**Retail Store Data Warehouse Project**

**Topics covered:**

* Data Extraction and Transformation from Source (Flat files or OLTP)
* Loading data to Staging
* Designing Star Schema in Data Warehouse (Facts and Dimension tables)
* Implementing SCD(Type 0, Type 1 and Type 2)
* CDC (Change Data Capture)
* Error Handling in SSIS

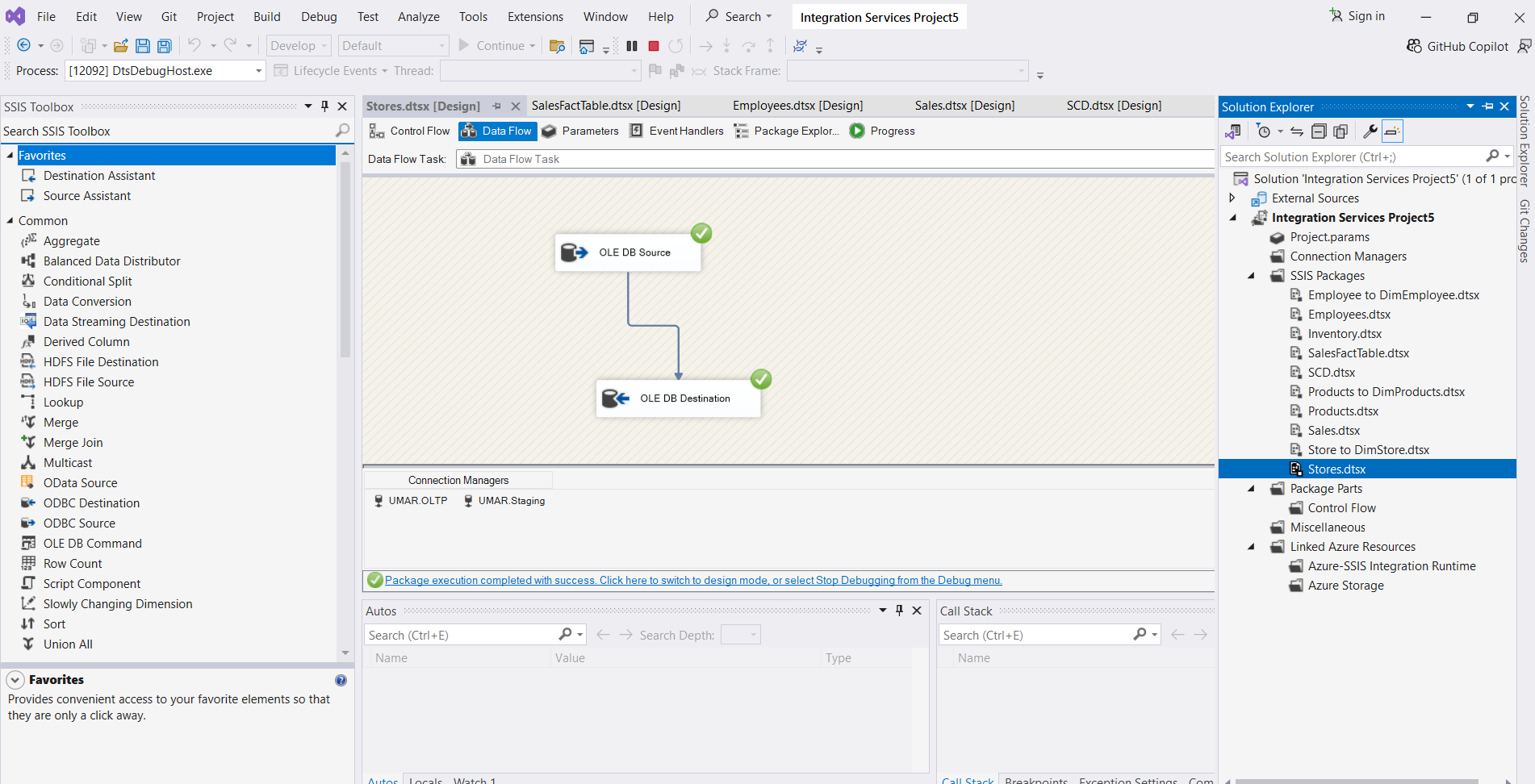
**Data Extraction:**

We were given 5 flat files named Stores, Products, Employees, Inventory and Sales. The data provided in flat files were uncleaned and we must transform it according to our destination so that it can be loaded successfully. It is a necessary part of ETL, unlike in ELT, transformation is not necessary before writing the data into destination. We can say that ETL follows schema-on-write approach while ELT follows schema-on-read approach.

In the start I consumed much of my time in data transformation and later Sir Farrukh told me to skip this step and move on to next tasks. So I took cleaned data and manually inserted it and now we can say that is our “Source”.

**Loading Data to Staging:**

It was a simple step because everything was matching between source and destination.



