Sri Lanka Institute of Information Technology

Data warehousing and Business Intelligence

Assignment 1

Student Registration No: IT20186142

Student Name: Wijesooriya H.M.A.H.

Step 1: Data Set Selection

This data set contains shopping online analytics of a famous super store which centers are distributed globally. They reach out to customers who do orders online. Each online order has a relevant customer, a product, a market and a shipment mode. Through this data set, it is able to profile the customers based on their of purchases and to profile countries based on the sales.

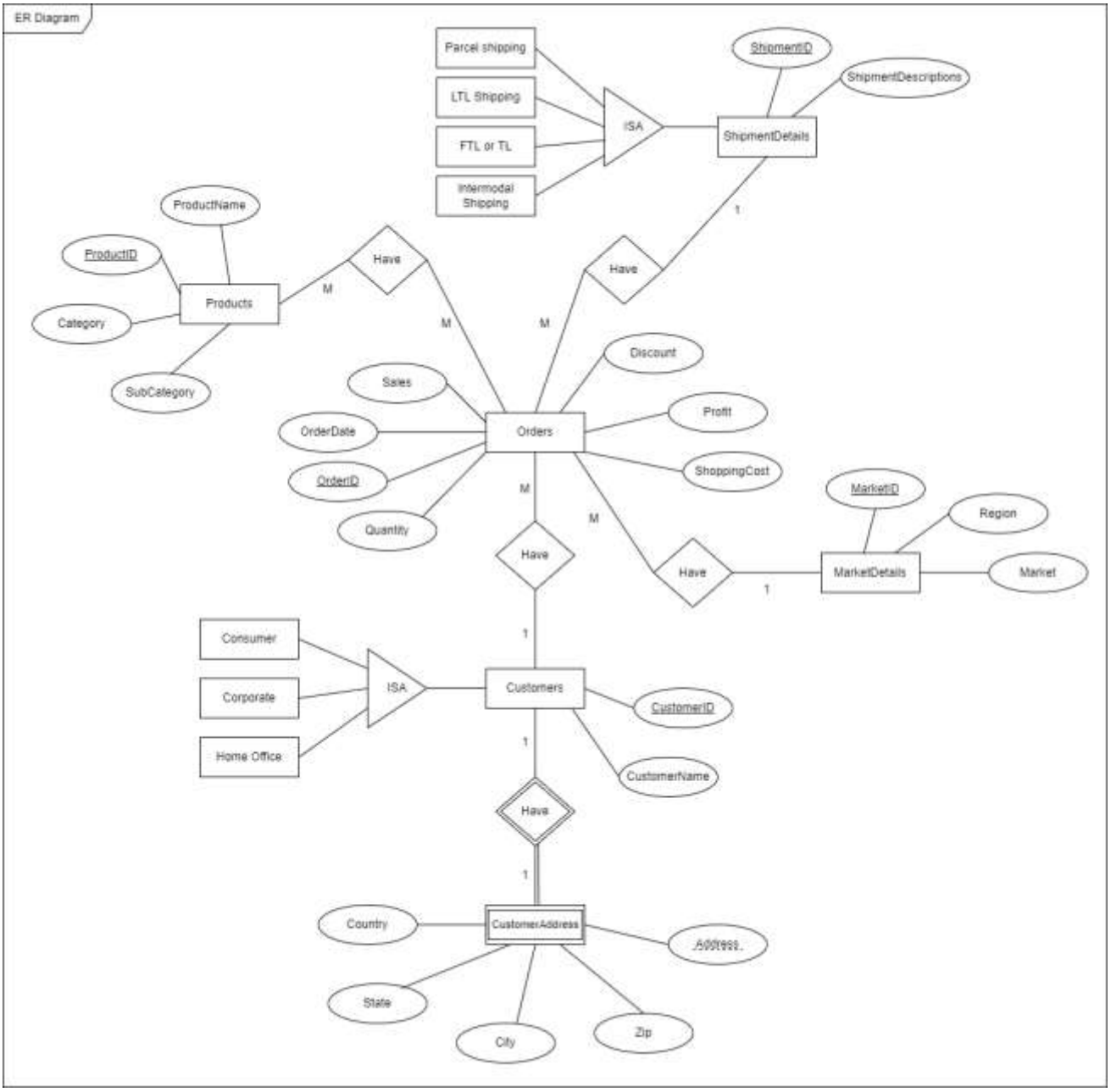
This data set includes details about 20,000 orders which were happened throughout 4 years and the number of customers involved in these orders are over 1500. In those orders there are about 2600 significant products which varies with category and subcategory.

This dataset contains Super store details,

- Customer details
- Customers segment details
- Customer addresses
- Order details
- Market details
- Product details
- Shipment details
- Product category details
- Product subcategory details

Also, there are some added details to this database.

Following ER- diagram will describe the scenario of the selected dataset



Step 2: Preparation of Data Sources

The whole of data was in ‘csv’ file type and they were separated into the following data sources, Database, Text, Excel and csv. And they were used to create the following,

1.Database(.bak)

Category.xls, MarketDetails.xls, OrderPriorityDetails.xls, Products.xls, SubCategory.xls and ShipmentDetails.xls files were imported to the Globa_Super_Store Database.

2.Text(.txt)

CustomerAddress.txt was used directly.

3.Excel(.xls)

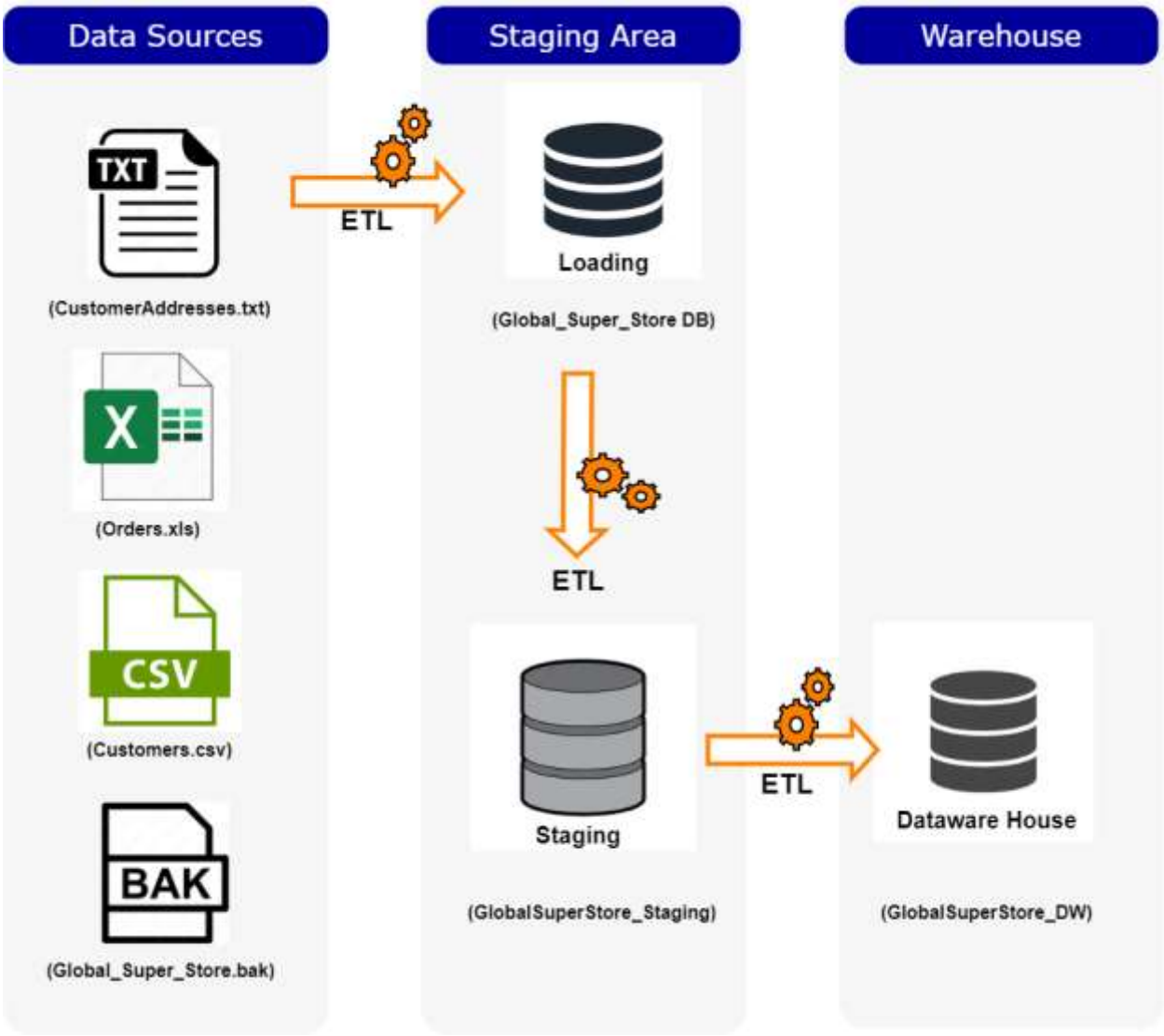
Orders.xls was used.

4.Comma Separated Values (.csv)

Customers.csv was used.

Data Source Type	Source Name	Column Name	Data Type	Description
Database File (.bak)	dbo.Products	Product_ID	nvarchar(50)	Unique ID
		Product_Name	nvarchar(150)	Name of Product
		CategoryID	int	Product Category ID
		Sub_CategoryID	int	Product Subcategory ID
	dbo.Category	CategoryID	int	Unique ID
		Category	nvarchar(50)	Product Category name
	Dbo.SubCategory	Sub_CategoryID	int	Unique ID
		Sub_Category	nvarchar(50)	Product SubCategory name
	dbo.MarketDetails	MarketID	int	Unique ID
		Market	nvarchar(50)	Market Name
		Region	nvarchar(50)	Region of market
	dbo.OrderPriorityDetails	OrderPriorityID	int	Unique ID
		OrderPriority	nvarchar(50)	Order Priority types
	Dbo.ShipmentDetails	ShipmentID	int	Unique ID
		Ship_Mode	nvarchar(50)	Shipment types
		ShipmentDescriptions	nvarchar(100)	Shipment type descriptions
Excel File	Orders.xls	Row ID	nvarchar(50)	Unique ID
		OrderID	nvarchar(50)	ID of the order
		OrderDate	date	Order placed date
		OrderPriority ID	int	ID of Order Priority
		Customer ID	nvarchar(50)	ID of Customer
		Sales	money	Sales
		Quantity	int	Quantity
		Discount	decimal(4, 3)	Discount of Product
		Profit	money	Profit
		Shipping Cost	money	Shipping Cost
		MarketID	int	ID of Market
		Product ID	nvarchar(50)	ID of Product
		ShipementID	int	ID of Shipment mode
		ShipmentDate	date	Shipment placed date
CSV File	Customers.csv	Customer ID	varchar(50)	Unique ID
		Customer Name	varchar(50)	
		Segment	varchar(50)	
Text File	CustomerAddresses.txt	Customer ID	varchar(50)	Unique ID
		Country	varchar(50)	Customer’s Country
		State	varchar(50)	Customer’s State
		City	varchar(50)	Customer’s City
		ZIP	varchar(50)	ZIP code of the Customer
		Address	varchar(60)	Customer’s Address

Step 3: Solution Architecture



Above architecture shows the high-level BI solution to the warehouse design.

Data Sources

'.txt' component represents Text files, '.xls' component is used to represent Excel files, '.csv' component is used to display Comma Separated files and '.bak' component represents database files.

Staging Area

Loading DB component represents the process of the creating database tables. Test, Patient, TestPrices, AddmissionFees and Attendance text files was imported to the database and was used to create the tables. And these tables were used as the DB source data.

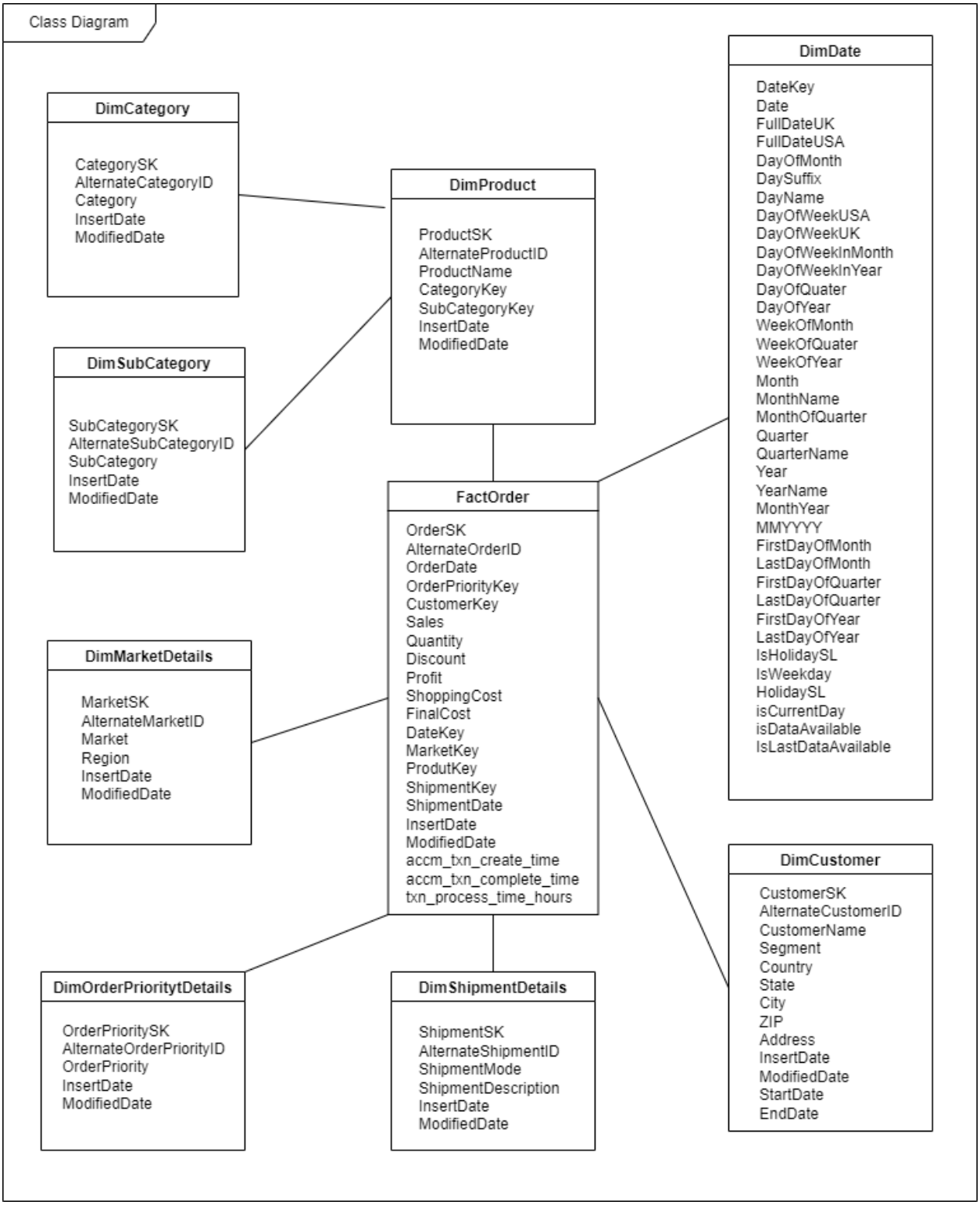
Staging DB component represents creating staging level tables through the ‘Extract’.

