Sri Lanka Institute of Information Technology (SLIIT)

IT19957180

P.M.D.C.B Wijerathna

Y3.S2.04.WE

IT3021

DWBI Assignment1- Report

Contents

Table of Figures

1. Data set selection

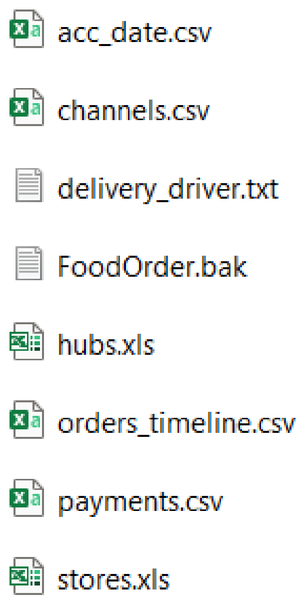
Data Set: Delivery Center: Food & Goods orders in Brazil

Site: Kaggle

Source Link: https://www.kaggle.com/datasets/nosbielcs/brazilian-delivery-center

The dataset for assignment 01 was created using a real-world business entity called "Delivery Center." Because it is a real business, sensitive information such as the personal information of their drivers is not disclosed in detail, and just a few factors have been revealed to the public. This data set is about Delivery Center details. It consists of Food & Goods orders in Brazil. The data collection is made up of orders and deliveries processed by the Delivery Center between 2019 and 2022.

There are four csv, two excel, one txt and one data base backup for Order details files in the data set.



acc_date.csv
channels.csv
delivery_driver.txt
FoodOrder.bak
hubs.xls
orders_timeline.csv
payments.csv
stores.xls

Channels: This dataset includes details on the sales channels (marketplaces) where our retailers' goods and food are sold.

Deliveries: This dataset contains data on deliveries performed by our delivery partners.

Drivers: This dataset has information about the partner deliverers. They stay in our hubs and every time an order is processed, they deliver it to consumers' homes.

Hubs: This dataset has information about Delivery Center hubs. Understand that Hubs are the distribution centers for orders and that's where deliveries come from.

Orders: This dataset has information about sales processed through the Delivery Center platform.

Payments: This dataset has information about payments made to the Delivery Center.

Orders: This dataset has information about the tenants. They use the Delivery Center Platform to sell their items (good and/or food) on marketplaces.

Orders Timeline: This dataset contains all the Order dates and delivery dates according to the order ID.

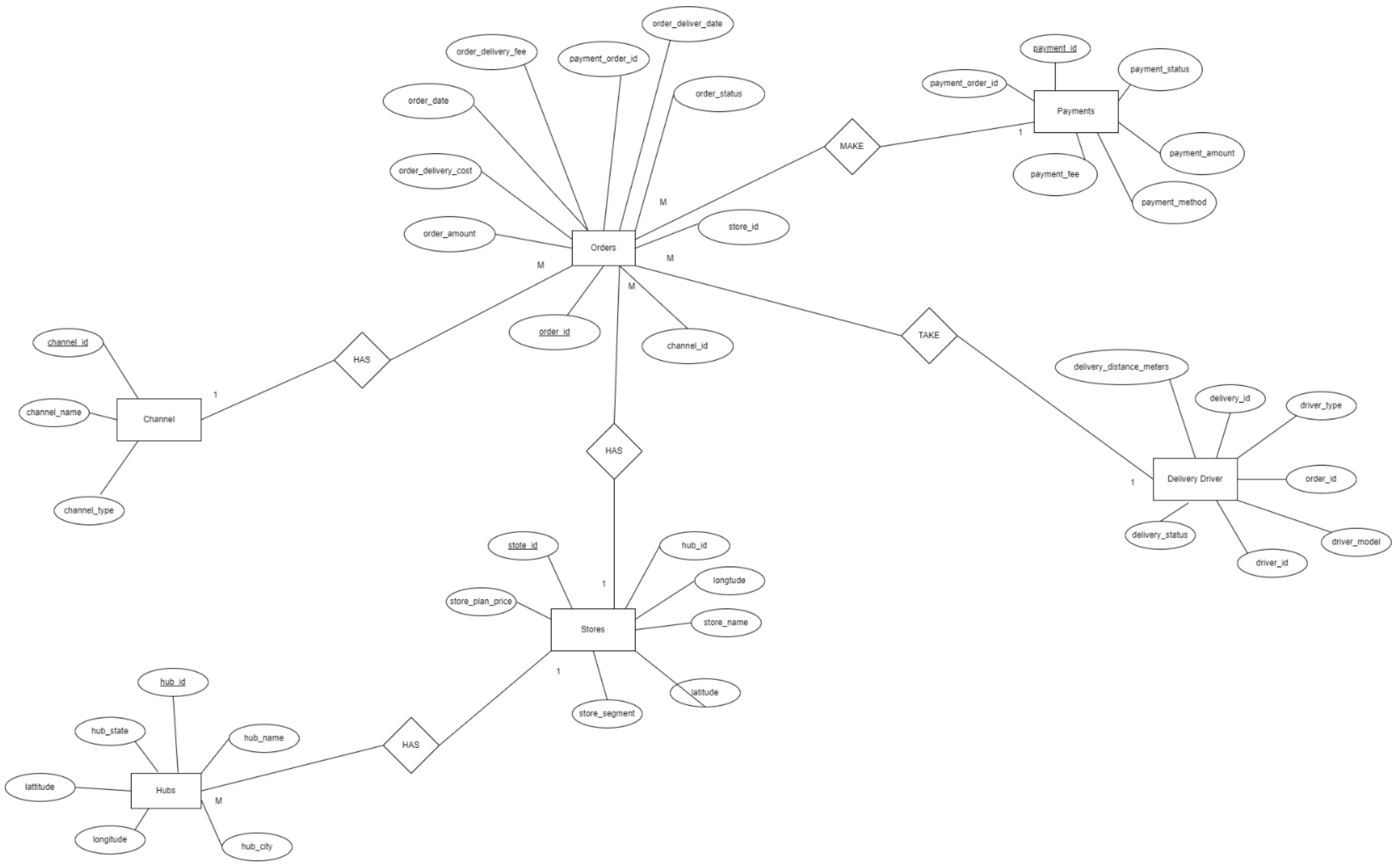


Figure 1 : High Level ER Diagram of Dataset

2. Preparation of Data Source

2.1. Data Quality and Preparation

There are almost 400 000 thousand records in the primary dataset, all with 59 variables. However, when evaluating the dataset's overall data quality, most of the variables had null values. As a result, null values were eliminated from the original dataset and separated into several sources. In addition, further data was contributed to the dataset with the purpose of improving their overall quality.

2.2. Data Source Details

Table Name	Column Name	Data Type	Link Table	Link Column
Channel	**Channel_id**	**Int**		
	Channel_name	**Varchar (50)**		
	Channel_type	**Varchar(50)**		
DeliveryDriver	**Driver_id**	**Int**		
	Driver_order_id	**Int**	**order**	**Delivery_order_id**
	Delivery_distance_meters	**Varchar(50)**		
	delivery_status	**Varchar(50)**		
	driver_modal	**Varchar(50)**		
	driver_type	**Varchar(50)**		
	Email Address	**Varchar(50)**		
	FirstName LastName	**Varchar(50)**		
	Phone Number	**Varchar(50)**		
	Phone Number	**Varchar(50)**		
Hubs	**hub_id**	**Int**	**store**	**Hub_id**
	hub_name	**Varchar(50)**		
	hub_city	**Varchar(50)**		
	hub_state	**Varchar(50)**		

OrderDates	**Order_id** **Order_date** **Deliver_date** **Order_time_hour** **Order_time_minute** **Deliver_time_hour** **Deliver_time_minute**	**Int** **Datetime** **Datetime** **Int** **Int** **Int** **int**	**order**	**Order_id**
Orders	**Order_deliver_id** **Order_id** **Store_id** **Channel_id** **Payment_id** **Order_status** **Order_amount** **Order_delivery_fee** **Order_delivery_cost**	**Int** **Int** **Int** **Int** **Int** **Varchar(50)** **Money** **Money** **Money**	**DeliveryDriver** **Store** **Channel** **Payment**	**Delivery_order_id** **Stoe_id** **Channel_id** **Payment_order_id**
Payments	**Payment_id** **Payment_method** **Payment_order_id** **Payment_fee** **Payment_amount** **Payment_status**	**Int** **Varchar(50)** **Int** **Money** **Money** **Varchar(50)**	**order**	**Order_id**
Stores	**Store_id** **Hub_id** **Store_name** **Store_segment** **Store_plan_price**	**Int** **Int** **Varchar(50)** **Varchar(50)** **money**	**hub**	**Hub_id**

3. Solution Architecture

The diagram below shows the overall architecture of the Delivery Center: Food & Goods orders in Brazil Datawarehouse and Business Intelligence Solution that is being implemented. Data Sources, ETL (Extract, Transform, Load), Storage Layer Components, and Data Consumption are the four components that make the architecture.

- **Data Sources**, it comprises of structured data in the format of text, SQL and excel. The database files are stored in SQL Server DB, and other formats are stored in the local folder.
- **ETL**, it is performed at two instances, first instance when extracting data from the sources and loading it to the Staging Layer and in second instance when performing extraction, and transformation on Staging Layer to load data into Datawarehouse Layer.
- **Storage Layer**, there are two main layers of storage, intermediate and core. Staging tables are implemented in intermediate layer, while the Datawarehouse and OLAP Server are implemented in core layer.
- **Data Consumption**, primary focus of this component is the analysis of the data that is being stored in Datawarehouse and OLAP Server. Analysis can be performed inform of reporting, dashboard, and self-service BI.

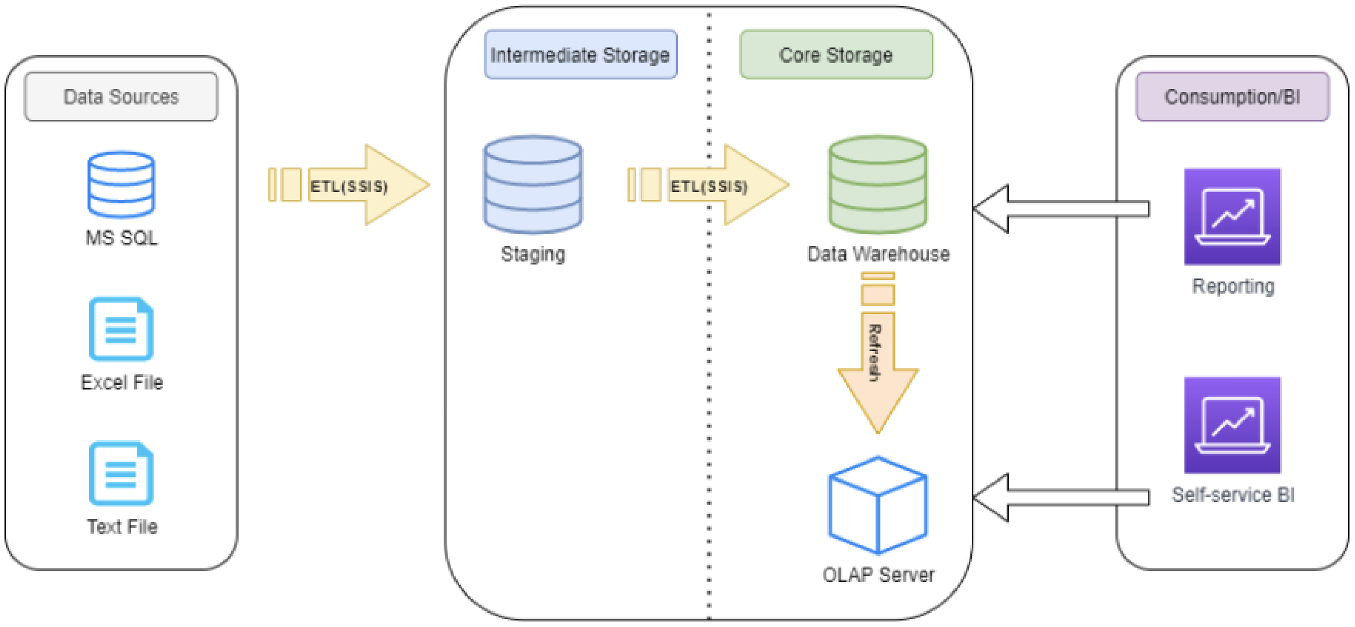


Figure 2 : High-Level Architecture

4. Datawarehouse Design & Development

1.1. Dimensional Model

Snowflake schema was selected to design the Datawarehouse schema for our dataset, considering the number of dimensional tables, fact tables and the relationship among these table, in addition to the aspects such as performance, speed, and availability.

There are mainly eight dimensional tables, DimDate, DimChannel, DimDeliveries, DimDrivers, DimHubs, DimOrderDetails, DimPayments, DimStore and one fact table FactOrder. All the dimensional tables are linked with fact table, except DimDrivers and DimHubs, as it is linked with DimDeliveries with “driver_id” and DimHubs with “hub_id” foreign key constraint.

Dimension:

Dimension Name	Type
DimDate	**Static/Date/Role Playing**
DimChannel	**Role Playing**
DimDeliveries	**Parent-Child Dimension**
DimDrivers	**Parent-Child Dimension /Slowly Changing Dimension**
DimHubs	**Parent-Child Dimension**
DimOrderDetails	**Role Playing**
DimPayments	**Role Playing**
DimStore	**Parent-Child Dimension /Role Playing**

Fact Table:

Fact Table Name	Type

FactOrders	**Accumulating Fact Table**

Hierarchies:

- DimHub hierarchical breakdown from state > city
- DimDelivery is mapped as hierarchical dimension of DimDriver
- DimStore is mapped as hierarchical dimension of DimHub
- DimDate hierarchical breakdown from Year > Quarter > Month > Date

Additional Fields:

Additional fields were added to all tables in the Datawarehouse layer, as shown below. These fields are derived attributes, with the date field getting its values from the system date and time. The DimDriver has two additional date field “start_date and end_date” as it is classified as slowly changing dimension. Further, surrogate keys have been added to all the dimensional tables, which is auto incrementing, and it’s used to map with tables.

