ETL SSIS

Exercise Handout

v2.1 ~ 28/3/2025

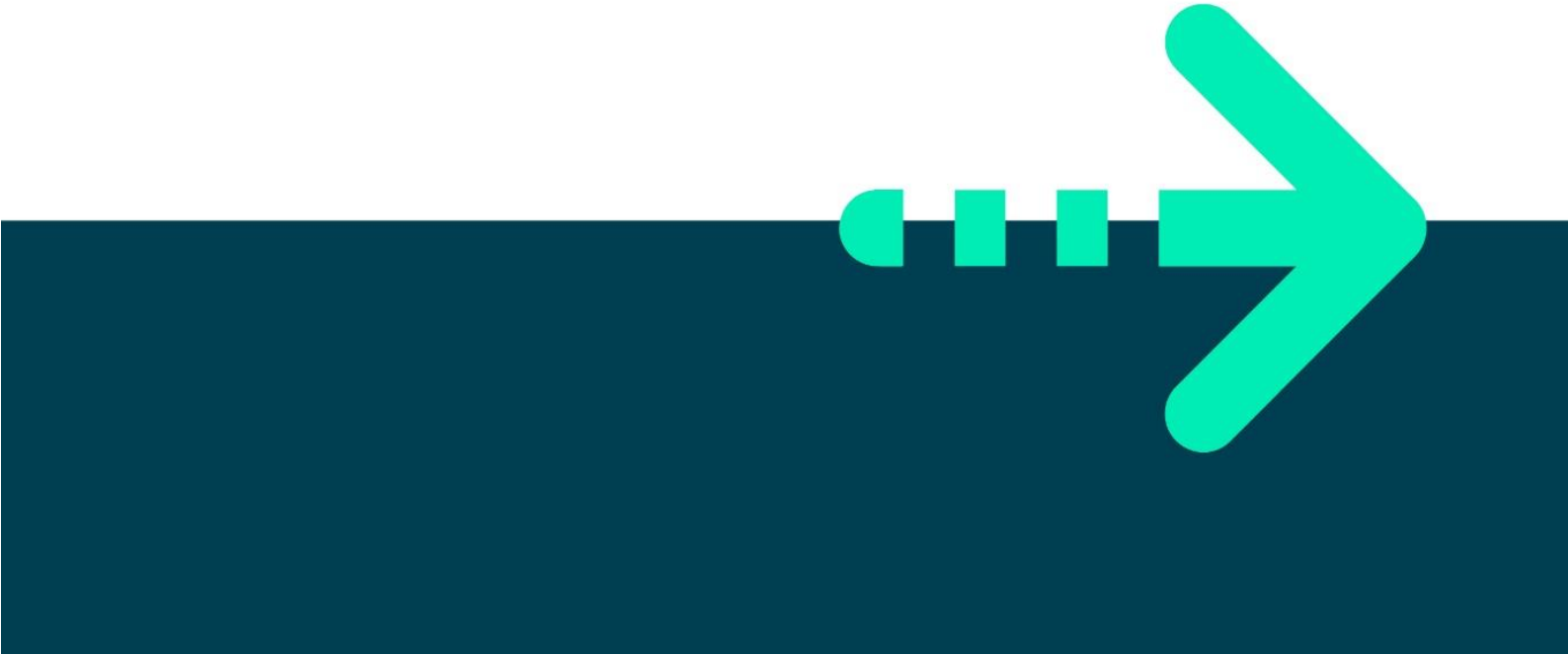


Table of Contents

M2 - Design and implement a data warehouse

Setup: Using SQL Server Management Studio

Open SQL Management Studio (SSMS). Connect with the following settings:

- Server name: **localhost**
- Authentication: **Windows Authentication**
- Encryption: **Optional**

LAB B: Creating the data warehouse

In this lab, you will:

- Create the data warehouse database.
- Add additional filegroups to the database.
- Create a partition function and partition scheme.

Exercise 1 – Create the data warehouse database

1. In SSMS open the file: `C:\Course Files\DESSIS\Mod02\`**`Mod2-LabB-Ex1.sql`**
2. **Highlight and execute step 1** to create the QASQLETLDW database.
3. If the object explorer is not visible, select the option Object Explorer from the View menu.
4. Expand the Database branch of the Object Explorer pane.
   The QASQLETLDW database should show. If it does not, right-click on the Database folder and click Refresh.
5. **Highlight and execute step 2** to create additional filegroups. These will be used for the dimension tables and the fact table.
6. **Highlight and execute step 3**, add files to the additional filegroups.

Exercise 2 – Create a Partition function and schema

1. In SSMS open the file: `C:\Course Files\DESSIS\Mod02\`**`Mod2-LabB-Ex2.sql`**
2. **Highlight and execute step 1** to create a partition function that will be used to split dates based on the vales 20210101 and 20220101.
3. **Highlight and execute step 2** to create a partition scheme. This will use the partition function DatePartitioningFunction and split the tables over the filegroups previously created: Facts2020Before, Facts2021, andFacts2022.

LAB C: Create a fact table

In this lab, you will create the fact tables using the created partition scheme.

Exercise 1 – Create the fact tables

1. In SSMS open the file: `C:\Course Files\DESSIS\Mod02\`**`Mod2-LabC-Ex1.sql`**
2. **Highlight and execute step 1** to create the FactInternetSales table.
3. **Highlight and execute step 2** to create the FactResellerSales table.

LAB D: Create dimension tables

In this lab, you will:

- Create tables DimProduct, DimGeography, DimCustomer, & DimReseller.
- Create a date table and populate it with data.
- Create the relationships between the tables.
- Create indexes.

Exercise 1 – Create the dimension tables

1. In SSMS open the file: `C:\Course Files\DESSIS\Mod02\`**`Mod2-LabD-Ex1.sql`**
2. **Execute the whole script** to create all four dimension tables.

Exercise 2 - Create a date table and add data to it

1. In SSMS open the file: `C:\Course Files\DESSIS\Mod02\`**`Mod2-LabD-Ex2.sql`**
2. **Highlight and execute step 1** to create the DimDate table.
3. **Highlight and execute step 2** to populate DimDate with data.
4. **Highlight and execute step 3** to view the DimDate table.

Exercise 3 – Create relationships between dimension tables and fact tables

1. Open the query file: `C:\Course Files\DESSIS\Mod02\`**`Mod2-LabD-Ex3.sql`**
2. Review the statements and execute them to create the relationships between the tables in the Data Warehouse.

Exercise 4 – Create the indexes

1. In SSMS open the file: `C:\Course Files\DESSIS\Mod02\`**`Mod2-LabD-Ex4.sql`**
2. Review the statements and execute them to create the necessary indexes.

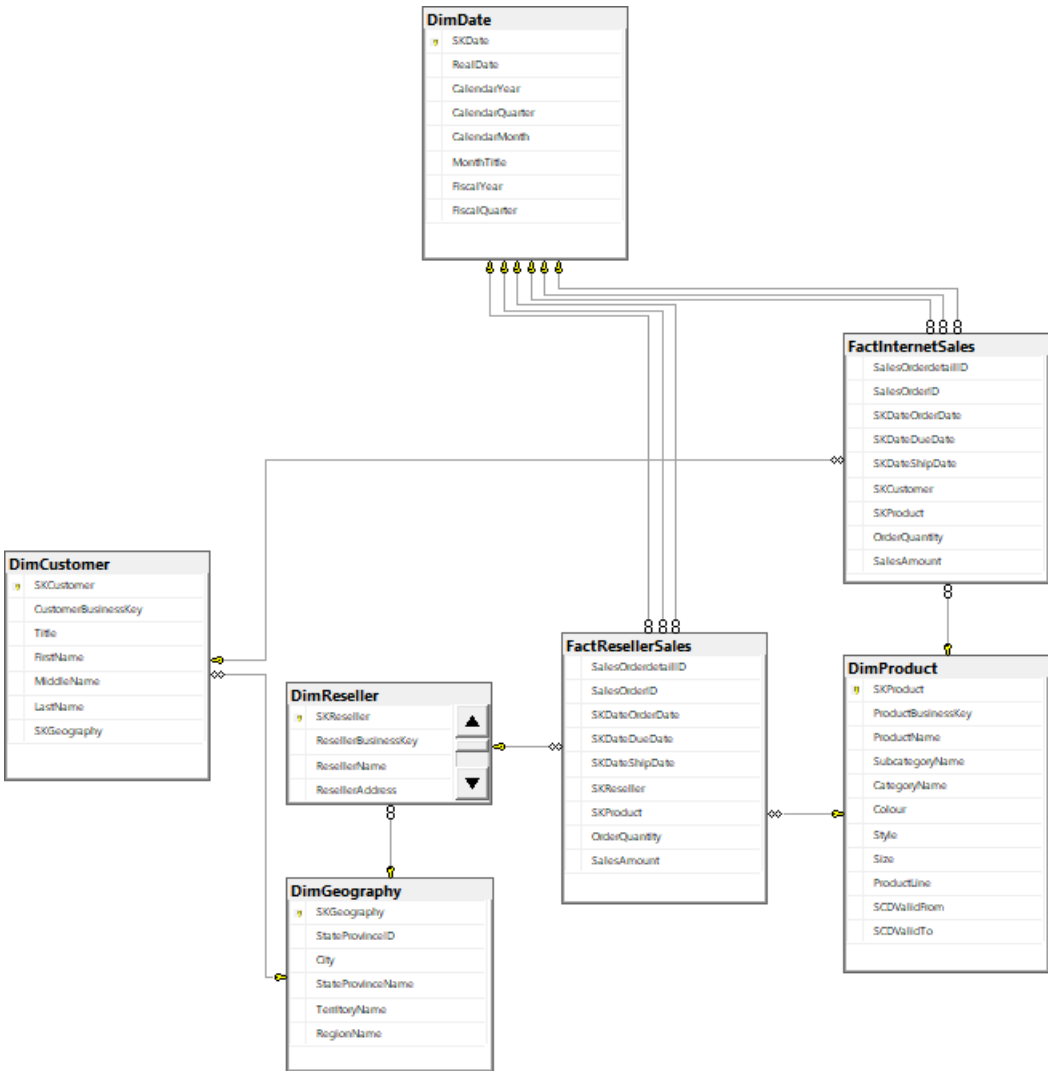
Exercise 5 – Create the database diagram

1. Grant authorisation to sa.

```
alter authorization on database::QASQLETLDW to SA
```

   This is needed to create the diagram ~ sa is the “system admn” user.

2. In Object Explorer – find the QASQLETLDW database and right click: Database Diagram → New Database Diagram. Then click Yes.
3. Select all the tables and click Add
4. Wait for the table to load. Then click Close
5. You should now see a database diagram for the tables you’ve just created.
6. You can rearrange them to look similar to this:

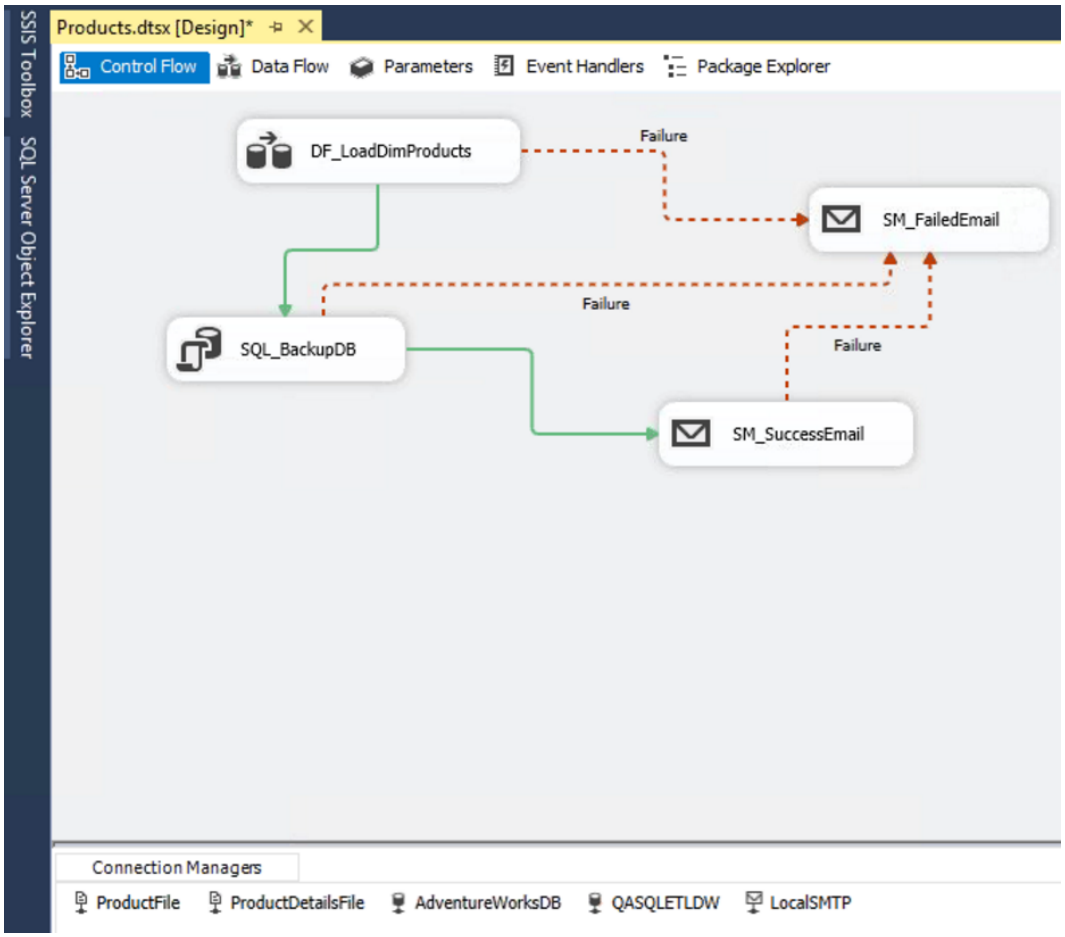


M3 - Implementing Control Flow in an SSIS Package

Lab A: Create a basic Control Flow task

In this lab we will:

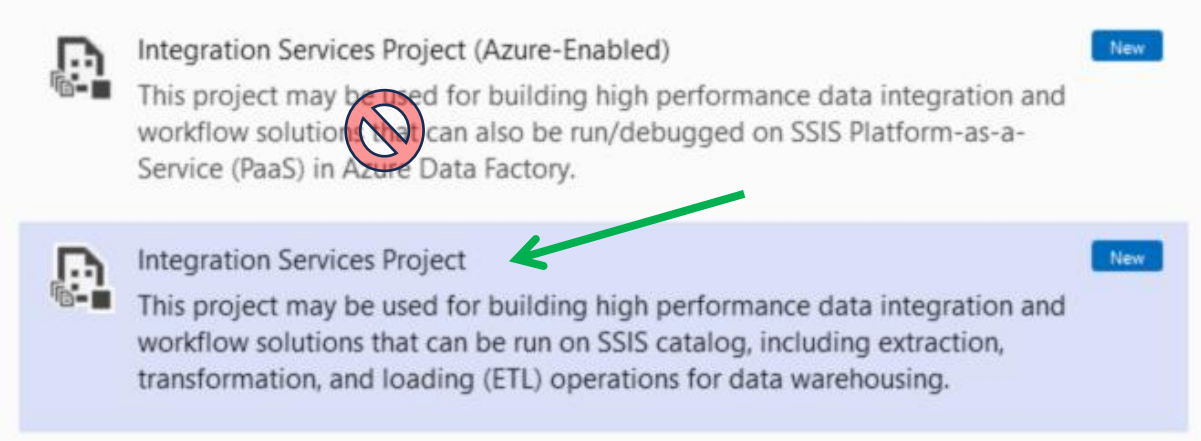
- Create a SSIS Package
- Add Connection managers
- Create a Control Flow Structure with Success/Failure paths



Exercise 1 – Create the SSIS Package

1. Open **Windows File Explorer** and navigate to the folder:
   `C:\Course Files\DESSIS\Mod03`
2. Examine the contents of the **Products** and **ProductDetails** folders.
3. Start **Visual Studio 2022**.
4. Under Get Started click: **Create a new Project**

5. In the dialog box select “Integrated Services Project” (**not** the Azure-enabled one) and click **Next** ~ (if you can’t see it on the list then search for it)



6. Enter the following, and then click **Next**
    a. Name: EtlLab
    b. Location: `C:\Course Files\DESSIS\Mod03\Mod3labA\Starter`
    c. Solution Name: EtlLab
7. Click **Create**
8. When the project is open select the **Solution Explorer** (on the right)

    Under SSIS Packages a file is created called: **Package.dtsx**

    Right-click on Package.dtsx and rename it to: **Products.dtsx**

Exercise 2 – Create Connection Managers

1. With the Products.dtsx package open, right-click in the connection manager section at the bottom of the window.
2. Select the option for: New Flat-file connection…
3. In the Flat File Connection Manager Editor set the following settings:
    a. Connection manager name: **ProductFile**
    b. Filename (via Browse):
    `C:\Course Files\DESSIS\Mod03\Products\Products.txt`
    c. Text qualifier: `"` (one double quote)
    d. **Untick** “Column names in the first data row”
4. Select the **Columns** tab and you should see: Bananas, Apples, Pears etc.
5. Select the **Advanced** tab and set the following settings:
    a. Column Name: **ProductName** (change from Column 0)

   b. DataType: string [DT_STR]
   c. OutputColumnWidth: 50

6. Select the **Preview** tab to confirm the contents of the Products text file and then click OK to save.
7. Repeat steps 1 to 6 for **ProductDetails** using the following settings:
   a. Name: **ProductDetailsFile**
   b. Filename (via browse):
      `C:\Course Files\DESSIS\Mod03\ProductDetails\productdetails1.txt`
   c. Text qualifier: `"` (one double quote).
   d. Untick “Column names in the first row”
   e. Advanced:
      i. Column 0: **ProductName** ~ string [DT_STR] ~ 50
      ii. Column 1: **Colour** ~ string [DT_STR] ~ 20
      iii. Column 2: **Cost** ~ four-byte signed integer [DT_I4]
8. Create an OLEDB Connection, by right-clicking in the connection manager section at the bottom of the window and select “New OLEDB connection…” Click “New”.
   a. Provider: **Native OLE DB\Microsoft OLE DB Driver for SQL Server**
   b. Server or file name: **localhost**
   c. Initial catalog: **Adventureworks**
   d. Click: Test Connection (it should succeed)
   e. Select OK twice to save
9. Right-click on the connection manager (localhost.Adventureworks) and rename it to **AdventureworksDB**
10. Create another OLEDB Connection, by right-clicking in the connection manager section and select “New OLEDB connection…” then click “New”.
    a. Server or file name: **localhost**
    b. Initial catalog: **QASQLETLDW**
    c. Click: Test Connection
    d. Select OK twice to save
11. Right-click on the connection manager (localhost. QASQLETLDW) and rename it to **QASQLETLDW**
12. Save your package.

Exercise 3 – Create a Control Flow Structure

1. Within the Products.dtsx package, select **the Control Flow** tab.
2. Drag on a:
    a. **Data Flow Task** and rename it to DF_LoadDimProducts.
    b. **Execute SQL Task** and rename to SQL_BackupDB
    c. **Send Mail Task** and rename it to SM_SuccessEmail
3. Copy and paste SM_SuccessEmail and rename it as SM_FailedEmail
4. Link the tasks as follows:
    a. From DF_LoadDimProducts to SQL_BackupDB with a **success** link
    b. From DF_LoadDimProducts to SM_FailedEmail with a **failure** link.
    c. From SQL_BackupDB to SM_SuccessEmail with a **success** link
    d. From SQL_BackupDB to SM_FailedEmail with a **failure** link
5. Right-click and **Edit** any of the links to SM_FailedEmail and select the Multiple Constraint value as: **Logical OR**
6. Right-click SQL_BackupDB and **edit**. In the Execute SQL Task Editor select the SQL Statement Connection as **QASQLETLDW**
7. Click the ellipses (…) in the SQL Statement row and enter:

```
backup database QASQLETLDW to disk = 'C:\SQLETLDB\qasqletldw.bak'
```

8. Click OK twice to close.
9. Right-click SM_FailedEmail and **edit**. Choose **Mail** in the page list.
10. In SmtpConnection create a <New connection…> with the following settings:
    a. Name: LocalSMTP
    b. SMTP server: localhost
    c. Select the Use Windows Authentication option
11. Click OK. Then enter the following:
    a. From: ssis@here.com
    b. To: admin@here.com
    c. Subject: Failed
12. Click OK to close.
13. Right-click SM_SuccessEmail and click **Edit…**

14. Choose mail in the page list and set the SMTP Connection as the LocalSMTP connection
15. Within the send mail dialog set the following settings:
    a. From: ssis@here.com
    b. To: admin@here.com
    c. Subject: Success
16. Click OK to close. Then save.
17. **Execute** the package by pressing **F5** or clicking the **green start button** at the top of the screen.

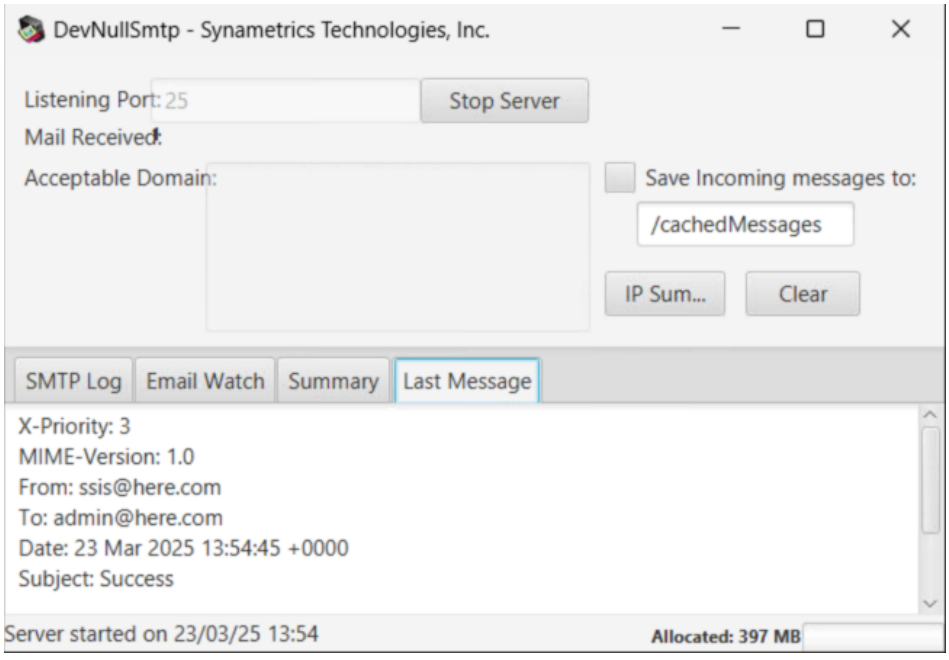
    After a few seconds, if the tasks have all completed successfully, they should all have a green tick on.

    A task in progress will have an animated amber circle (displayed for a few seconds on the execute SQL task while the database is being backed up).
18. Save all your files
19. Keep Visual Studio open for the next lab

Note:

If the SM_SuccessEmail step fails, then you may need to download and run a dummy SMTP server. Alternatively, you can just ignore the SMTP failure errors.

An example of a dummy SMTP server is: https://www.xeams.com/DevNull.htm



Lab B: Using Variables and Parameters

In this lab we will:

- Create a SSIS Variable
- Create a SSIS Parameter
- Combine the Parameter and Variable into and Expression

Exercise 1 - Create a variable

1. Using **Products.dtsx,** double click an empty area of the **Control Flow** surface
2. On the **View** menu, point to **Other Windows** and click **Variables**.
3. In the variables pane, click **Add Variable** and add a variable with the following properties:
    a. Name: filename *<~ Note: needs to be lowercase*
    b. Scope: Products
    c. Data Type: string
    d. Value: productdetails1.txt

Exercise 2 – Create a Parameter

1. In Solution Explorer, double click **Project.Params.**
2. Click **Add Parameter** and add a parameter with the following properties:
    a. Name: FolderPath
    b. Data Type: String
    c. Value: `C:\Course Files\DESSIS\Mod03\ProductDetails\` (**make sure** you include the final '\')
    d. Sensitive: False
    e. Required: True
    f. Description: Folder containing the Product Details files

3. Save all files and close the Project.Params [Design] window.

Exercise 3 – Create an Expression

1. On the Control Flow package and in the Connection Managers pane, click the **ProductDetailsFile** Connection Manager and then press **F4** to go to the Properties pane.
2. In the Properties pane, in the **Expressions** property box, click the ellipses (…) button.
3. Then in the **Property Expressions** Editor,
   and in the **Property** box select **ConnectionString**
   and in **Expression** box click the ellipses (…) button.
4. Expand the Variables and Parameters list
   and drag $Project::FolderPath to the Expression box
5. **In the** Expression **box,** type a **+**
   and then drag **User::filename** to the **Expression** box.
6. You should end up with the following expression

```
@[$Project::FolderPath] + @[ User::filename]
```

7. Click **Evaluate Expression**. The result should come back as:

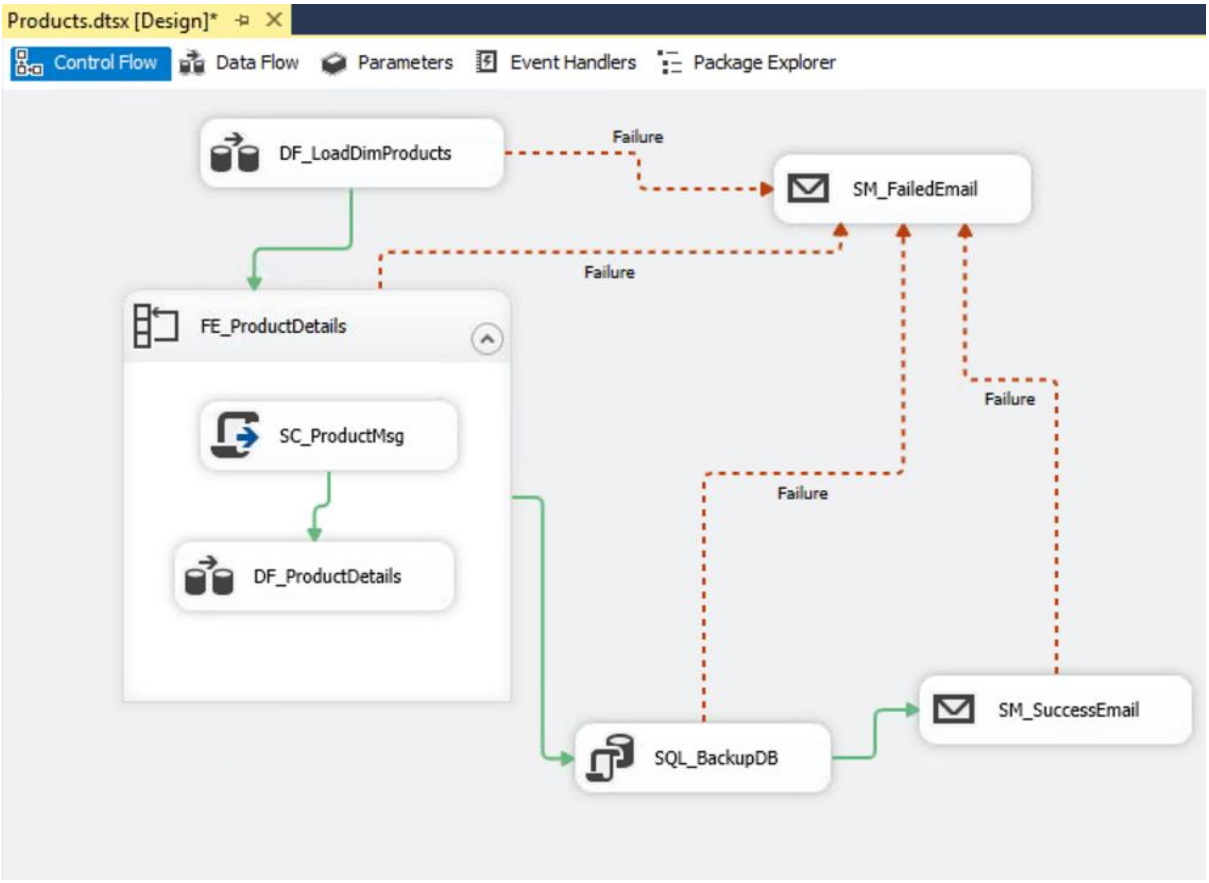
   ```
   C:\Course Files\DESSIS\Mod03\ProductDetails\productdetails1.txt
   ```

   Press OK twice.
8. We will make use of the expression in the next Lab.
9. Save all your files and keep Visual Studio open for the next lab.