Data Warehouse

Data warehouse DB component is used display the cratering dimension tables in the warehouse using ‘Transform’ and ‘Load.’

Step 4: Data Warehouse Design & Development

Following figure will show how the fact table and dimension tables was combined in a rational manner.



Schema Type

For this scenario, snowflake schema type was used.

Dimension Types

- Hierarchical Dimension
    - Date – all the hierarchies in date
    - Product – product → category → subcategory
    - CustomerAddress – country → state → city → ZIP → address
- Slowly Changing Dimension
    - Customer – used type 2.
    - Following columns were set as changing attributes.
        - Segment
        - Country
        - State
        - City
        - ZIP code
        - Address

- Fact Table
    - Numbers – Sales, Quantity, Discount, Profit, ShippingCost, FinalCost
    - FK - CustomerKey, OrderDateKey, OrderPriorityKey, MarketKey, ProductKey, ShipmentKey

Assumptions

- Customer dimension was considered as a slowly changing dimension.

Step 5: ETL development

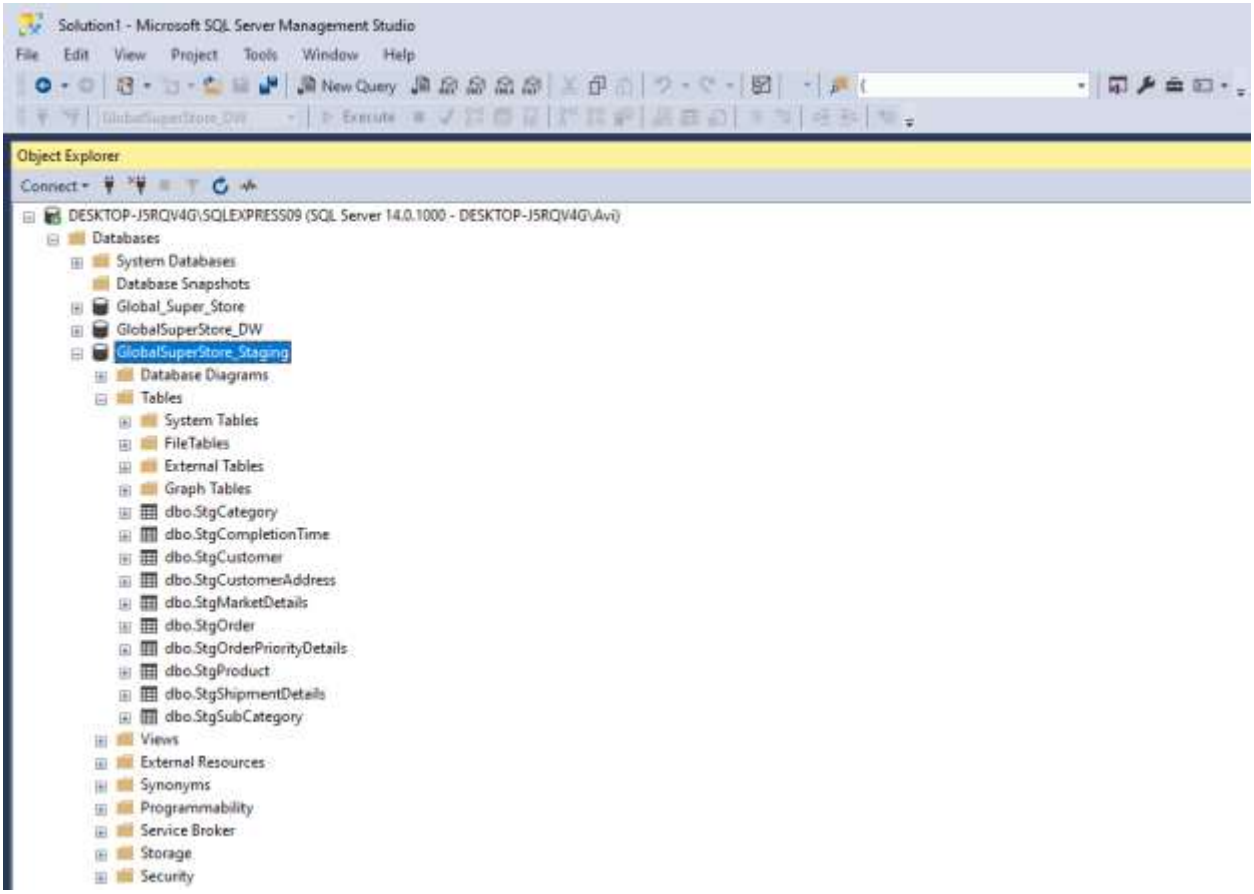
1.Extract

In this step, All the data sources were imported to the staging tables by using the relevant Data connection.

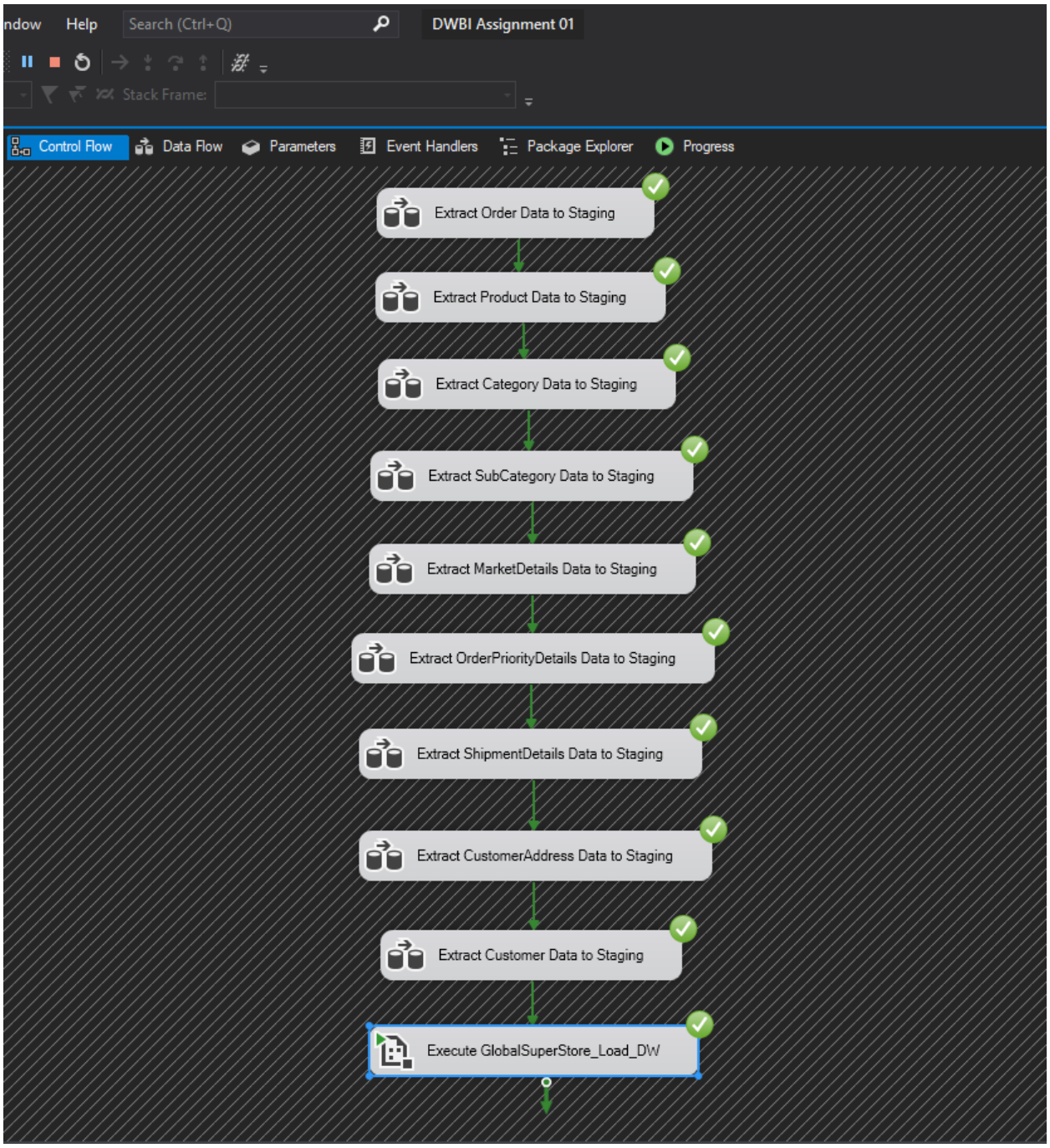
Flat file connection was used for text files and csv files, Excel file connections for excel file, DB source connection for DB file. All those tables were imported to the GlobalSuperStore_Staging DB, which contains the below tables,

1. StgCategory
2. StgCustomer
3. StgCustomerAddress
4. StgMarketDetails
5. StgOrder
6. StgOrderPriorityDetails
7. StgProduct
8. StgShipmentDetails
9. SgtSubCategory