Assumption(s)

- No assumption made during the Datawarehouse Design & Development Stage

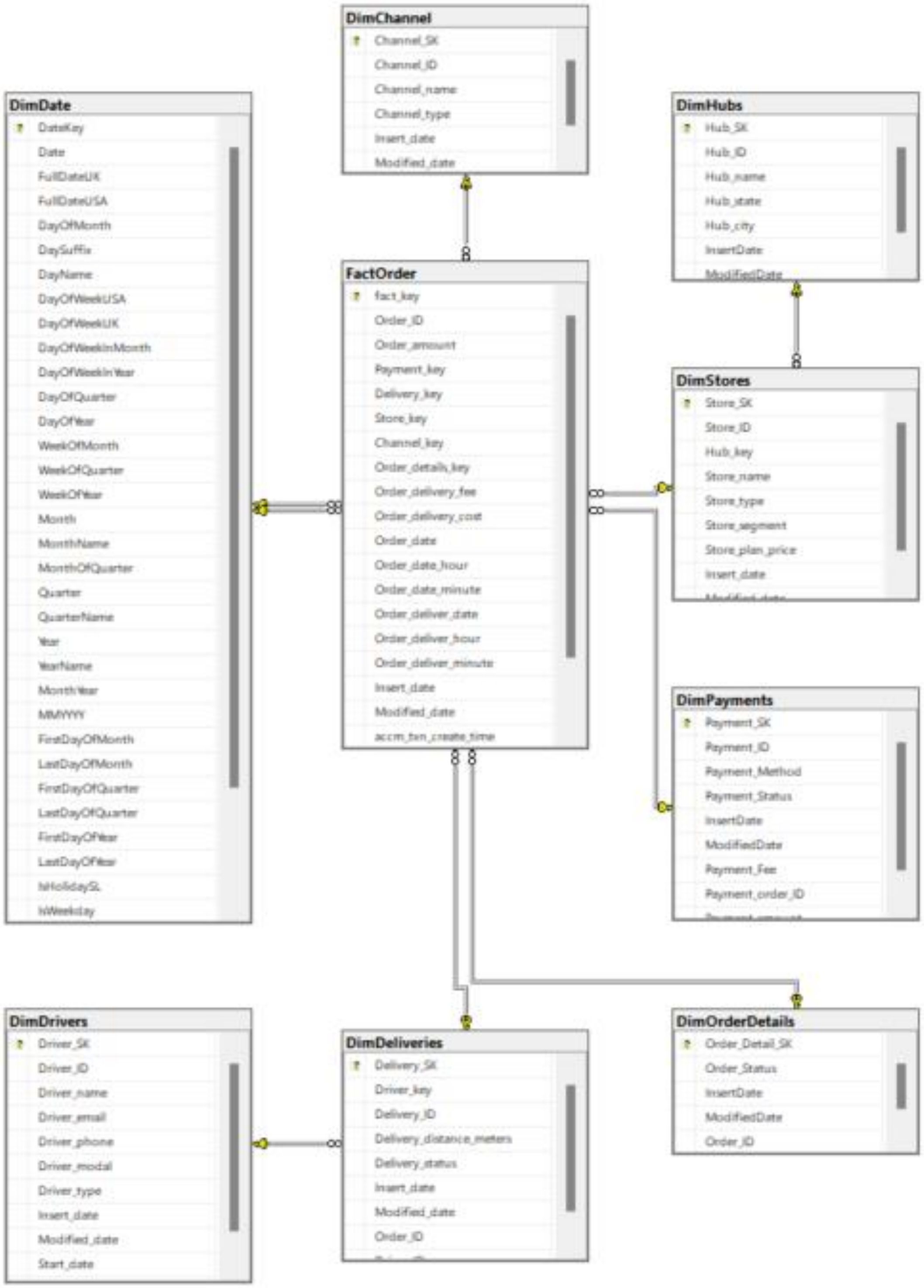


Figure 3 :Dimensional Model (Snowflake Schema)

5. ETL Development

5.1. ETL Process Overview

The diagram below illustrates the ETL development process in a high level along with the order of execution within each layer staring from Staging.

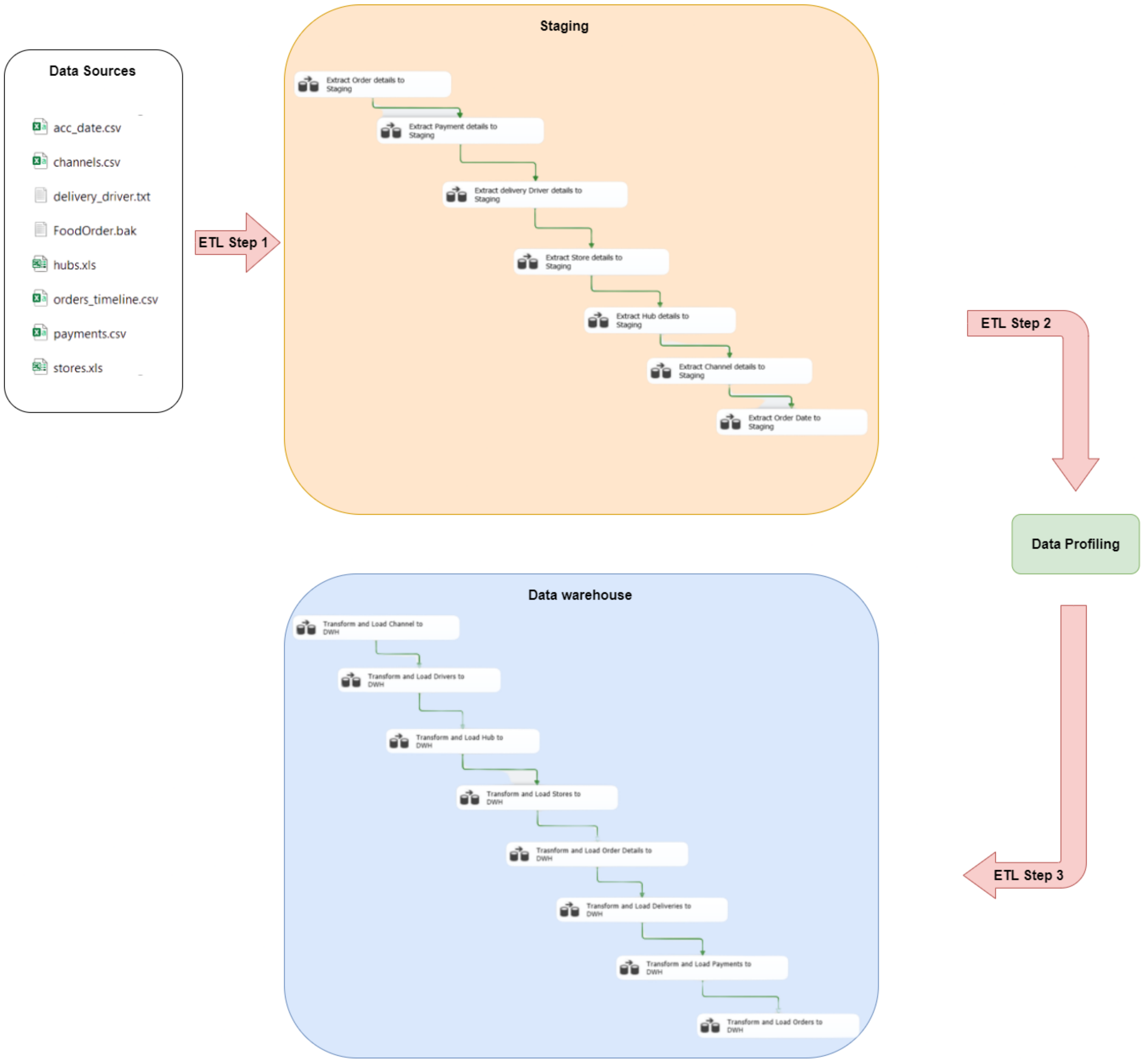
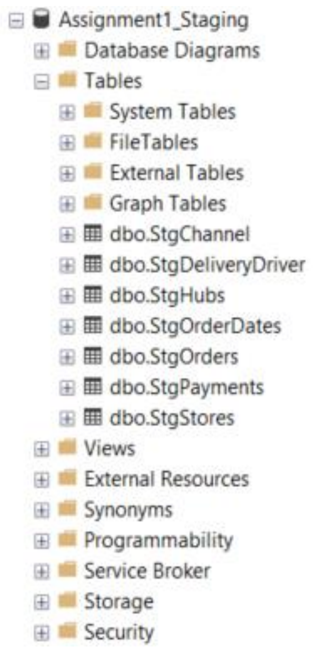

Data Sources
acc_date.csv
channels.csv
delivery_driver.txt
FoodOrder.bak
hubs.xls
orders_timeline.csv
payments.csv
stores.xls
ETL Step 1
Staging
Extract Order details to Staging
Extract Payment details to Staging
Extract delivery Driver details to Staging
Extract Store details to Staging
Extract Hub details to Staging
Extract Channel details to Staging
Extract Order Date to Staging
ETL Step 2
Data Profiling
ETL Step 3
Data warehouse
Transform and Load Channel to DWH
Transform and Load Drivers to DWH
Transform and Load Hub to DWH
Transform and Load Stores to DWH
Trasnform and Load Order Details to DWH
Transform and Load Deliveries to DWH
Transform and Load Payments to DWH
Transform and Load Orders to DWH

Figure 4 :ETL Process High-Level Overview

5.2. **ETL Development Process**

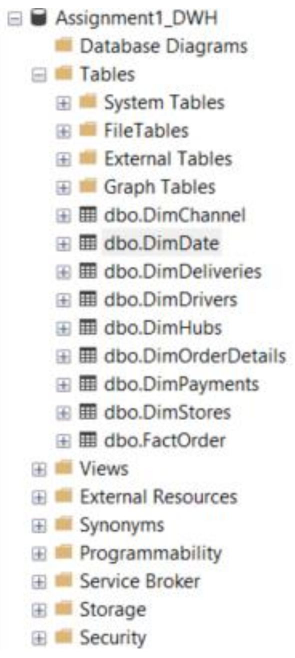
5.2.1. **Environment and Data Sources**

- The Data warehouse Solution is design and implement using MSSQL Server. In here two additional databases ware developed for staging as "Assignment1_Staging" and data warehouse process as "Assignment1_DWH". The Datawarehouse layer will extract, process, and load data from the staging layer, whereas the Staging layer will extract, transform, and load data from the source files.
- The Staging database tables are not pre-defined, because in staging layer it will be auto generated through the ETL process executed by SSIS.
- The Data warehouse tables are predesigned, total nine tables ware created .one for Fact table and rest of the tables are dimensions.



```
Assignment1_Staging
  Database Diagrams
  Tables
    System Tables
    FileTables
    External Tables
    Graph Tables
    dbo.StgChannel
    dbo.StgDeliveryDriver
    dbo.StgHubs
    dbo.StgOrderDates
    dbo.StgOrders
    dbo.StgPayments
    dbo.StgStores
  Views
  External Resources
  Synonyms
  Programmability
  Service Broker
  Storage
  Security
```

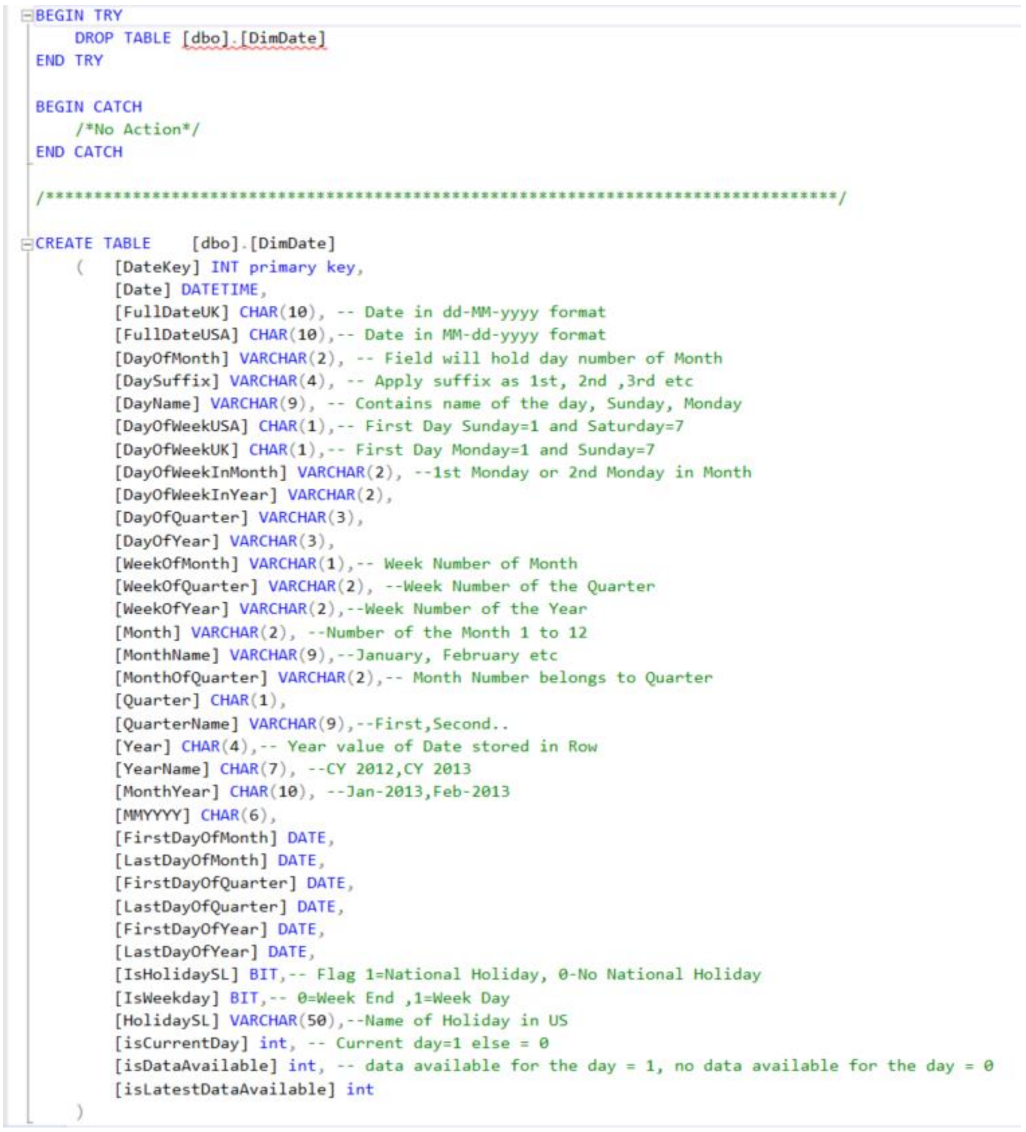
Figure 5 :Staging Database Structure



```
Assignment1_DWH
  Database Diagrams
  Tables
    System Tables
    FileTables
    External Tables
    Graph Tables
    dbo.DimChannel
    dbo.DimDate
    dbo.DimDeliveries
    dbo.DimDrivers
    dbo.DimHubs
    dbo.DimOrderDetails
    dbo.DimPayments
    dbo.DimStores
    dbo.FactOrder
  Views
  External Resources
  Synonyms
  Programmability
  Service Broker
  Storage
  Security
```

Figure 6 : DWH Database Structure

- “DimDate” is static and loaded manually, the values for the table were generated by using “Date Master” query in SQL.