Lab C: Using a Foreach loop

In this lab we will:

- Add a Foreach loop to the Products package
- Configure the foreach loop



Exercise 1 – Add a ForEach Loop containing a Dataflow Task

1. Within **Products.dtsx**, delete the green control flow line between **DF_LoadDimProducts** and **SQL_BackupDB**
2. Drag on a:
    a. **Foreach Loop Container** and rename it to FE_ProductDetails
    b. **Script Task** into FE_ProductDetails and rename it to SC_ProductMsg
    c. **Data Flow Task** into FE_ProductDetails and rename it to DF_ProductDetails.

3. Link the tasks as follows:
    a. From DF_DimLoadProducts to FE_ProductDetails with a **success** link.
    b. From FE_ProductDetails to SQL_BackupDB with a **success** link.
    c. From FE_ProductDetails to SM_FailedEmail with a **failure** link.
    d. From SC_ProductMsg to DF_ProductDetails with a **success** link
4. Save all your files

Exercise 2 – Configure the ForEach Loop

1. Right-click FE_ProductDetails and **edit**.
2. In the Foreach Loop editor, click the **Collection** page.
3. Choose the Foreach File Enumerator.
4. In the **Enumerator Configuration** set the following settings:
    a. Folder: `C:\Course Files\DESSIS\Mod03\ProductDetails`
    b. Files: *.txt
    c. Retrieve file name: Name and extension
5. Select the **Variable Mappings** page.
   In the Variables table from the dropdown select **User::filename**
6. Click OK to close
7. Right-click SC_ProductMsg and **edit**.
8. On the **script** tab click the ellipsis (…) in the ReadOnlyVariables section and tick **User::filename** and click OK and then click **Edit Script…**
9. In the C# script, scroll down to the section **//TODO: Add your code here**
10. Add the following code:

```
string fname = "Product File name: ";

fname += Dts.Variables["User::filename"].Value;

MessageBox.Show(fname);
```

Note: The code can be copied and pasted from:

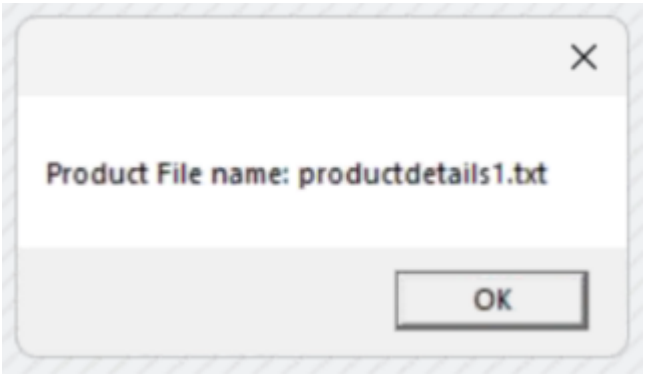
`C:\Course Files\DESSIS\Mod03\Mod3LabC\Starter\script.txt`

11. The result should look like this:

```
public void Main()
{
    // TODO: Add your code here

    string fname = "Product File name: ";

    fname += Dts.Variables["User::filename"].Value;

    MessageBox.Show(fname);

    Dts.TaskResult = (int)ScriptResults.Success;
}
```

12. Save your changes and use **File | Exit** to close the .cs file
13. You should now be back in Visual Studio 2022
    Click OK in the Script Task Editor
14. Save all your files.
15. **Execute** the package by pressing **F5** or clicking the green **start** button at the top of the screen.

    After a few seconds, the SC_ProductMsg task should have an animated amber circle, and a pop-up message should appear with the name of one of the files in the ProductDetails folder:



    This box should appear 3 times each with a different filename.
16. Click OK on each popup box.
17. When the task has finished click the red stop button to Stop Debugging and return to design view.
18. Save and Close the Solution from the File Menu

Lab D: Using transactions and checkpoints

In this lab you will:

- Create the Databases and Tables necessary for the lab
- Run the Transaction Package with an Error
- Create a Transaction
- Fix the Problem and Re-run the Package
- Set Checkpoints

Setup

Make sure that the **Distributed Transaction Coordinator** is running:

1. To open Services, on the Start menu, search for: **Services**
2. Scroll down and select: **Distributed Transaction Coordinator**
3. If the status is not set to **Running** then right click and select Start

Exercise 1 - Create the Databases and Tables Necessary for the Lab

1. Open SQL Management Studio (SSMS). When asked, connect to the localhost instance of SQL using Windows Authentication.
2. In SSMS open the query file:
   `C:\Course Files\DESSIS\Mod03\Mod3LabD\Starter\`**`Mod3-LabD-Ex1.sql`**
3. Highlight and execute step 1 to create the CustomerDB database.
4. Highlight and execute step 2 to create the Customers table.
5. Highlight and execute step 3 to insert data into the Customers table.
6. Highlight and execute step 4 to create the QAETLStagingDB database.
7. Highlight and execute step 5 to create the tables for the staging database.

8. Run the following query (use QAETLStagingDB):

```
select * from staging.Customers;
```

9. Notice that the table is empty.

Exercise 2 – Run the Transaction Package with an Error

1. Start Visual Studio 2022. Click: Continue without code …
2. Within Visual Studio 2022 go to File | Open | Project/Solution

3. Navigate to:
   `C:\Course Files\DESSIS\Mod03\Mod3LabD\Starter\Mod03LabDProject`
   and open **Mod03LabDProject.sln**
4. Open **transactions.dtsx** from the Solution Explorer.
5. Select **the Control Flow** tab.
6. Execute the package by pressing F5 or clicking the green start button at the top of the screen.

   Notice that the Dataflow task DF_CopyProducts succeeds ✓

   But the Script Task SC_UpdateSales fails ✗
   Reason: `DTS_W_MAXIMUMERRORCOUNTREACHED`
7. Stop the package running.
8. Go back to SSMS and run the following query (in QAETLStagingDB):

```
select * from staging.Customers;
```

9. Notice that the table now has entries

   But the TotalSales column are all set to 0.00.

Exercise 3 – Create a Transaction in SSIS

1. In SSMS type the following query to empty the table:

```
truncate table staging.customers;
```

2. In Visual Studio 2022, click on the Control Flow screen and go to the **Properties** pane.
3. Set Transactions → **TransactionOption** to **Required**.
4. Click on DF_CopyProducts and go to the **Properties** pane.
5. Set: Execution → Set **FailPackageOnFailure** to **True**
   and: Transactions → **TransactionOption** to **Supported**
6. Make these same changes to: SC_UpdateSales
7. Execute the package by clicking the green start button or pressing F5

   Notice the Dataflow task DF_CopyProducts succeeds ✓

   But the Script Task SC_UpdateSales fails ✗
8. Stop the package executing and save all files.
9. Go back to SSMS and re-run the following query:

```
select * from staging.Customers;
```

10. Notice that the table is now empty!
    The whole of the package was rolled back.

Exercise 4 – Fix the problem and re-run the Package

1. In Visual Studio 2022, **edit** SC_UpdateSales.
2. Click in the **SQLStatement** section and click on the ellipsis (…) button.

   Notice the calculation includes a divide by TotalSales which is initialy set to 0

   The step is failing, therefore, with a divide by zero error.
3. Change the statement to the following:

```
update staging.Customers
set TotalSales = TotalSales + 10;
```

4. Click OK twice and rerun the package

   This time both steps should succeed.
5. Stop the package executing and save all files.
6. Go back to SSMS and re-run the following query:

```
select * from staging.Customers;
```

7. Notice TotalSales is now set to 10.00

Exercise 5 – Set Checkpoints

1. In Visual Studio 2022, from Solution Explorer, open the **Checkpoints.dtsx** package.
2. Click on the Control Flow screen and go to the **properties** pane.
3. Set: Execution → **MaximumErrorCount to 100**
4. Execute the package by pressing F5 or clicking the green start button at the top of the screen.

   Notice that the SQL_ClearCustomersStaging task and the Dataflow task DF_CopyProducts both succeed ✓

   But the Script Task SC_UpdateSales fails ✗

5. Stop debugging

6. Click on the Control Flow screen and go to the **Properties** pane.
7. In the properties pane under Checkpoints, set the following settings:
    a. **CheckpointFileName** :
    `C:\Course Files\DESSIS\Mod03\checkpoint.chk`
    b. **CheckpointUsage**: IfExists
    c. **SaveCheckpoints**: True
8. Set the property Execution → **FailPackageOnFailure** to **True** for each of the 4 tasks.
9. Execute the package. As before, SC_UpdateSales fails ❌
10. Stop executing the package.
11. Using Windows File Explorer, go to: `C:\Course Files\DESSIS\Mod03`

    Notice the checkpoint.chk file.
12. Back in Visual Studio 2022, **edit** SC_UpdateSales.
13. Click in the **SQLStatement** section and click on the ellipsis (…) button.

    Notice the calculation includes a divide by TotalSales which is initial set to 0.

    The step is failing, therefore, with a divide by zero error.
14. Change the statement to the following:

```
update staging.Customers
set TotalSales = TotalSales + 10;
```

15. Click OK and rerun the package.

    Notice that the package resumes from SC_UpdateSales which succeeds. ✓

    The SC_Messagebox task should also succeed ✓
16. Click OK on the "All done!!" message box.
17. Stop the package executing & then re-run it.

    This time all the steps should run successfully.
18. Stop the package executing and save all files.
19. Using Windows File Explorer, go to: `C:\Course Files\DESSIS\Mod03`

    Notice the **checkpoint.chk** file is no longer there!
20. In Visual Studio 2022, save and close the solution.

M4 - Implement ETL with SSIS

Lab A: Exporting data from SQL Server (SSMS)

In this lab you will:

- Use the SQL Server Import/Export wizard to export data to a CSV file
- Use the `bcp` command line tool.

Exercise 1 – Use the Import/Export Wizard

1. Open SQL Management Studio (SSMS). When asked, connect to the localhost instance of SQL using Windows Authentication.
2. Expand the **Database** branch of the Object Explorer pane and expand the **Adventureworks** database and then **Tables**.
3. Right-click **Person.Countryregion** and **select Top 1000 Rows**. Examine the results and notice that there are 3 columns: CountryRegionCode, Name and ModifiedDate.
4. Right-click the **Adventureworks** database, go to **Tasks** and pick **Export Data.**
5. On the **Data Source** page choose the following settings
    a. Data Source: Microsoft OLE DB Driver for SQL Server
    b. In Properties:
        1. Enter a server name: localhost
        2. Enter information to log on: Windows Authentication
        3. Select the database: Adventureworks

    Click OK then Next
6. On the **Destination** page choose the following settings:
    a. Destination: Flat File Destination
    b. File name:
    `C:\Course Files\DESSIS\Mod04\Mod4labA\CountryRegions.csv`
    c. Format: Delimited
    d. Text qualifier: `"` (one double quote)
    e. Column names in the first data row: ticked

    Click Next

7. On the Specify Table Copy or Query choose:
   Write a query to specify the data to transfer and click Next
8. On the Source Query page type the following:

```
SELECT CountryRegionCode, Name FROM Person.CountryRegion
```

   Click Next
9. On the Configure Flat File Destination page:

   Click **Edit Mappings** and have a look at the settings.

   CountryRegionCode should be mapped to CountryRegionCode
   Name should be mapped to Name

10. Click OK then click **Preview** to see that data and then click **Next**.
11. On the Save and Run Package page

    Make sure **Run immediately** is selected

    Click **Next** and **Finish**.
12. When the extraction has completed, close the wizard.
13. Using Windows File Explorer, navigate to the folder:

    `C:\Course Files\DESSIS\Mod04\Mod4labA`

    and open **CountryRegions.csv** with notepad.
14. The file should contain 239 records.
15. Close notepad.

