Implementation and ETL Process

Each of the four parts of the assignment is worth **1 point**.

The first three parts of this assignment consists of three step-by-step labs described on the next 3 pages. Throughout your lab work, you need to take 2-4 screen shots per project to document the ETL process, save them into `Assign10_SSIS_Demo.docx` Word file, and label them accordingly. The last part of the assignment will use the order entry operational database, and the associated data mart you designed in the previous assignment, to demonstrate the application of the ETL process using SSIS.

1. SSIS – First Integration Service Project[1]

Step 1:Use the `MySQL10_Product_Tbl.sql` file to create the **ProductsDemo** table on your SQL Server Express in a database named **SSIS_Demo** (or similar). Once you create the table, it will show up as **dbo.ProductsDemo** under **Tables** folder.

Step 2: Create a new **Integration Services Project** in **Visual Studio** or **SQL Server Development Tools**, navigate to `10_Implem_ETL` folder, `Assign` subfolder, and save the project as `Assign10_SSIS_Demo`. The main window in the **SSDT** environment is the **design canvas**, where you build data integration logic, which is contained within so-called **Packages**, the default package name is **Package.dtsx**.

Step 3: Continue by adding a **Data Flow Task** from the top of the **SSIS Toolbox** to the canvas, and then double-click on it to create data flow logic that loads data from a CSV file into a SQL Server table. Add the **Source Adapter** by double-clicking on the **Source Assistant** in the Toolbox. Select **Flat File** as the source type, and click on **New…** to create the source connection, and Browse for the `Assign10_Product_Data.csv` file on your hard drive.

Step 4: Click on the Columns tab to review the columns SSIS has detected in the CSV file, and be aware that the Flat File Source adapter reads in the contents of files as strings by default. Use Advanced tab to tell SSIS to suggest data types by scanning the file and confirm that it made the expected choices.

Step 5: Next add a destination adapter which will define where data leaves the data flow. Double-click the **Destination Assistant** on the Toolbox, chose **SQL Server** as the destination, and click on **New…** to crate the destination connection. Assuming you installed SQL Server Express locally, type **your_computer_name\SQLEXPRESS** in the **Server name** drop-down. Select **Windows Authentication**, and take another screen-shot of the elements on the canvas, and paste into Word document.

Step 6: Select the **source adapter**, and notice that the two arrows re-appeared. The **blue arrow** represents the normal data output, and the **red arrow** the error output. Click the **normal output**, drag it over to the **destination adapter**, and click again to connect the two.

Step 7: Double-click on the destination adapter to configure it, paying attention to the instructional message at the bottom of the window on what to do next. Start by picking **dbo.ProductsDemo** table, and then switch to the **Mappings** page, and establish the mappings between the columns of the source CSV file and the destination SQL Server table.

[1] The SSIS material for the first three parts was adopted from Prof. De Liu's notes

Step 8: Run the package by clicking on the **green Start** button. After the package executes, you should see two (2) green check marks on top of both the source and destination adapters, along with a message that 99 rows have been transferred. Take a screen-shot of the elements on the canvas, make sure to select only the controls, and paste them into `Assign10_SSIS_Demo.docx` Word file. End the run by clicking the **red Stop** button, and make sure to save all the changes to the project. Switch to **SQL Server Management Studio**, right-click on **dbo.Products Demo** table and **Select Top 1000 Rows** to visually verify the results.

2. SSIS – Control Flow Basics

Step 1: In this lab, we simulate integrating the execution of the SSIS package into an external system via the use of semaphore files that indicate whether the execution succeeded or not. Note that you have to use the two (2) subfolders, `Semaphores` and `Result` under new `Assign10_SSIS_Ctrl_Flow` folder, with two(2) blank files `Failure` and `Success` in the `Semaphores` folder.

Step 2: Click on the **Control Flow** tab at the top of the design canvas, add a **File System Task** from the Toolbox, double-click on it to set the **DestinationConnection** property by creating a **<New Connection>**, **Existing file** as usage type, and browse for the `Assign10_SSIS_Ctrl_Flow\Results` folder and select `Success` file. Set **OverwriteDestination** property to **True** so you can run this package repeatedly, and keep the **Operation** as **Copy file**.