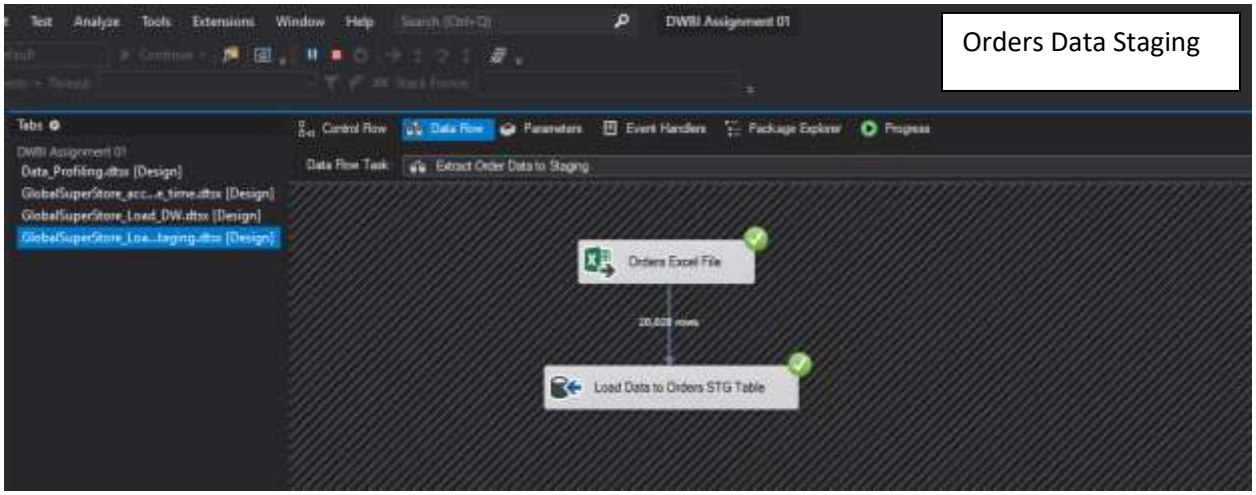
- **Snapshot of SSMS Staging Database**

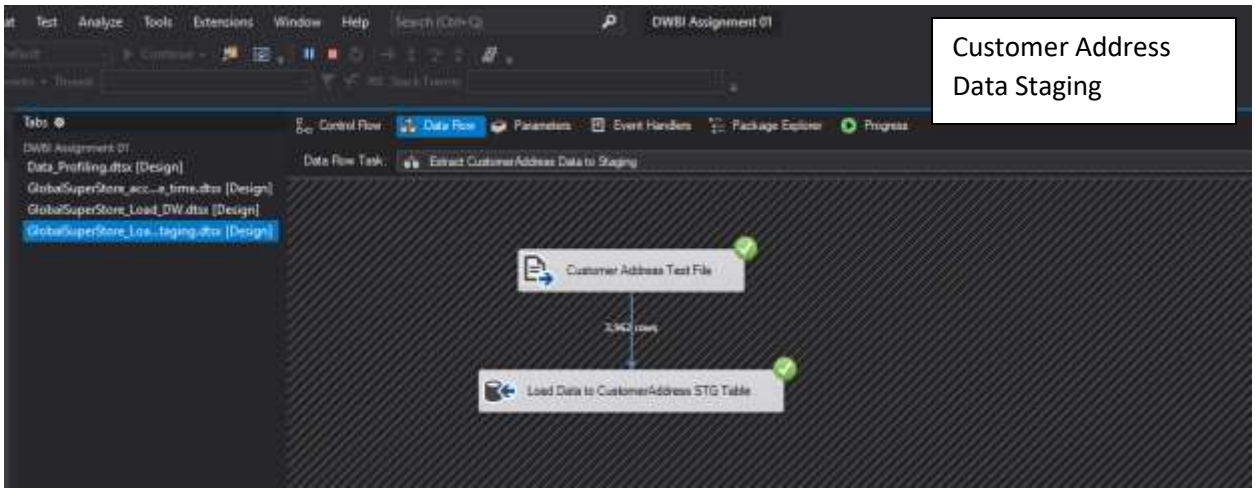


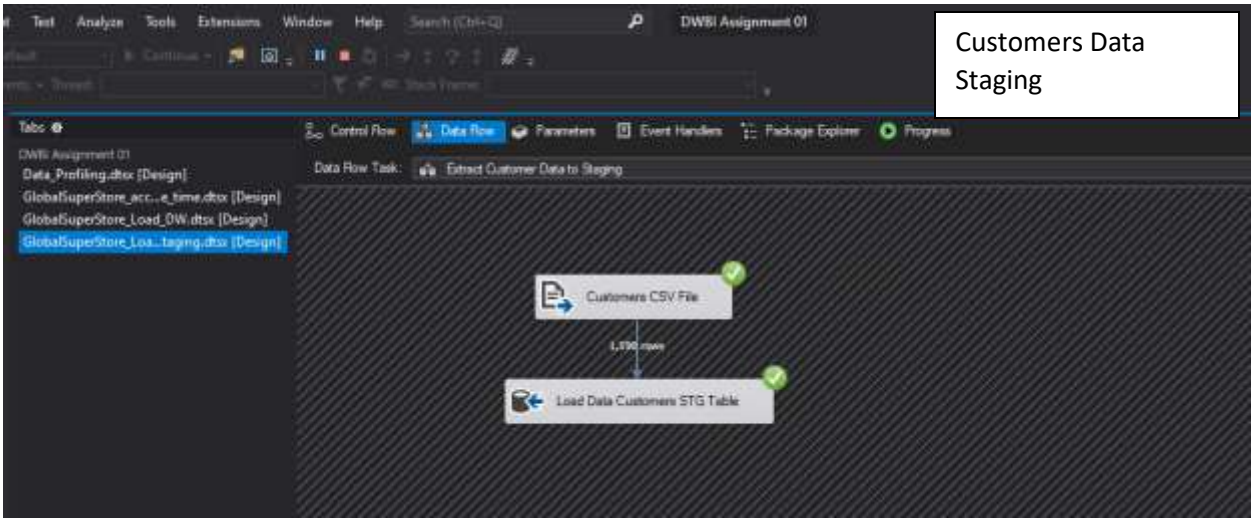
- **Snapshot of Visual Studio Control Flow of Extract**



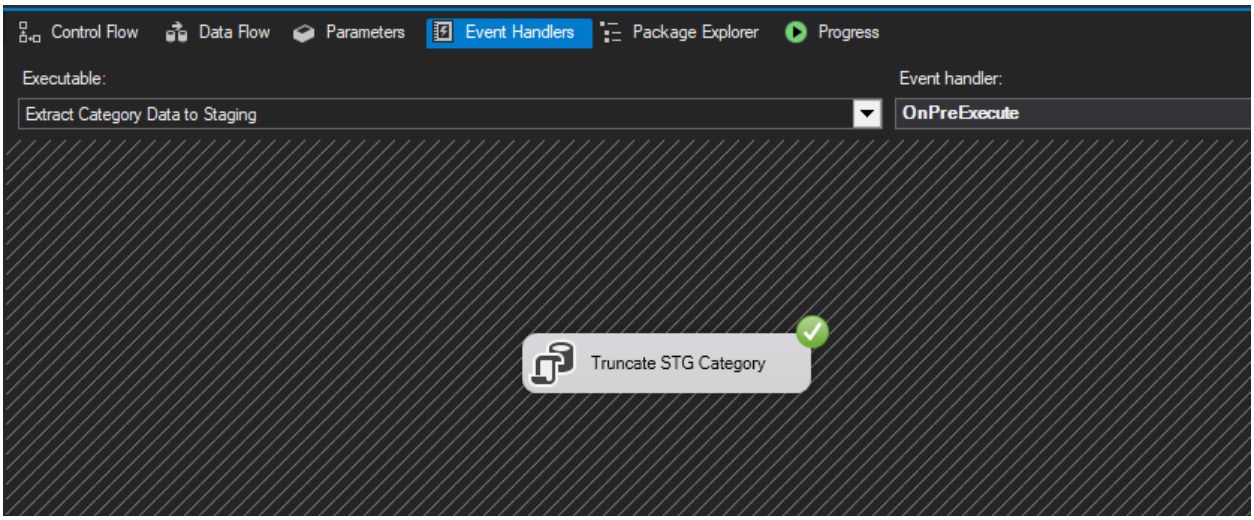
- **Snapshots of several data types of Data Flows**

