|  |  |
| --- | --- |
| Image result for csueb logo | **BAN 622 – Data Warehousing and BI** |
| **Project 2**  **Points: 15** |

Extreme Mountain Bike (EMB) is a fictitious company which sells mountain bikes to its customers. The company is capturing following master and transaction data in flat files which is provided to you.

1. Employee (EmployeeID, EmployeeFirstName, EmployeeLastName, DepartmentID, EmployeeAddress, Gender, EmployeeBirthDate, Salary, RegionID)
2. Product (ProductID, ProductName, Cost, WholeSalePrice, MSRP)
3. Customer (CustomerID, CustomerFirstName, CustomerLastName, CustomerLocation, CustomerAge, YearsOfExperience)
4. Department (DepartmentID, DepartmentName)
5. Region (RegionID, RegionName)
6. SalesOrder (OrderID, PODate, ProductID, CustomerID, CustomerPO, EmployeeID, Quantity, UnitPrice)

**Test the Database**

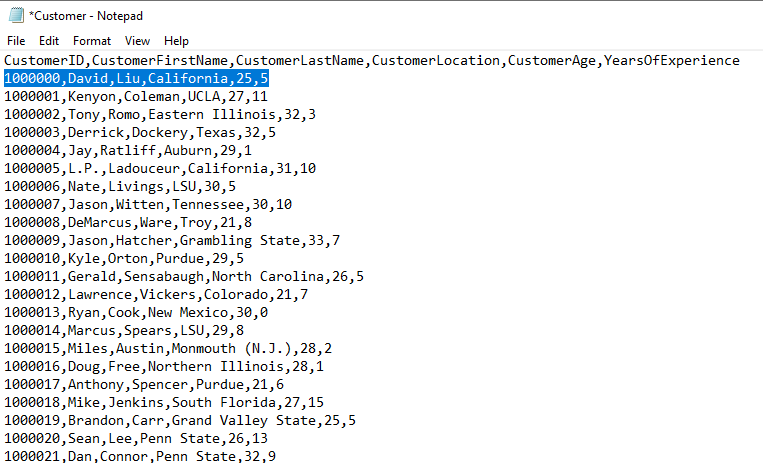
Perform the following updates to the source files *after* loading the data in each table.

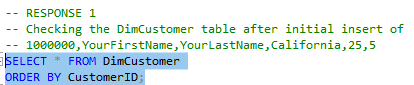
**Response 1:**

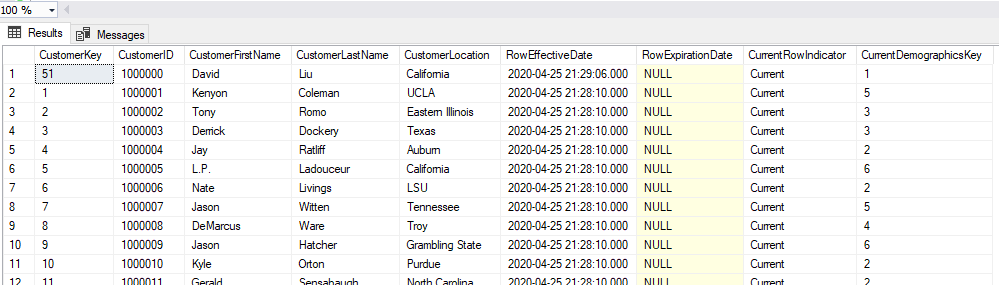
Add following row in the Customer source file (row 2 below the header) with your name.

1000000,YourFirstName,YourLastName,California,25,5

Execute the Customer data flow to insert the above row in the Customer dimension.





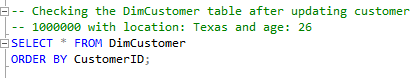


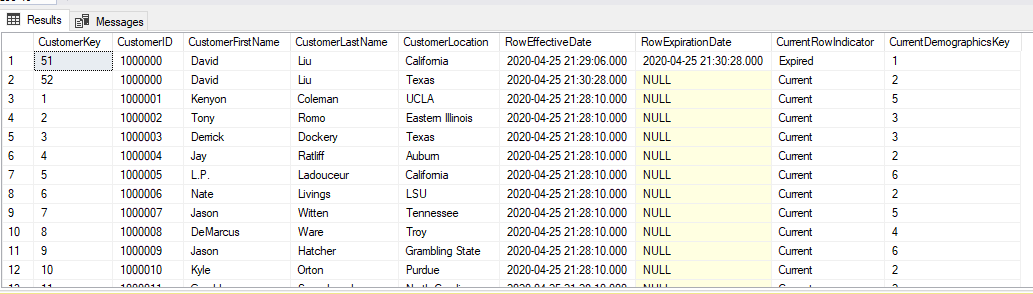
Update the CustomerLocation to Texas and CustomerAge to 26 for CustomerID 1000000 in the

Customer source file.



Execute the data flow to update the record in the dimension. Submit the screenshot of the Customer dimension after sorting by CustomerID.



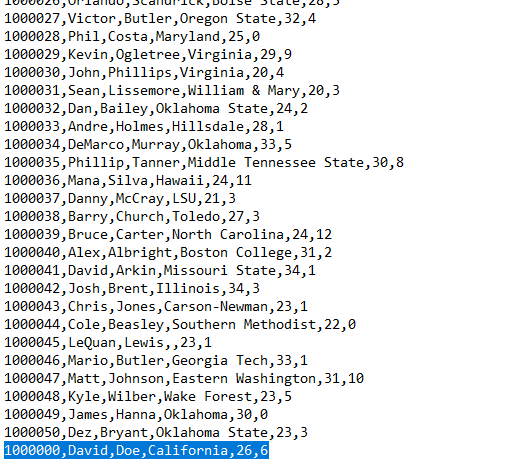


**RESULT:**

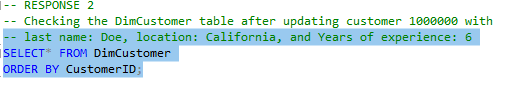
**The location update for customer 1000000 retained the previous location, hence forming a new row, which is a type 2 change. Furthermore, the RowExpirationDate now contains a date, signaling that the row information has expired, and the new row for customer 1000000 (CustomerKey 52) is now the current row. Lastly, the CurrentDemographicsKey has changed from 1 to 2 as customer 1000000 now belongs in the age group of 26-30, but still have less than 6 years of experience.**

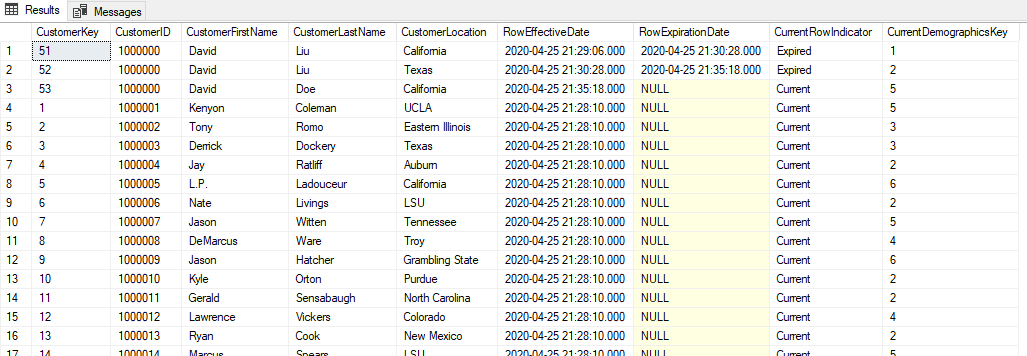
**Response 2:**

Again update the CustomerLastName to Doe, CustomerLocation to California, and YearsOfExperience to 6 for CustomerID 1000000 in the Customer source file and save it.



Submit the screenshot of the Customer dimension after running your model.



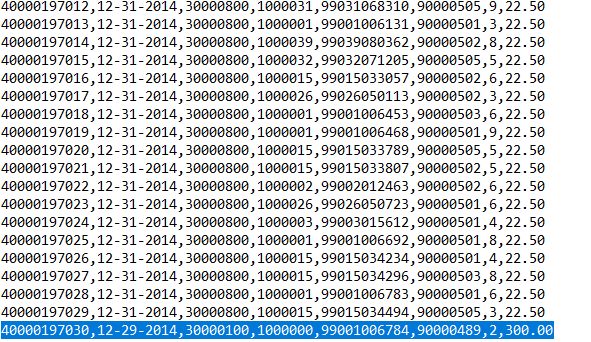


**RESULT:**

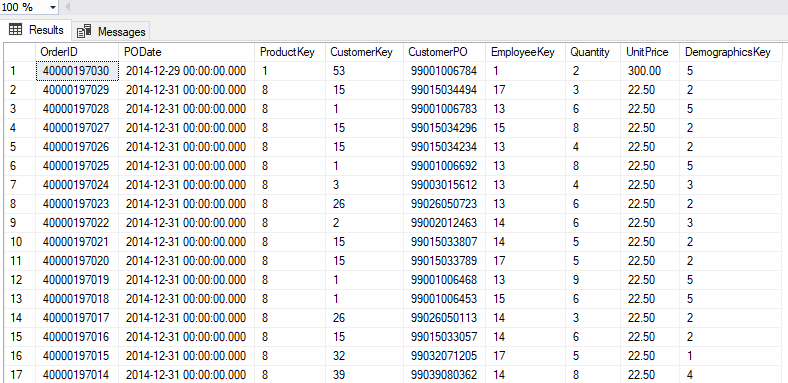
**The location update for customer 1000000 retained the previous location, hence forming a new row, which is a type 2 change. Furthermore, the Previous RowExpirationDate now contains a date, signaling yet another row information has expired, and the new row for customer 1000000 (CustomerKey 53) is now the current row. Lastly, the CurrentDemographicsKey has changed again from 2 to 5 as customer 1000000 is between the age of 26-30, and has more than 5 years of experience.**

**Response 3:** Add following transactions in the SalesOrder source file. Submit the screenshot of the fact table after running your model.