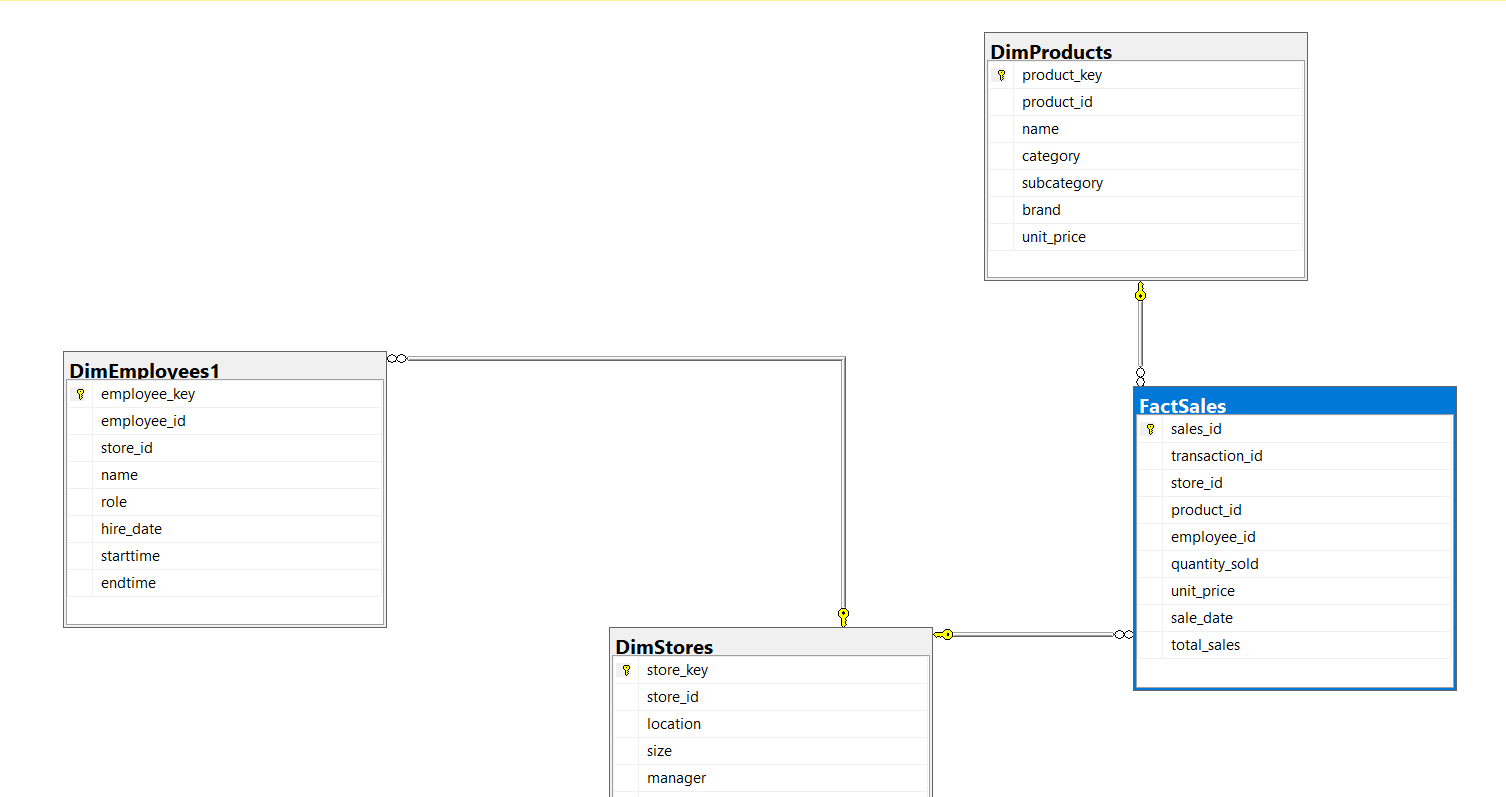
Similarly, I performed same method to load other tables. I created separate packages for every task so that it will be easy to debug if there’s any error.

**Designing Star Schema in Data Warehouse:**

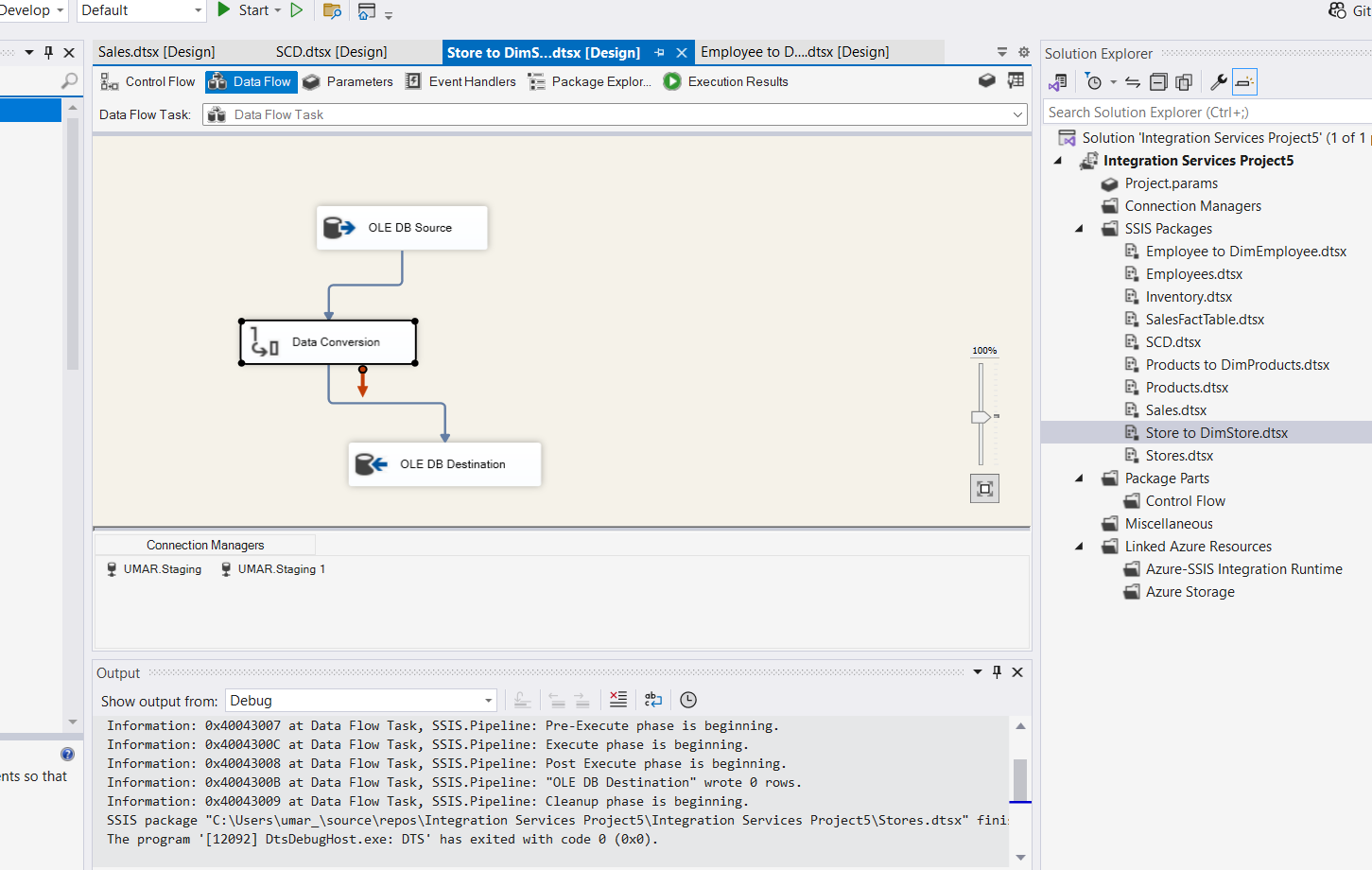
Now that we have successfully moved the data from OLTP to Staging, next step is to design star schema. I used same database for “Staging” and “Datawarehouse” because I was facing some errors.

I created three Dimension tables named “DimEmployee”, “DimStore” and “DimProducts” using the same as structure as in “Staging” or “OLTP”, and one Fact table named “FactsSales”.

Following is the visual representation of Star Schema.

