

Assignment 06:

ETL Part 1 Data Extraction

Part 1: Overview

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

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

Goals

Specifically the goals of this assignment are to:

- Introduce you to ETL tooling, specifically SQL Server integration services
- Learn how to extract and stage data from data sources using tooling
- Learn how to troubleshoot problems with the ETL process

Effort

This assignment is best done individually. Please work alone.

Technical Requirements

To complete this assignment you will need the following:

- Access to the course **ist-cs-dw1.ad.syr.edu** SQL Server, and specifically the Northwind Traders database and your **ist722_yournetid_stage** database. You should connect to this server before starting the assignment.
- Access to the **SQL Server data tools** that are part of the **Visual Studio 2017** (or higher) developer environment. This is the tooling we will use to create the ETL Packages.
- Files that will help you create the DDS enterprise bus architecture and help populate the inventory fact table (you need *either* the design workbook *or* the SQL script but not both):
 - **UPDATED_Northwind-Detailed-Dimensional-Modeling-Workbook-KimballU.xlsm** is a completed design workbook that may be used to generate the SQL script to create the DDS bus architecture. Save the generated SQL as Northwind-ROLAP-Bus-Architecture.sql file. **If you are already given the SQL script file (below), skip this step.**
 - **Northwind-ROLAP-Bus-Architecture.sql** file is the script for creating the DDS enterprise bus architecture of conformed dimensions across two business processes for Northwind: sales reporting and inventory analysis. It will remove unnecessary tables in the schema.

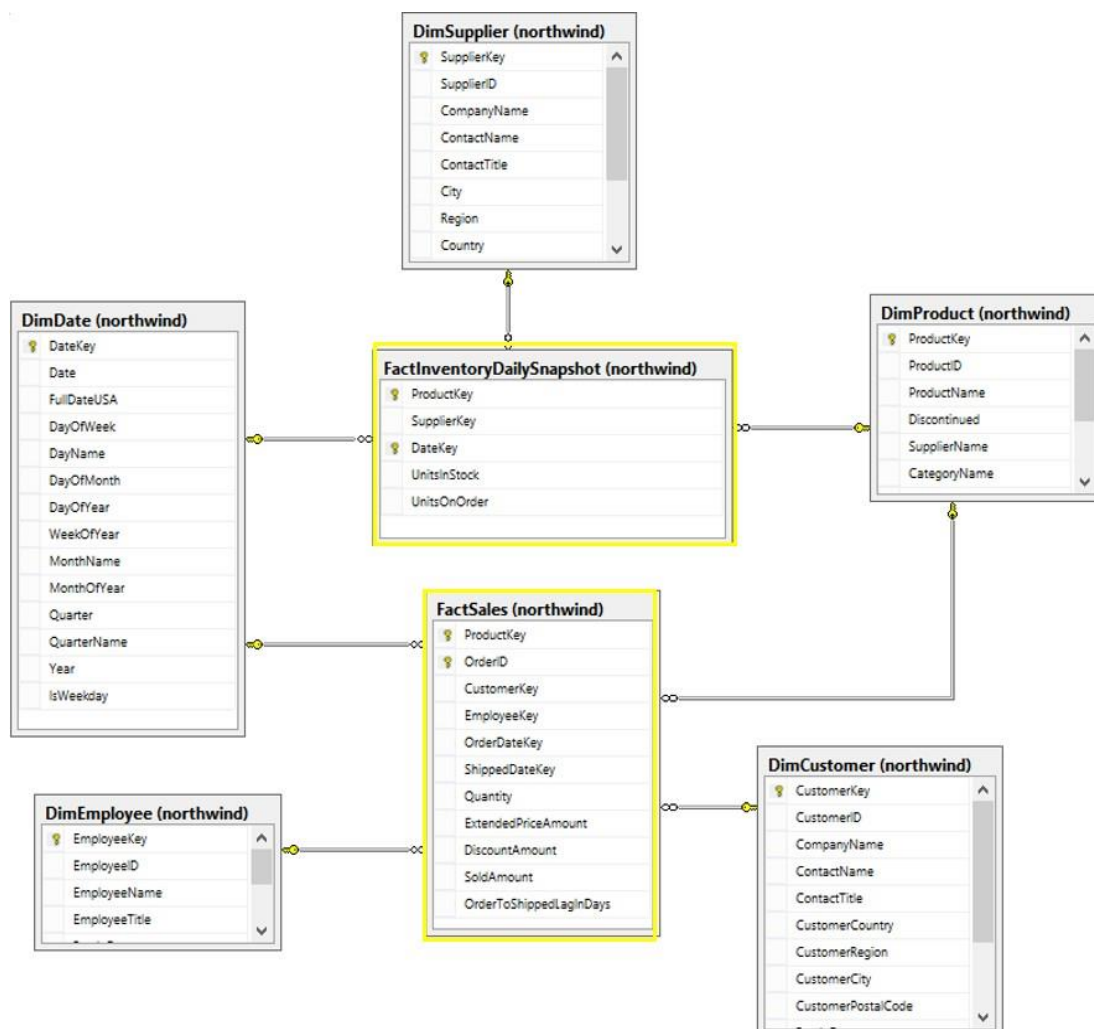
- **NorthwindDailyInventoryLevelsOneWeek.csv** file. This file simulates a weeks' worth of daily snapshots of changes to inventory levels. It is used as a data source for the inventory analysis business process.
- **SSIS-Assignment-Files.zip** file (in the same place you found this assignment). They contain the above two files. Please copy them out of the zip file into some known location (e.g., your home directory or desktop).

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 **Northwind-ROLAP-Bus-Architecture.sql** file you will see SQL code to create the DDS star schemas for both of these business processes. You will also notice we are using a Kimball Enterprise **bus architecture** as we are conforming our dimensions across fact tables.

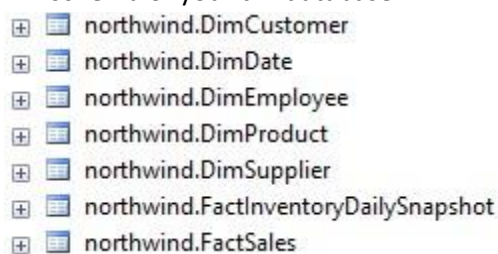


Northwind Enterprise Bus Architecture. Fact tables outlined in yellow.

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 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.

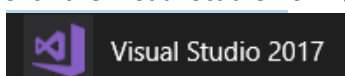
Part 2: Walk-Through

In this part, we'll overview the SSIS tooling, including how to launch the program and get around the basic user interface. We will then create an SSIS package to perform data extraction.

Launch SQL Data Tools

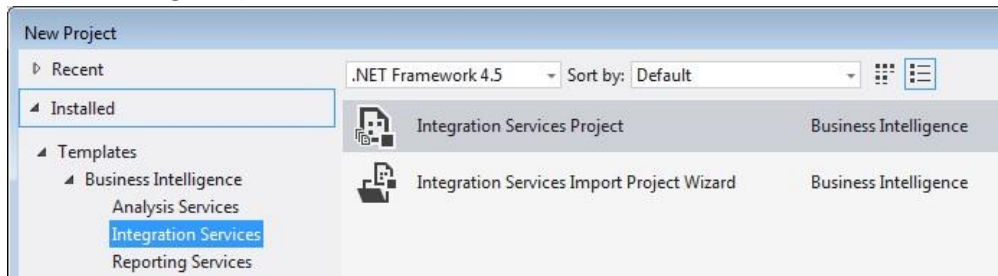
In order build our ETL solution with SSIS, first we will need to run SQL Data Tools. These are part of Visual Studio. **DO THIS:**

1. Click **Windows Start**.
2. Click the **Visual Studio 2017** (or higher) 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 screenshots were 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:

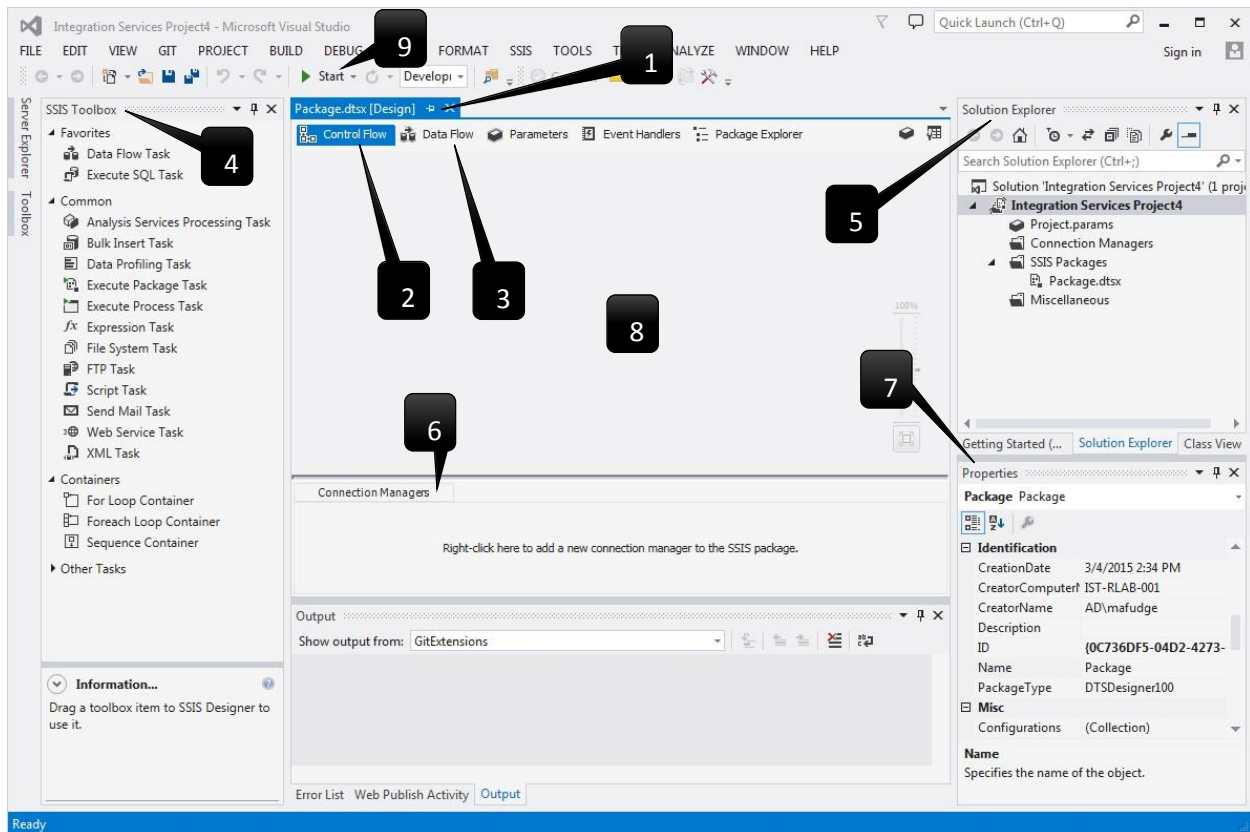
1. From the Menu, select **TOOLS → Options**.
2. From the options dialog select the **Light** color theme.



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 one **Control Flow [2]** with one 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 that should be carried out by the package. SSIS is as a visual programming language, and you can set up 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. Your 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 **Start [9]** in the toolbar to place the package in run mode.

Data Extraction 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, 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 get a nice, gentle introduction to SSIS by creating a package to populate the date dimension, creating the package: **DateDimensionImport.dtsx**
- In Part 2.2, we will create a package called **Stage_InventoryLevels.dtsx** to extract our external world data into our stage database. We will stage the data "as-is" and use the "truncate and load" ETL pattern.

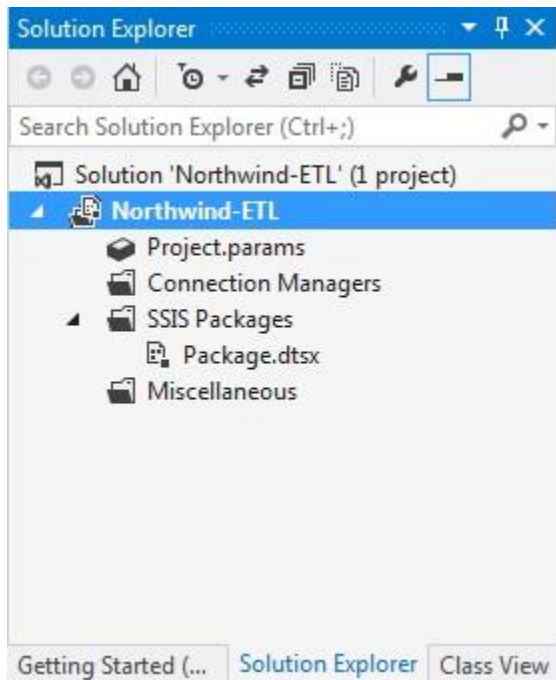
Create Our Solution and Project

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 your solution is saved. You'll need to get it later.

Click **Browse...** to select a different folder.
3. Click **OK** when you're ready.
4. Your Solution Explorer should look like this:

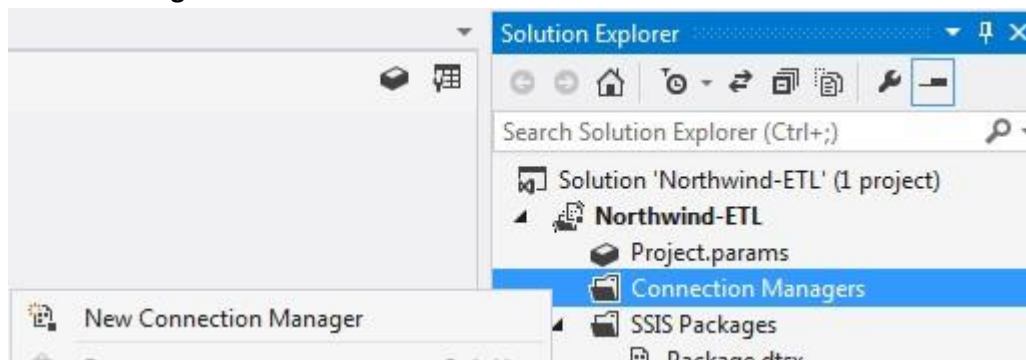


NOTE: if you do not see the Solution Explorer window, press **[CTRL] + [ALT] + [L]** to bring it up.

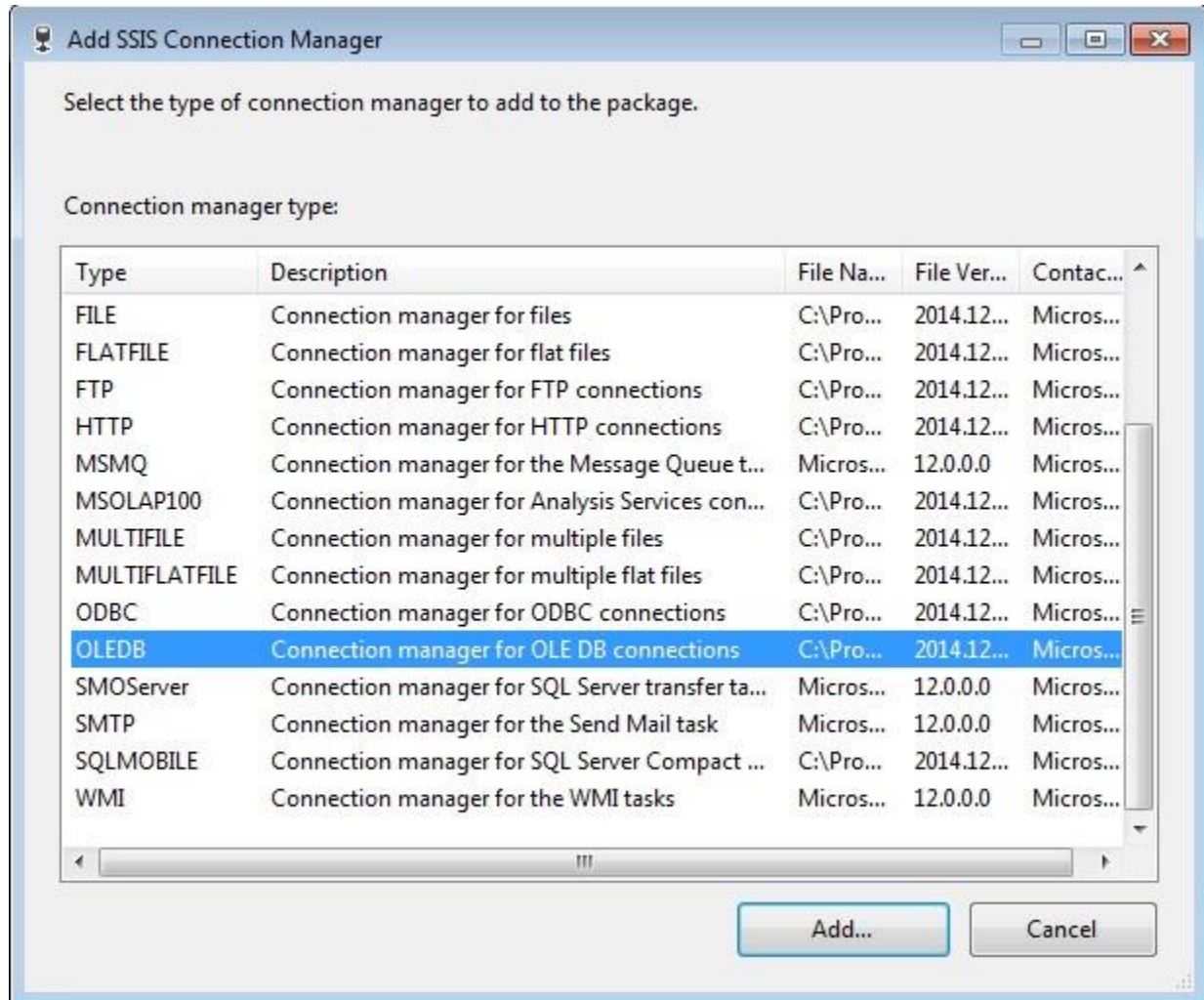
Establish Connections Used by the Project

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 reused by other packages. Let's do this now.

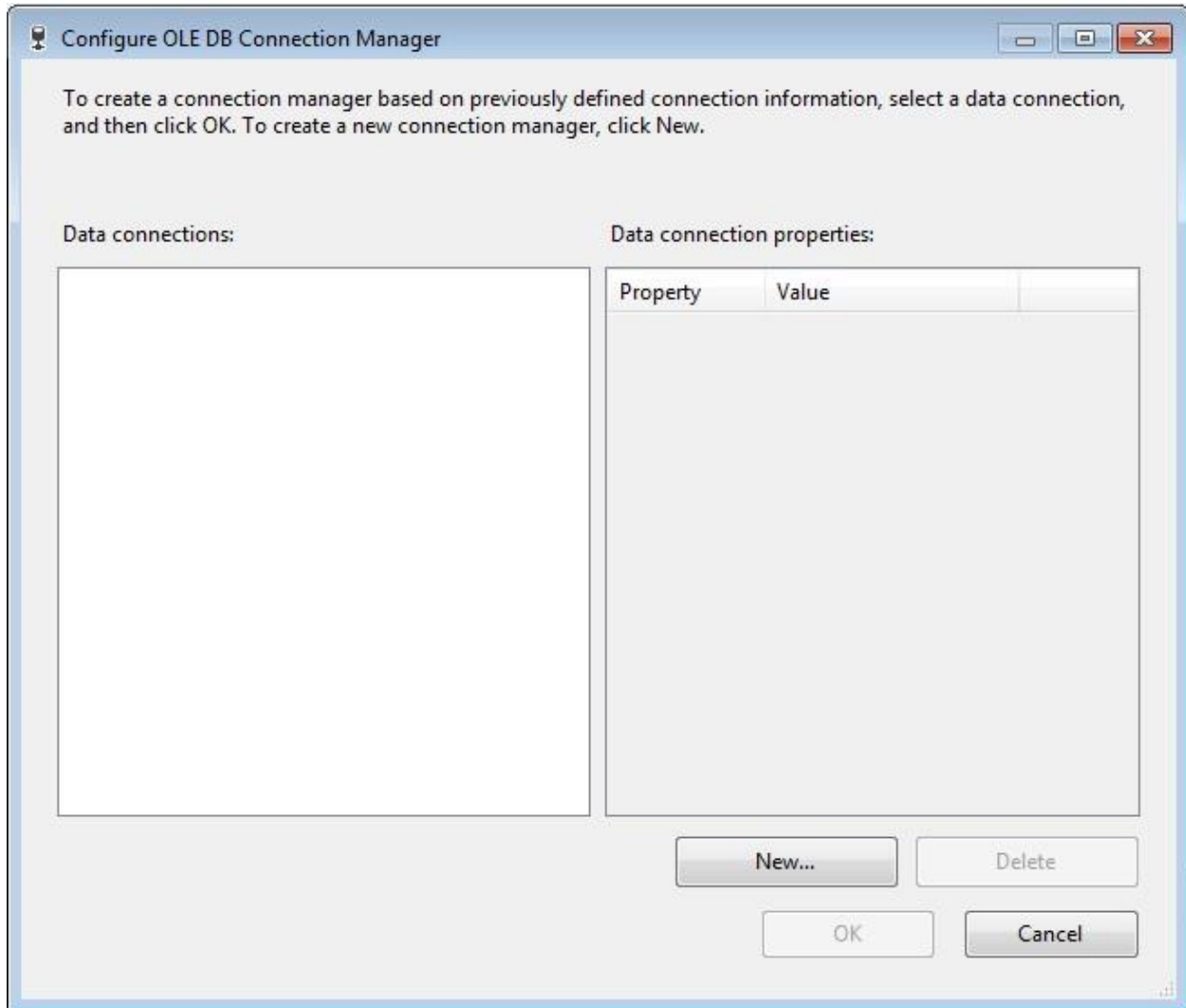
1. First, in Solution Explorer, right-click **Connection Managers** and select **New Connection Manager** from the menu.



