Sri Lanka Institute of Information Technology

Assignment I

Data Warehouse & Business Intelligence

2022

Submitted By:

Ariyasinghe P.A.D.N.I

IT20033828

Table of Contents

1. Data set selection

The selected dataset is taken from "Kaggle website" which is about a food delivery company. With different operational hubs spread throughout Brazil, the Food Delivery Centers are the places that takes food orders from the users and do deliveries in the region. Firstly, the users required to order foods and make necessary payments online through the system. After the food making process is finished, the paid order will be delivered to relevant places by the delivery partners (drivers) spread across all regions of the country.

Modifications were done to the dataset as needed to meet the requirements as mentioned in the guidelines.

Link to the source dataset is given below.

https://www.kaggle.com/datasets/nosbielcs/brazilian-delivery-center?select=hubs.csv

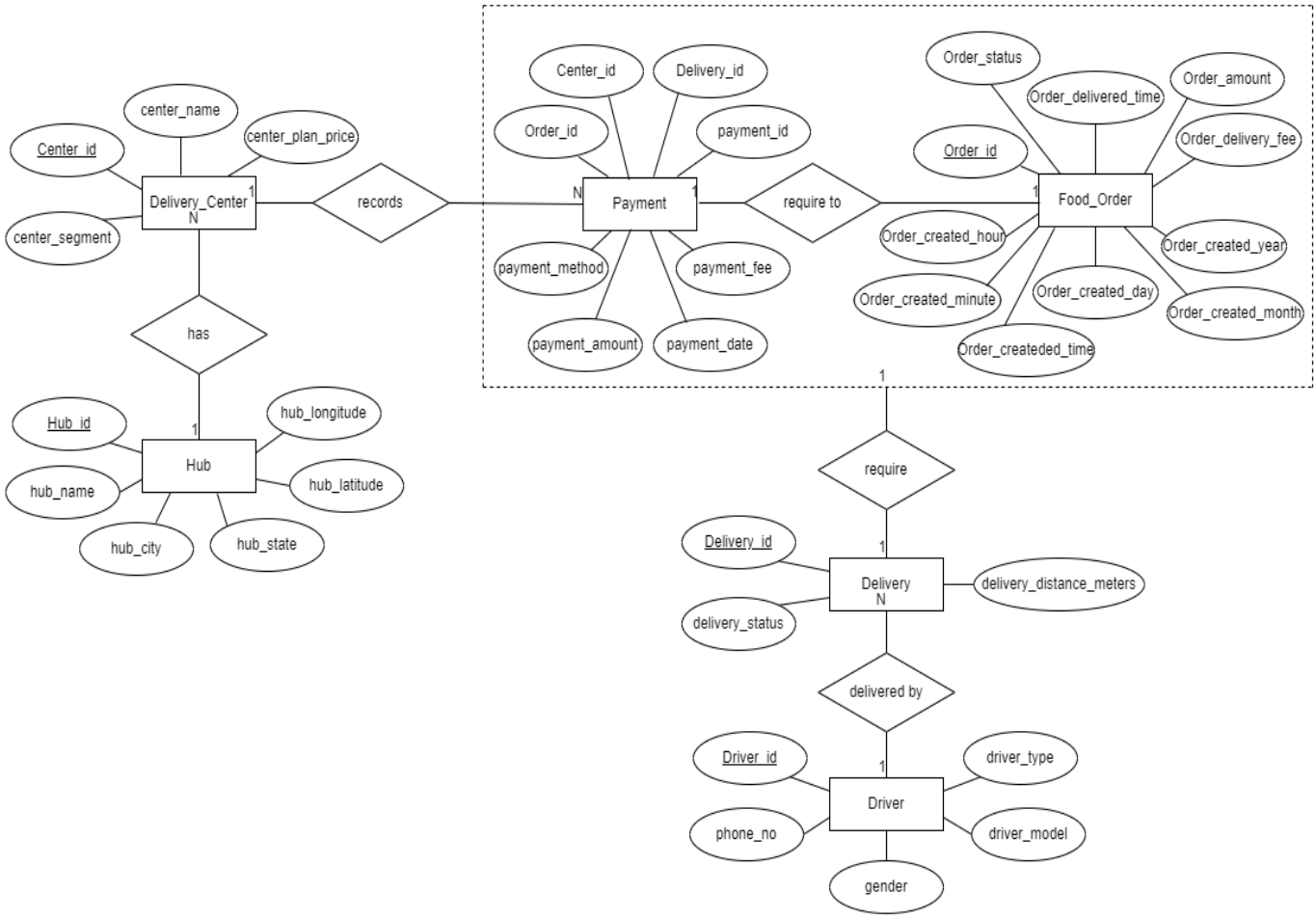
The main sources are listed below:

- SQL Database
- One text file – Drivers Data
- CSV files
  - Deliveries
  - Hubs
  - Food Orders
  - Payments
  - Delivery Centers

Also, to calculate the time to complete a payment for an order in the assignment, I have created a **sql table** as "Complete_time_details" with following two columns:

1. fact_table_natural_key (tonnid) – int (By considering payment_id as a unique key)
2. accm_txn_complete_time - datetime

1.1. Er diagram:

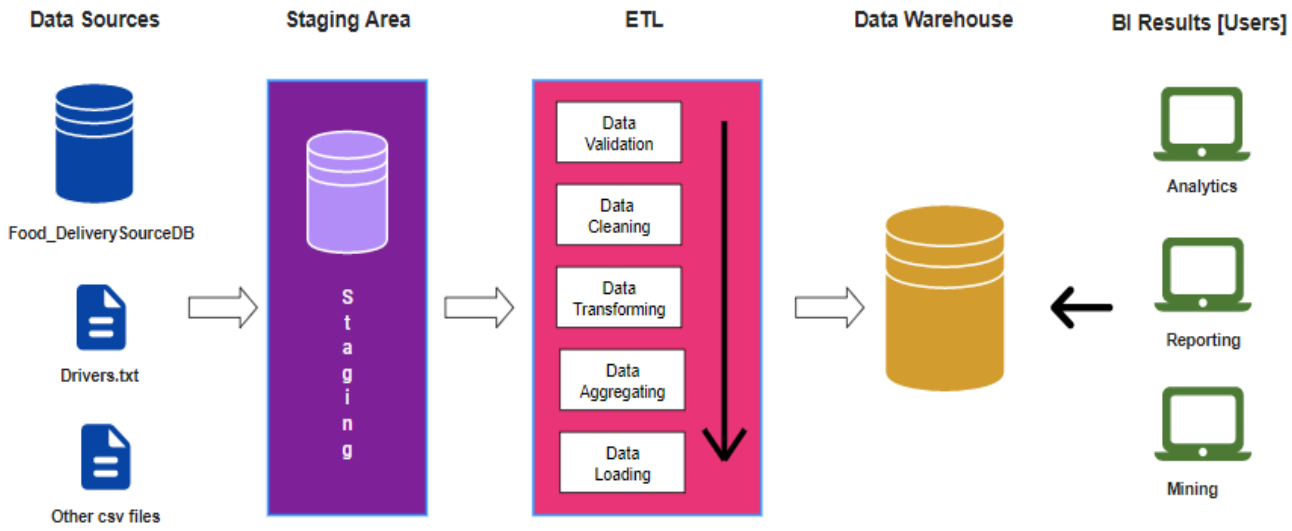


2. Preparation of Data sources - Description of the dataset

Table name	Column name	Data type	Description
Hubs	Hub_id	tinyint	Summary of the hubs
	Hub_name	nvarchar(50)	
	Hub_city	nvarchar(50)	
	Hub_state	nvarchar(50)	
	Hub_latitude	float	
	Hub_longitude	float	
Delivery Centers	Center_id	smallint	Summary of the delivery centers which are delivering food orders
	Hub_id	Tinyint	
	Center_name	nvarchar(50)	
	Center_segment	nvarchar(50)	
	Center_plan_price	float	
Deliveries	Delivery_id	int	Details of payments
	Driver_id	int	
	Delivery_distance	int	
	Delivery_status	nvarchar(50)	
Drivers	driver_id	int	Details of drivers who delivers food orders
	driver_modal	nvarchar(50)	
	driver_type	nvarchar(50)	
	phone_no	nvarchar(50)	
	gender	nvarchar(50)	
Payments	Order_id	Int	Details of the customer transactions
	Center_id	smallint	
	Delivery_id	int	
	Payment_id	int	
	Payment_amount	Float	
	Payment_fee	Float	
	Payment_method	Nvarchar(50)	
	Payment_date	datetime	
Food Orders	Order_id	Int	Details about the food orders made through the delivery center platform
	Order_status	Nvarchar(50)	
	Order_amount	Float	
	Order_delivery_fee	Float	
	Order_created_hour	Tinyint	
	Order_created_minute	Tinyint	
	Order_created_day	Tinyint	
	Order_created_year	smallint	

	Order_created_time	Datetime2	
	Order_delivered_time	Datetime2	

3. Solution Architecture

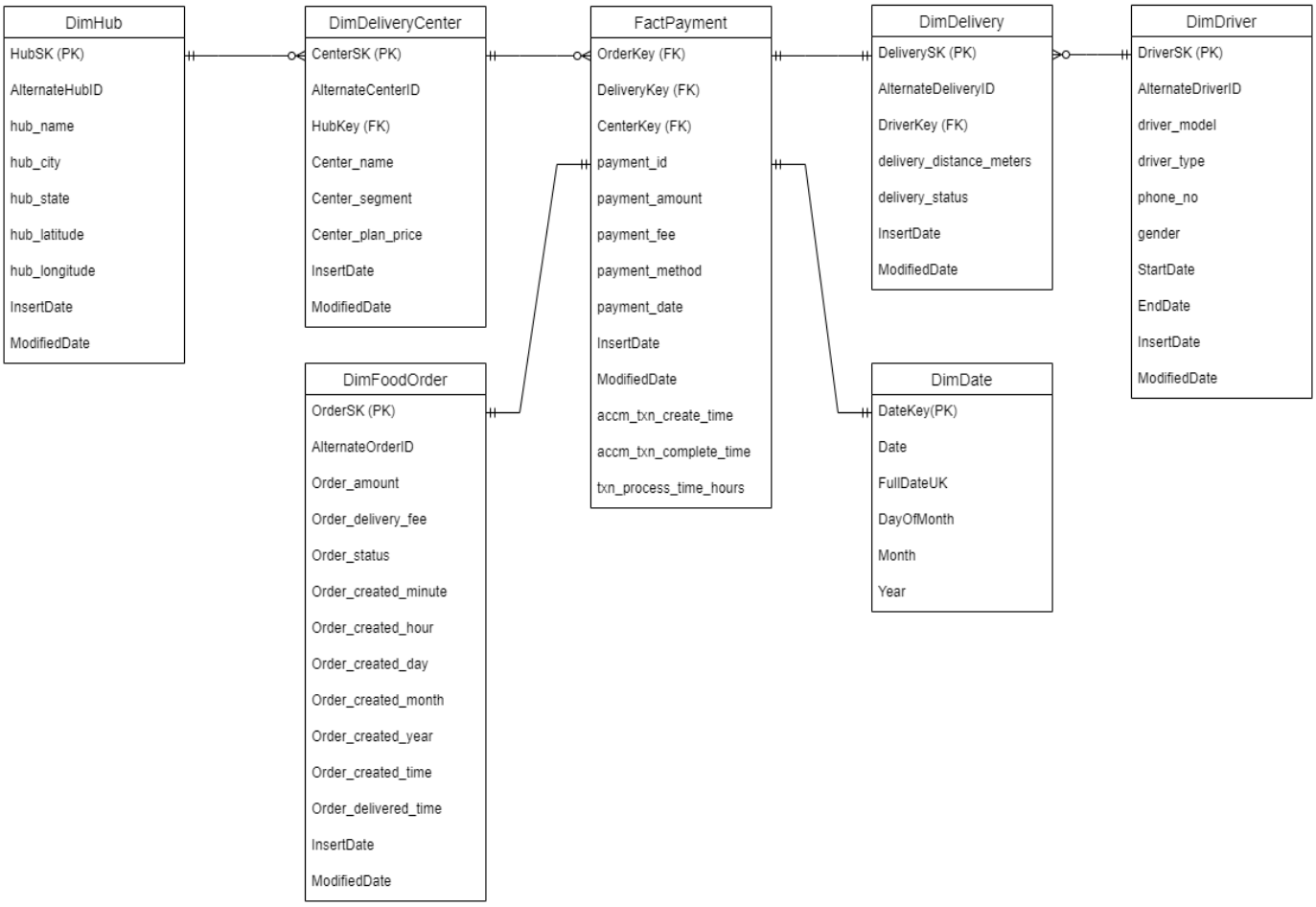


After creating the Food_DeliverySourceDB with the help of other source files, the first step to do is staging the source data set. After the staging layer developed, the below mentioned staging tables were created: StgHubs, StgDeliveryCenters, StgDeliveries, StgDrivers, StgFoodOrders and StgPayments.

Then the staged tables were profiled, and aggregations were performed when necessary. The next step is data transforming and loading with the ETL process. After completing the described stages, the Datawarehouse is created. Names of data warehouse tables are: DimHub, DimDeliveryCenters, DimDrivers, DimDeliveries, DimFoodOrders, DimPayments, DimDate, FactPayments.

After the warehouse is created BI results such as OLAP analysis, Reports, Data visualization, Data mining can be obtained as results after further modifications.

4. Data warehouse design and development



Snowflake schema is used to design the above dataset into a data warehouse. There is one fact table as Fact Payment which contains the transaction details of the food orders. All the others are dimensions. DimDrivers dimension is taken as a slowly changing dimension.

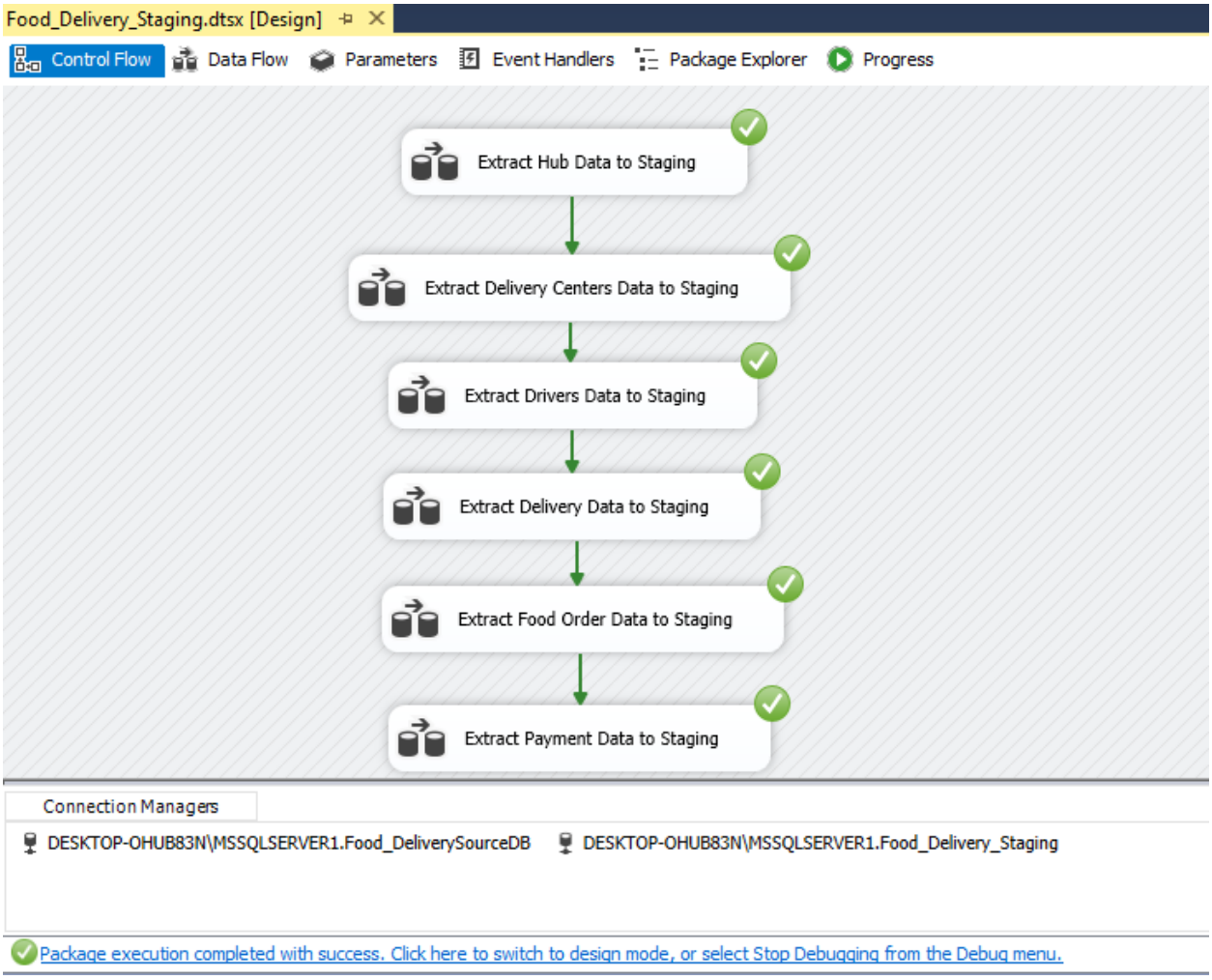
Assumptions:

Users must pay for food orders to confirm the order. If they cancel the order later, the paid money will be returned. Users can pay by the online system or voucher, debit card or paychecks. Online paid orders will be home delivered.

5. ETL Development

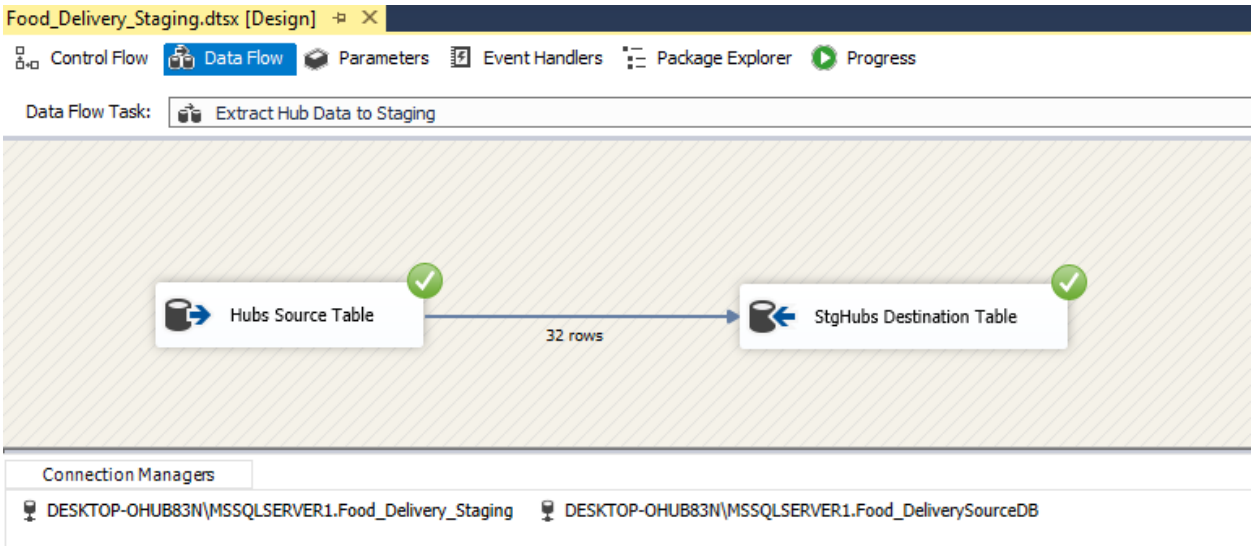
5.1. Source database to Staging database

The first ETL process is extracting data from the "Food_DeliverySourceDB" using data flow tasks and transferring them to the "Food_Delivery_StagingDB." To do this task a SSIS package named "Food_Delivery_Staging" has been created. A truncate table for every staging table was created. All the data flow tasks were joined as shown below at the end:

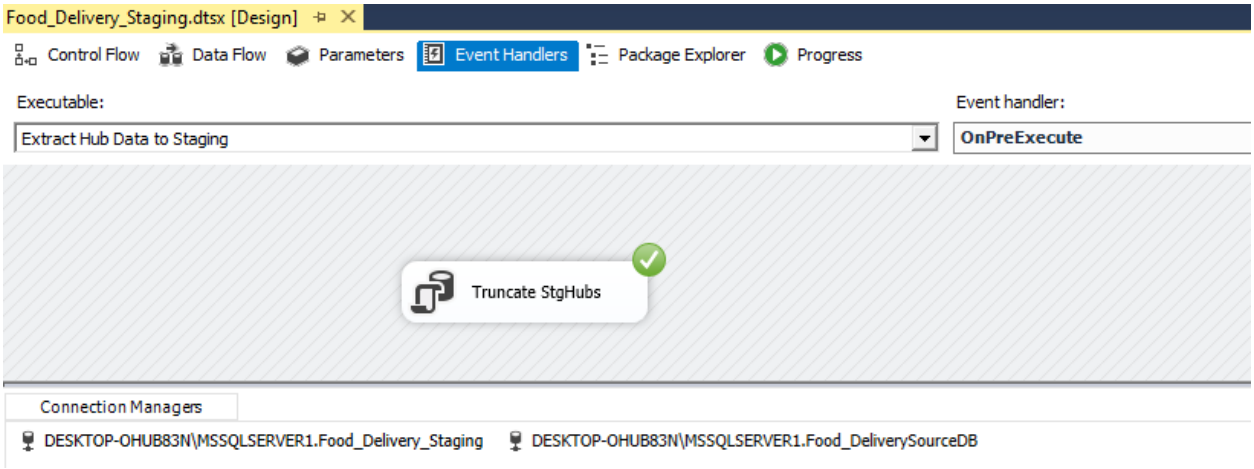


Screenshots of all the data sources that were staged, truncate tables created, and view of the staging tables are attached below:

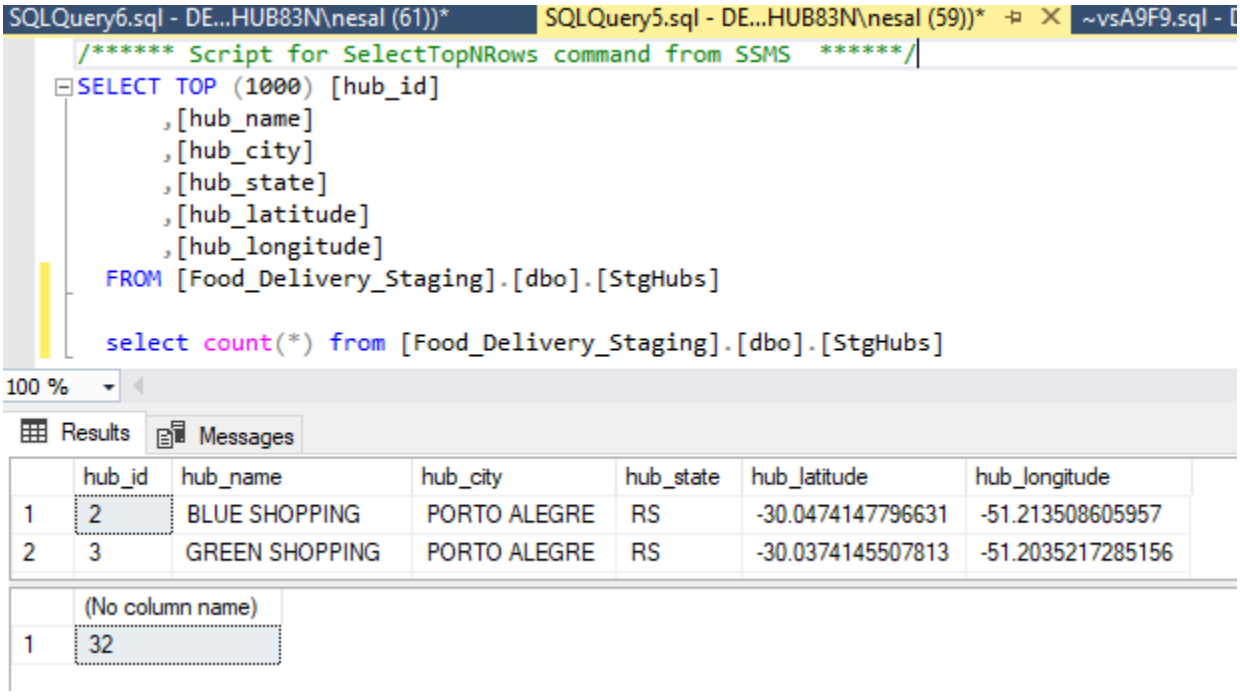
5.1.1. Staging Hub Details:



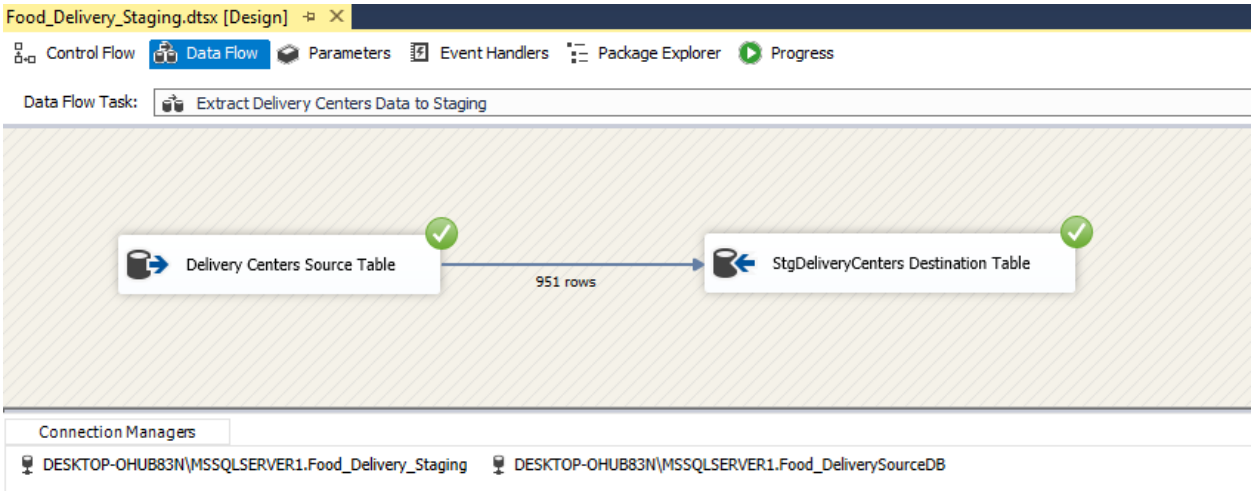
5.1.1.1. Truncate Hubs Staging table:



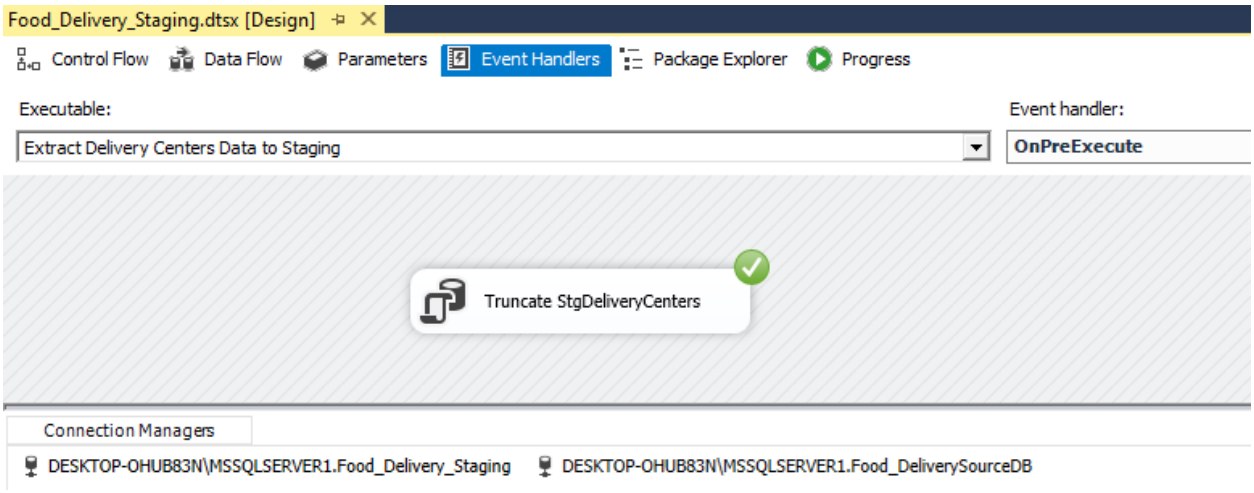
5.1.1.2. Hubs Staging table after extracting data from source database:



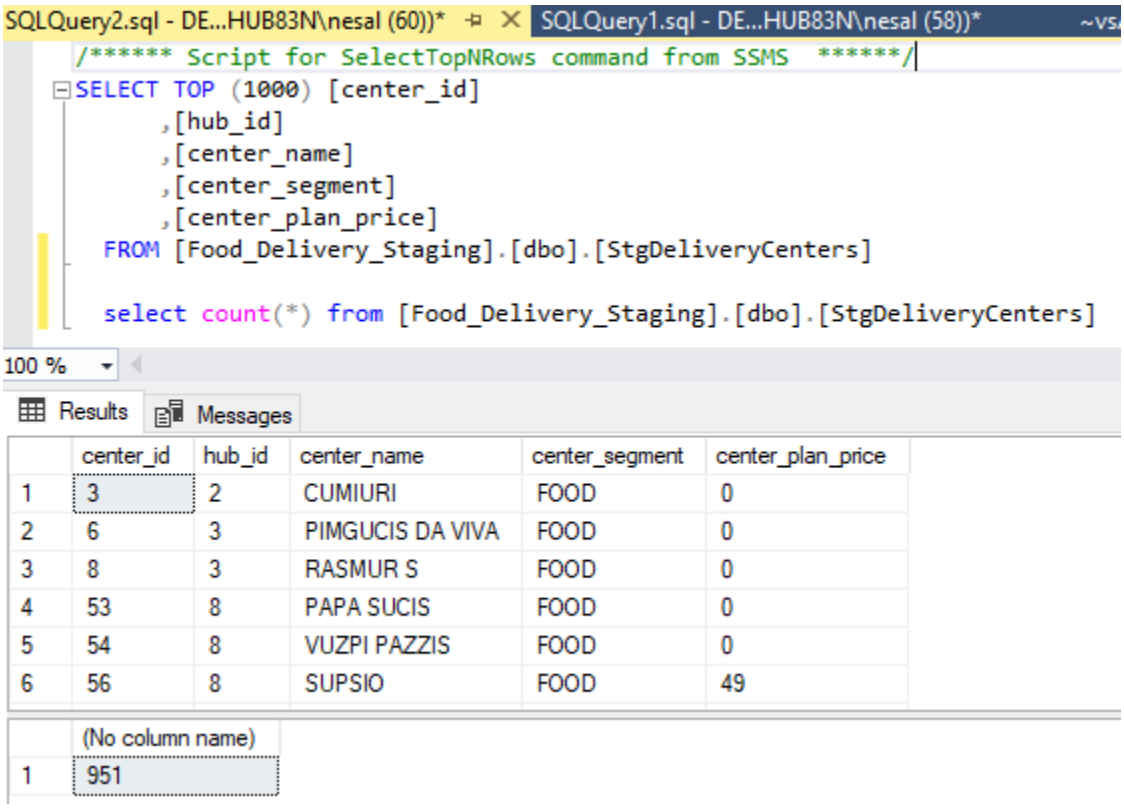
5.1.2. Staging Delivery Center Details:



5.1.2.1. Truncate Delivery Centers Staging table:



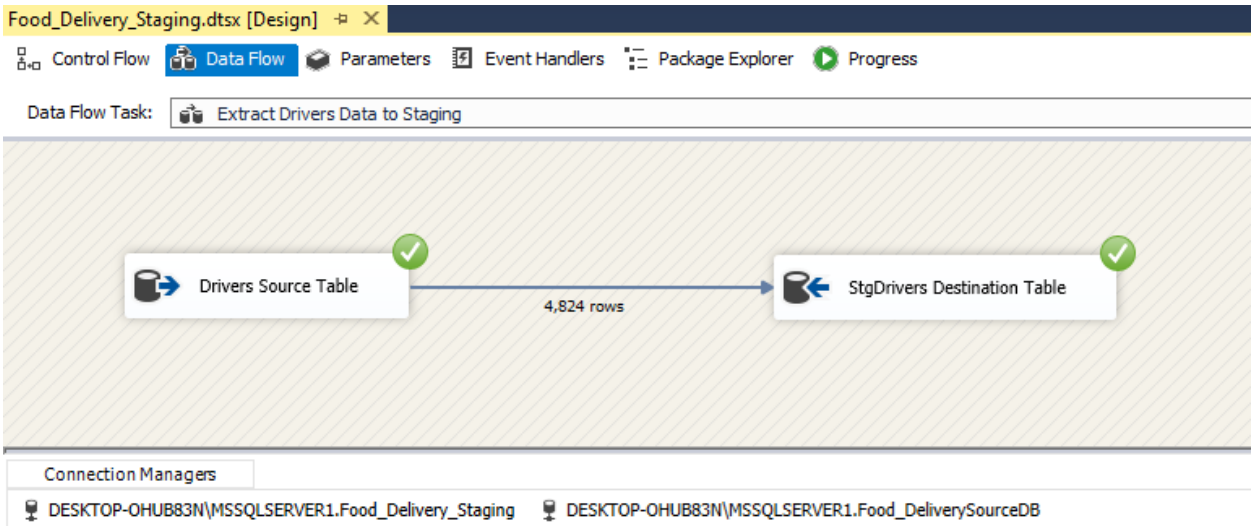
5.1.2.2. Delivery Centers Staging table after extracting data from source database:



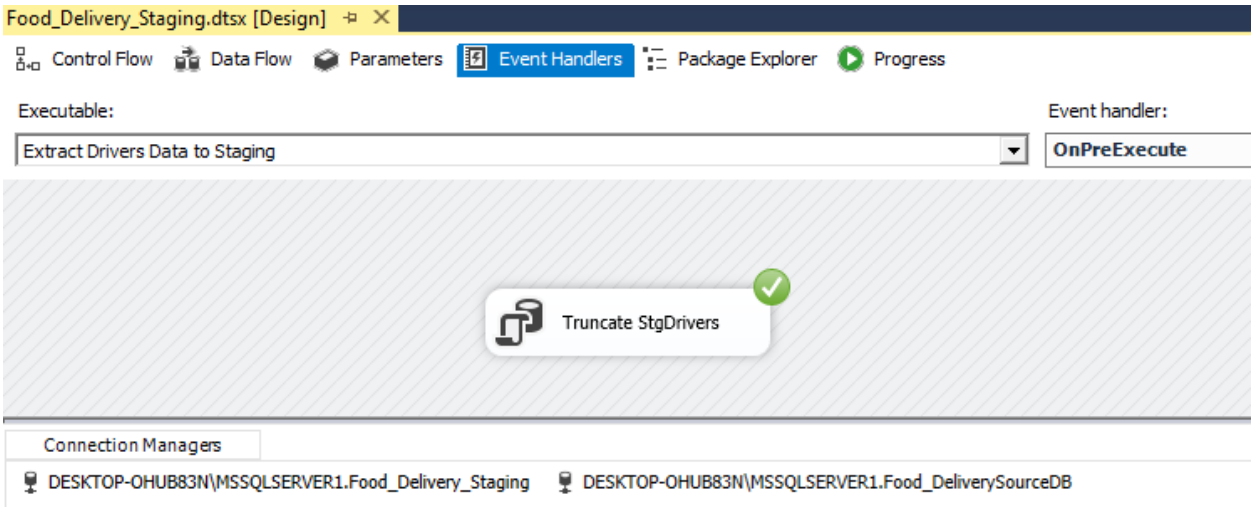
	center_id	hub_id	center_name	center_segment	center_plan_price
1	3	2	CUMIURI	FOOD	0
2	6	3	PIMGUCIS DA VIVA	FOOD	0
3	8	3	RASMUR S	FOOD	0
4	53	8	PAPA SUCIS	FOOD	0
5	54	8	VUZPI PAZZIS	FOOD	0
6	56	8	SUPSIO	FOOD	49

	(No column name)
1	951

5.1.3. Staging Driver Details:



5.1.3.1. Truncate Drivers Staging table:



5.1.3.2. Drivers Staging table after extracting data from source database:

SQLQuery3.sql - DE...HUB83N\nesal (61))* | SQLQuery2.sql - DE...HUB83N\nesal (60))*

```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [driver_id]
      ,[driver_modal]
      ,[driver_type]
      ,[phone_no]
      ,[gender]
  FROM [Food_Delivery_Staging].[dbo].[StgDrivers]

  select count(*) from [Food_Delivery_Staging].[dbo].[StgDrivers]
```

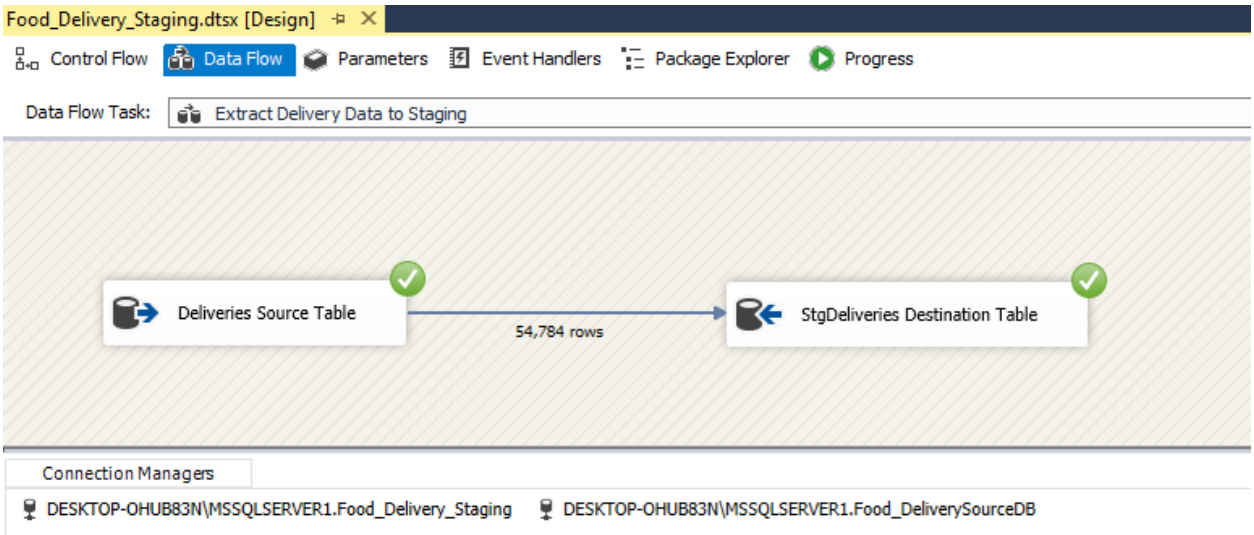
100 %

Results | Messages

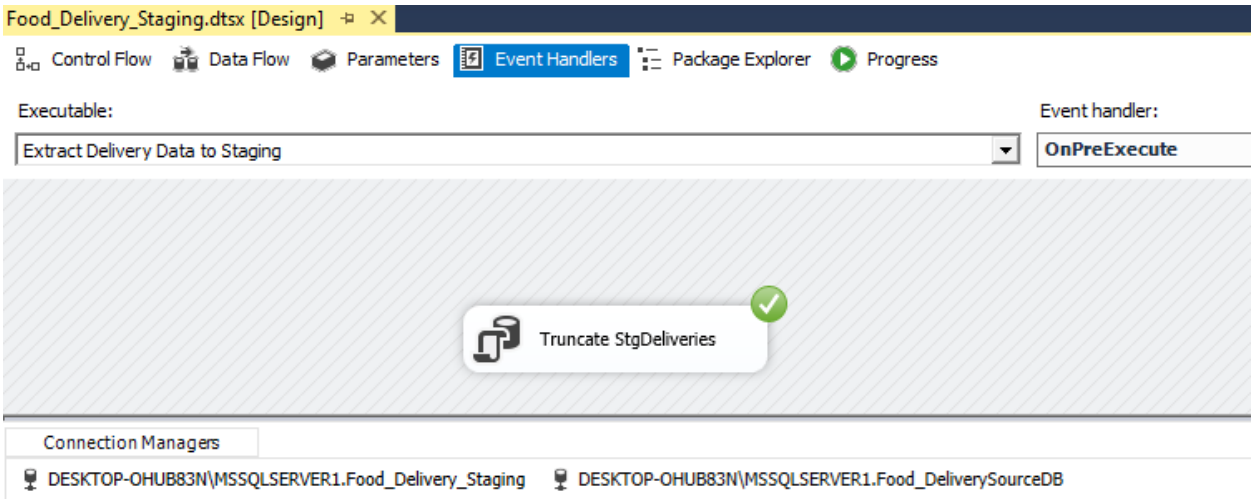
	driver_id	driver_modal	driver_type	phone_no	gender
1	38780	MOTOBOY	FREELANCE	732-555-0139	F
2	38803	MOTOBOY	FREELANCE	912-555-0186	F
3	38813	MOTOBOY	FREELANCE	1 (11) 500 555-0147	F