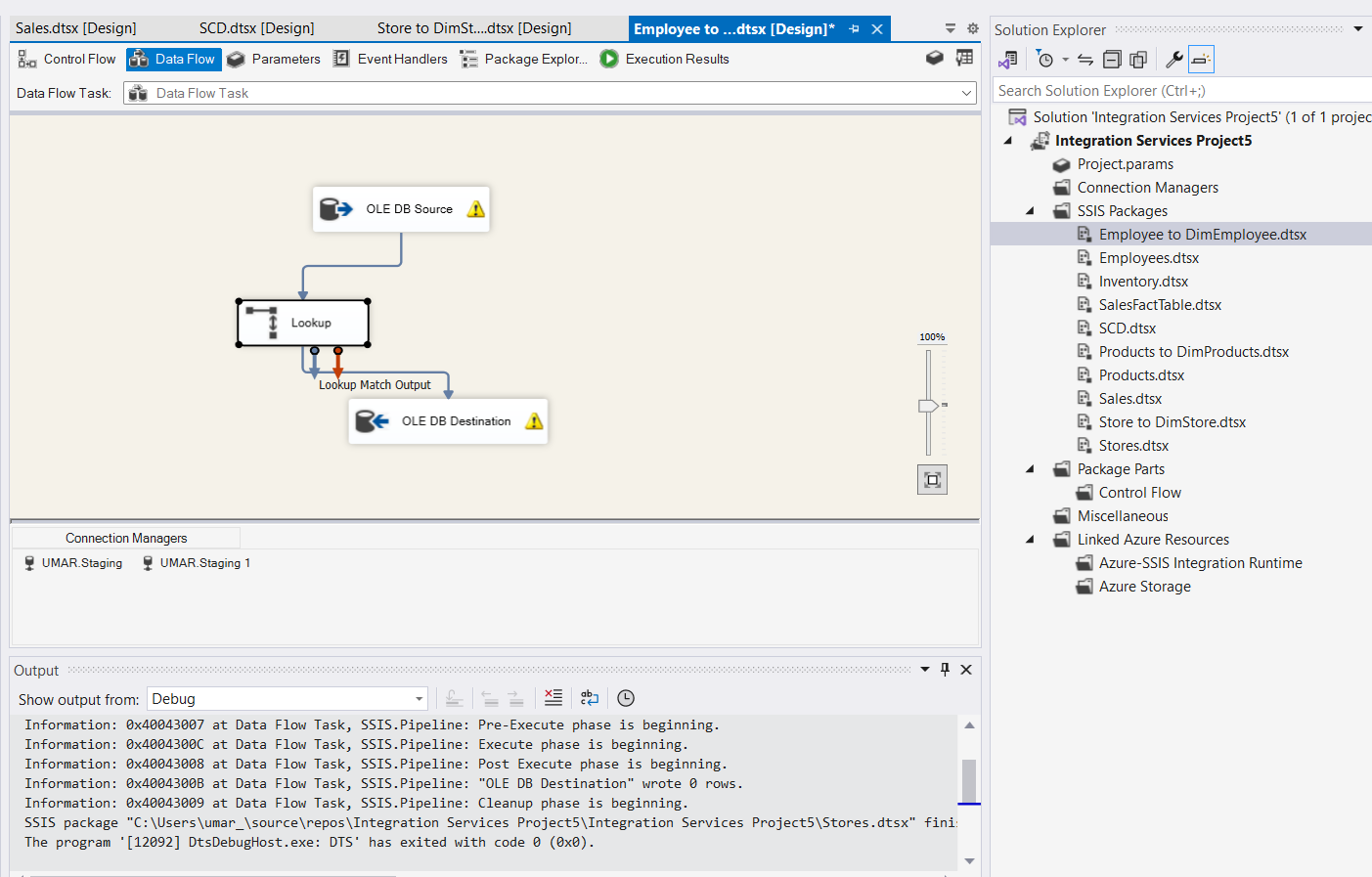


**Transferring data from “Stores” to “DimStore” (SSIS package):**



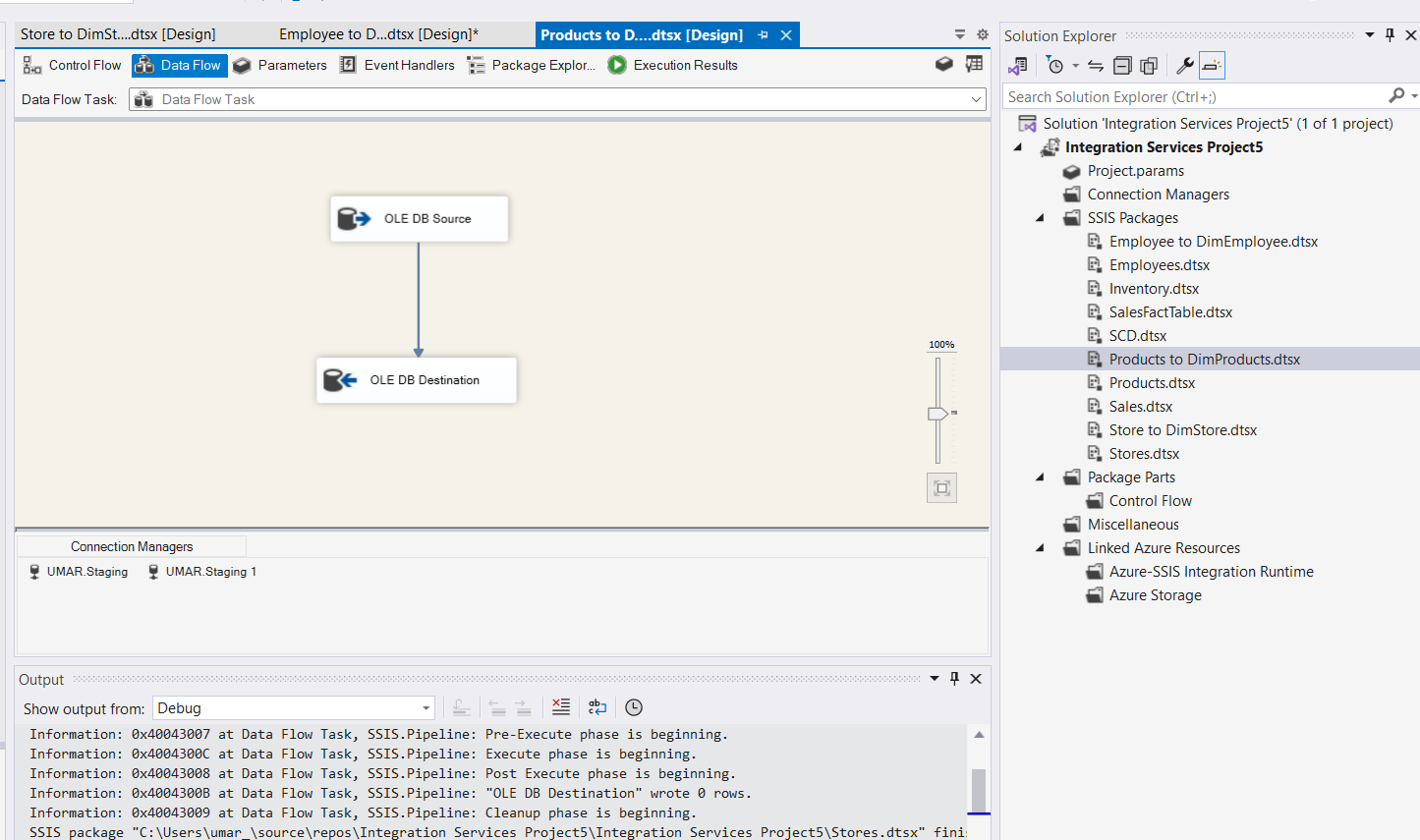
Added a “Data Conversion” box because it was giving an error due to difference of data types.

