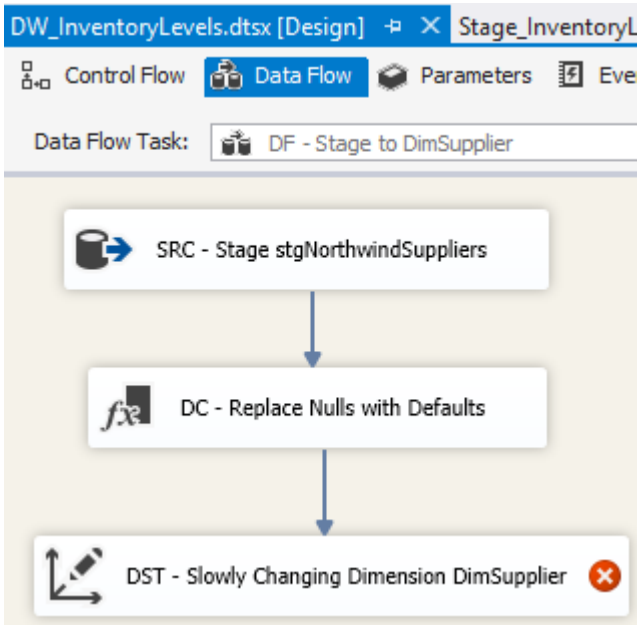
   Change the name from **Derived Column 1** to **RegionNA**
   Change the Expression from **[Region]** to **LEFT(REPLACENULL(Region,"N/A"),15)**
   This expression replaces nulls with "N/A" and truncates the data to under 15 characters.

Derived Column Name	Derived Column	Expression	Data Type	Length
RegionNA	<add as new column>	LEFT(REPLACENULL(Region,"N/A"),15)	Unicode string [DT_WSTR]	15

e. Click **OK** to close.

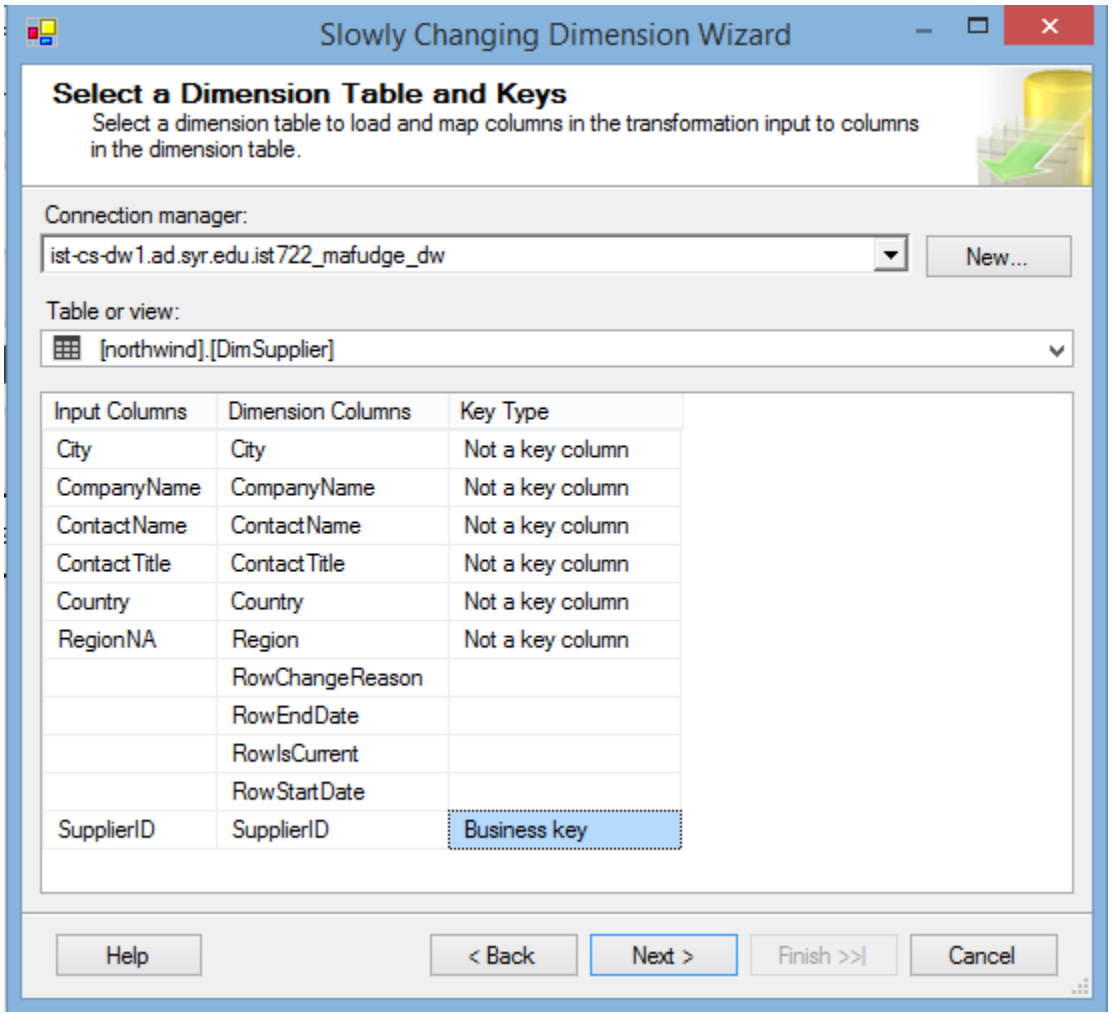
5. Save your work.
6. Again, we're ready to populate our Type 2 dimension.
   a. Drag and drop the **Slowly Changing Dimension** destination on the design surface.
   b. Rename it → **DST - Type 2 Slowly Changing Dimension DimSupplier**

c. Connect the output of **DC - Replace Nulls with Defaults** to the input of **DST - Slowly Changing Dimension DimSupplier**