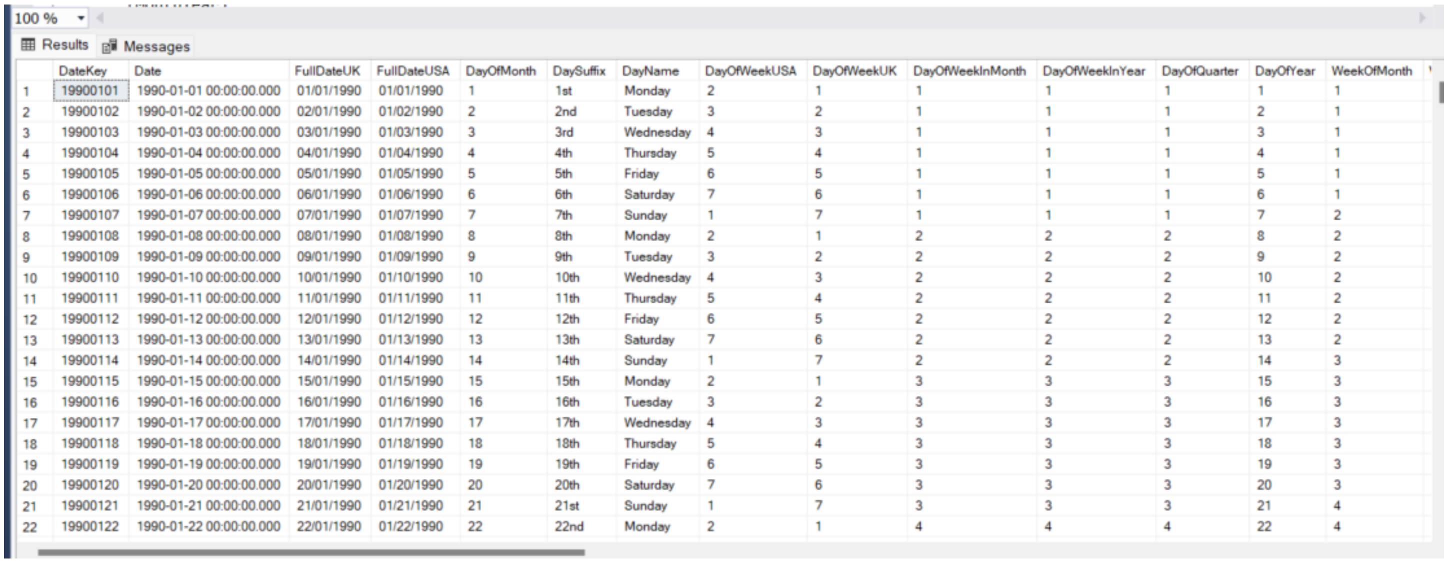


```sql
BEGIN TRY
    DROP TABLE [dbo].[DimDate]
END TRY

BEGIN CATCH
    /*No Action*/
END CATCH

/**********************************************************************************/

CREATE TABLE    [dbo].[DimDate]
    (   [DateKey] INT primary key,
        [Date] DATETIME,
        [FullDateUK] CHAR(10), -- Date in dd-MM-yyyy format
        [FullDateUSA] CHAR(10),-- Date in MM-dd-yyyy format
        [DayOfMonth] VARCHAR(2), -- Field will hold day number of Month
        [DaySuffix] VARCHAR(4), -- Apply suffix as 1st, 2nd ,3rd etc
        [DayName] VARCHAR(9), -- Contains name of the day, Sunday, Monday
        [DayOfWeekUSA] CHAR(1),-- First Day Sunday=1 and Saturday=7
        [DayOfWeekUK] CHAR(1),-- First Day Monday=1 and Sunday=7
        [DayOfWeekInMonth] VARCHAR(2), --1st Monday or 2nd Monday in Month
        [DayOfWeekInYear] VARCHAR(2),
        [DayOfQuarter] VARCHAR(3),
        [DayOfYear] VARCHAR(3),
        [WeekOfMonth] VARCHAR(1),-- Week Number of Month
        [WeekOfQuarter] VARCHAR(2), --Week Number of the Quarter
        [WeekOfYear] VARCHAR(2),--Week Number of the Year
        [Month] VARCHAR(2), --Number of the Month 1 to 12
        [MonthName] VARCHAR(9),--January, February etc
        [MonthOfQuarter] VARCHAR(2),-- Month Number belongs to Quarter
        [Quarter] CHAR(1),
        [QuarterName] VARCHAR(9),--First,Second..
        [Year] CHAR(4),-- Year value of Date stored in Row
        [YearName] CHAR(7), --CY 2012,CY 2013
        [MonthYear] CHAR(10), --Jan-2013,Feb-2013
        [MMYYYY] CHAR(6),
        [FirstDayOfMonth] DATE,
        [LastDayOfMonth] DATE,
        [FirstDayOfQuarter] DATE,
        [LastDayOfQuarter] DATE,
        [FirstDayOfYear] DATE,
        [LastDayOfYear] DATE,
        [IsHolidaySL] BIT,-- Flag 1=National Holiday, 0-No National Holiday
        [IsWeekday] BIT,-- 0=Week End ,1=Week Day
        [HolidaySL] VARCHAR(50),--Name of Holiday in US
        [isCurrentDay] int, -- Current day=1 else = 0
        [isDataAvailable] int, -- data available for the day = 1, no data available for the day = 0
        [isLatestDataAvailable] int
    )
```



100 %

Results | Messages

	DateKey	Date	FullDateUK	FullDateUSA	DayOfMonth	DaySuffix	DayName	DayOfWeekUSA	DayOfWeekUK	DayOfWeekInMonth	DayOfWeekInYear	DayOfQuarter	DayOfYear	WeekOfMonth
1	19900101	1990-01-01 00:00:00.000	01/01/1990	01/01/1990	1	1st	Monday	2	1	1	1	1	1	1
2	19900102	1990-01-02 00:00:00.000	02/01/1990	01/02/1990	2	2nd	Tuesday	3	2	1	1	1	2	1
3	19900103	1990-01-03 00:00:00.000	03/01/1990	01/03/1990	3	3rd	Wednesday	4	3	1	1	1	3	1
4	19900104	1990-01-04 00:00:00.000	04/01/1990	01/04/1990	4	4th	Thursday	5	4	1	1	1	4	1
5	19900105	1990-01-05 00:00:00.000	05/01/1990	01/05/1990	5	5th	Friday	6	5	1	1	1	5	1
6	19900106	1990-01-06 00:00:00.000	06/01/1990	01/06/1990	6	6th	Saturday	7	6	1	1	1	6	1
7	19900107	1990-01-07 00:00:00.000	07/01/1990	01/07/1990	7	7th	Sunday	1	7	1	1	1	7	2
8	19900108	1990-01-08 00:00:00.000	08/01/1990	01/08/1990	8	8th	Monday	2	1	2	2	2	8	2
9	19900109	1990-01-09 00:00:00.000	09/01/1990	01/09/1990	9	9th	Tuesday	3	2	2	2	2	9	2
10	19900110	1990-01-10 00:00:00.000	10/01/1990	01/10/1990	10	10th	Wednesday	4	3	2	2	2	10	2
11	19900111	1990-01-11 00:00:00.000	11/01/1990	01/11/1990	11	11th	Thursday	5	4	2	2	2	11	2
12	19900112	1990-01-12 00:00:00.000	12/01/1990	01/12/1990	12	12th	Friday	6	5	2	2	2	12	2
13	19900113	1990-01-13 00:00:00.000	13/01/1990	01/13/1990	13	13th	Saturday	7	6	2	2	2	13	2
14	19900114	1990-01-14 00:00:00.000	14/01/1990	01/14/1990	14	14th	Sunday	1	7	2	2	2	14	3
15	19900115	1990-01-15 00:00:00.000	15/01/1990	01/15/1990	15	15th	Monday	2	1	3	3	3	15	3
16	19900116	1990-01-16 00:00:00.000	16/01/1990	01/16/1990	16	16th	Tuesday	3	2	3	3	3	16	3
17	19900117	1990-01-17 00:00:00.000	17/01/1990	01/17/1990	17	17th	Wednesday	4	3	3	3	3	17	3
18	19900118	1990-01-18 00:00:00.000	18/01/1990	01/18/1990	18	18th	Thursday	5	4	3	3	3	18	3
19	19900119	1990-01-19 00:00:00.000	19/01/1990	01/19/1990	19	19th	Friday	6	5	3	3	3	19	3
20	19900120	1990-01-20 00:00:00.000	20/01/1990	01/20/1990	20	20th	Saturday	7	6	3	3	3	20	3
21	19900121	1990-01-21 00:00:00.000	21/01/1990	01/21/1990	21	21st	Sunday	1	7	3	3	3	21	4
22	19900122	1990-01-22 00:00:00.000	22/01/1990	01/22/1990	22	22nd	Monday	2	1	4	4	4	22	4

5.2.2. Staging

A new SSIS package was created for this process called “StgAssignment” in a Data Tool application.

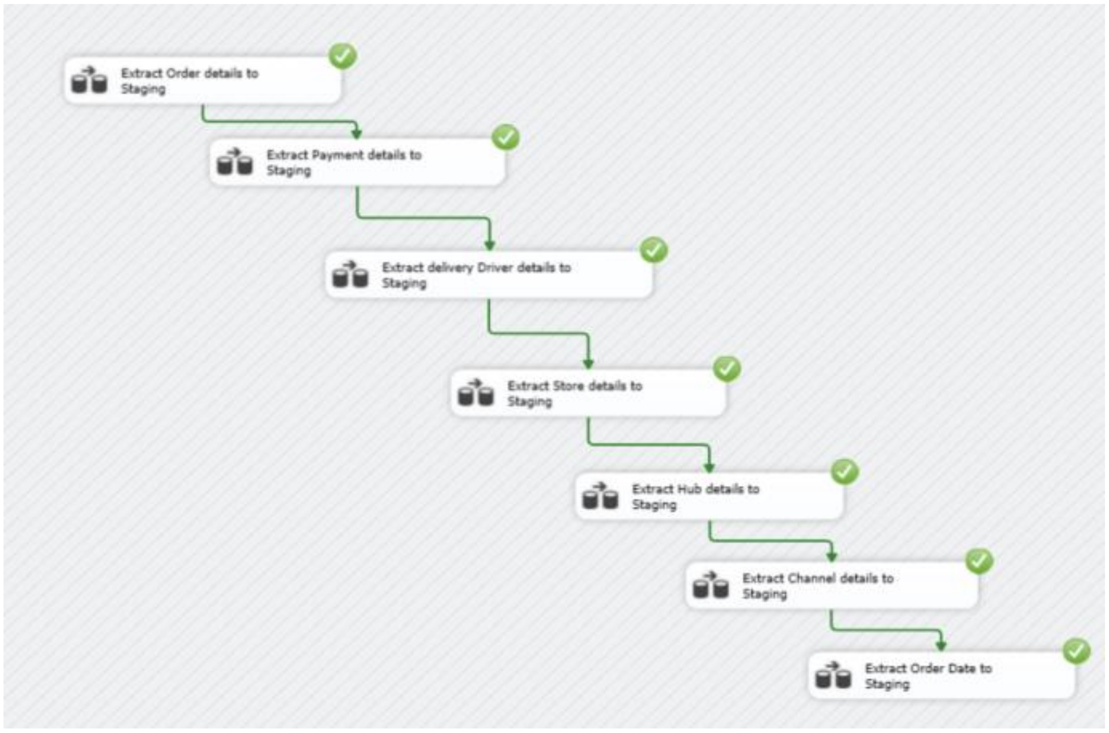

Extract Order details to Staging
Extract Payment details to Staging
Extract delivery Driver details to Staging
Extract Store details to Staging
Extract Hub details to Staging
Extract Channel details to Staging
Extract Order Date to Staging

Figure 7 :Overall Staging

(a) Extract & Load Order Details to the Staging

In the file sources Order details are stored in a FoodOrder.bak. after restoring this database backup file, its structure is showed in below figure.

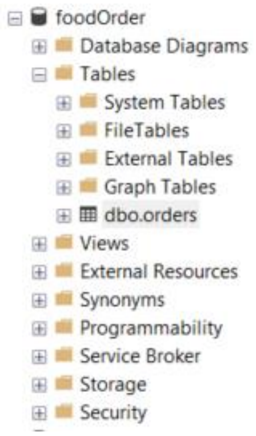


Figure 8 :Food Order database Structure

In the same package, a control task was created. withing the control task the data flow was designed along with an event handler as onPreExecute.