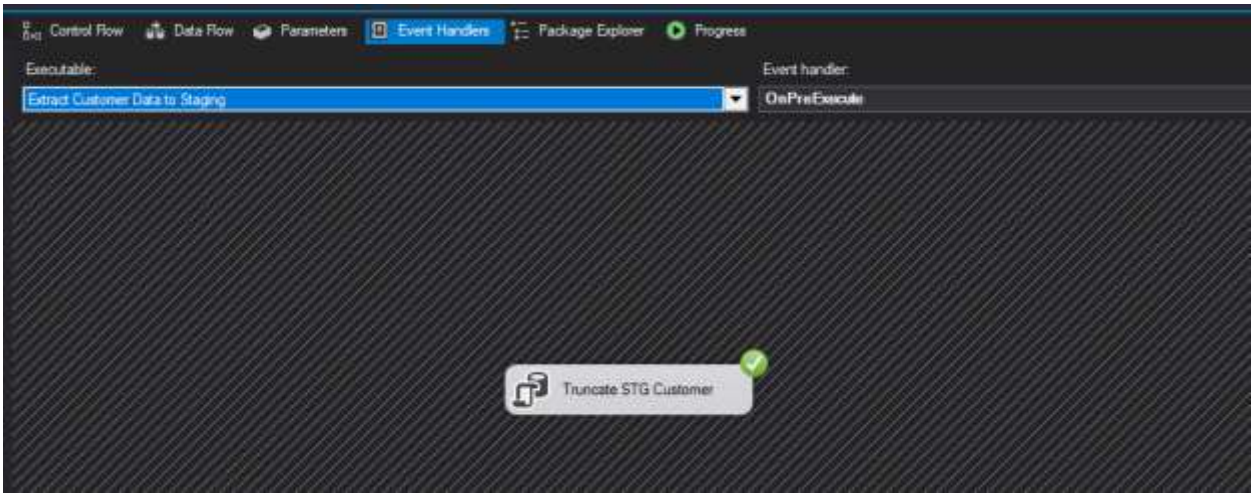


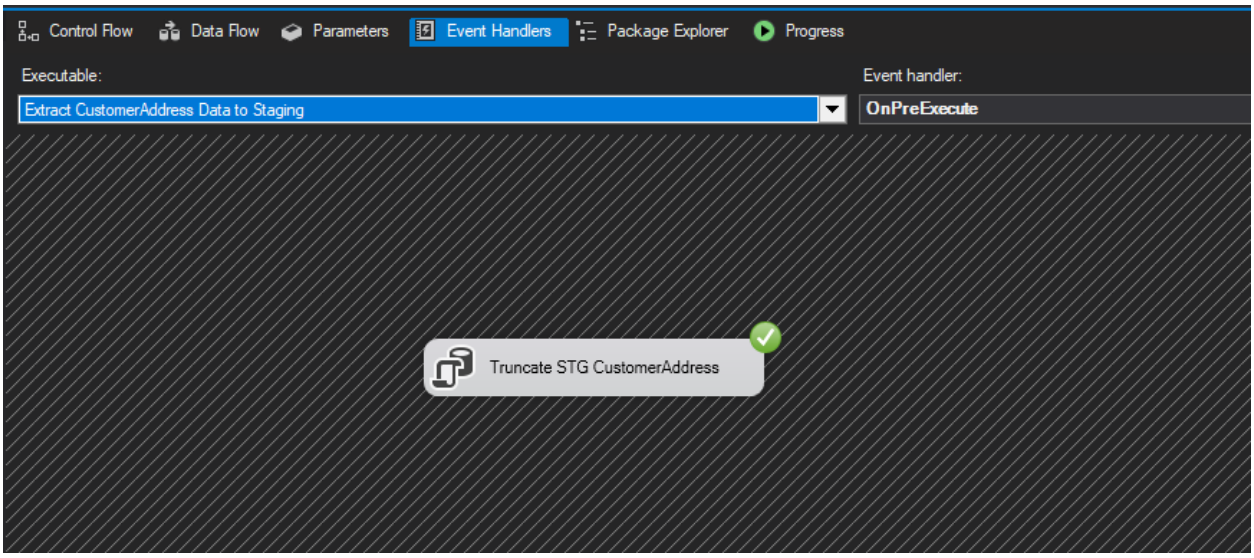


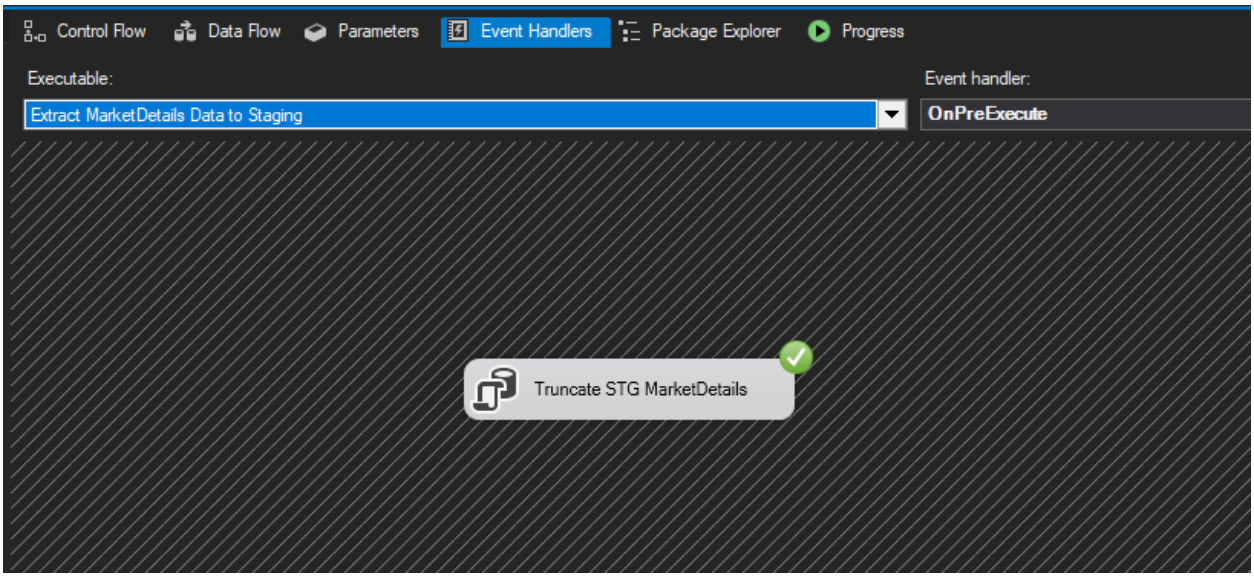


- **Event Handling (Truncate Staging Data)**

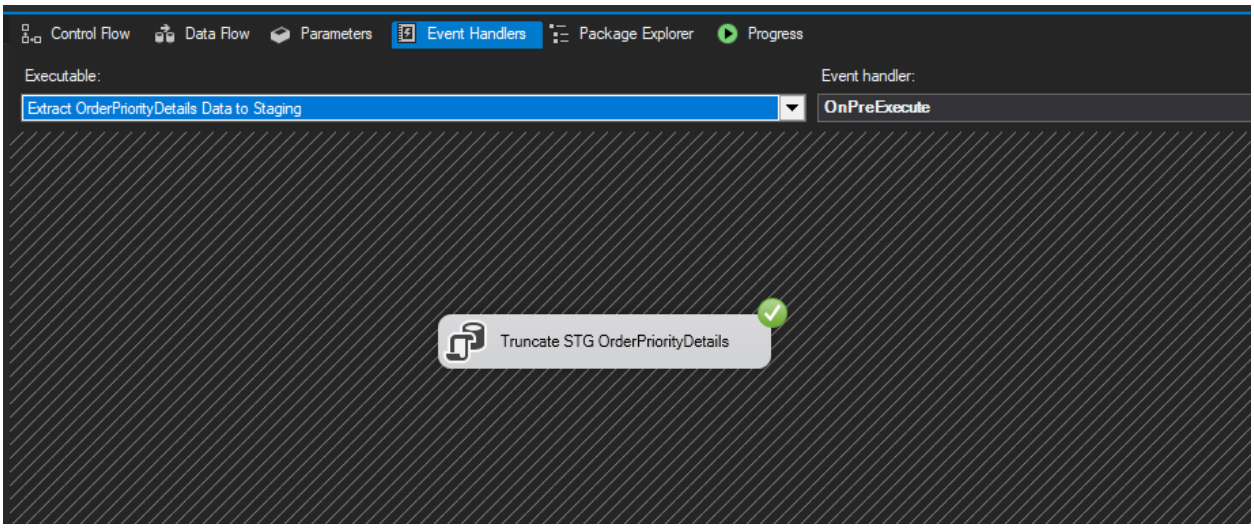


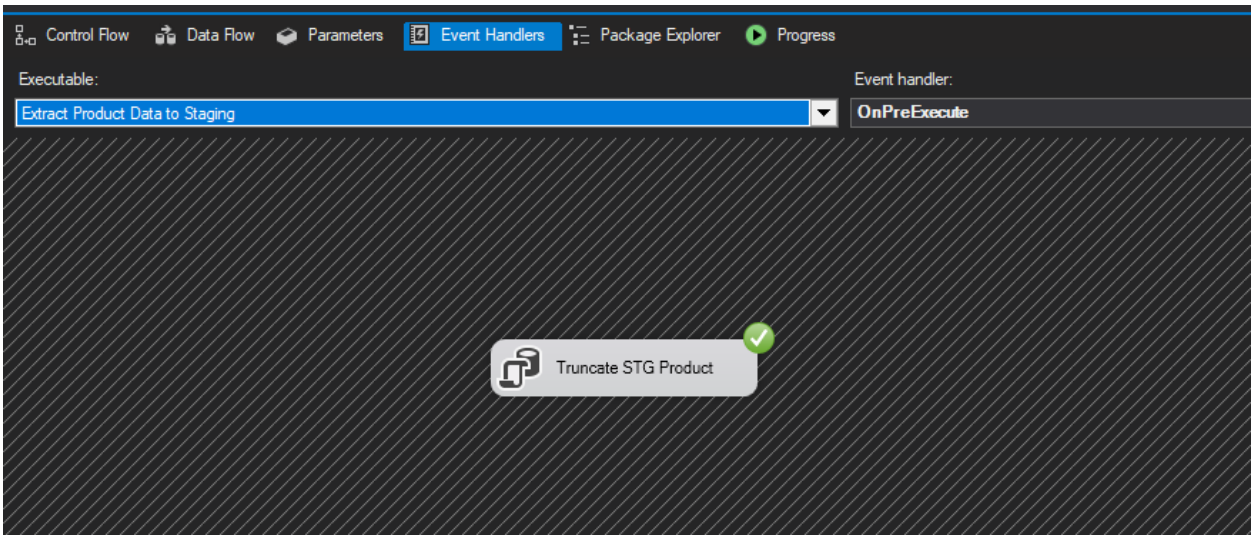


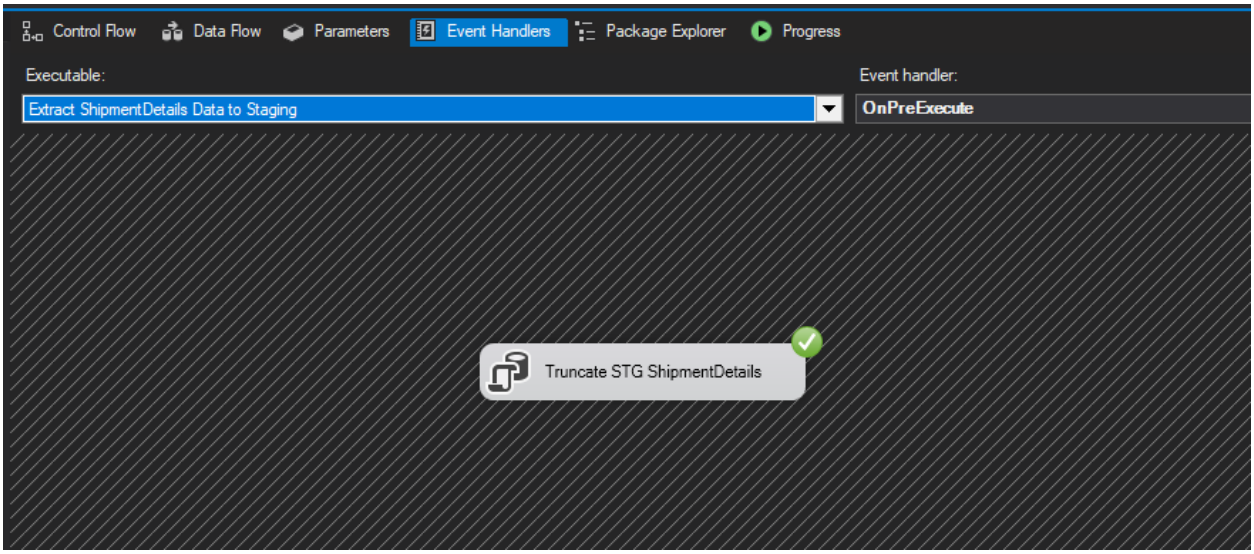


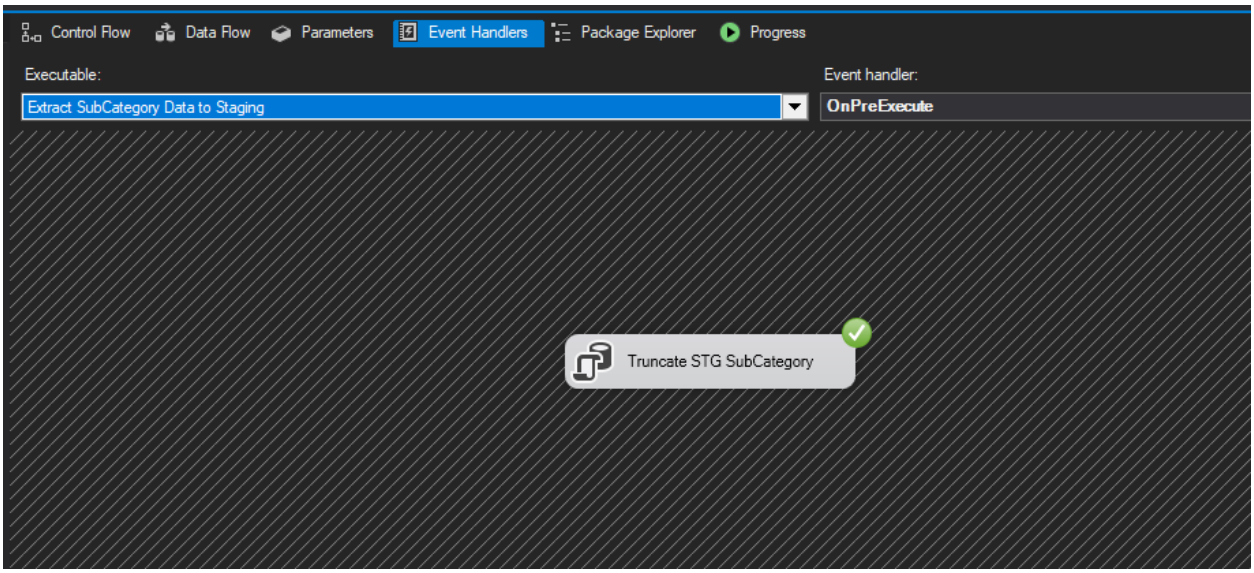

Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Executable:
Extract MarketDetails Data to Staging
Event handler:
OnPreExecute
Truncate STG MarketDetails


Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Executable:
Extract Order Data to Staging
Event handler:
OnPreExecute
Truncate STG Order


Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Executable:
Extract OrderPriorityDetails Data to Staging
Event handler:
OnPreExecute
Truncate STG OrderPriorityDetails

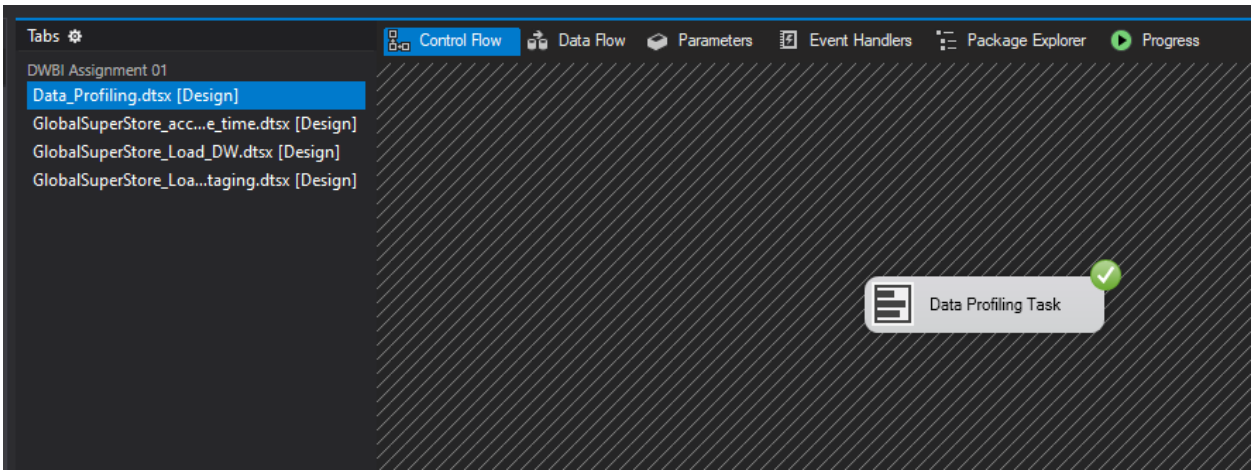

Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Executable:
Extract Product Data to Staging
Event handler:
OnPreExecute
Truncate STG Product

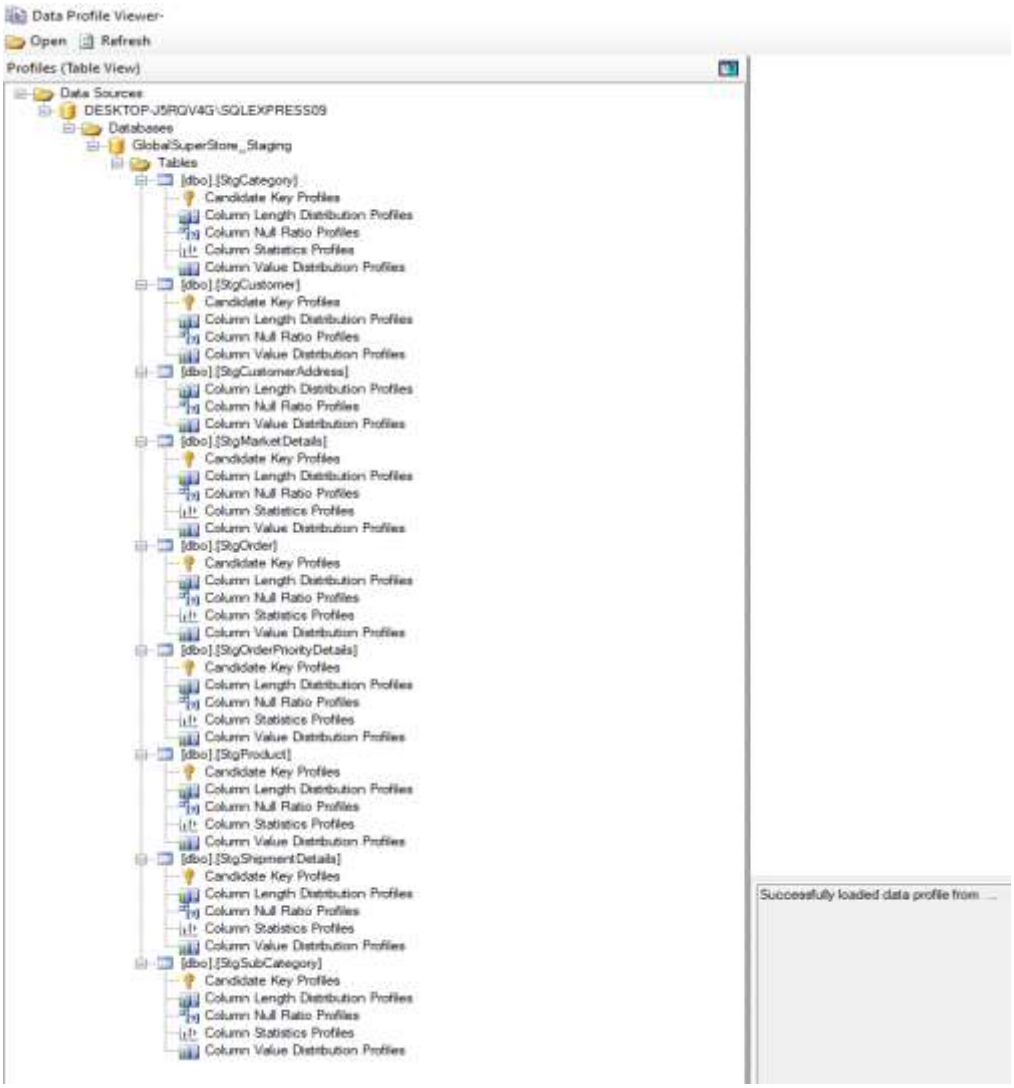

Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Executable:
Extract ShipmentDetails Data to Staging
Event handler:
OnPreExecute
Truncate STG ShipmentDetails


Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Executable:
Extract SubCategory Data to Staging
Event handler:
OnPreExecute
Truncate STG SubCategory

- **Data profiling**

  Used Data_Profiling package to profiling the staging tables





3.Transform & Load

In this step, both the ‘Transform’ and ‘Load’ are done. Firstly, The Dimension tables in the Datawarehouse DB data were created. Then, using the relevant components, data from the staging tables was loaded into the warehouse tables, GlobalSuperStore_DW, which contains the below tables.