Step 3: Create another connection manager for the **SourceConnection** property at the bottom of the list, this time for the **Existing file**, and browse for `Assign10_SSIS_Ctrl_Flow\Semaphores\Success` file. Observe the **Connection Managers** tray below the canvas, and note the Success and Success 1 connection managers you just created. Rename the first one **SuccessDestination** and the second one **SuccessSource**.

Step 4: Rename Data Flow Task as **Load data into Products table**, and File System Task as **Copy success semaphore**. Connect the two tasks by clicking the **green arrow** from the data flow to semaphore task. The green arrow indicates that this is an "on success" constraint, meaning that the downstream task will execute only if the upstream tasks finished without any errors.

Step 5: Next we need to add the logic in case the data flow task encounters an error. We need two (2) "failure" connection managers, one for source and one for destination. Right-click in the **Connection Managers** tray below the canvas, select **New File Connection**, browse for `Assign10_SSIS_Ctrl_Flow\Semaphores\Failure` file, and rename the connection **FailureSource**. Right-click again, select the **New File Connection**, **Existing file** usage type, browse for the `Assign10_SSIS_Ctrl_Flow\Results` folder, select `Failure` file, and rename the connection **FailureDestination**.

Step 6: Add a new **File System Task**, rename it **Copy failure semaphore**, and pick the failure destination and source connection managers you just created. Create a link / constraint between the data flow and failure semaphore tasks, right click on it and change the type to **Failure**. Take the screen-shot of the elements on the canvas, and paste them into Word document.

3. SSIS – Data Flow Basics

Step 1: This lab extends the previous one by sending data from a single CSV file to two tables instead of one. You need to first create the exact same products table structure, but with a different name, **ProductsExpensiveDemo** instead.

Step 2: Open the data flow task, delete the existing path (blue arrow), and make some space between the source and destination adapters because we are going to add a transformation between them. Add a **Conditional Split** transformation to the canvas which will be used to send the expensive products to a different table from the normal products.

Step 3: Connect the source adapter with the conditional split, and then configure the conditional split transformation by building an expression that identifies expensive products. You do this by typing **Normal** under **Output Name** and **[Unit Cost] <= 50** under **Condition**, and then **Expensive** with **[Unit Cost] > 50**. Rename default output name at the bottom to something more descriptive like **Neither Normal Nor Expensive**.

Step 4: Rename the existing destination adapter (OLE DB Destination) as **Normal Products Table**. Copy/paste it and rename **Expensive Products Table**, and redirect the export to **ProductsExpensive** table on the SQL Server. Connect the conditional split with both destination adaptors, choosing the appropriate output as **Normal** and then **Expensive**.

Step 5: Because some data transformations could generate errors (ex. dividing by zero) because of bad data in the input file, and in order to prevent the package execution failure in the middle of the data transfer, we need to configure the error output of the conditional split transform. Double-click on the **Conditional Split** and **Configure Error Output** button at the bottom. Change all the **Error** and **Truncation** handling to **Redirect Row**. This will indicate that the bad rows be sent out to the error output rather than failing the entire process. Please note the warning on the conditional split. SSIS noticed that you configured the error output, but haven't connected it to anything yet. This means that the rows will still be lost, and SSIS wants to make sure you are aware of this fact.