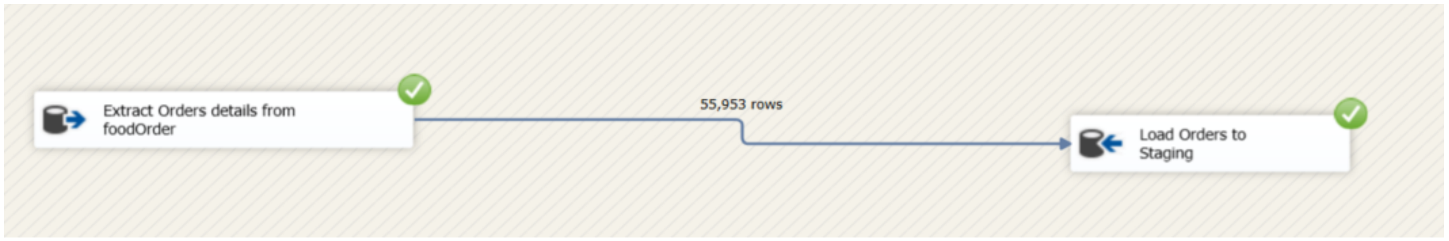


Figure 9 :Data flow task StgOrders

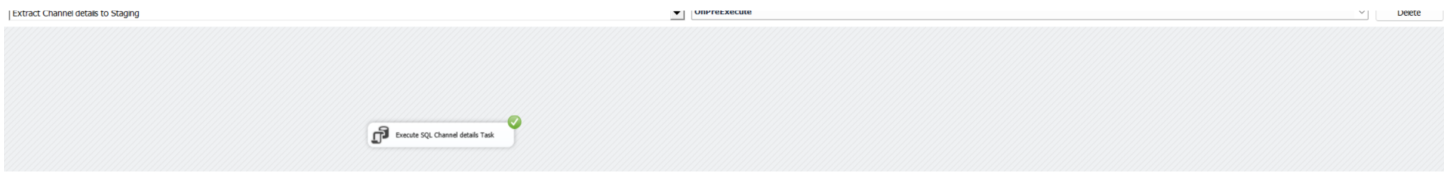
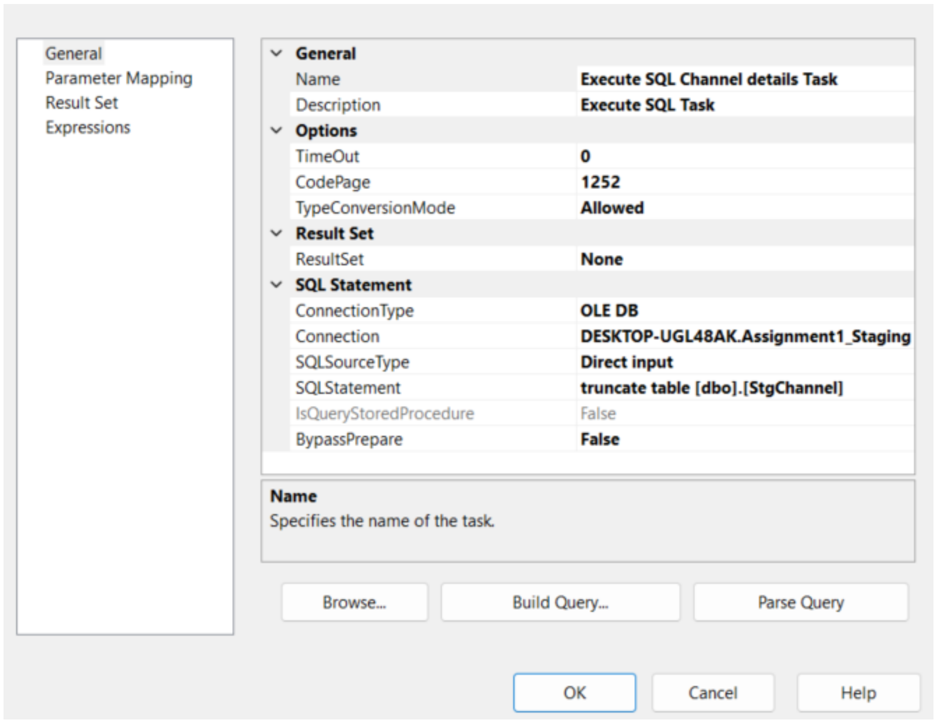

Extract Channel details to Staging
OnPreExecute
Delete
Execute SQL Channel details Task

Figure 10 : Event Handler

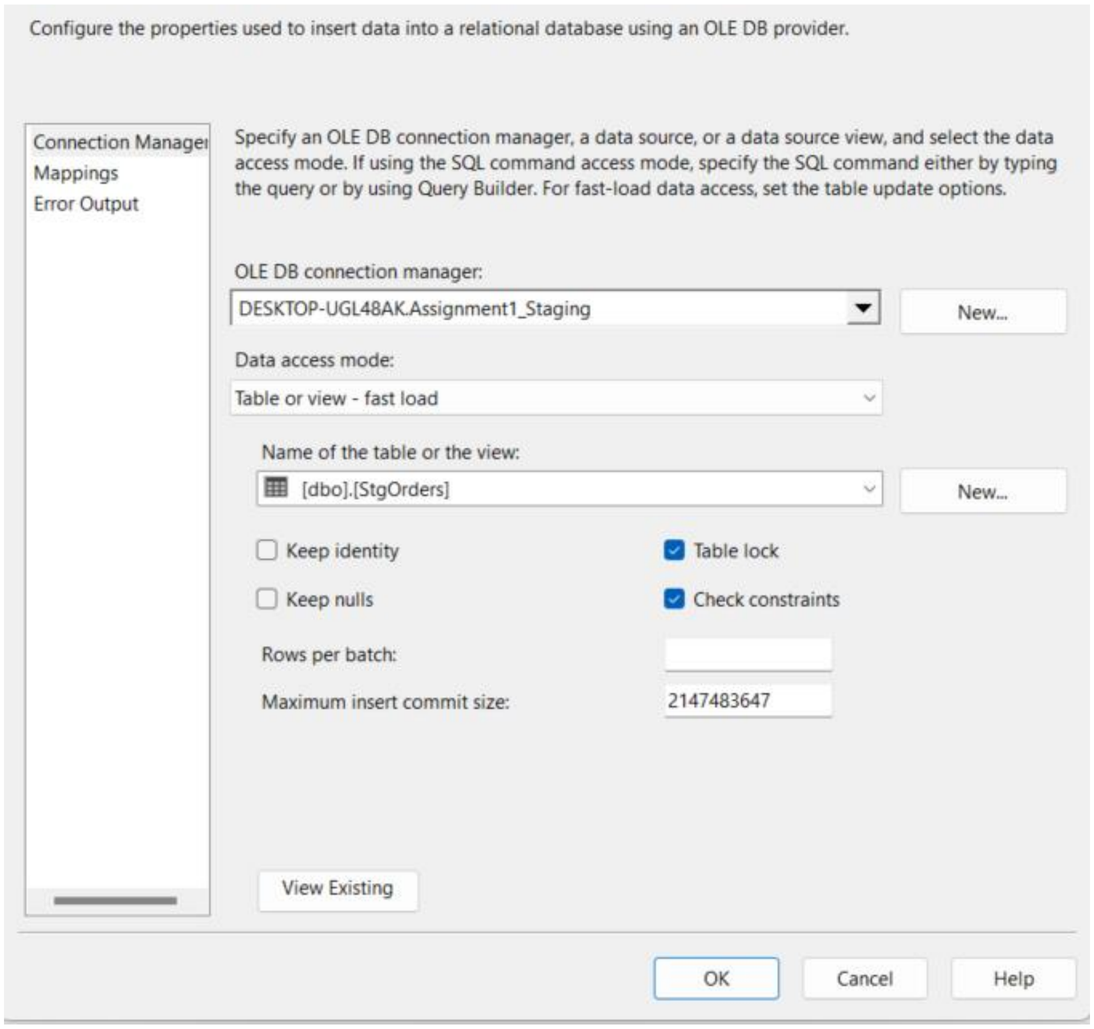
- Same as other Data flow task also have Event Handlers



General	
Name	**Execute SQL Channel details Task**
Description	**Execute SQL Task**
Options	
TimeOut	**0**
CodePage	**1252**
TypeConversionMode	**Allowed**
Result Set	
ResultSet	**None**
SQL Statement	
ConnectionType	**OLE DB**
Connection	**DESKTOP-UGL48AK.Assignment1_Staging**
SQLSourceType	**Direct input**
SQLStatement	**truncate table [dbo].[StgChannel]**
IsQueryStoredProcedure	False
BypassPrepare	**False**

Name
Specifies the name of the task.

Figure 11 :Event Handler Properties


Configure the properties used to insert data into a relational database using an OLE DB provider.

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder. For fast-load data access, set the table update options.

OLE DB connection manager: DESKTOP-UGL48AK.Assignment1_Staging

Data access mode: Table or view - fast load

Name of the table or the view: [dbo].[StgOrders]

☐ Keep identity

✓ Table lock

☐ Keep nulls

✓ Check constraints

Rows per batch:

Maximum insert commit size: 2147483647

Figure 12 :Staging Orders connection

(b) Extract & Load Payment Details to the Staging

The source file type for Payment details is CSV. According to the data type, to Extract Data from CSV, the best option is Flat file source.

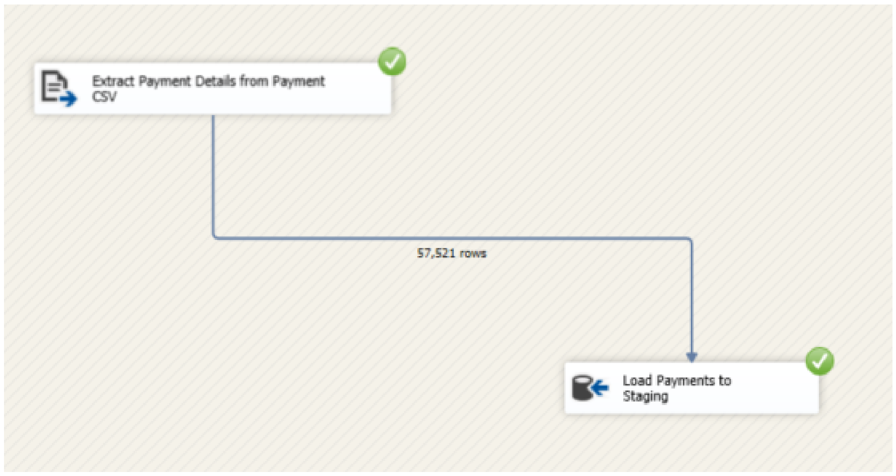
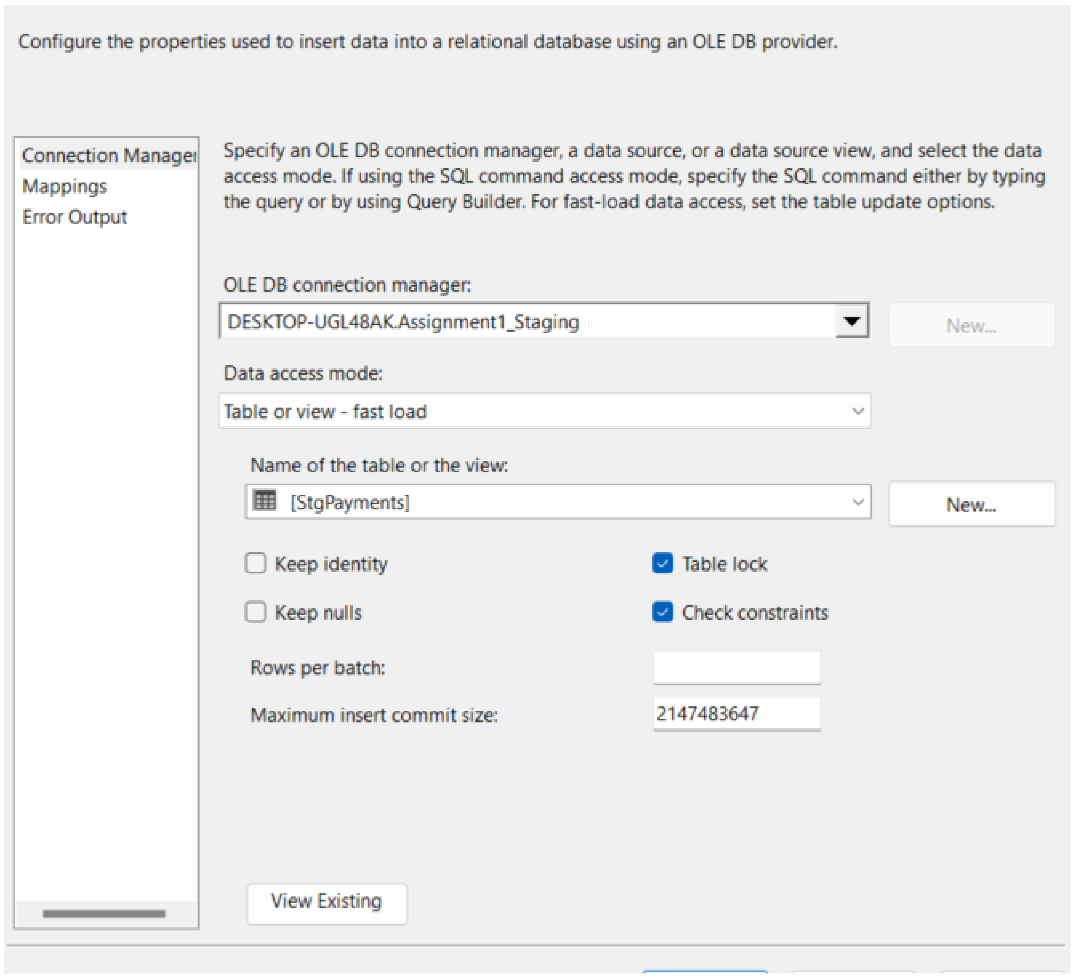


Figure 13 : Data flow Task for StgPayment



Configure the properties used to insert data into a relational database using an OLE DB provider.

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder. For fast-load data access, set the table update options.