**Transferring data from “Employees” to “DimEmployees1” (SSIS package):**

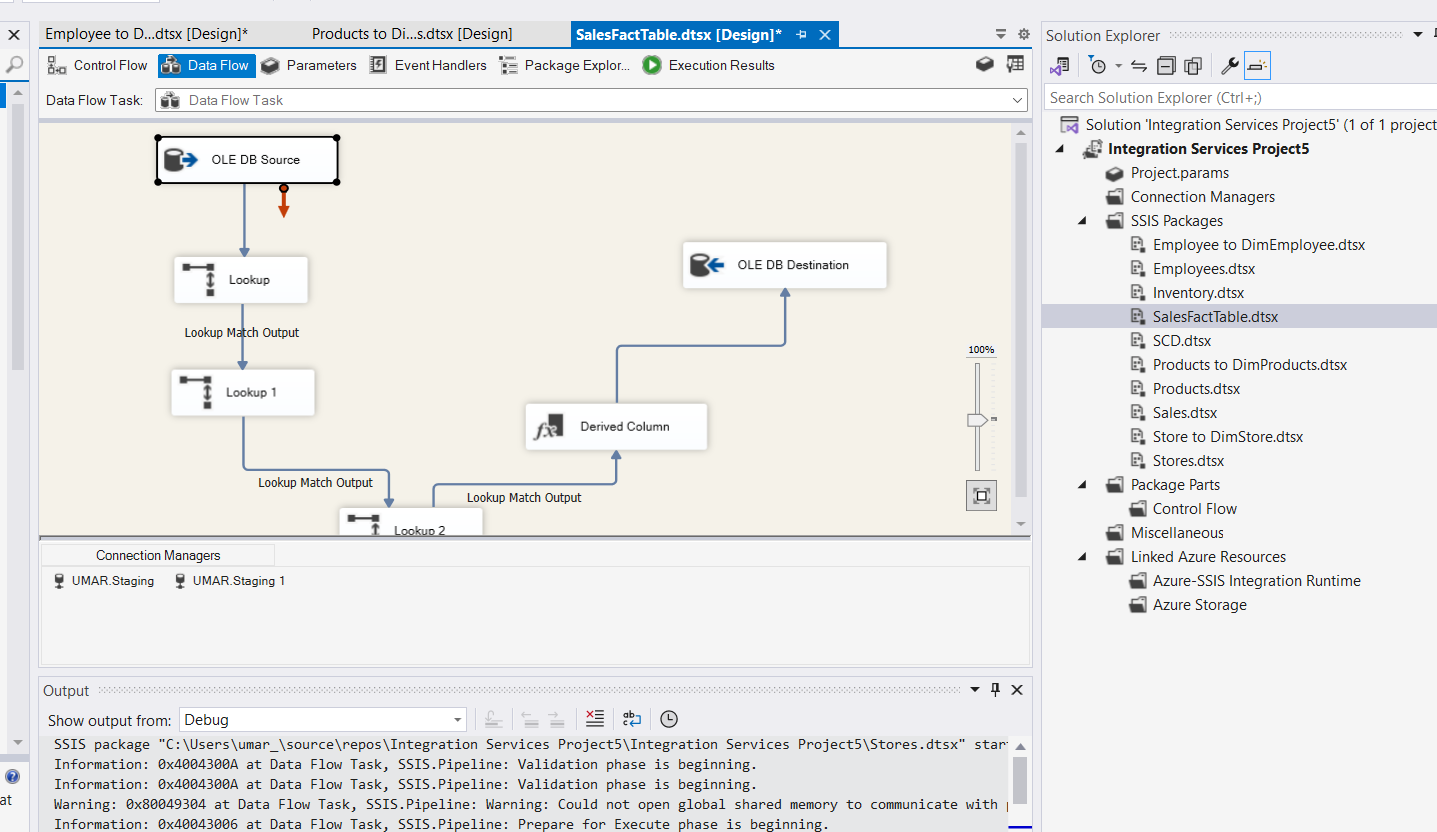


I used “Lookup” tab to bring the matching data between the Employees and DimEmployees based on “store\_id”.

**Transferring data from “Products” to “DimProducts” (SSIS package):**



**Transferring data to “SalesFactTable” table:**



I used following SQL query to join on two tables:

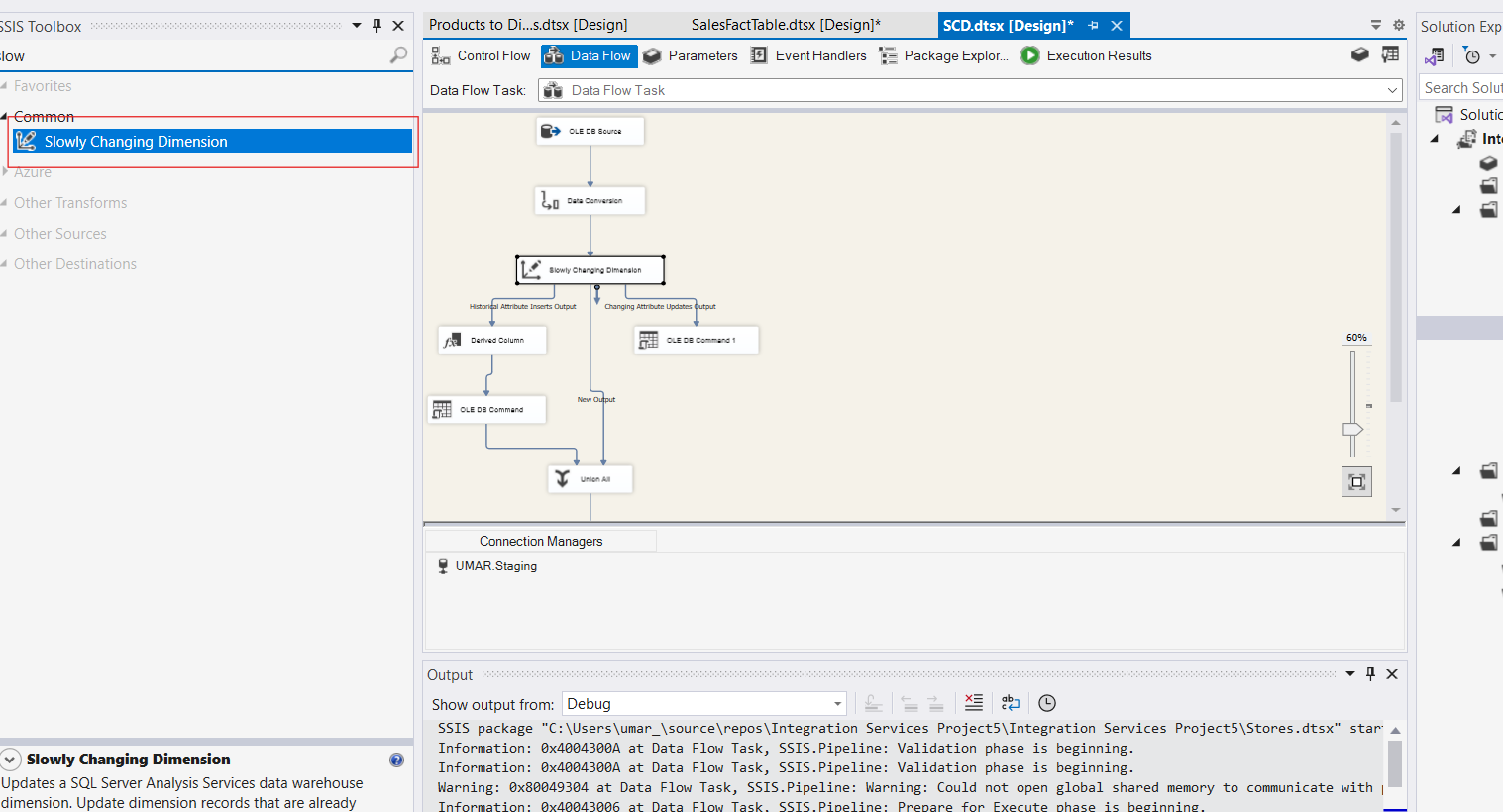
SELECT s.transaction\_id, s.store\_id, s.product\_id, s.quantity, s.sale\_date, s.cashier\_id, p.unit\_price

FROM Sales s

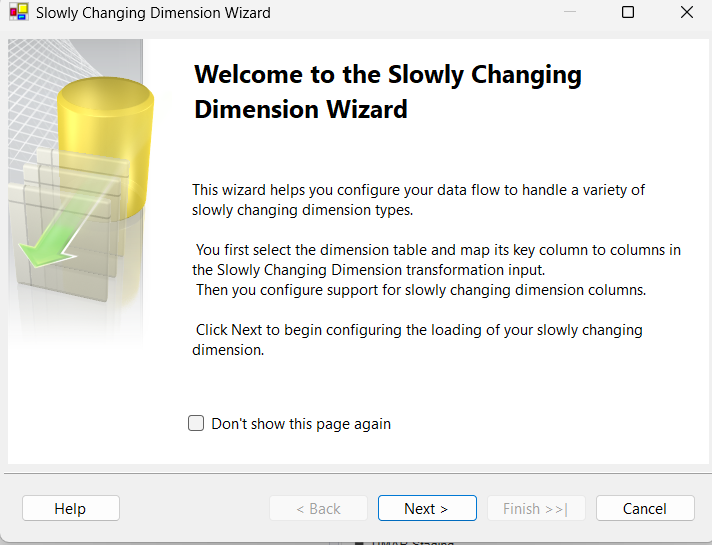
JOIN Products p ON s.product\_id = p.product\_id;

Then I used “Lookup” tabs to match data based on “store\_id” from rest of the tables.

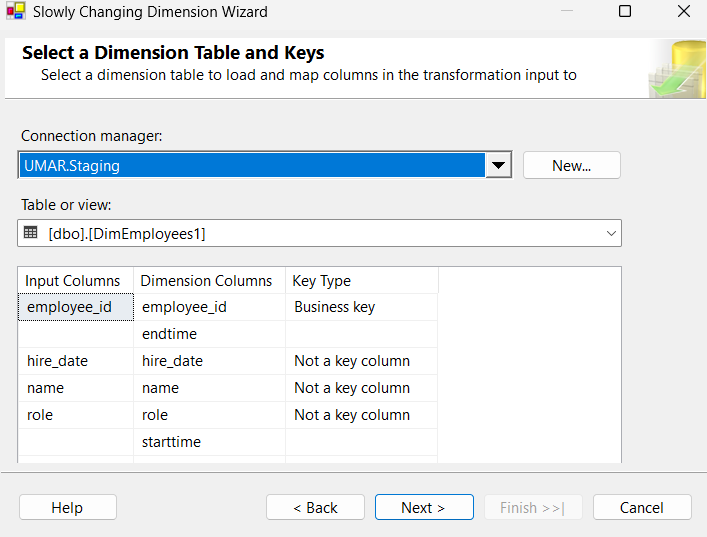
**Implementing SCD(Type 1 and Type 2):**



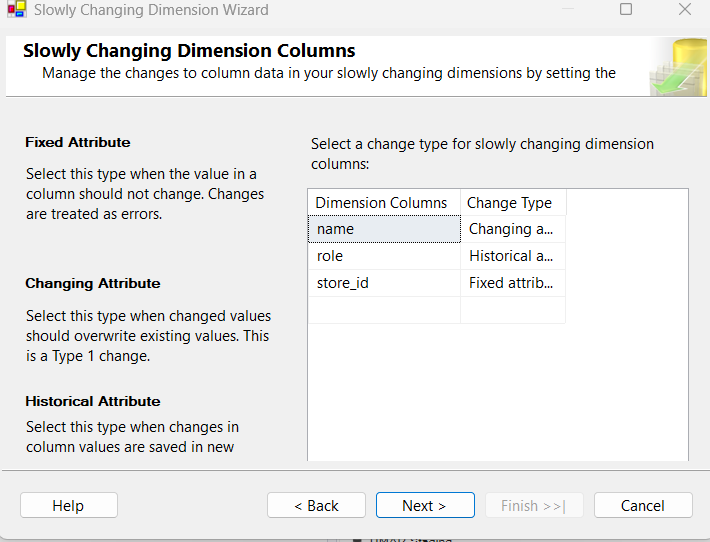
Drag and drop Slowly Changing Dimension.