40000197030,12-29-2014,30000100,1000000,99001006784,90000489,2,300.00







**RESULT:**

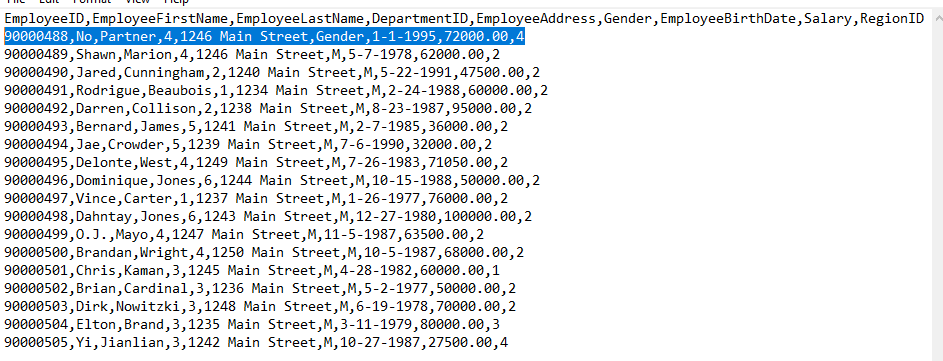
**OrderID of the order is, as listed in the above table, 40000197030, with an order date (PODate) of 12-29-2014, Customer product order (CustomerPO) of 99001006784, Quantity of 2, and a unit price (UnitPrice) of 300.00. Because the order had a product id of 30000100, customer id of 1000000, and employee id of 90000489, the values for ProductKey, CustomerKey, and EmployeeKey came out as 1, 53, and 1, respectively. Demographics key of the customer is 5, given that that was the customer’s CurrentDemographicsKey value.**

**Response 4:**

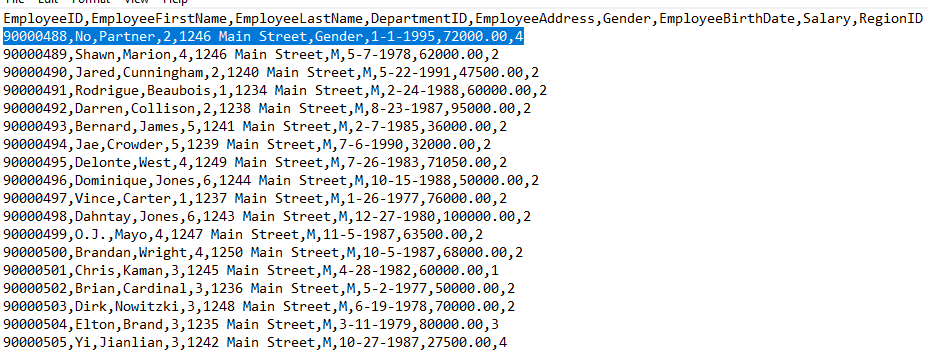
Add following row in the Employee source file (row 2 below the header) with your project partner’s name and gender.

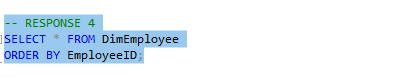
90000488,PartnerFirstName,PartnerLastName,4,1246 Main Street,Gender,1-1-1995,72000.00,4

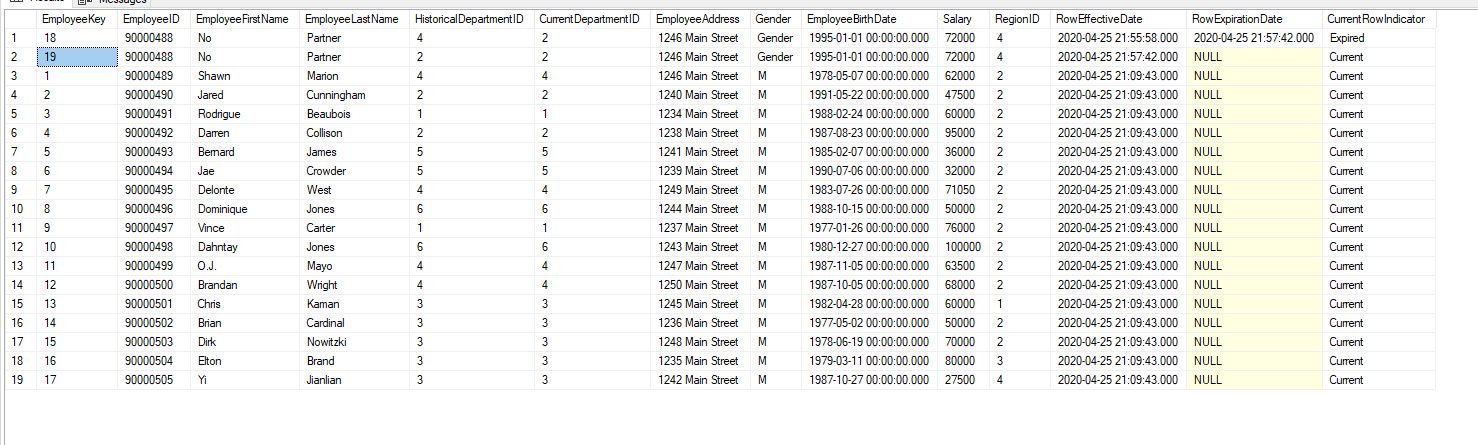
Execute the Employee data flow to insert the above row in the Employee dimension.



Update the DepartmentID to 2 for EmployeeID 90000488 in the Employee source file and save it. Submit the screenshot of the Employee dimension after running your model.





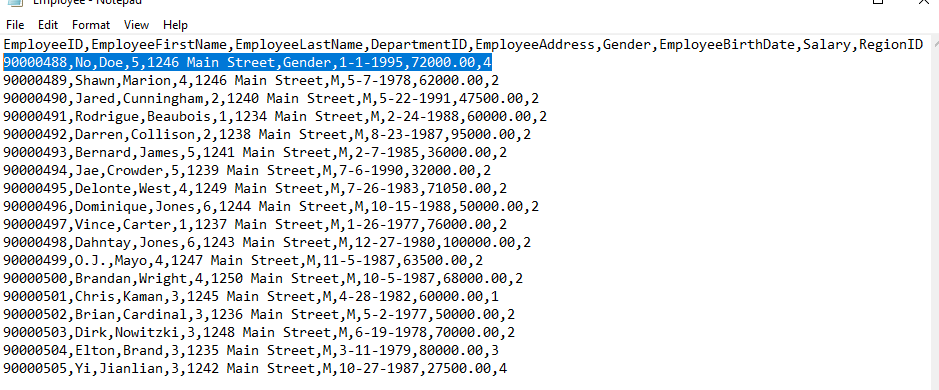


**RESULT:**

**The row** 90000488,No,Partner,4,1246 Main Street,Gender,1-1-1995,72000.00,4 **was inserted into the database. Department ID of the inserted employee was then updated from 4 to 2. Shown in the dimension above, the HistoricalDepartmentID column kept the historical record for the customer (4 and 2). However, the CurrentDepartmentID column now only contains the department ID of the current department, which is 2. RowExpirationDate now contains an expiration date for Employee 90000488, signifying that the row pertaining to EmployeeKey 18 is now expired. Also, the CurrentRowIndicator column now contains the “Expired” status for EmployeeKey 18 and “Current” for EmployeeKey 19 of Employee 90000488.**

**Response 5:**

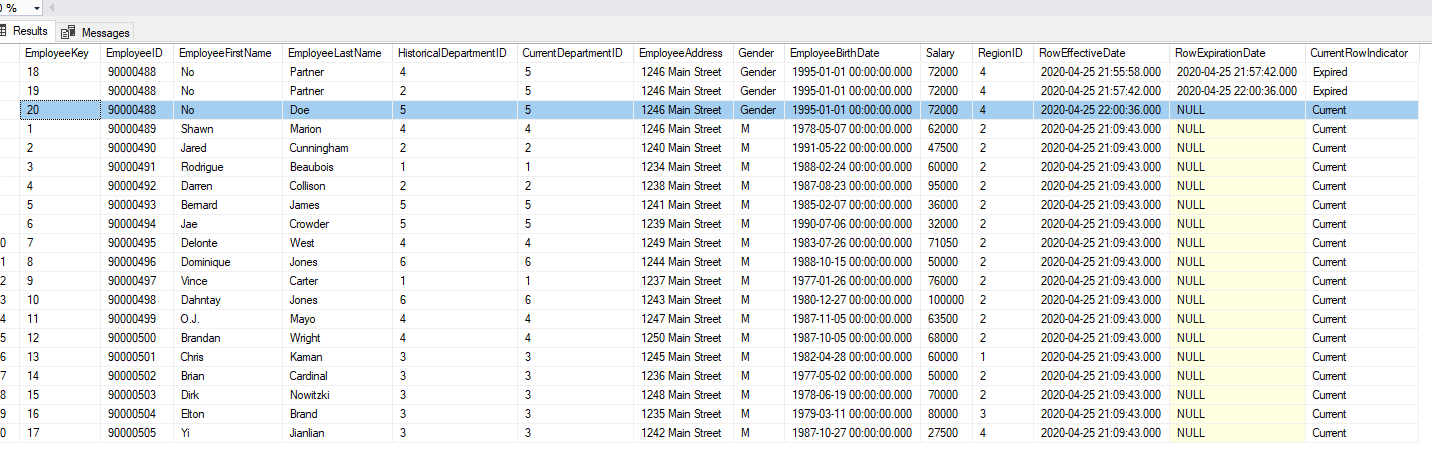
Again update the DepartmentID to 5 and EmployeeLastName to Doe for EmployeeID 90000488 in the Employee source file and save it. Submit the screenshot of the Employee dimension after running your model.