Exercise 2 – Use the bcp command line tool

1. Open a PowerShell to get to a Windows command line
2. Run the following command in the PS command line:

```
bcp "SELECT * FROM AdventureWorks.Person.CountryRegion"
queryout "C:\backup\CountryRegions.csv" -c "-t," -T -S
"localhost"
```

3. Open the file `C:\backup\Countryregions.csv` with notepad.

   Note: The file contains 238 rows (not 239) as bcp does not store a header.

Lab B: Profiling a data source

In this lab we will learn how to profile data sources.

Exercise 1 – Use the Data Profiling Task

1. Start Visual Studio 2022. Click: Continue without code …
2. Within Visual Studio 2022 go to File | Open | Project/Solution
3. Navigate to: `C:\Course Files\DESSIS\Mod04\Mod4LabB\Starter\EtlLab` and open **EtlLab.sln**
4. In the **Solution Explorer** create a **New SSIS Package**.
5. Rename Package1.dtsx to **dataprofile.dtsx** and then double click dataprofile.dtsx to open it.
6. Within the **dataprofile.dtsx** package, right-click in the connection manager section at the bottom of the window.
7. Create a **New ADO.Net Connection**.
8. Click "New".
    a. Server name: localhost
    b. Authentication: Windows Authentication
    Encrypt: Optional (False)
    c. Select of enter a database name: Adventureworks
    d. Click: Test Connection
    e. Select OK twice to save
9. Add a **Data Profiling Task** from the Toolbox onto the Control Flow surface

    Then right click and click **edit**
10. On the **General** tab:
    a. Click in the **Destination** section and the **Destination** field and pick **New File connection** from the drop-down list.
    b. In the **File Connection Manager Editor** choose the following settings and click OK:
    c. Usage type: Create file
    d. File: `C:\Course Files\DESSIS\Mod04\profile.xml`
    e. Set **OverWriteDestination** to **True**.
11. Click **Profile Requests** tab and click: **Column Statistics Profile Request**
    a. In the **Request Properties** section below set the following settings:

i. Connectionmanager: localhost.AdventureWorks1
ii. TableOrView: [Sales].[SalesOrderHeader]
iii. Column: OrderDate

12. In **Profile Requests** tab click: **Column Length Distribution Profile Request**
    a. In the Request Properties section set the following settings:
        i. Connectionmanager: localhost.AdventureWorks1
        ii. TableOrView: [Production].[Product]
        iii. Column: Name
13. In **Profile Requests** tab click: **Column Null Ratio Profile Request**
    a. In the Request Properties section set the following settings:
        i. Connectionmanager: localhost.AdventureWorks1
        ii. TableOrView: [Production].[Product]
        iii. Column: Color
14. Click OK.
15. Execute the package by pressing F5 or clicking the green start button at the top of the screen.
16. When the Data Profiling Task has completed leave the package still running.
17. Right-click the Data Profiling Task and **edit.** Then click **Open Profile Viewer**.
18. Click on: **Column Length Distribution Profiles**

    Examine the different lengths of the **Name** column.

    Double-click on 1 of the lengths and examine range of the rows returned.
19. Click on: **Column Null Ratio Profiles**

    Examine the **Null Count.**

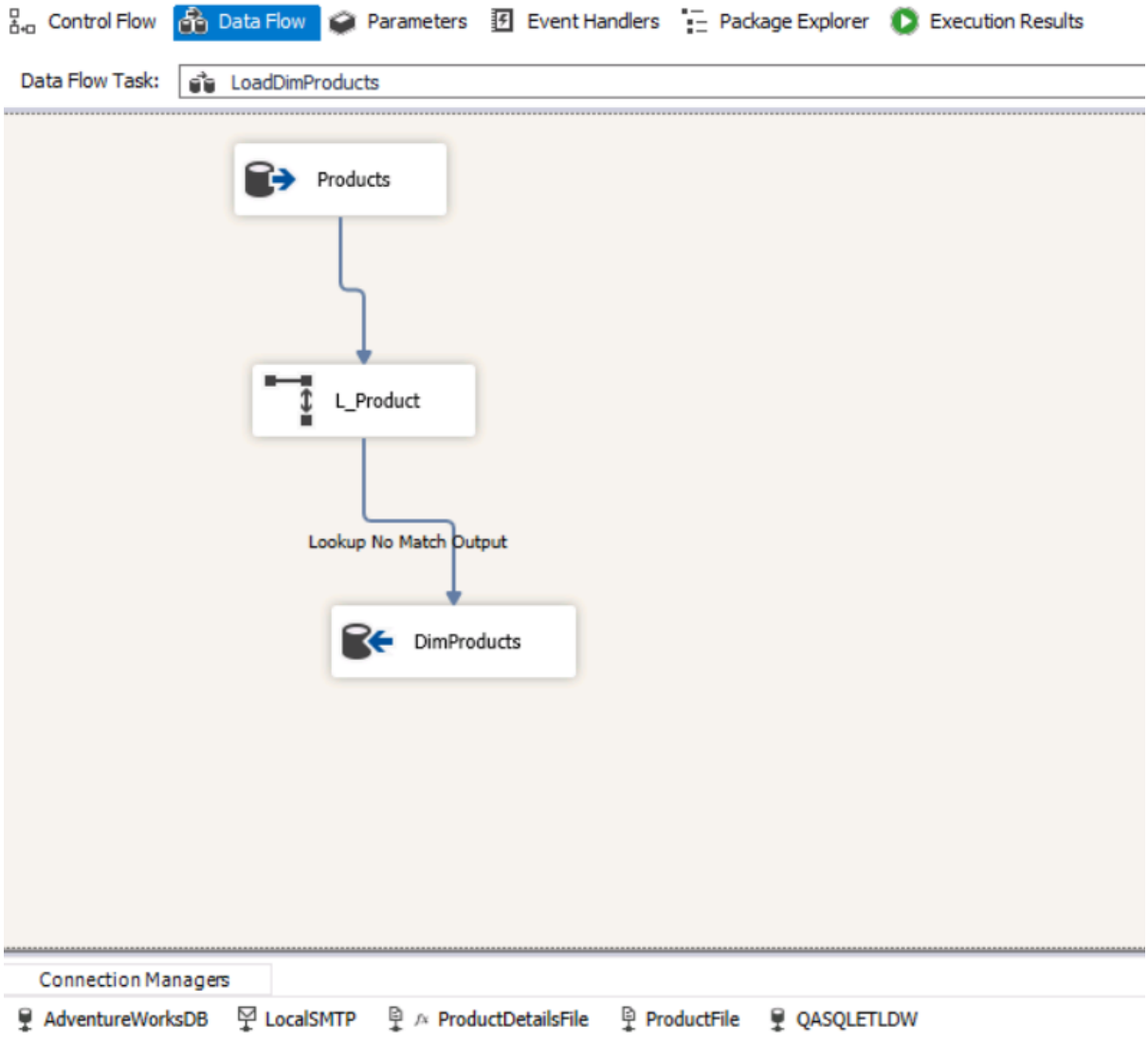
    Double-click on the **Color** column to see what rows have NULL for color.
20. Click on **Column Statistics Profile**

    Examine the minimum and maximum vales for the **OrderDate** column.
21. Close the **Data Profile Viewer** and close the **Data Profiling Task Editor**
22. Stop the package execution.
23. Save all the files.

Lab C: Implement a data flow task with transformations

In this lab we will:

- Implement data flow task to transfer product information between the Adventureworks database and the QASQLETLDW Data Warehouse.
- Implement a Lookup transformation
- Add a row into the products table in Adventureworks to see the effect



Exercise 1- Implement a Data Flow Task and a Data Source

1. Within Visual Studio 2022 go to File | Open | Project/Solution
2. Navigate to: `C:\Course Files\DESSIS\Mod04\Mod4LabC\Starter\EtlLab` and open **EtlLab.sln**
3. From the Solution Explorer open **Products.dtsx**.

4. Within the Products package, select the **Data Flow** tab, and in the Data Flow Task dropdown select **DF_LoadDimProducts.**
5. Drag the **OLE DB Source** task from the SSIS Toolbox onto the design pane. Rename as **Products** and then **Edit**.
6. Select **AdventureworksDB** from the dropdown list of **Connection Managers**.
7. Select **SQL Command** from the Data Access Mode dropdown
8. **Browse** and select the following file:
   `C:\Course Files\DESSIS\Mod04\Mod4LabC\Starter\`**`Mod4-LabC-Ex1.sql`**
   This query will retrieve category, subcategory and product details from the Adventureworks database.
9. Copy and paste the query into the box: **SQL command text**
10. **Preview** the data and then close the preview
11. Review the **Columns** tab to see the available column names
12. Click OK.

Exercise 2 - Add a Lookup Transformation

1. Drag a Common > **Lookup** transformation onto the design pane and place it directly underneath the Products source.
2. Rename the Lookup as **L_Product.**
3. Connect the **blue** arrow from the Products source to the L_Product transformation, and then **edit** L_Product.
4. On the **specify how to handle rows with no matching entries** dropdown, select **Redirect rows to no match output.**
   We only want to insert *new* rows into our Data Warehouse.
5. Select the **Connection** tab.
   Select the **QASQLETLDW** as the connection manager.
6. In the **Use results of an SQL query**, type the following code:

```
SELECT ProductBusinessKey from dbo.DimProduct
```

7. Click **Preview** to check the results.
   You should see that the query will return no results at the moment.
8. Select the **Columns** tab and drag the **ProductID** column onto **ProductBusinessKey** in the lookup table. A line will be drawn between the two columns. This will set up the lookup criteria.
9. Click OK.

Exercise 3 – Add a Destination

1. Drag the **OLE DB Destination** task from the SSIS Toolbox onto the design pane and rename it as **DimProducts**
2. Drag the **blue** arrow from L_Products to DimProducts. Select **Lookup No Match Output** on the **Output** section i.e., only new rows will flow through into the destination.
3. Click OK.
4. Edit the DimProducts destination.
5. Change the **OLEDB Connection Manager** to **QASQLETLDW**
6. On the **Name of the table or the view** select the table **dbo.DimProduct**.
7. Select the **Mappings** tab.
8. In the **Available Input Columns** list drag **ProductID** to the **ProductBusinessKey** column in the **Available Destination Columns** list.
9. In the second half of the Mapping tab, select input column **Color** (currently listed as "ignore") to the Destination column **Colour** to match them up.
10. Click OK.
11. Save all your files.
12. Execute the package by pressing F5 or clicking the green start button at the top of the screen.
13. Click on the data flow tab and notice the number of rows reported on the link between **Products** and **L_Product**, and the number of rows reported on the **Lookup No Match Output** link. There should be the same number of rows in each case.
14. Click the red stop button to return to design view.

Exercise 4 – Adding an Additional Row into the Products Table to test the Lookup

1. Using SSMS, open the query file `C:\Course Files\DESSIS\Mod04\Mod4LabC\Starter\`**`Mod4-LabC-Ex2.sql`**
2. Execute the code to add the additional row to the Production.Product table.
3. Back in Visual Studio 2022, execute the package by pressing F5 or clicking the green start button at the top of the screen.
4. Click on the data flow tab and notice the number of rows reported on the Lookup No Match Output link. You should see 1 row reported.
5. Click the red stop button to return to design view.
6. Save all files.

M5 - Debugging & Troubleshooting SSIS Packages

Lab A: Debugging an SSIS Package

In this Lab you will learn how to:

- Debug a Control Flow
- Debug a Script Task
- Use a Data View to debug a Data Flow

Exercise 1 – Debug a Control Flow

1. Start SQL Server Management Studio (SSMS) and open `C:\Course Files\DESSIS\Mod05\Mod5LabA\Starter\`**`Mod5-LabA-Ex1.sql`**

   Run the script to create the staging database: **QAETLStagingDB**
2. Start Visual Studio 2022. Click: Continue without code …
3. Within Visual Studio 2022 go to File | Open | Project/Solution
4. Navigate to: `C:\Course Files\DESSIS\Mod05\Mod5LabA\Starter\EtlLab` and open **EtlLab.sln**
5. Open **Products.dtsx** from the Solution Explorer.
6. Right-click **DF_ProductDetails** and **Edit Breakpoints**.
7. Examine the options for when a breakpoint will occur and tick:
   Break when the container receives the OnPreExecute event.
   Leave the **Hit Count Type** on **Always** and click OK.
8. In Visual Studio, execute the package by pressing F5 or clicking the green start button at the top of the screen.
9. Press OK on any pop-up Message box.
10. The execution will break when it reaches DF_ProductDetails task.
11. From the Debug menu choose Windows | Locals.
12. In the Locals windows, expand **Variables** and notice the value set for **User::filename**.
13. Right-click User::filename and **Add Watch**.
    You will see that the variable is added to a Watch window.
14. Click the green Continue button or F5 to continue the execution.

15. When the message box appears (it may be minimised on the Task bar) press OK.
16. Notice how user::filename has changed in the Watch window.
17. Use the red stop button to stop the package execution.
18. Click DF_ProductDetails and expand the **Debug** menu.
    Select **Delete All Breakpoints**.