1. DimCategory
2. DimCustomer
3. DimDate
4. DimMarketDetails
5. DimOrderPriorityDetails
6. DimProduct
7. DimShipmentDetails
8. DimSubCategory
9. FactOrders

Used Transformation Tasks

1. Lookups

   DimCustomer’s CustomerID is looked when loading using DimOrder
   DimProduct’s ProductSK is looked when loading using DimMarket

2. Derived Columns

   Derived column is used in FactOrders to derive both StartDate and EndDate by using GETDATE() expression and to derive the FinalCost too.

3. Union

   Union is used in the Extract step to combine and get all the data from data files.

4. Sort and Merge

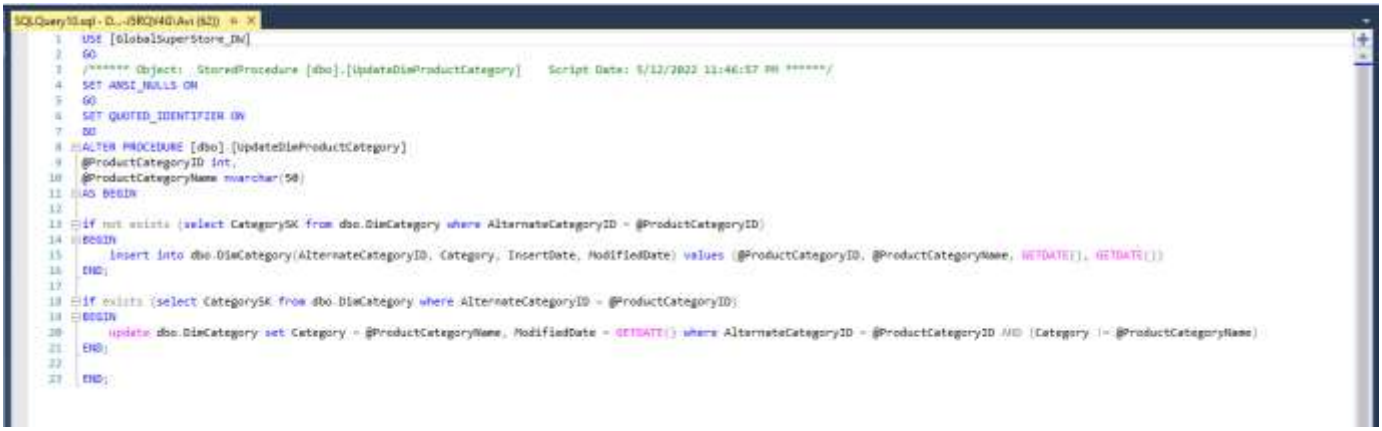
   ‘Sort’ is used in DimProduct to sort out the Product and Category data and they are merged using CategoryID.
   ‘Sort’ is used sort out the Product and Sub-Category data and they are merged using SubCategoryID.

Update Functions

- DimMarketDetails

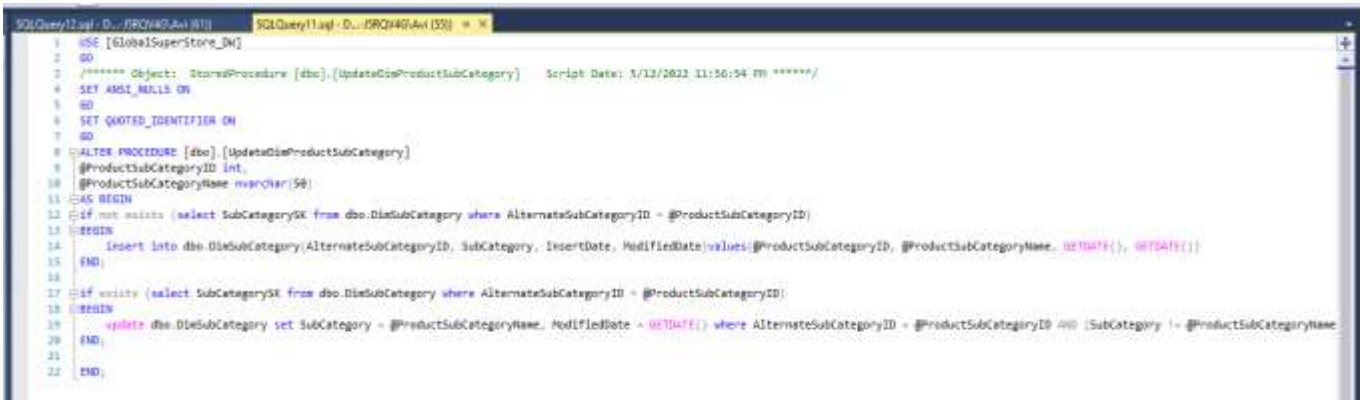
```
USE [GlobalSuperStore_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimMarketDetails]    Script Date: 5/12/2022 11:39:00 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimMarketDetails]
@MarketID int,
@Market nvarchar(50),
@Region nvarchar(50)

AS BEGIN

if not exists (select MarketSK from dbo.DimMarketDetails where AlternateMarketID = @MarketID)
BEGIN
    insert into dbo.DimMarketDetails(AlternateMarketID, Market, Region, InsertDate, ModifiedDate) values (@MarketID, @Market, @Region, GETDATE(), GETDATE())
END;

if exists (select MarketSK from dbo.DimMarketDetails where AlternateMarketID = @MarketID)
BEGIN
    update dbo.DimMarketDetails set Market = @Market, Region = @Region where AlternateMarketID = @MarketID AND (Market != @Market OR Region != @Region)
END;

END;
```

- DimProductCategory



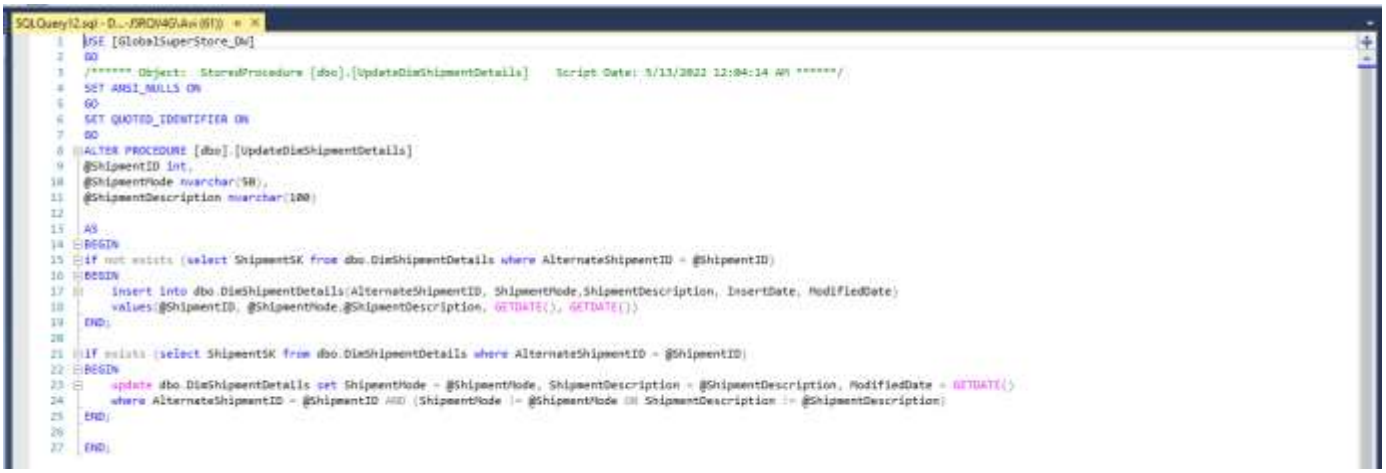
```
USE [GlobalSuperStore_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimProductCategory]    Script Date: 5/12/2022 11:46:57 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimProductCategory]
@ProductCategoryID int,
@ProductCategoryName nvarchar(50)
AS BEGIN

if not exists (select CategorySK from dbo.DimCategory where AlternateCategoryID = @ProductCategoryID)
BEGIN
    insert into dbo.DimCategory(AlternateCategoryID, Category, InsertDate, ModifiedDate) values (@ProductCategoryID, @ProductCategoryName, GETDATE(), GETDATE())
END;

if exists (select CategorySK from dbo.DimCategory where AlternateCategoryID = @ProductCategoryID)
BEGIN
    update dbo.DimCategory set Category = @ProductCategoryName, ModifiedDate = GETDATE() where AlternateCategoryID = @ProductCategoryID AND (Category != @ProductCategoryName)
END;

END;
```

- DimProductSubCategory



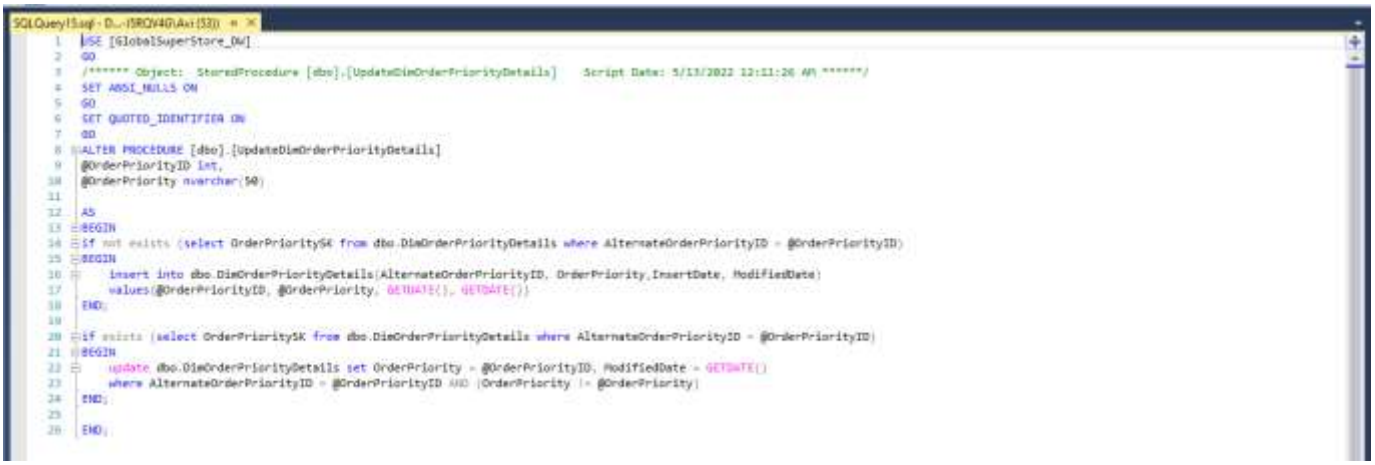
```
USE [GlobalSuperStore_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimProductSubCategory]    Script Date: 5/12/2022 11:56:54 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimProductSubCategory]
@ProductSubCategoryID int,
@ProductSubCategoryName nvarchar(50)
AS BEGIN
if not exists (select SubCategorySK from dbo.DimSubCategory where AlternateSubCategoryID = @ProductSubCategoryID)
BEGIN
    insert into dbo.DimSubCategory(AlternateSubCategoryID, SubCategory, InsertDate, ModifiedDate)values(@ProductSubCategoryID, @ProductSubCategoryName, GETDATE(), GETDATE())
END;

if exists (select SubCategorySK from dbo.DimSubCategory where AlternateSubCategoryID = @ProductSubCategoryID)
BEGIN
    update dbo.DimSubCategory set SubCategory = @ProductSubCategoryName, ModifiedDate = GETDATE() where AlternateSubCategoryID = @ProductSubCategoryID AND (SubCategory != @ProductSubCategoryName
END;

END;
```

- DimShipmentDetails



```
USE [GlobalSuperStore_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimShipmentDetails]    Script Date: 5/13/2022 12:04:14 AM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimShipmentDetails]
@ShipmentID int,
@ShipmentMode nvarchar(50),
@ShipmentDescription nvarchar(100)

AS
BEGIN
if not exists (select ShipmentSK from dbo.DimShipmentDetails where AlternateShipmentID = @ShipmentID)
BEGIN
    insert into dbo.DimShipmentDetails(AlternateShipmentID, ShipmentMode,ShipmentDescription, InsertDate, ModifiedDate)
    values(@ShipmentID, @ShipmentMode,@ShipmentDescription, GETDATE(), GETDATE())
END;

if exists (select ShipmentSK from dbo.DimShipmentDetails where AlternateShipmentID = @ShipmentID)
BEGIN
    update dbo.DimShipmentDetails set ShipmentMode = @ShipmentMode, ShipmentDescription = @ShipmentDescription, ModifiedDate = GETDATE()
    where AlternateShipmentID = @ShipmentID AND (ShipmentMode != @ShipmentMode OR ShipmentDescription != @ShipmentDescription)
END;

END;
```