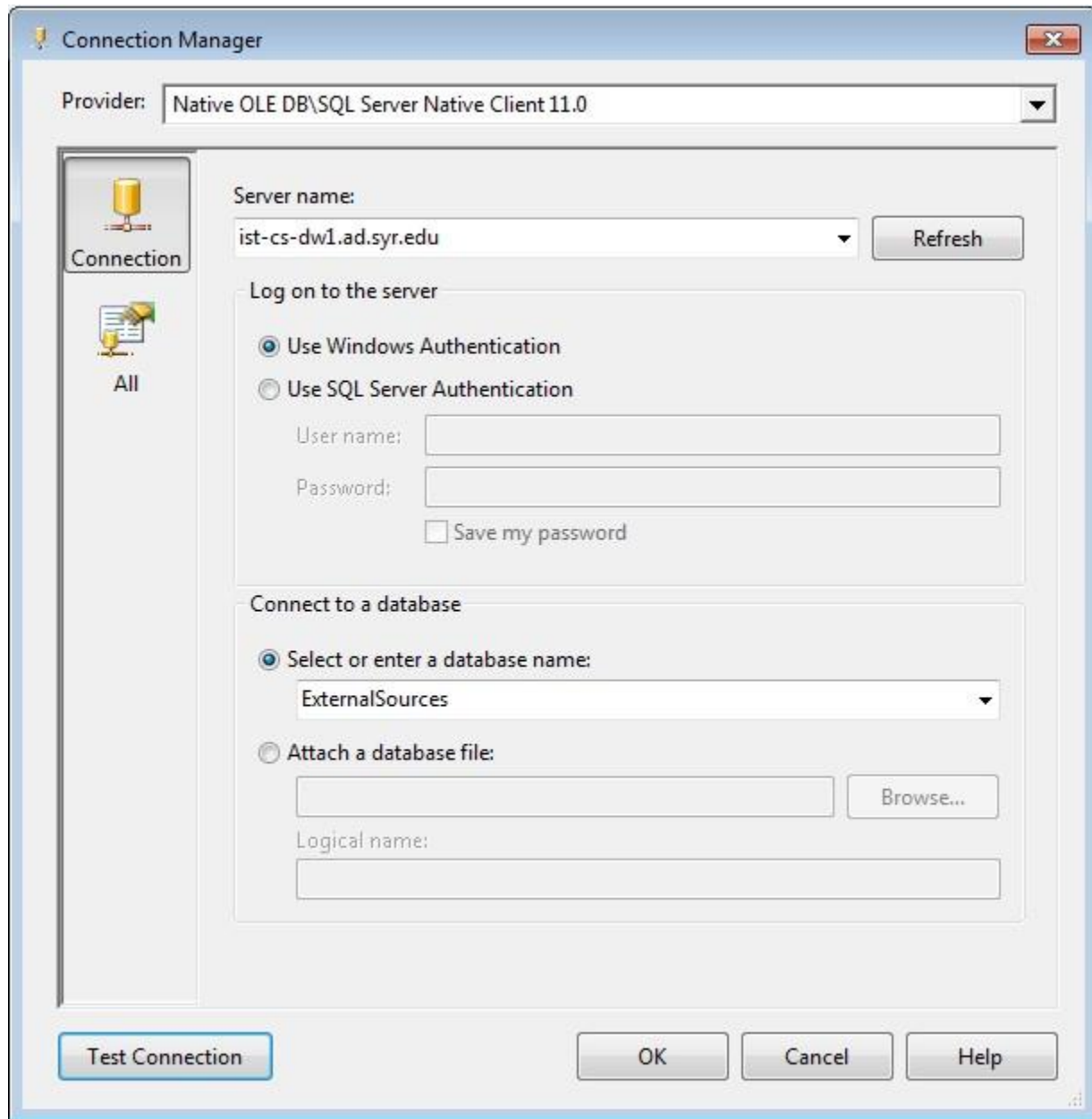
2. This brings up the **Add SSIS Connection Manager** dialog. There are many to choose from, and not all are for data connections. For most database-type connections we will use OLEDB and ODBC as a fallback when there is no OLEDB driver available. Choose **OLEDB** and click **Add...**



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

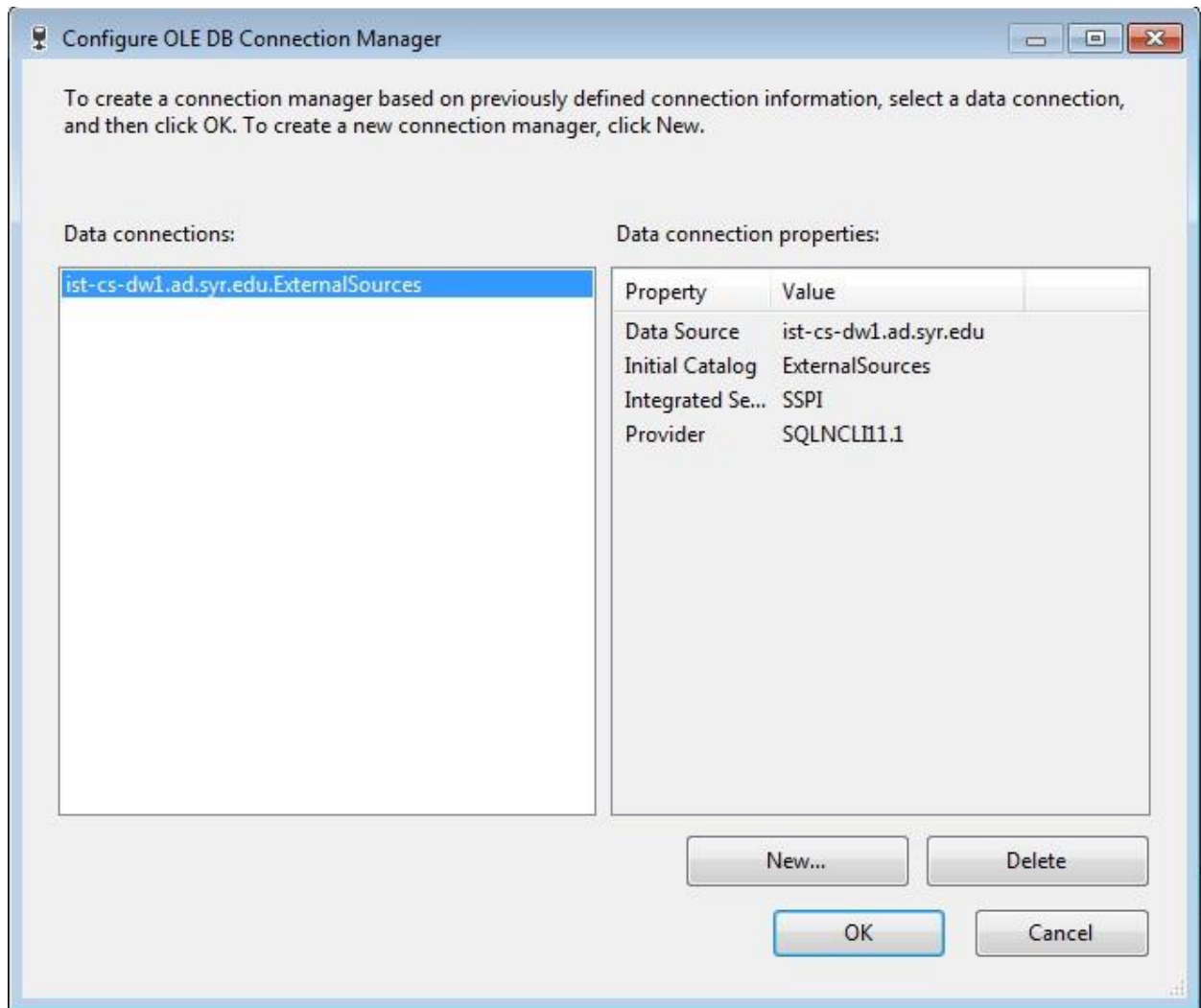


4. You will now see a **Connection Manager** dialog. In here, you need to choose the following options:
Provider: **SQL Server Native Client**
Server name: **ist-cs-dw1.ad.syr.edu**
Log on to the server: **Use Windows Authentication**
Select or enter a database name: **ExternalSources2**

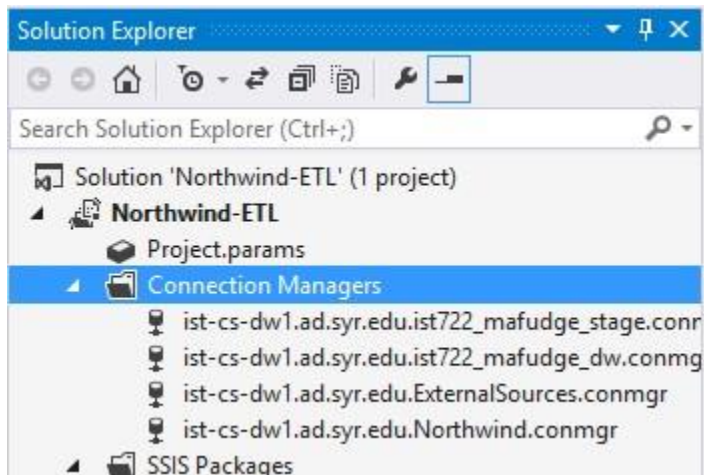


When you've got the information filled in, click **Test Connection** first to verify the connection works. After you get the connection to work, click **OK** to save the connection.

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.ExternalSources2** that we just created:

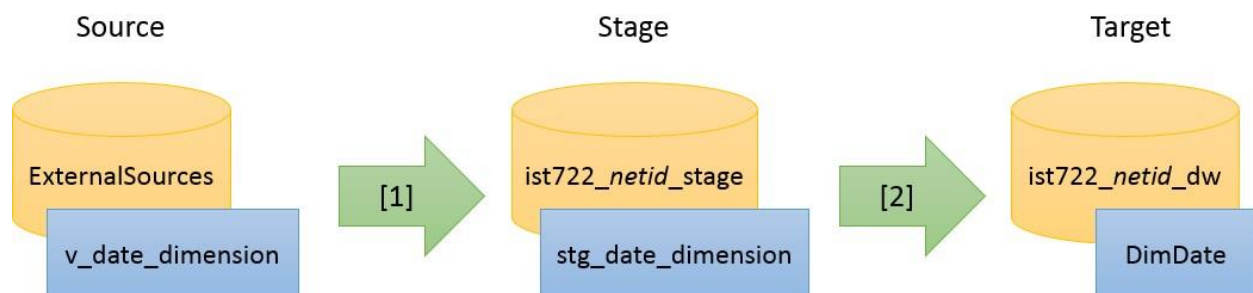


6. Select the ExternalSources connection and click **OK** to add the connection to the project.
7. Repeat this process (steps 1 through 6) three more times for your **ist722_netid_dw** and **ist_netid_stage** databases and for the **Northwind** source database respectively. Here's an example showing 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 predetermined, 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 **date_dimension** in the **ExternalSources2** 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 import the same data more than once.

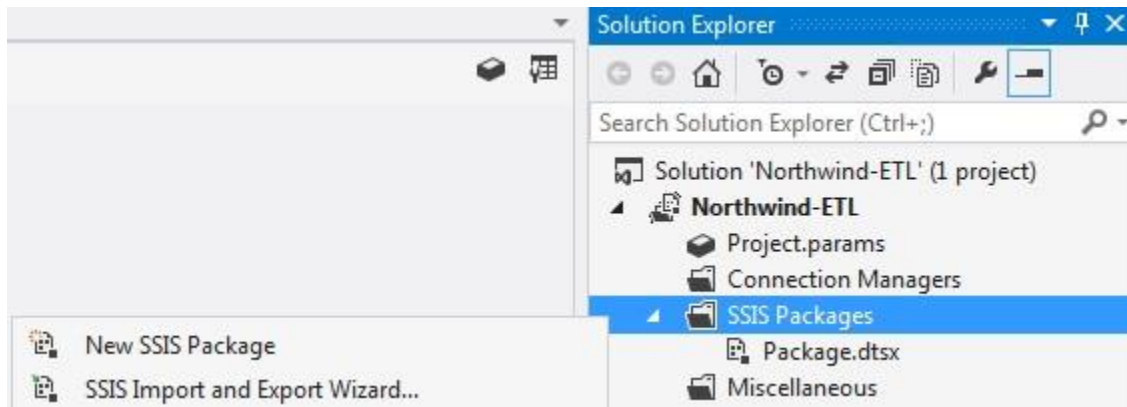
Let's get started!

Step 2.1.1: Create the Package

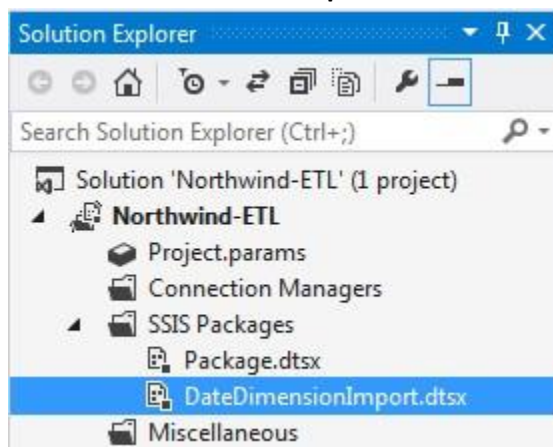
First, let's create a new package, called **DateDimensionImport** to contain our ETL program.

1. In **Solution Explorer**, right-click **SSIS Packages** and select **New SSIS Package**.

NOTE: If you do not see **Solution Explorer**, open it from the **VIEW** menu.



2. Right-click the package named **Package1.dtsx** 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**.

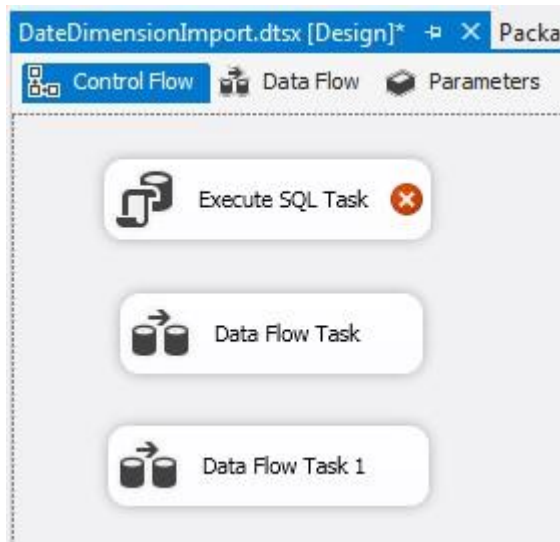


5. Finally, double-click the **DateDimensionImport.dtsx** package to open it up so we can place tasks on the design surface.

Step 2.1.2: Set Up Control Flow

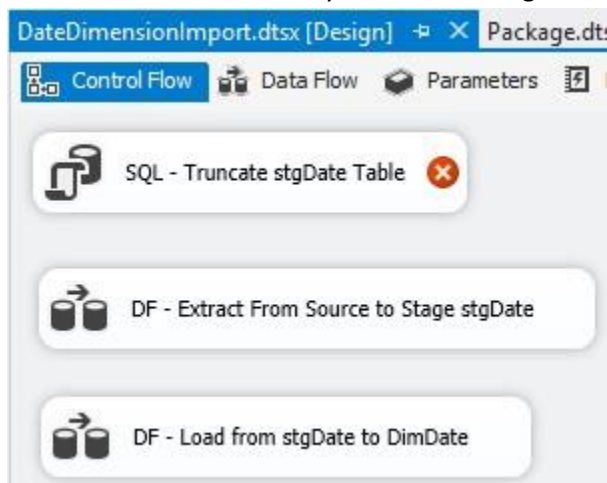
Next, let's set up the control flow for this package.

1. From the SSIS Toolbox drag and drop one **Execute SQL Task** and two **Data Flow Tasks** on to the work surface. Like this:

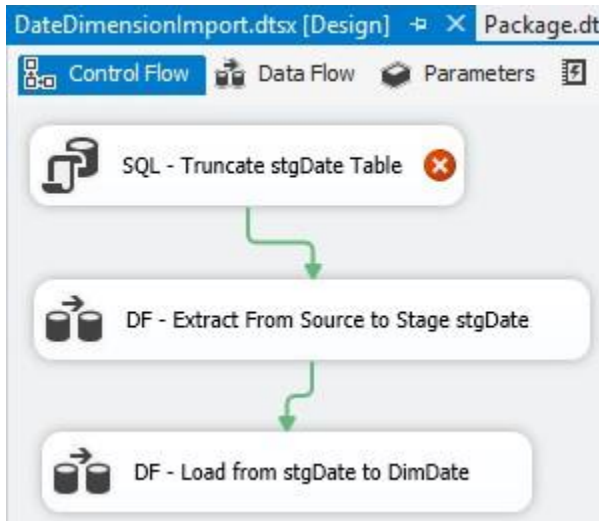


HELP!: Can't find the SSIS Toolbox? Try **Menu → SSIS → SSIS Toolbox**.