OLE DB connection manager: DESKTOP-UGL48AK.Assignment1_Staging

Data access mode: Table or view - fast load

Name of the table or the view: [StgPayments]

Keep identity [ ]
Table lock [✓]
Keep nulls [ ]
Check constraints [✓]

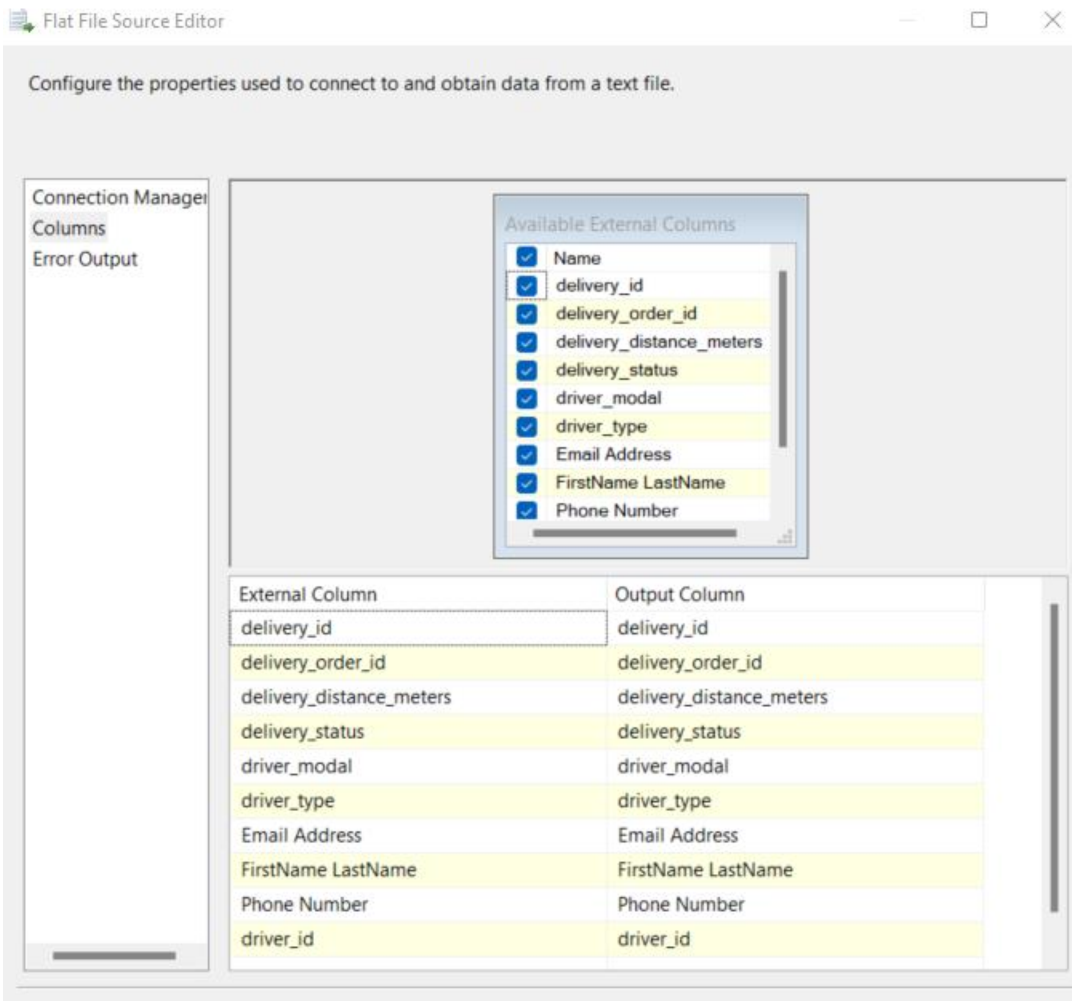
Rows per batch:

Maximum insert commit size: 2147483647

Figure 14 :Staging Payment connection

(c) Extract & Load Delivery Driver Details to the Staging

The File type of the Source is txt. Therefore, most suitable data flow task is flat file source. Because of the file format, need to clarify delimiter when creating the connection manager.


Flat File Source Editor

Configure the properties used to connect to and obtain data from a text file.

Connection Manager
Columns
Error Output

Available External Columns

- [✓] Name
- [✓] delivery_id
- [✓] delivery_order_id
- [✓] delivery_distance_meters
- [✓] delivery_status
- [✓] driver_modal
- [✓] driver_type
- [✓] Email Address
- [✓] FirstName LastName
- [✓] Phone Number

External Column	Output Column
delivery_id	delivery_id
delivery_order_id	delivery_order_id
delivery_distance_meters	delivery_distance_meters
delivery_status	delivery_status
driver_modal	driver_modal
driver_type	driver_type
Email Address	Email Address
FirstName LastName	FirstName LastName
Phone Number	Phone Number
driver_id	driver_id

Figure 15 :Column Mapping StgPayment

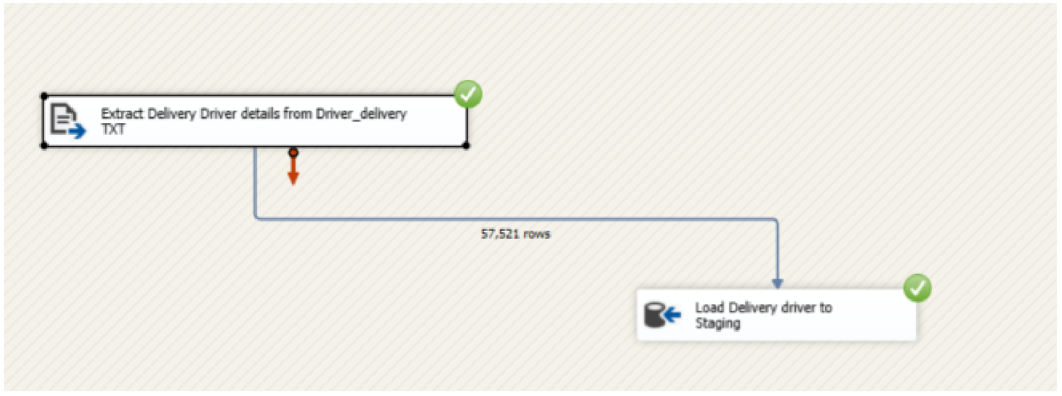

Extract Delivery Driver details from Driver_delivery TXT
57,521 rows
Load Delivery driver to Staging

Figure 16 :Data flow task for Stg Payment

(d) Extract & Load Store Details to the Staging

Store Details are contained in Excel file**. Extract** all the data using Excel Source in Data flow to OLED DB destination source.

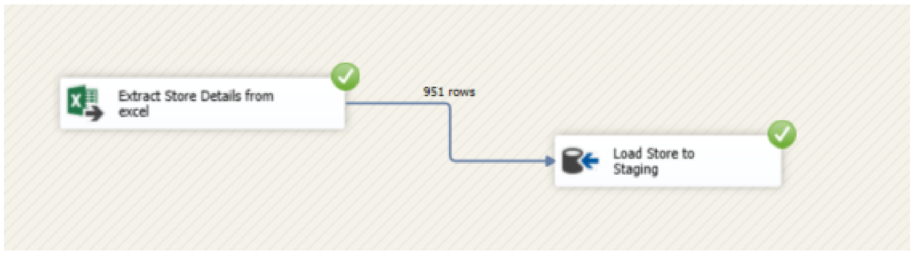
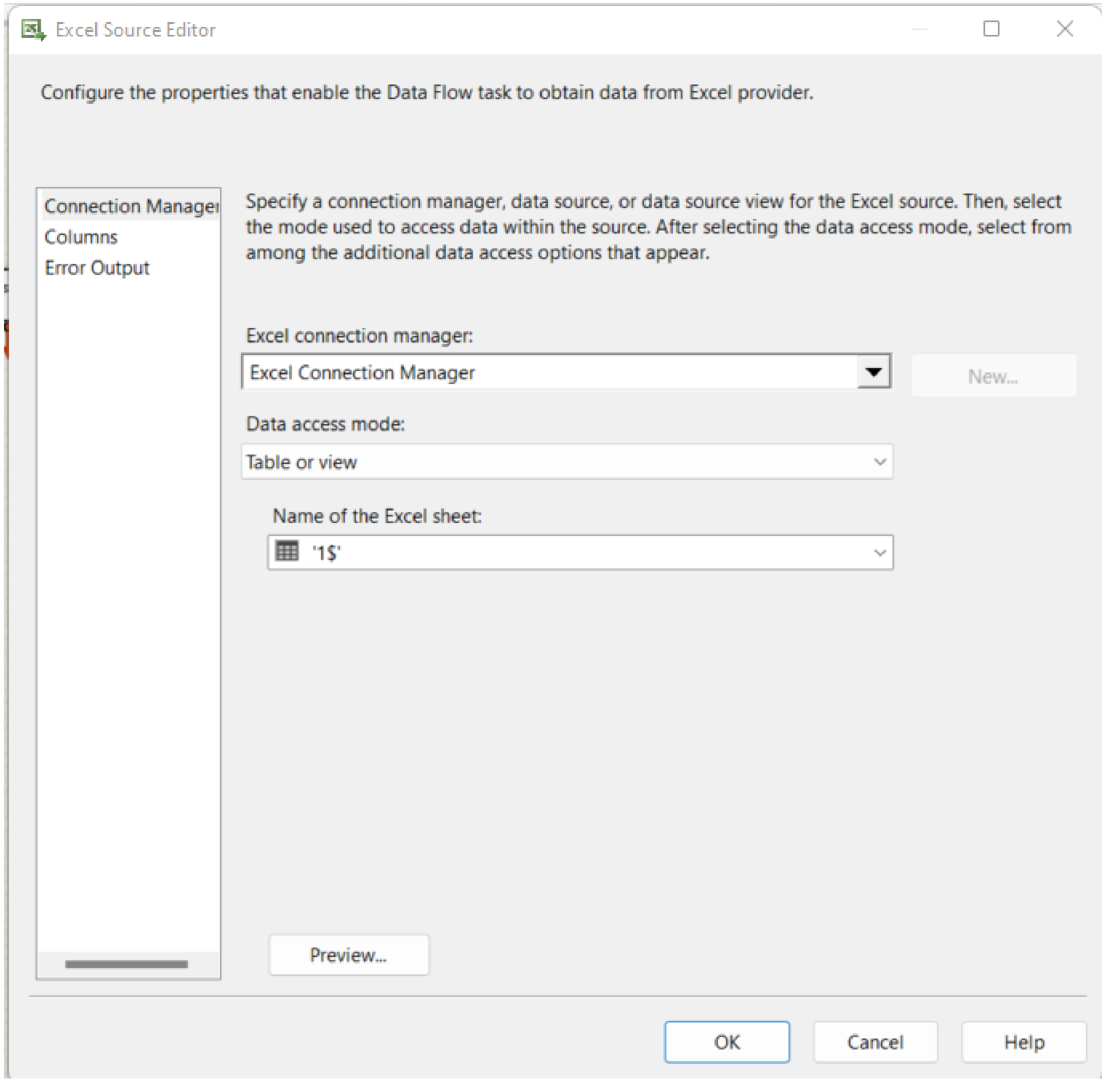


Figure 17 :Data flow task for Stg Store



Configure the properties that enable the Data Flow task to obtain data from Excel provider.

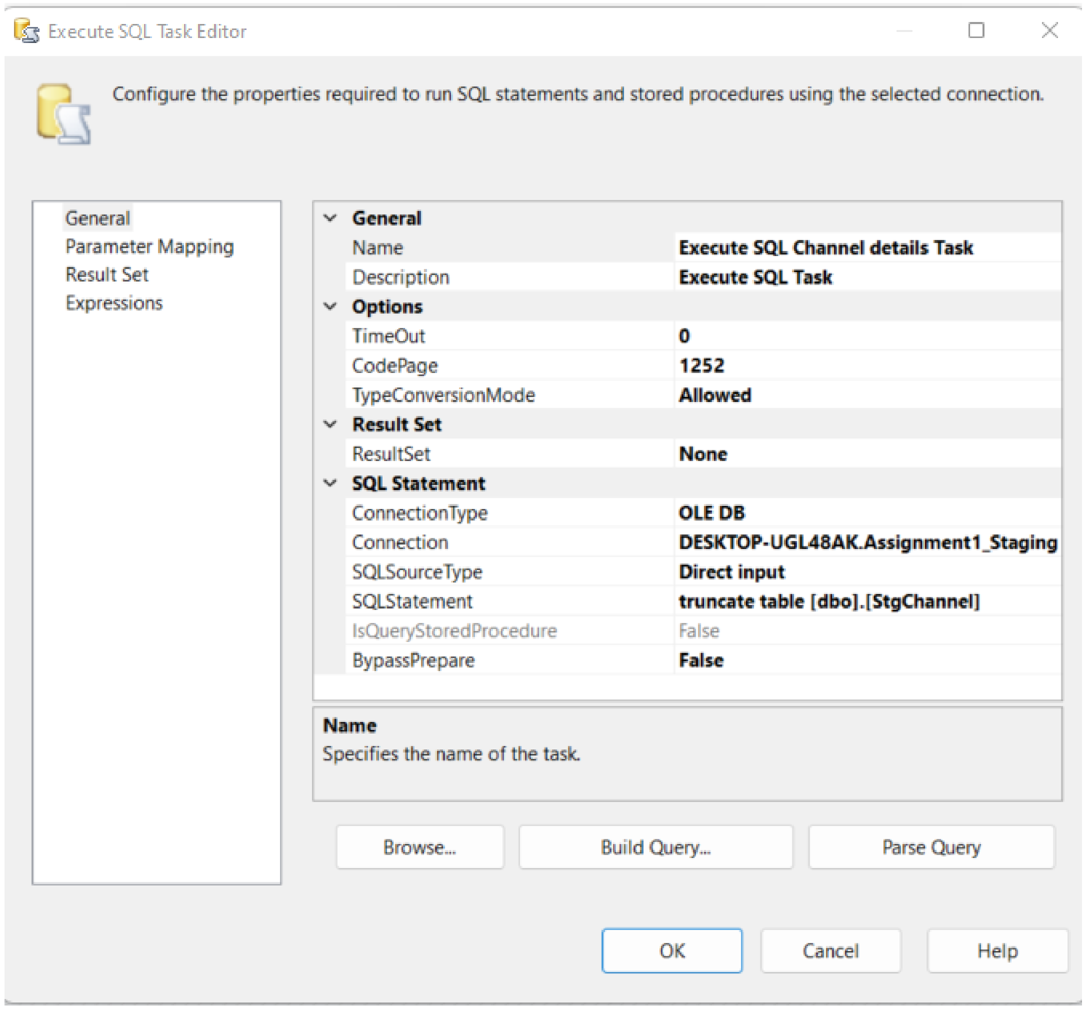
Specify a connection manager, data source, or data source view for the Excel source. Then, select the mode used to access data within the source. After selecting the data access mode, select from among the additional data access options that appear.