	(No column name)
1	4824

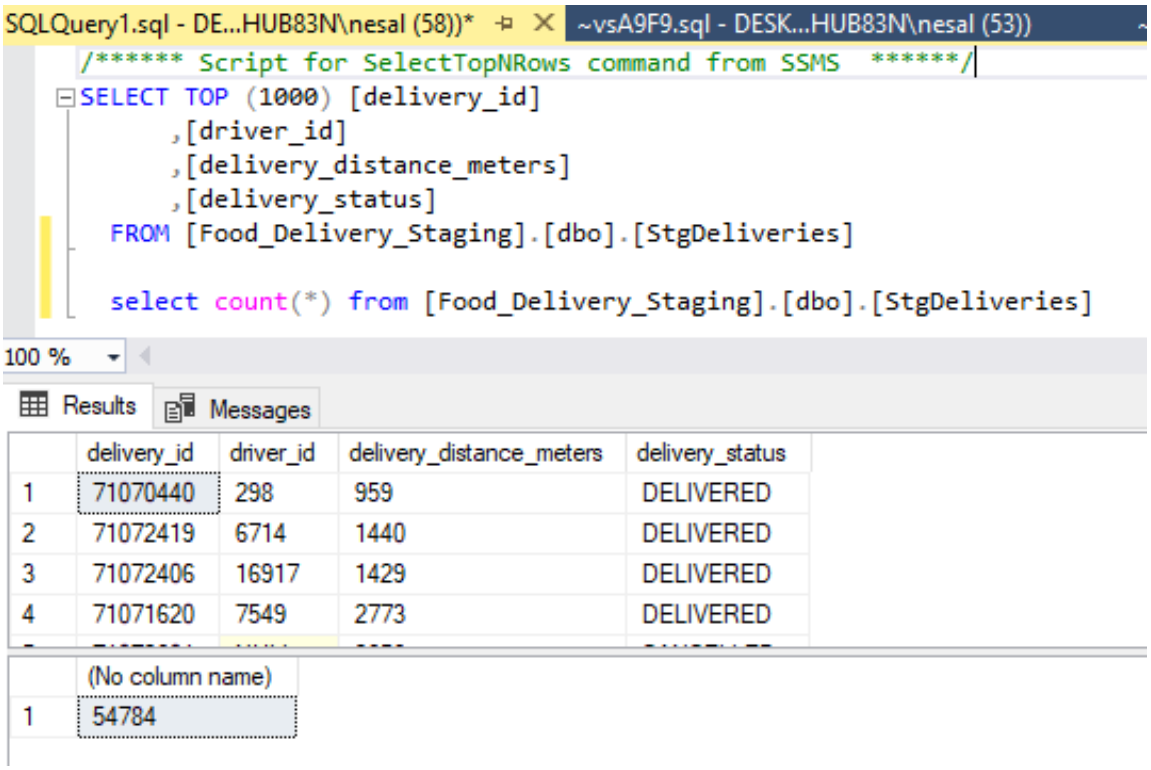
5.1.4. Staging Delivery Details:



5.1.4.1. Truncate Deliveries Staging Table:



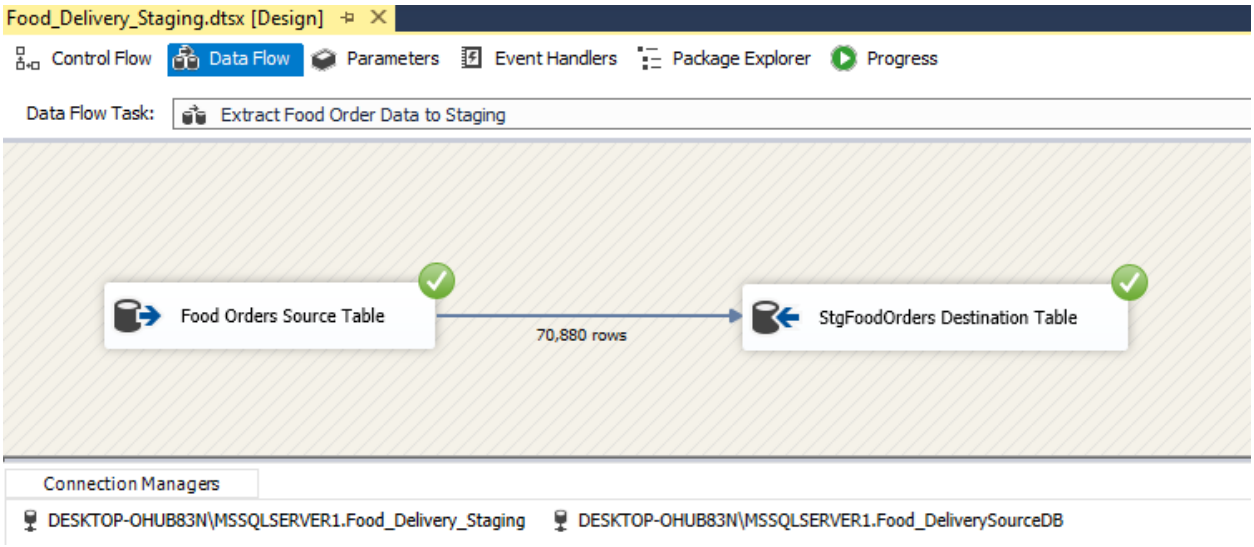
5.1.4.2. Deliveries Staging table after extracting data from source database:



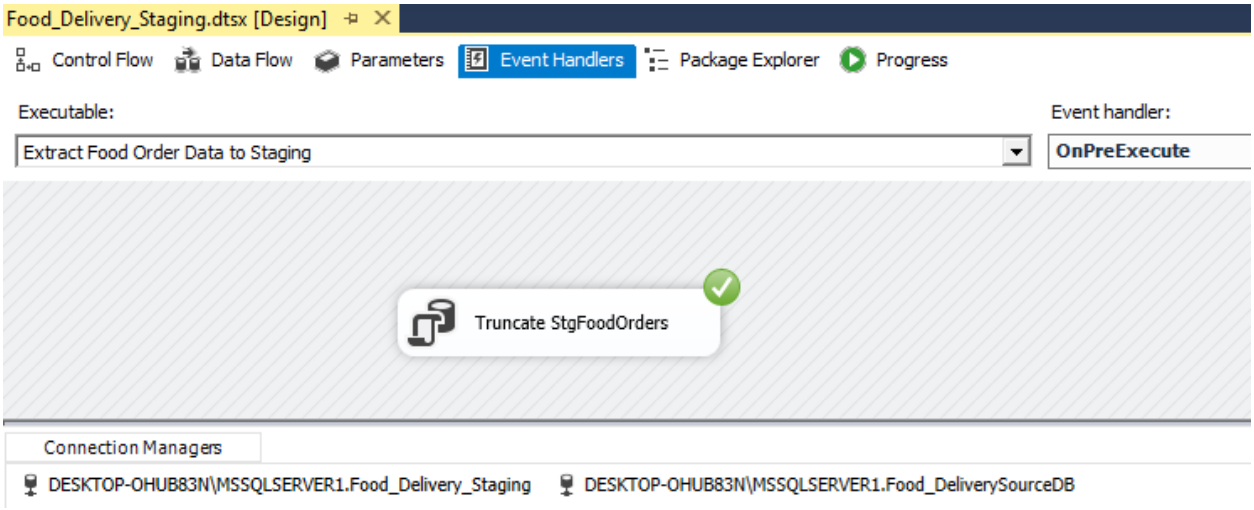
	delivery_id	driver_id	delivery_distance_meters	delivery_status
1	71070440	298	959	DELIVERED
2	71072419	6714	1440	DELIVERED
3	71072406	16917	1429	DELIVERED
4	71071620	7549	2773	DELIVERED

	(No column name)
1	54784

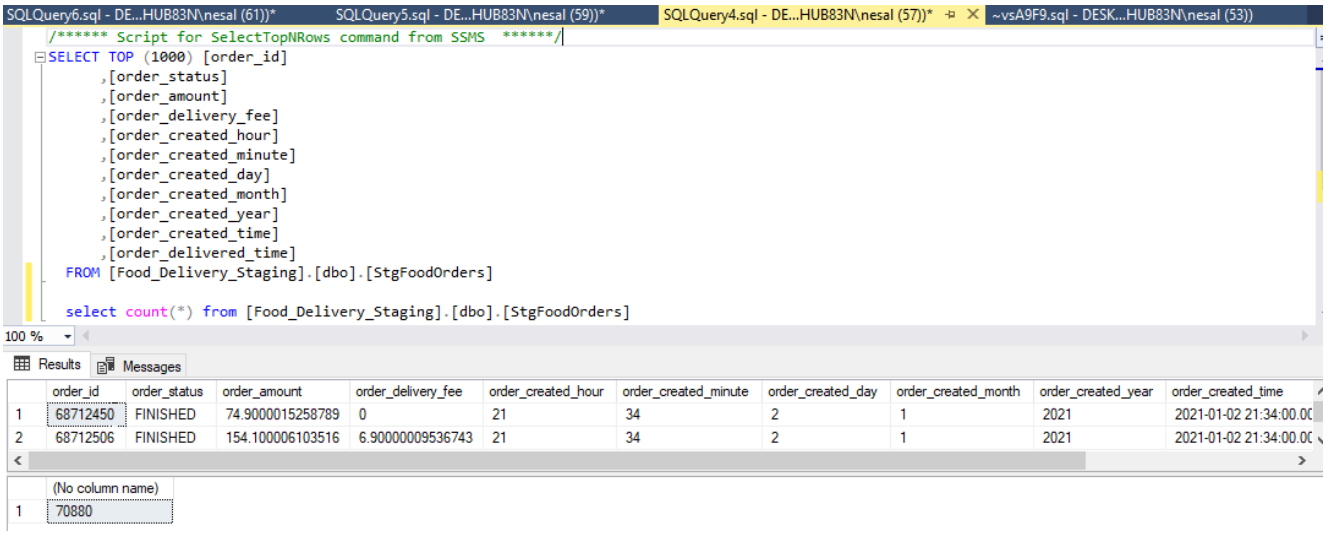
5.1.5. Staging Food Order Details:



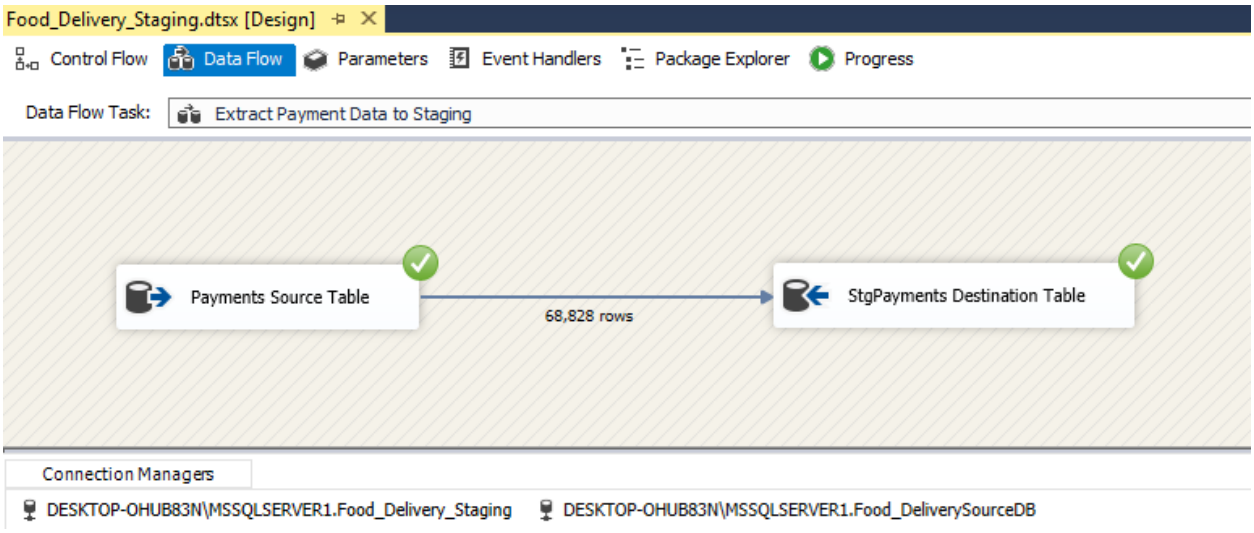
5.1.5.1. Truncate Food Orders Staging table:



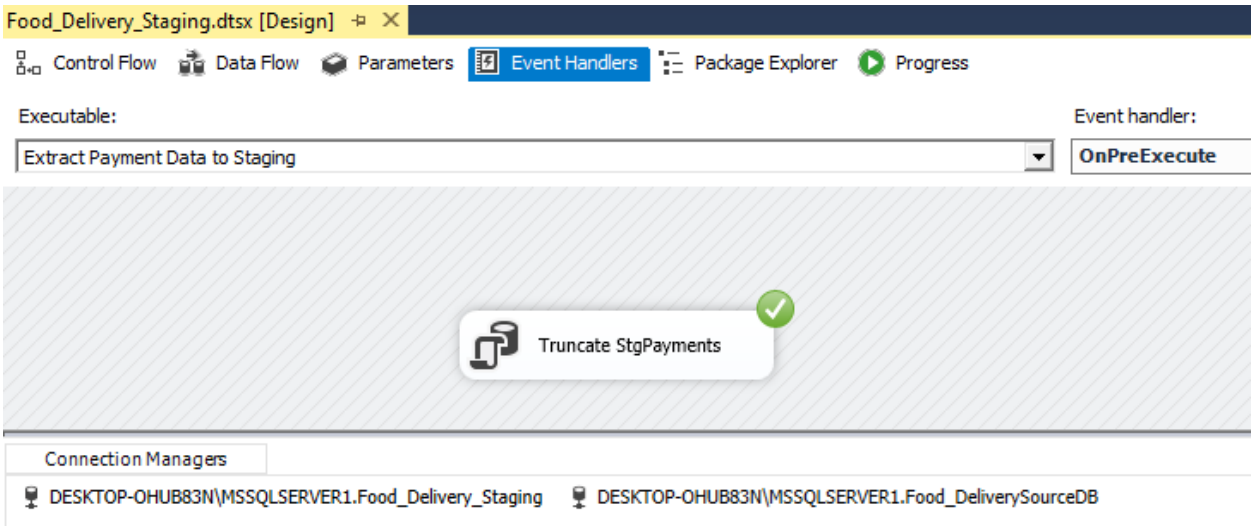
5.1.5.2. Food Orders Staging table after extracting data from source database:



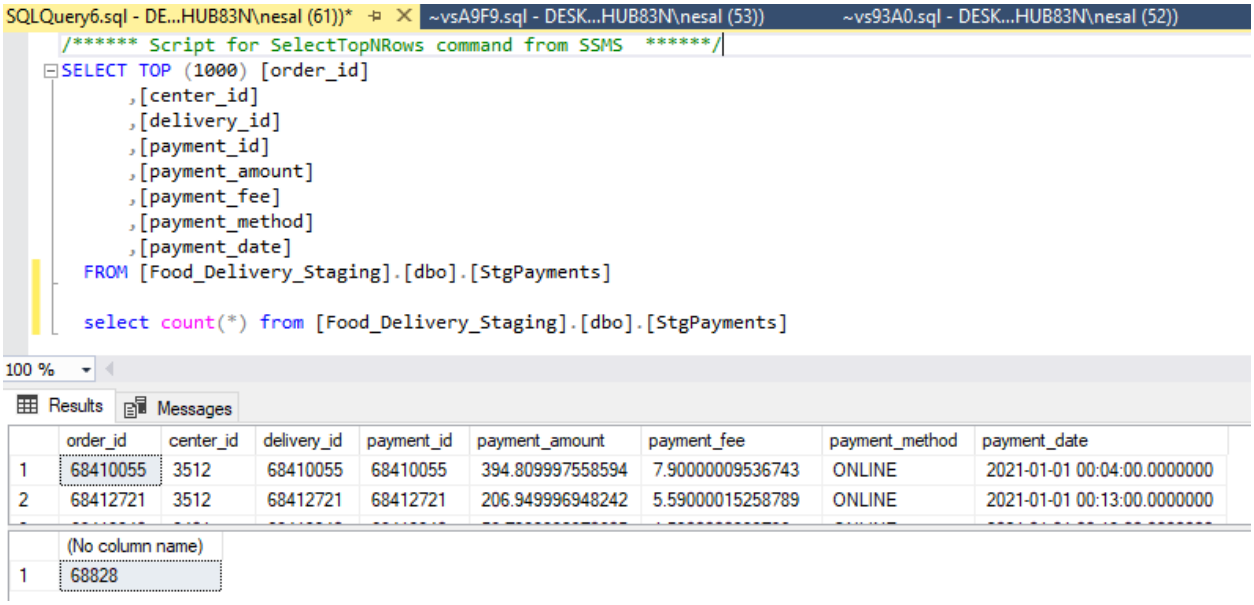
5.1.6. Staging Payment Details:



5.1.6.1. Truncate Payments Staging table:



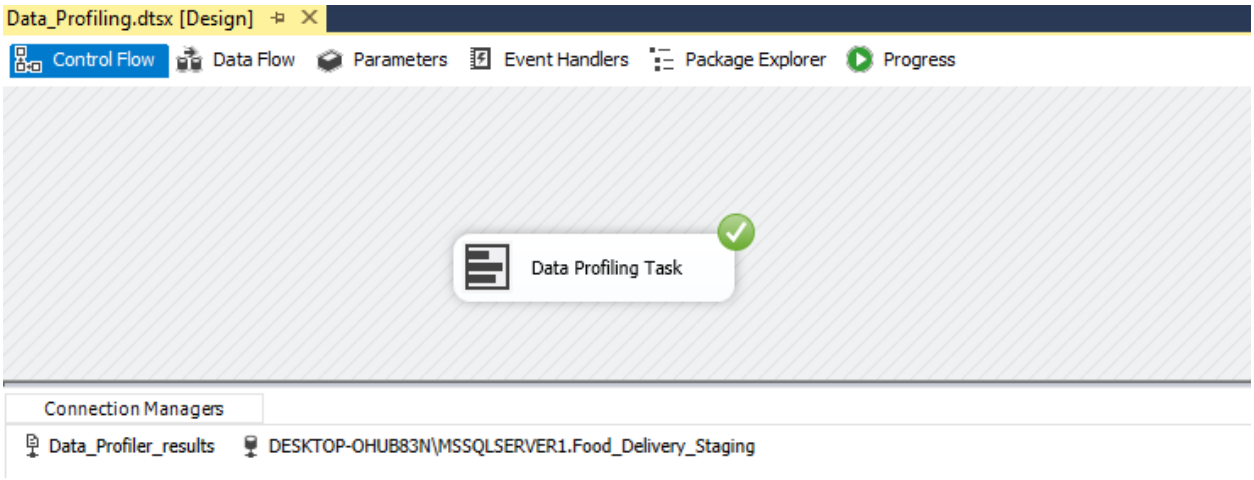
5.1.6.2. Payments Staging table after extracting data from source database:



5.2. Data Profiling:

Next, to use the staging table data to analyze how the data looks like to determine what type of transformations we need to perform on the data, a **data profiling** task has been executed by selecting all tables option as shown below:

A separate SSIS package named “Data_Profiling” was created for this task.