- DimOrderPriorityDetails

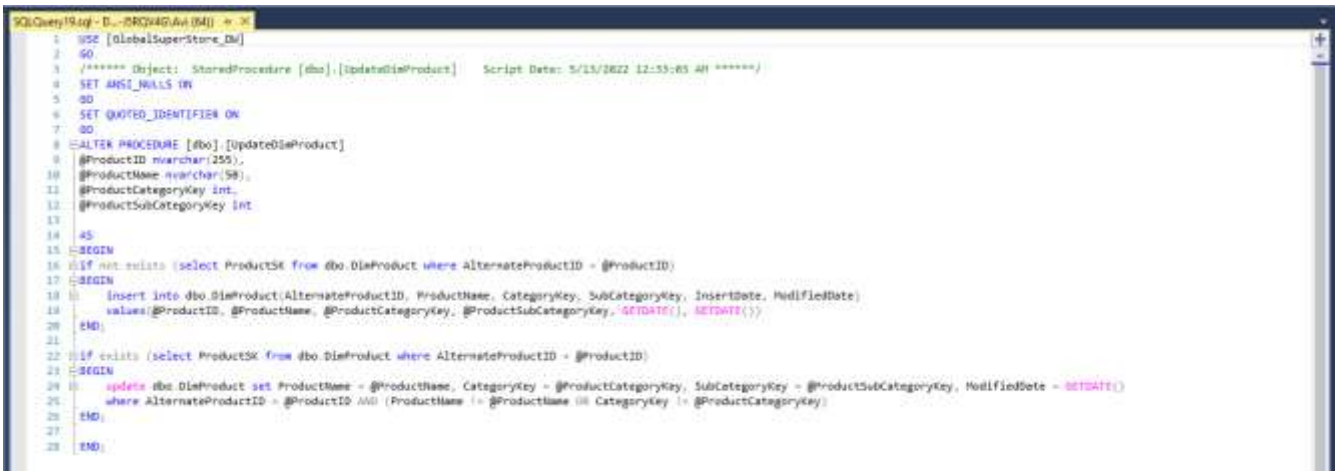


```
USE [GlobalSuperStore_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimOrderPriorityDetails]    Script Date: 5/13/2022 12:11:26 AM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimOrderPriorityDetails]
@OrderPriorityID int,
@OrderPriority nvarchar(50)

AS
BEGIN
if not exists (select OrderPrioritySK from dbo.DimOrderPriorityDetails where AlternateOrderPriorityID = @OrderPriorityID)
BEGIN
    insert into dbo.DimOrderPriorityDetails(AlternateOrderPriorityID, OrderPriority,InsertDate, ModifiedDate)
    values(@OrderPriorityID, @OrderPriority, GETDATE(), GETDATE())
END;

if exists (select OrderPrioritySK from dbo.DimOrderPriorityDetails where AlternateOrderPriorityID = @OrderPriorityID)
BEGIN
    update dbo.DimOrderPriorityDetails set OrderPriority = @OrderPriorityID, ModifiedDate = GETDATE()
    where AlternateOrderPriorityID = @OrderPriorityID AND (OrderPriority != @OrderPriority)
END;

END;
```

- FactOrders

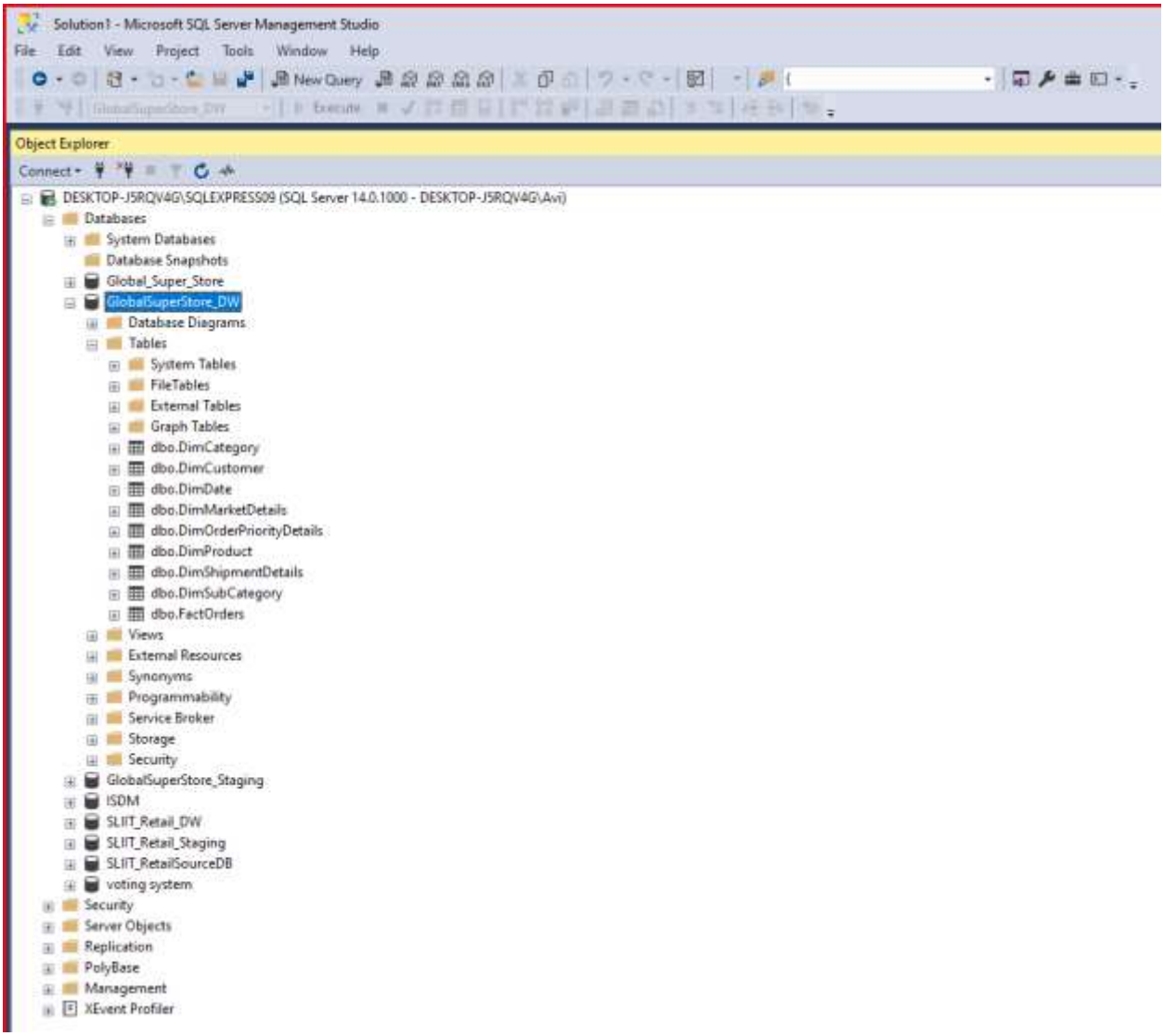
```
USE [GlobalSuperStore_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateFactOrders]    Script Date: 5/13/2022 12:10:42 AM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateFactOrders]
@RowID int,
@accm_txn_complete_time datetime,
@txn_process_time_hours int
AS
BEGIN
if exists (select RowID
from dbo.FactOrders
where RowID = @RowID)
BEGIN
update dbo.FactOrders
set
accm_txn_complete_time=@accm_txn_complete_time,
txn_process_time_hours=@txn_process_time_hours
where RowID = @RowID
END;
END;
```

- DimProduct

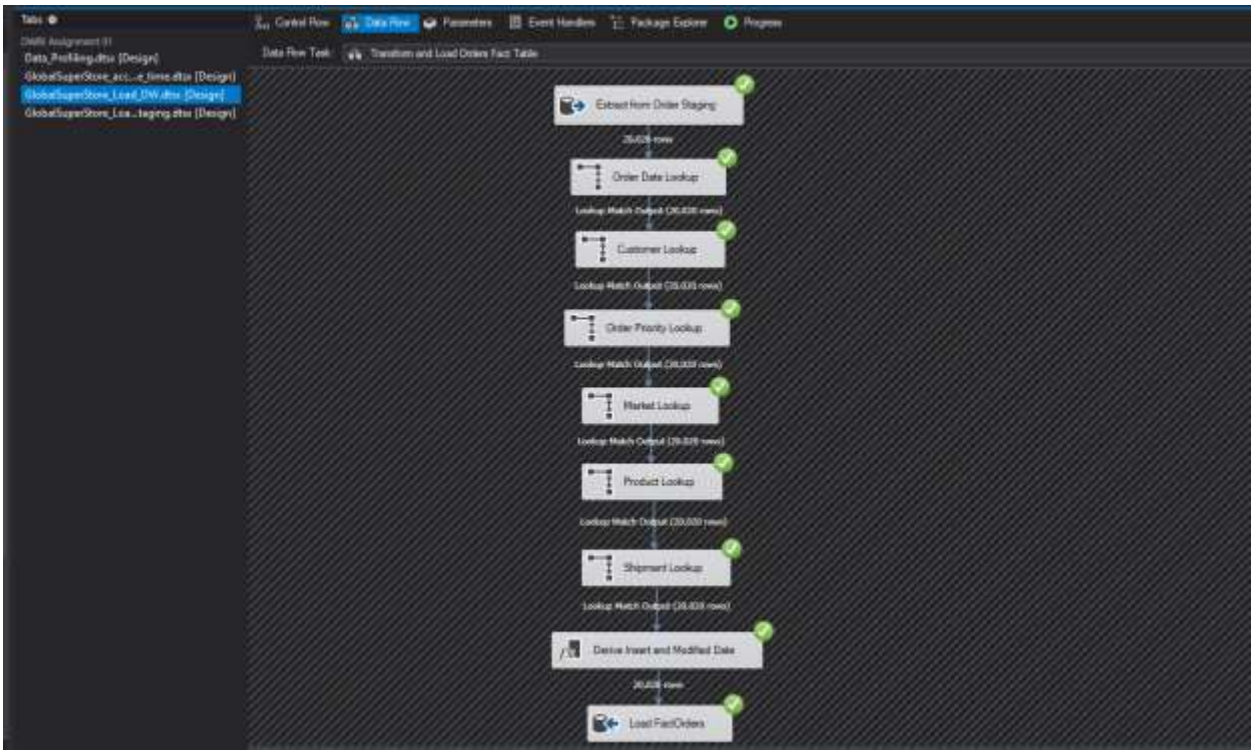


```
USE [GlobalSuperStore_DW]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimProduct]    Script Date: 5/15/2022 12:55:05 AM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimProduct]
@ProductID nvarchar(255),
@ProductName nvarchar(50),
@ProductCategoryKey int,
@ProductSubCategoryKey int

AS
BEGIN
if not exists (select ProductSK from dbo.DimProduct where AlternateProductID = @ProductID)
BEGIN
    insert into dbo.DimProduct(AlternateProductID, ProductName, CategoryKey, SubCategoryKey, InsertDate, ModifiedDate)
    values(@ProductID, @ProductName, @ProductCategoryKey, @ProductSubCategoryKey, GETDATE(), GETDATE())
END

if exists (select ProductSK from dbo.DimProduct where AlternateProductID = @ProductID)
BEGIN
    update dbo.DimProduct set ProductName = @ProductName, CategoryKey = @ProductCategoryKey, SubCategoryKey = @ProductSubCategoryKey, ModifiedDate = GETDATE()
    where AlternateProductID = @ProductID AND (ProductName != @ProductName OR CategoryKey != @ProductCategoryKey)
END

END
```

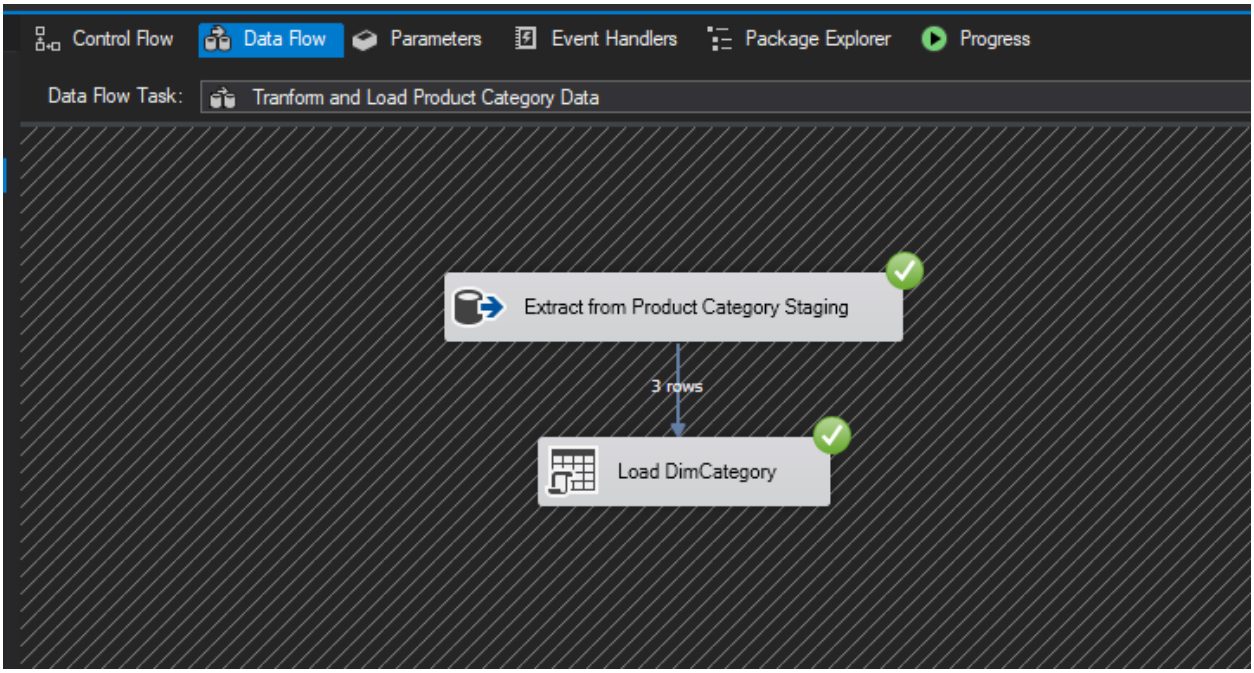
- **Snapshot of SQL server Data warehouse Database**