2. Next rename each task. To do this click 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:



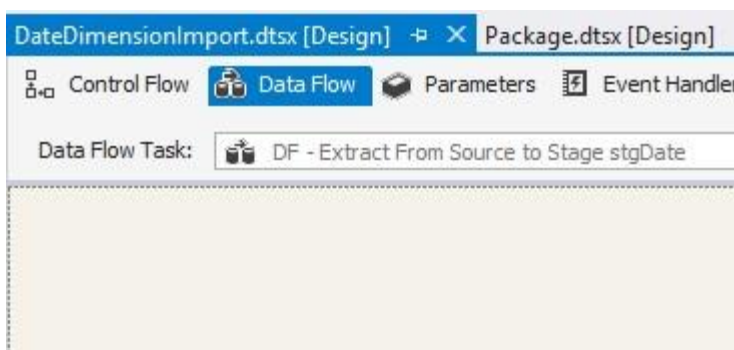
3. Next, connect the tasks so that they run sequentially. Click the **SQL - Truncate stgDate Table** task and a green arrow will appear. Drag the arrow and drop it on the **DF - Extract From Source To Stage stgDate** task. Repeat this process to connect the last two tasks so they are configured to execute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



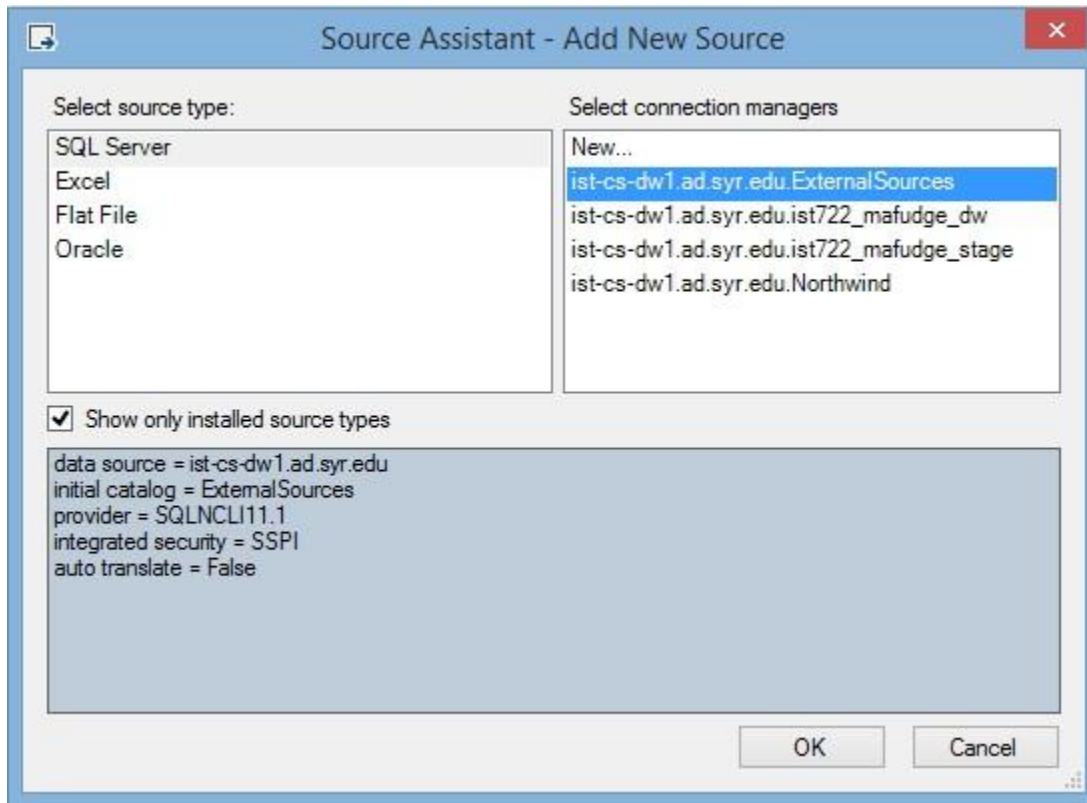
Step 2.1.3: Configure Source to Stage Data Flow

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:

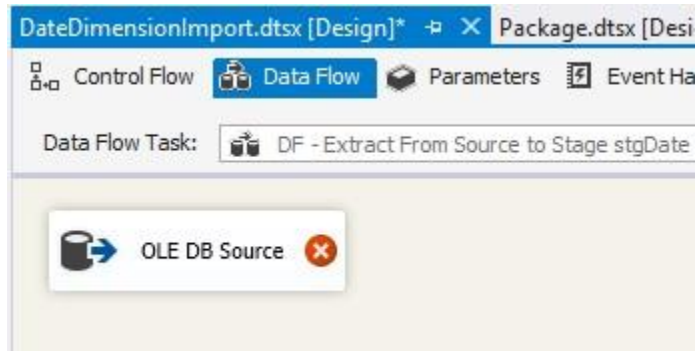


2. From the **SSIS Toolbox**, drag and drop the **Source Assistant** to the design surface. This will open a dialog. Select source type: **SQL Server** and the **ExternalSources2** connection manager.

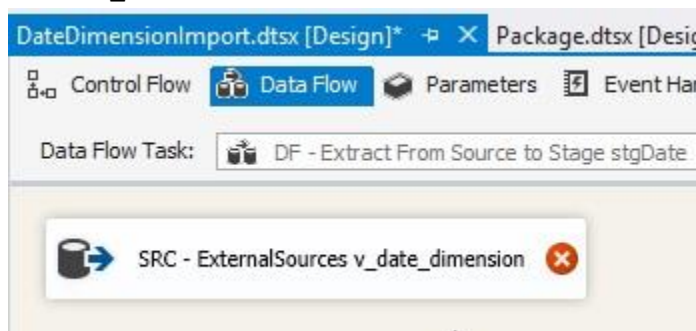


Click **OK** when you're ready.

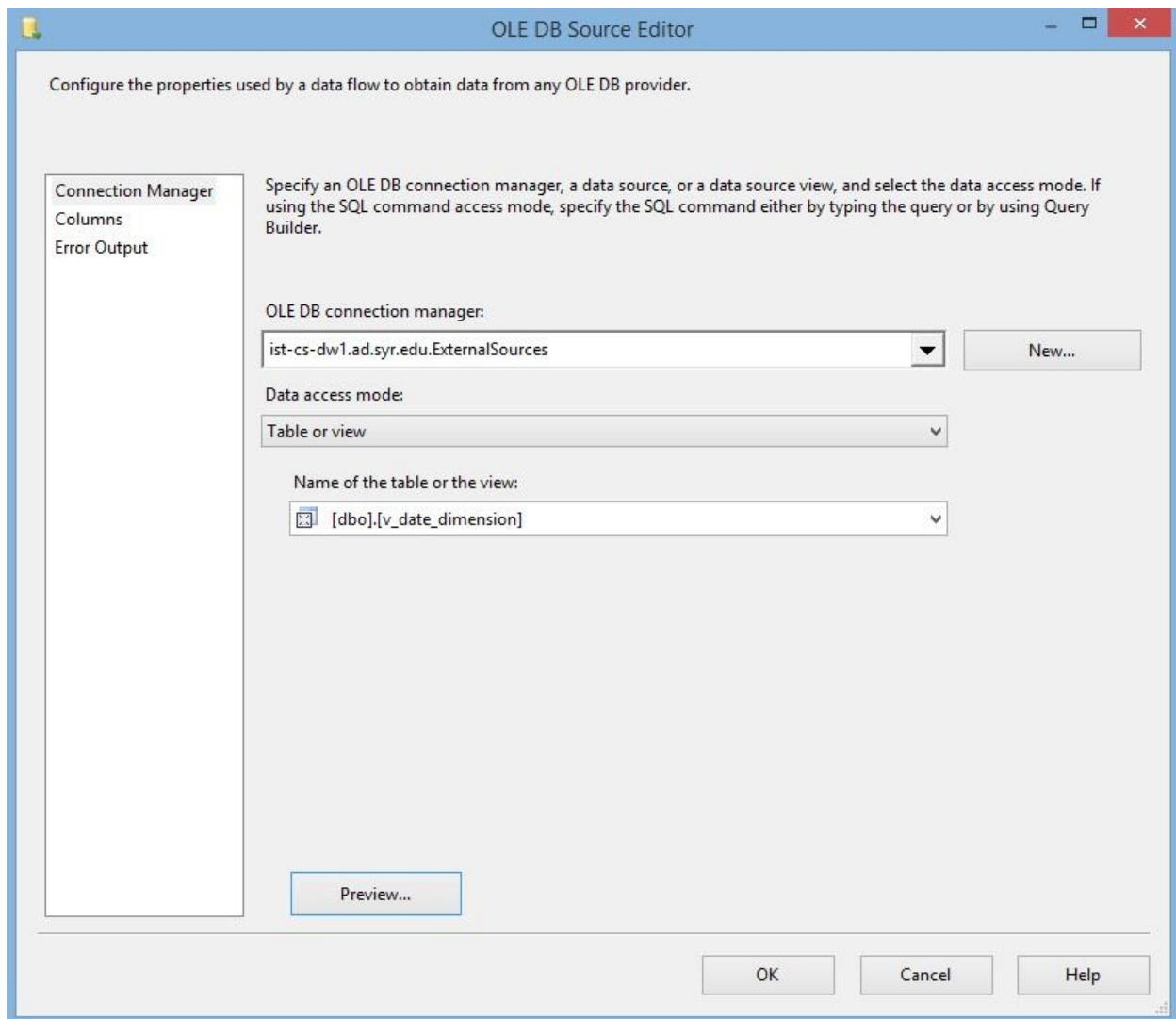
3. You should now have an unnamed and unconfigured source on your design surface:



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

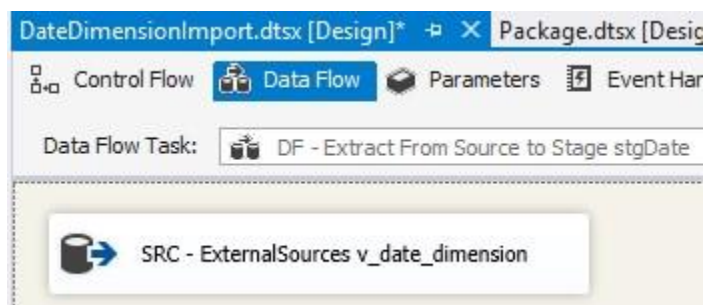


- Double-click the source to configure it. This brings up the **OLE DB Source Editor**. From the dropdown for **Name of the table or the view** select **[dbo].[date_dimension]** as the source. You can click **Preview...** to view the source data.




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

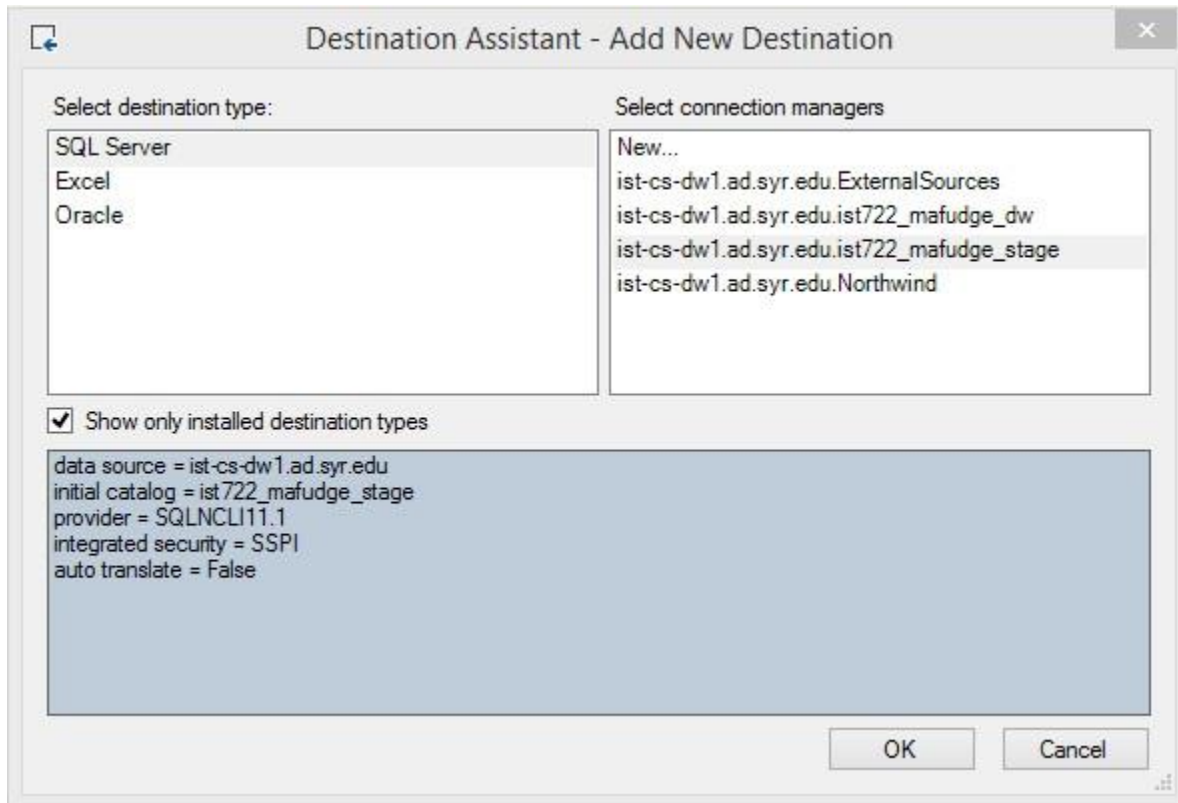
- Your source is now configured. You can see this because there is no longer a red [x] error icon next to the source:



- Next, we need to configure the destination. From the **SSIS Toolbox**, drag and drop the **Destination**

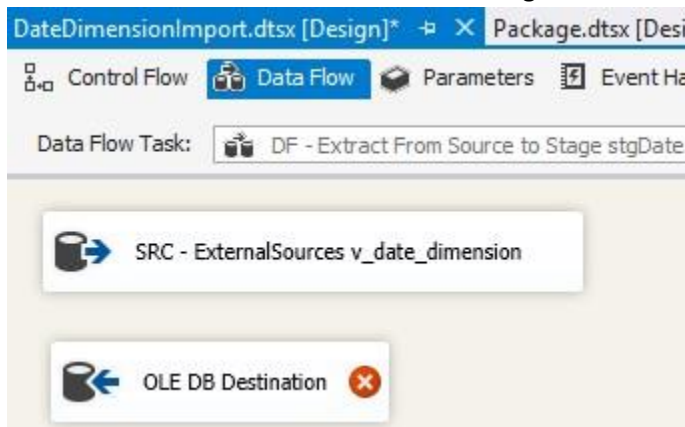
Assistant  Destination Assistant to the design surface. This will open a dialog. Select source type:

SQL Server and the **ist722_yournetid_stage** connection manager.

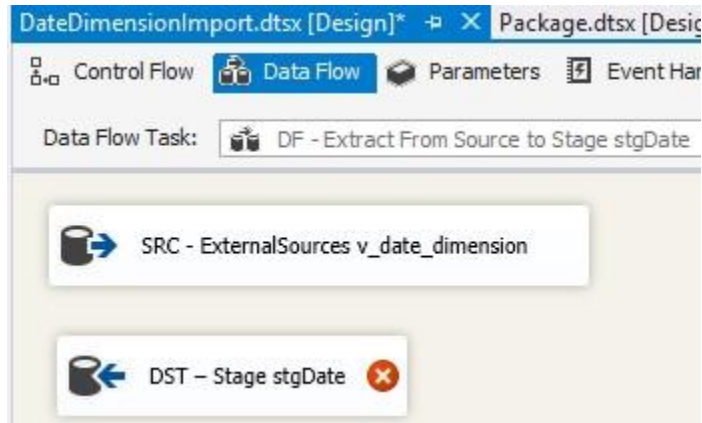


Click **OK** when you're ready.

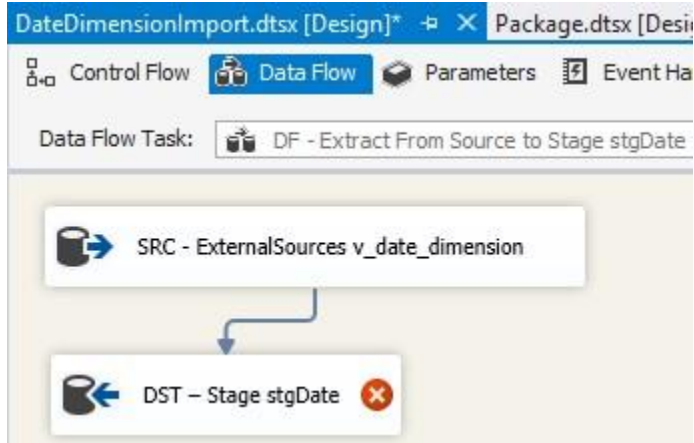
8. We now have an unnamed and unconfigured *destination* on your design surface:



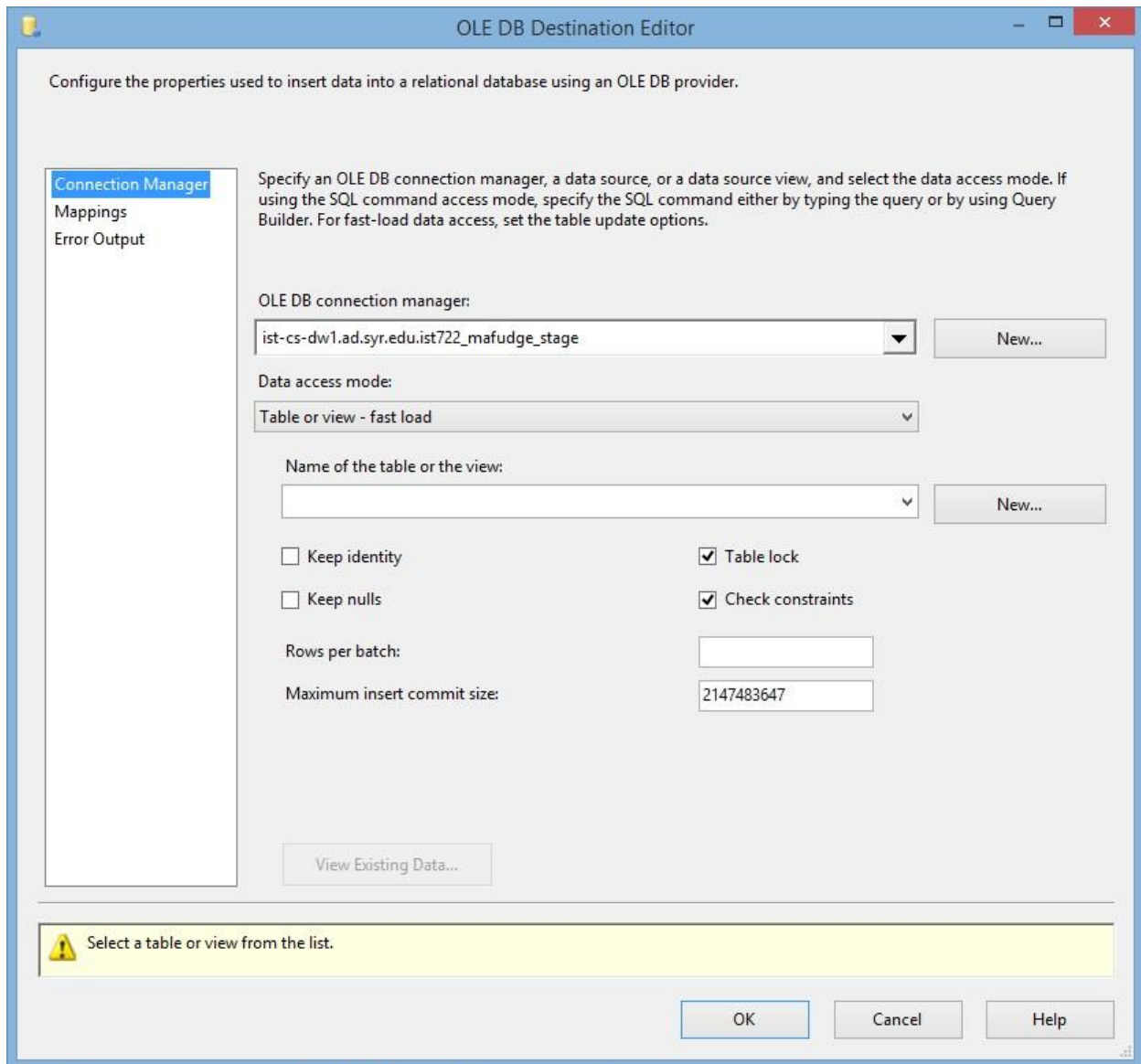
9. Click the destination and press **[F2]** to rename it to **DST – Stage stgDate**.



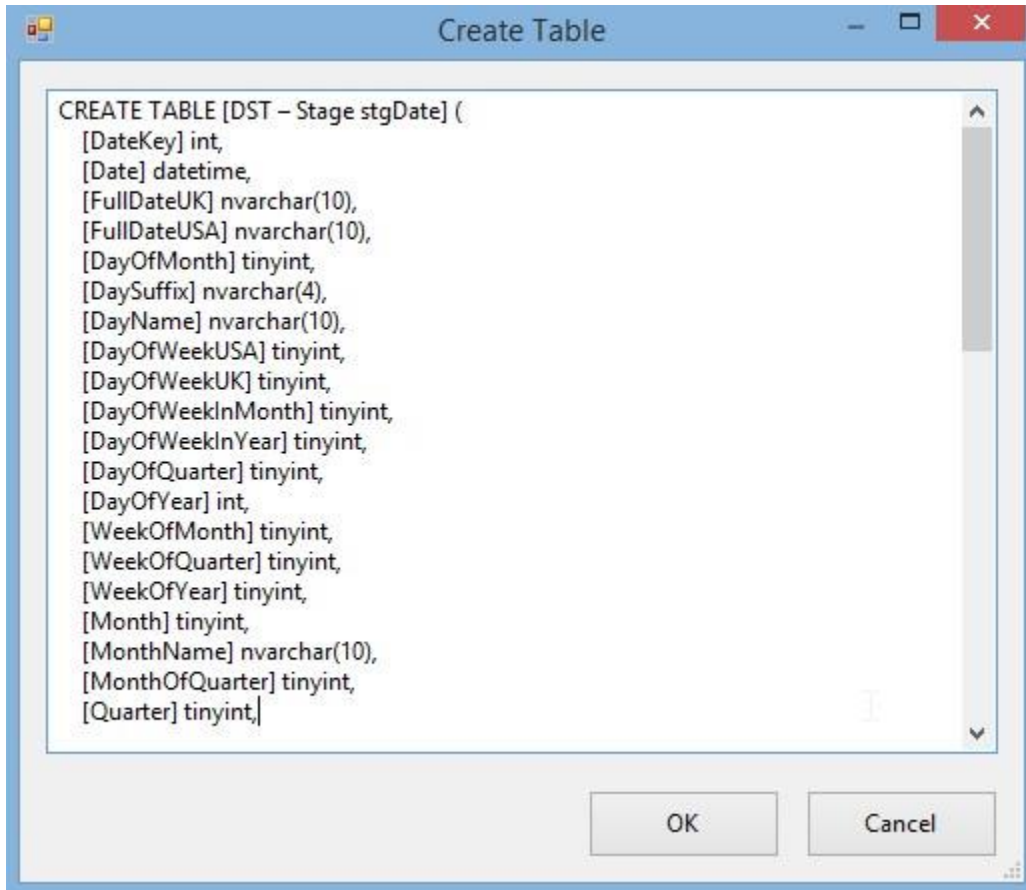
10. You probably haven't saved in a while. Press **CTRL + S** to save.
11. Now we need to take the **output** of the source and connect to the **input** of the destination.
To do that we simply click the source then drag and drop the blue arrow onto the destination.