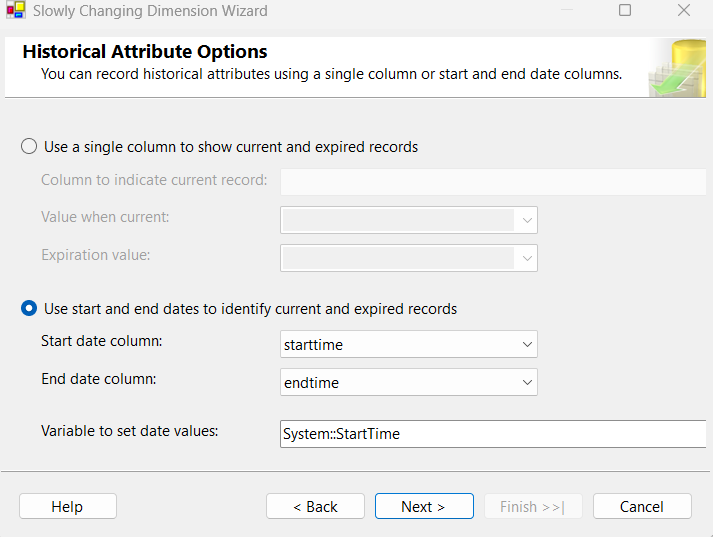
Click on “Next”.



Select the destination table where you want to implement SCD.

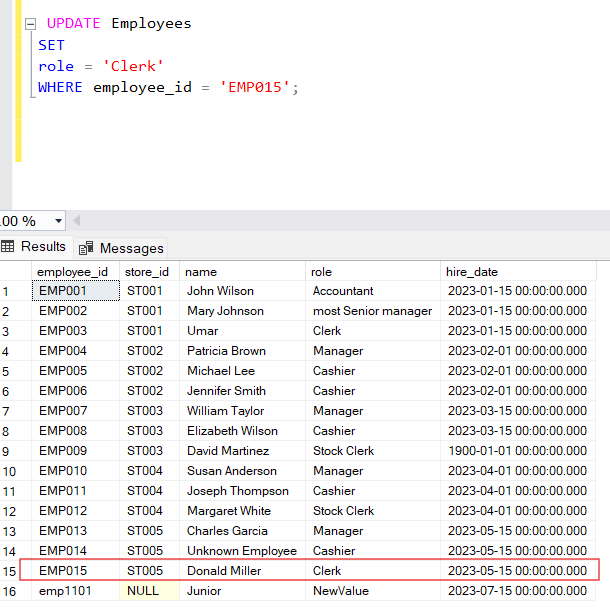


Select the columns to apply SCD type 0,1 and 2 respectively.



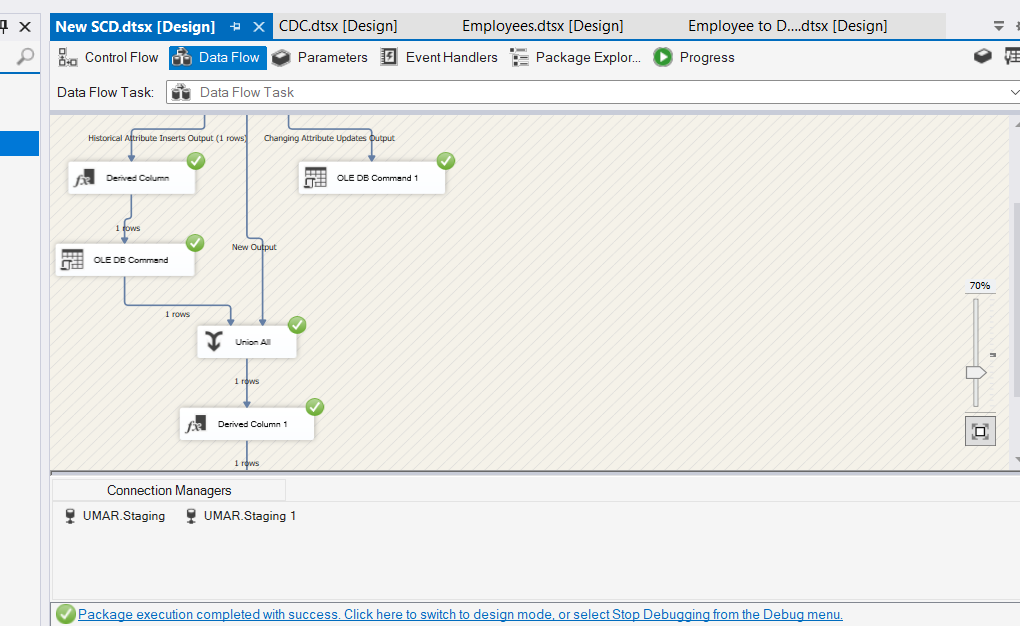
You must need two columns “start\_time” and “end\_time” in your dimension table to track the changes in historical attribute (type 2).

Now let’s test it in the SQL if it’s working or not.

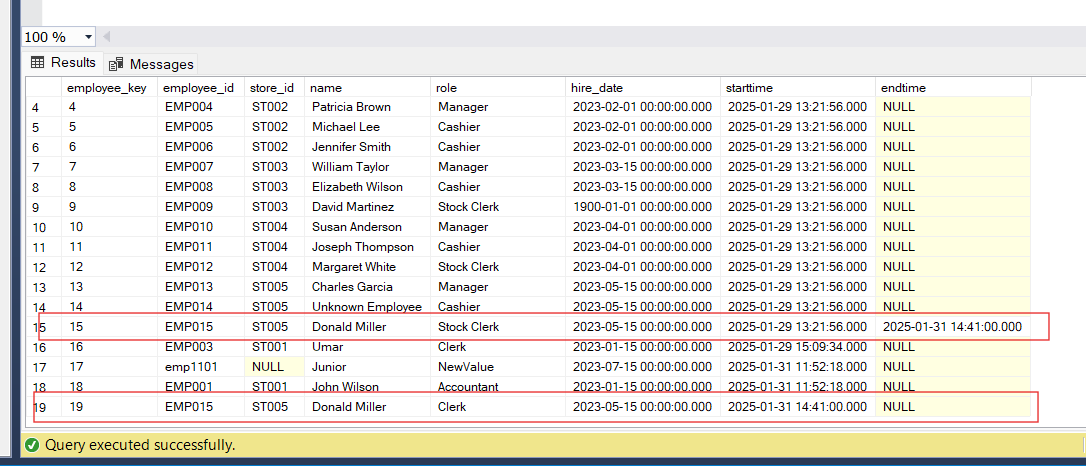


We updated the “role” to “Clerk” where “employee\_id ” is “EMP015” in our source table.