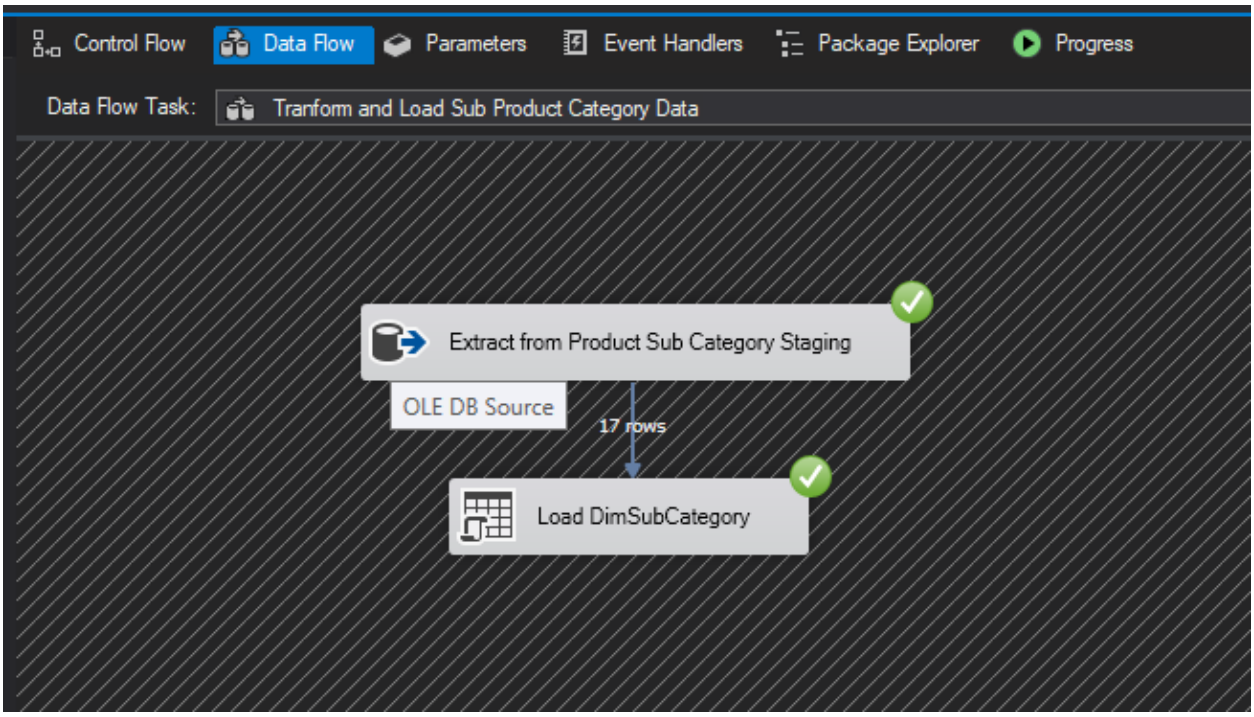
- **Snapshot of Visual Studio Control Flow of Extraction**



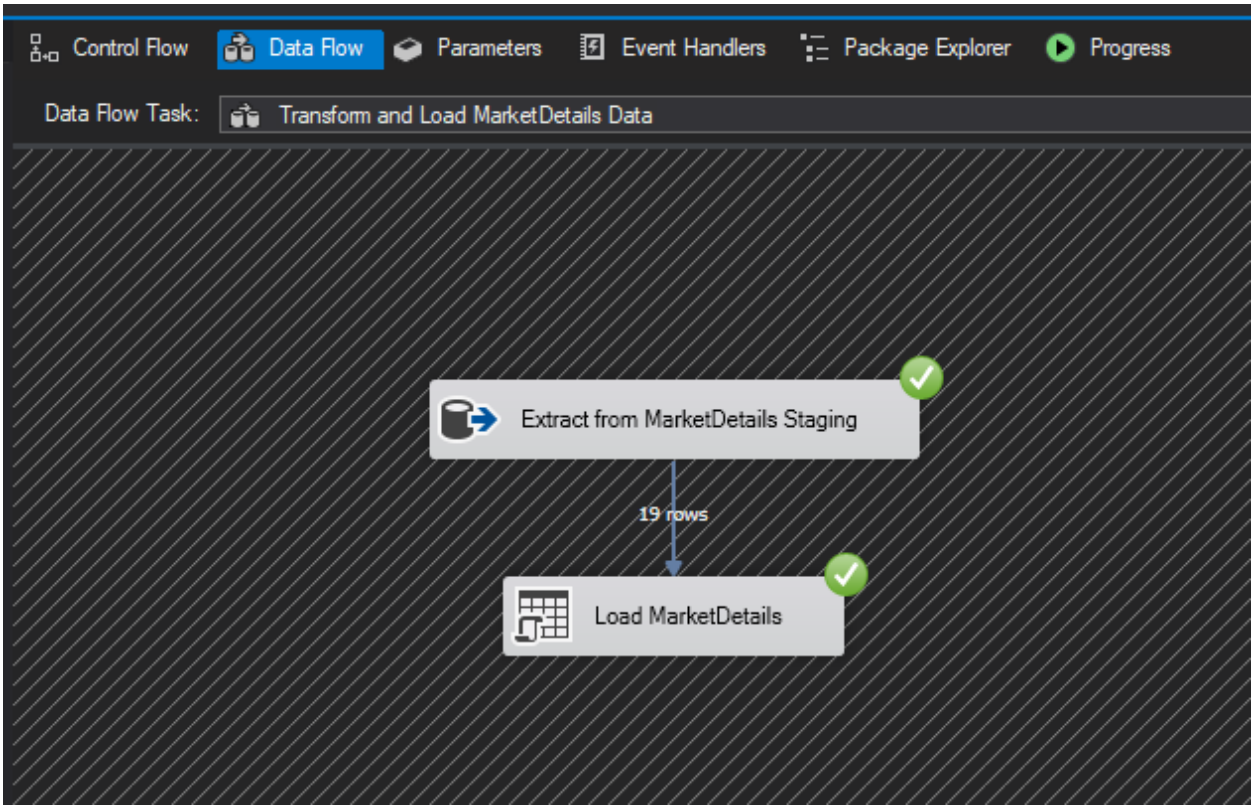
- **Product Category Data Transform and Load**



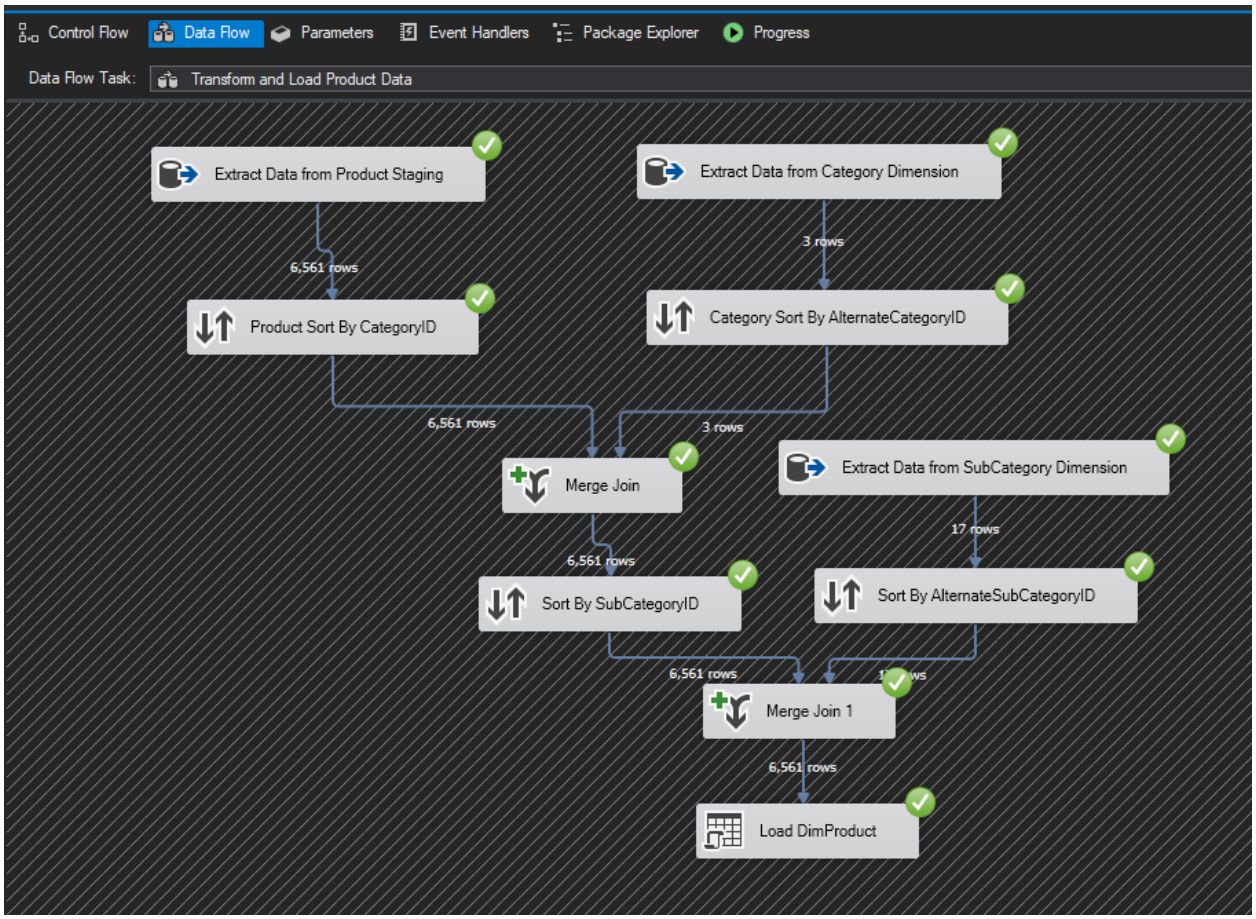
- **Product Subcategory Data Transform and Load**



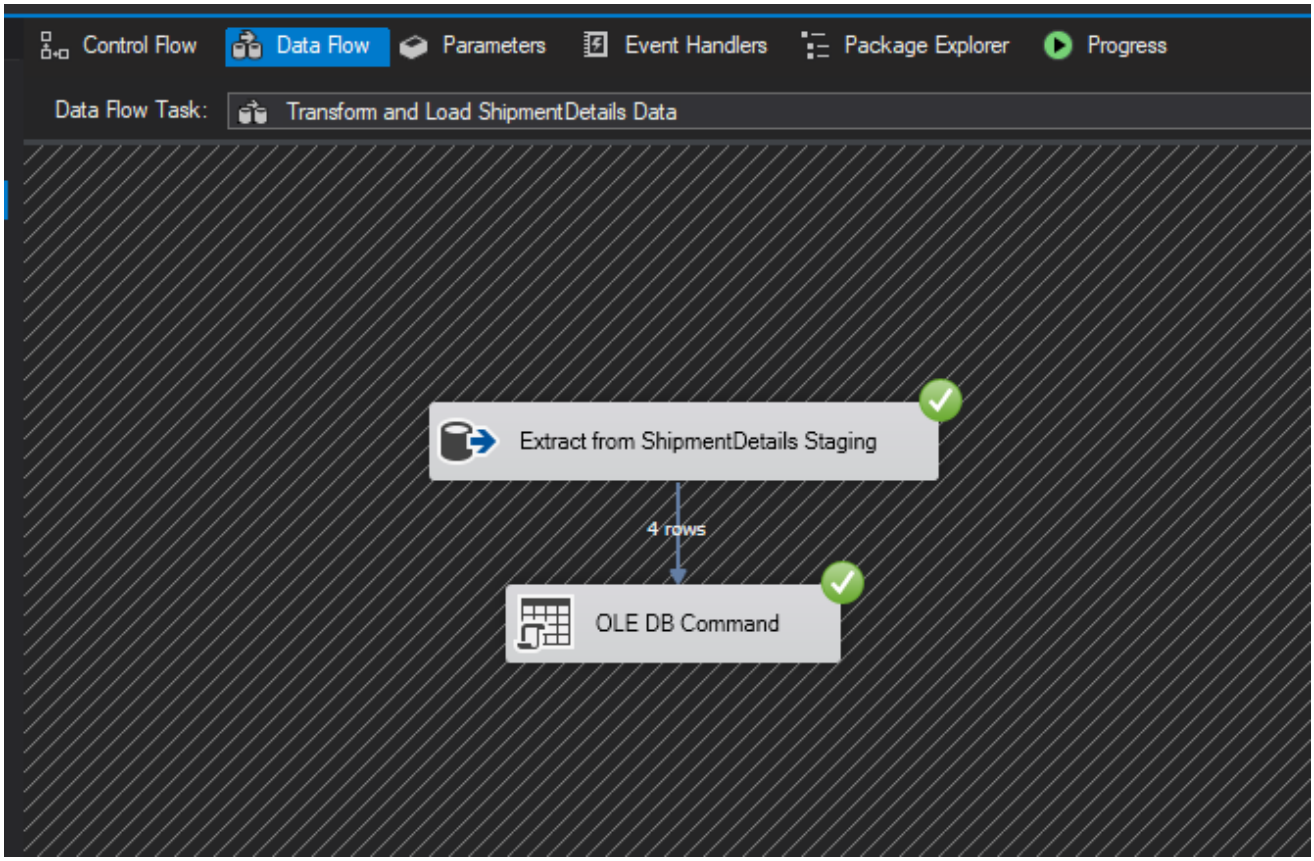
- **Market Details Data Transform and Load**