Excel connection manager: Excel Connection Manager

Data access mode: Table or view

Name of the Excel sheet: '1$'

Figure 18 :Stg Store connection


Execute SQL Task Editor

Configure the properties required to run SQL statements and stored procedures using the selected connection.

General	
Name	**Execute SQL Channel details Task**
Description	**Execute SQL Task**
Options	
TimeOut	**0**
CodePage	**1252**
TypeConversionMode	**Allowed**
Result Set	
ResultSet	**None**
SQL Statement	
ConnectionType	**OLE DB**
Connection	**DESKTOP-UGL48AK.Assignment1_Staging**
SQLSourceType	**Direct input**
SQLStatement	**truncate table [dbo].[StgChannel]**
IsQueryStoredProcedure	False
BypassPrepare	**False**

Name
Specifies the name of the task.

Figure 19 :Stg Store event handler

(e) Extract & Load Hub Details to the Staging

Hub Details are contained in Excel file**. Extract** all the data using Excel Source in Data flow to OLED DB destination source.

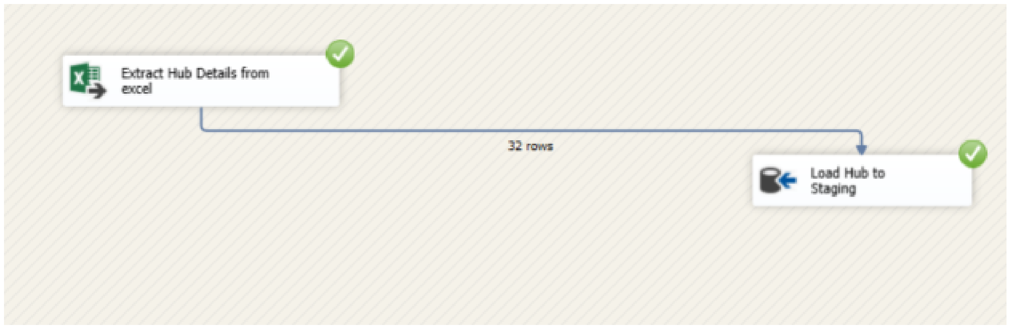
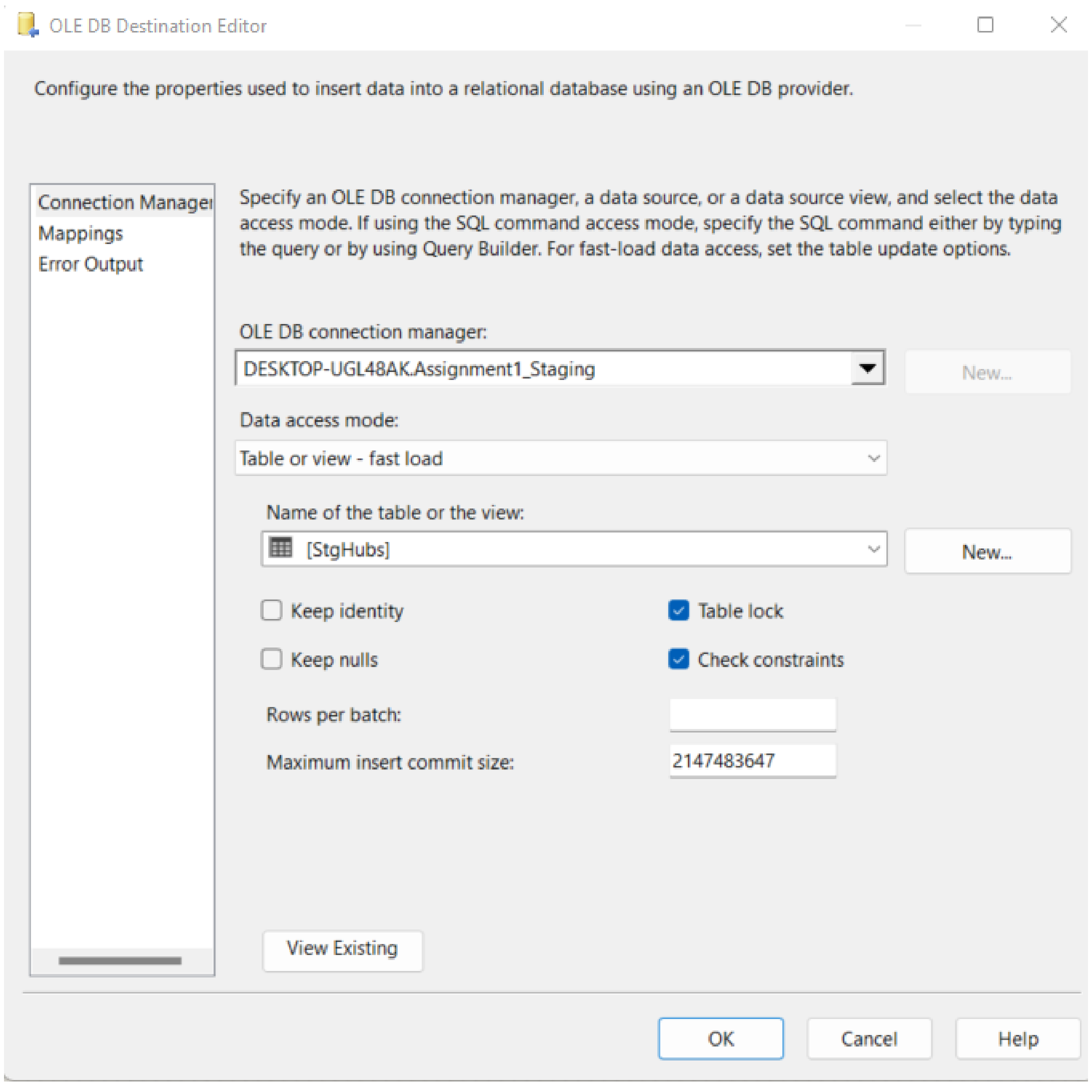


Figure 20 :Stg Hubs Data flow task


OLE DB Destination Editor

Configure the properties used to insert data into a relational database using an OLE DB provider.

Connection Manager
Mappings
Error Output

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder. For fast-load data access, set the table update options.

OLE DB connection manager: DESKTOP-UGL48AK.Assignment1_Staging

Data access mode: Table or view - fast load

Name of the table or the view: [StgHubs]

- [ ] Keep identity
- [x] Table lock
- [ ] Keep nulls
- [x] Check constraints

Rows per batch:

Maximum insert commit size: 2147483647

Figure 21 :Stg hub connection

(f) Extract & Load Channel Details to the Staging

The source file type for Channel details is CSV. According to the data type, to Extract Data from CSV, the best option is Flat file source.

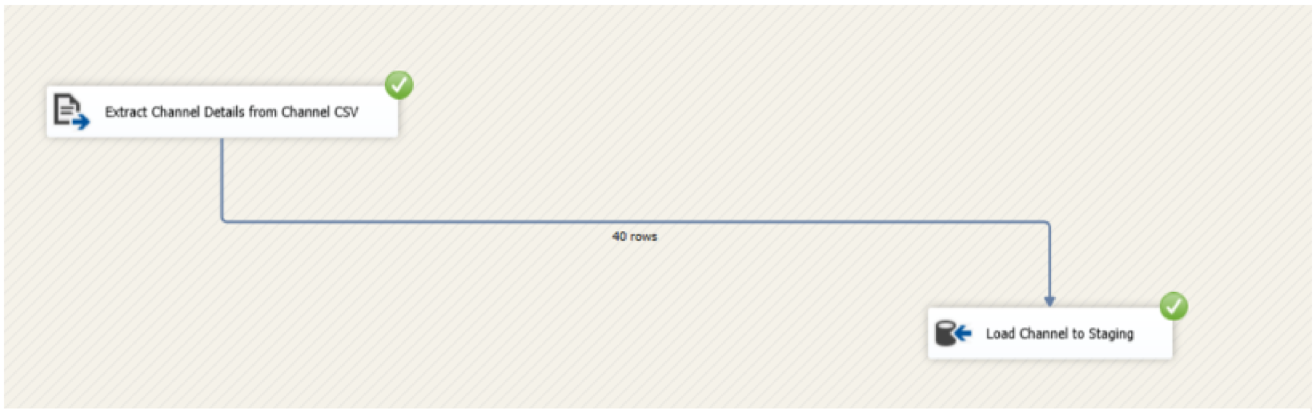
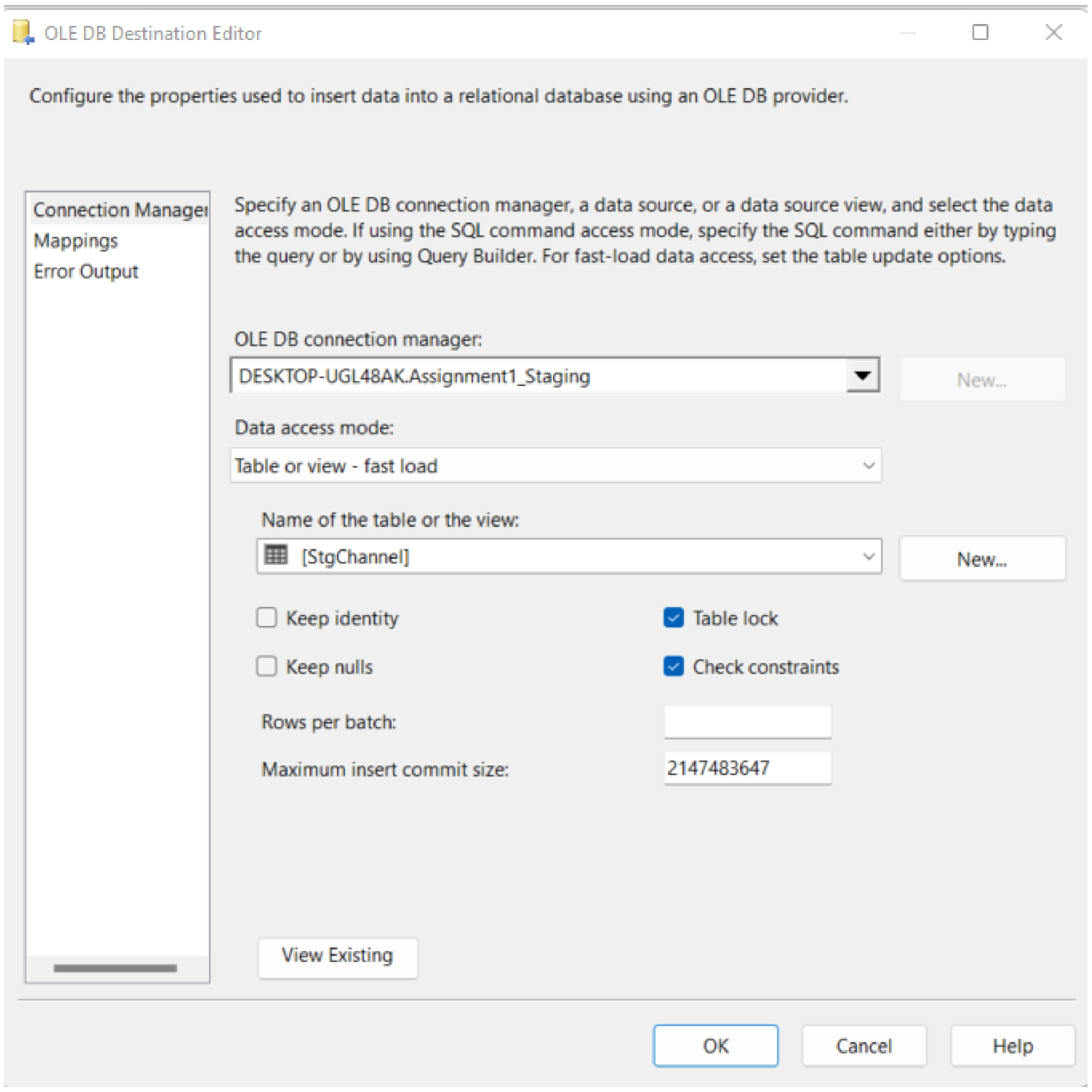


Figure 22 : Stg Channel data flow task


OLE DB Destination Editor

Configure the properties used to insert data into a relational database using an OLE DB provider.

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder. For fast-load data access, set the table update options.