Now lets run the SSIS package to check if the record is updated in destination or not along with previous record, because in SCD type 2, the previous record is also kept along with the new one.



We can see that its only taking one row into the destination. After running this package we will check our dimension table.

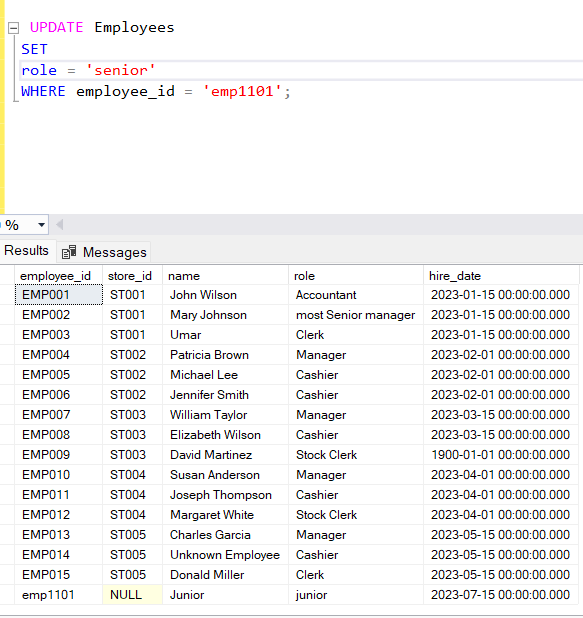


As we can see that old record is kept with its end time and new record is active now.

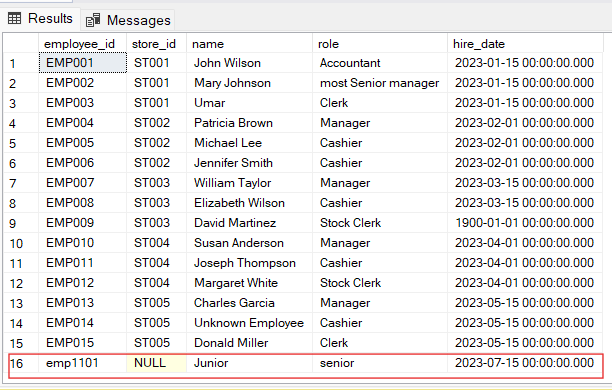
**Implementing CDC (Change Data Capture) Insert, Update and Delete:**

In CDC, if we make any changes in our source table, it will not check only changes, rather then it will truncate the existing destination table and bring all the data from source. In this way we cannot miss any changes made in our source table and track everything precisely.

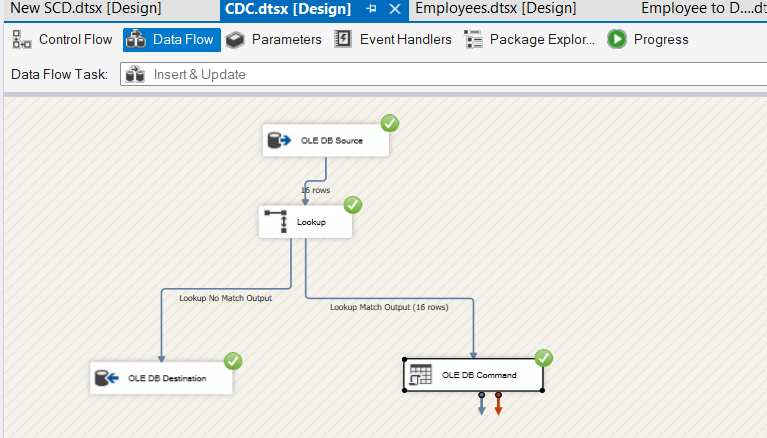
**Updating:**



Updating the “role” column in source table.



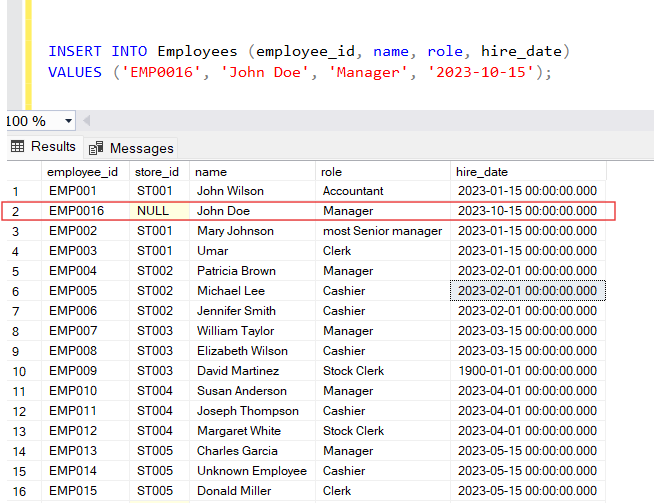
Source table is updated. Now will run SSIS package.



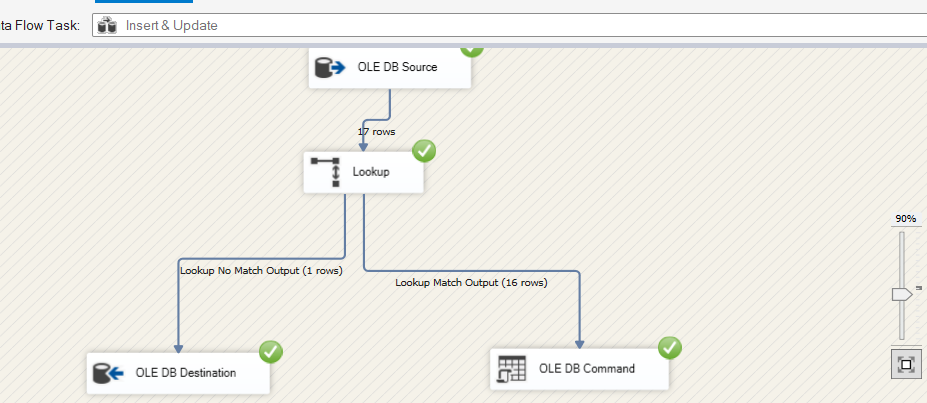
As we can see that we have made change only in 1 row but all the 16 rows are directed towards the destination.