-- RESPONSE 4 and 5

SELECT \* FROM DimEmployee

ORDER BY EmployeeID;



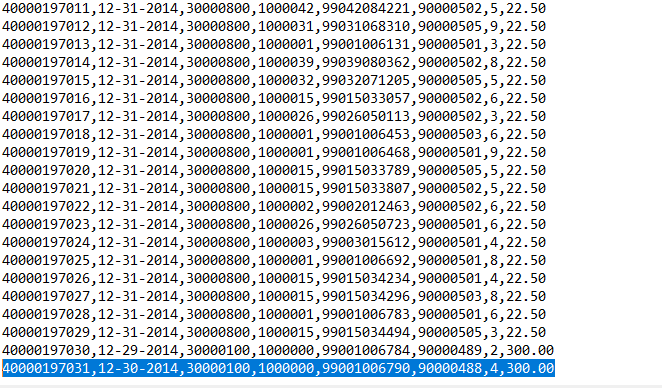
**RESULT:**

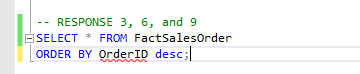
**The department ID of employee 90000488 was updated to 5. HistoricalDepartmentID now contains the department ID of 4, 2, and 5 for the three rows that keeps record of employee 90000488’s information. CurrentDepartmentID column now only contains 5 as the current department ID. The new row (with EmployeeKey 20) is now the current employee row as indicated in the CurrentRowIndicator column, and both the previous employee rows (with EmployeeKey 18 and 19) now have an expired date. Last name of employee is now Doe.**

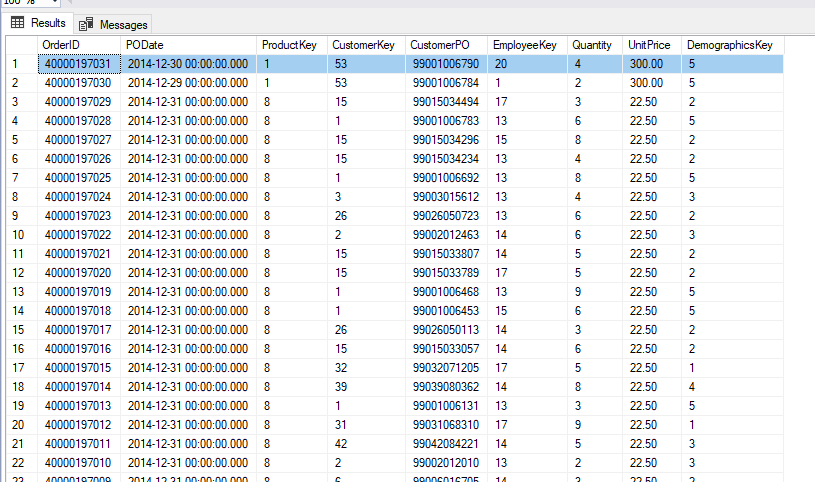
**Response 6:**

Add following transactions in the SalesOrder source file. Submit the screenshot of the fact table after running your model.

40000197031,12-30-2014,30000100,1000000,99001006790,90000488,4,300.00





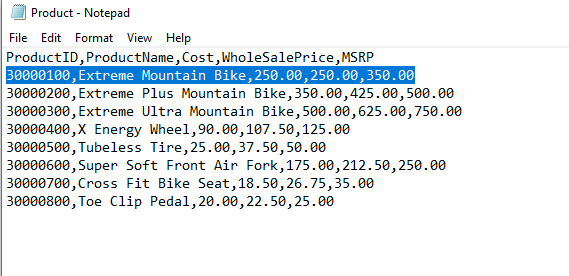


**RESULT:**

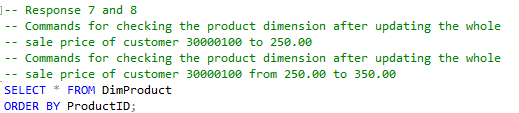
**OrderID of the order is, as listed in the above table, 40000197031, with an order date (PODate) of 12-30-2014, Customer product order (CustomerPO) of 99001006790, Quantity of 4, and a unit price (UnitPrice) of 300.00. Because the order had a product id of 30000100, customer id of 1000000, and employee id of 90000488, the values for ProductKey, CustomerKey, and EmployeeKey came out as 1, 53, and 20, respectively. Demographics key of the customer is 5, given that that was the customer’s CurrentDemographicsKey value.**

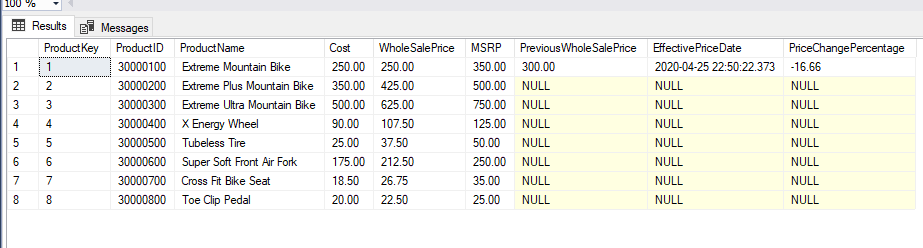
**Response 7:**

Update the WholeSalePrice to 250.00 for ProductID 30000100 in the Product source file and save it.



Submit the screenshot of the Product dimension after running your model.



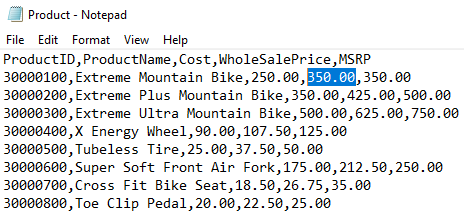


**RESULT:**

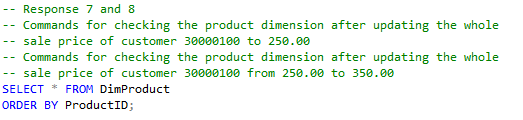
**Wholesale Price for product 30000100 was updated from 300 to 250. Running this change through the data flow resulted in a value of 300.00 (Previous product wholesale price) for the PreviousWholeSalePrice column for product 30000100. Furthermore, the effective date and time for the price change was filled into the EffectivePriceDate column, and the percentage of wholesale price change (-16.66%) was recorded in the PriceChangePercentage column.**

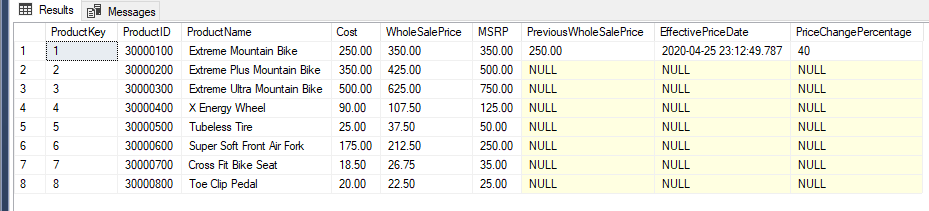
**Response 8:**

Again update the WholeSalePrice to 350.00 for ProductID 30000100 in the Product source file and save it.



Submit the screenshot of the Product dimension after running your model.





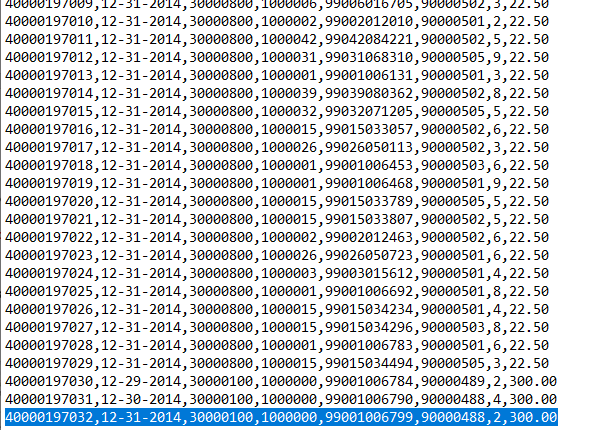
**RESULT:**

**Wholesale Price for product 30000100 was updated from 250 to 350. Running this change through the data flow resulted in a value of 250.00 for the PreviousWholeSalePrice column for product 30000100. Furthermore, the effective date and time for the price change was filled into the EffectivePriceDate column, and the percentage of wholesale price change (40%) was recorded in the PriceChangePercentage column.**

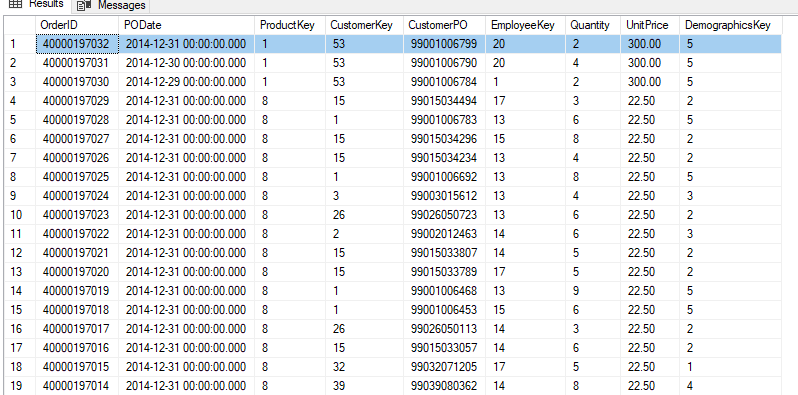
**Response 9:**

Add following transactions in the SalesOrder source file. Submit the screenshot of the fact table after running your model.

40000197032,12-31-2014,30000100,1000000,99001006799,90000488,2,300.00







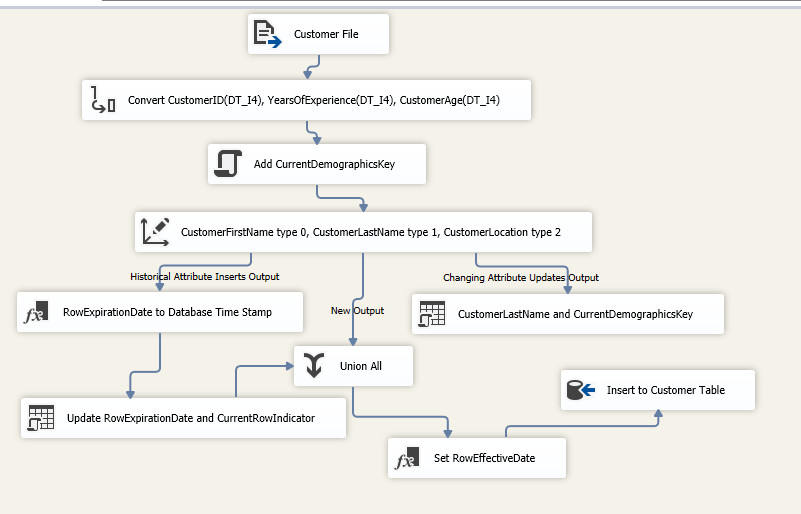
**RESULT:**

**OrderID of the order is, as listed in the above table, 40000197032, with an order date (PODate) of 12-31-2014, Customer product order (CustomerPO) of 99001006799, Quantity of 2, and a unit price (UnitPrice) of 300.00. Because the order had a product id of 30000100, customer id of 1000000, and employee id of 90000488, the values for ProductKey, CustomerKey, and EmployeeKey came out as 1, 53, and 20, respectively. Demographics key of the customer is 5, given that that was the customer’s CurrentDemographicsKey value.**

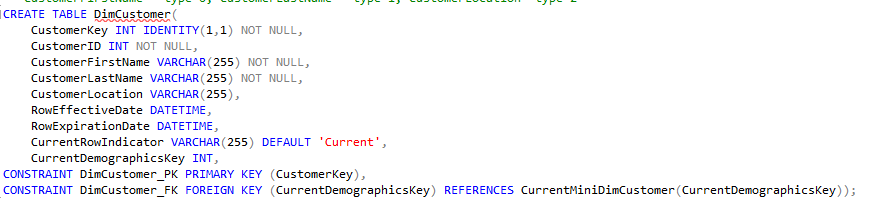
**DATA FLOW CHART EXPLANATIONS**

In a Word document stepwise briefly describe the data flow of Customer, Product, and Employee dimensions and also the SalesOrder fact table.