Exercise 2 – Debug a Script Task

1. Right-click **SC_ProductMsg** and **Edit.**
2. In the Script Task Editor, click **Edit Script**.
3. When **ScriptMain.cs** open right-click the line with **MessageBox.Show(fname)** on it and choose **Breakpoint | Insert Breakpoint**.
4. Select **File | Exit** to close scriptmain.cs. Press **OK** in the Script Task Editor.
5. In Visual Studio, **execute** the package by pressing **F5** or clicking the green **start** button at the top of the screen.
6. After several seconds, **scriptmain.cs** will open and you will see the execution has broken at the message box.
7. Click on the Debug > Windows > **Locals** window and see the value of **fname**
8. Click the red **stop** button and click **OK** on the message box to continue.
9. Click the red **stop** button again to stop the execution.
10. Right-click SC_ProductMsg. Choose **Edit**. **Edit Script** and delete the breakpoint on the **MessageBox** line in **Scriptmain.cs**
11. Exit Scriptmain.cs and click **OK** the Script Task Editor.

Exercise 3 - Use a Data View to debug a Data Flow

1. Right-click DF_LoadDimProducts and choose **Edit** to go to the Data Flow tab.
2. Right-click the blue line between **Products** and **L_Product** and **Enable Data Viewer**.
3. Repeat for the blue line between **L_Product** and **DimProducts**.
4. Using SSMS, open the file:
   `C:\Course Files\DESSIS\Mod05\Mod5LabA\Starter\`**`Mod5-LabA-Ex3.sql`**
5. **Execute** the code to add the additional row to the **Production.Product** table.

6. In Visual Studio 2022, execute the package by pressing F5 or clicking the green start button at the top of the screen.
7. The first Data Viewer will open. You should see over 200 rows of data.
8. Close the popup window with the 200 products

9. The second Data Viewer will open. You should see the extra row that you added to the Production.Product table.
10. Close the popup window with the 1 new product

11. Click OK on the 3 popups (they may be minimised on the taskbar)
12. Press the red stop button to stop the execution of the package.
13. Save all your files and close Visual Studio 2022.

Lab B: Configuring event logging

In this lab you will learn how to implement SSIS Logging.

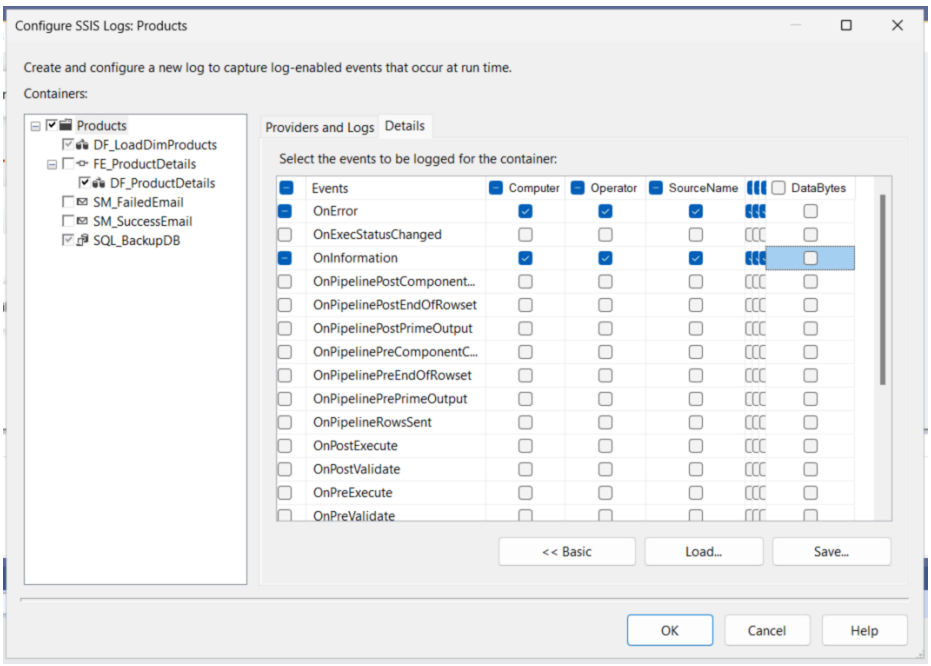
Exercise 1 – Examine the DF_ProductDetails Data Flow Task

1. Start Visual Studio 2022. Click: Continue without code …
2. Within Visual Studio 2022 go to File | Open | Project/Solution
3. Navigate to: `C:\Course Files\DESSIS\Mod05\Mod5LabB\Starter\EtlLab` and open **EtlLab.sln**
4. Open **Products.dtsx** from the Solution Explorer. This is a slightly modified version of the solution with the Script Task removed.
5. Right-click DF_ProductDetails and **Edit**.
6. Examine the Data Flow. This data flow opens a **productdetails** file (based on the Foreach loop variable set in Module 3.
7. Edit the **Cost Calculation** derived column task. This task creates a new column called **ListPrice** that multiples the **Cost** column by 1.5.
8. Close the Derived column editor
9. Edit the **Load Staging Table** destination. Notice that the data is going to be loaded into the **staging.Products** table in the **QAETLStagingDB**.
10. Close the OLE DB Destination editor.
11. In Visual Studio, execute the package by pressing F5.
12. Notice that DF_ProductDetails fails ❌
    Reason: `DTS_W_MAXIMUMERRORCOUNTREACHED`
13. Click the red stop button to return to design view.

Exercise 2 – Configure SSIS Logging

1. On the Visual Studio 2022 menu click **Extensions** then **SSIS**.
   On the SSIS menu click **Logging…**
2. In the Configure SSIS Logs dialog box: select **SSIS log provider for Windows Event Log** from the **Provider Type** drop-down. Click **Add**.
3. In the Configure SSIS Logs dialog box, select **SSIS log provider for SQL Server** from the **Provider Type** drop-down. Click **Add**.
4. In the **Configuration** column for the **SSIS log provider for SQL Server** select **QAETLStagingDB** from the drop-down list.
5. In the **Containers** section, select **Products** and tick the checkbox and then select and tick **SSIS log provider for Windows Event Log** checkbox.

6. On the **Details** tab, select the **OnError** and **OnInformation** checkboxes.
7. In the **Containers** section, **clear** the **SM_FailedEmail** and **SM_SuccessEmail** checkboxes.
8. In the **Containers** section, **clear** the **FE_ProductDetails** checkbox and then expand FE_ProductDetails.
9. Select **DF_ProductDetails** checkbox *twice* to select it (make sure there is a tick there) and then select the **SSIS log provider for SQL Server** checkbox in the Providers and Logs section.
10. On the **Details** tab, select the **OnError** and **OnInformation** checkboxes.
11. Click the **Advanced** button and uncheck **Databytes** for both events.



12. Click OK.
13. Save all files.
14. In Visual Studio, **execute** the package by pressing **F5** or clicking the green **start** button at the top of the screen.
15. Note: the package fails ❌ ~ But we will not fix the problem yet.

Exercise 3 – View the Logs

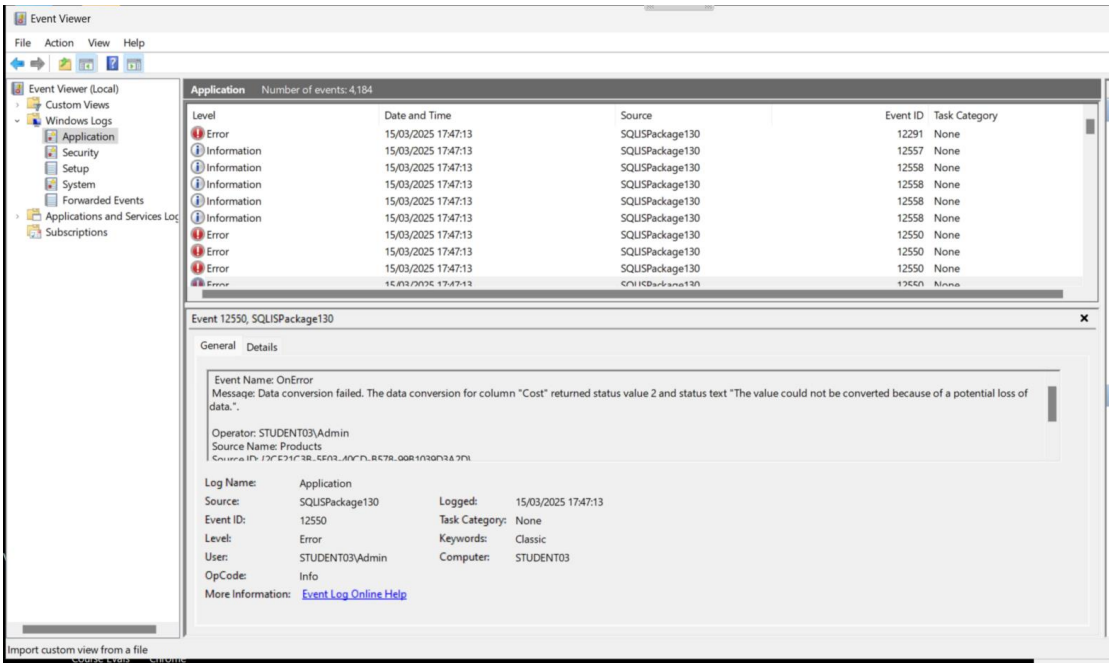
1. When the package has stopped executing, open **Event Viewer** from the Windows **Start** menu.

   Note: This is a MS Windows program

2. Expand **Windows Logs** and select the **Application** log.

3. Notice several events with a source of **SQLISPackage130**.

4. Examine the red error messages.



Can you determine what the error is?

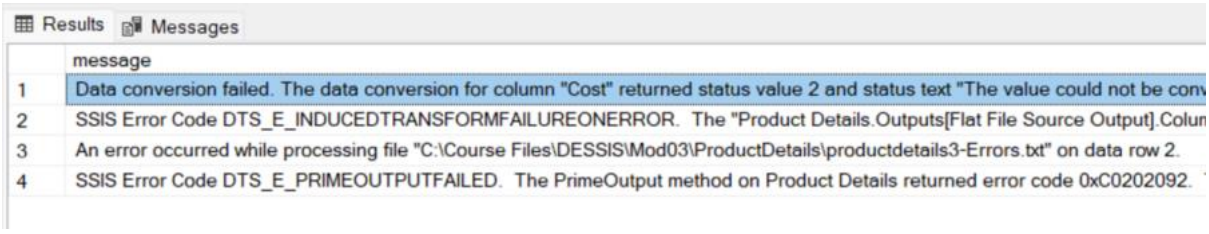
```
Message: Data conversion failed. The data conversion for column
"Cost" returned status value 2 and status text "The value could
not be converted because of a potential loss of data.".
```

5. Close the Event Viewer.

6. Open SQL Server Management Studio (SSMS) and run the following code:

```
use QAETLStagingDB

select starttime, message
from dbo.sysssislog
where event = 'OnError';
```

7. Examine the OnError events.

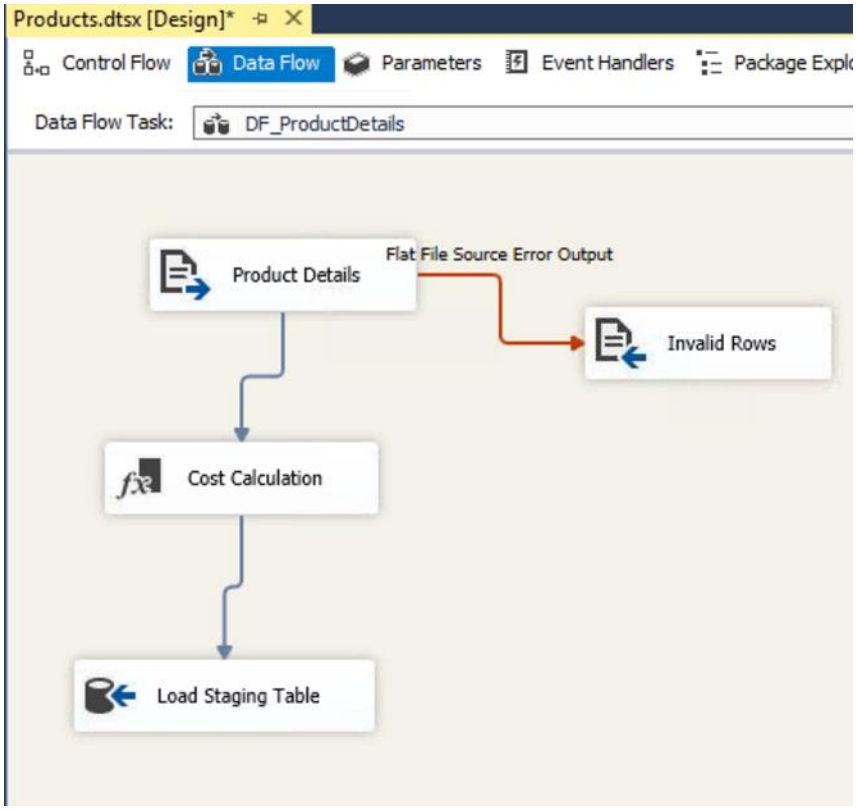


8. In Visual Studio, click the red **stop** button, if you haven't already done so. We will not fix the problem yet.

9. Keep Visual Studio open for the next Lab.

Lab C: Implement Error Handling

In this Lab we will configure Error Handling.



Exercise 1 – Implement Error Handling

1. Click on the **Data Flow** tab and make sure the Data Flow task is **DF_ProductDetails**.
2. Right-click **Product Details** and **Edit**.
3. Click the **Error Output** tab.
4. For each of **ProductName, Colour and Cost**, change the **Error** drop-down to **Redirect Row** and click **OK**.
5. Click OK.
6. In the SSIS Toolbox , in the **Other Destinations** section drag a **Flat File Destination** to just the right of **Product Details**.
7. Rename the Flat File Destination as **Invalid Rows**
8. Drag the **Red** arrow from **Product Details** to **Invalid Rows**
9. Ensure all three columns have an Error of **Redirect Rows.** Click **OK**.
10. Edit **Invalid Rows**.
11. Click **New** to create a new connection manager.