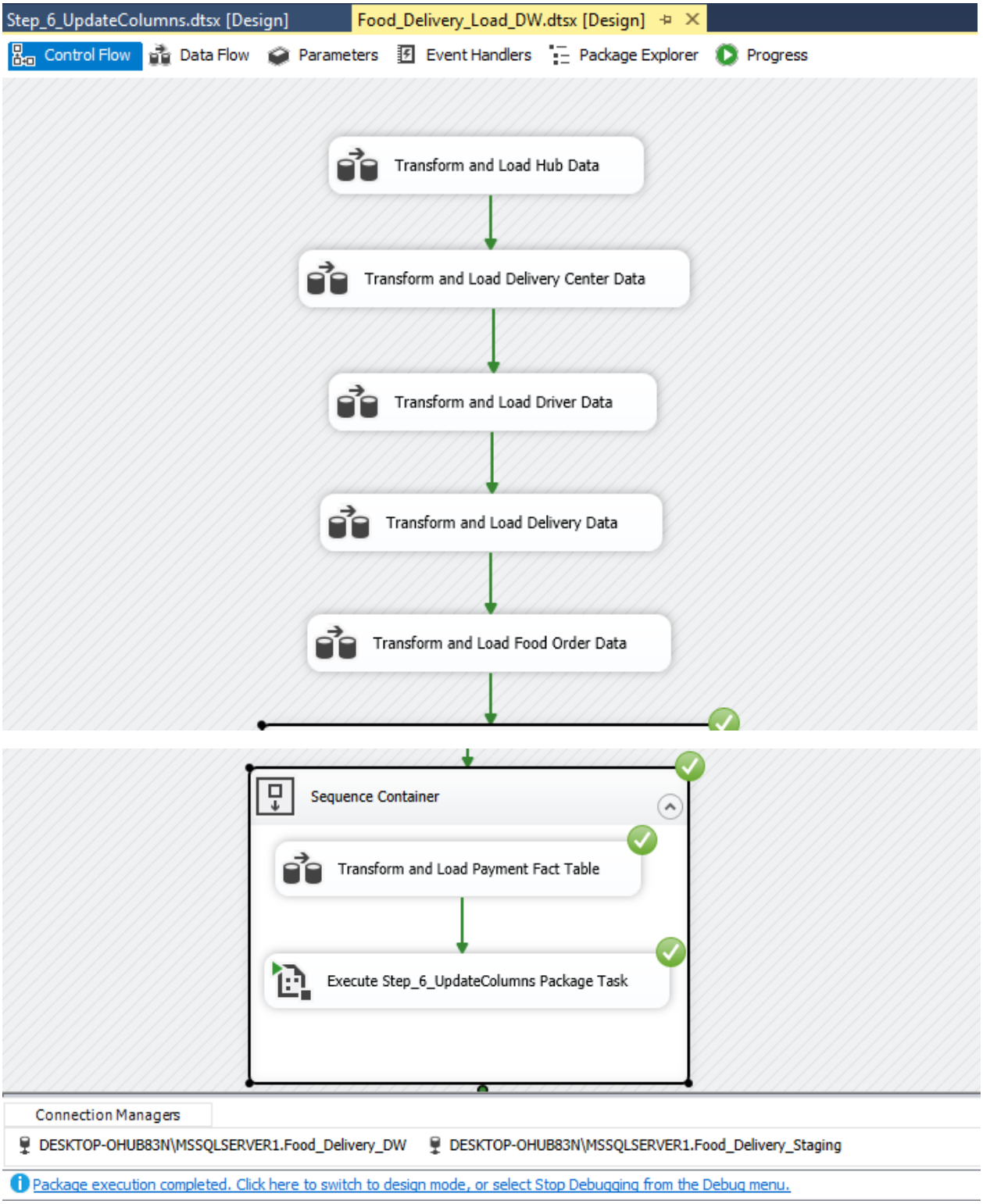


5.3. Staging database to Data Warehouse:

The second ETL process is to transfer data from Staging database to Data Warehouse. Unlike the first ETL process this process has a lot of transformations to be done to the data before they can be loaded into the data warehouse dimension and fact tables such as data transformations, data cleaning, data enrichments. In the second ETL process the ETL task execution order is very most important. To preserve the links between dim and fact tables and to populate derived key columns with relevant surrogate keys, business keys and surrogate keys have been created.

Data Flow tasks have not been assigned any event handlers to truncate any of the data warehouse tables because truncating data warehouse tables may cause issues of unexpected surrogate key changes which might be the root cause to end up with a faulty set of data in the data warehouse. Data for all data warehouse tables are either updated or inserted only.

A separate SSIS package named “Food_Delivery_Load_DW” has been created to transform and load data from staging database to data warehouse.



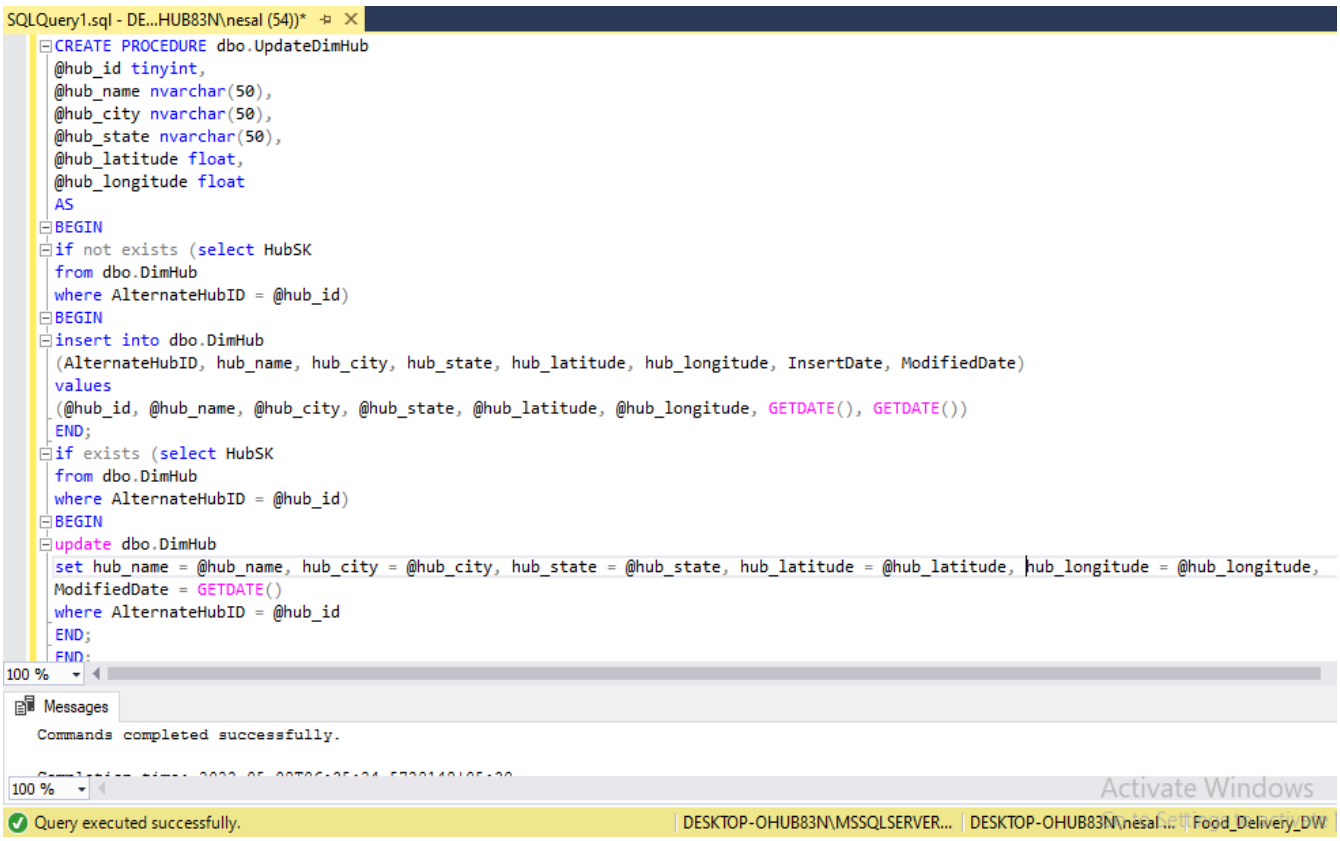
Before performing any ETL task, fact and dimension tables needed to be created in data warehouse. Then the ETL tasks were executed.

5.3.1. Transform and Load Hubs data to Data Warehouse:

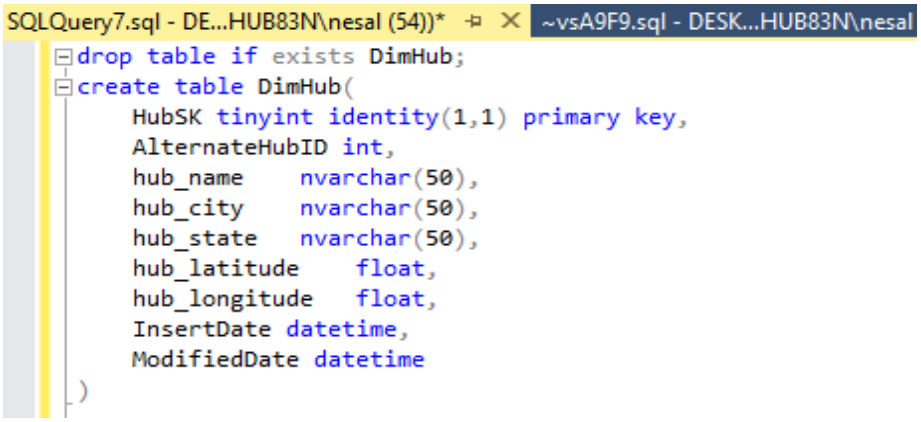
Since there were no transformations to be done, the extracted data is sent to an OLE DB command component to pass the data to a stored procedure in the data warehouse to do the insert and update accordingly.

Stored procedure call: **exec dbo.UpdateDimHub ?, ?, ?, ?, ?, ?**

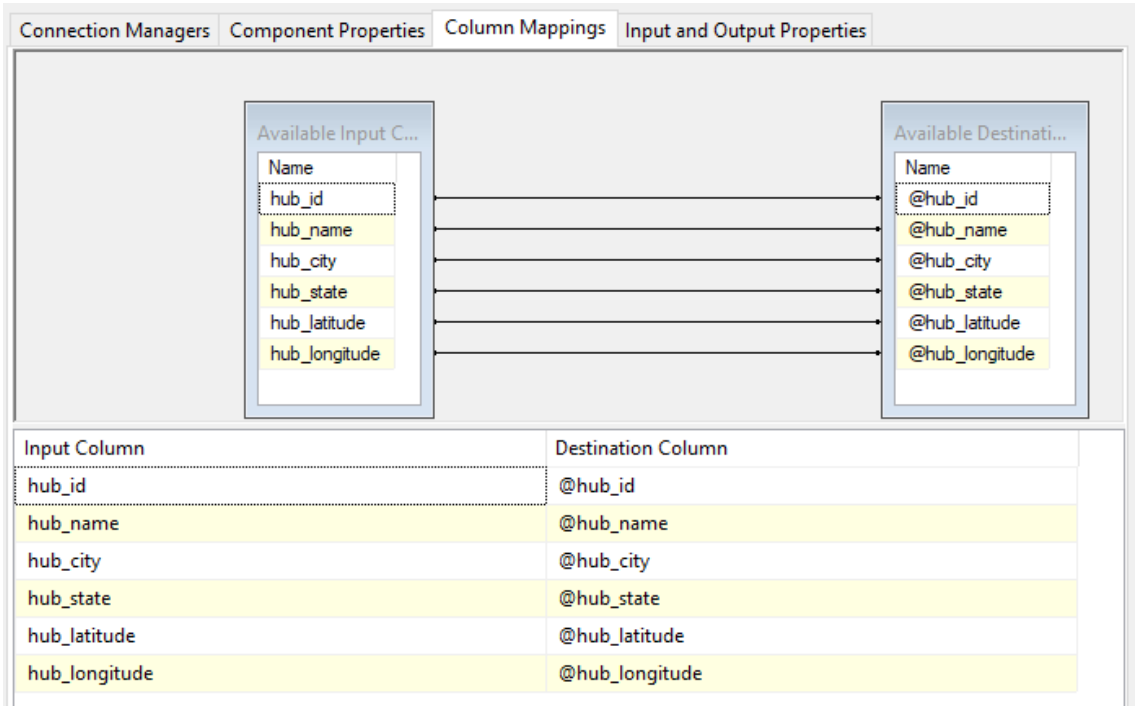
Stored procedure to do insertions and updates: UpdateDimHub

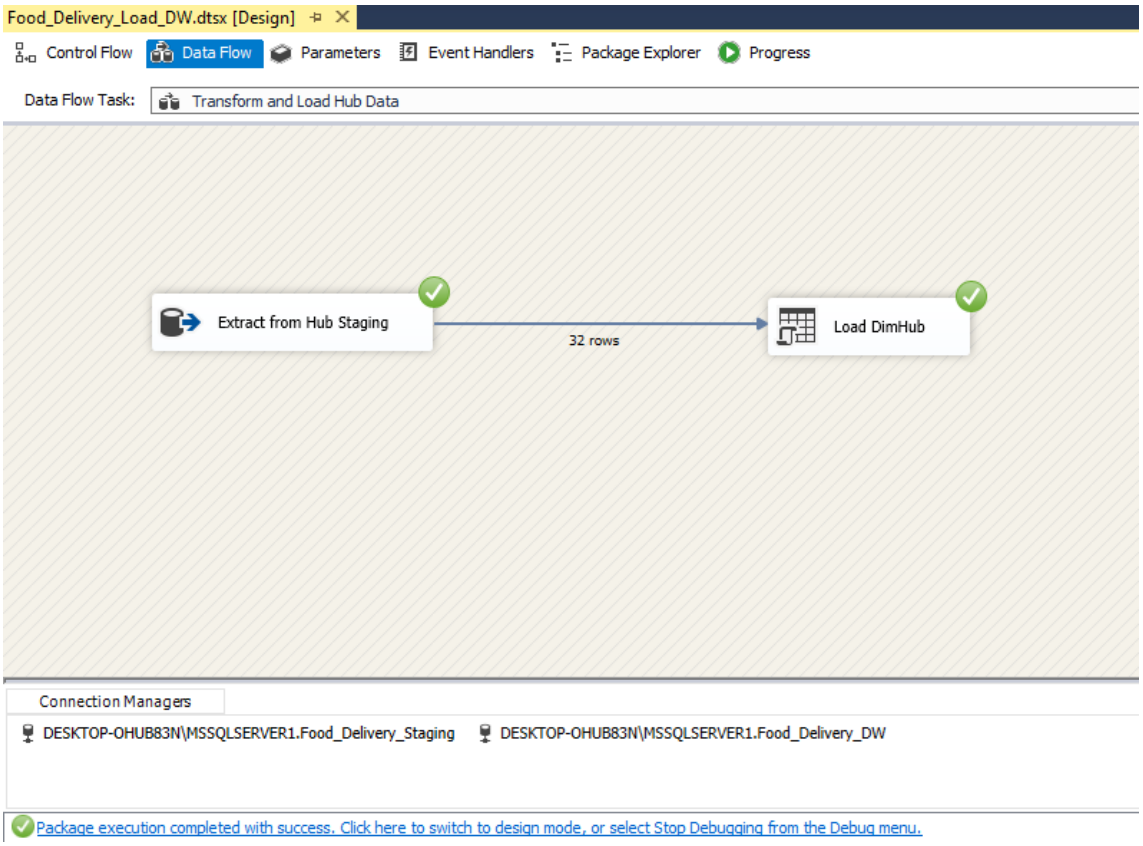


SQL query to create the Hub Dimension: DimHub



Column mappings to load data:



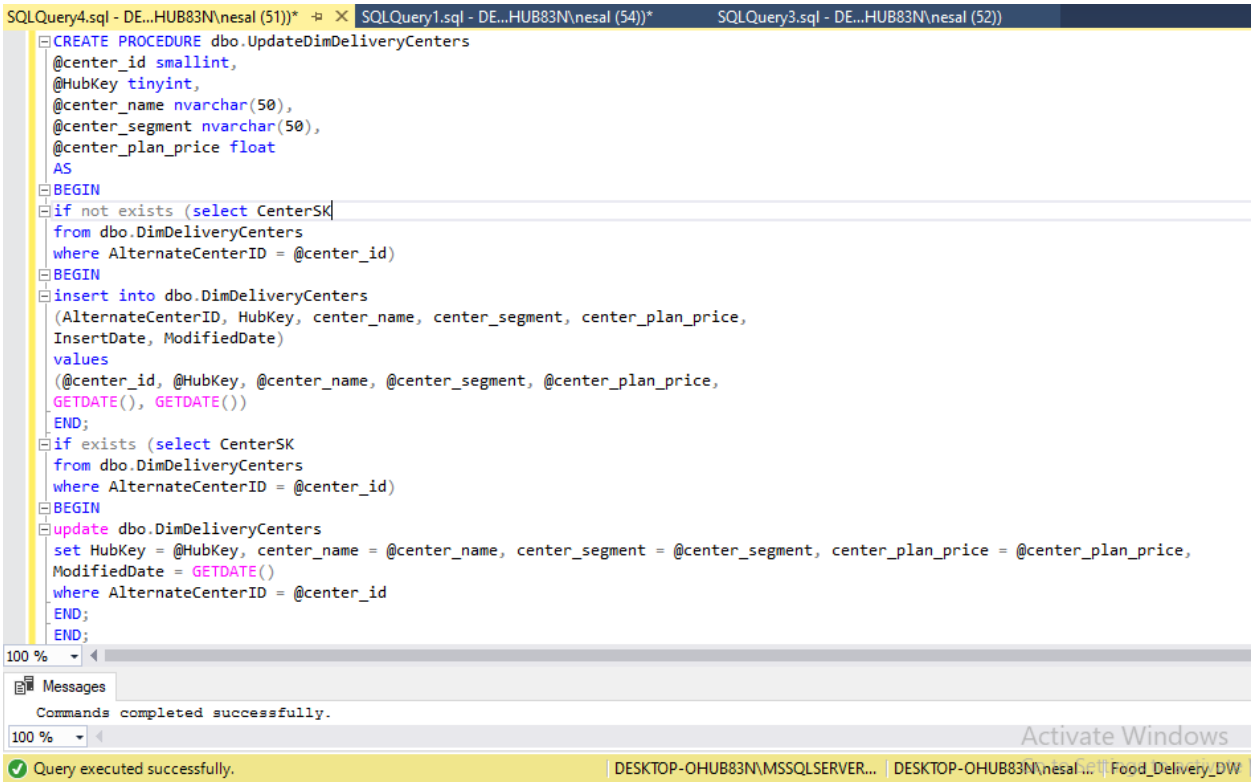


5.3.2. Transform and Load Delivery Centers data to Data Warehouse:

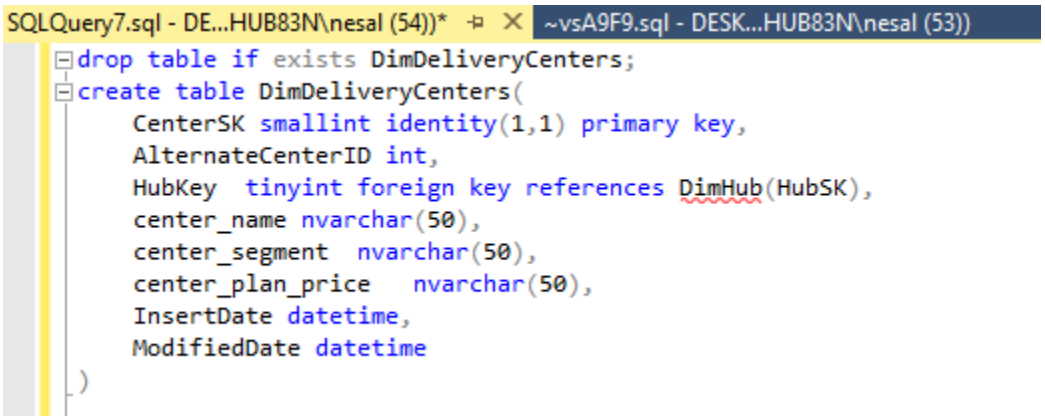
Since it has foreign key reference to Hub dimension it is chosen secondly.

Stored procedure call: **exec dbo.UpdateDimDeliveryCenters ?, ?, ?, ?, ?**

Stored procedure to do insertions and updates: UpdateDimDeliveryCenters



SQL query to create the Delivery Centers Dimension: DimDeliveryCenters



Since delivery_centers table has a foreign key from hubs table, both delivery_centers and hubs tables needed to be joined in order to load and transfer data from delivery centers staging to the delivery center dimension table. We can use either merge join component or a lookup component to do this task. Here I have used the merge join method.