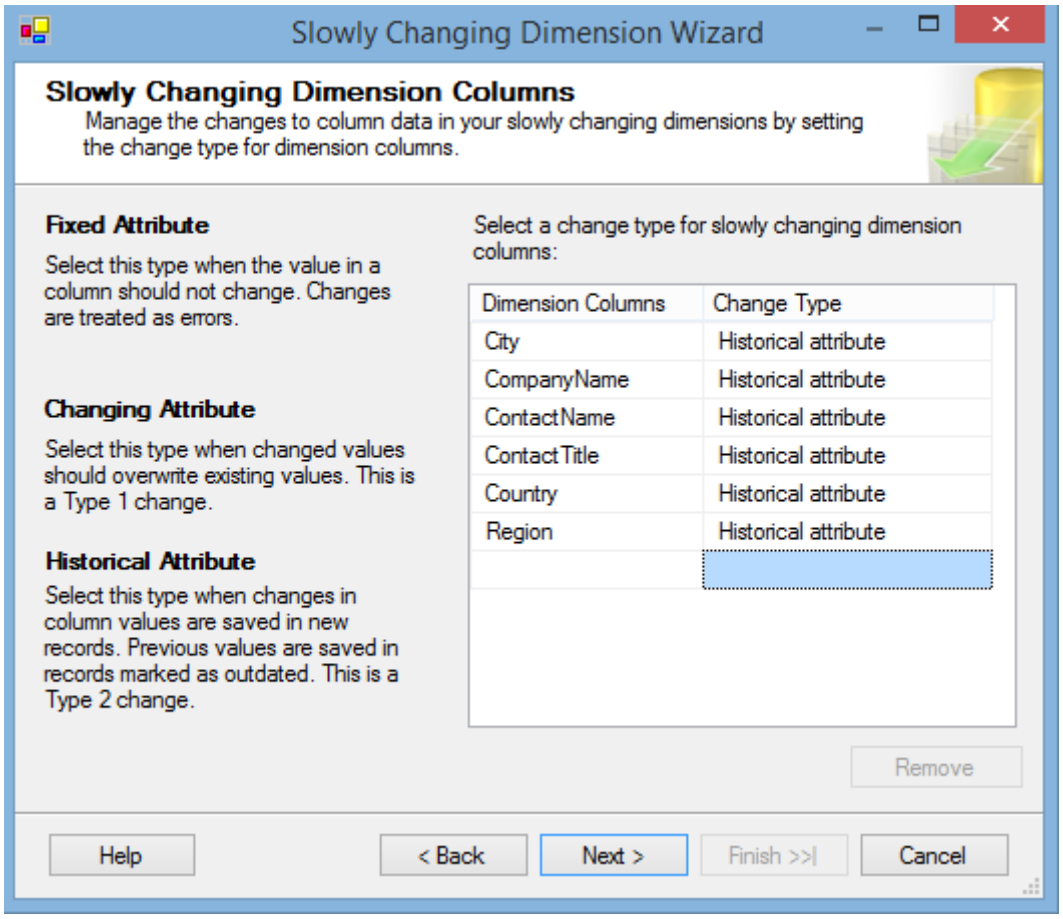
7. Double click on **DST - Type 2 Slowly Changing Dimension DimSupplier** to configure it.
   a. Click **Next >** past the first page of the wizard.
   b. Use **ist722_*yournetid*_dw** as the connection.
   c. Select **northwind.DimSupplier** as the dimension.
   d. Configure the input Column: RegionNA → Region

e. Set **SupplierID** as the business key.



f. Click **Next >**

g. Add City, CompanyName, ContactName, ContactTitle, Country, and Region as **Historical Attributes**



h. Click **Next >**

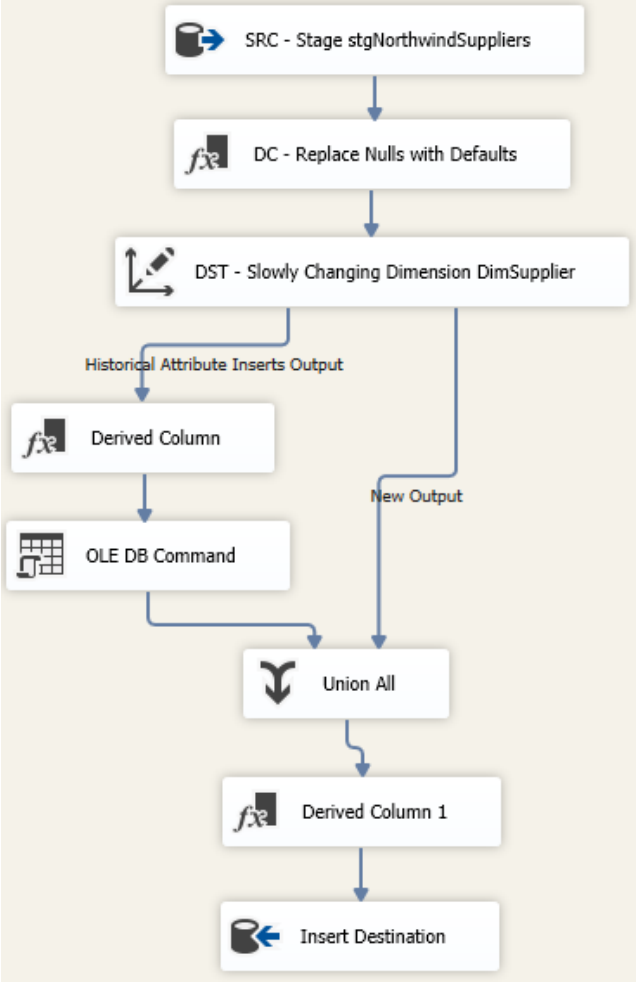
i. Again configure the type 2 SCD metadata columns.
Select **RowIsCurrent** as the column to indicate current record. And value when current to **True**.

j. Click **Next >**

k. **Turn off** inferred member support, by **clearing the checkbox.** Click **Next >**

l. Click **Finish** to generate the Type 2 SCD logic.

8. Save your work.



9. Execute the package. Make sure it works. Troubleshoot it if you have problems.
10. Verify there are **30 rows** in the **northwind.DimSupplier** dimension. (29 ETL rows + the unknown member)

Step 2.4.6: Data Flow Stage to FactInventoryDailySnapshot

In this last step we complete this ETL for the fact table. This is a more complicated process as we must look up the Dimension primary keys using our business keys (this is called the surrogate key pipeline).

1. Double click on **DF - Stage to FactInventoryDailySnapshot** to open the data flow.
2. Use the **Source Assistant** to create the data source:
    a. Type: **SQL Server**
    b. Connection Manager → **ist722_*yournetid*_stage**
    c. Rename to → **SRC - Stage stgNorthwindInventory**
3. Double-click on the data source to configure it. The source we will use is the **[stgNorthwindInventory]** table. Select it for the **Name of the table or the view**. Click **OK** to close.
4. Since our staged data does not match the schema of our dimension, we've got some conversions to process:

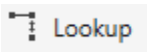
Stage Database	DW Database
dbo.stgNorthwindInventory Columns Day (int, null) ProductID (int, null) SupplierID (int, null) UnitsInStock (int, null) UnitsOnOrder (int, null)	northwind.FactInventoryDailySnapshot Columns ProductKey (PK, FK, int, not null) SupplierKey (FK, int, not null) DateKey (PK, FK, int, not null) UnitsInStock (int, not null) UnitsOnOrder (int, not null)

We need to convert the **Day** column into a DateKey by adding in our **InitialDate** parameter. Typically we would require some fact calculation at this step but the nature of facts in this example do not warrant it.