There are two scenarios using “Lookup” tab. If a new row is inserted it will check in the “Lookup” tab that if its matching or not. If its matched it will go to “OLE DB Command” tab to update the record based on the “employee\_id”. If its new record it will not match in the existing record so it will go directly towards destination.

**Inserting new record:**

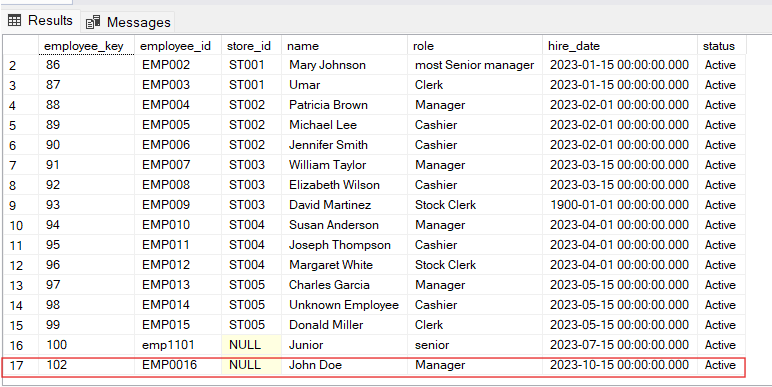


A new record is inserted into the source table. Now let’s run the SSIS package.



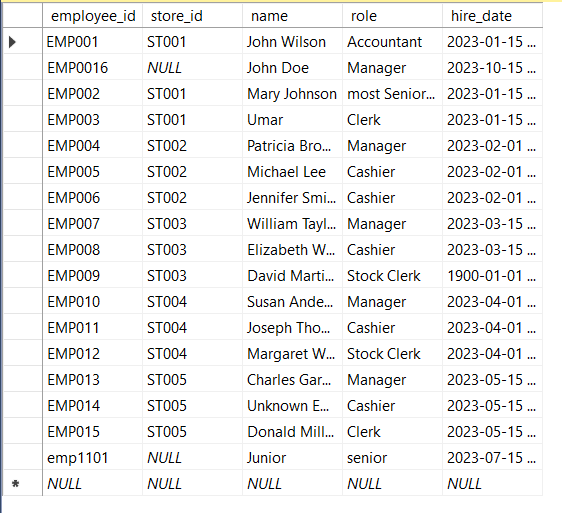
As we can see that 17 rows are coming from source table. 1 row is redirected towards insertion because the unique ID was not matching and rest 16 are going towards update tab.

Let’s check the destination table.



The new record is inserted successfully.

**Deleting:**

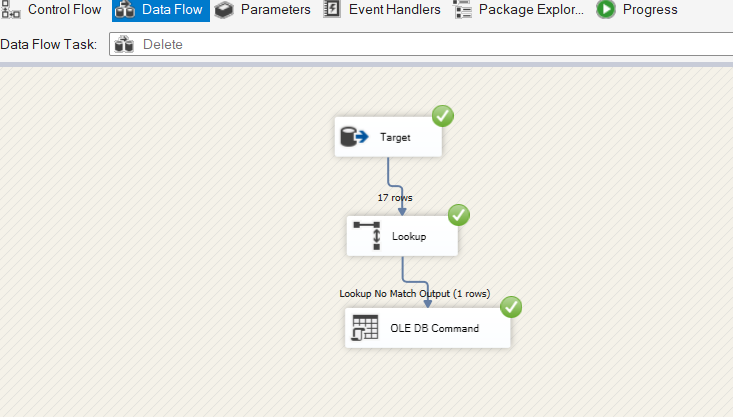


We can delete manually from the interface because there was Foreign Key constraint error. Here we are going to delete the row where employee\_id is “EMP014”.

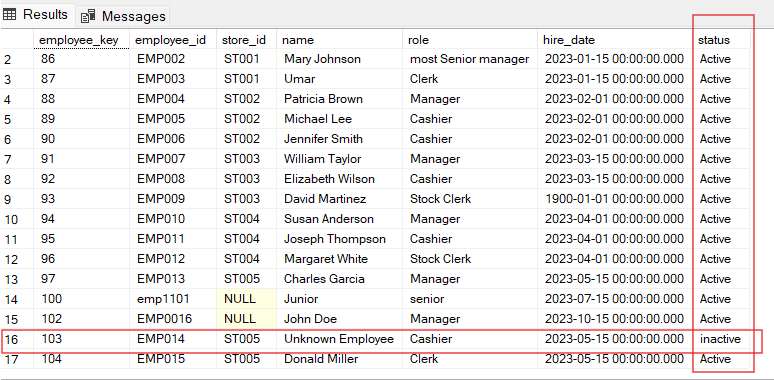
A screenshot of a computer

Description automatically generated

The deletion has been done. Let’s run and configure SSIS package.

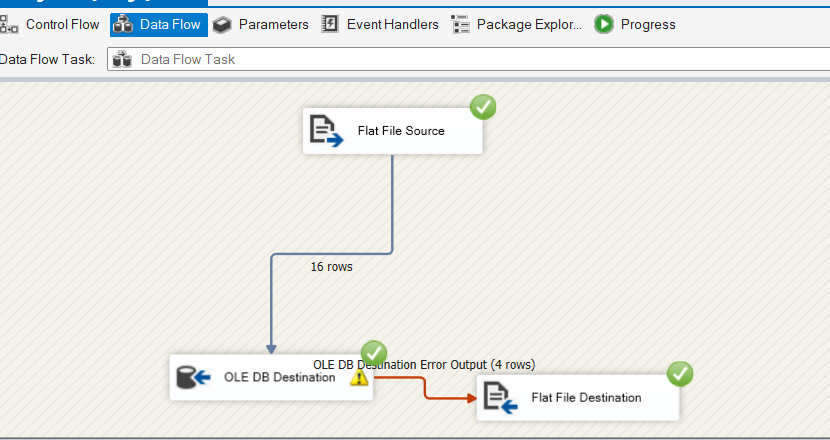


We need to add new “status” column in our destination table show to show the status i.e, “active, inactive”. Here in “OLE DB Source” we will take our destination table to compare it with our source table in “Lookup” tab, and in “OLE DB Command” we will use an update query that if there is a mismatch from destination to the source, update the “status” tab to “inactive”. Later on we can filter out the records based on this column in our visualization tools.



As we can see that Employee with employee\_id “EMP014” is not the part of our retail store.

**Error Handling in SSIS:**



Error Handling is very important aspect in SSIS. We must know how to handle and debug errors in order to smoothly run the package. If we face any error in loading data from flat files, we can add another destination which will take the problem causing rows. In this way we can understand that which rows need to be cleaned or transformed before loading into our destination.