OLE DB connection manager: DESKTOP-UGL48AK.Assignment1_Staging

Data access mode: Table or view - fast load

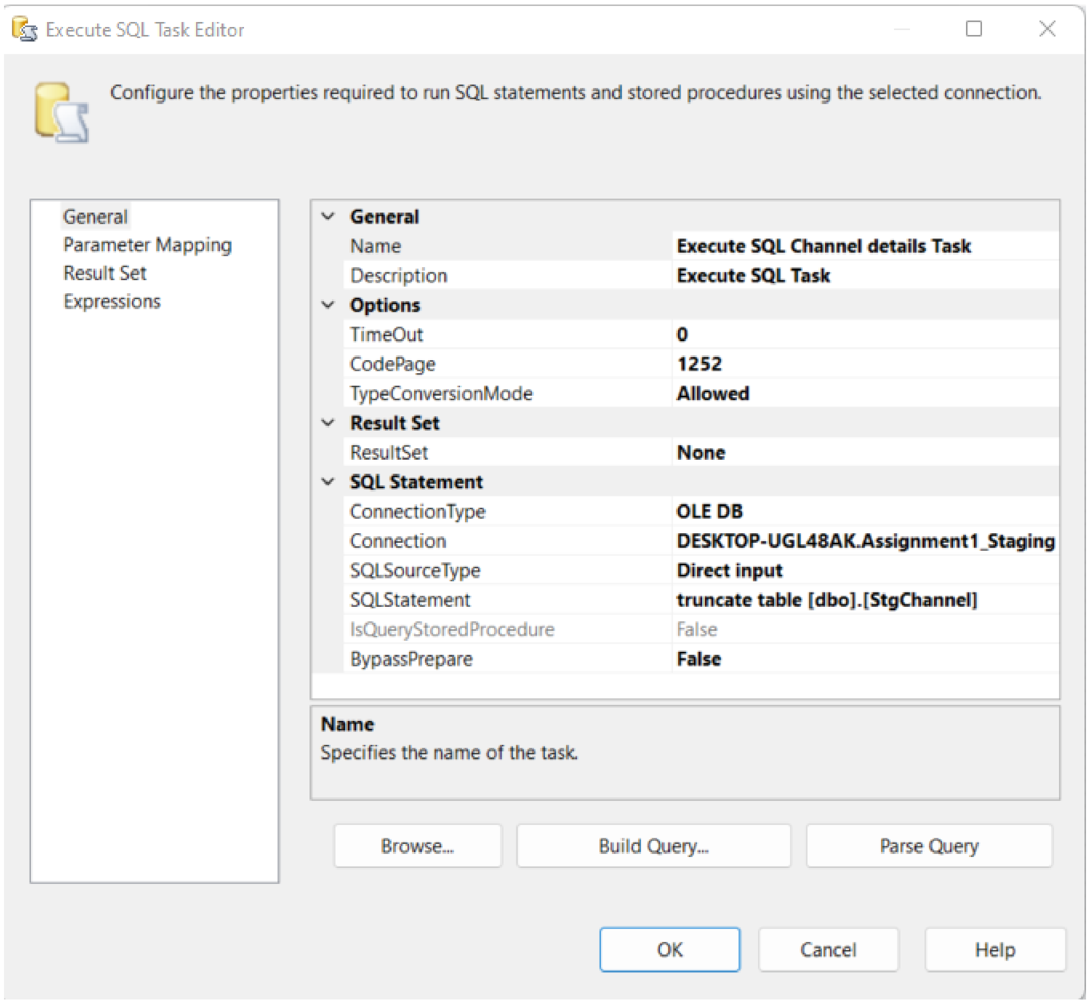
Name of the table or the view: [StgChannel]

[ ] Keep identity
[✓] Table lock
[ ] Keep nulls
[✓] Check constraints

Rows per batch:

Maximum insert commit size: 2147483647

Figure 23 :Stg channel connection


Execute SQL Task Editor

Configure the properties required to run SQL statements and stored procedures using the selected connection.

General	
Name	Execute SQL Channel details Task
Description	Execute SQL Task
Options	
TimeOut	0
CodePage	1252
TypeConversionMode	Allowed
Result Set	
ResultSet	None
SQL Statement	
ConnectionType	OLE DB
Connection	DESKTOP-UGL48AK.Assignment1_Staging
SQLSourceType	Direct input
SQLStatement	truncate table [dbo].[StgChannel]
IsQueryStoredProcedure	False
BypassPrepare	False

Name
Specifies the name of the task.

Figure 24 : Stg Channel event handler

(g) Extract & Load Order Date Details to the Staging

The source file type for Order date details is CSV. According to the data type, to Extract Data from CSV, the best option is Flat file source.

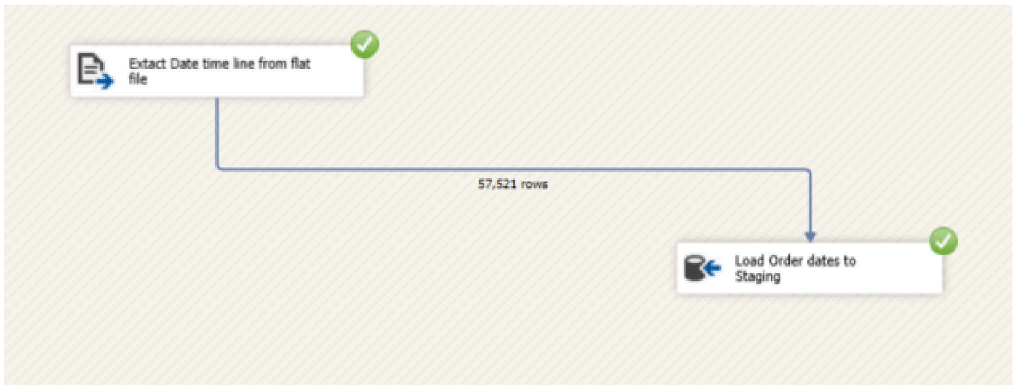
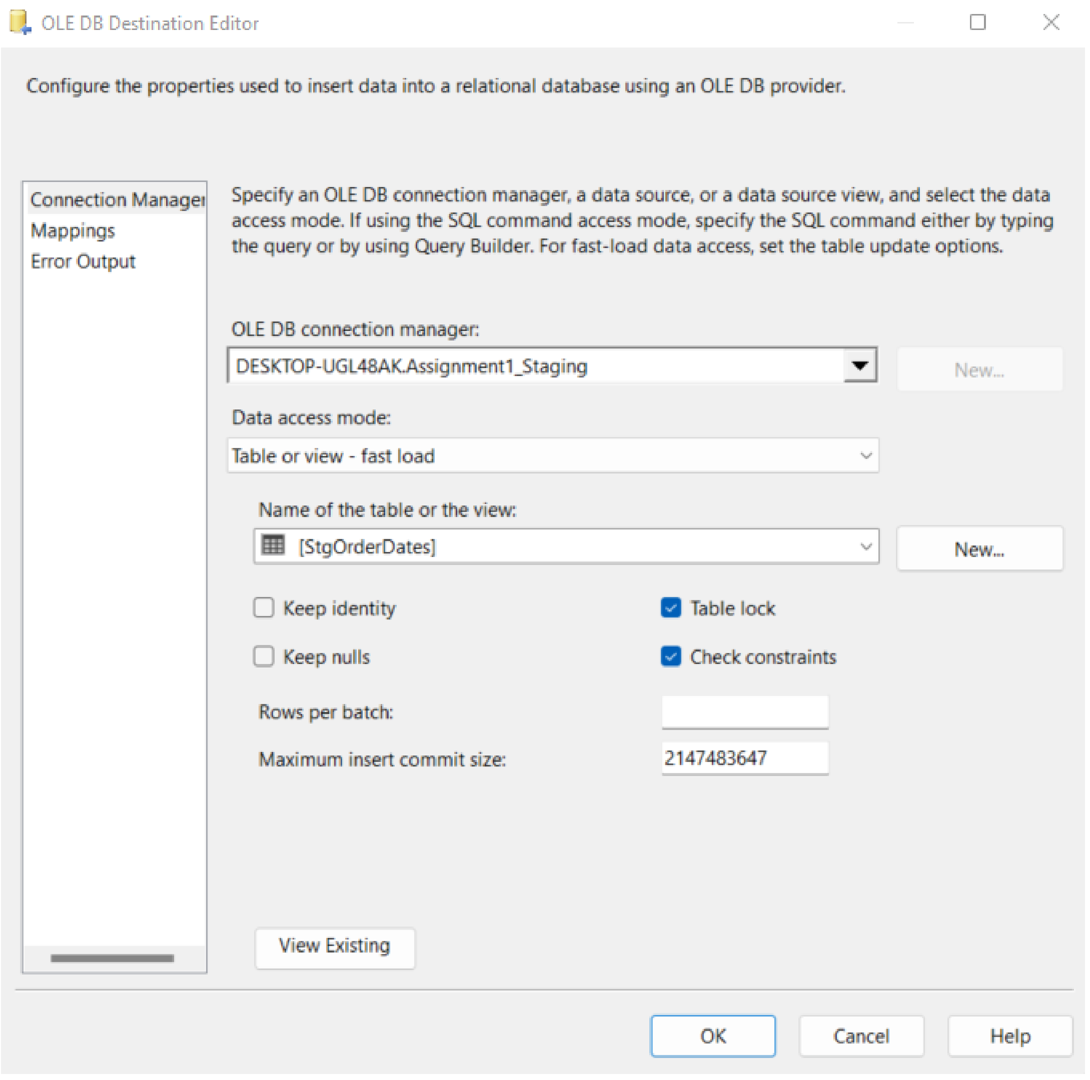

Extact Date time line from flat file
57,521 rows
Load Order dates to Staging

Figure 25 : Stg order dates data flow task



OLE DB Destination Editor

Configure the properties used to insert data into a relational database using an OLE DB provider.

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder. For fast-load data access, set the table update options.

OLE DB connection manager: DESKTOP-UGL48AK.Assignment1_Staging

Data access mode: Table or view - fast load

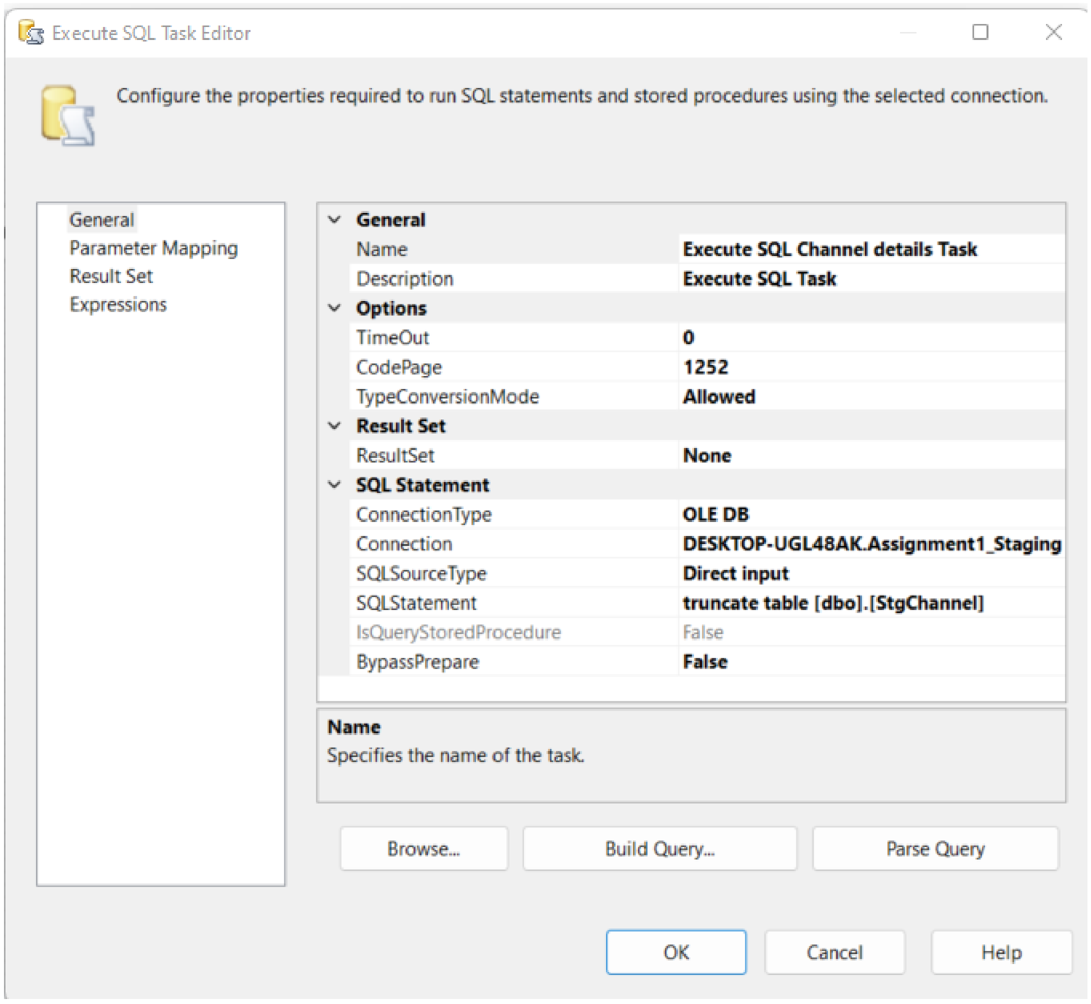
Name of the table or the view: [StgOrderDates]

- [ ] Keep identity
- [x] Table lock
- [ ] Keep nulls
- [x] Check constraints

Rows per batch:

Maximum insert commit size: 2147483647

Figure 26 : Stg Order dates connection


Execute SQL Task Editor

Configure the properties required to run SQL statements and stored procedures using the selected connection.

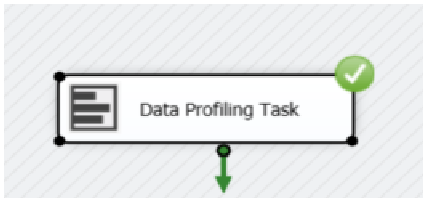
General	
Name	Execute SQL Channel details Task
Description	Execute SQL Task
Options	
TimeOut	0
CodePage	1252
TypeConversionMode	Allowed
Result Set	
ResultSet	None
SQL Statement	
ConnectionType	OLE DB
Connection	DESKTOP-UGL48AK.Assignment1_Staging
SQLSourceType	Direct input
SQLStatement	truncate table [dbo].[StgChannel]
IsQueryStoredProcedure	False
BypassPrepare	False