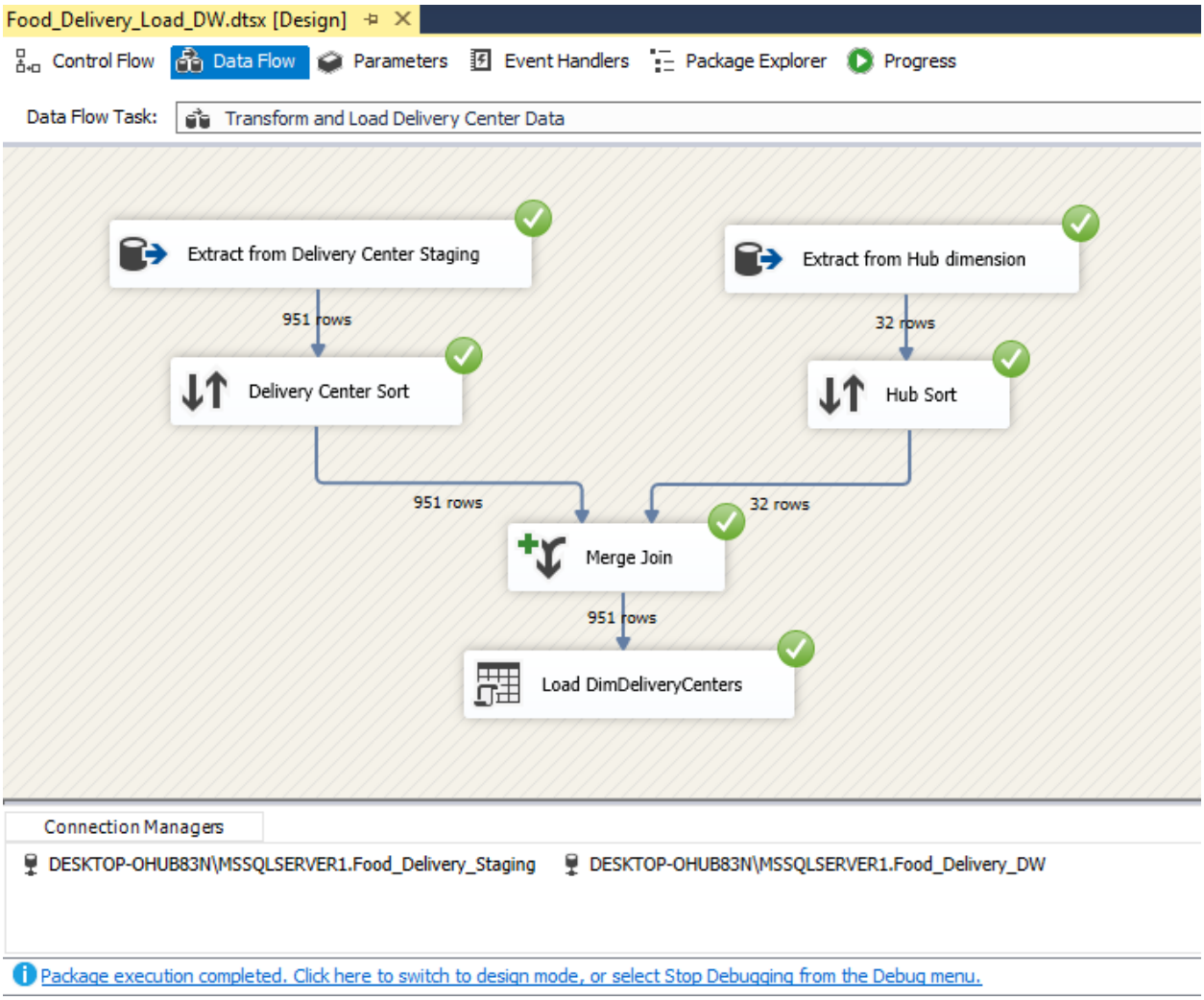
Steps:

1). Extract data from delivery centers staging table and the Hub dimension.

2). Delivery center details were sorted according to delivery id and hub details were sorted according to HubSK.

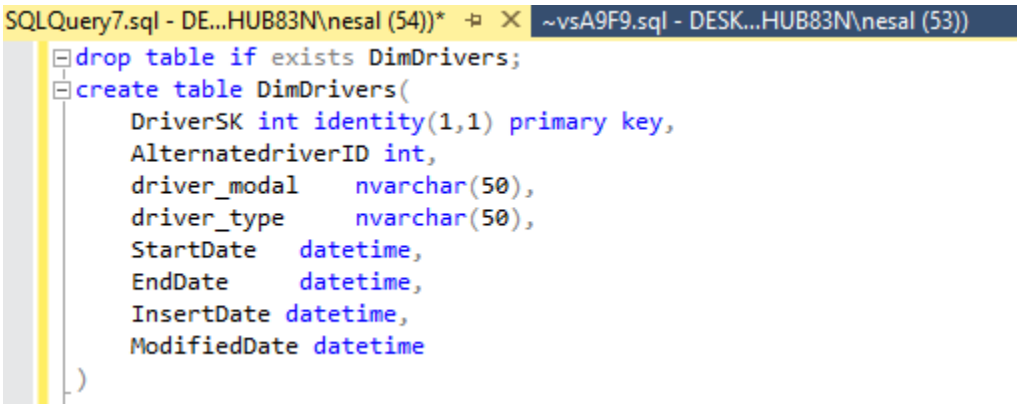
3). Two tables were joined using the Left Outer Join.

4). Finally, the data were loaded into DimDeliveryCenters after the specified mappings were done in the OLE DB Destination component.



5.3.3. Transform and Load Drivers data to Data Warehouse:

Driver's dimension (DimDrivers) was considered as the Slowly Changing dimension as it was assumed that Food Delivery Company is interested in the types and models of the drivers who are doing delivery tasks and would like to perform analysis based on the driver types and models. Driver model and phone number were set to be Changing Columns which means if one these column values were changed, they will simply get updated in the existing record itself. Driver type was set as Historical Column which means if it was updated in an existing row in DimDrivers, the existing row will be expired and a new row will be inserted, preserving the history of the location based on main columns mentioned earlier. All other non-specified columns were considered as Fixed Columns. Down below is the query to create the dimension table for drivers in mssql.



Steps to load and transform data from drivers staging table to drivers slowly changing dimension table is given below.

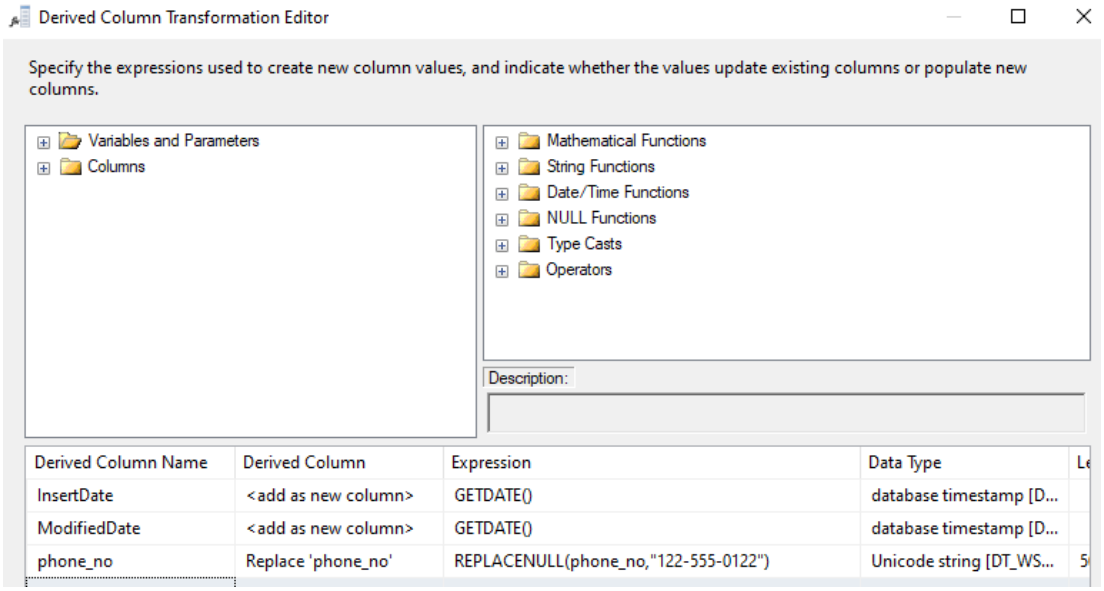
1) Extract data from drivers staging table (StgDrivers).
2) Sorted records by driver id.
3) Then three derived columns were specified, InsertDate and ModifiedDate columns to get the current timestamp and phone_no column to replace null value with a default phone number.
4) Dimension was specified as slowly changing dimension by selecting a change type for slowly changing dimension columns. StartDate and EndDate were set to system start time.

   OLE DB Command Component

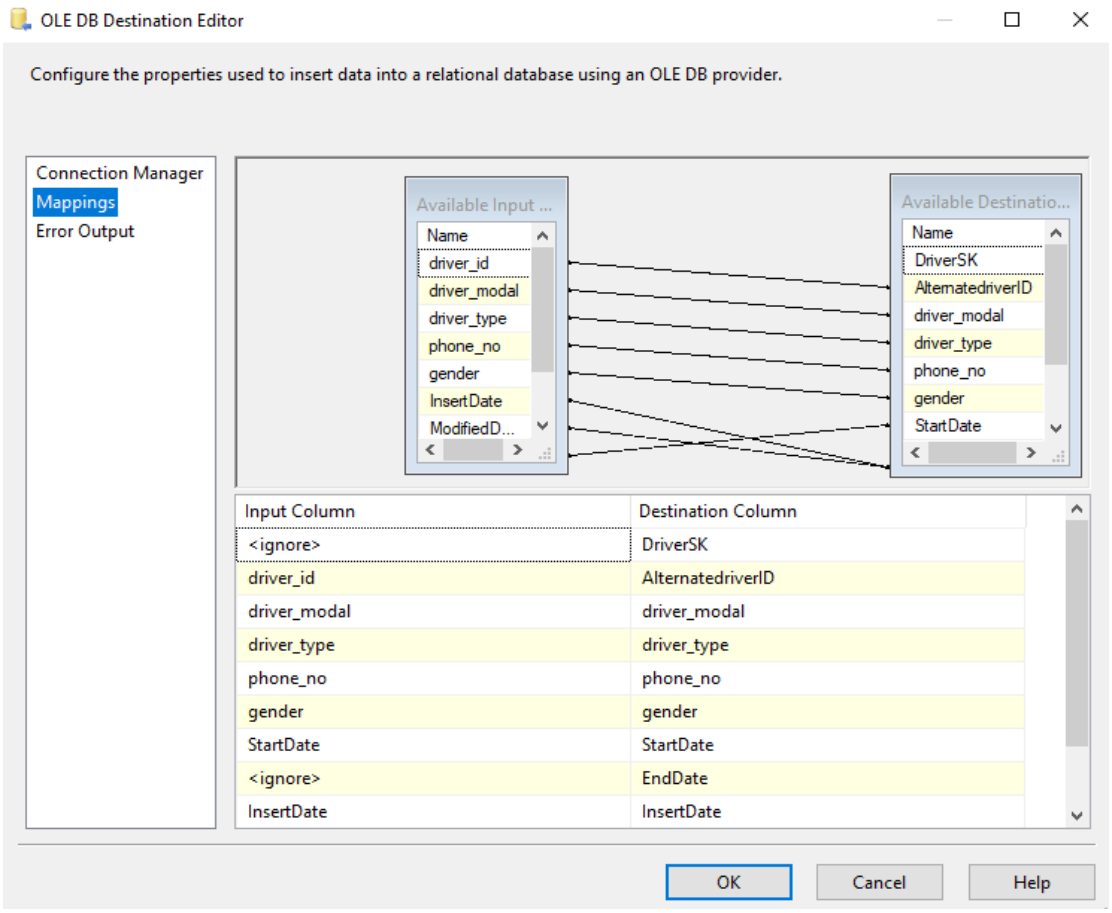
   UPDATE [dbo].[DimDrivers] SET [EndDate] = ? , ModifiedDate = GETDATE() WHERE [AlternatedriverID] = ? AND [EndDate] IS NULL

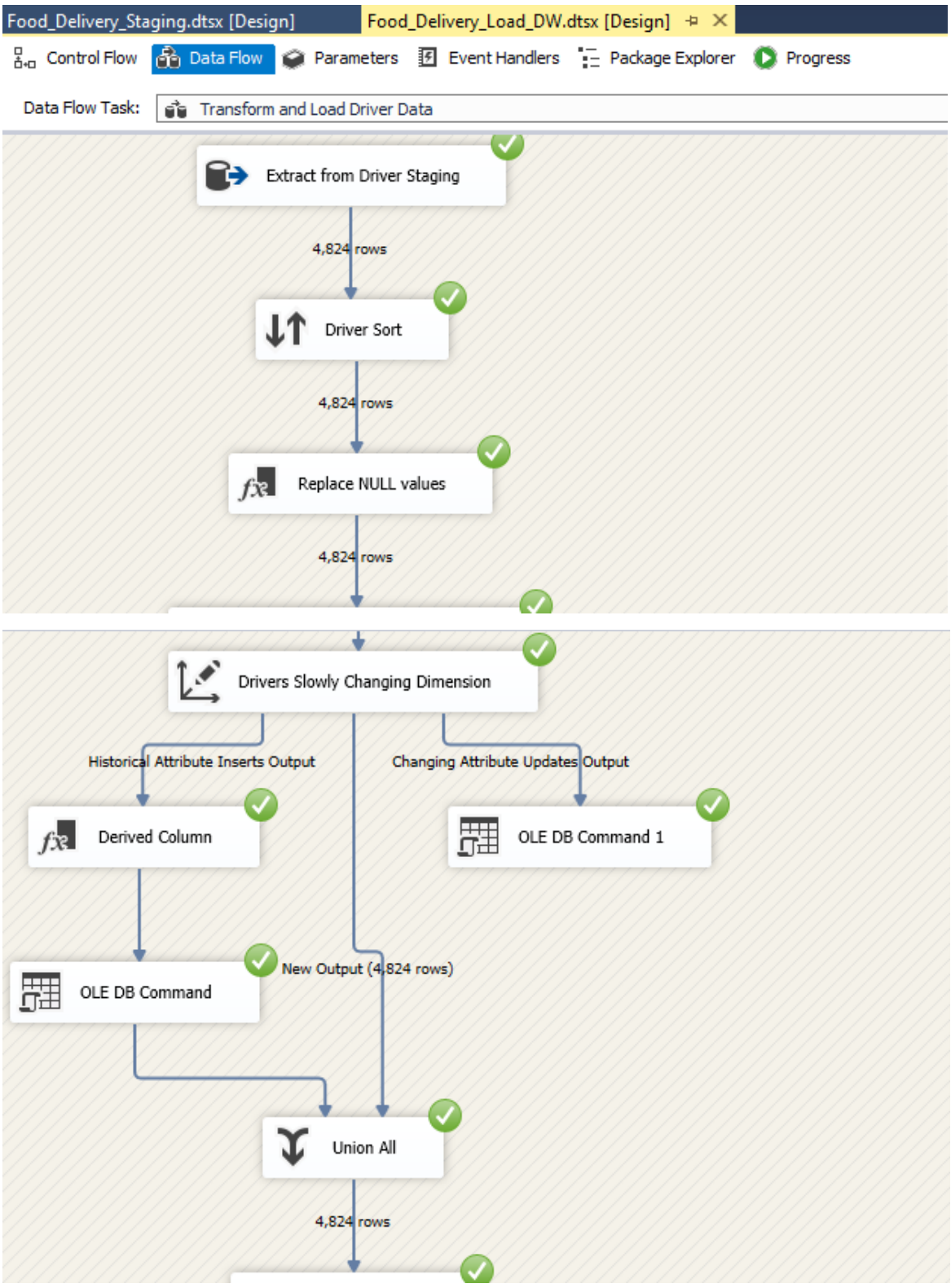
   OLE DB Command 1 Component

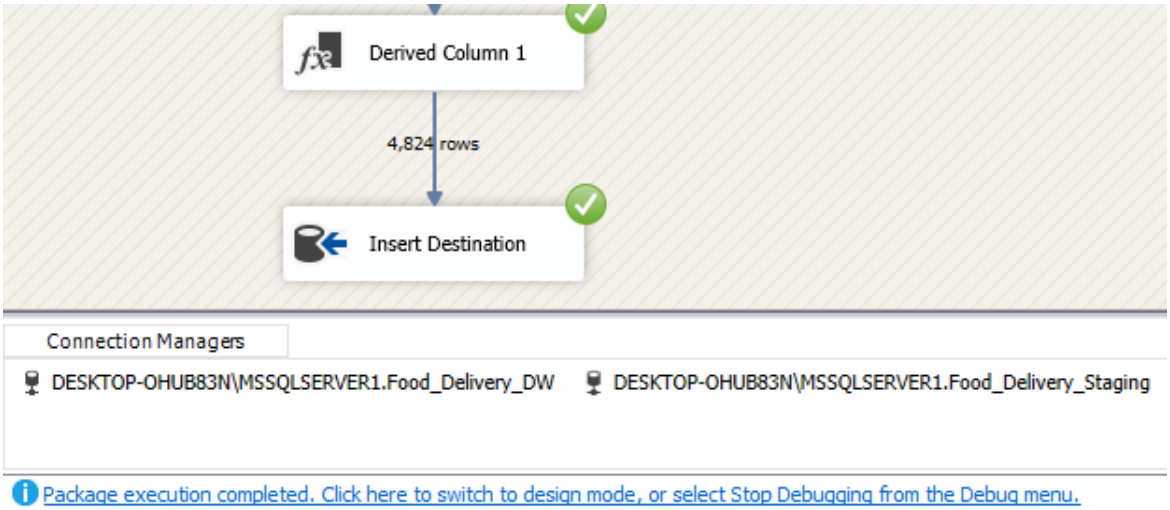
   UPDATE [dbo].[DimDrivers] SET [driver_modal] = ?,[phone_no] = ? , ModifiedDate = GETDATE() WHERE [AlternatedriverID] = ? AND [EndDate] IS NULL



5) Finally, the changes will get either inserted or updated in the DimDrivers dimension according to the specified mappings done in the OLE DB Destination component in the end.




Food_Delivery_Staging.dtsx [Design]
Food_Delivery_Load_DW.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Transform and Load Driver Data
Extract from Driver Staging
4,824 rows
Driver Sort
4,824 rows
Replace NULL values
4,824 rows
Drivers Slowly Changing Dimension
Historical Attribute Inserts Output
Changing Attribute Updates Output
Derived Column
OLE DB Command 1
OLE DB Command
New Output (4,824 rows)
Union All
4,824 rows



5.3.4. Transform and Load Deliveries data to Data Warehouse:

Since it has foreign key reference to Drivers dimension it has chosen next.

Stored procedure call: **exec dbo.UpdateDimDeliveries ?, ?, ?, ?**

Stored procedure to do insertions and updates: UpdateDimDeliveries

SQLQuery16.sql - D...HUB83N\nesal (58))* | SQLQuery14.sql - D...HUB83N\nesal (53)) | SQLQuery12.sql - D...HUB83N\nesal (59))

```
CREATE PROCEDURE dbo.UpdateDimDeliveries
@delivery_id int,
@DriverKey int,
@delivery_distance_meters int,
@delivery_status nvarchar(50)
AS
BEGIN
if not exists (select DeliverySK
from dbo.DimDeliveries
where AlternateDeliveryID = @delivery_id)
BEGIN
insert into dbo.DimDeliveries
(AlternateDeliveryID, DriverKey, delivery_distance_meters, delivery_status,
InsertDate, ModifiedDate)
values
(@delivery_id, @DriverKey, @delivery_distance_meters, @delivery_status,
GETDATE(), GETDATE())
END;
if exists (select DeliverySK
from dbo.DimDeliveries
where AlternateDeliveryID = @delivery_id)
BEGIN
update dbo.DimDeliveries
set DriverKey = @DriverKey, delivery_distance_meters = @delivery_distance_meters, delivery_status = @delivery_status,
ModifiedDate = GETDATE()
where AlternateDeliveryID = @delivery_id
END;
END;
```

100 %

Messages

```
Commands completed successfully.
```

100 %

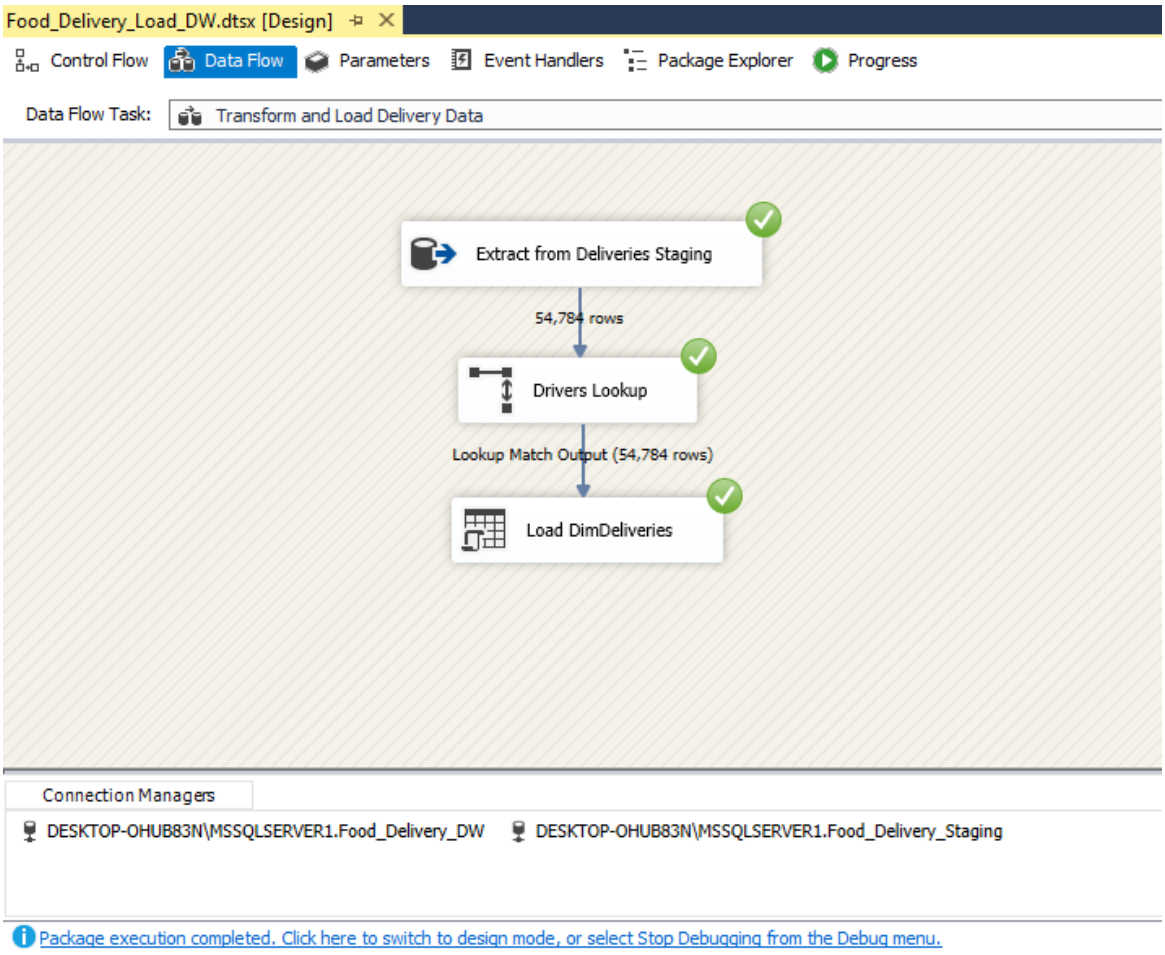
Query executed successfully. | DESKTOP-OHUB83N\MSSQLSERVER... | DESKTOP-OHUB83N\nesal...