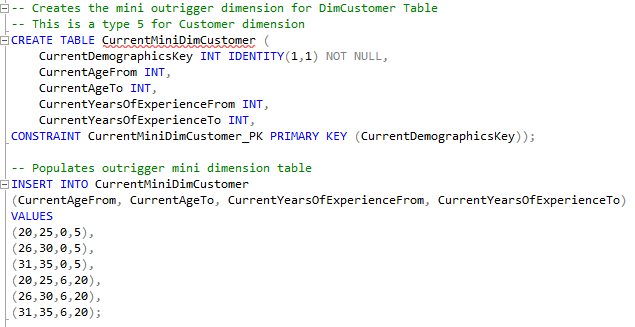
**Customer Table Data Flow Explanation:**

****

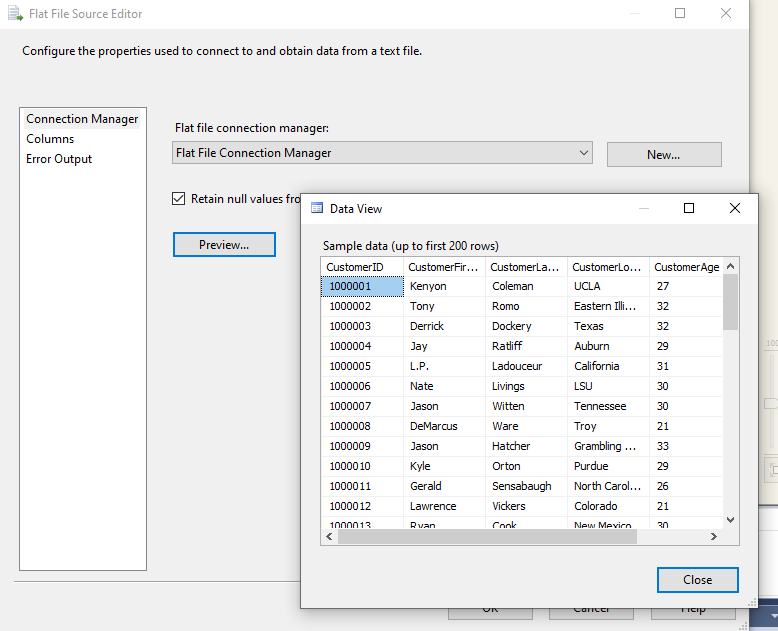
1. **A customer dimension table, DimCustomer, must first be created. I have chosen to set the surrogate and natural keys as integer type. Below are the SQL commands necessary for building the DimCustomer Table.**

****

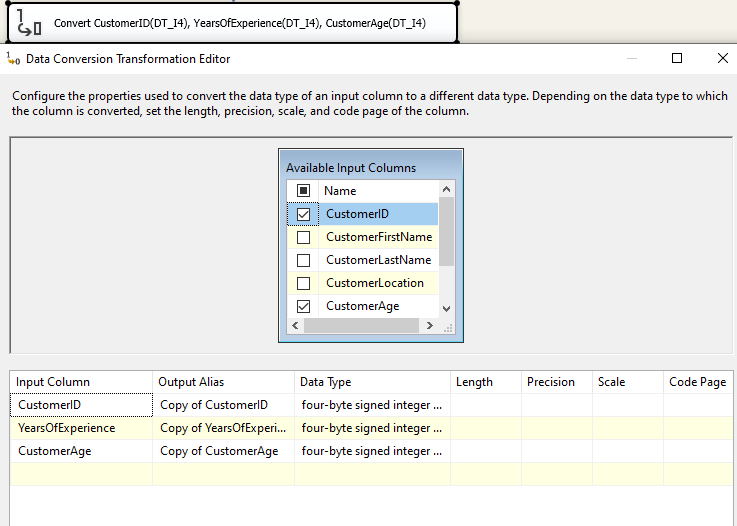
1. **The outrigger dimension, CurrentMiniDimension, must also be created as it is needed to populate the CurrentDemographicsKey column of the DimCustomer table. Below are the SQL commands necessary to build the outrigger.**

****

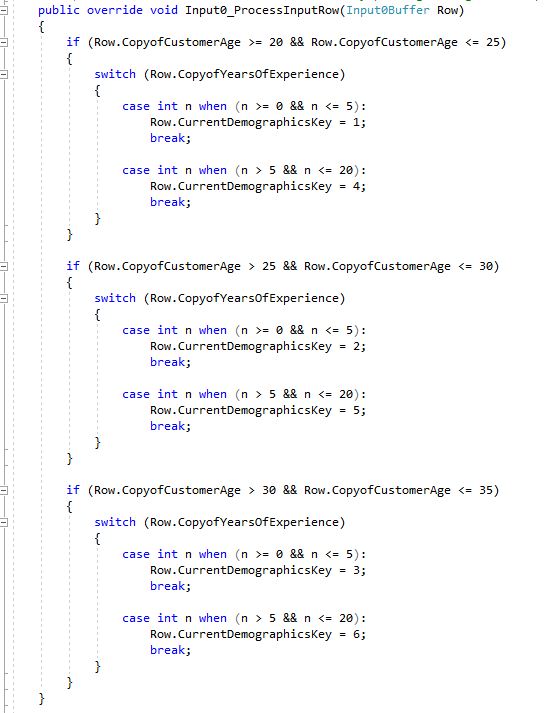
1. **Source tool, Flat File Source, was used to import the Customer table data. The tool is renamed to “Customer File”.**

****

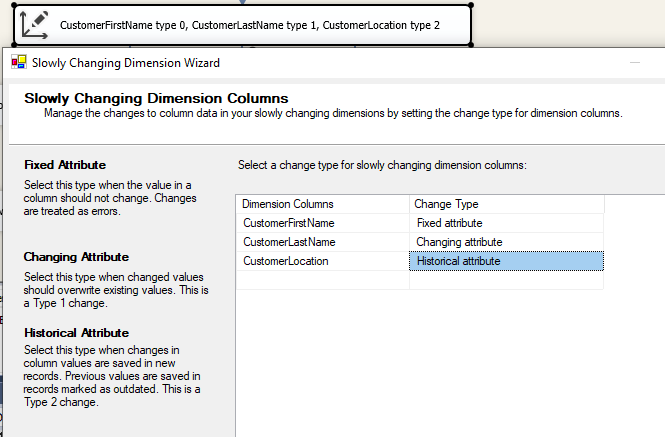
1. **Since all imported data are strings, some of them needs to be converted with the Data Conversion Tool. In this case, I have chosen to convert CustomerID, YearsOfExperience, and CustomerAge into the four-byte signed integer type. This conversion tool is renamed to “Convert CustomerID(DT\_I4), YearsOfExperience(DT\_I4), CustomerAge(DT\_I4)”.**

****

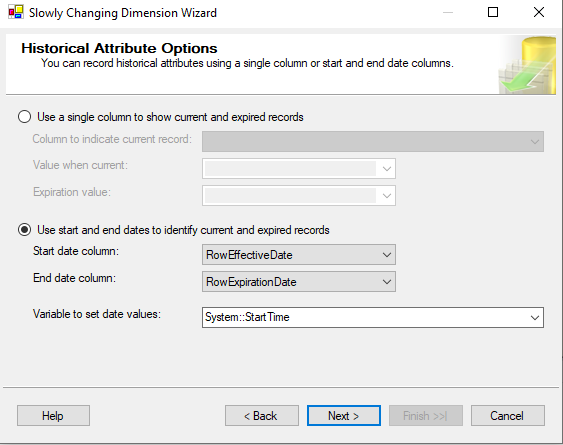
1. **After the data were converted, they were sent to the Script Component tool where the CurrentDemographicsKey was added to each customer row. This tool was renamed “Add CurrentDemographicsKey”.The C# script below was used to calculate and insert the demographics keys for each customer row.**

****

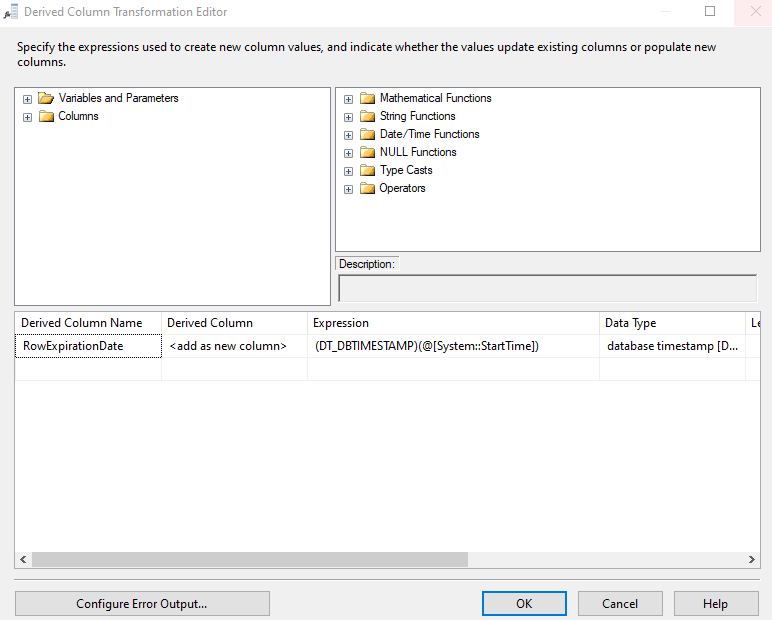
1. **All combined data from step 5 were then transferred to the Slowly Changing Dimension tool. Here, the CustomerFirstName column of DimCustomer is set to “Fixed Attributes” (type 0), CustomerLastName column is set to “changing attribute” (type 1), and CustomerLocation is set to “Historical Attribute” (type 2). This tool was renamed to “CustomerFirstName type 0, CustomerLastName type 1, CustomerLocation type 2”**