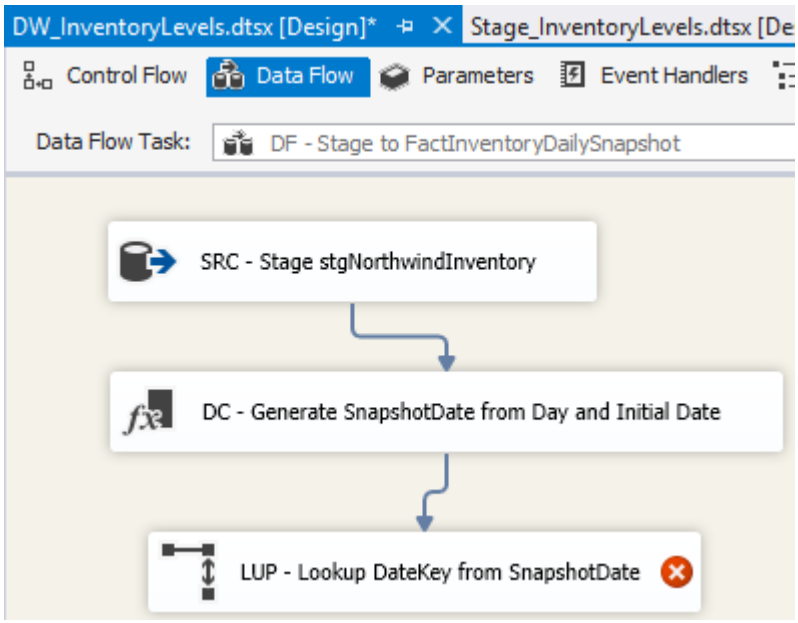
a. Drag and drop the **Derived Column** tool onto the design surface.
b. Connect the output from **SRC - Stage stgNorthwindInventory** to the input of the **Dervied Column**.
c. Rename the derived column **DC - Generate SnapshotDate from Day and Initial Date**
d. Double-click on it to configure. This opens the **Derived Column Transformation Editor** dialog.
e. Under **Columns** drag **Day** to the area under **Expression** and drop.
   Change the name from **Derived Column 1** to **SnapshotDate**
   Change the Expression from **[Day]** to **DATEADD("d",Day,@[$Project::InitialDate])**
   This adds **Day** days to the **InitialDate**

Derived Column Name	Derived Column	Expression	Data Type
SnapshotDate	<add as new column>	DATEADD("d",Day,@[$Project::InitialDate])	database timestamp [DT_DBTIMESTAMP]

f. Click **OK** to close.

5. Save your work.
6. Now we need to look up the DateKey using the SnapshotDate. We will then repeat this process 2 more times for the SupplierKey and ProductKey.
   a. Drag and drop the **Lookup**  tool onto the design surface.
   b. Connect the output from **DC - Generate SnapshotDate from Day and Initial Date** to the input of **Lookup**.

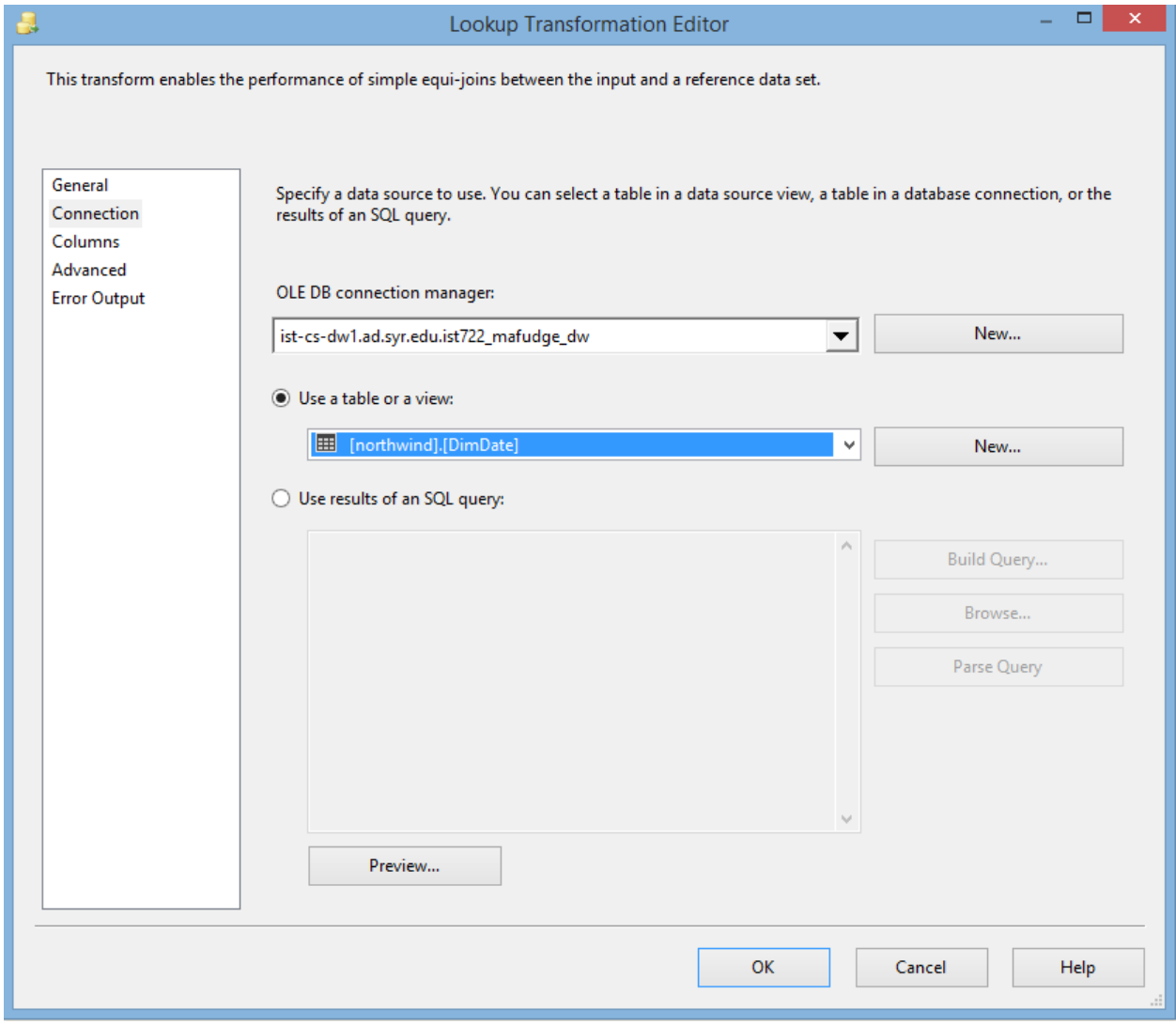
c. Rename the Lookup to **LUP - Lookup DateKey from SnapshotDate.** Here's what you should have at this point:



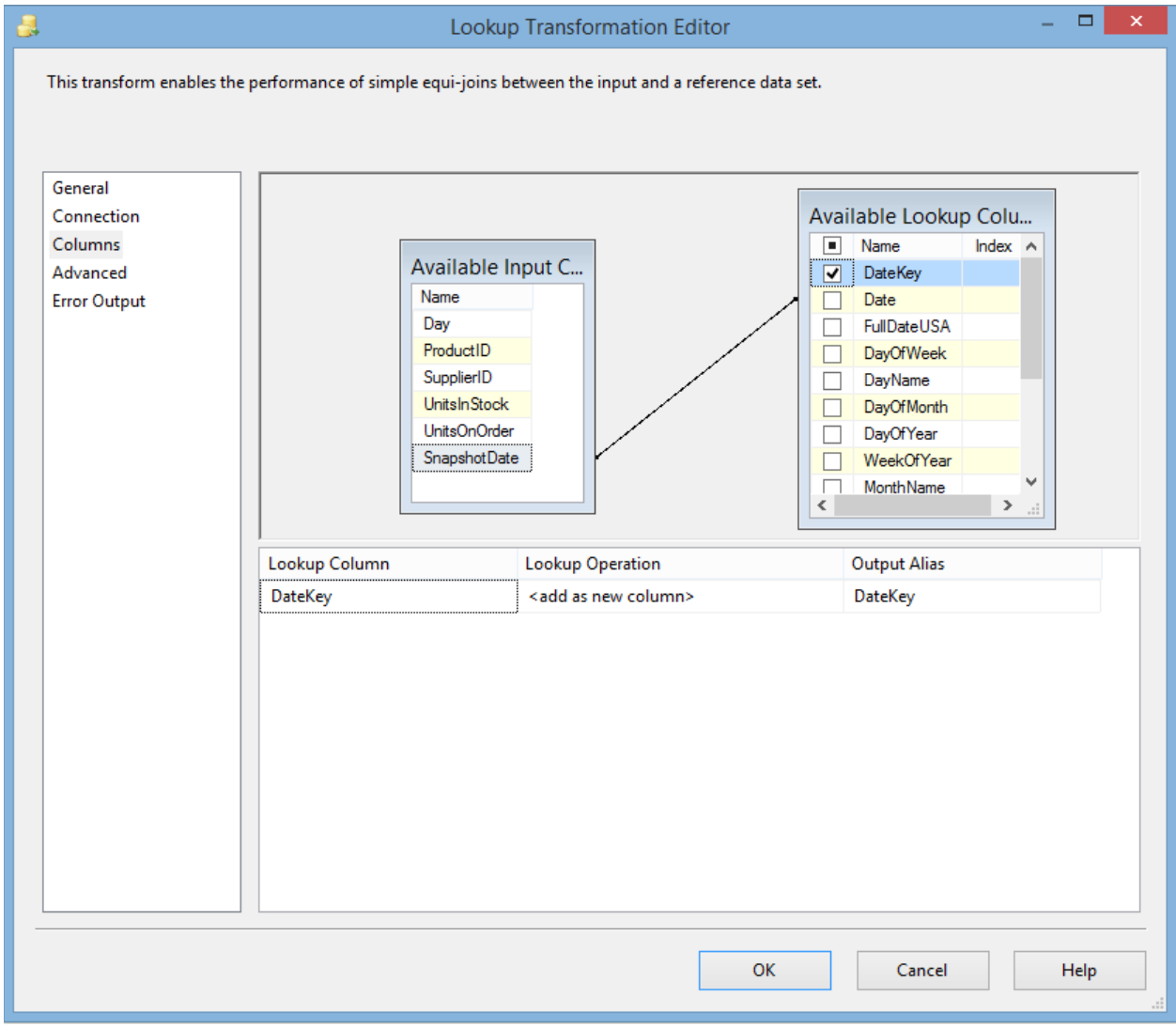
d. Double click on **LUP - Lookup DateKey from SnapshotDate** to configure it. This displays the **Lookup Transformation Editor**.

e. Click on **Connection** section left hand side.
Choose **ist722_*yournetid*_dw** as the connection. And use **northwind.DimDate** as the

table or view.

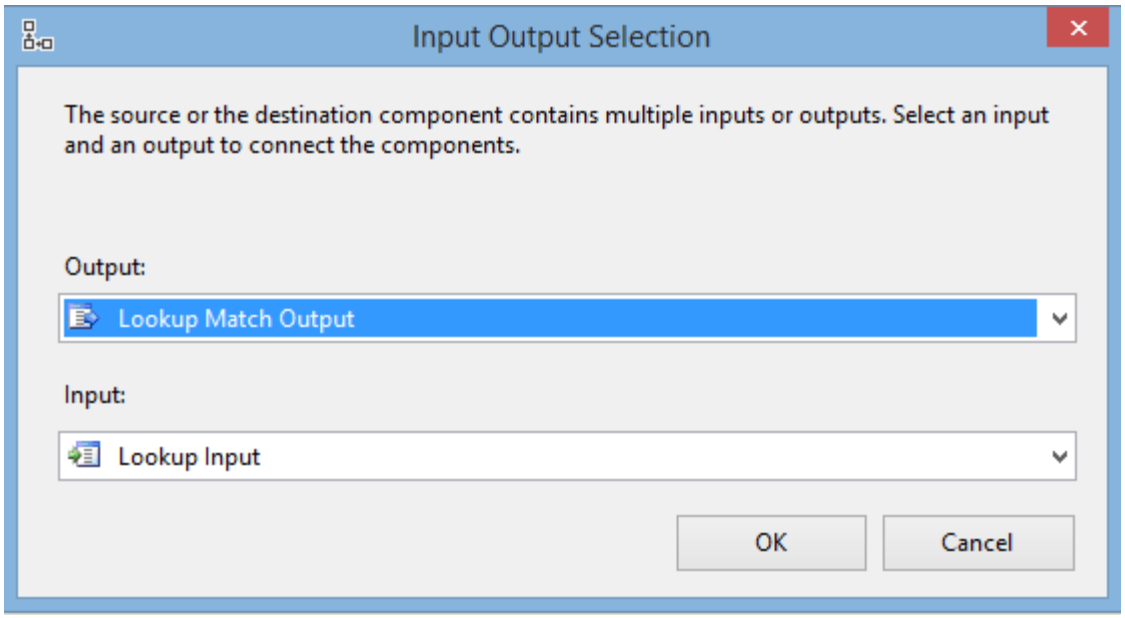


f. Click on the **Columns** section on the left-hand side. Join the two tables on their common business key **SnapshotDate ← → Date**. Then check **DateKey** to add it to the output.



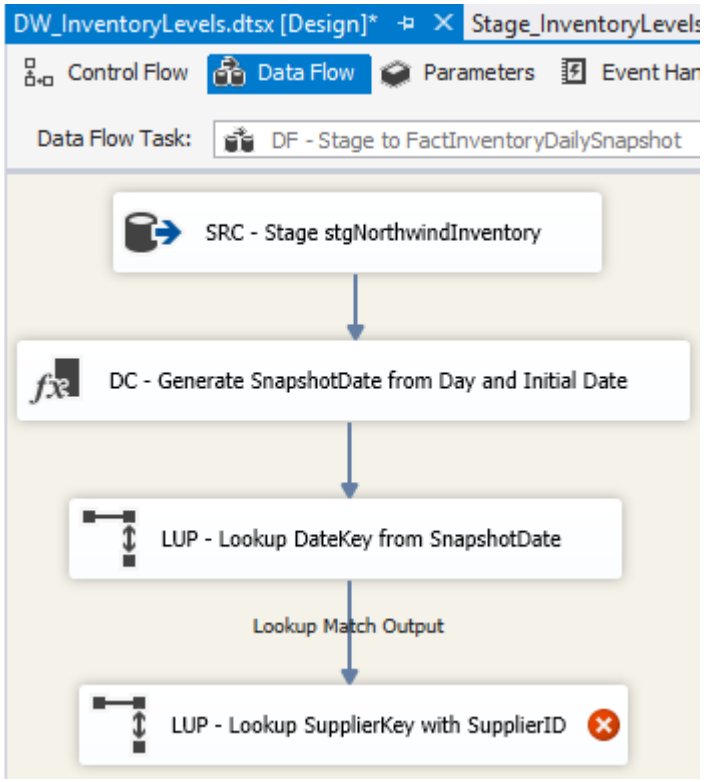
Click **OK** to close the dialog.