12. Choose **Delimited** and click **OK**.
13. Use the following settings:
    a. Connection manager name: InvalidRows
    b. Filename (via Browse):
    `C:\Course Files\DESSIS\Mod05\Mod5LabC\invalid.csv`
    c. Text qualifier: `"` (one double quote)
14. Click **OK**.
15. Click the **Mapping** tab and examine the input and destination columns.
16. Click **OK**.
17. In Visual Studio, **execute** the package by pressing **F5** or clicking the green **start** button at the top of the screen.
18. You should see that the Data Flow task now works. You should also see that there is 1 row sent to the **Invalid Rows** task.
19. Click the red **stop** to stop execution of the package.
20. Using Notepad open:
    `C:\Course Files\DESSIS\Mod05\Mod5LabC\invalid.csv`
21. Notice that the 3rd field (which is **cost**) has the word “fresh” rather than a numeric value.
22. Return to Visual Studio 2022.
23. Save all files and close Visual Studio.

M6 - Incrementing an Incremental ETL Process

Setup: Make sure that your machine name matches SSMS

1. IN SSMS run the file: `C:\Course Files\DESSIS\MachineName.sql`
2. To restart SQL Server - In the Object Explorer: right click localhost → Restart

Lab A: Using Change Data Capture (CDC)

In this lab we will:

- Create the databases and tables for the lab
- Enable Change Data Capture (CDC)
- Configure initial data extraction
- Modify data in the source table
- Configure an incremental CDC package

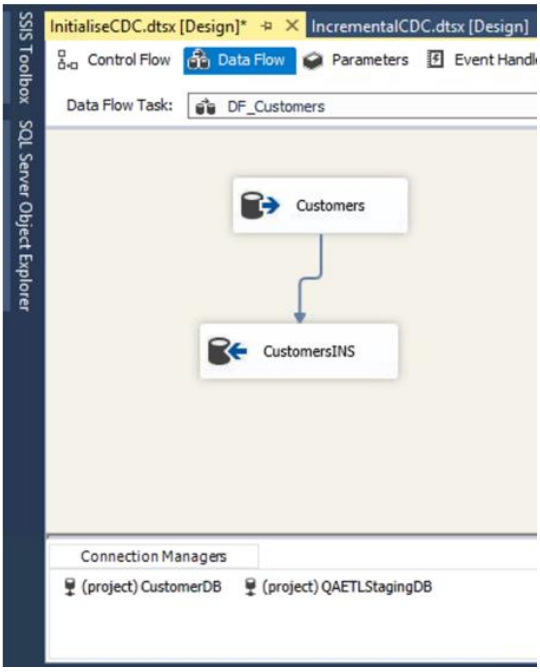
Exercise 1 - Create the Databases and Tables Necessary for the Lab

1. Open SQL Management Studio (SSMS). When asked, connect to the localhost instance of SQL using Windows Authentication.
2. In SSMS open the query file:
   `C:\Course Files\DESSIS\Mod06\Mod6LabA\Starter\`**`Mod6-LabA-Ex1.sql`**
3. Highlight and execute step 1 to create the CustomerDB database
4. Highlight and execute step 2 to create the dbo.Customers table
5. Highlight and execute step 3 to insert data into the Customers table
6. Highlight and execute step 4 to create the QAETLStagingDB database
7. Highlight and execute step 5 to create the tables for incremental data

Exercise 2 – Enable Change Data Capture in CustomerDB

1. In SSMS open the query file:
   `C:\Course Files\DESSIS\Mod06\Mod6LabA\Starter\`**`Mod6-LabA-Ex2.sql`**
2. Highlight and execute step 1 to check the ownership of CustomerDB.
   It needs to have an owner if we want to enable it for CDC.
3. Highlight and execute step 2 to enable CDC on CustomerDB.
4. Highlight and execute step 3 to enable CDC on the dbo.Customers table.

Exercise 3 – Configure Initial Data Extraction



1. Start Visual Studio 2022. Click: Continue without code …
2. Within Visual Studio 2022 go to File | Open | Project/Solution
3. Navigate to:
   `C:\Course Files\DESSIS\Mod06\Mod6LabA\Starter\IncrementalETL`
   and open **IncrementalETL.sln**
4. Open **InitialiseCDC.dtsx** from the Solution Explorer.
5. Select **the Control Flow** tab.
6. Drag on a:
   a. **CDC Control Task** and rename it to CDC_InitialLoadStart.
   b. **Data Flow Task** and rename to DF_Customers.
   c. **CDC Control Task** and rename it to CDC_InitialLoadEnd
7. Link the tasks as follows:
   a. From CDC_InitialLoadStart to DF_Customers with a **success** link
   b. From DF_Customers to CDC_InitialLoadEnd with a **success** link.
8. Right-click CDC_InitialLoadStart and **edit**.
9. Within the CDC Control Task Editor dialog set the following settings:
   a. SQL Server CDC Database ADO.NET Connection Manager: CustomerDB
   b. CDC control operation: Mark initial load start

c. Variable containing the CDC state: click New and accept the default of CDC_State.

d. Automatically store state in a database table: selected.

e. Connection manager for the database where the state is stored: QAETLStagingDB

f. Table to use for storing state: click New, and then click **Run** to create the cdc_states table.

g. State name: CDC_State

10. Click **OK** to close.

11. Right-click DF_Customers and **edit.**

12. On the Data Flow tab, drag on an **ADO.NET Source** and rename it Customers, and then **edit.**

13. Select **“dbo”.”Customers”** from the **Name of the table or the view** dropdown list. This is in the CustomerDB table – which is first on the list

14. Click on the **Columns** section and then click **OK**.

15. Drag on an **ADO.NET Destination** and rename it CustomersINS

16. Link Customers to CustomersINS with a **success** link, and then **edit** CustomersINS .

17. Select QAETLStaging as the Connection Manager and **“Staging”.”CustomersINS”** as the table.

18. Click on the **Mappings** section and click OK.

19. Click on the **Control Flow** tab.

20. Right-click CDC_InitialLoadEnd and **edit**.

21. Within the CDC Control Task Editor dialog set the following settings:

a. SQL Server CDC Database ADO.NET Connection Manager: CustomerDB

b. CDC control operation: Mark initial load end

c. Variable containing the CDC state: CDC_State

d. Automatically store state in a database table: selected.

e. Connection manager for the database where the state is stored: QAETLStagingDB

f. Table to use for storing state: cdc_states.

g. State name: CDC_State

22. Execute the package by pressing F5 or clicking the green start button at the top of the screen.

23. Click on the Data Flow tab and notice that 3 rows will have been transferred.
24. Stop the package execution and save all files.

Exercise 4 – Modify the data in the Source table

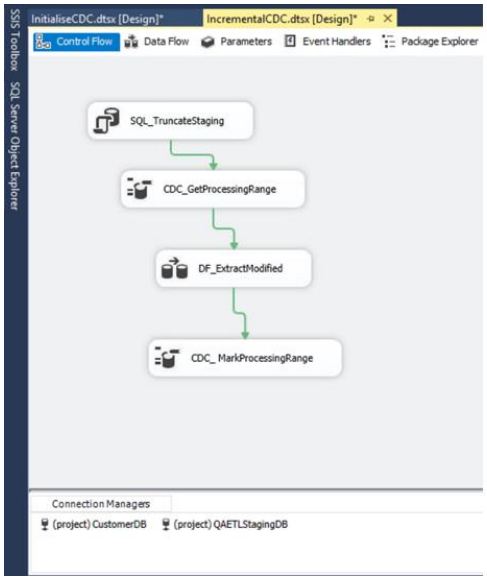
1. In SSMS, open the query file:
   `C:\Course Files\DESSIS\Mod06\Mod6LabA\Starter\`**`Mod6-LabA-Ex3.sql`**
2. Notice that the script:
   a. inserts 2 new rows into the Customers table,
   b. deletes 1 row.
   c. and updates 2 rows.
3. **Execute** all of step 1.
4. **Execute** step 2

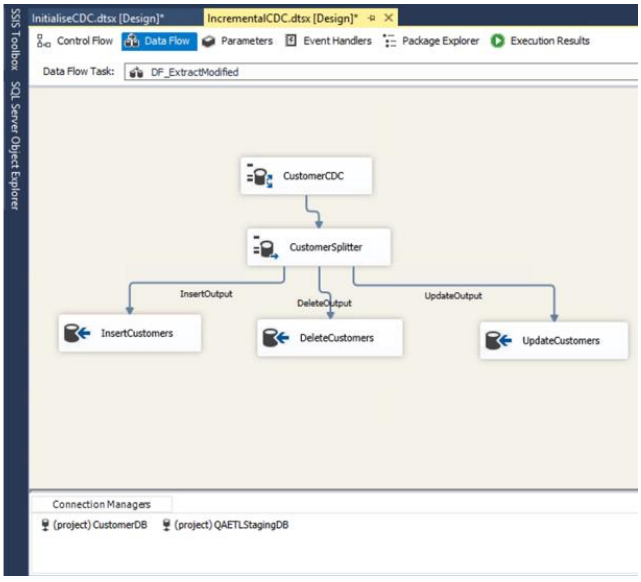
   This shows what information is stored in the CDC_States table currently.

   It should show ILEND

ILEND	The LSN where the last incremental log scan ended.
TFLSN	The Log Sequence Number where the transactional started.
TFENDLSN	The LSN where the latest transactional flush ended.
ILSN	The LSN where incremental log scanning started.

Exercise 5 – Configure an Incremental Package





1. In Visual Studio 2022, open **IncrementalCDC.dtsx** from the Solution Explorer

2. Select **the Control Flow** tab.
3. Drag on a:
    a. **Execute SQL Task** and rename as SQL_TruncateStaging.
    b. **CDC Control Task** and rename it to CDC_GetProcessingRange.
    c. **Data Flow Task** and rename to DF_ExtractModified.
    d. **CDC Control Task** and rename it to CDC_ MarkProcessingRange.
4. Link the tasks as follows:
    a. From SQL_TruncateStaging to CDC_GetProcessingRange with a **success** link.
    b. From CDC_GetProcessingRange to DF_ExtractModified with a **success** link
    c. From DF_ExtractModified to CDC_ MarkProcessingRange with a **success** link.
5. Right-click SQL_TruncateStaging and **edit**
6. In the **ConnectionType** field, choose **ADO.NET**.
7. In the **Connection** field, choose **QAETLStagingDB**
8. Click the ellipses (…) in the **SQLStatement** field. Add the following code (notice the semicolons between each statement:

```
truncate table staging.CustomersDEL ;
truncate table staging.CustomersUPD ;
truncate table staging.CustomersINS
```

9. Click **OK**.
10. Right-click CDC_GetProcessingRange and **edit**
11. Within the CDC Control Task Editor dialog set the following settings:
    a. SQL Server CDC Database ADO.NET Connection Manager: CustomerDB
    b. CDC control operation: Get processing range
    c. Variable containing the CDC state: select the existing value from the dropdown, or click New and accept the default of CDC_State.
    d. Automatically store state in a database table: selected
    e. Connection manager for the database where the state is stored: QAETLStagingDB
    f. Table to use for storing state: use the drop-down cdc_states table.
    g. State name: CDC_State.

12. Right-click DF_ExtractModified and **edit**
13. On the Data Flow tab, drag on an **CDC Source**, rename it **CustomerCDC,** and then **edit.**
14. Within the Microsoft CDC Source dialog set the following settings:
    a. ADO.NET Connection Manager: CustomerDB
    b. CDC enabled table: dbo.Customers
    c. Capture Instance: dbo_Customers
    d. CDC processing mode: Net
    e. Variable containing the CDC state: User::CDC_State.
15. Click the **Columns** section and press OK.
16. Drag on a **CDC Splitter** and rename it to **CustomerSplitter**.
17. Connect CustomerCDC to CustomerSplitter with a **success** link.
18. Drag on a **ADO.NET Destination** and rename it as **InsertCustomers**.
19. Connect CustomerSplitter to InsertCustomers with a **success** link. When the Input Output Selection box appears, choose **InsertOutput** and click OK.
20. **Edit** InsertCustomers. Choose **QAETLStaging** as the Connection manager and **“staging”.”CustomersINS”** as the table.
21. Click in the **Mappings** sections and click OK.
22. Repeat steps 18 – 21 with the following:
    a. ADO.NET Destination: DeleteCustomers.
    b. **Success** link from the splitter to the new destination choosing **DeleteOutput** as the Output.
    c. Connection manager: QAETLStaging.
    d. Table: “staging”.”CustomersDEL”
23. Repeat steps 18 – 21 with the following:
    a. ADO.NET Destination: UpdateCustomers.
    b. **Success** link from the splitter to the new destination which will automatically choose **UpdateOutput** as the Output.
    c. Connection manager: QAETLStaging.
    d. Table: “staging”.”CustomersUPD”
24. Click on the Control Flow tab.
25. Right-click CDC_MarkProcessingRange and **edit**
26. Within the CDC Control Task Editor dialog set the following settings:

a. SQL Server CDC Database ADO.NET Connection Manager: CustomerDB

b. CDC control operation: Mark processed range

c. Variable containing the CDC state: CDC_State.

d. Automatically store state in a database table: selected

e. Connection manager for the database where the state is stored: QAETLStagingDB

f. Table to use for storing state: Use the drop-down and pick the cdc_states table.

g. State name: CDC_State.