****

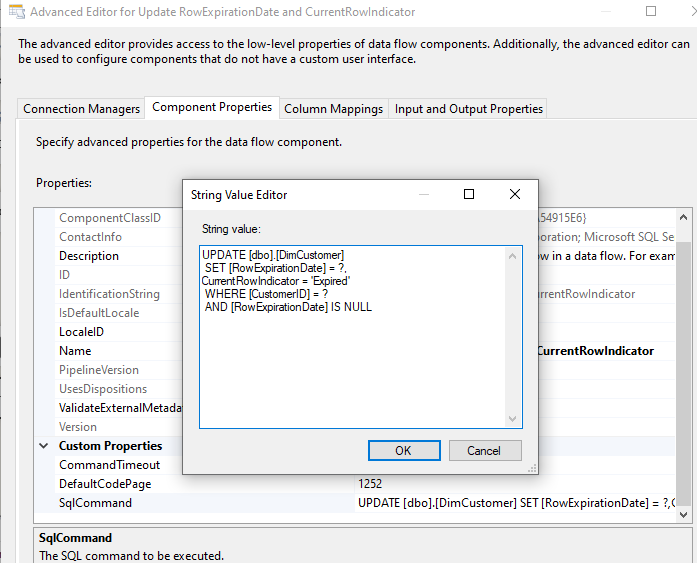
1. **Still on the Slowly Changing Dimension tool, the RowEffectiveDate and RowExpirationDate columns were specified to use the “System::StartTime” function whenever they appear in the DimCustomer table. After finishing all interfaces in the Slowly Changing Dimension tool, several more tools were automatically created, and they branched out into three different paths. The next three steps will refer to these three paths (Steps 8 = type 2 path), 9 = type 1 path and 10 = combining path).**



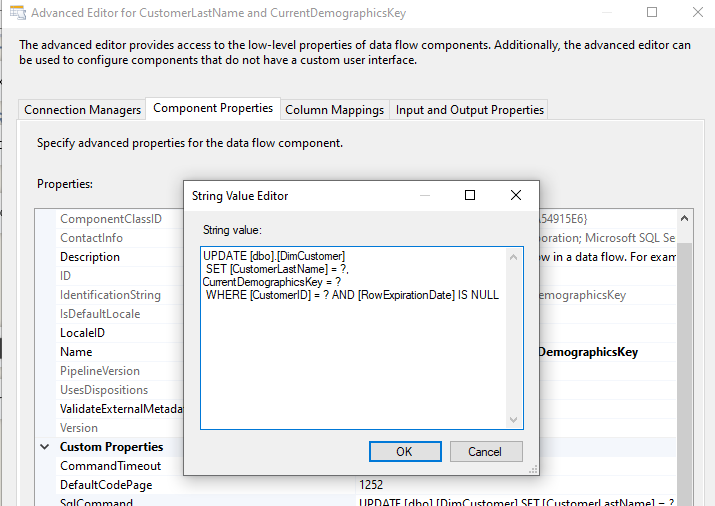
1. **The data flow utilizes the functionality of the type 2, or the historical attribute, path if a new location data is detected in a row in the source file.** 
   1. **For this path, the rows with updated location data first enter the Derived Column tool which basically adds a new column. In this case, the new column is RowExpirationDate, and the current date and time value will be entered into it. The column is then combined with the rows with updated locations. This tool was renamed to “RowExpirationDate to Database Time Stamp”**



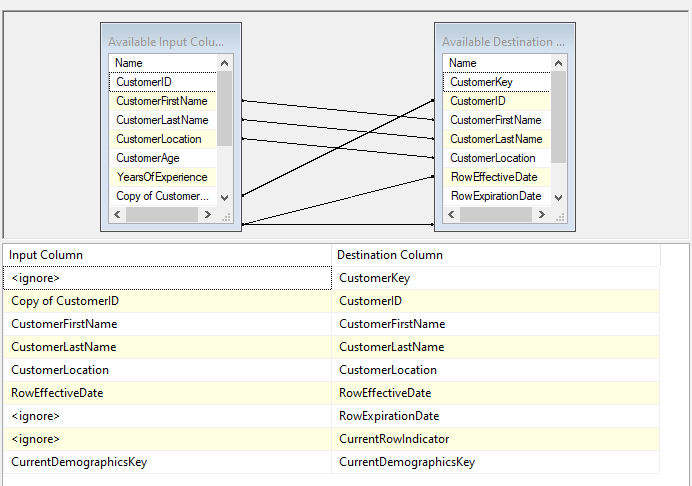
* 1. **The combined data from step 8a are then transferred into the OLE DB Command tool, which will also insert the “Expired” value into the CurrentRowIndictor column for these rows using the below SQL commands. This tool is renamed to “Update RowExpirationDate and CurrentRowIndicator”.**



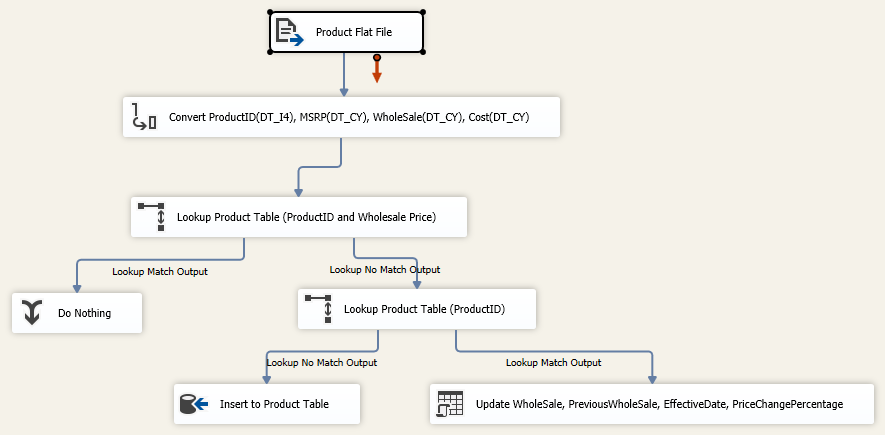
1. **The data flow utilizes the type 1, or the changing attribute, path when an update happens on the customer’s last name only.**
   1. **From the slowly changing dimension tool, the data flows into an OLE DB Command tool, where the below script changes the last name of a customer when an update on the customer’s last name is detected. This tool was renamed to “CustomerLastName and CurrentDemographicsKey”.**

****

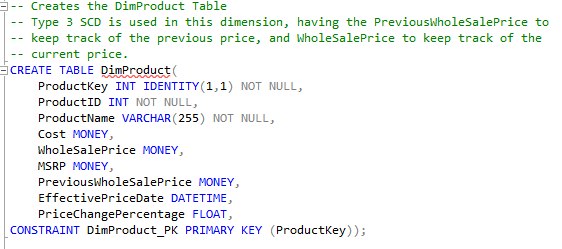
1. **The updated, and new, rows from step 8b and all other rows flowing in from the previous slowly changing dimension tool in step 6 – 7 combines through the Union All tool.**
2. **All the combined data flows into another Derived Column tool, renamed as “Set RowEffectiveDate”, which sets the current date/time for all new rows in the RowEffectiveDate column of DimCustomer.**
3. **Lastly, all data flows into the OLE DB Destination tool, renamed to ”Insert to Customer Table” to insert all data into the DimCustomer dimension. The mappings for the data to columns are shown below. Note that the “Copy of..” in front of a input column name simply means that the column’s data type has been converted previously in step 4.**

****

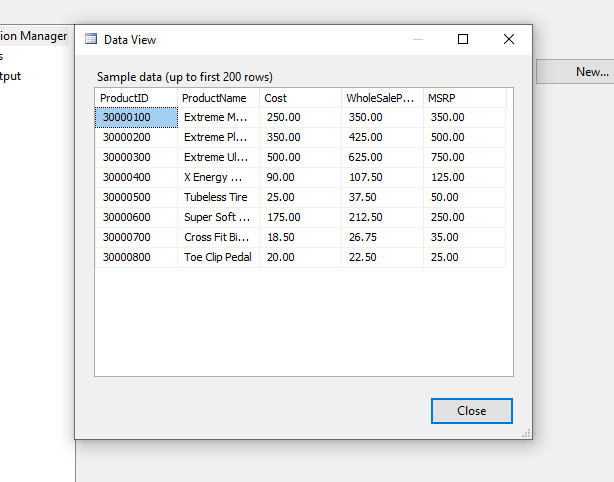
**Product Table Data Flow Explanation:**

****

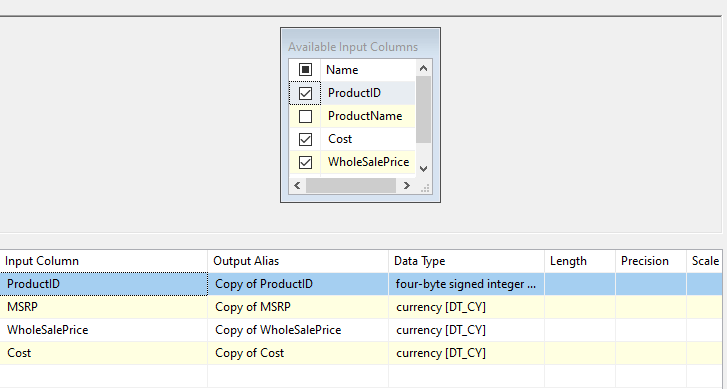
1. **The Product table needs to be created for the database. Below are the SQL commands.**

****

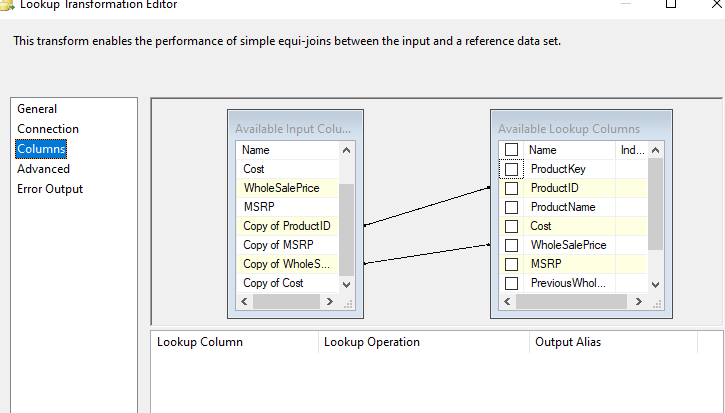
1. **Flat File Source tool (renamed: “Product Flat File”) was used to retrieved the Product data from the text file.**