SQL query to create the Deliveries Dimension: DimDeliveries

SQLQuery7.sql - DE...HUB83N\nesal (54))* ~vsA9F9.sql - DESK...HUB83N\nesal (53))

```
drop table if exists DimDeliveries;
create table DimDeliveries(
    DeliverySK int identity(1,1) primary key,
    AlternateDeliveryID int,
    DriverKey   int foreign key references DimDrivers(DriverSK),
    delivery_distance_meters    int,
    delivery_status     nvarchar(50),
    InsertDate datetime,
    ModifiedDate datetime
)
```

Since deliveries table has a foreign key from drivers table, it is also developed as the delivery centers dimension. In this process I have used lookup component instead of merge join to join to load and transfer data from deliveries staging to the deliveries dimension table. In lookup I have selected full cache mode, OLE DB Connection Manager type and ignore failure when there are rows with no matching entries.

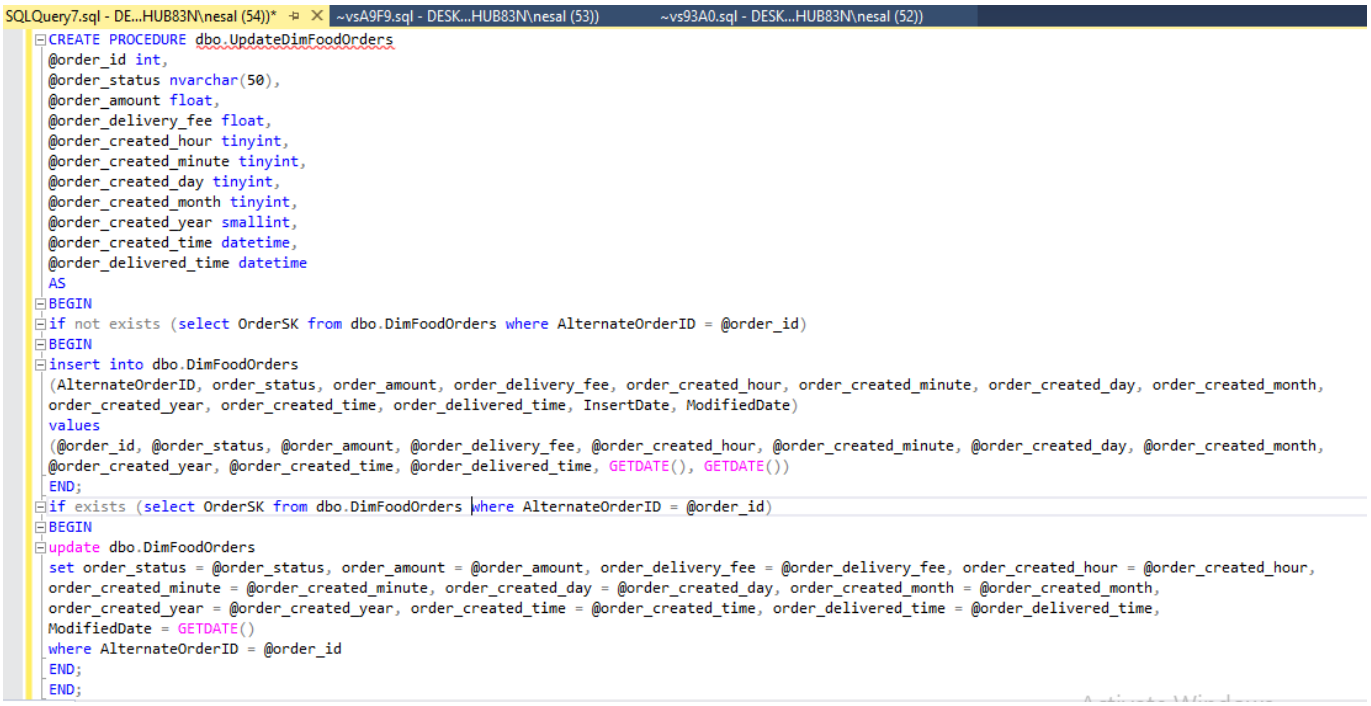


5.3.5. Transform and Load Food Orders data to Data Warehouse:

Since this was as same as Hubs table there were no transformations to be done, the extracted data is sent to an OLE DB command component to pass the data to a stored procedure in the data warehouse to do the insert and update accordingly.

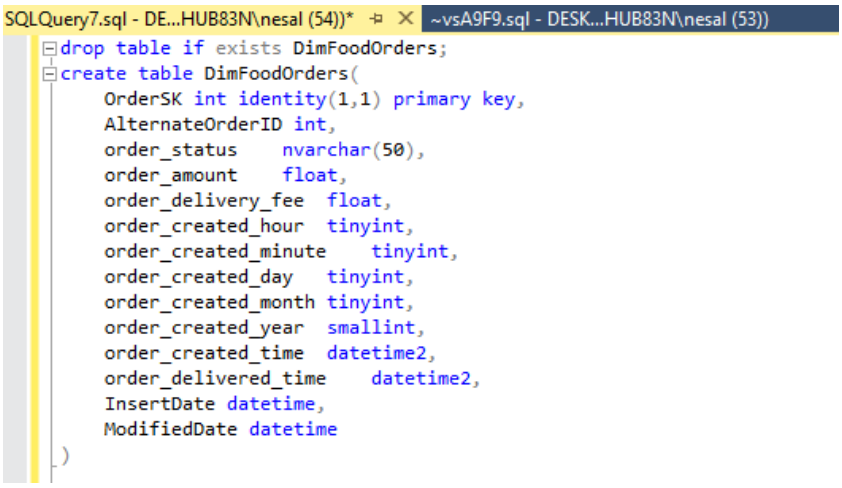
Stored procedure call: **exec dbo.UpdateDimFoodOrders ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?**

Stored procedure to do insertions and updates: UpdateDimFoorOrders

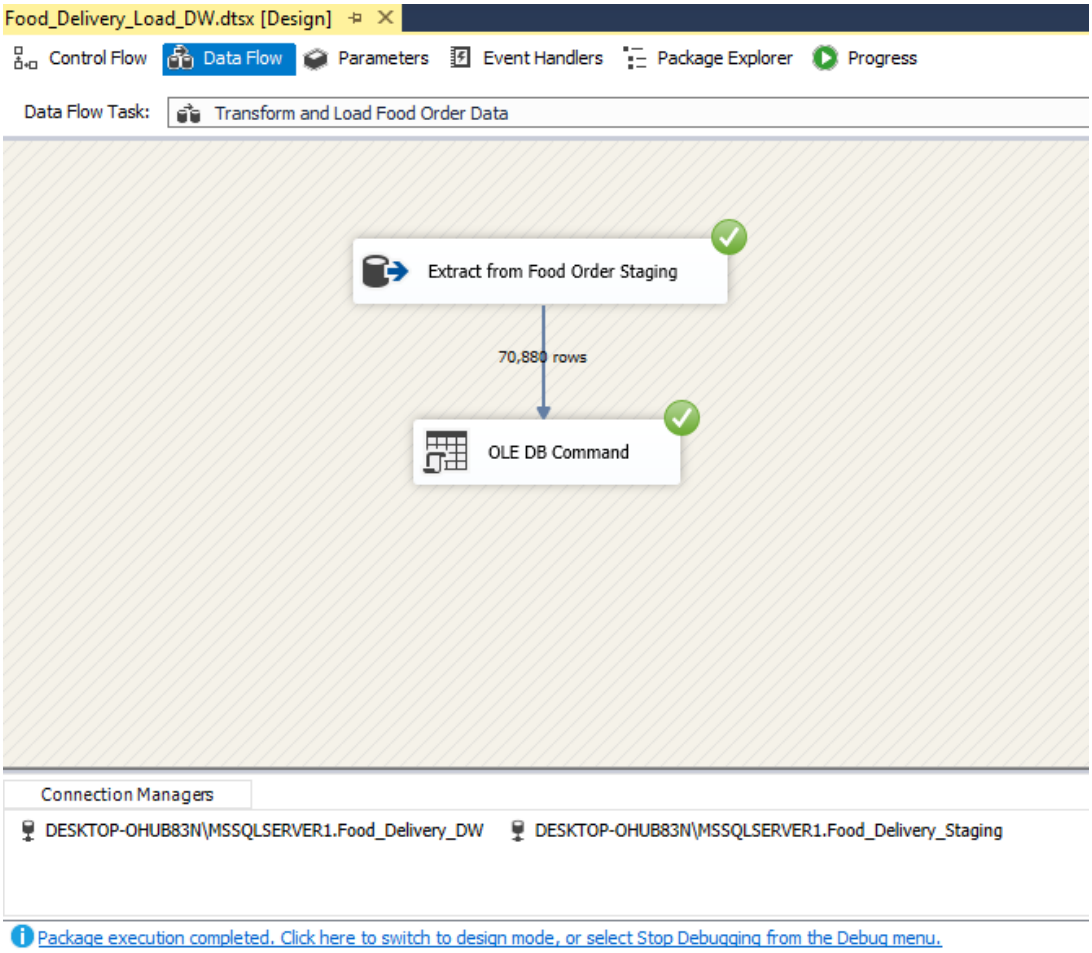


```
SQLQuery7.sql - DE...HUB83N\nesal (54))*   ~vsA9F9.sql - DESK...HUB83N\nesal (53))   ~vs93A0.sql - DESK...HUB83N\nesal (52))

CREATE PROCEDURE dbo.UpdateDimFoodOrders
@order_id int,
@order_status nvarchar(50),
@order_amount float,
@order_delivery_fee float,
@order_created_hour tinyint,
@order_created_minute tinyint,
@order_created_day tinyint,
@order_created_month tinyint,
@order_created_year smallint,
@order_created_time datetime,
@order_delivered_time datetime
AS
BEGIN
if not exists (select OrderSK from dbo.DimFoodOrders where AlternateOrderID = @order_id)
BEGIN
insert into dbo.DimFoodOrders
(AlternateOrderID, order_status, order_amount, order_delivery_fee, order_created_hour, order_created_minute, order_created_day, order_created_month,
order_created_year, order_created_time, order_delivered_time, InsertDate, ModifiedDate)
values
(@order_id, @order_status, @order_amount, @order_delivery_fee, @order_created_hour, @order_created_minute, @order_created_day, @order_created_month,
@order_created_year, @order_created_time, @order_delivered_time, GETDATE(), GETDATE())
END;
if exists (select OrderSK from dbo.DimFoodOrders where AlternateOrderID = @order_id)
BEGIN
update dbo.DimFoodOrders
set order_status = @order_status, order_amount = @order_amount, order_delivery_fee = @order_delivery_fee, order_created_hour = @order_created_hour,
order_created_minute = @order_created_minute, order_created_day = @order_created_day, order_created_month = @order_created_month,
order_created_year = @order_created_year, order_created_time = @order_created_time, order_delivered_time = @order_delivered_time,
ModifiedDate = GETDATE()
where AlternateOrderID = @order_id
END;
END;
```

SQL query to create the Food Orders Dimension: DimFoodOrders

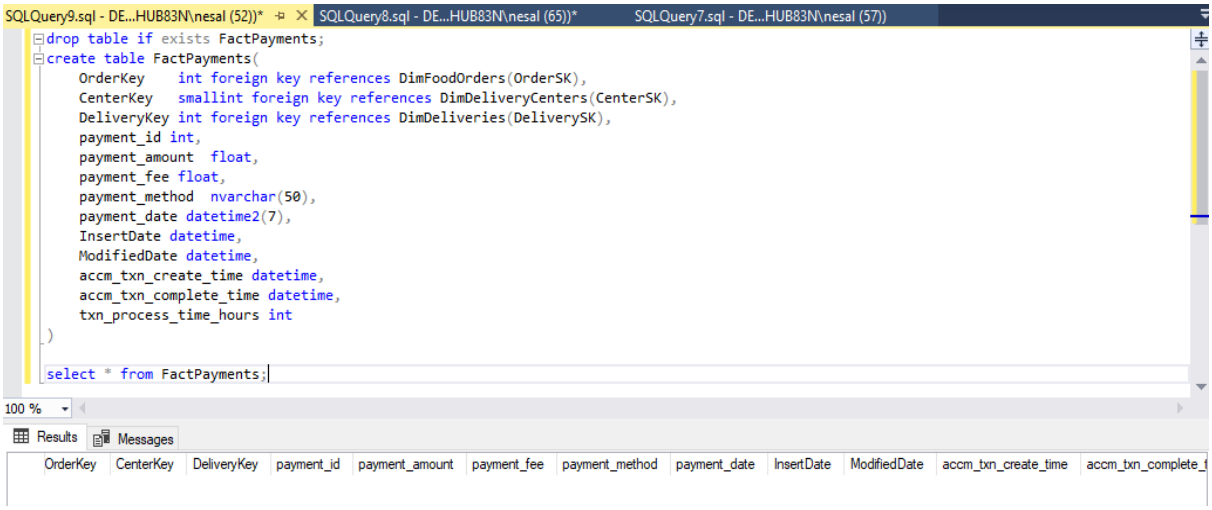


```
SQLQuery7.sql - DE...HUB83N\nesal (54))*   ~vsA9F9.sql - DESK...HUB83N\nesal (53))

drop table if exists DimFoodOrders;
create table DimFoodOrders(
    OrderSK int identity(1,1) primary key,
    AlternateOrderID int,
    order_status    nvarchar(50),
    order_amount    float,
    order_delivery_fee  float,
    order_created_hour  tinyint,
    order_created_minute    tinyint,
    order_created_day   tinyint,
    order_created_month tinyint,
    order_created_year  smallint,
    order_created_time  datetime2,
    order_delivered_time    datetime2,
    InsertDate datetime,
    ModifiedDate datetime
)
```



5.3.6. Transform and Load Payments data to Data Warehouse:

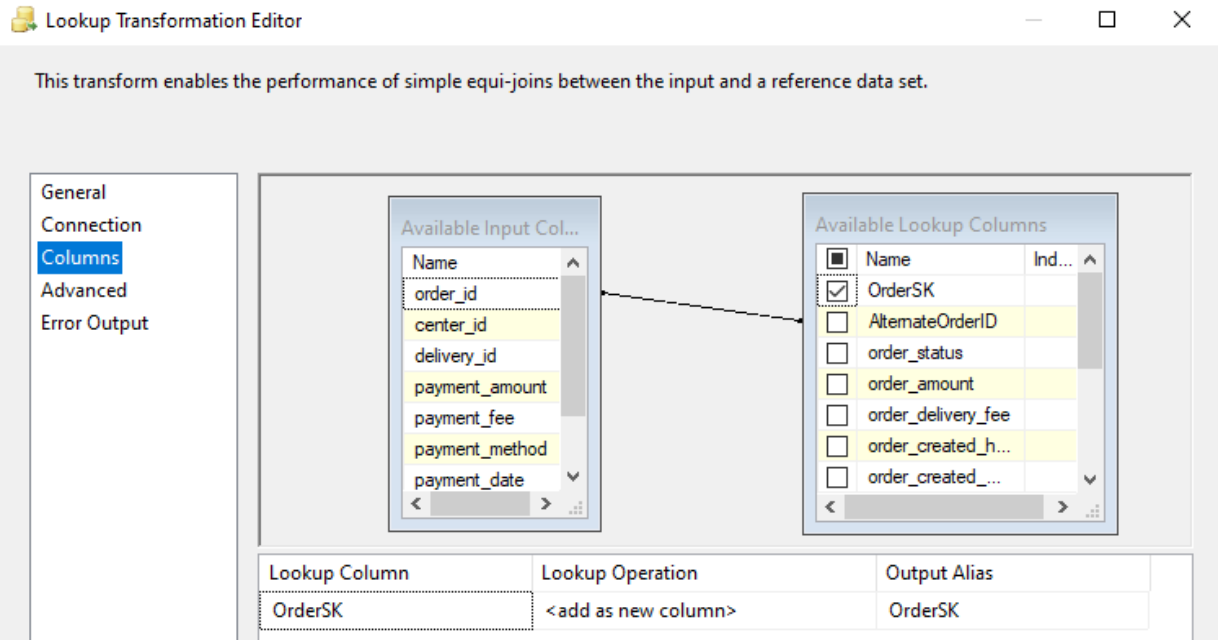
SQL query to create the Payments Fact Table: Fact Payments



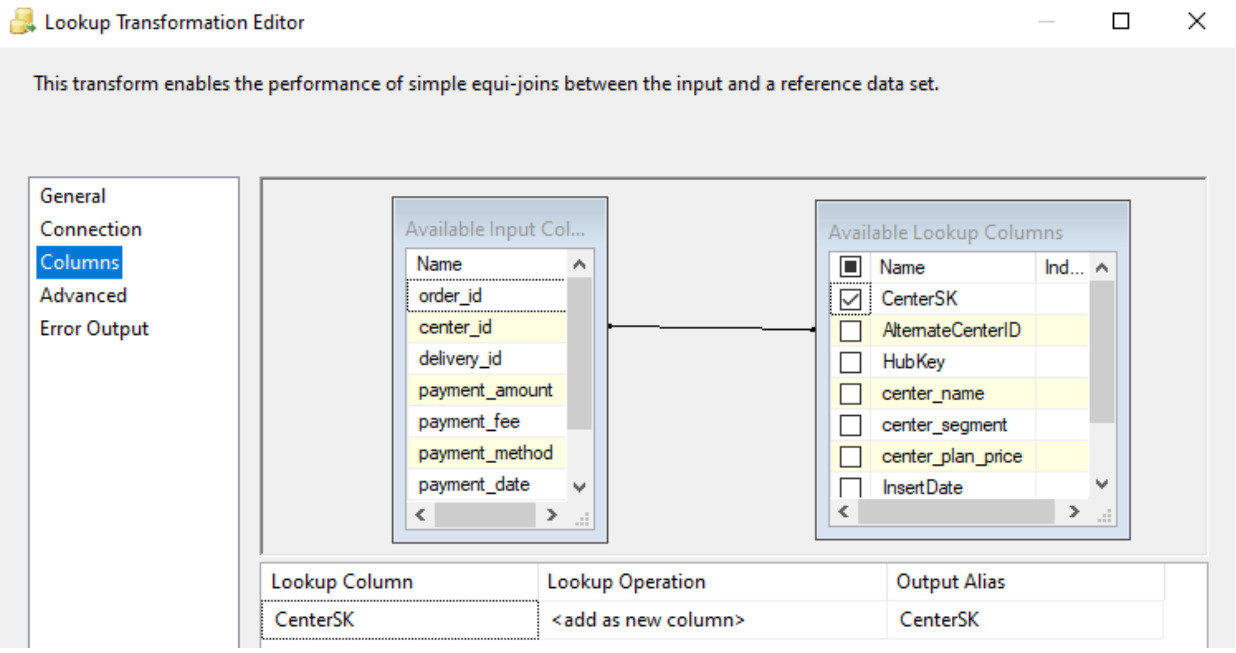
After loading to all the dimensions, lastly data was loaded to the fact table. The below steps were followed:

1). Firstly, the payment details were extracted from Payment staging table.