27. **Execute** the **incrementalCDC** package by pressing **F5** or clicking the green **start** button at the top of the screen.

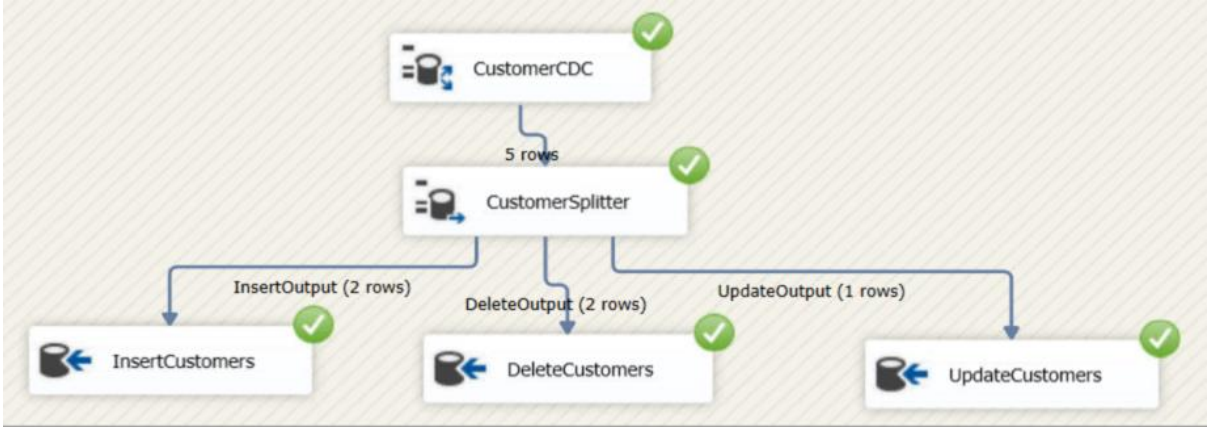
28. Click on the Data Flow tab and notice that:

    a. 2 rows will have been transferred to InsertCustomers

    b. 2 rows to DeleteCustomers

    c. 1 row to UpdateCustomers

If no rows are showing, stop the package and execute it a second time.



29. Stop the package execution and save all files.

    Go back to SSMS and re-run the following code

```
use QAETLStagingDB
go

select * from cdc_states;
```

It should show: TFEND ~ Trickle Feed Update Ended

30. Close SSMS and close Visual Studio 2022.

Lab B: Implementing Slowly Changing Dimensions

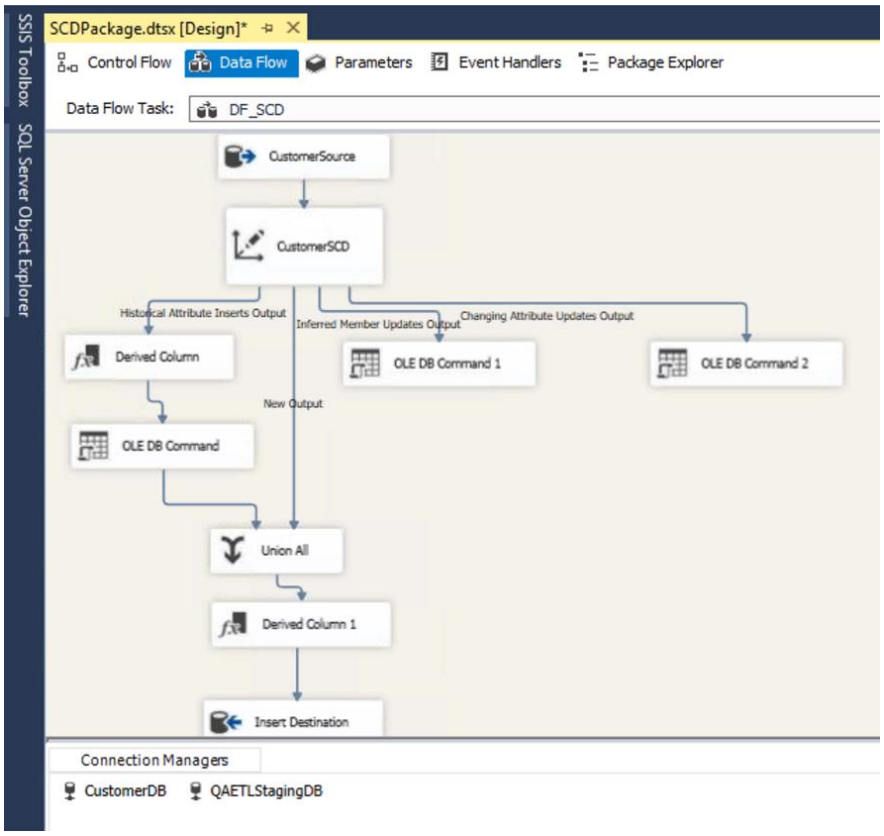
In this Lab we will:

- Create the databases and tables for the Lab.
- Configure an SCD Package.
- Modify the source table.
- Rerun the package

Exercise 1 - Create the Databases and Tables Necessary for the Lab

1. Open SQL Management Studio (SSMS). When asked, connect to the localhost instance of SQL using Windows Authentication.
2. In SSMS open the query file:
   `C:\Course Files\DESSIS\Mod06\Mod6LabB\Starter\`**`Mod6-LabB-Ex1.sql`**
3. Highlight and execute step 1 to create the CustomerDB database.
4. Highlight and execute step 2 to create the Customers table.
5. Highlight and execute step 3 to insert data into the Customers table.
6. Highlight and execute step 4 to create the QAETLStagingDB database.
7. Highlight and execute step 5 to create the staging table.

Exercise 2 – Configure an SCD Package



1. Start Visual Studio 2022. Click: Continue without code …
2. Within Visual Studio 2022 go to File | Open | Project/Solution
3. Navigate to: `C:\Course Files\DESSIS\Mod06\Mod6LabB\Starter\SCDProject` and open **SCDProject.sln**
4. Open **SCDPackage.dtsx** from the Solution Explorer.
5. The **Control Flow** has a data flow task already on it.
   Double click on the task DF_SCD to open the **Data Flow tab**.
6. Drag on an **OLE DB Source** task and rename it as **CustomerSource**.
   Then **Edit.**
7. Make sure **CustomerDB** is the OLE DB Connection Manager.
8. Pick **dbo.Customers** as the table and click OK.
9. Drag on a Common > **Slowly Changing Dimension** task and rename it as CustomerSCD.
10. Link the task with a **success** link.
11. **Edit** CustomerSCD. This will take you into the SCD Wizard.

    Click Next.
12. On **Select a Dimension Table and Keys** choose the following.

a. Connection Manager: QAETLStagingDB

b. Table or View: staging.Customers

c. Select **CustomerID** (from the dropdown) in the input column to match **CustomerAlternateID** in the Dimension Columns and choose **Business Key** (from the dropdown) in the Key Type column.

d. Click Next

13. On **Slowly Changing Dimension Columns** choose the following. Click Next:

    a. Add **city** with a change type of **Changing attribute.**

    b. Add **country** with a change type of **Changing attribute**

    c. Add **name** with a change type of **Historical attribute.**

    d. Click Next

14. On **Fixed and Changing Attribute Options** tick **Change all matching records, including outdated record, when changes are detected in a changing attribute** and
Click Next.

15. On **Historical Attribute Options** select **Use start and end dates to identify current and expired records,** choose the following and click Next:

    a. Start date column: RecordValidStartDate.

    b. End date column: RecordValidEndDate.

    c. Variable to set date values: System::StartTime.

    Click on the 'Variable to set date values:' input box,
    and SCROLL THE WHEEL MOUSE to change between variables.

    Expand the width of the window if you can't see the field Variable

    d. Click Next

16. On **Inferred Dimension Members** click Next, then Finish.

17. Notice the various steps that have been created.

18. Edit some of the steps to examine what they are doing.

19. **Execute** the **SCDPackage** package by pressing **F5** or clicking the green **start** button at the top of the screen.

20. Click on the Data Flow tab and notice that 3 rows will have been transferred via the Historical Attribute Inserts Output path.

21. **Stop** the package execution and save all files.

Exercise 3 – Modify the data in the Source table

1. In SSMS, open the query file:
   `C:\Course Files\DESSIS\Mod06\Mod6LabB\Starter\`**`Mod6-LabB-Ex2.sql`**
2. Highlight and **execute Step 1** to see the contents of **staging.customers**.

   Notice the RecordValidStartDate columns has now be filled in,

   but that RecordValidEndDate still contains NULLs.
3. Highlight and **execute Step 2** to make some changes to the source Customers table.

Exercise 4 – Rerun the package

1. Go back to Visual Studio 2022
2. **Execute** the SCDPackage package by pressing **F5** or clicking the green **start** button at the top of the screen.
3. Click on the Data Flow tab

   Notice that 1 row will have been transferred via the Historical Attribute Inserts Output path

   and 1 row via the Changing Attribute Updates Output path.
4. **Stop** the package execution and save all files.
5. Go back to SSMS and **execute Step 3**

   Notice that:

   The value of city for CustomerAlternateID = 3 has just been overwritten
   There is an extra row for CustomerAlternateID = 1 with the new name.
   The RecordValidEndDate for the original row has now been filled in.

Results Messages

	CustomerID	CustomerAlternateID	name	city	country	RecordValidStartDate	RecordValidEndDate
1	1	1	Fred	London	UK	2025-03-30 22:13:58.000	2025-03-30 22:17:03.000
2	2	2	Lily	Denver	US	2025-03-30 22:13:58.000	NULL
3	3	3	Reza	Liverpool	UK	2025-03-30 22:13:58.000	NULL
4	4	1	Freddy	London	UK	2025-03-30 22:17:03.000	NULL

6. Close SSMS and close Visual Studio 2022.

M7 - Deploying and Configuring SSIS Packages

Lab A: Creating SSIS Catalog & Environments & Deploying Project

In this lab you will see how to:

- Create an Integration Services Catalog
- Deploy a SSIS package
- Create Environments and variables

Exercise 1 – Create an Integration Services Catalog

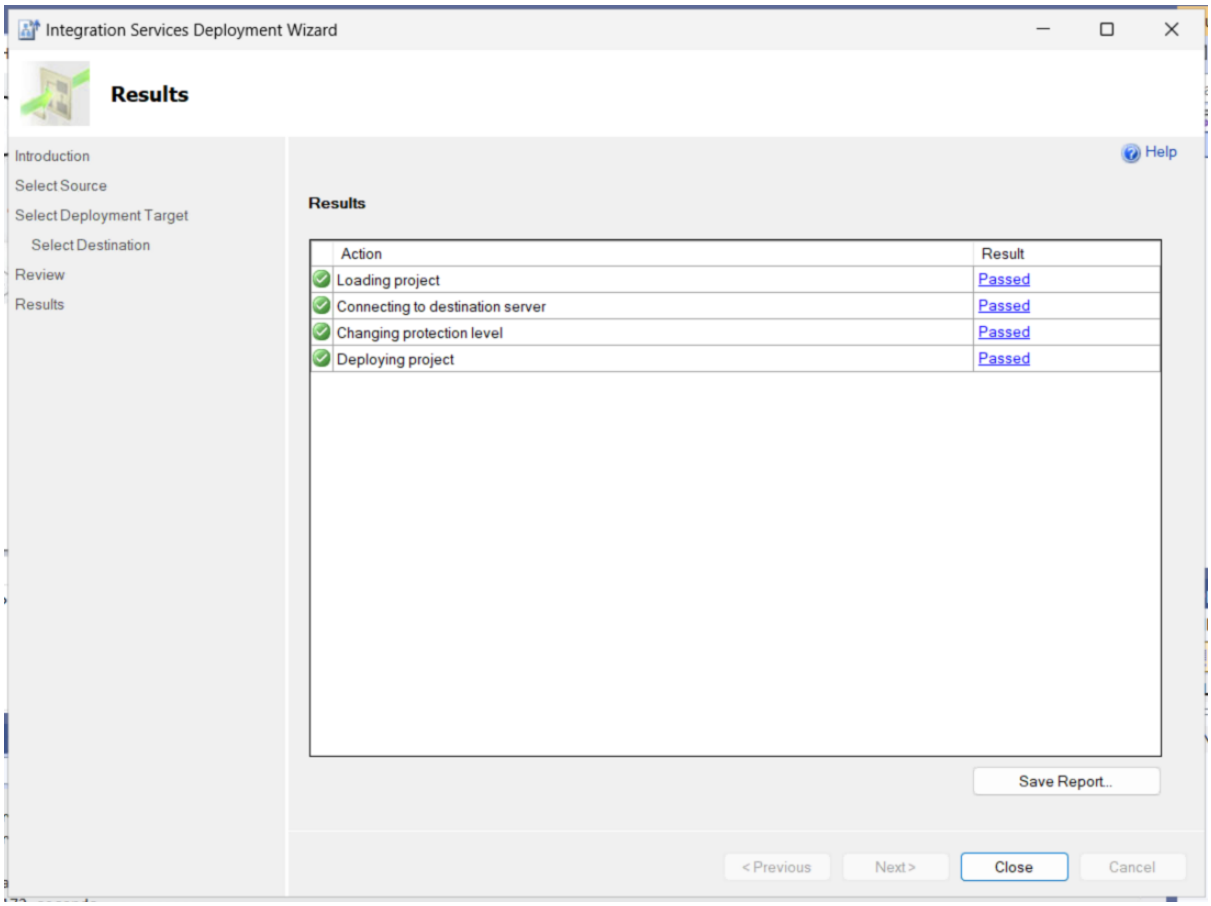
1. Open SQL Management Studio (SSMS). When asked, connect to the localhost instance of SQL using Windows Authentication.
2. In SSMS open the query file:
   `C:\Course Files\DESSIS\Mod07\Mod7LabA\Starter\`**`Mod7-LabA-Ex1.sql`**

   Run the script. This will create the QAETLStagingDB and the Products table.
3. In SQL Management Studio (SSMS) in the Object Explorer, expand **Integration Services Catalogs**
4. Right-click **Integration Services Catalogs** and click **Create Catalog…**
5. Click to **Enable CLR Integration**,
6. Click to **Enable Automatic Execution of Integration Services Stored Procedure at SQL Server Startup**.
7. Enter **Pa55w.rd** in the 2 password boxes
8. Click OK.
   This should create a folder under Integrated Service Catalogs called: **SSISDB**
9. Right-click the **SSISDB** folder and choose **Create Folder…**
10. Under Folder Name enter: **ETLLab**
11. Click OK.