****

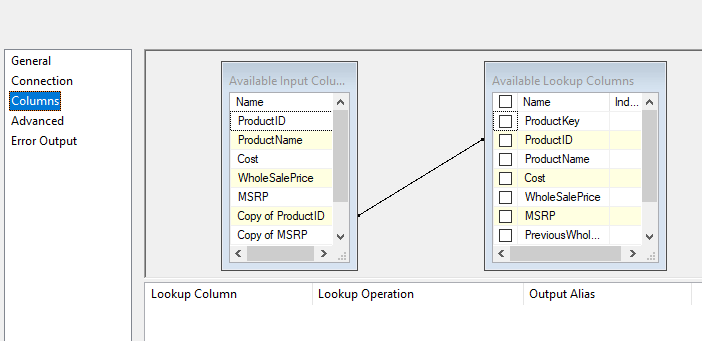
1. **All string data from the Product flat file were needed to be converted to its correct typing using the Data Conversion tool (renamed: “Convert ProductID(DT\_I4), MSRP(DT\_CY), WholeSale(DT\_CY), Cost(DT\_CY)”). ProductID was converted to integer type, and MSRP, WholeSalePrice, and Cost were converted to currency type.**

****

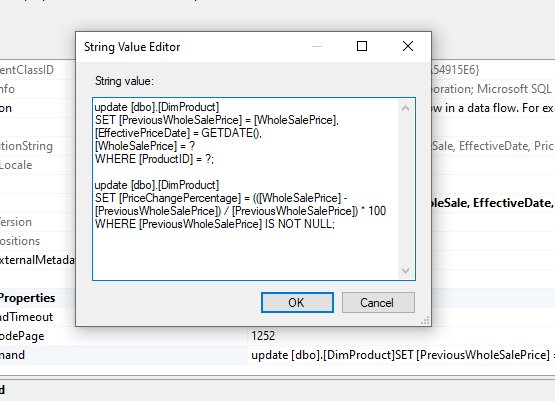
1. **Converted data from step 3 were transferred to a Lookup tool (renamed: “Lookup Product Table (ProductID and Wholesale Price)”). Using the mapping, shown below, Lookup retrieved all rows from the converted Product data where the row’s ProductID and WholeSale values matched the ProductID and WholeSale values of the rows from the actual DimProduct table. Rows with matching values for both columns suggested no new pricing updates for wholesale and/or no new products, whereas rows with non-matching values suggested the opposite. Using the Lookup tool’s “Redirect rows to no match output” option, matching rows will be redirected to a Union All tool where nothing will happen (signaling no new data), whereas non-matching rows will be redirected to another Lookup tool for further sorting, as this may suggest new data.**

****

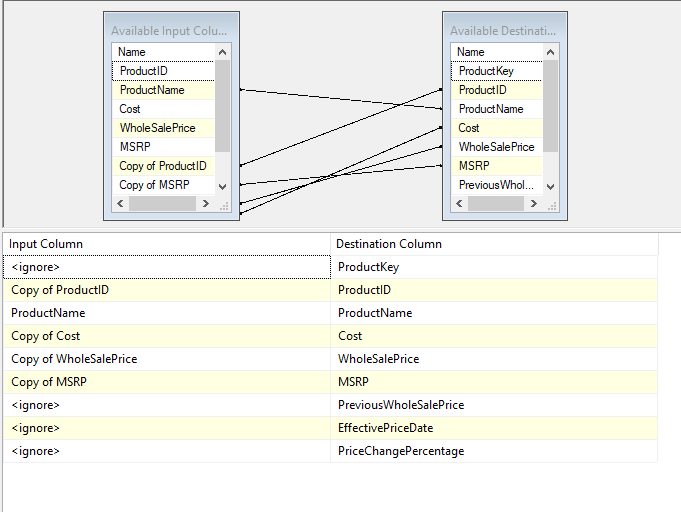
1. **Non-matching rows will be transferred to another Lookup tool (renamed: “Lookup Product Table (ProductID)”) where the rows will be checked for whether their ProductID matches the ProductID of the rows in the actual DimProduct table. If none matches, this would indicate that the rows are new products, and will be transferred directly to an OLE DB Destination tool where it will be added to the DimProduct table. If ProductID in any row matches, the matching rows will be transferred to an OLE DB Command tool.**



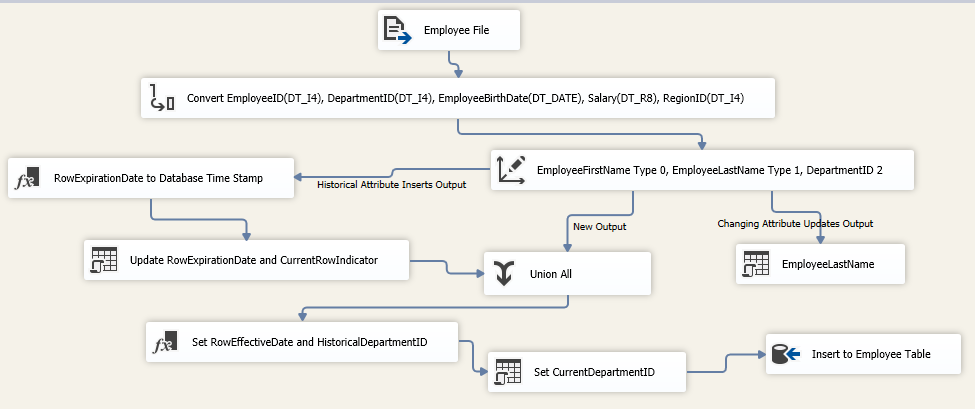
1. **The rows from step 5 that had a match with DimProduct’s ProductID will be transferred to an OLE DB Command tool (named: “Update WholeSale, PreviousWholeSale, EffectiveDate, PriceChangePercentage”). Being transferred to this tool would mean that there was an update on the row’s wholesale price. Using the SQL commands, shown below, the wholesale price will be assigned to the PreviousWholeSalePrice column, and the new wholesale price will be assigned to the WholeSalePrice column. Next, the EffectivePriceDate column will be assigned the current date. Lastly, a percent change is calculated (using (WholeSalePrice-PreviousWholeSalePrice)/PreviousWholeSalePrice) for the change in wholesale price and will be assigned to the PriceChangePercentage column. The data in this row will replace the data in the DimProduct table row that had the matching ProductID value.**



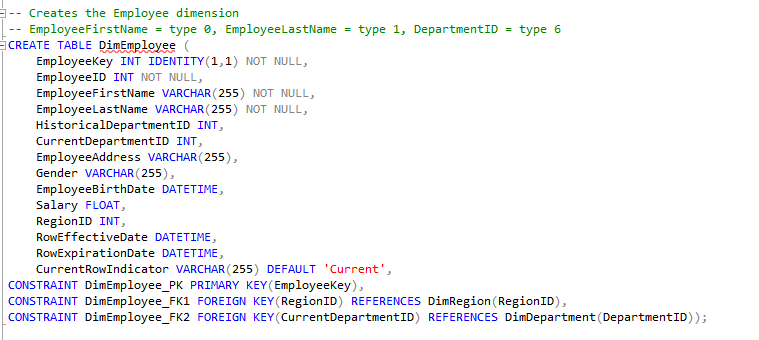
1. **All data will eventually be inserted into the DimProduct table using the OLE DB Destination tool (renamed: “Insert to Product Table”). Below shows the mapping of the input columns to the table columns.**

****

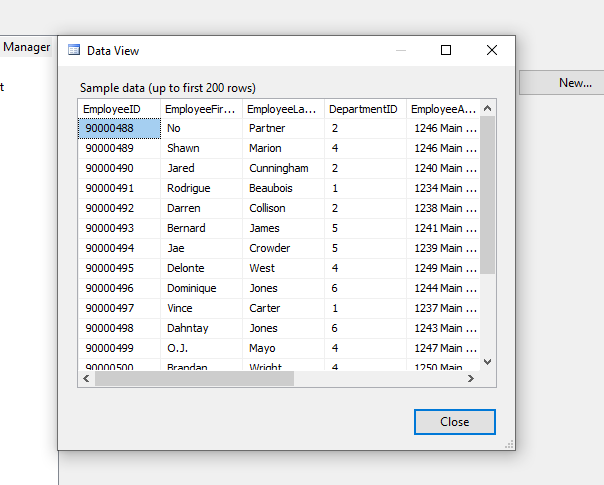
**Employee Table Data Flow Explanation:**

****

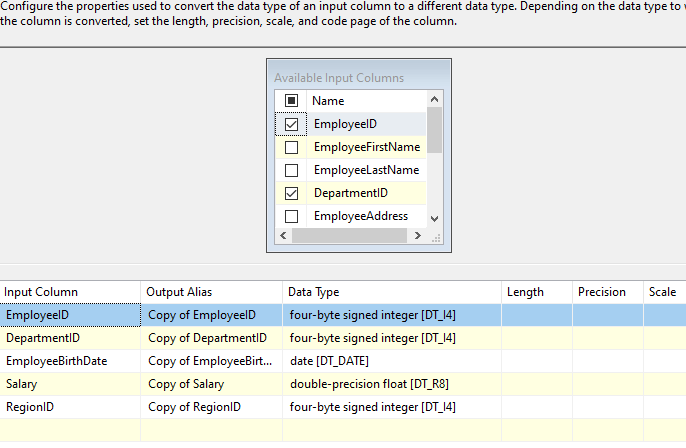
1. **A employee dimension table, DimEmployee, must first be created.**

****

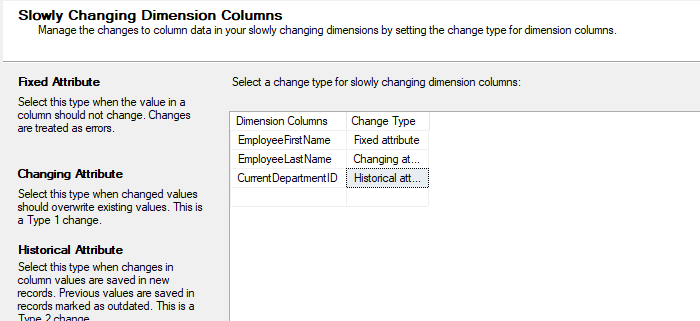
1. **Source tool, Flat File Source, was used to import the Employee table data. The tool is named “Employee File”.**