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

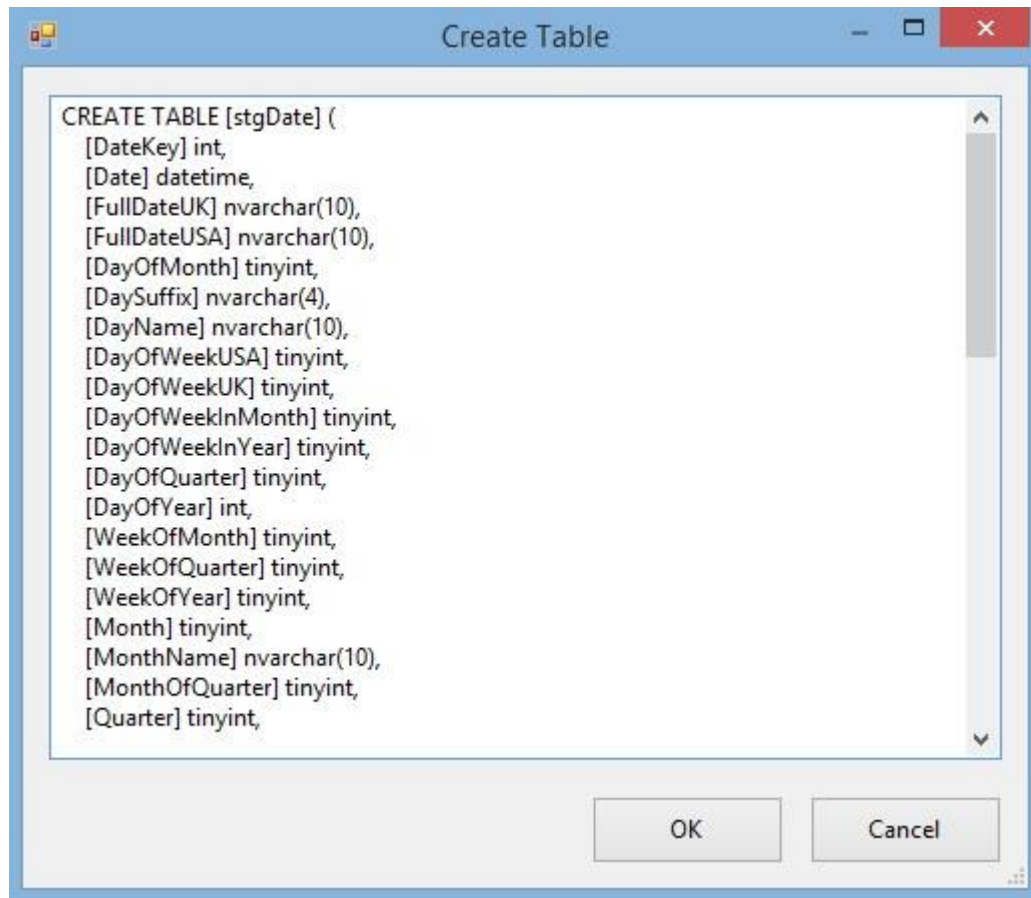


13. Our destination table does not exist. What do we do? The good news is SSIS can create the table for us. Where does it get the information to create this table? SSIS looks at the schema of the source data and automatically generates a CREATE TABLE statement from it! Click **New...** to create the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



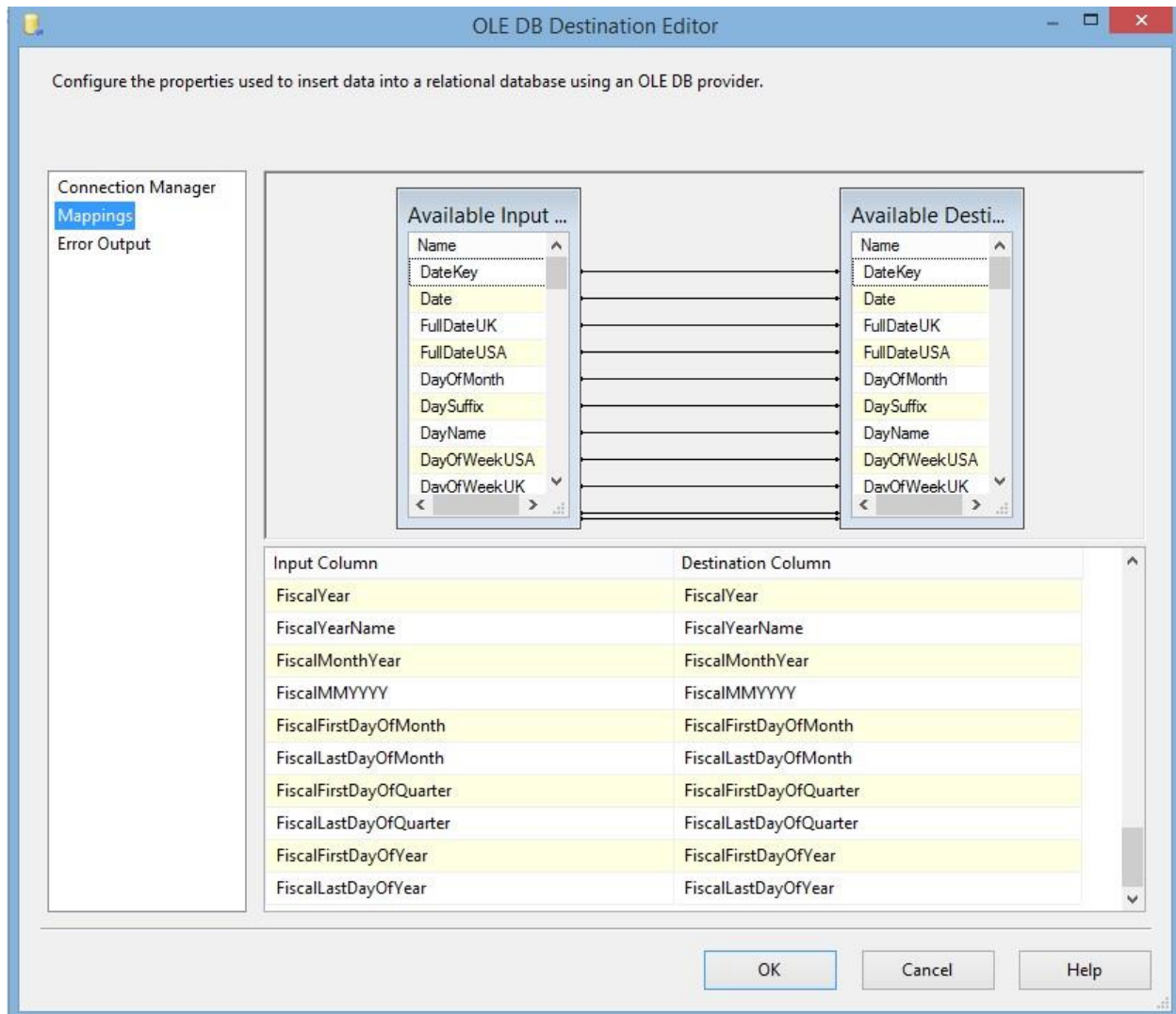
14. The only problem is the name of the table is wrong. Please change the first line of the statement to read:

CREATE TABLE [stgDate] (

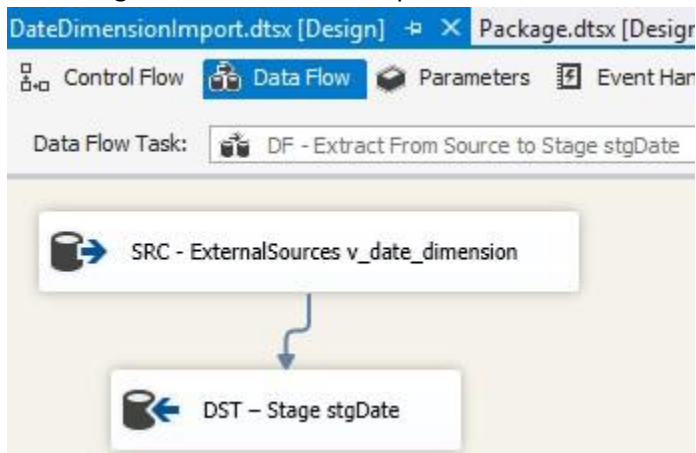


Then click **OK** to create the table.

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 menu, click **Mappings** to display the generated column mappings:



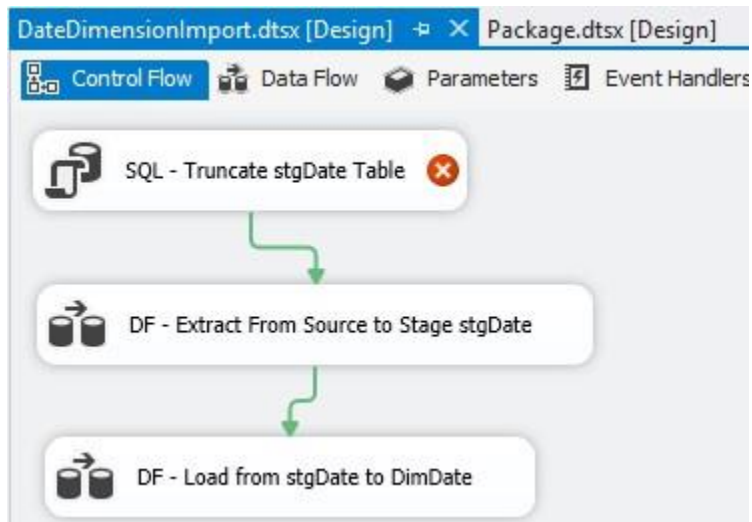
17. Click **OK** to complete the configuration of your destination. Your Data Flow design surface configuration should be complete.



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

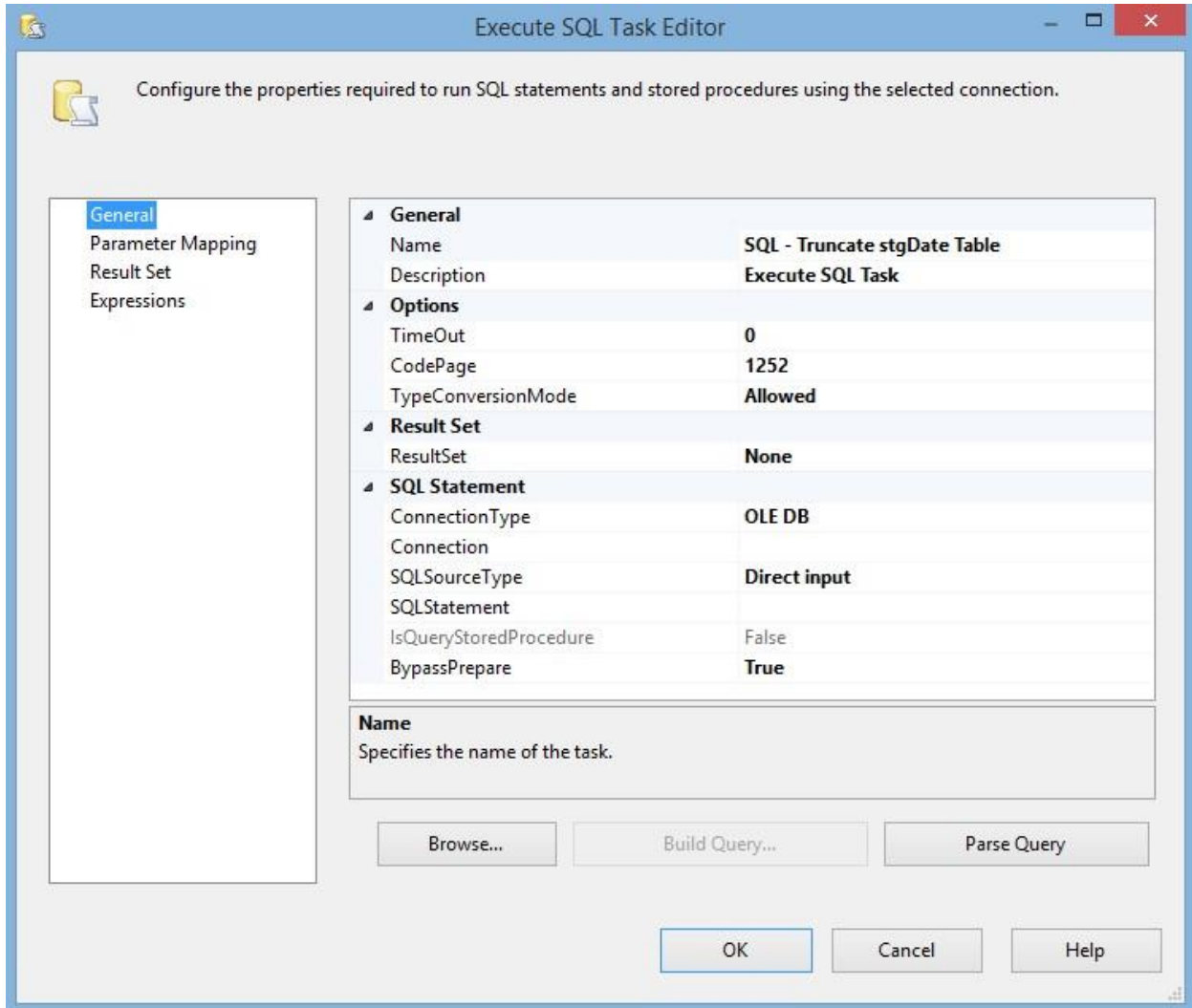
Step 2.1.4: Set Up the Truncate Table SQL Task

At this point we have completed half 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 the **SQL - Truncate stgDate Table** task to configure it. This will open the **Execute SQL Task Editor**.



The **Execute SQL Task Editor** dialog box is shown. It has a title bar with standard Windows window controls. Below the title bar is a subtitle: "Configure the properties required to run SQL statements and stored procedures using the selected connection." On the left is a sidebar with a tree view containing "General" (selected), "Parameter Mapping", "Result Set", and "Expressions". The main area displays the "General" tab with a table of properties. Below the table is a "Name" field with a description. At the bottom are buttons for "Browse...", "Build Query...", "Parse Query", "OK", "Cancel", and "Help".

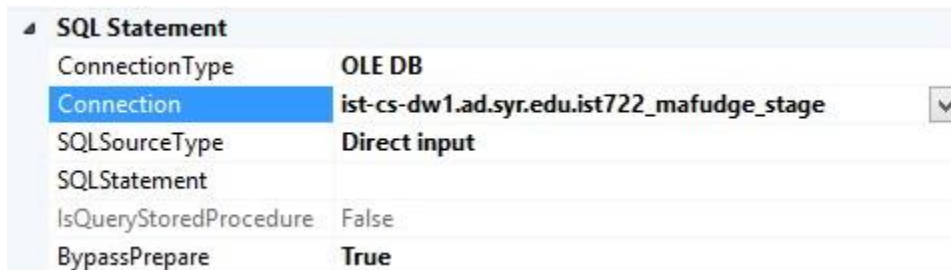
General	
Name	SQL - Truncate stgDate Table
Description	Execute SQL Task
Options	
TimeOut	0
CodePage	1252
TypeConversionMode	Allowed
Result Set	
ResultSet	None
SQL Statement	
ConnectionType	OLE DB
Connection	
SQLSourceType	Direct input
SQLStatement	
IsQueryStoredProcedure	False
BypassPrepare	True

Name
Specifies the name of the task.

Browse... Build Query... Parse Query

OK Cancel Help

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.



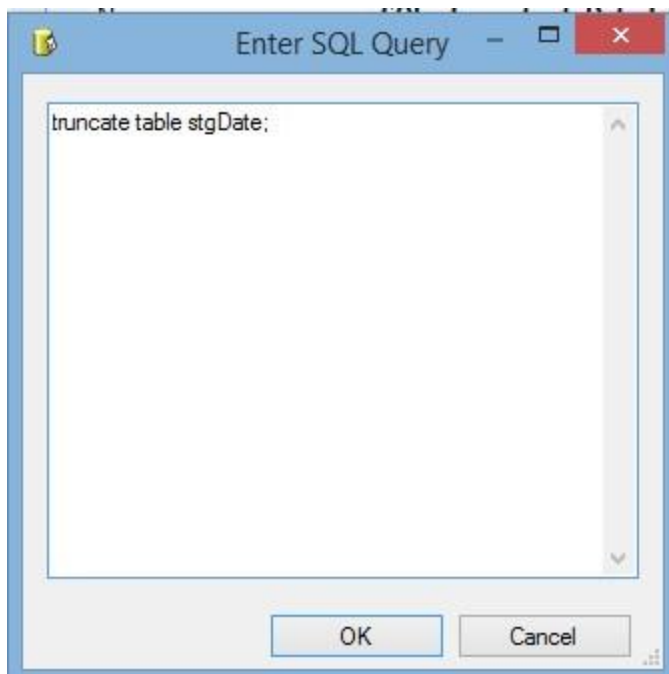
A close-up of the "SQL Statement" section of the dialog box. The "Connection" property is highlighted, and its dropdown menu is open, showing the selected value "ist-cs-dw1.ad.syr.edu.ist722_mafudge_stage".

SQL Statement	
ConnectionType	OLE DB
Connection	ist-cs-dw1.ad.syr.edu.ist722_mafudge_stage
SQLSourceType	Direct input
SQLStatement	
IsQueryStoredProcedure	False
BypassPrepare	True

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

SQL Statement	
ConnectionType	OLE DB
Connection	ist-cs-dw1.ad.syr.edu.ist722_mafudge_stage
SQLSourceType	Direct input
SQLStatement	
IsQueryStoredProcedure	False
BypassPrepare	True

- 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;



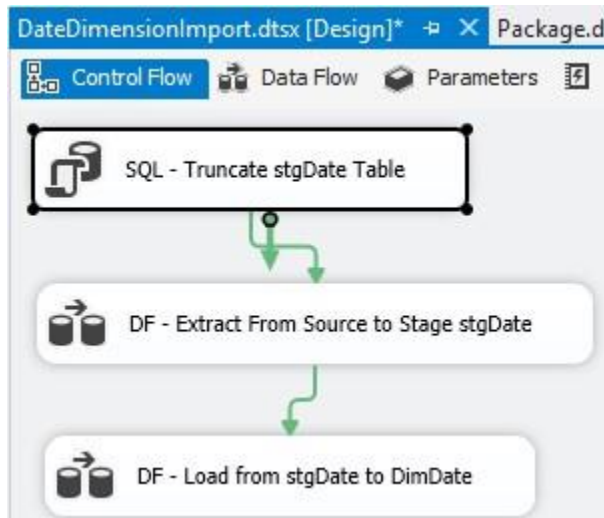
A dialog box titled "Enter SQL Query" with a text area containing the SQL command "truncate table stgDate;". The dialog has "OK" and "Cancel" buttons at the bottom.

IMPORTANT: It should be noted that 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, and then copy /paste the code into this box.