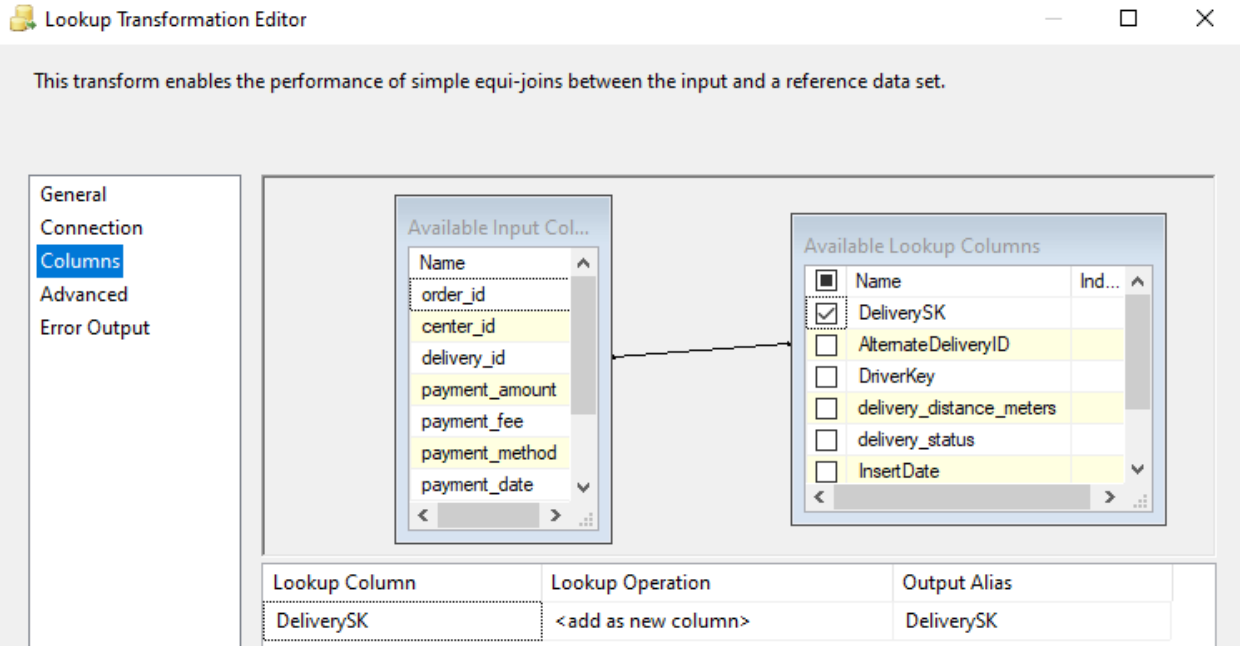
2). Since the fact table refers a lot of dimension tables, lookups were used to get the corresponding surrogate keys to establish table references. 1st Lookup is done to retrieve the OrderSK from DimOrder by comparing the business key of an Order which is order_id.



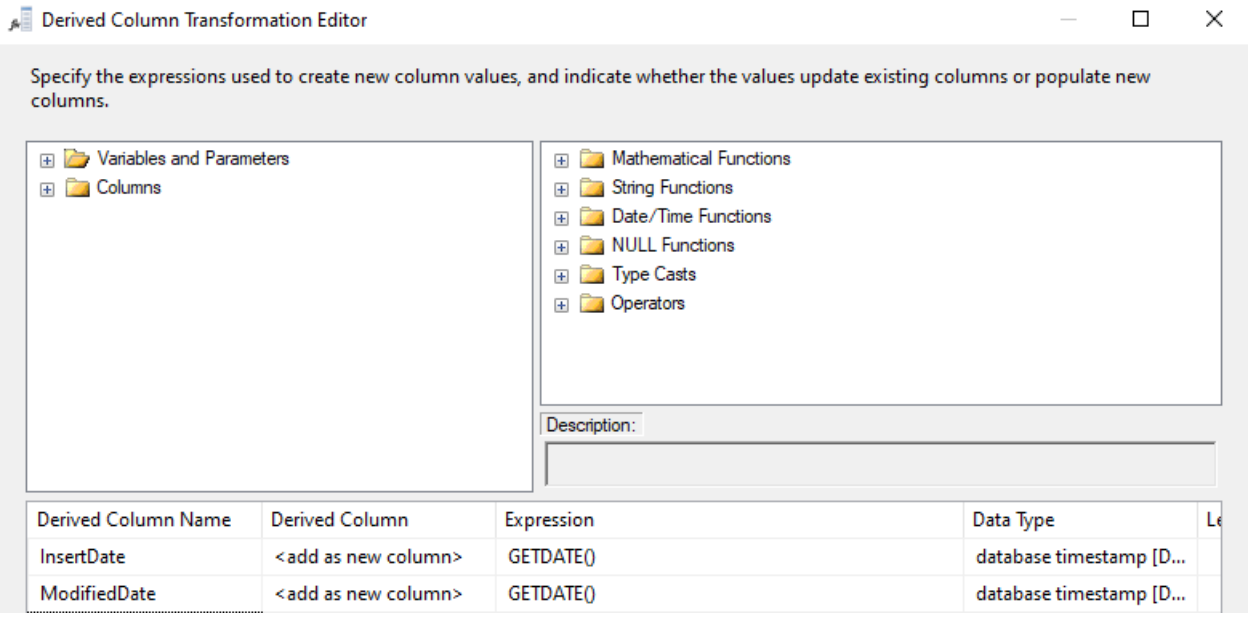
3). 2nd Lookup is done to retrieve the CenterSK from DimDeliveryCenters by comparing the business key of a delivery center which is center_id.



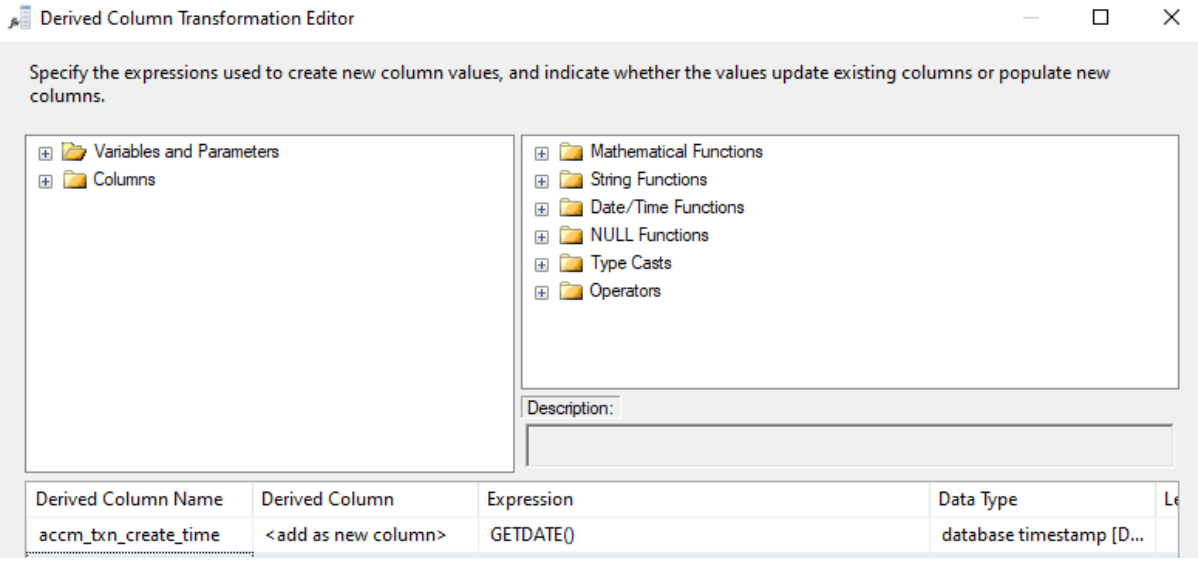
4). 3rd Lookup is done to retrieve the DeliverySK from DimDeliveries by comparing the business key of a delivery which is delivery_id.



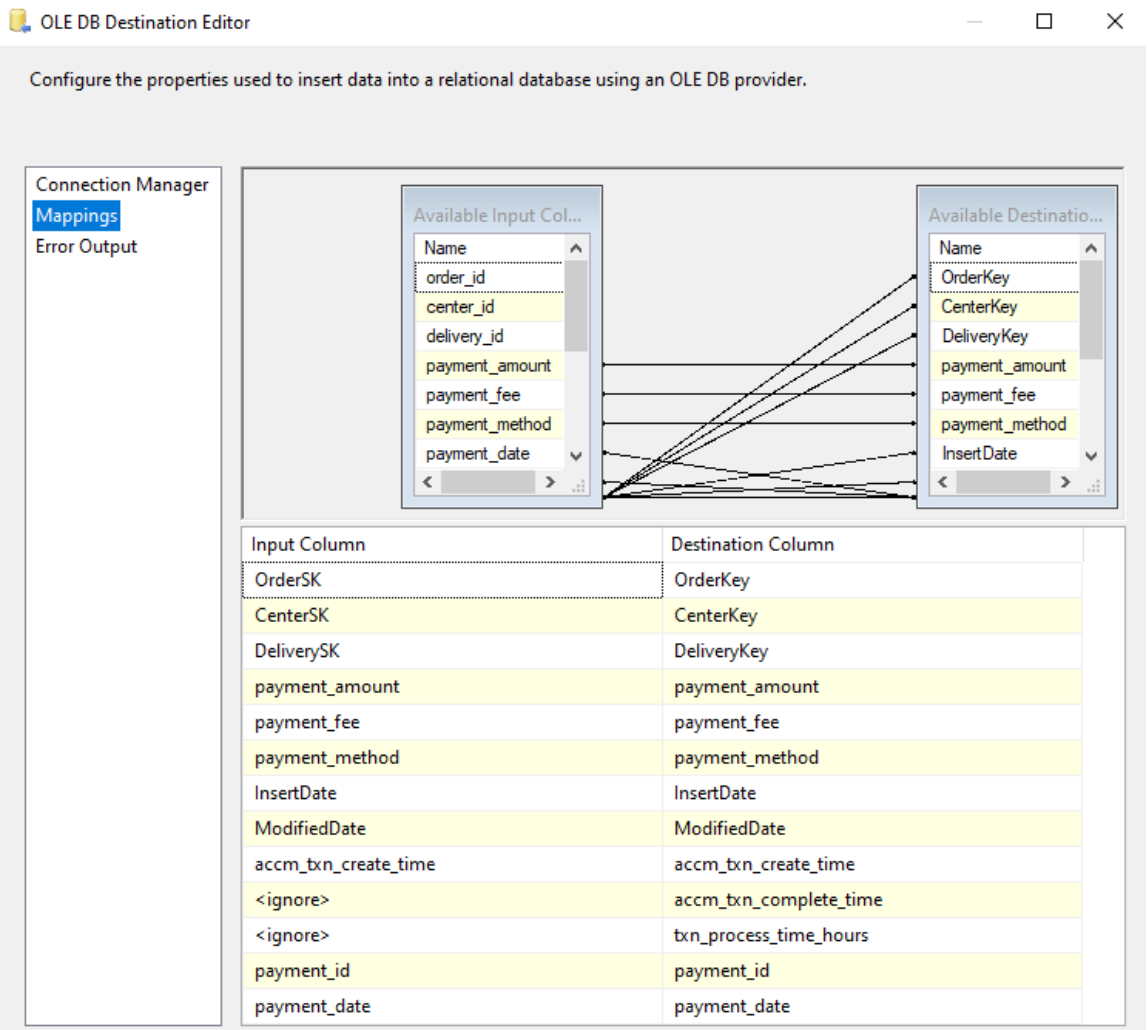
5). Then a derived column is used to assign insert date and modified date the current date.

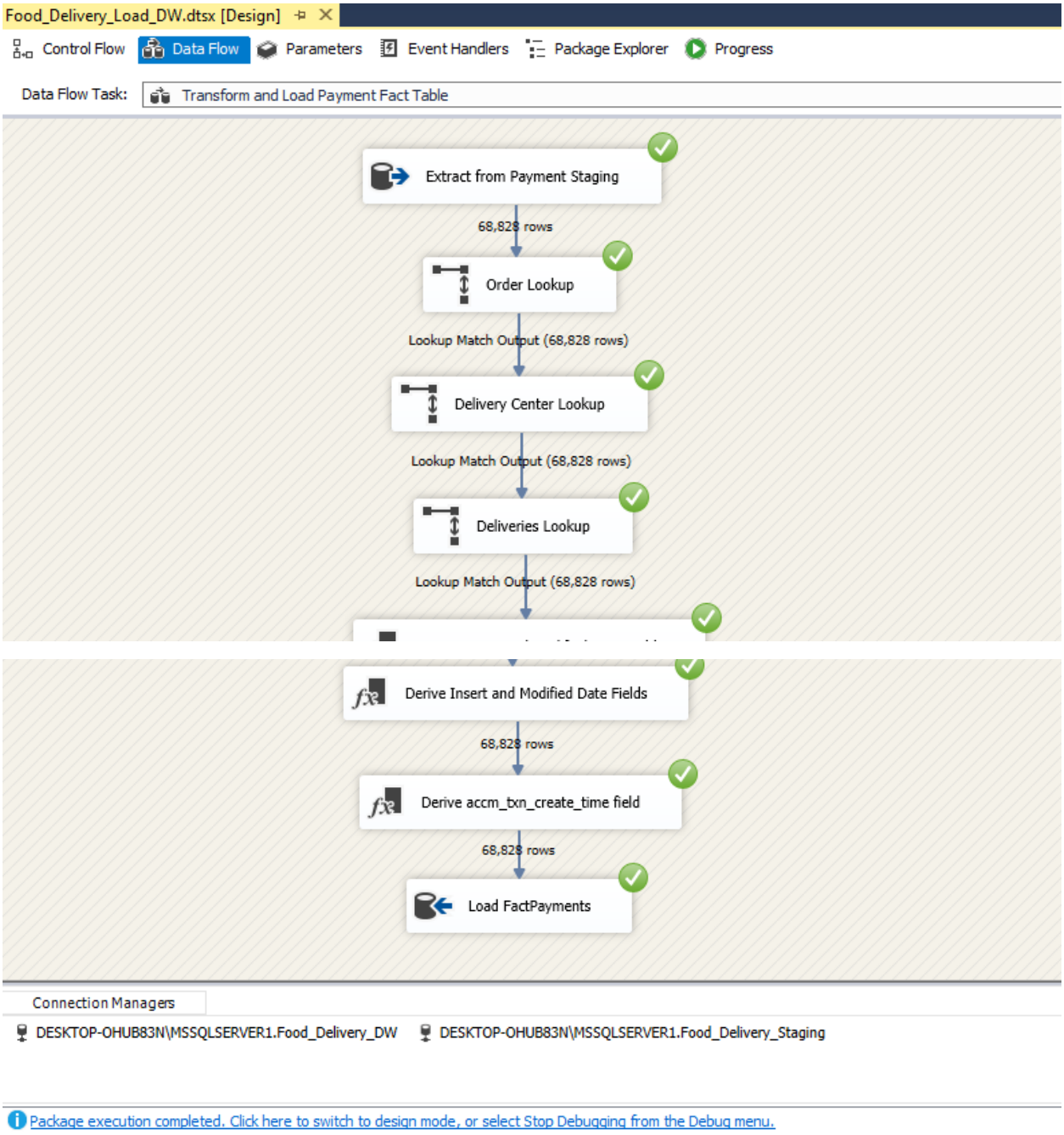


6). Another derived column is used to assign accm_txn_create_time column the event time as same as the above.



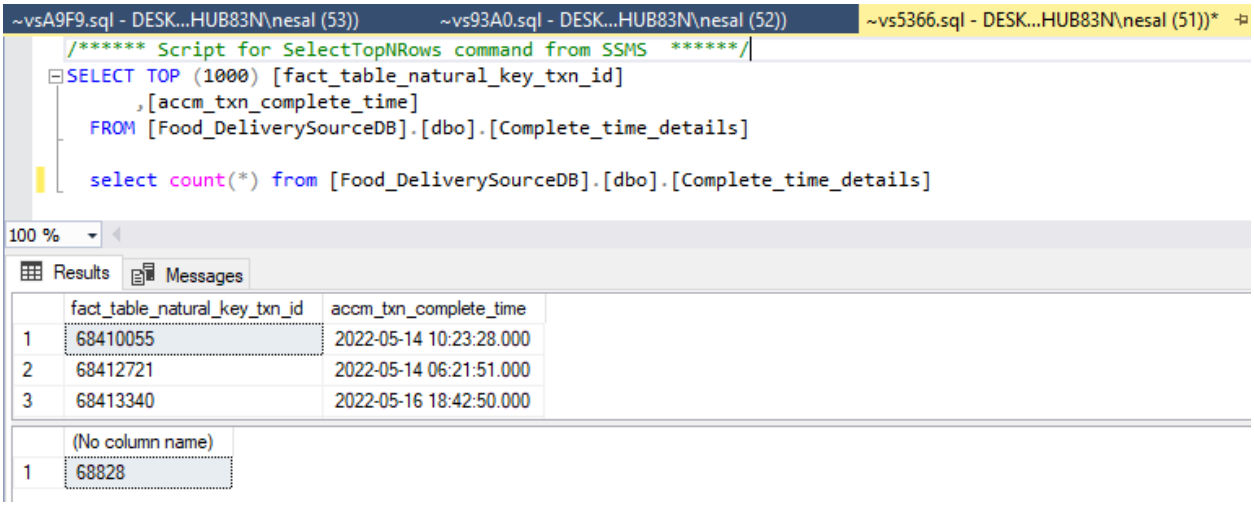
7). Finally, the data was loaded into fact payment by matching columns.




Food_Delivery_Load_DW.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Transform and Load Payment Fact Table
Extract from Payment Staging
68,828 rows
Order Lookup
Lookup Match Output (68,828 rows)
Delivery Center Lookup
Lookup Match Output (68,828 rows)
Deliveries Lookup
Lookup Match Output (68,828 rows)
Derive Insert and Modified Date Fields
68,828 rows
Derive accm_txn_create_time field
68,828 rows
Load FactPayments
Connection Managers
DESKTOP-OHUB83N\MSSQLSERVER1.Food_Delivery_DW
DESKTOP-OHUB83N\MSSQLSERVER1.Food_Delivery_Staging
Package execution completed. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

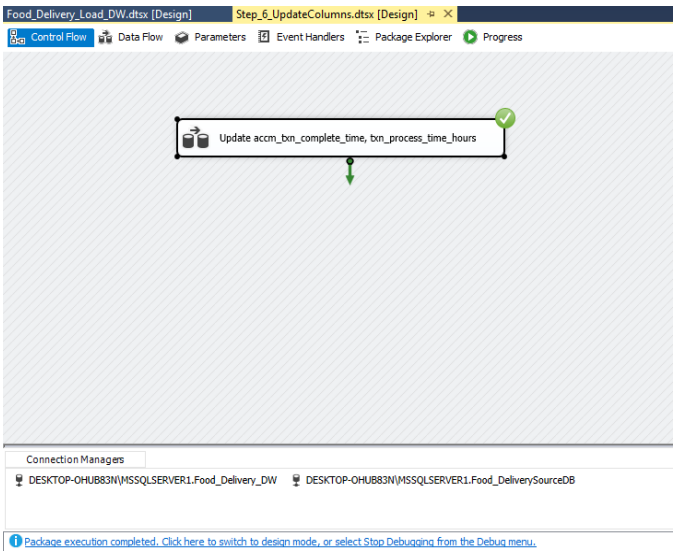
I have added three additional columns to the fact table named accm_txn_create_time, accm_txn_complete_time and txn_process_time_hours to calculate the time (number of hours) to complete a payment. I assigned accm_txn_create_time to be equal to the current system date when loading the fact table data. The other two columns were null.

Then I prepared a separate sql table name Complete_time_details in the source database as shown below to fill the other two columns (I have exported the sql table as a flat file and added into data sources folder):



Then I created a separate ETL ssis package named “Step_6_UpdateColumns” which reads data from this file and update the corresponding accm_txn_complete_time in the DW fact table and update txn_process_time_hours by taking the hours difference between accm_txn_create_time and accm_txn_complete_time.