****

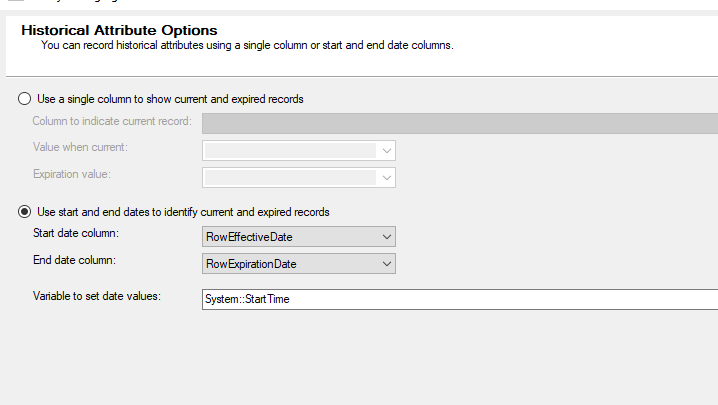
1. **Since all imported data are strings, some of them needs to be converted with the Data Conversion Tool (rename: “Convert EmployeeID(DT\_I4), DepartmentID(DT\_I4), EmployeeBirthDate(DT\_DATE), Salary(DT\_R8), RegionID(DT\_I4)”). In this case, I have chosen to convert EmployeeID, DepartmentID, and RegionID into the four-byte signed integer type. EmployeeBirthDate and Salary were converted to date and double-precision float, respectively.**



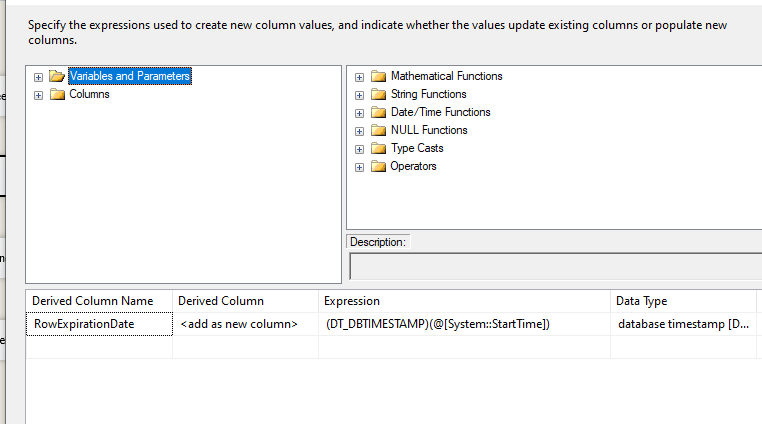
1. **All combined and converted data is sent to the Slowly Changing Dimension tool (rename: “EmployeeFirstName Type 0, EmployeeLastName Type 1, DepartmentID 2”). Here, the EmployeeFirstName column of DimEmployee is set to Fixed Attributes (type 0), EmployeeLastName column is set to changing attribute (type 1), and CurrentDepartmentID is set to Historical Attribute (type 2).**

****

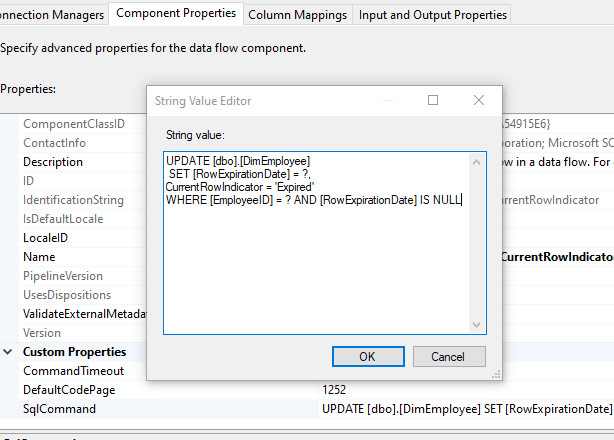
1. **Still on the Slowly Changing Dimension tool, the RowEffectiveDate and RowExpirationDate columns are specified to use the “System::Start Time” function for whenever they appear in the table. After finishing all interfaces in this tool, several more tools will be automatically created, and will branch out from this tool into three different paths. These three paths will be explained in the next 3 steps (step 6 = type 2 path, 7 = type 1 path, and 8 = combining path).**

****

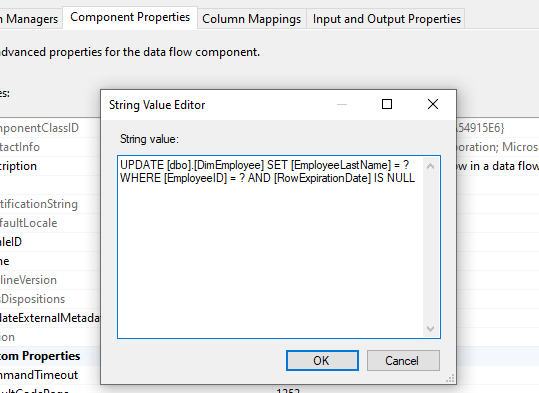
1. **The data flow utilizes the functionality of the type 2, or the historical attribute, path if a new location data is detected in a row in the source file.** 
   1. **For this path, the rows with updated location data first enter the Derived Column tool (renamed: “RowExpirationDate to Database Time Stamp”) which basically adds a new column. In this case, the new column is RowExpirationDate, and the current date and time value will be entered into it. The column values will then be combined with the rows with updated location values.**

****

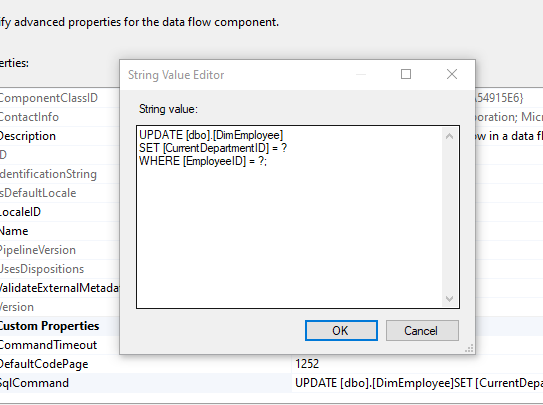
* 1. **The combined data will then be transferred into the OLE DB Command tool (renamed: Update RowExpirationDate and CurrentRowIndicator), which will also insert the “Expired” value into the CurrentRowIndicator column for these rows using the below SQL commands.**

****

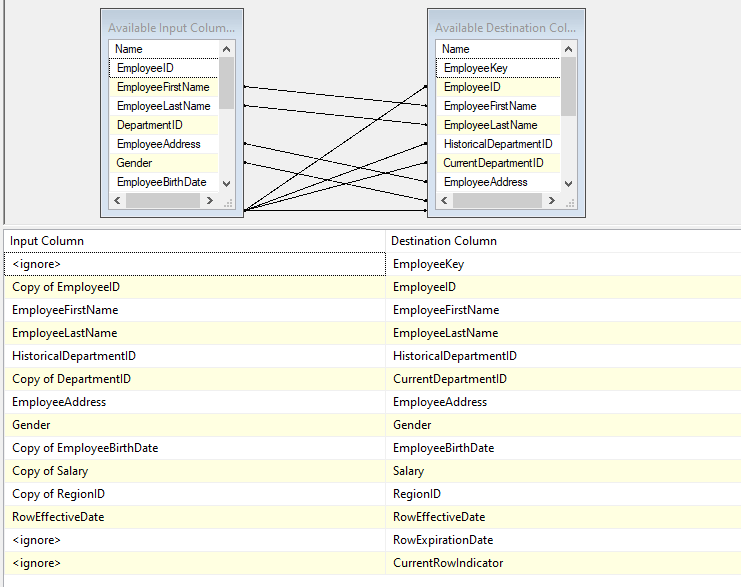
1. **The data flow utilizes the type 1, or the changing attribute, path when an update happens on the employee’s last name only.**
   1. **From the slowly changing dimension tool, the data flows into an OLE DB Command tool (renamed: “EmployeeLastName”) where the below script changes the last name of an employee when an update on the employee’s last name is detected.**

****

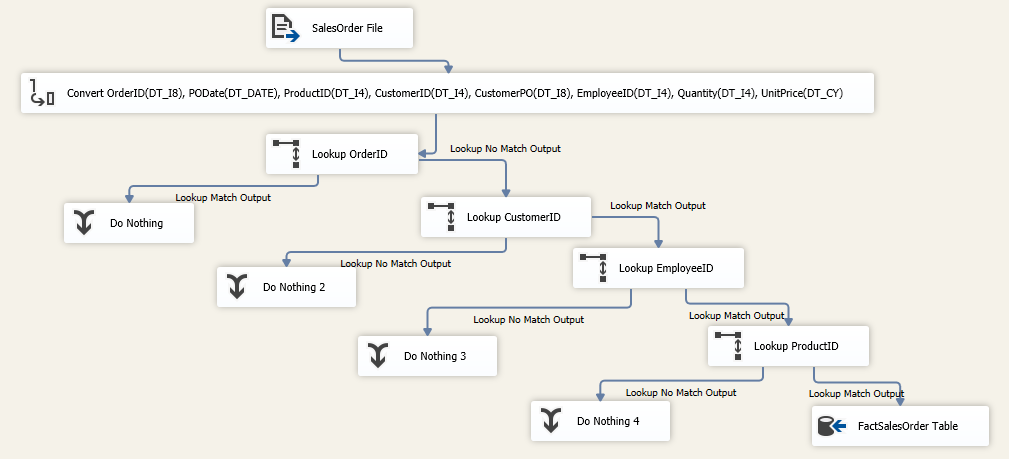
1. **The updated, and new, rows from step 6b and all other rows flowing in from the previous slowly changing dimension tool in step 4 – 5 combines through the Union All tool.**
2. **All the combined data flows into another Derived Column tool, named “Set RowEffectiveDate and HistoricalDepartmentID”, which sets theRowEffectiveDate column to the current date/time, as well as set the HistoricalDepartmentID column of DimEmployee to DepartmentID.**
3. **The data is then transferred to another OLE DB Command tool (name: “Set CurrentDepartmentID”). Using the SQL command, shown below, the CurrentDepartmentID column will be filled with the DepartmentID.**