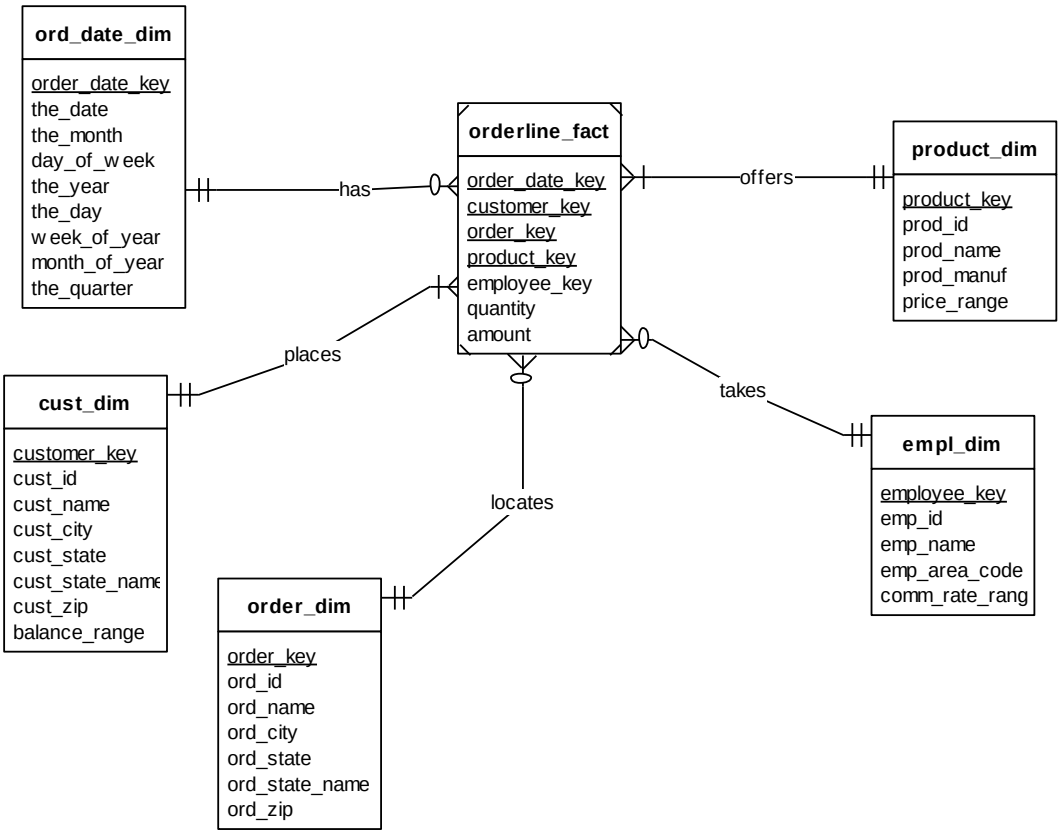
Step 6: Add a **Flat File Destination** adapter from the Toolbox (from the **Other Destinations** group at the bottom). Connect the error output of the conditional split to the flat file designation adapter. Configure the destination for error rows by creating an new flat file connection manager, choose **Delimited**, call it **Conditional Split Error Rows**, and specify the path to `Assign10_Data_Flow\ErrorRows.csv` and save as CSV type. Check the **Column names in the first data row** at the bottom, and verify **Columns** and **Advanced**.

Step 7: Double-check the Mappings and notice the Error related columns. Make sure to remove all 99 rows from the **ProductsDemo** table first. Run the package and verify the results in SQL Server. Take the screen-shot of the elements on the canvas, and paste them into Word document. There should be **48 rows** in the **ProductsDemo** table, with unit costs under $50, with the remaining **51 rows** over $50 in the **ProductsDemoExpensive** table.

4. SSIS – Order Entry ETL

Use the `ERD9_OrdEntry_Data_Mart.erd` dimensional model (shown below or similar) from the previous assignment to implement the **Order Entry** data mart on **SQL Server**, and design the ETL process that will extract, transform and load the data from the operational **OrderEntry** database, created using `MySQL10_OrdEntry_ALL.sql`, into the Order Entry data mart named **OrdEntry_Data_Mart** or similar.

All the SQL code for creating dimension and fact tables, and all the needed lookups (see Canvas for few examples of those), views and queries for the ETL process, must be saved in `MySQL10_OrdEntry_DM.sql` file. You must use INT IDENTITY(1,1) for all surrogate keys to auto-generate their values. Note: The partially completed code in the SQL file is based on the dimensional model below.



For the order date dimension, you should generate a CSV file with all the dates in 2030, including all the attribute columns calculated with basic Excel functions. For the customer balance, I used <$100 as low, between $100 and $1,000 as medium, and >$1,000 as high. For commission ranges, <4% for low, 4%-6% for medium, and >6% as high could be the breakdown. Finally, for prices, a simple split into <$100 as low and >$100 as high could be the way to go. It must be easy to distinguish between high customer balance, high employee commission, and high price with values such as "High Balance", "High Commission", and "High Price", or similar. See the SQL file for details on these.

The couple of pages of screen shots documenting the ETL process should end up in in `Assign10_SSIS_OrdEntry.docx` Word file.

Submission: You must submit `Assign10_SSIS_Demo.docx`, `MySQL10_OrdEntry_DM.sql`, and `Assign10_SSIS_OrdEntry.docx` on Canvas by the designated due date.