7. Let's repeat this process for SupplierKey :
    a. Drag and drop the **Lookup** tool onto the design surface.
    b. Connect the output from **LUP - Lookup DateKey from SnapshotDate** to the input of **Lookup**.
    c. Since the lookup tool as **two outputs: *match & no match*.** We need to select the output we would like to match to this input.



Choose **Lookup Match Output** then click **OK**

Rename the Lookup to **LUP - Lookup SupplierKey with SupplierID.** Here's what you should have:

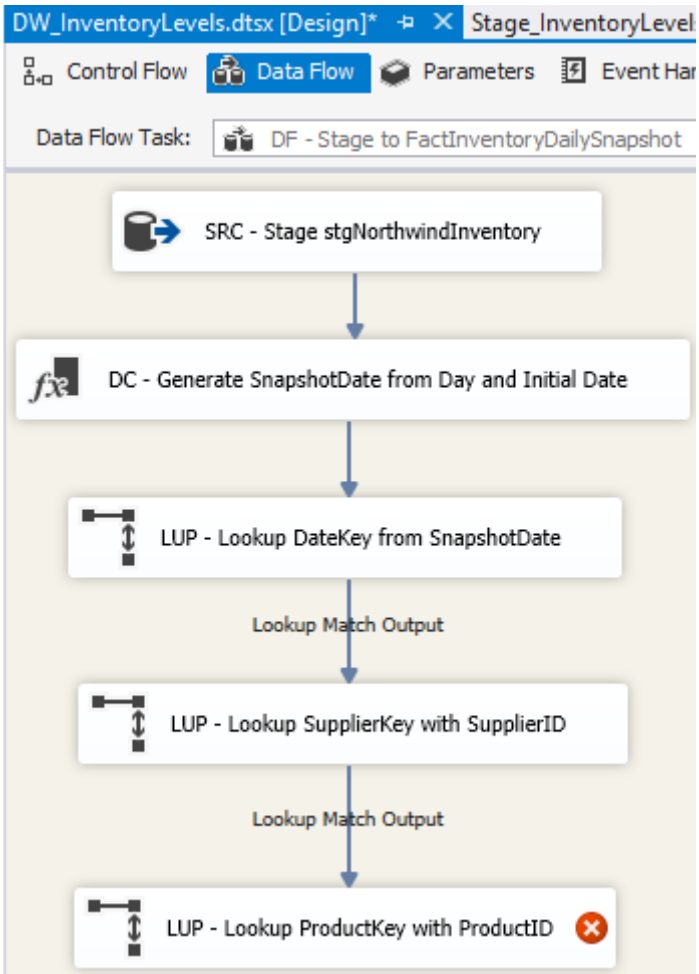


d. Double click on **LUP - Lookup SupplierKey with SupplierID** to configure it. This displays the **Lookup Transformation Editor**.

e. Click on **Connection** section left hand side.
Choose **ist722_*yournetid*_dw** as the connection. And use **northwind.DimSupplier** as the table or view.

f. Click on the **Columns** section on the left-hand side. Join the two tables on their common business key **SupplierID ← → SupplierID**. Then check **SupplierKey** to add it to the output.
g. Click **OK** to close the dialog.

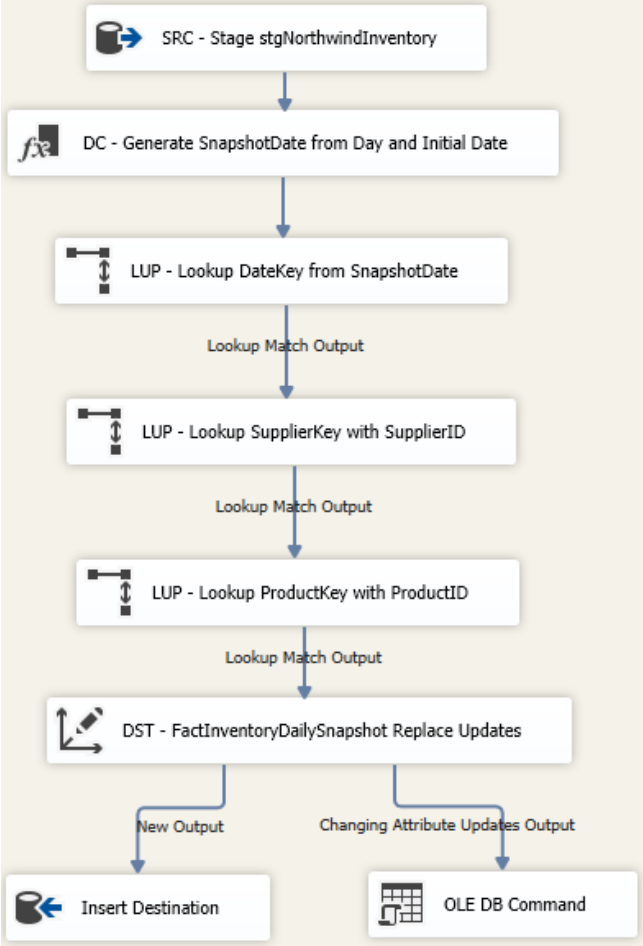
8. Let's repeat this process one last time for ProductKey :
   a. Drag and drop the **Lookup** tool onto the design surface.
   b. Connect the output from **LUP - Lookup SupplierKey with SupplierID** to the input of **Lookup**.
   c. For the **Input Output Selection** choose an Output of **Lookup Match Output** then click **OK**
   d. Rename the Lookup to **LUP - Lookup ProductKey with ProductID.** Here's what you should have:



   e. Double click on **LUP - Lookup ProductKey with ProductID** to configure it. This displays the **Lookup Transformation Editor**.
   f. Click on **Connection** section left hand side.
      Choose **ist722_*yournetid*_dw** as the connection. And use **northwind.DimProduct** as the table or view.
   g. Click on the **Columns** section on the left-hand side. Join the two tables on their common business key **ProductID ← → ProductID**. Then check **ProductKey** to add it to the output.