Exercise 2 – Deploy a SSIS Package

1. Start Visual Studio 2022. Click: Continue without code …
2. Within Visual Studio 2022 go to File | Open | Project/Solution

3. Navigate to: `C:\Course Files\DESSIS\Mod07\Mod7LabA\Starter\EtlLab` and open **EtlLab.sln**
4. In Visual Studio, choose **Build → Build Solution**
5. When this has finished, in Visual Studio, go to the **Solution Explorer,** right-click **EtlLab (SQL Server 2016)** and choose **Deploy**.
6. When the **Integration Services Deployment Wizard** appears, read the instructions, and then click **Next**. If a warning appears, click **OK.**
7. On the **Select Deployment Target** screen click **next.**
8. Enter **localhost** for the Server name and then click Connect.
9. Click **Browse** next to **Path**.
10. Choose the **SSISDB → ETLLab** folder and click **OK**.
11. Click **Next** and then **Deploy**.

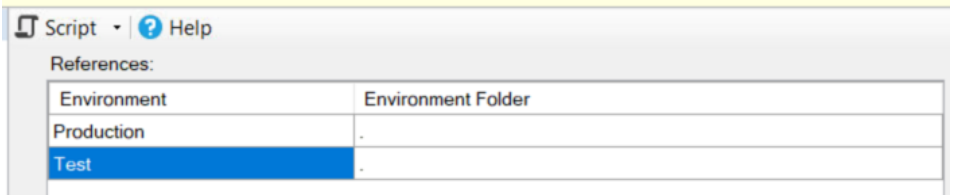


12. When this has finished click **Close**.

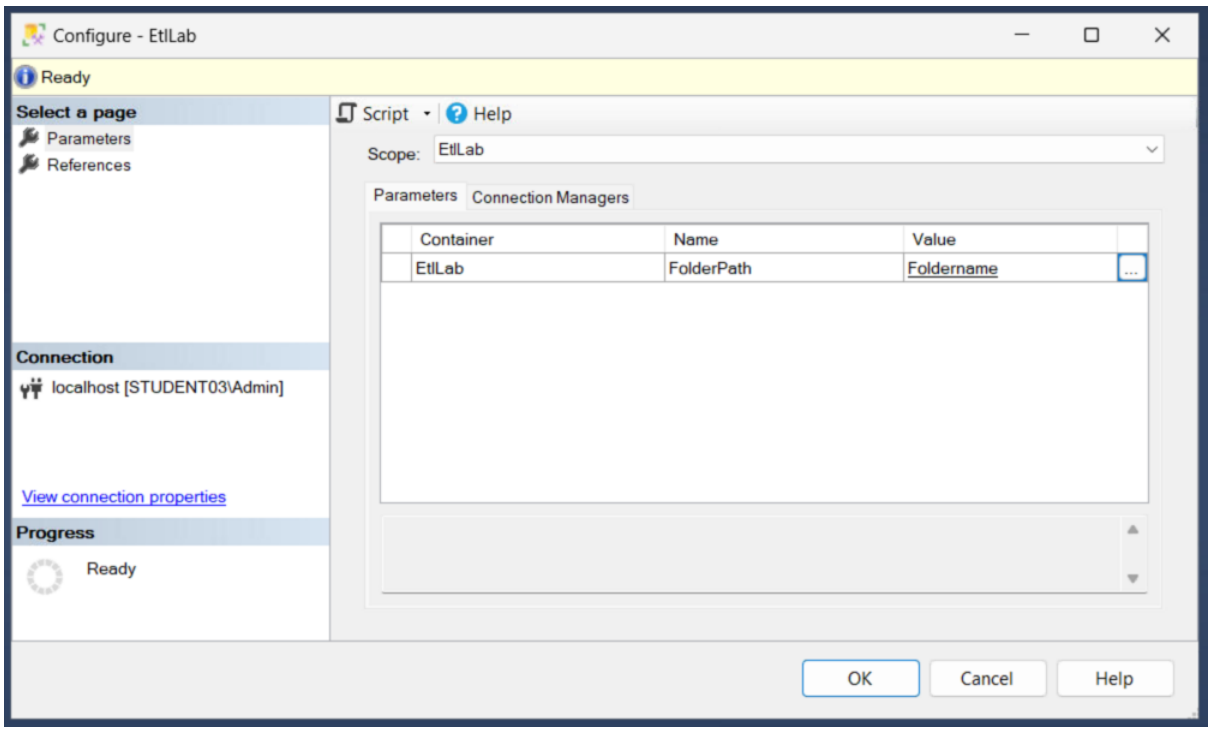
Exercise 3 - Create Environments and variables

1. Back in **SSMS**, expand the **ETLLab** folder and then the **Projects** folder.

   You should see that the ETLLab project has been deployed.
2. Right-click the **Environments** folder and choose **Create Environment**.
3. Create an environment called **Production**
4. Create a second environment called **Test**
5. Right-click **Production** and go to **Properties**.
6. In **Properties** click on the **Variables** section.
7. Create a variable with the following settings:
   a. Name: Foldername
   b. Type: String
   c. Description: The folder name
   d. Value: `C:\Course Files\DESSIS\Mod07\Production\` (make sure you include the final ****)
   e. Sensitive: No (so **not** selected)
8. Click **OK**.
9. Right-click **Test** and go to **Properties**.
10. In **Properties** click on the **Variables** section.
11. Create a variable with the following settings:
    a. Name: Foldername
    b. Type: String
    c. Description: The folder name
    d. Value: `C:\Course Files\DESSIS\Mod07\Test\` (make sure you include the final ****)
    e. Sensitive: No
12. Click **OK**.
13. In the **Projects** folder, right-click **ETLLab** and choose **Configure**.
14. On **References** section, **Add** both the **Production** and **Test** environments.



15. Click on the **Parameters** section, click on the **Scope** list (dropdown) and choose **ETLLab**.
16. On the **Parameters** tab, click the ellipses (…) in the **Value** section
17. Select **Use Environment variable** and pick **Foldername**.
18. Click **OK**



19. Click **OK**
20. Leave SSMS open for the next lab.

Lab B: Running an SSIS Package

In this lab you'll see how to:

- Run a package in SSMS
- Schedule a SQL Agent job to run the package

Exercise 1 - Using SQL Server Management Studio to run an SSIS Package

1. Using **Windows File Explorer** and navigate to the folder:
   `C:\Course Files\DESSIS\Mod07\Production`

   Notice that there are **4 productdetails** files.

2. Using **Windows File Explorer** and navigate to the folder:
   `C:\Course Files\DESSIS\Mod07\Test`

   Notice that there are **2 productdetails** files.

3. In SSMS, expand the **ETLLab** folder and **packages**.

   Integration Services Catalogs → SSISDB → EtlLab → Projects → EtlLab

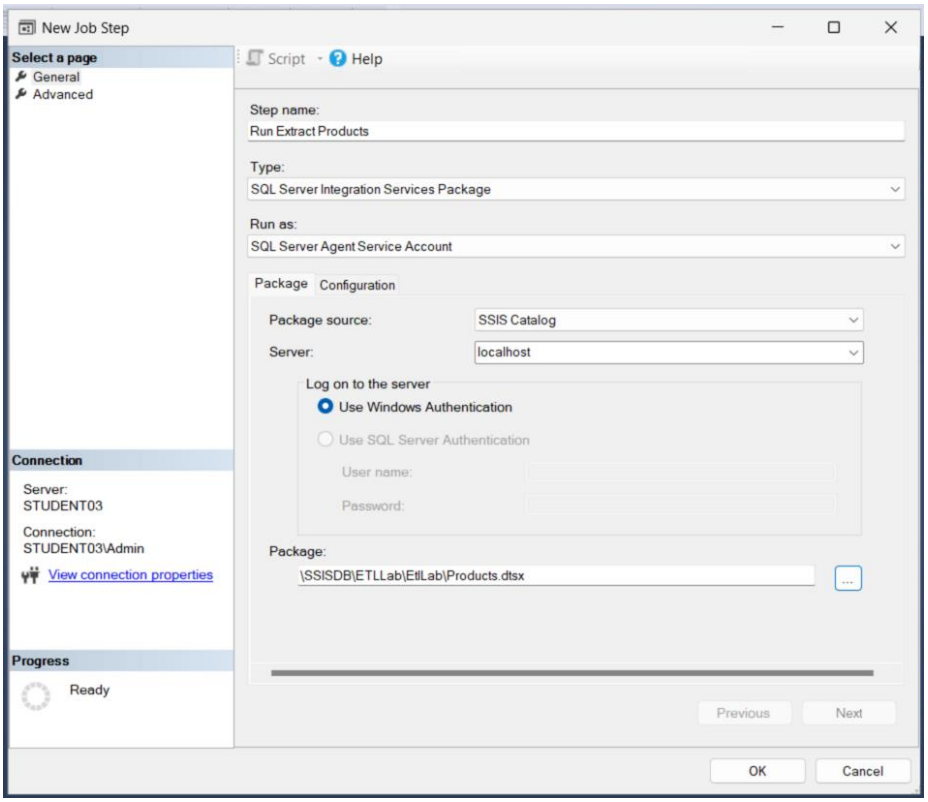
4. Right-click **Products.dtsx** and choose **Execute**.
5. Click the **Environment** checkbox (at the bottom of the box) and pick **.\Test**
6. Click OK to run the package.
7. Click **No** on the Overview Report box.
8. **Execute** the package again, this time choosing the **.\Production** environment.
9. Click **No** on the Overview Report box.
10. Right-click the **SSISDB** folder, point to **Reports | Standard Reports** and choose **Integration Services Dashboard**.
11. Click on the **All Executions** link
12. Notice you should have 2 successful executions.
13. For each of the executions, click the **Overview** link.

    Notice the **Environment** name, and the **FolderPath.**

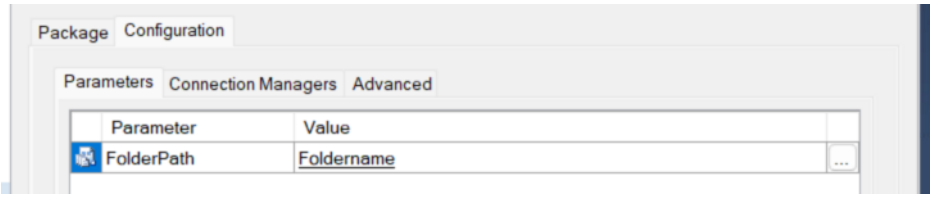
14. Close any reports.

Exercise 2 - Using SQL Server Agent to schedule an SSIS Package as an automated job

1. In Object Explorer, right-click **SQL Server Agent → New,** and click **Job**.
2. In the New Job dialog box, in the Name box enter: **Product Job**
3. In the Owner box replace the existing value with: **sa**
4. Click the **Steps** page and click **New**.
5. Type **Run Extract Products** in the **Step name** box.
6. In the **Type** dropdown select: **SQL Server Integration Services Package**
7. Make sure **SSIS Catalog** is selected in the **Package source** list.
8. Type **localhost** in the **Server** field and Use Windows Authentication
9. Click the ellipses (...) button for the Package box,
   and in **the Select an SSIS Package** dialog box,
   expand **SSISDB | ETLLab | ETLLab** and select **Products.dtsx**,
   and then click OK.



10. Click **Configuration** tab (under General next to the Package tab),
    select **Environment**, and select the **.\Production** environment.



11. Click on the **Advanced** section and in **On success action**: select **Quit the job reporting success**. Then click OK.
12. Click on the **Schedules** section and click **New**.
13. In the Name box, type **ETL Schedule**.
14. In the **Schedule type** list, select **One time**.
15. Ensure **Enabled** is selected.
16. In the **One-time occurrence** section:

    select the current day
    enter a time that is two minutes later than the current time (based on the SSMS's time)
17. Click OK to close the New Job Schedule dialog box
18. Then click OK to close the New Job dialog box.
19. In Object Explorer, expand SQL Server Agent

    Right-click **Job Activity Monitor**, and click **View Job Activity**.

    LastRun Outcome for Product Job should be: Unknown
20. Wait for two minutes.
21. In the Job Activity Monitor – localhost window

    Note the **Status** and **Last Run Outcome** values for the **Product** job.

    If the job has not yet completed, wait a minute.

    Then click **Refresh** until the job has completed successfully
22. Right click the job and select: View History
23. Click Close.
24. Close SQL Server Management Studio
25. Close Visual Studio 2022