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

SQL Statement	
ConnectionType	OLE DB
Connection	ist-cs-dw1.ad.syr.edu.ist722_mafudge_stage
SQLSourceType	Direct input
SQLStatement	truncate table stgDate;
IsQueryStoredProcedure	False
BypassPrepare	True

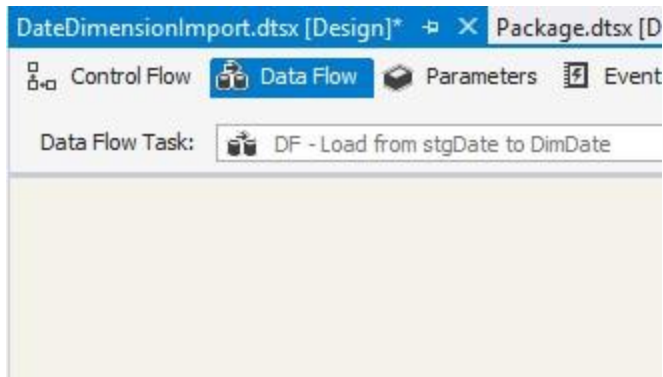
- Click **OK** to close the **Execute SQL Task Editor**, and then press **CTRL + S** to save. You should now have two steps in the control flow completed:



Step 2.1.5: Configure the Stage to Target Data Flow

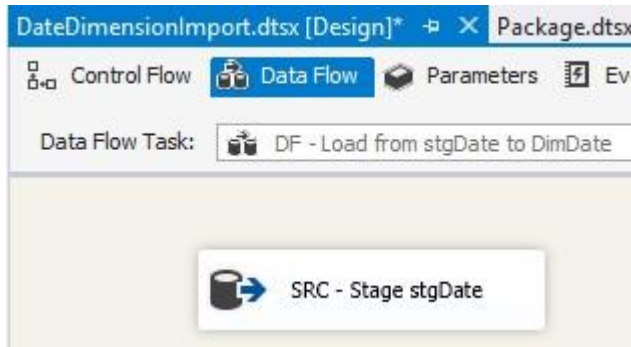
The final step is to configure the stage to target dimension data flow. In this data flow, we will use a Type 1 Slowly Changing Dimension to ensure that we do not unintentionally introduce the same business key more than once to the dimension.


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

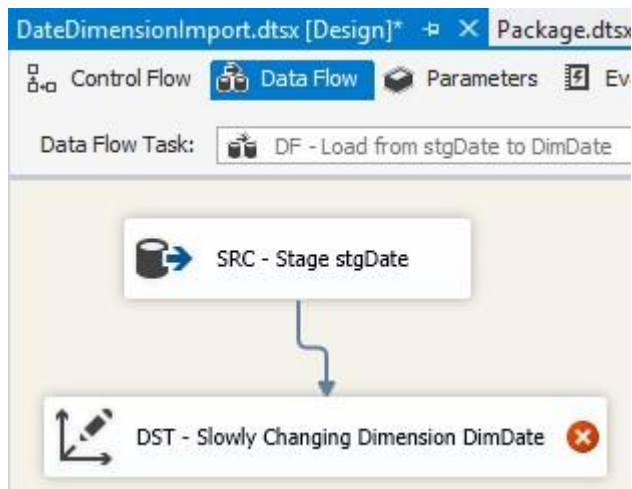


Since you've done a data flow task once before, you shouldn't need the same amount of hand-holding.

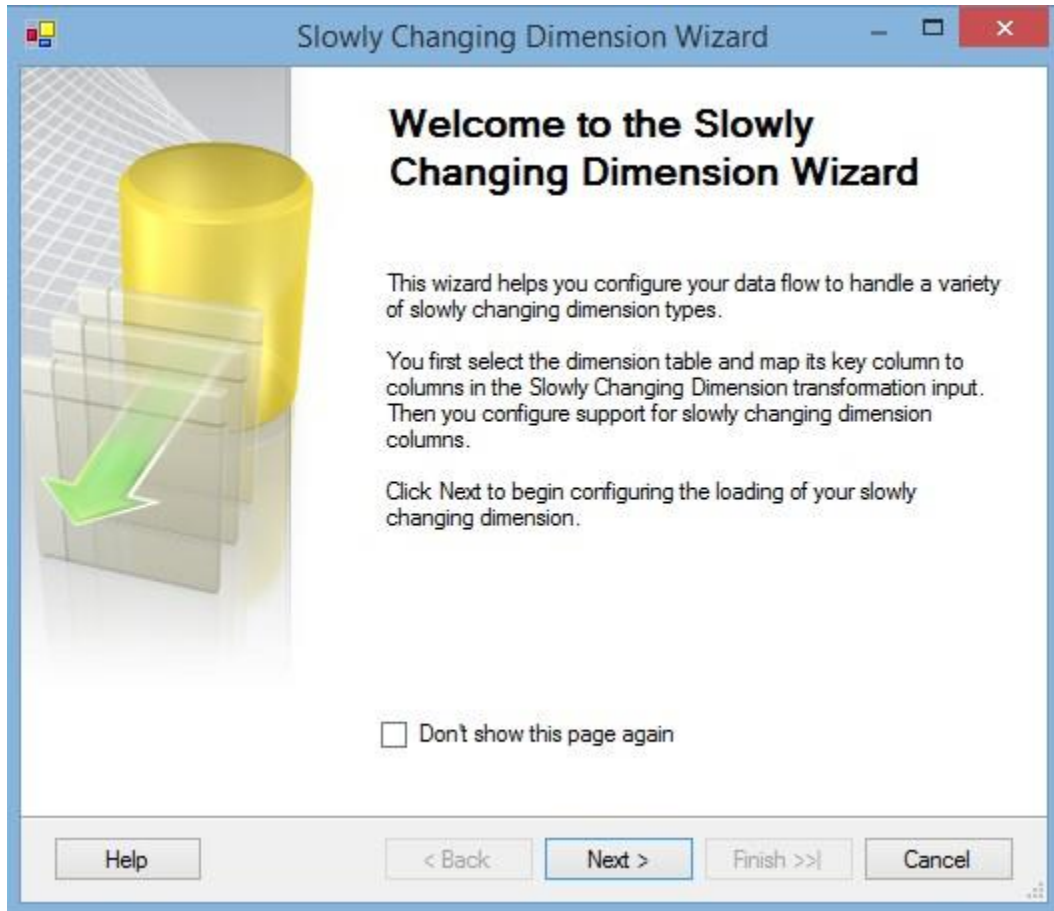
2. Drag and drop the **Source Assistant** onto the design surface. Select **SQL Server** as the **source type** and **ist722_yournetid_stage** as the connection manager.
3. Rename the source to **SRC - Stage stgDate**, then double-click it to configure.
4. From the **OLE DB Source Editor** select the **[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 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.

Select a Dimension Table and Keys
Select a dimension table to load and map columns in the transformation input to columns in the dimension table.

Connection manager:
ist-cs-dw1.ad.syr.edu.ist722_mafudge_dw

Table or view:
[northwind].[DimDate]

Input Columns	Dimension Columns	Key Type
Date	Date	Not a key column
DateKey	DateKey	Not a key column
DayName	DayName	Not a key column
DayOfMonth	DayOfMonth	Not a key column
	DayOfWeek	
DayOfYear	DayOfYear	Not a key column
FullDateUSA	FullDateUSA	Not a key column
	IsAWeekday	
MonthName	MonthName	Not a key column
	MonthOfYear	
Quarter	Quarter	Not a key column
QuarterName	QuarterName	Not a key column
WeekOfYear	WeekOfYear	Not a key column
Year	Year	Not a key column

⚠ At least one business key needs to be selected.

Help < Back Next > Finish >> Cancel

Now you need to configure the source to target mapping of the columns and select the column(s) that 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 Type 2).
- 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.

Select a Dimension Table and Keys
Select a dimension table to load and map columns in the transformation input to columns in the dimension table.

Connection manager:
ist-cs-dw1.ad.syr.edu.ist722_mafudge_dw

Table or view:
[northwind].[DimDate]

Input Columns	Dimension Columns	Key Type
Date	Date	Not a key column
DateKey	DateKey	Business key
DayName	DayName	Not a key column
DayOfMonth	DayOfMonth	Not a key column
DayOfWeekUSA	DayOfWeek	Not a key column
DayOfYear	DayOfYear	Not a key column
FullDateUSA	FullDateUSA	Not a key column
IsWeekdayYesNo	IsAWeekday	Not a key column
MonthName	MonthName	Not a key column
Month	MonthOfYear	Not a key column
Quarter	Quarter	Not a key column
QuarterName	QuarterName	Not a key column
WeekOfYear	WeekOfYear	Not a key column
Year	Year	Not a key column

Help < Back Next > Finish >> Cancel

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.

Slowly Changing Dimension Columns
Manage the changes to column data in your slowly changing dimensions by setting the change type for dimension columns.

Fixed Attribute
Select this type when the value in a column should not change. Changes are treated as errors.

Changing Attribute
Select this type when changed values should overwrite existing values. This is a Type 1 change.

Historical Attribute
Select this type when changes in column values are saved in new records. Previous values are saved in records marked as outdated. This is a Type 2 change.

Select a change type for slowly changing dimension columns:

Dimension Columns	Change Type
Date	Changing attribute
DayName	Changing attribute
DayOfMonth	Changing attribute
DayOfWeek	Changing attribute
DayOfYear	Changing attribute
FullDateUSA	Changing attribute
IsAWekday	Changing attribute
MonthName	Changing attribute
MonthOfYear	Changing attribute
Quarter	Changing attribute
QuarterName	Changing attribute
WeekOfYear	Changing attribute
Year	Changing attribute

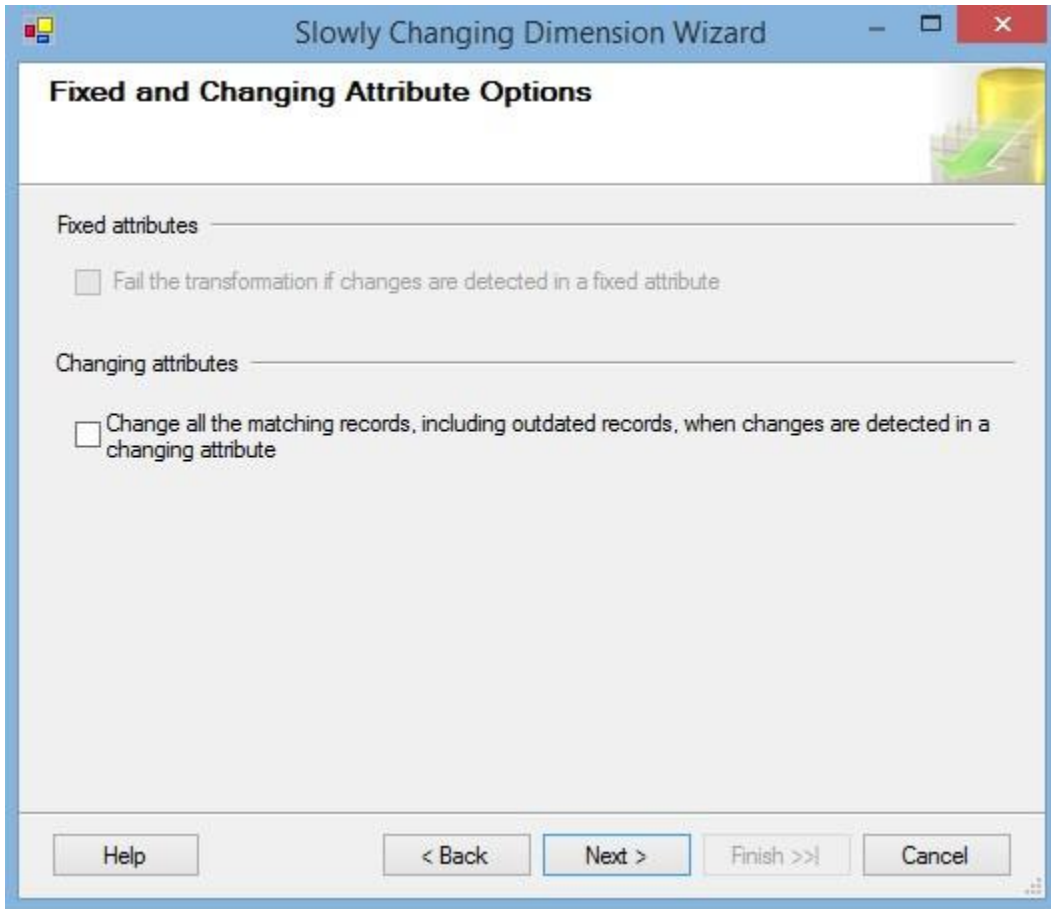
Remove

Help < Back Next > Finish >> Cancel

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

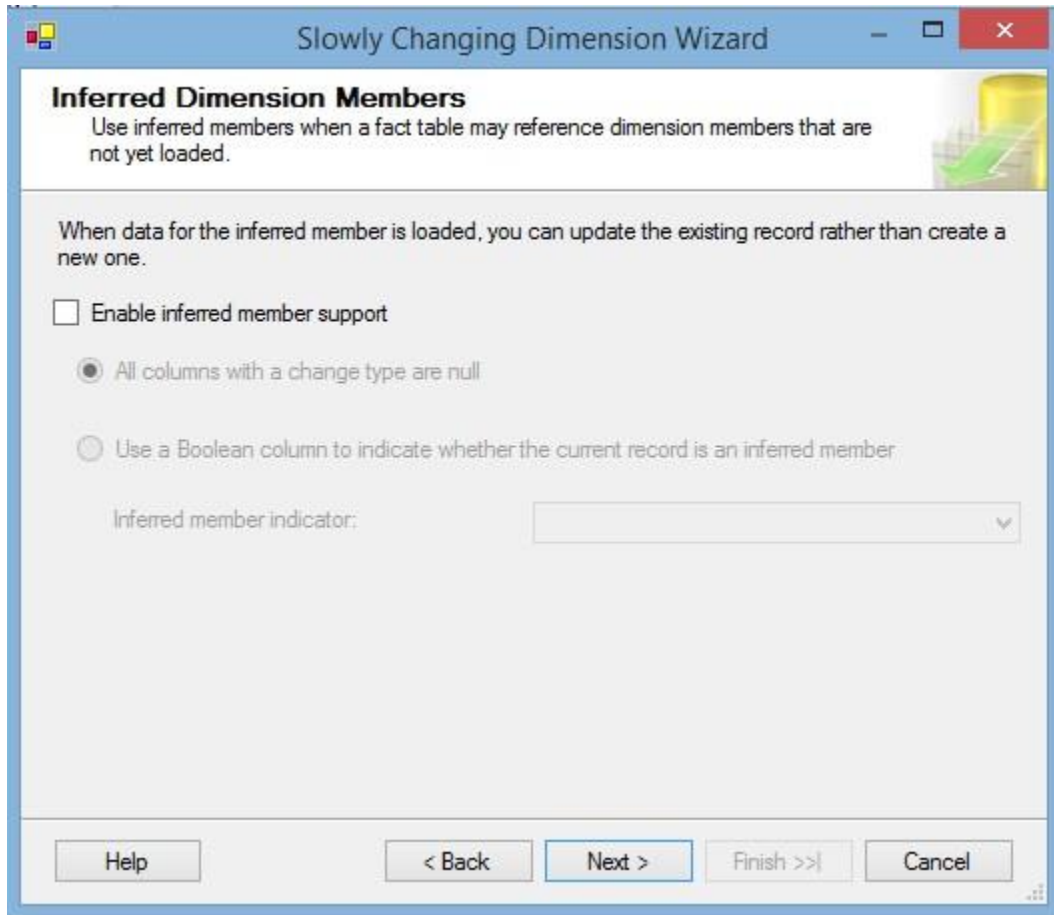
The user interface for this is a little wonky, using “magic dropdowns” that only appear when you click in the grid. Just remember the **[tab]** key is your friend here. When you’ve completed this step, click **Next >**.

- Next up is the **Fixed and Changing Attribute Options** screen. For the date dimension, we want **both of these check-boxes cleared** since an update here will only change one row.



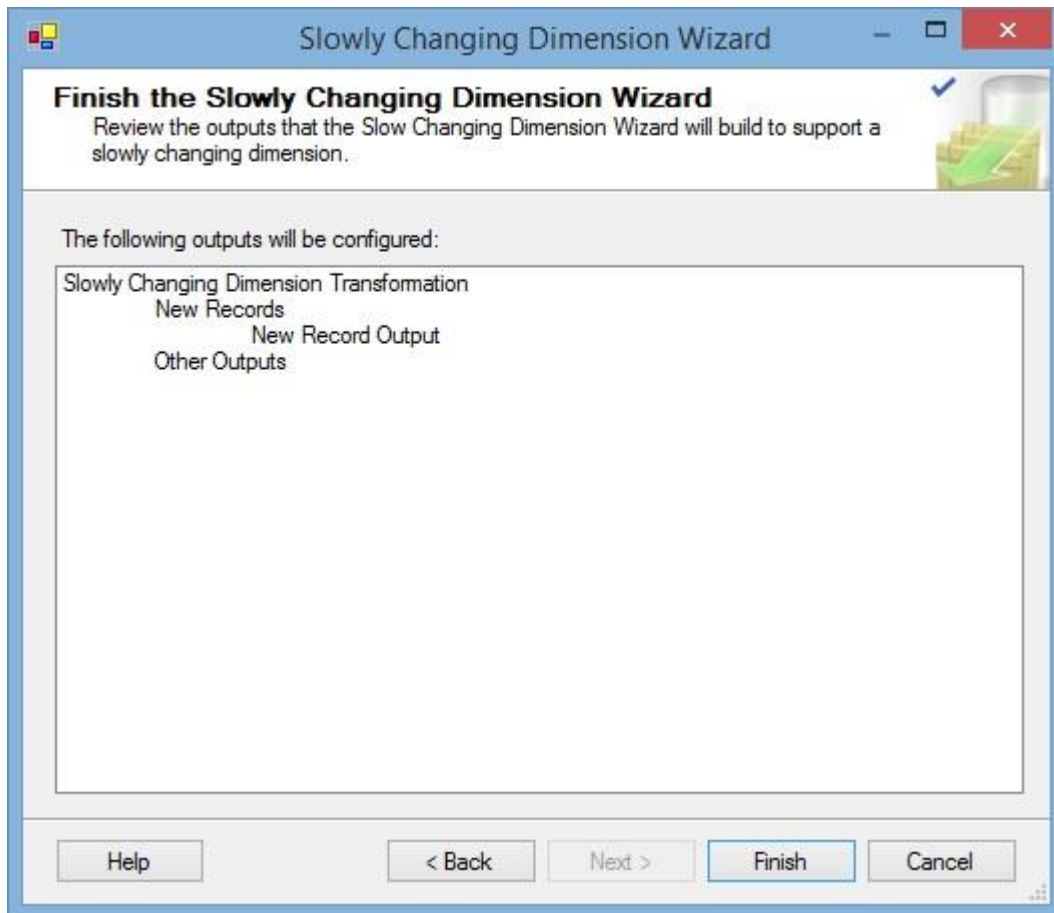
When you're ready, click **Next >**.

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 does not apply here, **clear the check box**.

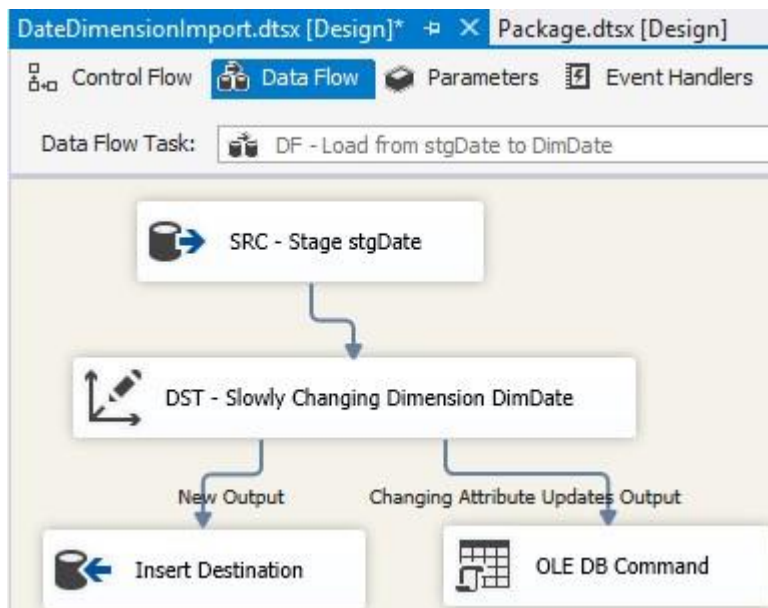


When you're ready, click **Next >**.

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.