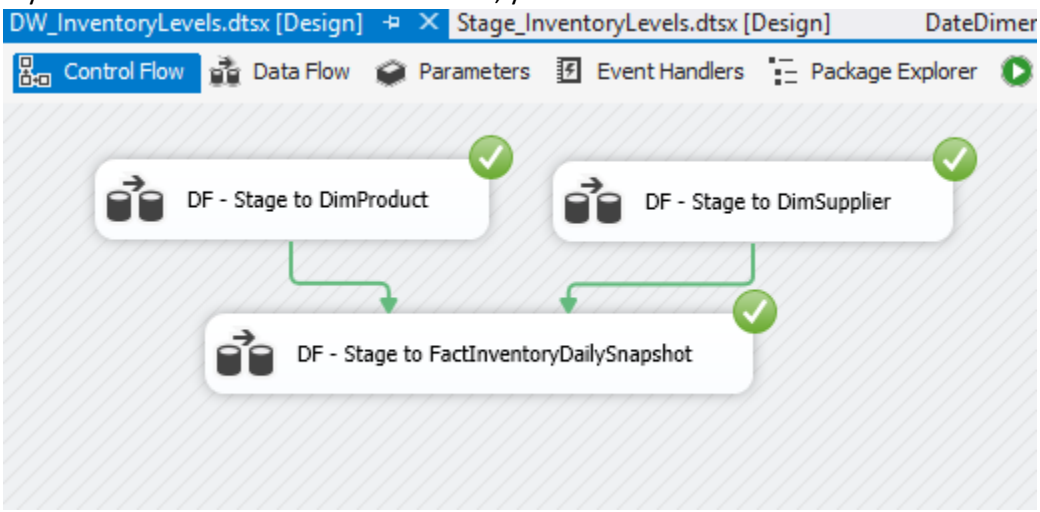
   h. Click **OK** to close the dialog.
9. Now would be a really good time to save your work!
10. Now that the surrogate key pipeline is complete it's time to add the facts to the fact table. Believe it or not we will use a Type 1 SCD to complete this. That way we will not re-add the same fact and if any of our facts change the row will be updated.
    a. Drag and drop the **Slowly Changing Dimension** destination on the design surface.
    b. Rename it → **DST - FactInventoryDailySnapshot Replace Updates**
    c. Connect the output of **LUP - Lookup ProductKey with ProductID** to the input of **DST - FactInventoryDailySnapshot Replace Updates** again choose **Lookup Match Output.**
11. Double click on **DST - FactInventoryDailySnapshot Replace Updates** to configure it.
    a. Click **Next >** past the first page of the wizard.
    b. Use **ist722_*yournetid*_dw** as the connection.
    c. Select **northwind.FactDailySnapshot** as the dimension.
    d. Set all of the keys as the business key. (**DateKey, ProductKey** and **SupplierKey**) This is common in fact tables as we only want to update the fact to correct an error.
    e. Click **Next >**
    f. Set **UnitsInStock** and **UnitsOnOrder** as **changing attributes**.
    g. Click **Next >**
    h. **Clear the check box** for change all matching records. It does not apply here.
    i. Click **Next >**
    j. **Turn off** inferred member support, by **clearing the checkbox.**
    k. Click **Next >**
    l. Click **Finish** to generate the Type 2 SCD logic.

12. You're done! Time to save your work.



13. Execute the package. Make sure it works. Troubleshoot it if you have problems.
14. Verify there are **616 rows** in the **northwind.FactInventoryDailySnapshot**.

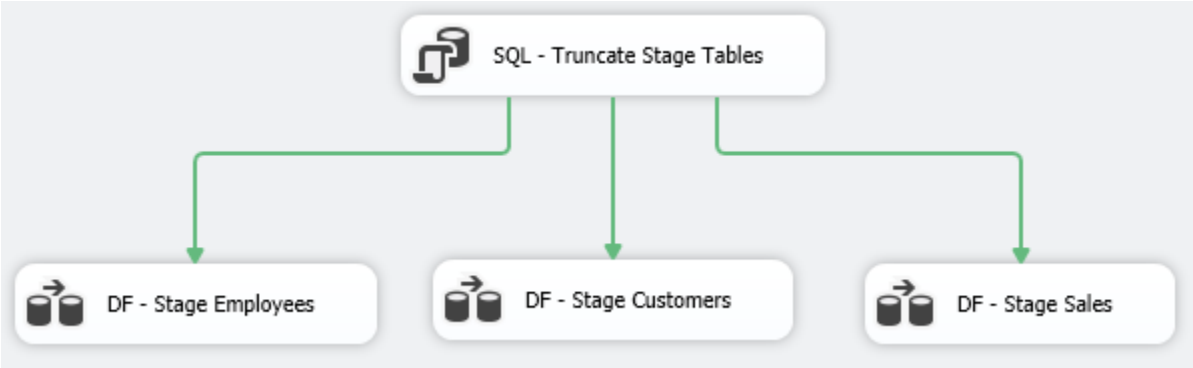
If you switch back to the **control flow** tab, you can see the entire Dimensional Model Loads correctly:

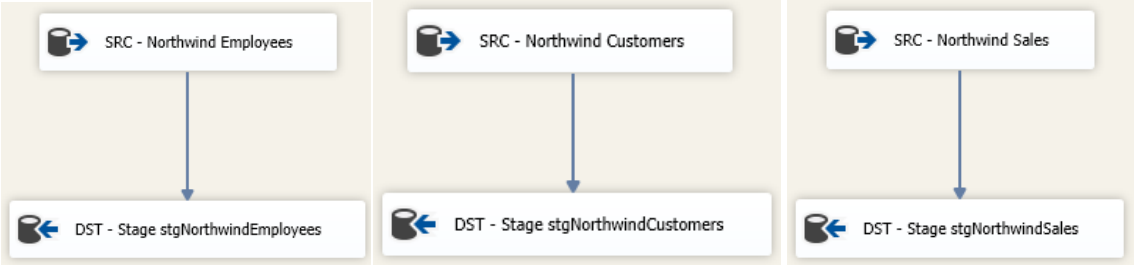


Part 3: On Your Own: ETL Solution for Sales

In this part, try to build out the ETL for Northwind Sales data mart. Here's a high level outline of the process. Remember you use what you learned in part 2 and apply it here.

1. Add a package to your project called **Stage_Sales.dtsx** which will stage source data. Here's the control flow and data flows you'll need to set-up:



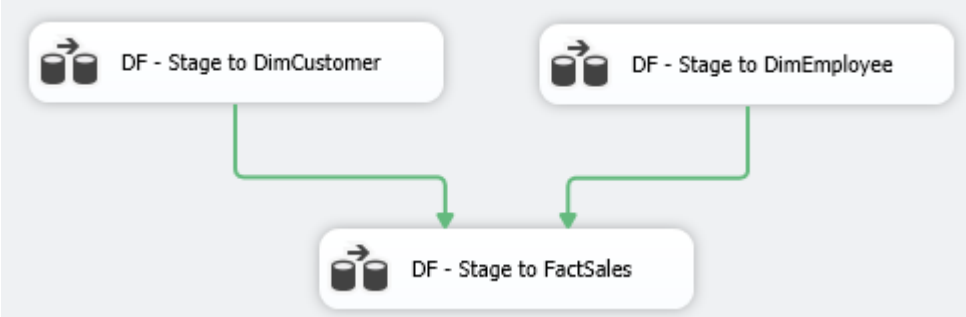


For each data flow, do the following:

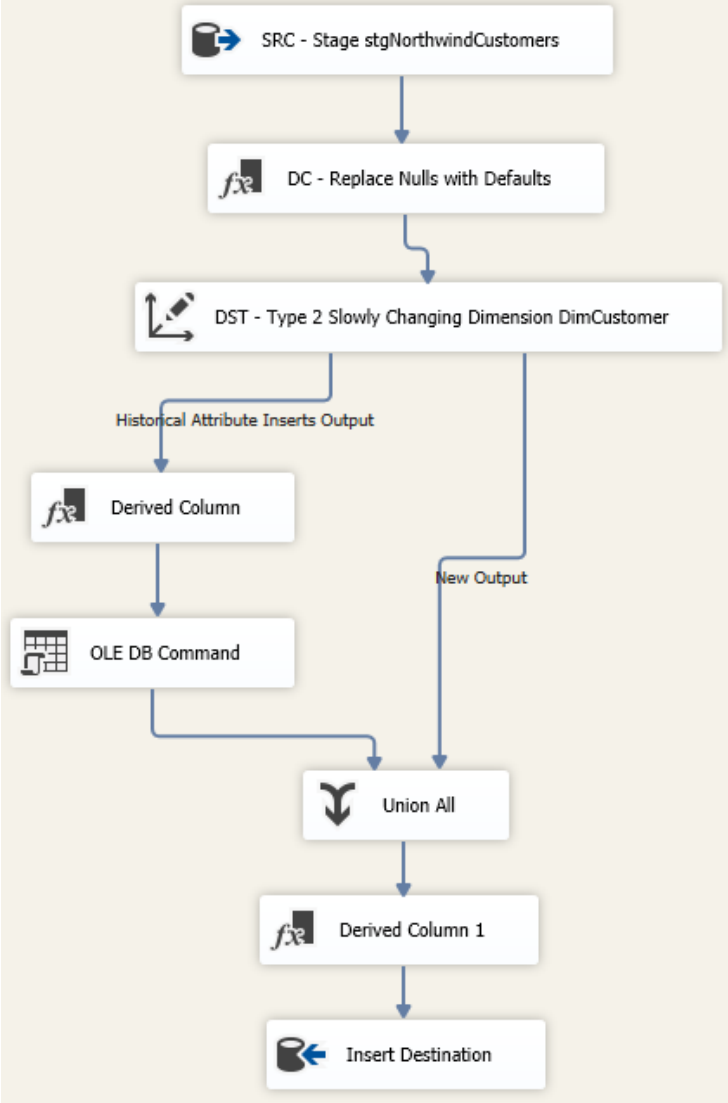
   a. Write an SQL query to source the data.
   b. Create a table in the stage database based on the schema of the source.
   c. Add a truncate table statement to the SQL task
   d. Execute the package to verify the data gets transferred

   NOTE: I encourage you to try and figure out the source SQL queries. If you cannot, refer to **Appendix A – Source SQL Queries**.

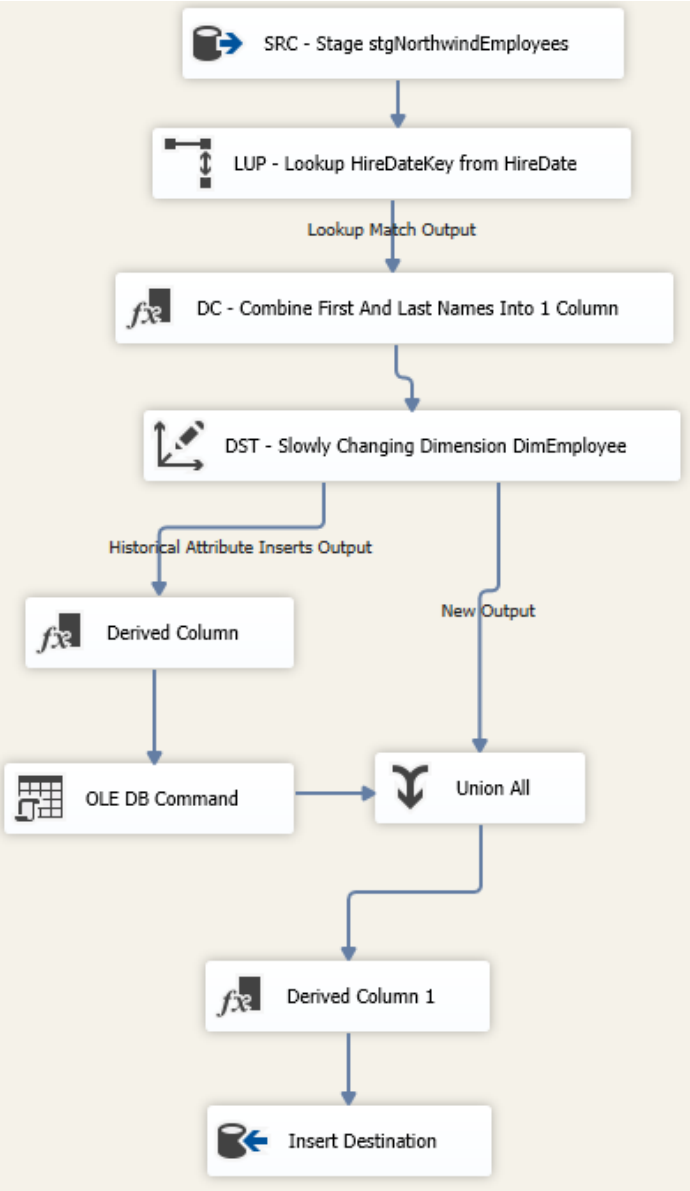
2. Add a package to your project called **DW_Sales.dtsx** which will load the staged data into the data warehouse. Here's the control flow you'll need to set-up:



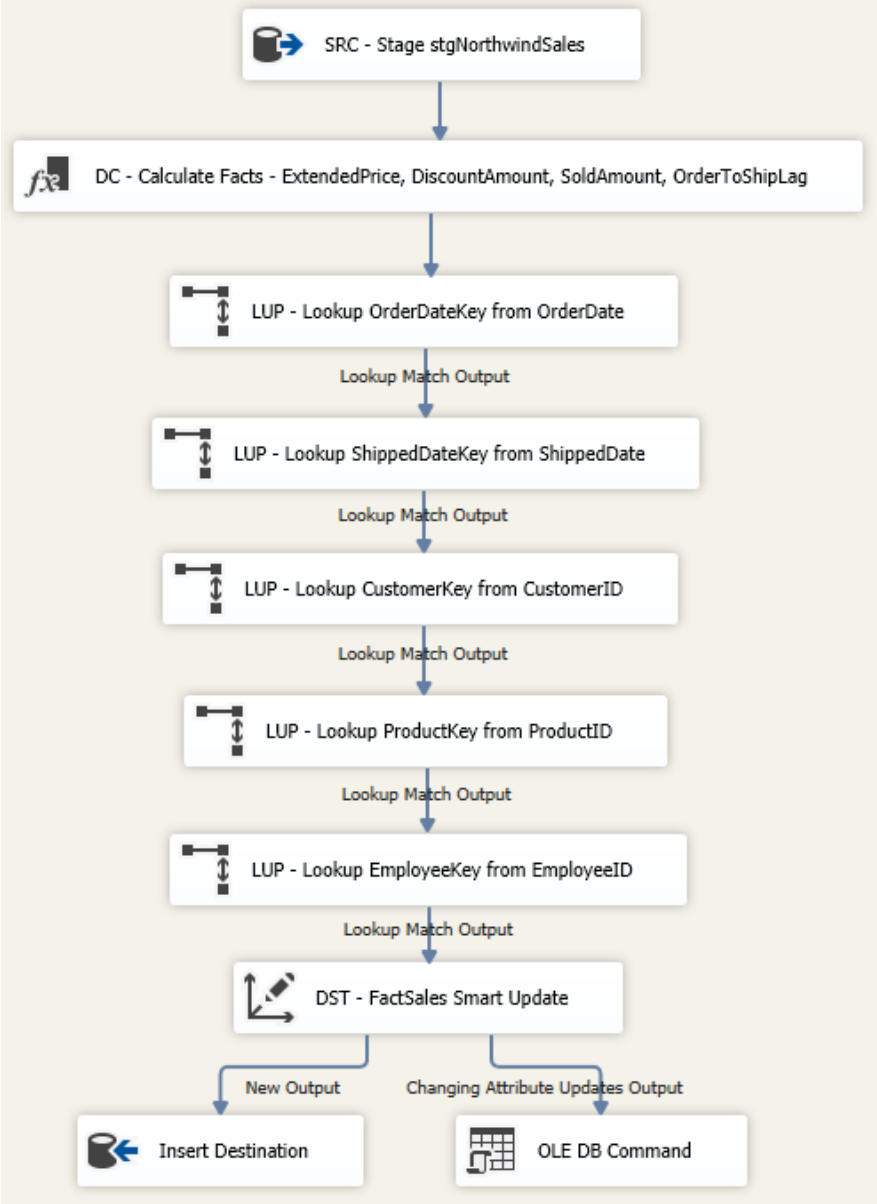
a. Here's the data flow you need to build for **Stage to DimCustomer**:



b. Here is the data flow you need to build for **Stage to DimEmployee**. Notice there is a lookup conversion of HireDate to HireDateKey.

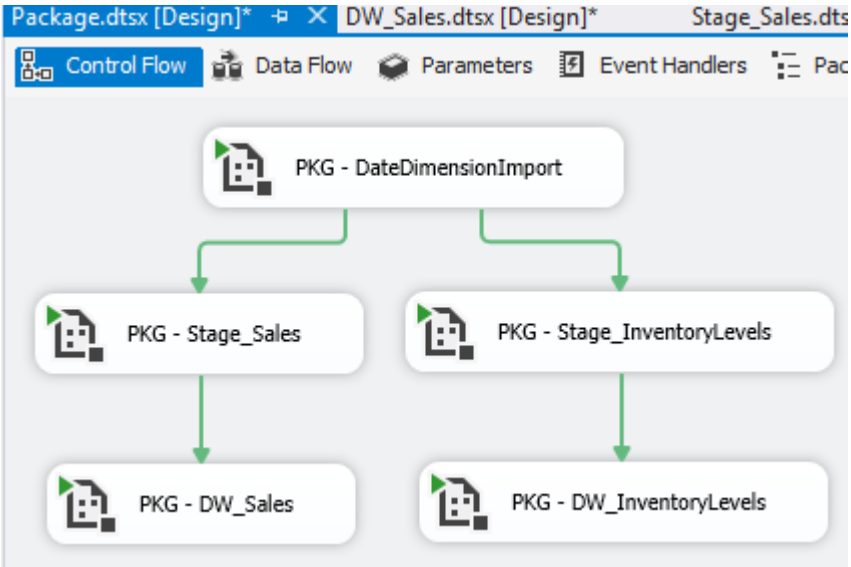


c. Here's the data flow you'll need to build for **Stage to FactSales:**



NOTE: I encourage you to try and figure out the derived columns and other transformations. If you are struggling, refer to **Appendix B – Column Transformations**.

3. In the final step, complete **Package.dtsx** – one package to run them all!!!



Good Luck!

Appendices

Appendix A – Source SQL Queries

Source	SQL
Northwind Customers	SELECT CustomerID, CompanyName, ContactName, ContactTitle, City, Region, PostalCode, Country FROM Customers
Northwind Employees	SELECT Employees.EmployeeID, Employees.LastName, Employees.FirstName, Employees.Title, Employees_1.HireDate, Employees_1.EmployeeID AS SupervisorID, Employees_1.LastName AS SupervisorLastName, Employees_1.FirstName AS SupervisorFirstName, Employees_1.Title AS SupervisorTitle FROM Employees left JOIN Employees AS Employees_1 ON Employees.ReportsTo = Employees_1.EmployeeID
Northwind Sales	SELECT [Order Details].OrderID, [Order Details].ProductID, Orders.CustomerID, Orders.EmployeeID, Orders.OrderDate, Orders.ShippedDate, [Order Details].UnitPrice, [Order Details].Quantity, [Order Details].Discount FROM [Order Details] INNER JOIN Orders ON [Order Details].OrderID = Orders.OrderID

Appendix B – Column Transformations

DC-Replace Nulls with Defaults

Derived Column Name	Derived Column	Expression	Data Type	Length
RegionNA	<add as new column>	LEFT(REPLACENULL(Region,"N/A"),15)	Unicode string [DT_WSTR]	15
PostalCodeUnknown	<add as new column>	LEFT(REPLACENULL(Region,"Unknown"),10)	Unicode string [DT_WSTR]	10

DC - Combine First And Last Names Into 1 Column

Derived Column Name	Derived Column	Expression	Data Type	Length	Precis
EmployeeName	<add as new column>	FirstName + " " + LastName	Unicode string [DT_WSTR]	31	
SupervisorName	<add as new column>	SupervisorFirstName + " " + SupervisorLastName	Unicode string [DT_WSTR]	31	

DC - Calculate Facts - ExtendedPrice, DiscountAmount, SoldAmount, OrderToShipLag

Derived Column Name	Derived Column	Expression	Data Type	Leng
ExtendedPriceAmount	<add as new column>	(DT_NUMERIC,18,4)(Quantity * UnitPrice)	numeric [DT_NUMERIC]	
DiscountAmount	<add as new column>	(DT_NUMERIC,18,4)(Quantity * UnitPrice * Discount)	numeric [DT_NUMERIC]	
SoldAmount	<add as new column>	(DT_NUMERIC,18,4)(Quantity * UnitPrice * (1 - Discount))	numeric [DT_NUMERIC]	
OrderToShippedLagInDays	<add as new column>	(DT_I2)(DATEDIFF("d",OrderDate,ShippedDate))	two-byte signed integer [DT_I2]	