****

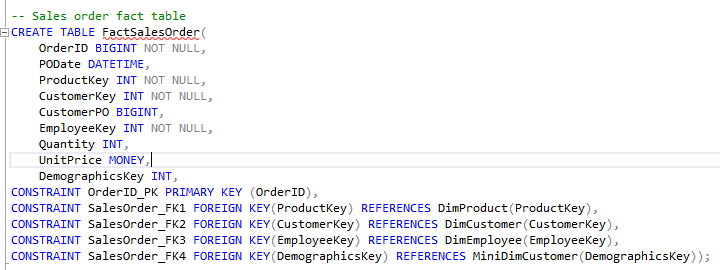
1. **Lastly, all data flows into the OLE DB Destination tool, renamed to ”Insert to Employee Table” to insert all data into the DimEmployee dimension. The mappings are shown below.**

****

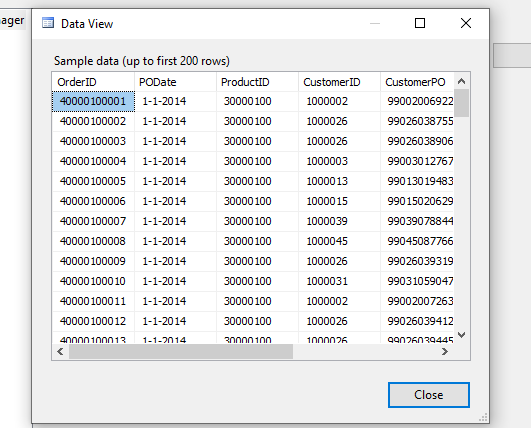
**SalesOrder Table Data Flow Explanation:**



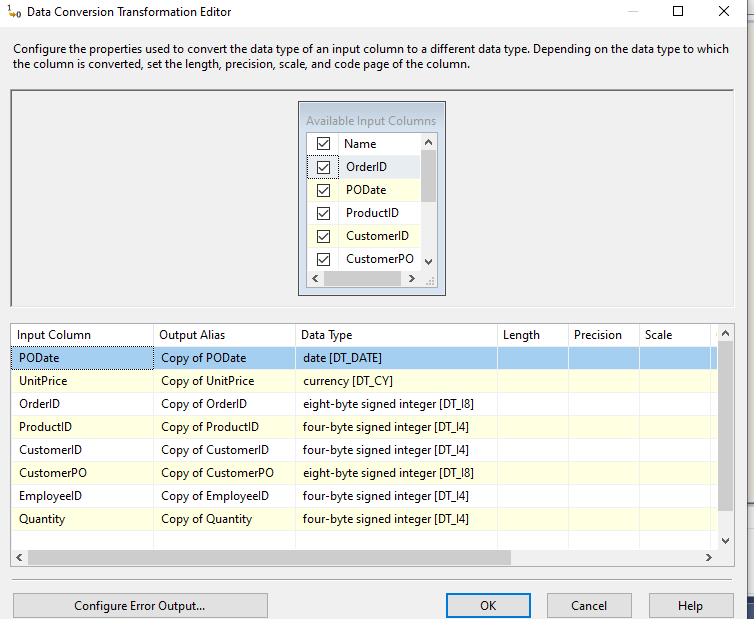
1. **The FactSalesOrder fact table must first be created.**

****

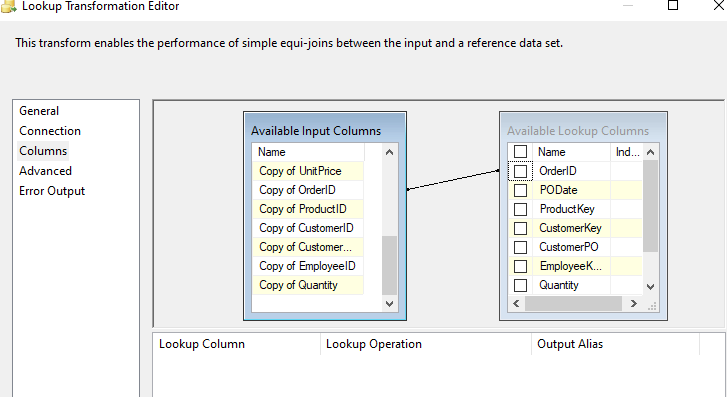
1. **The Flat File Source tool was used to import SalesOrder Data.**

****

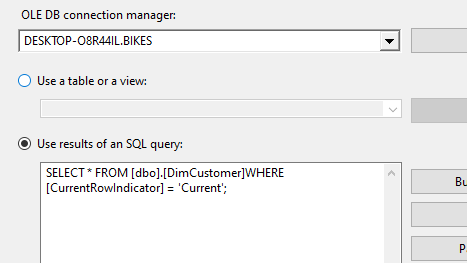
1. **The string data from SalesOrder flat file was converted to suitable types using Data Conversion tool (renamed: “Convert OrderID(DT\_I8), PODate(DT\_DATE), ProductID(DT\_I4), CustomerID(DT\_I4), CustomerPO(DT\_I8), EmployeeID(DT\_I4), Quantity(DT\_I4), UnitPrice(DT\_CY)”). In this case, ProductID, CustomerID, EmployeeID, and Quantity were converted to integer type; OrderID and CustomerPO were converted to large integer type; PODate was converted to date type; and UnitPrice was converted to currency type.**

****

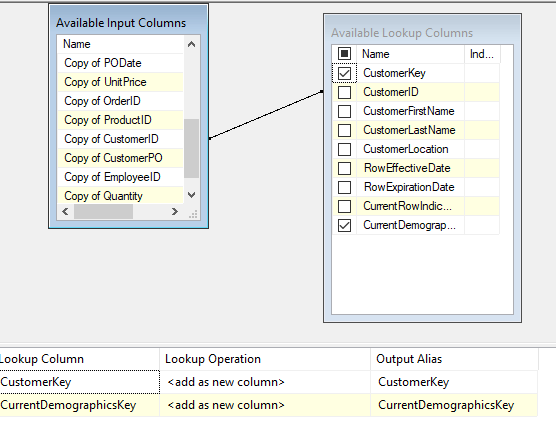
1. **All converted data were transferred to a Lookup tool (renamed: “Lookup OrderID”). Mapping below shows that the tool specifically isolates the rows of converted data that has an OrderID that does not match with the OrderID of the rows in the FactSalesOrder table. Nothing will be done to the rows with matching OrderID. Rows with non-matching OrderID will be transferred to the next Lookup tool.**

****

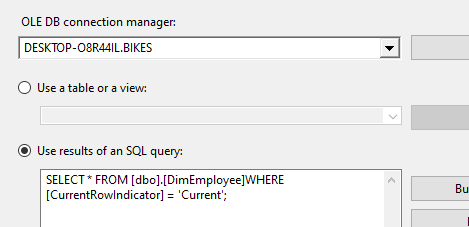
1. **Rows transferred from the previous step will be sorted again in this Lookup tool (renamed: “Lookup CustomerID”). This Lookup tool will only work with a subset of data from the DimCustomer table that has the “Current” row indicator, shown below.**

****

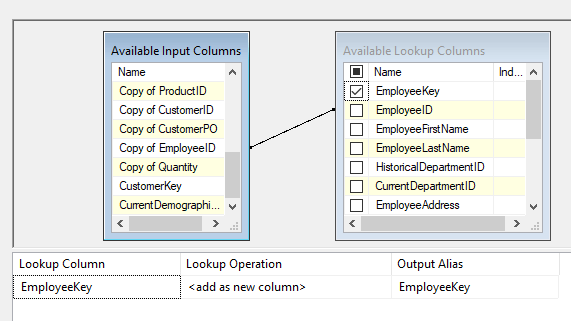
**The tool will isolate the transferred rows that has a CustomerID matching that of the CustomerID from the rows in the subset. The Lookup tool will then return the CustomerKey values and CurrentDemographicsKey values from the rows in the subset corresponding to the matched CustomerIDs. All matching transferred rows in this step, along with the CustomerKey and CurrentDemographicsKey column values, will then be transferred to the next Lookup tool. If there are no matches, nothing happens.**

****

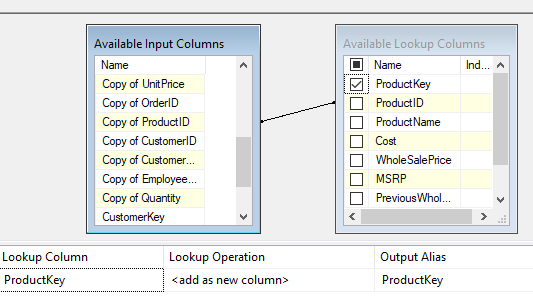
1. **Rows transferred from the previous step will be sorted again in this Lookup tool (renamed: “Lookup EmployeeID”). This Lookup tool will only work with a subset of data from the DimEmployee table that has the “Current” row indicator, shown below.**

****

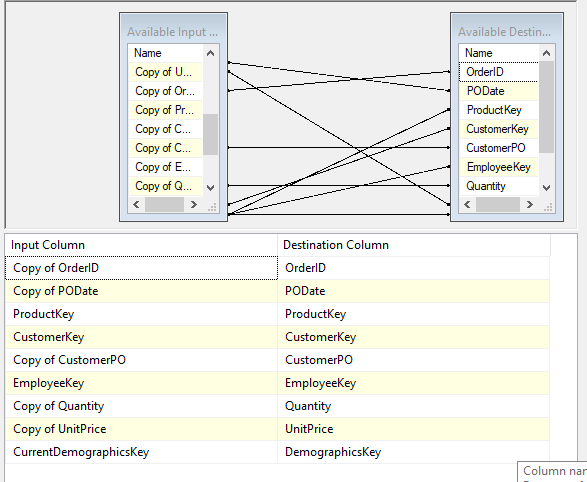
**The tool will isolate the transferred rows that has an EmployeeID matching that of the EmployeeID of the rows in the subset. The Lookup tool will then return the EmployeeKey values from the rows in the subset corresponding to the matched EmployeeIDs. All matching transferred rows in this step, along with the EmployeeKey values, will then be transferred to the next Lookup tool. If there are no matches, nothing happens.**

****

1. **Rows from the previous step will be sorted again in this Lookup tool (renamed: “Lookup ProductID”). The tool will isolate the transferred rows that has a ProductID matching that of the ProductID of the rows in the DimProduct table. The Lookup tool will then return the ProductKey values from the rows in the DimProduct table corresponding to the matched ProductIDs. All matching transferred rows in this step, along with the ProductKey values, will then be transferred to the OLE DB Destination. If there are no matches, nothing happens.**

****

1. **In place of all the ID keys, the OLE DB Destination tool (renamed: “FactSalesOrder Table”) will return all the Surrogate key values from DimProduct, DimCustomer, and DimEmployee that were returned from the Lookup tools. Resulting column attribute are shown below.**

****