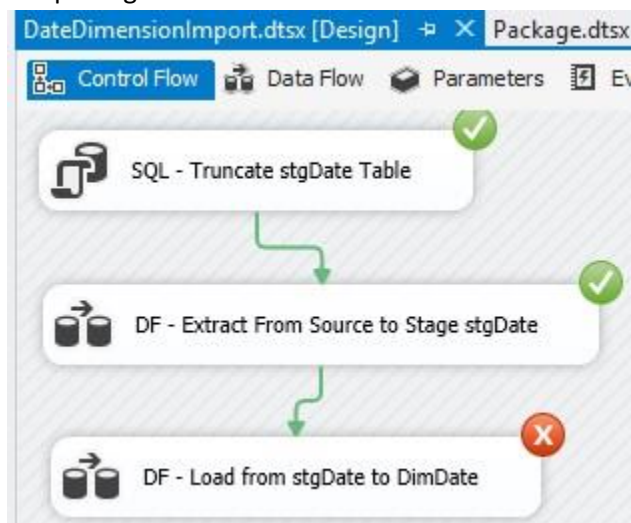
Step 2.1.6: Executing and Troubleshooting Your SSIS Package

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.

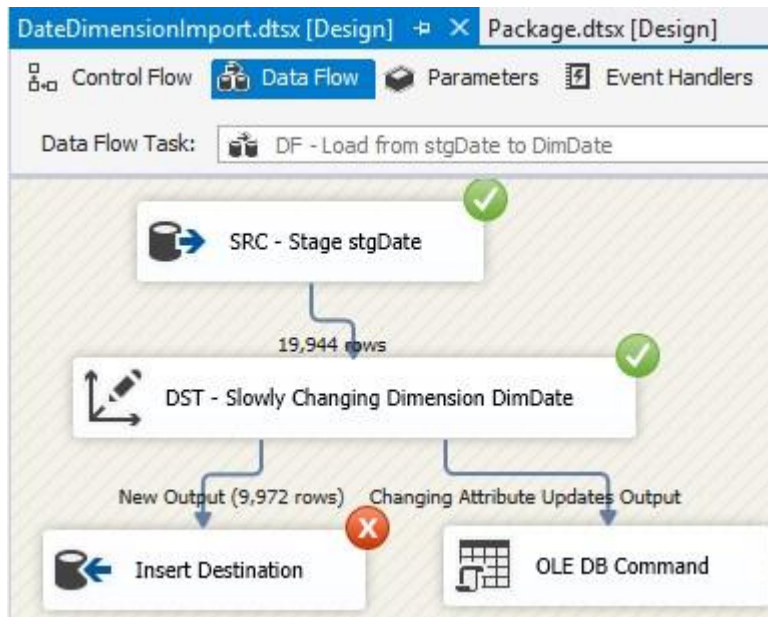
As we'll see in a minute, the SSIS errors are rather cryptic. Troubleshooting SSIS errors is a lot of common sense. You need to think about the operation you're doing and brainstorm the potential issues which may occur.

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.



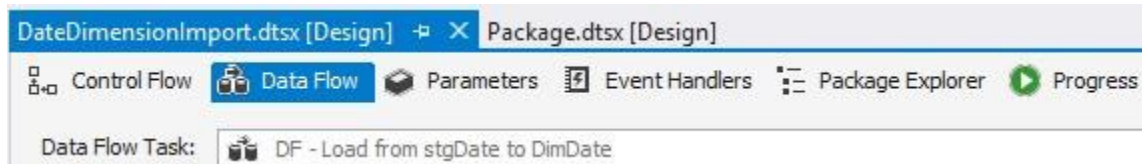
NOTE: If you haven't figured it out yet, this was by design. I intentionally added an error so you can learn how to troubleshoot.

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

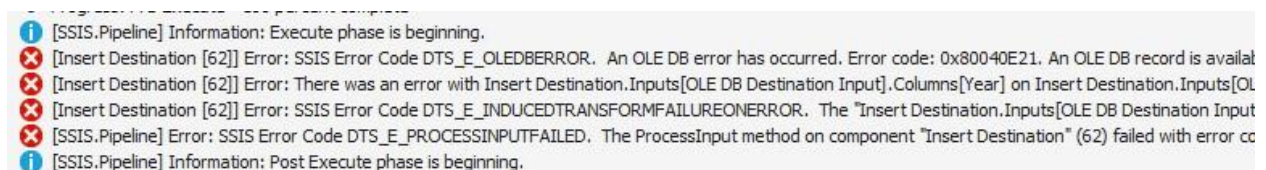


In the data flow you can see that the task failed when trying to insert data into the destination table **northwind.DimDate**.

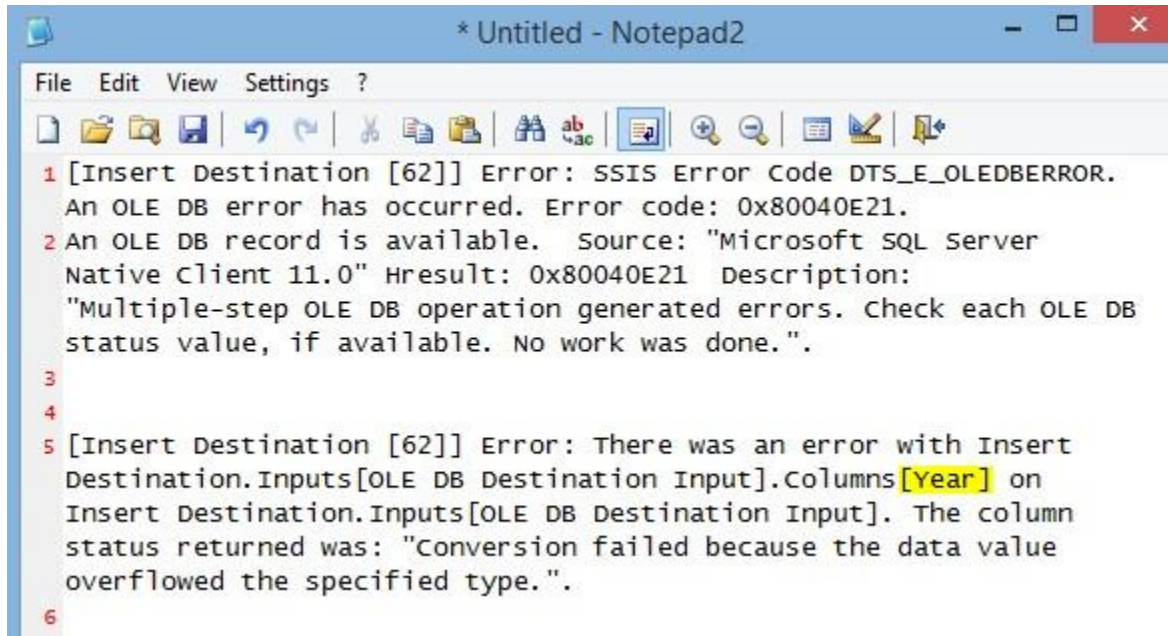
- Let's check the error message. At the top of the **design surface** you will see a **progress tab**.



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

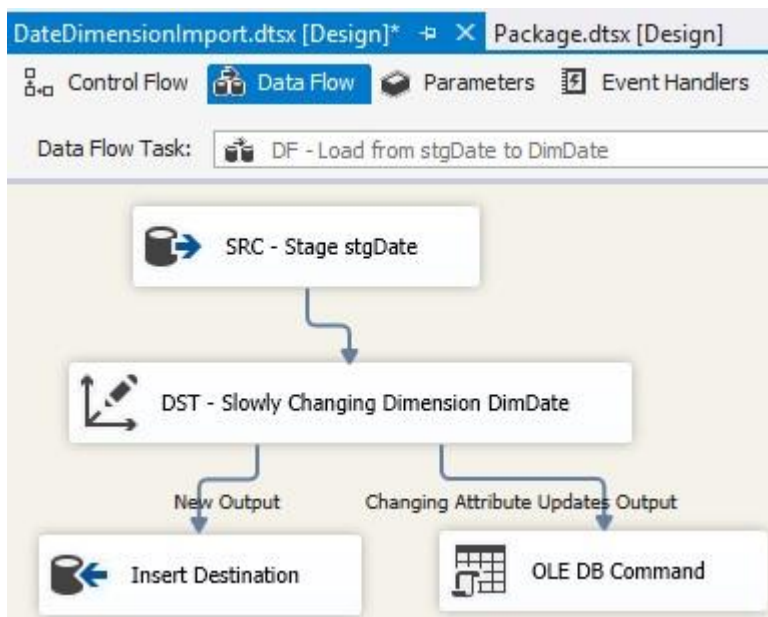


- Here's where things get really ugly. The error is so long, you cannot view it in the progress window! Ugh. Your best bet is to **right-click each error** then select **Copy Message Text** from the menu to copy the error message, and then paste it into a text editor like Word or Notepad. You should do this for each error until you find one that gives an actual hint as to what the problem might be.



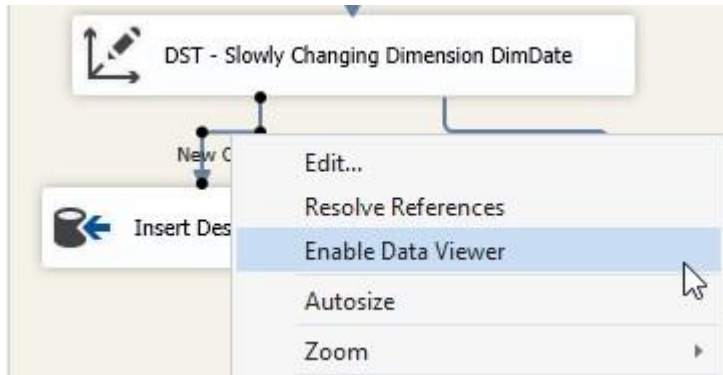
In the screenshot above I've highlighted the column giving us trouble – the **[Year]** column in the destination **northwind.DimDate**. For some reason it cannot insert the values from the source. Furthermore, the message specifies an “overflow,” which typically means the source data type is too large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

- 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 are no more diagonal lines in the background because the package is no longer in execution mode.

6. To enable the 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 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 rerun 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 four-digit years.

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

▶ Detach Copy Data

...	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
..	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 the **magnifying glass icon** and selecting **Disable Data Viewer**.

- 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 four-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.

```
139 CREATE TABLE [northwind].[DimDate](  
140     [DateKey] [int] NOT NULL,  
141     [Date] [datetime] NULL,  
142     [FullDateUSA] [nchar](11) NOT NULL,  
143     [DayOfWeek] [tinyint] NOT NULL,  
144     [DayName] [nchar](10) NOT NULL,  
145     [DayOfMonth] [tinyint] NOT NULL,  
146     [DayOfYear] [int] NOT NULL,  
147     [WeekOfYear] [tinyint] NOT NULL,  
148     [MonthName] [nchar](10) NOT NULL,  
149     [MonthOfYear] [tinyint] NOT NULL,  
150     [Quarter] [tinyint] NOT NULL,  
151     [QuarterName] [nchar](10) NOT NULL,  
152     [Year] [tinyint] NOT NULL,  
153     [IsAWeekday] varchar(1) NOT NULL DEFAULT (('N')),  
154     constraint pkNorthwindDimDate PRIMARY KEY ([DateKey])  
155 )
```

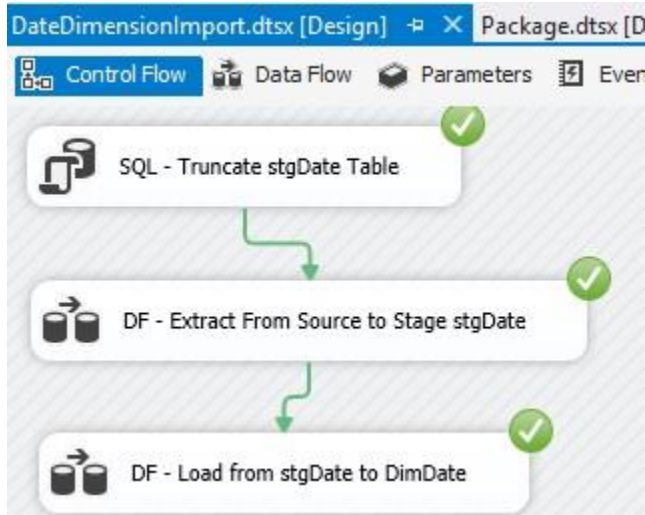
The **tinyint** data type is too small (it's only 1 byte in size), and can only represent numbers from -128 to +127. Four-digit years are outside this range.

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

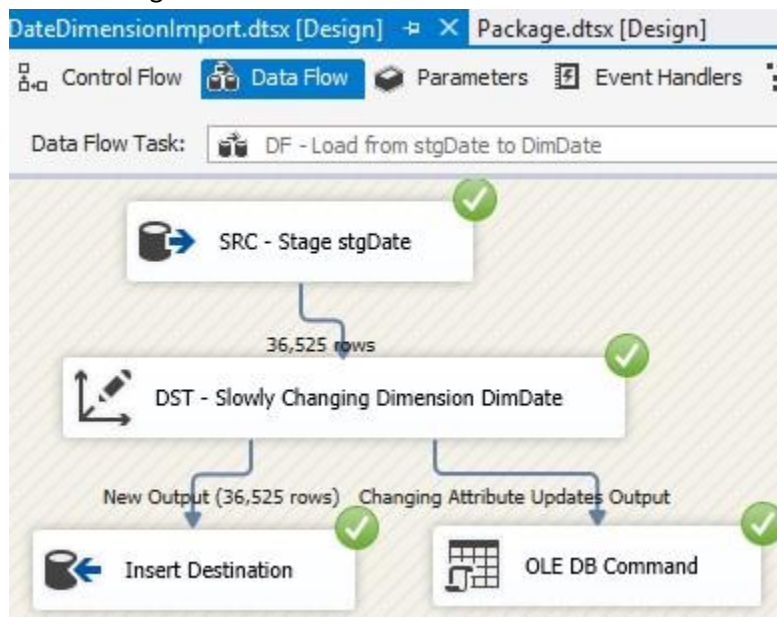
```
139 CREATE TABLE [northwind].[DimDate](  
140     [DateKey] [int] NOT NULL,  
141     [Date] [datetime] NULL,  
142     [FullDateUSA] [nchar](11) NOT NULL,  
143     [DayOfWeek] [tinyint] NOT NULL,  
144     [DayName] [nchar](10) NOT NULL,  
145     [DayOfMonth] [tinyint] NOT NULL,  
146     [DayOfYear] [int] NOT NULL,  
147     [WeekOfYear] [tinyint] NOT NULL,  
148     [MonthName] [nchar](10) NOT NULL,  
149     [MonthOfYear] [tinyint] NOT NULL,  
150     [Quarter] [tinyint] NOT NULL,  
151     [QuarterName] [nchar](10) NOT NULL,  
152     [Year] [smallint] NOT NULL,  
153     [IsAWeekday] varchar(1) NOT NULL DEFAULT (('N')),  
154     constraint pkNorthwindDimDate PRIMARY KEY ([DateKey])  
155 )
```

NOTE: A **smallint** is two 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 one billion rows, we'd be "wasting" 2 GB 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, 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 your SSIS packages 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!

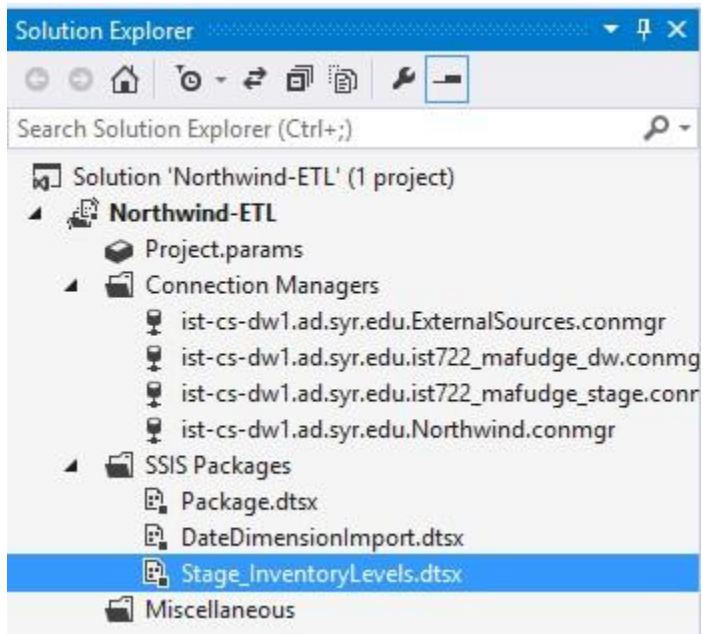
Part 2.2: Staging the Inventory Levels Dimensional Model

In this next part, we will create a package which performs half 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 **SSIS Packages** and select **New SSIS Package** from the menu.
2. Rename the package **Stage_InventoryLevels.dtsx**.

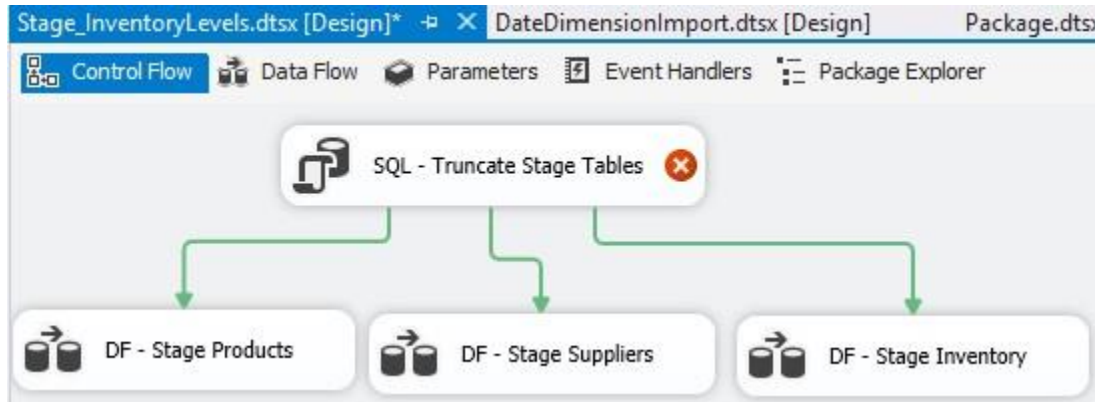


3. Double-click the **Stage_InventoryLevels.dtsx** package to open it in the **design surface**.

Step 2.2.2: Set Up the Control Flow

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

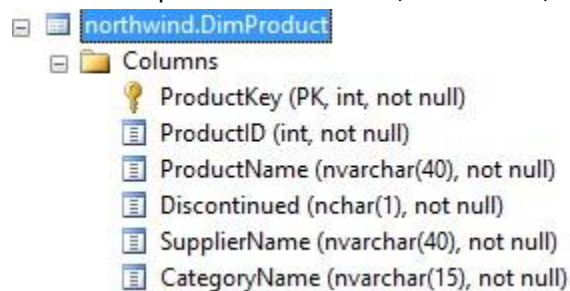
1. Add one **Execute SQL Task** and three **Data Flow Tasks** to your design surface.
2. Rename the **Execute SQL Task** → **SQL – Truncate Stage Tables**
3. Rename the three data flow tasks:
 - a. **DF – Stage Products**
 - b. **DF – Stage Suppliers**
 - c. **DF – Stage Inventory**
4. Connect the SQL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



Step 2.2.3: Staging Products

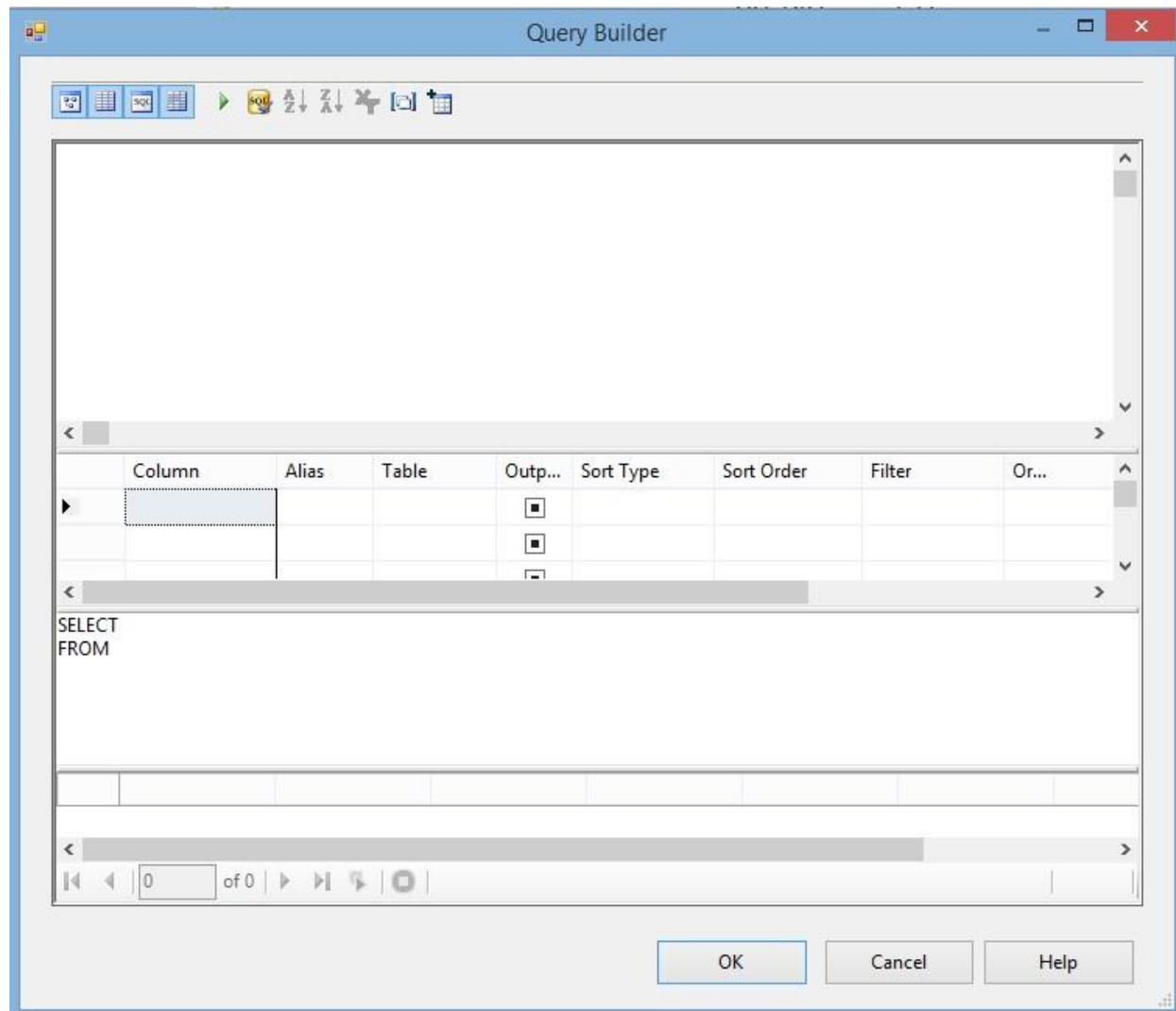
First we will work on staging products. Unlike the previous example where we build the entire package and then executed it, this time we will build, execute, and verify before staging the next source.