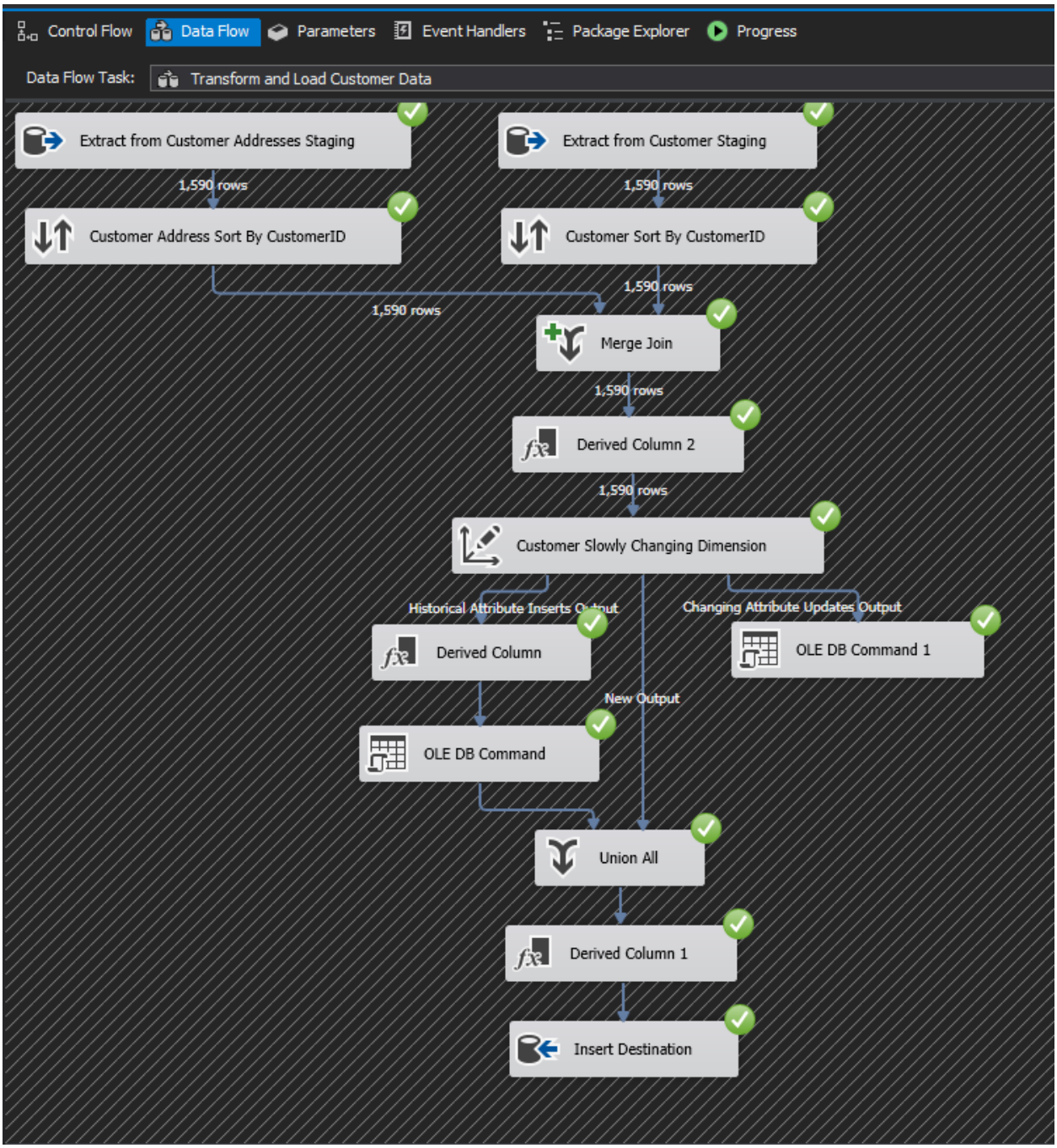
- **Product Data Transform and Load**



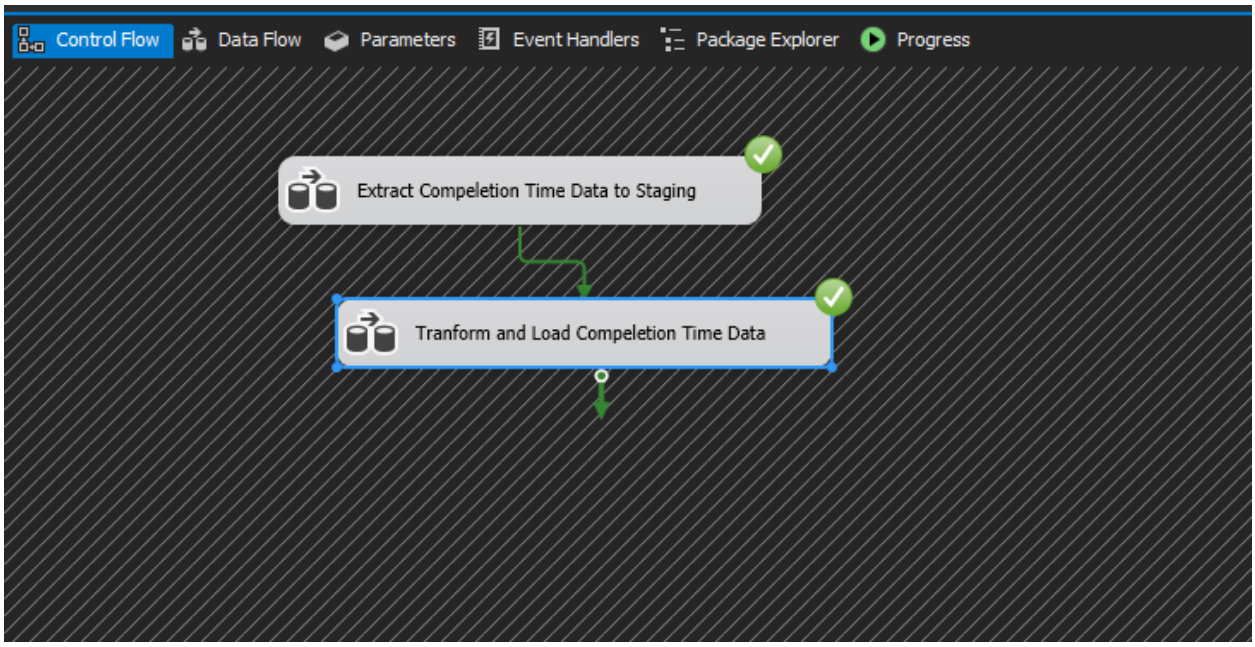
- **Shipment Details Data Transform and Load**

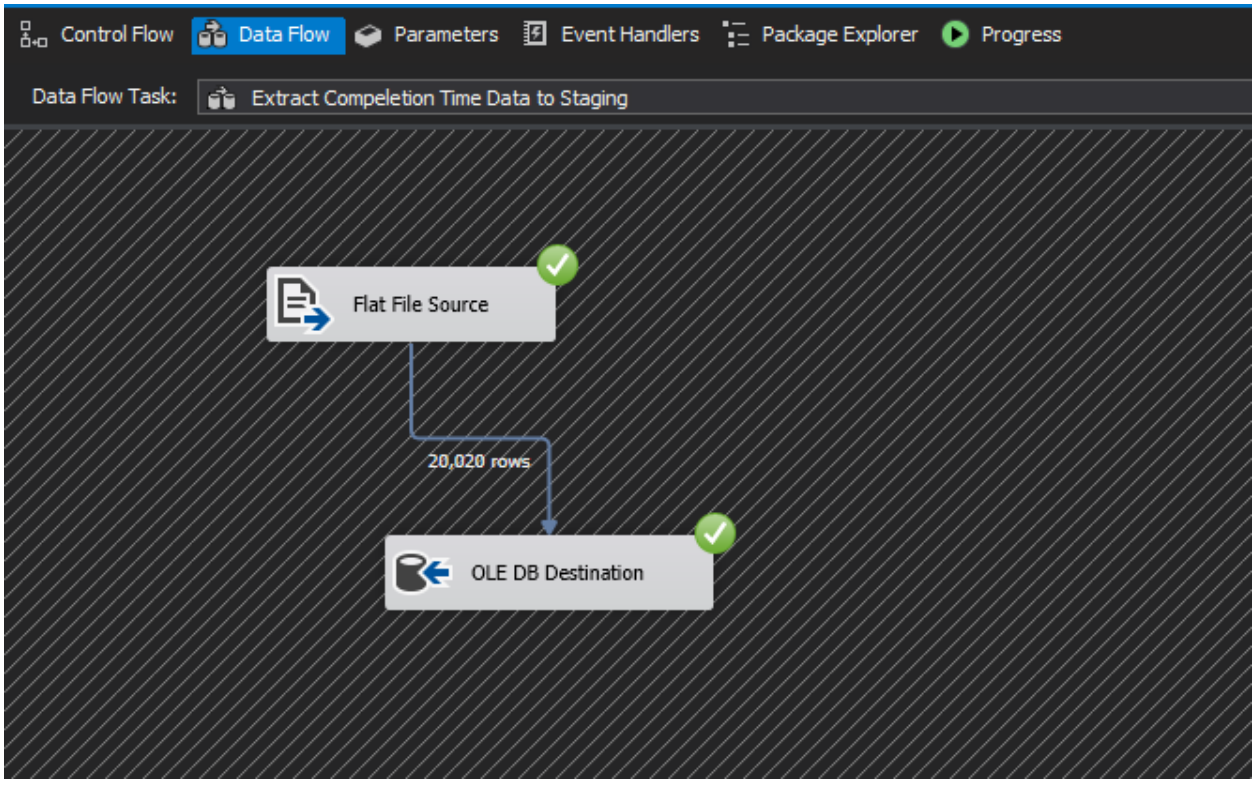


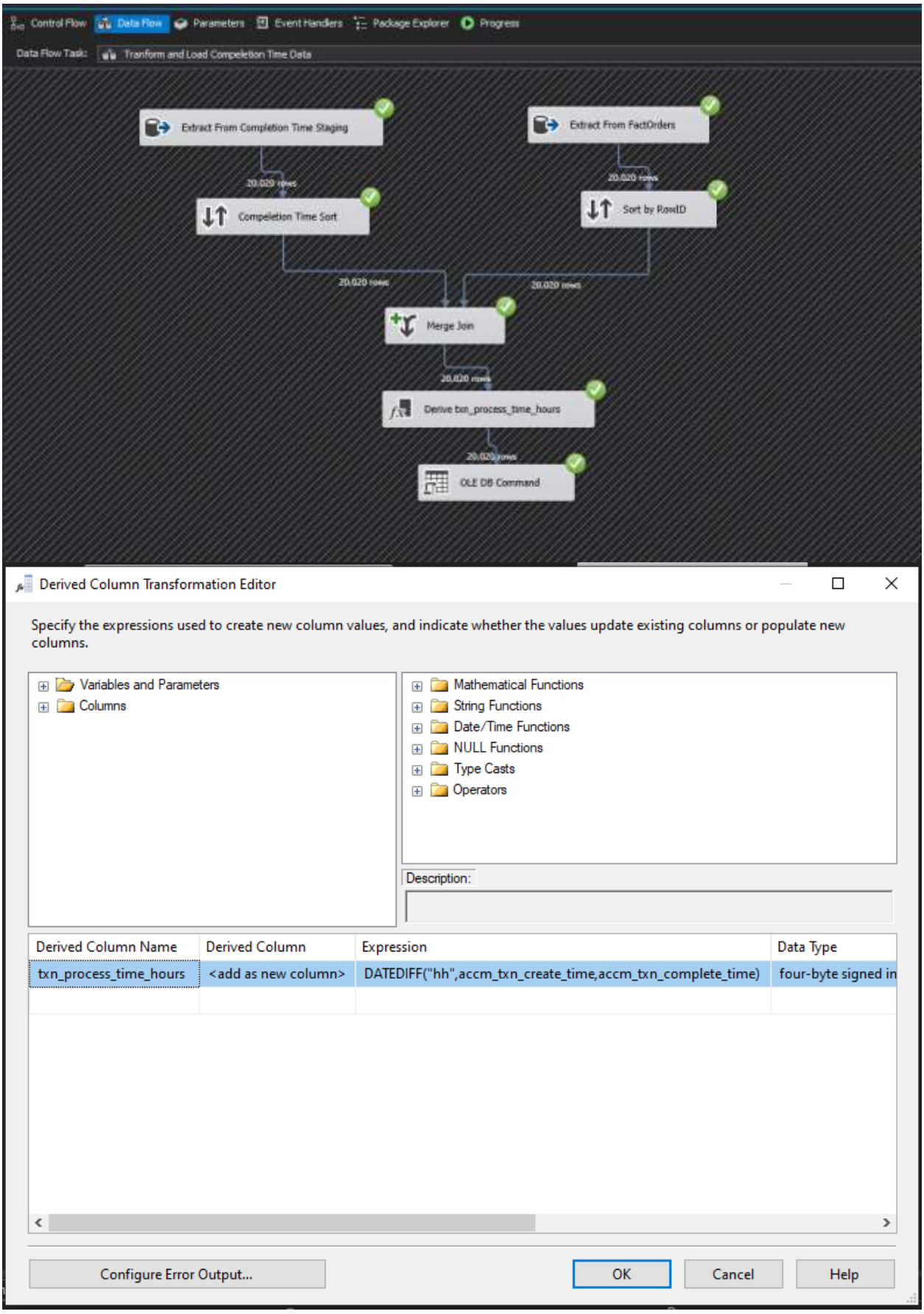
- **Customer Data Transform and Load**



- **Accumulating Fact Table**






Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Tranform and Load Compeletion Time Data
Extract From Completion Time Staging
Extract From FactOrders
20,020 rows
20,020 rows
Compeletion Time Sort
Sort by RowID
20,020 rows
20,020 rows
Merge Join
20,020 rows
Derive txn_process_time_hours
20,020 rows
OLE DB Command
Derived Column Transformation Editor
Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.
Variables and Parameters
Columns
Mathematical Functions
String Functions
Date/Time Functions
NULL Functions
Type Casts
Operators
Description:
Derived Column Name
Derived Column
Expression
Data Type
txn_process_time_hours
<add as new column>
DATEDIFF("hh",accm_txn_create_time,accm_txn_complete_time)
four-byte signed in
Configure Error Output...
OK
Cancel
Help