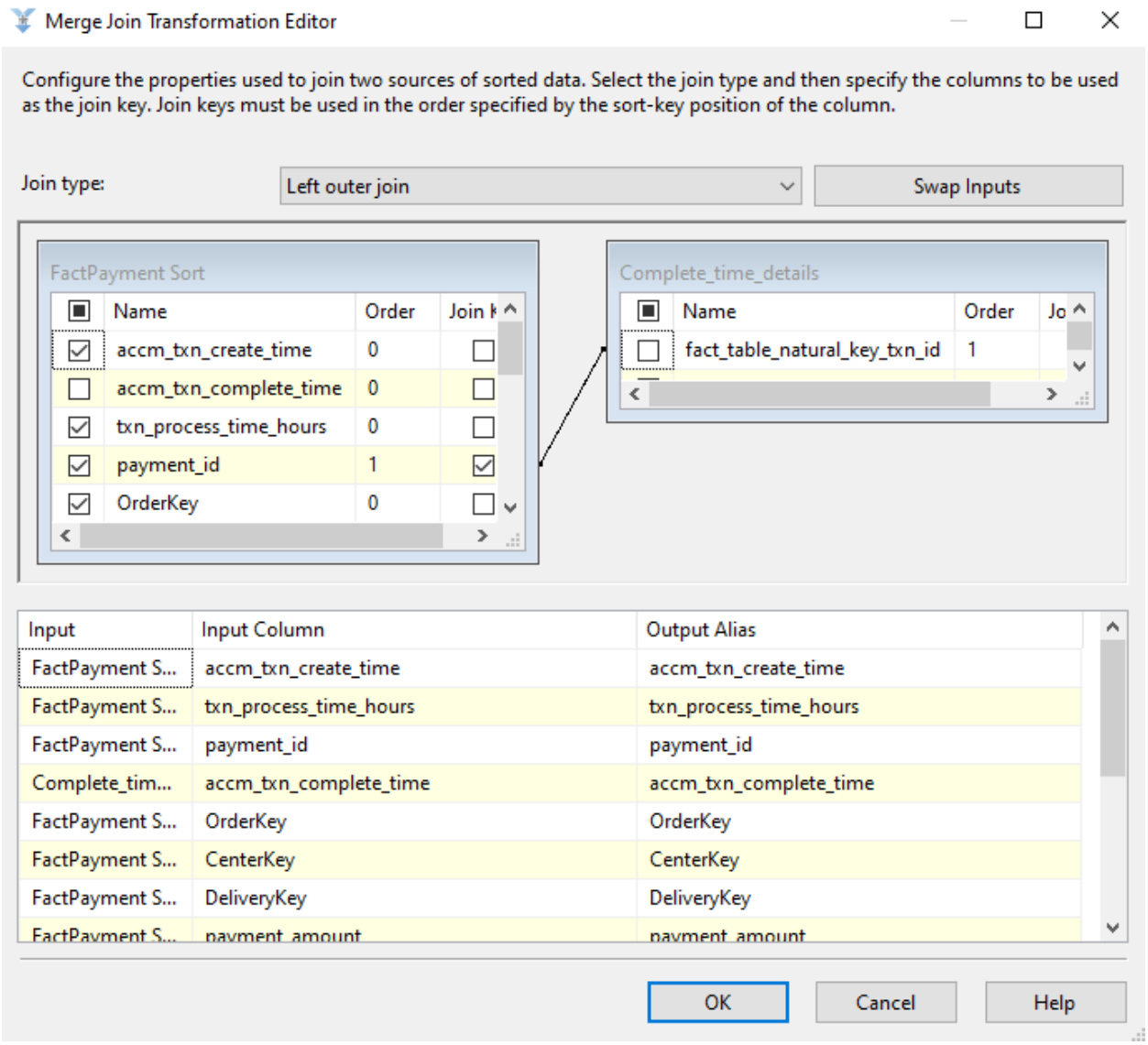
1). First, I added a data flow task named “Update accm_txn_create_time, accm_txn_complete_time” in the control flow



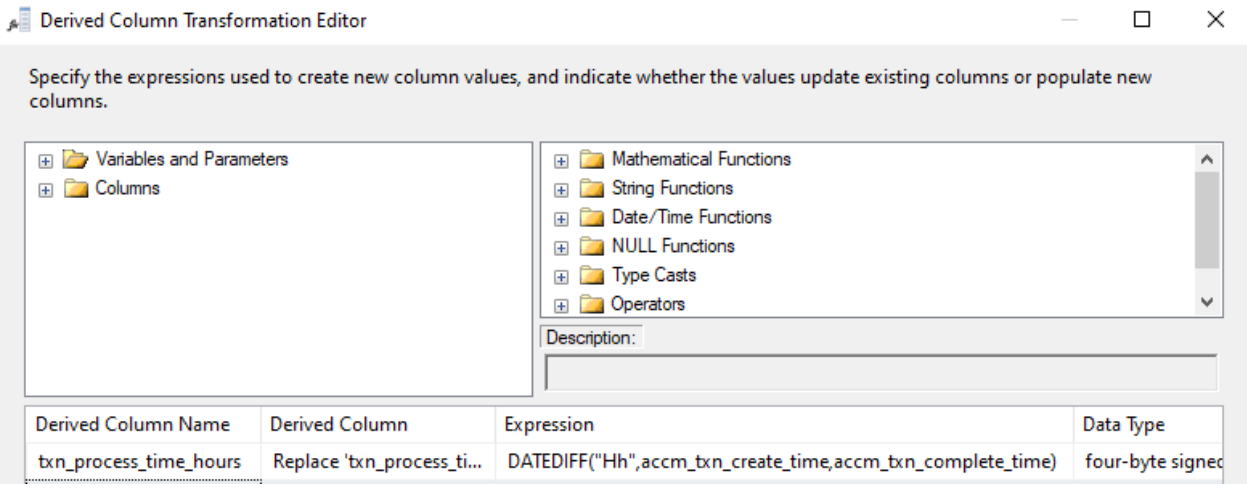
2). In the data flow, First I extracted data from Payments Fact table in the data warehouse and Complete time details from the source database.

3). The data were sorted according to payment id(unique) in the fact table and txn id(primary key) in Complete time details.

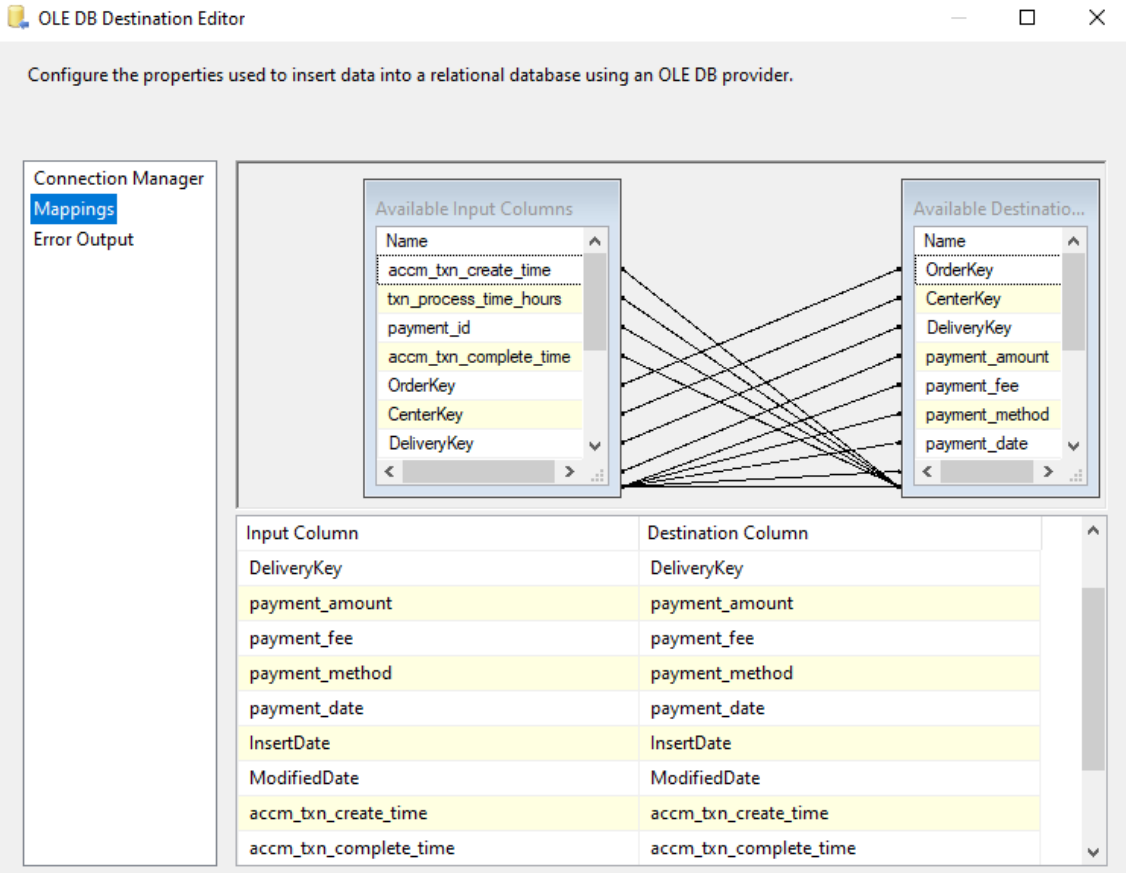
4). Two tables were joined using the merge join component as shown below:



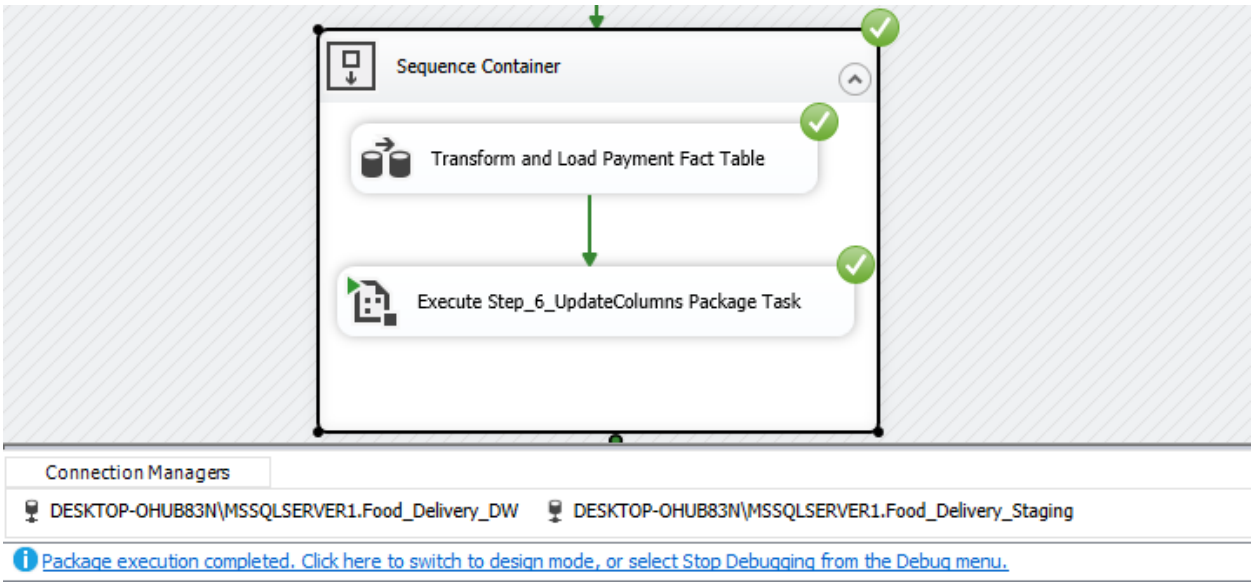
5). Then I added a derived column to calculate txn_process_time_hours as shown below:

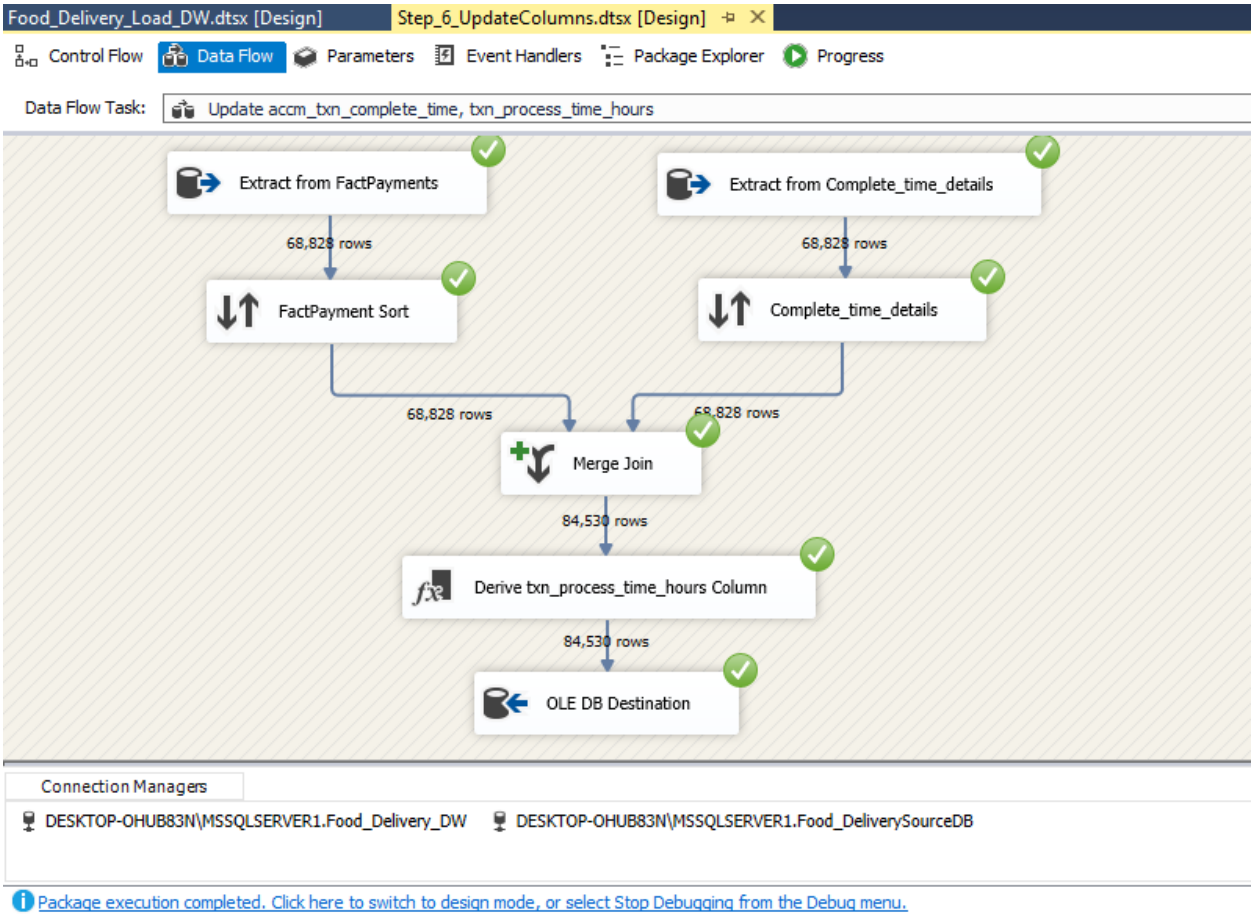


6). At the end of data flow a destination component was added and mapped the columns as shown below to update two null columns:

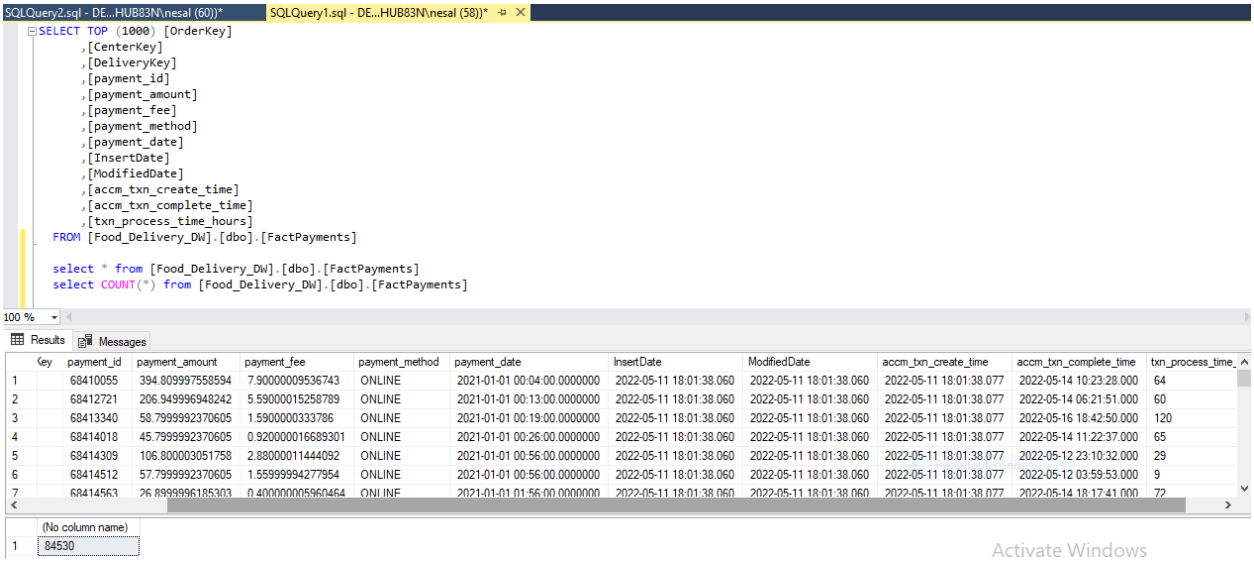


7). Finally, I added an execute task at the end of the data warehouse SSIS package below the fact table data flow, inside a container to be execute after loading the fact table and executed it:





The final fact table looks as shown below in the data warehouse:



```
SELECT TOP (1000) [OrderKey]
      ,[CenterKey]
      ,[DeliveryKey]
      ,[payment_id]
      ,[payment_amount]
      ,[payment_fee]
      ,[payment_method]
      ,[payment_date]
      ,[InsertDate]
      ,[ModifiedDate]
      ,[accm_txn_create_time]
      ,[accm_txn_complete_time]
      ,[txn_process_time_hours]
  FROM [Food_Delivery_DW].[dbo].[FactPayments]

  select * from [Food_Delivery_DW].[dbo].[FactPayments]
  select COUNT(*) from [Food_Delivery_DW].[dbo].[FactPayments]
```

	payment_id	payment_amount	payment_fee	payment_method	payment_date	InsertDate	ModifiedDate	accm_txn_create_time	accm_txn_complete_time	txn_process_time_
1	68410055	394.809997558594	7.90000009536743	ONLINE	2021-01-01 00:04:00.0000000	2022-05-11 18:01:38.060	2022-05-11 18:01:38.060	2022-05-11 18:01:38.077	2022-05-14 10:23:28.000	64
2	68412721	206.949996948242	5.59000015258789	ONLINE	2021-01-01 00:13:00.0000000	2022-05-11 18:01:38.060	2022-05-11 18:01:38.060	2022-05-11 18:01:38.077	2022-05-14 06:21:51.000	60
3	68413340	58.7999992370605	1.5900000333786	ONLINE	2021-01-01 00:19:00.0000000	2022-05-11 18:01:38.060	2022-05-11 18:01:38.060	2022-05-11 18:01:38.077	2022-05-16 18:42:50.000	120
4	68414018	45.7999992370605	0.920000016689301	ONLINE	2021-01-01 00:26:00.0000000	2022-05-11 18:01:38.060	2022-05-11 18:01:38.060	2022-05-11 18:01:38.077	2022-05-14 11:22:37.000	65
5	68414309	106.800003051758	2.88000011444092	ONLINE	2021-01-01 00:56:00.0000000	2022-05-11 18:01:38.060	2022-05-11 18:01:38.060	2022-05-11 18:01:38.077	2022-05-12 23:10:32.000	29
6	68414512	57.7999992370605	1.55999994277954	ONLINE	2021-01-01 00:56:00.0000000	2022-05-11 18:01:38.060	2022-05-11 18:01:38.060	2022-05-11 18:01:38.077	2022-05-12 03:59:53.000	9
7	68414563	26.8999996185303	0.400000005960464	ONLINE	2021-01-01 01:56:00.0000000	2022-05-11 18:01:38.060	2022-05-11 18:01:38.060	2022-05-11 18:01:38.077	2022-05-14 18:17:41.000	72

	(No column name)
1	84530