1. Double-click **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 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:

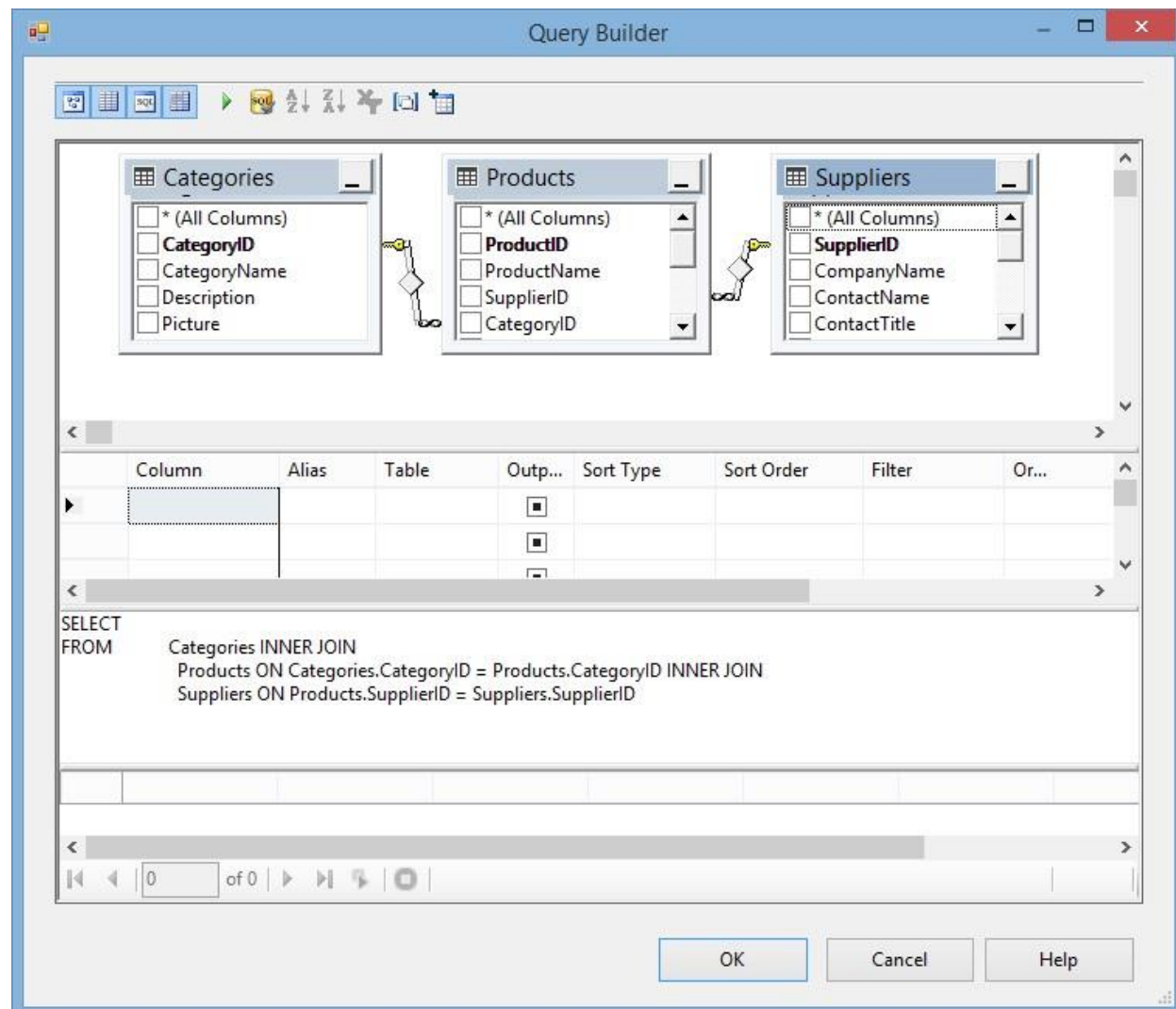


We need ProductID, ProductName, Discontinued, SupplierName and CategoryName. These do not come from the same table, so we'll need to write our own query:

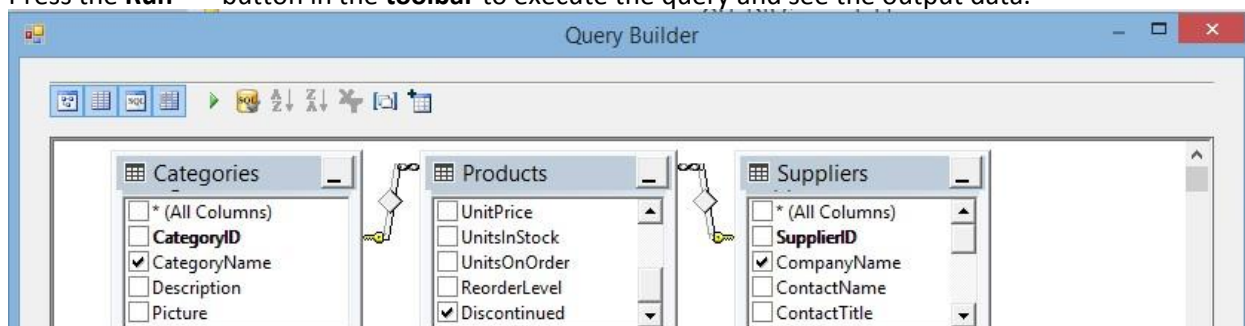
- a. In the **OleDb Source Editor** change the **Data access mode** to **SQL Command**.



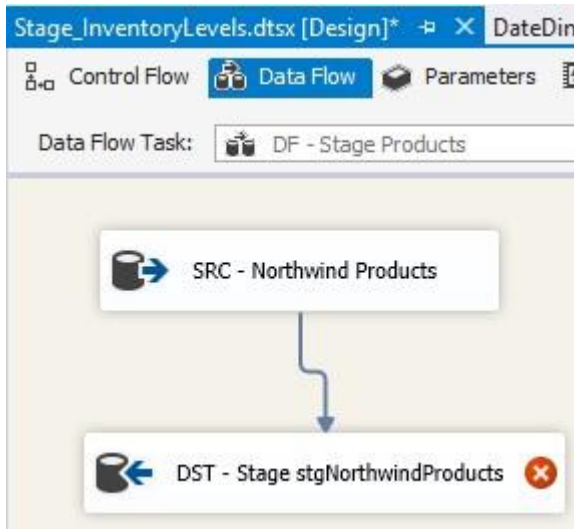
- Click the **Add Table**. 
- b. Click the **Build Query...** button to visually design our SQL query using the **Query Builder**!



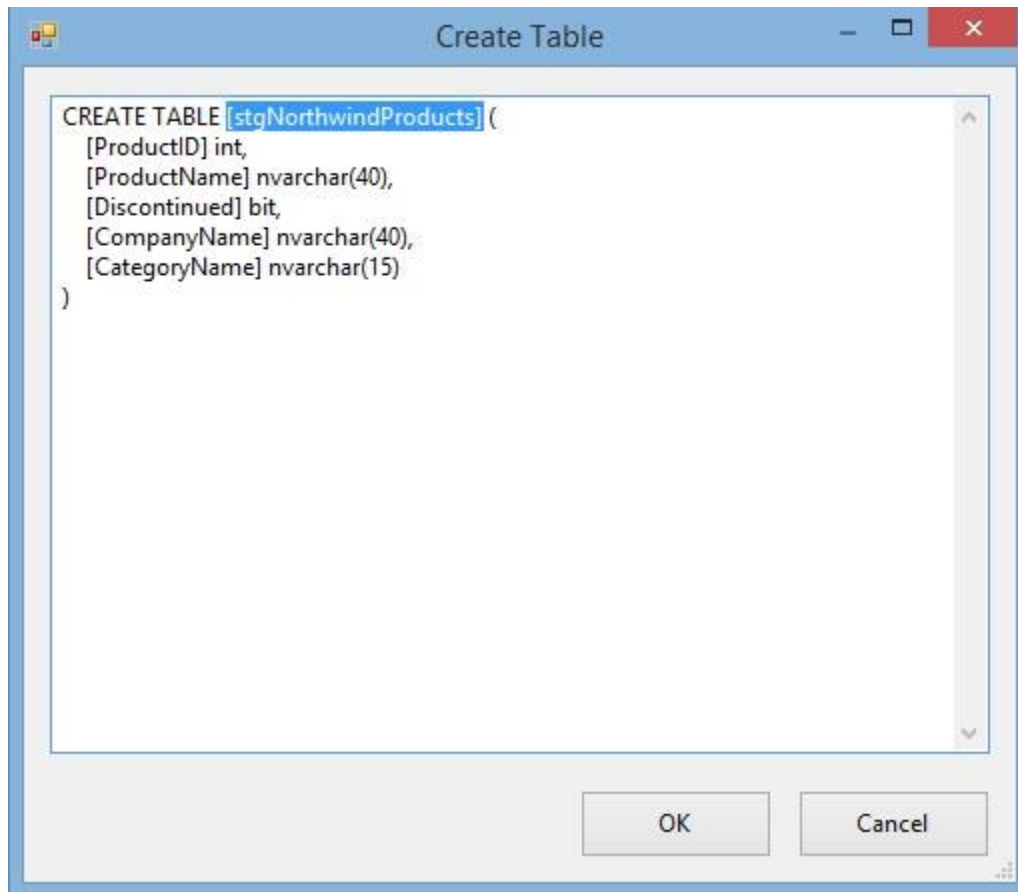
- c. button in the **toolbar**, then add the **Categories**, **Products**, and **Suppliers** tables and click close. The query builder is starting to create an SQL SELECT statement for you. Thanks to foreign keys, the join statements have been constructed for us!
- d. Next we add the columns we need. Simply click 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**.
- f. Press **OK** to save your query as the source for the **OLE DB Source Editor**.
- e. Press the **Run** button in the **toolbar** to execute the query and see the output data.



- 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 the destination to configure it. This will bring up the **OLE DB Destination Editor**.
 - a. Click **New...** 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]**.

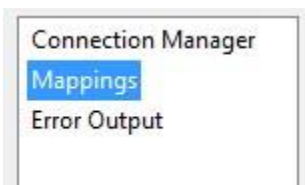


Click **OK** to create the table.

- 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 **mappings** on the left side of the window.

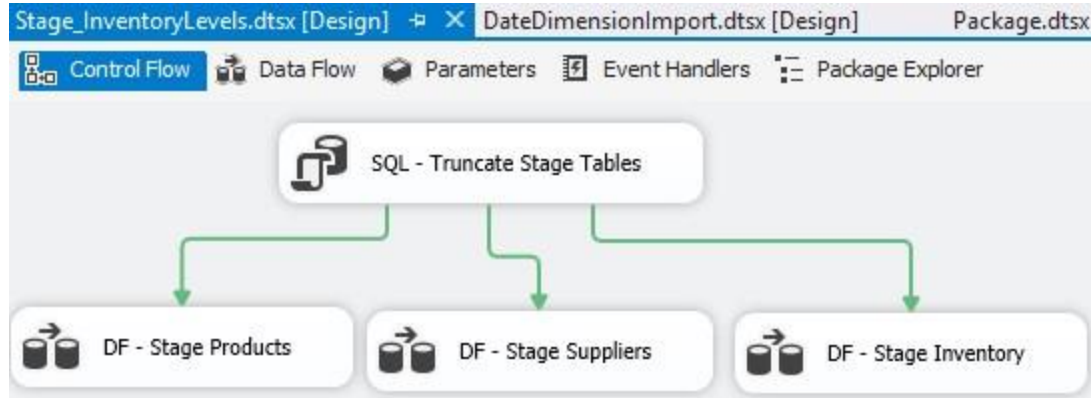


NOTE: This step is automatic because the source and destination have the same columns!

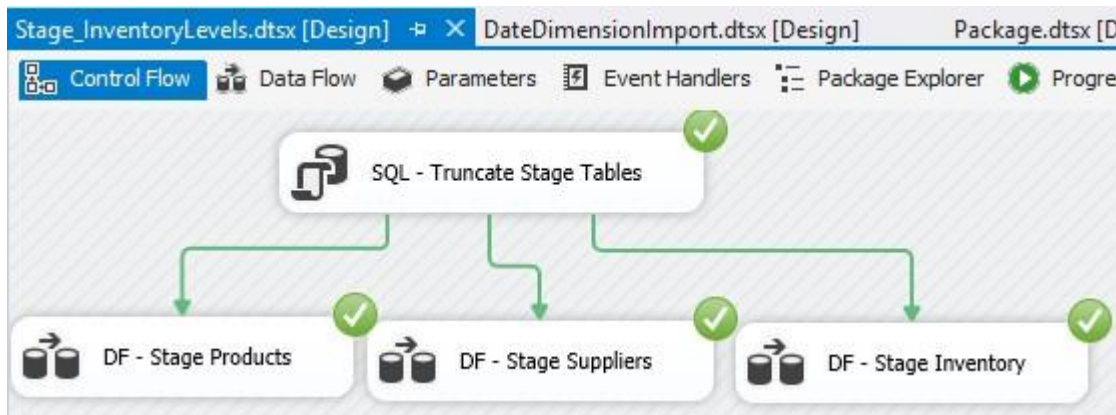
- e. Click **OK** to complete the configuration of the destination.
8. Save your work. Return to the **Control Flow** tab.
9. We need to truncate the table before the data flow occurs. Double-click the **SQL - Truncate Stage Tables** task.

- 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.)

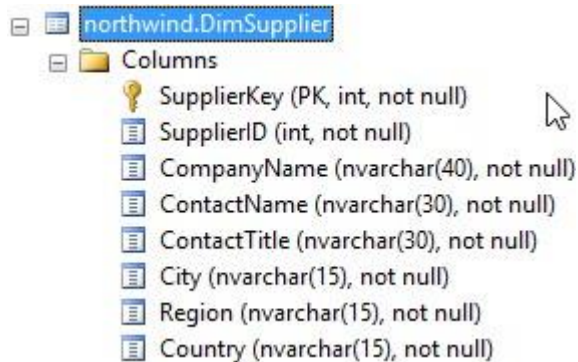
12. Stop package execution, by pressing **[Shift]+[F5]**
13. Verify there is data in the **stgNorthwindProducts** table in your **stage** database by running a simple SQL SELECT statement on the table. There should be **77 rows**.

Step 2.2.4: Staging Suppliers

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

1. Double-click **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**

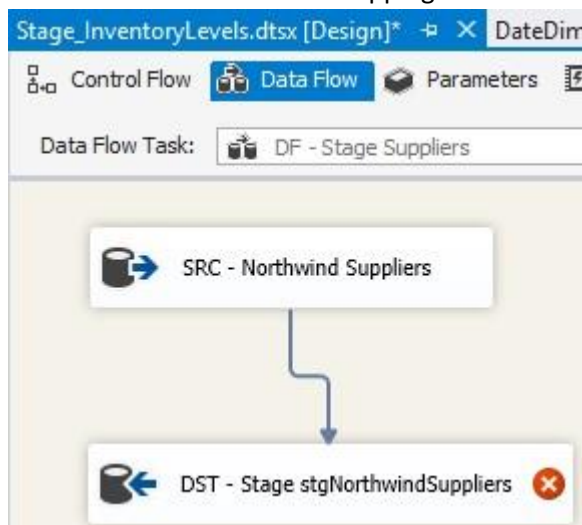
3. Double-click the data source to configure it. This opens the **OLEDB Source Editor**. Once again, we need to build a query to match the target data of the dimension:



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

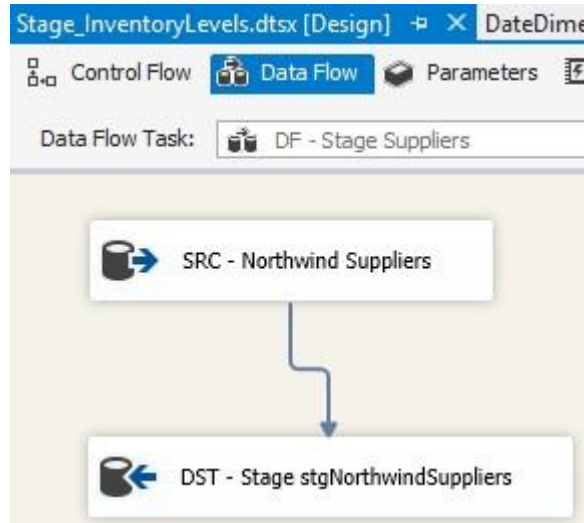
- a. For Data Access Mode select **SQL Command**
 - b. Use the **Build Query...** button to create the query. Here's the SQL if you're lazy:

```
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.

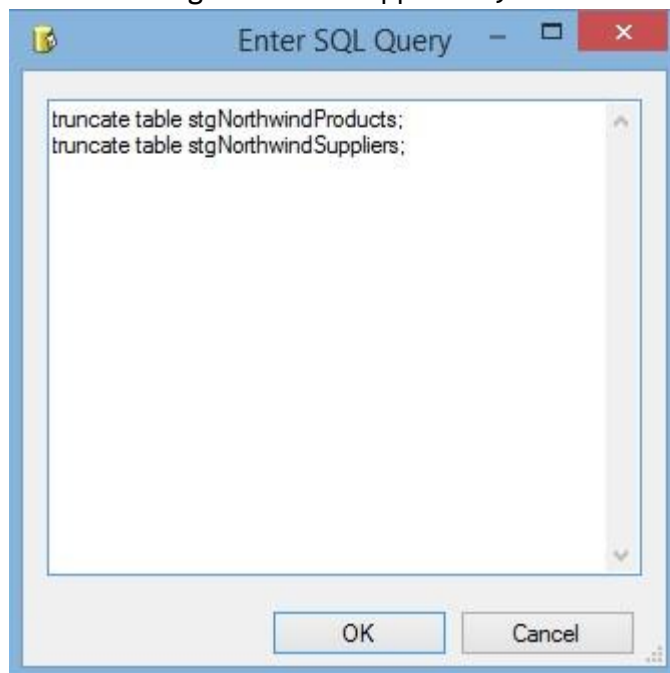


7. Double-click the destination to configure it. This will bring up the **OLE DB Destination Editor**.

- a. Create a new target table, named **[stgNorthwindSuppliers]**. Again, there's no table in the **stage** database right now.
- b. Click **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 the **Control Flow** tab.
9. Now we need to add the TRUNCATE TABLE statement to the **SQL - Truncate Stage Tables** task. Double-click it to bring up the **Execute SQL Task Editor**.
 - a. Under **SQL Statement** click **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.

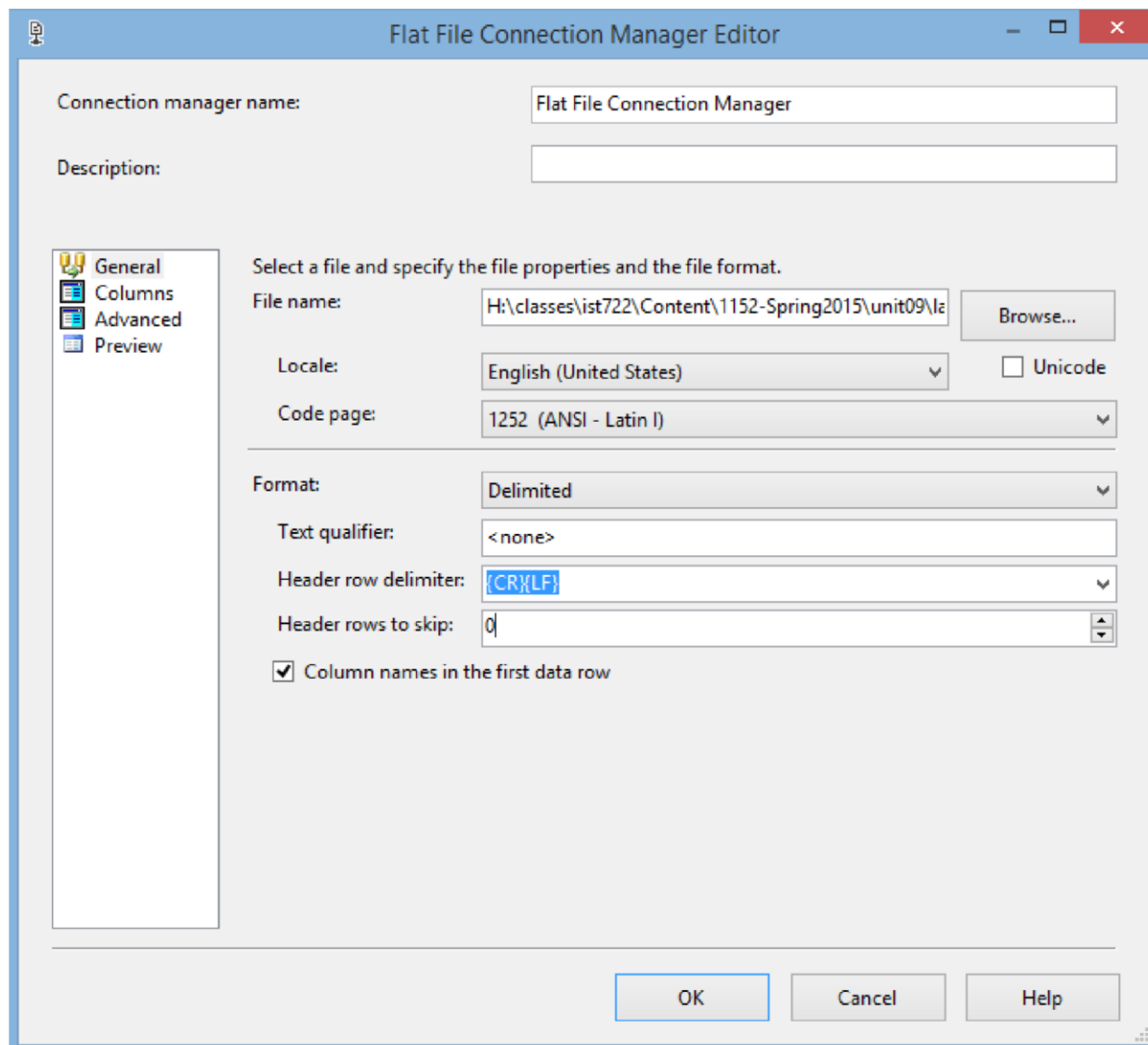
- c. Click **OK** to close the **Execute SQL Task Editor**.

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.

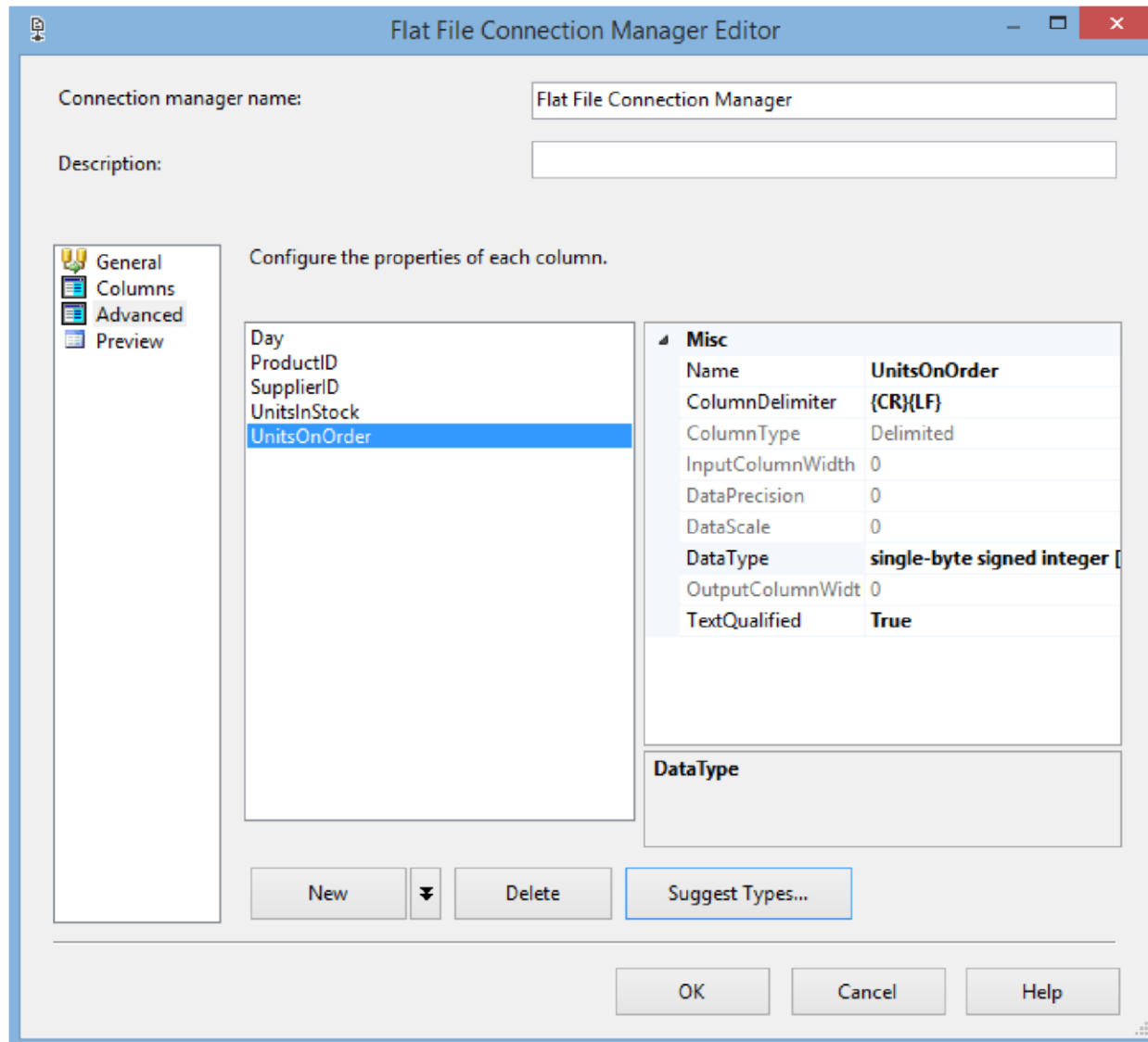
1. Double-click **DF - Stage Inventory** to open the data flow.



The image shows the 'Flat File Connection Manager Editor' dialog box. It has a title bar with standard window controls. The 'Connection manager name' is 'Flat File Connection Manager'. The 'Description' field is empty. On the left, there is a tree view with 'General' selected. The main area contains the following fields: 'File name' with the path 'H:\classes\ist722\Content\1152-Spring2015\unit09\la' and a 'Browse...' button; 'Locale' set to 'English (United States)' with a 'Unicode' checkbox; 'Code page' set to '1252 (ANSI - Latin I)'; 'Format' set to 'Delimited'; 'Text qualifier' set to '<none>'; 'Header row delimiter' set to 'CR|LF'; 'Header rows to skip' set to '0'; and a checked checkbox for 'Column names in the first data row'. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

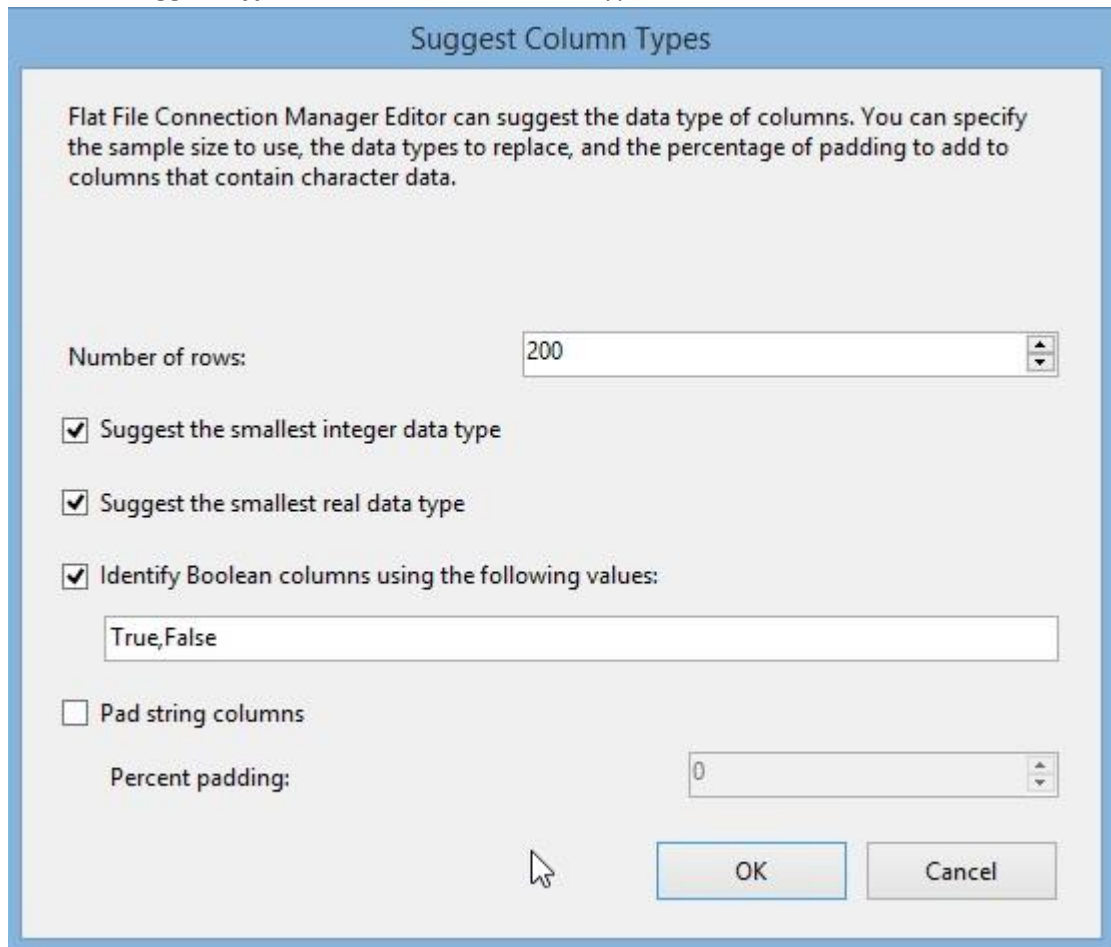
2. Use the **Source Assistant** to create the data source:
 - a. Type: **Flat File**

- b. Connection Manager → **New...** (since we have no connection for this)



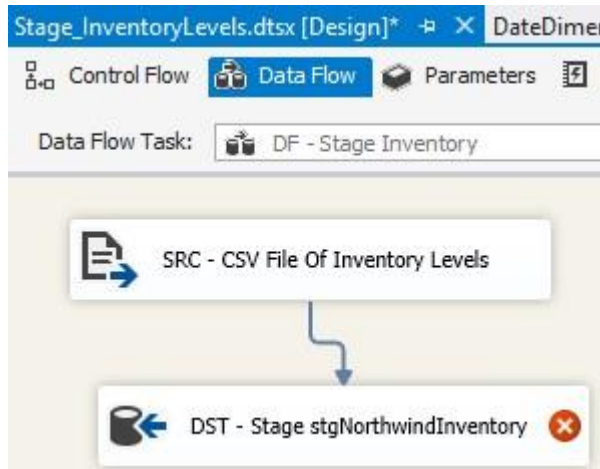
- c. The **Flat File Connection Manager Editor** dialog appears. Click **Browse...** to locate the **NorthwindDailyInventoryLevelsOneWeek.csv** file .
- d. By default, all columns are treated as text. We need to change that. Click the **Advanced** setting in the left side of the dialog.

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

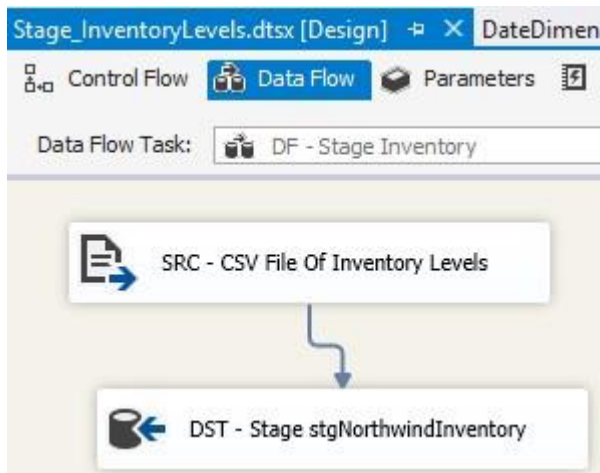


- f. Click **OK** to detect the data types.

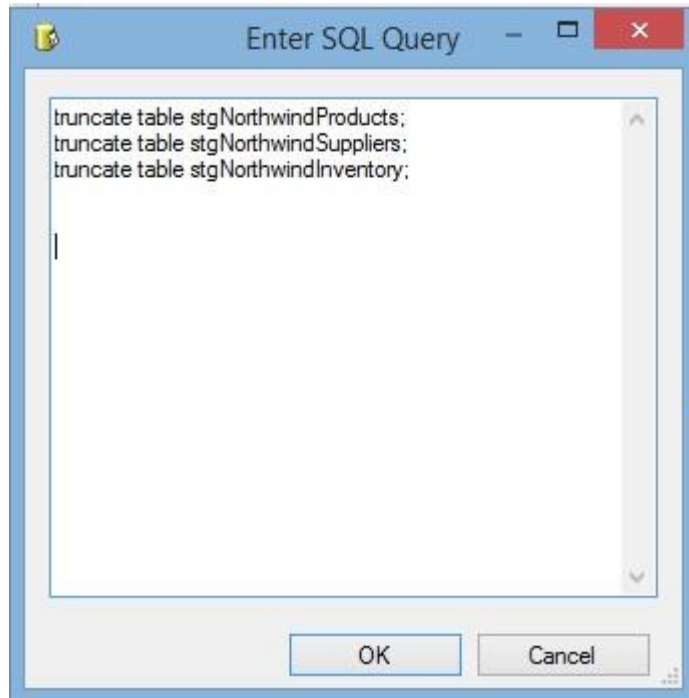
- g. Click **OK** to close the connection Manager
- h. Rename to → **SRC - CSV File of Inventory Levels**
3. Save your work.
4. Use the **Destination Assistant** to create the data target:
- a. Type: **SQL Server**
 - b. Connection Manager → **ist722_yournetid_stage**
 - c. Rename to → **DST - Stage 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 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 **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 the **Control Flow** tab.
9. Now add the **TRUNCATE TABLE** statement to the **SQL - Truncate Stage Tables** task. Double-click 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.

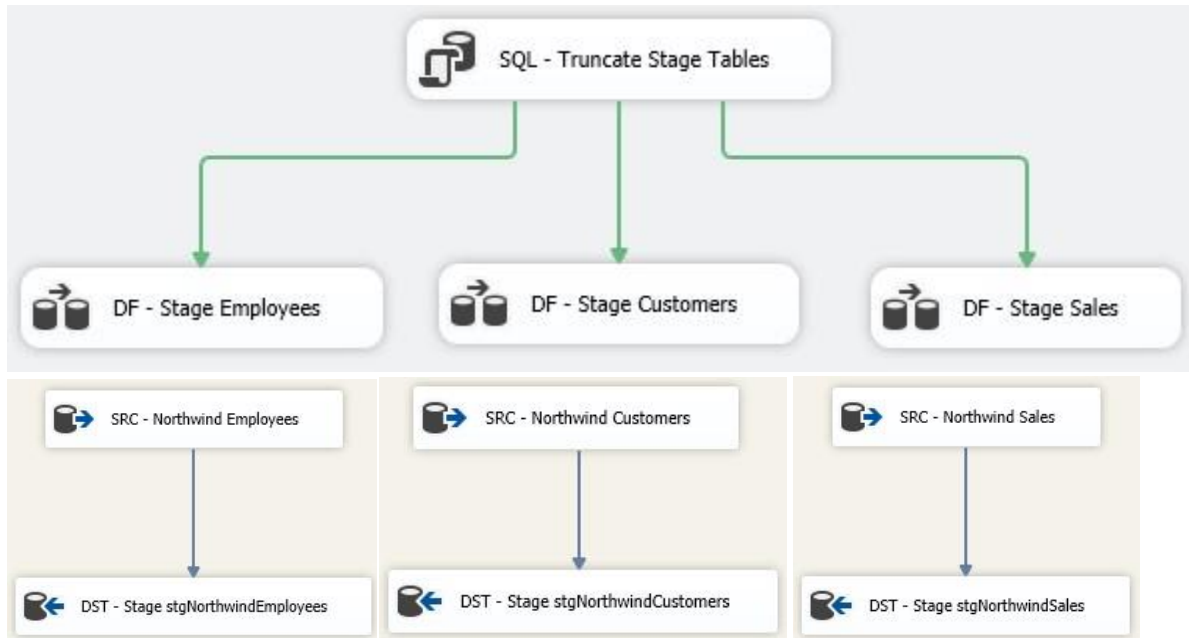
- f. Click **OK** to close the **Execute SQL Task Editor**.
- 10. Save your work. Time to test!
- 11. Execute the package by pressing **[F5]**. Troubleshoot any issues that may arise. Stop package execution with **[Shift]+[F5]**.
- 12. Verify there is data in the **stgNorthwindInventory** table in your **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 3: On Your Own

In this part, try to build out the ETL for Northwind Sales Reporting data mart. Here's a high-level outline of the process. Remember to 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 are 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 to figure out the source SQL queries. If you cannot, refer to the appendix in the next section.

Appendix A – Source SQL Queries

Source	SQL
Northwind Customers	SELECT CustomerID, CompanyName, ContactName, ContactTitle, City, Region, PostalCode, Country FROM Customers

Northwind Employees	<pre> 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 </pre>
Northwind Sales	<pre> 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 </pre>

Turning It In

Please turn in a Word document with your name, NetID, and date at the top. Paste a screenshot of your **Stage_Sales.dtsx** executing successfully (green checkmarks next to all the tasks) and make sure your name and NetID appear on the screenshot.

The SQL Server will be checked to verify that your data was successfully copied into your **ist722_yournetid_stage** database.