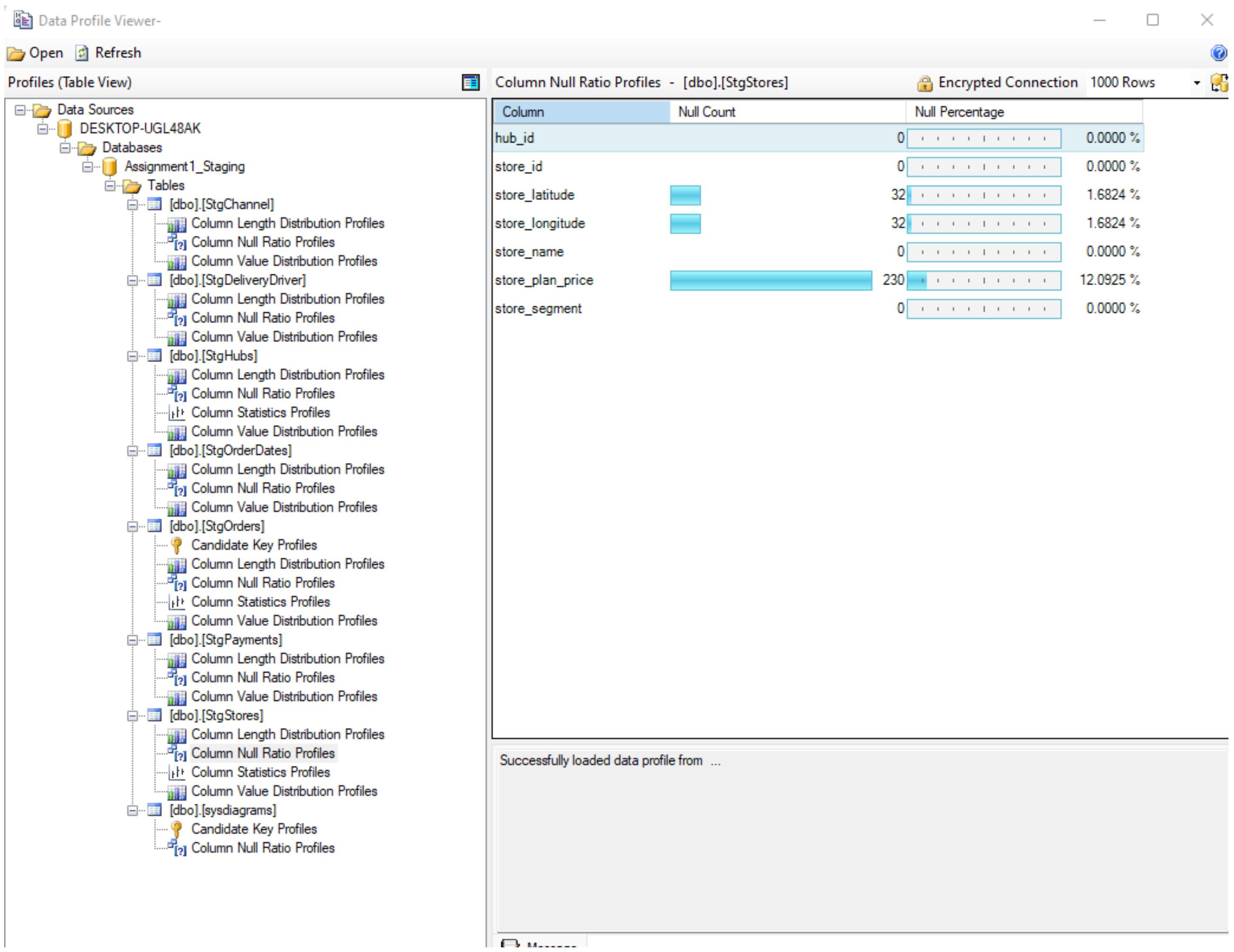
Name
Specifies the name of the task.

Figure 27 : Stg order Dates event handler

5.2.3. Data Profiling

Once the Step 01 was completed successfully, with an objective of assessing the quality, nature and null values of the data loaded to the staging layer Data Profiling was performed on all the tables.





Data Profile Viewer-

Open Refresh

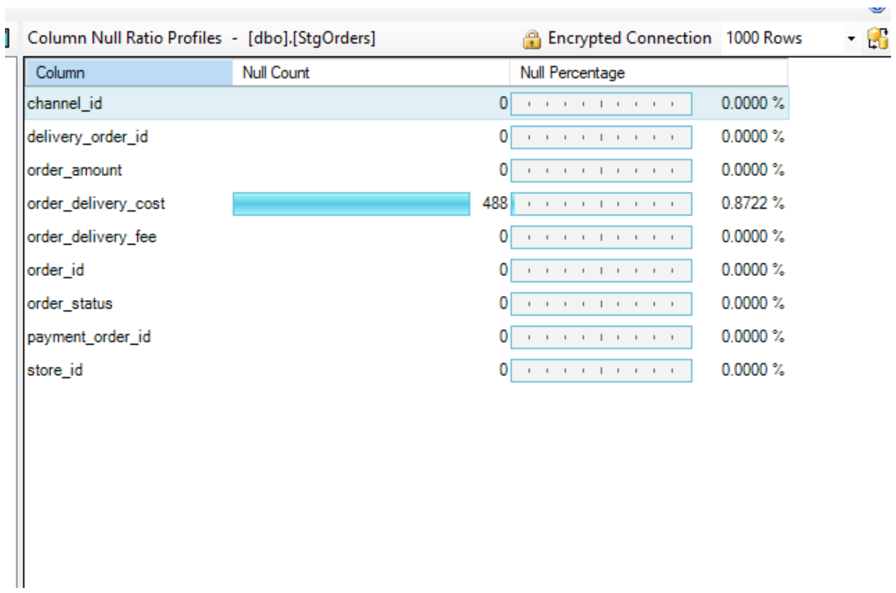
Profiles (Table View)

- Data Sources
  - DESKTOP-UGL48AK
    - Databases
      - Assignment1_Staging
        - Tables
          - [dbo].[StgChannel]
            - Column Length Distribution Profiles
            - Column Null Ratio Profiles
            - Column Value Distribution Profiles
          - [dbo].[StgDeliveryDriver]
            - Column Length Distribution Profiles
            - Column Null Ratio Profiles
            - Column Value Distribution Profiles
          - [dbo].[StgHubs]
            - Column Length Distribution Profiles
            - Column Null Ratio Profiles
            - Column Statistics Profiles
            - Column Value Distribution Profiles
          - [dbo].[StgOrderDates]
            - Column Length Distribution Profiles
            - Column Null Ratio Profiles
            - Column Value Distribution Profiles
          - [dbo].[StgOrders]
            - Candidate Key Profiles
            - Column Length Distribution Profiles
            - Column Null Ratio Profiles
            - Column Statistics Profiles
            - Column Value Distribution Profiles
          - [dbo].[StgPayments]
            - Column Length Distribution Profiles
            - Column Null Ratio Profiles
            - Column Value Distribution Profiles
          - [dbo].[StgStores]
            - Column Length Distribution Profiles
            - Column Null Ratio Profiles
            - Column Statistics Profiles
            - Column Value Distribution Profiles
          - [dbo].[sysdiagrams]
            - Candidate Key Profiles
            - Column Null Ratio Profiles

Column Null Ratio Profiles - [dbo].[StgStores] Encrypted Connection 1000 Rows

Column	Null Count	Null Percentage
hub_id	0	0.0000 %
store_id	0	0.0000 %
store_latitude	32	1.6824 %
store_longitude	32	1.6824 %
store_name	0	0.0000 %
store_plan_price	230	12.0925 %
store_segment	0	0.0000 %

Successfully loaded data profile from ...



Column Null Ratio Profiles - [dbo].[StgOrders] Encrypted Connection 1000 Rows

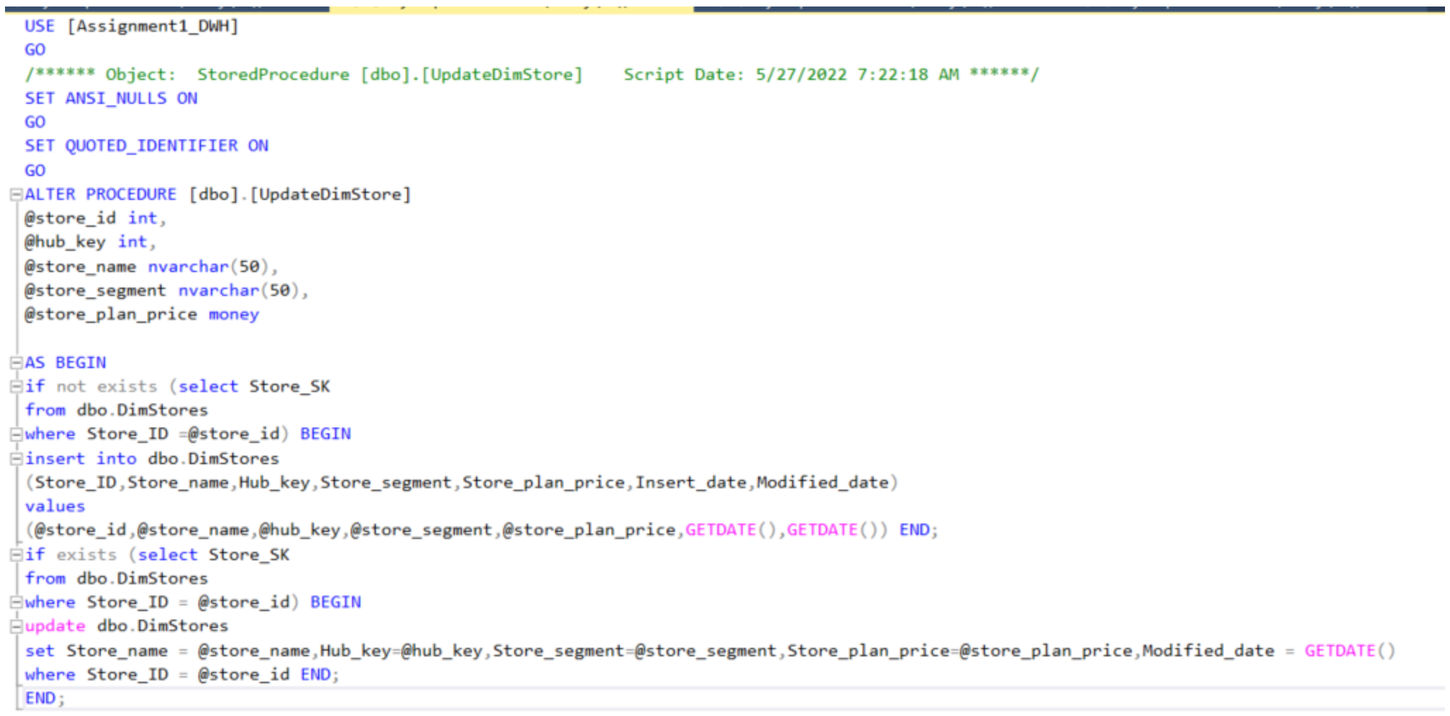
Column	Null Count	Null Percentage
channel_id	0	0.0000 %
delivery_order_id	0	0.0000 %
order_amount	0	0.0000 %
order_delivery_cost	488	0.8722 %
order_delivery_fee	0	0.0000 %
order_id	0	0.0000 %
order_status	0	0.0000 %
payment_order_id	0	0.0000 %
store_id	0	0.0000 %

- Only contain null values StgOrders and StgStores tables.

5.2.4. Data Extraction, Transformation, and Ingestion from Staging to Data warehouse

- In contrast to Step 01, this is the final step of the Datawarehouse solution; Step 03 will implement data transformation when extracting from the staging tables. In Step 0, the tables in the Datawarehouse layer were already designed, also the values for the "DimDate" table have been loaded.
- Before the design of control flow in SSIS, first a stored procedure was generated in the SQL server. The purpose of this, is to update the record that is already existing when data are extracted and add if it's completely a new record. The record will be identified via the surrogate key.

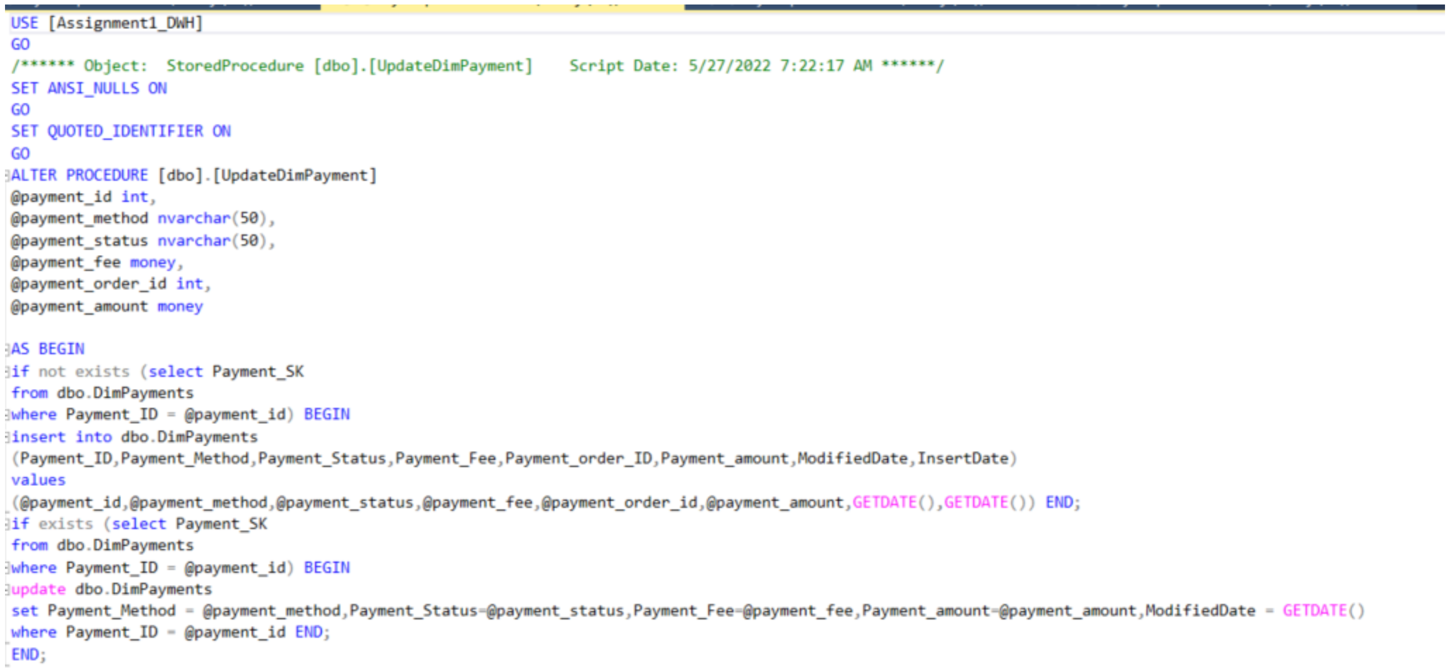
5.2.4.1. Stored Procedures



```
USE [Assignment1_DWH]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimStore]    Script Date: 5/27/2022 7:22:18 AM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimStore]
@store_id int,
@hub_key int,
@store_name nvarchar(50),
@store_segment nvarchar(50),
@store_plan_price money

AS BEGIN
if not exists (select Store_SK
from dbo.DimStores
where Store_ID =@store_id) BEGIN
insert into dbo.DimStores
(Store_ID,Store_name,Hub_key,Store_segment,Store_plan_price,Insert_date,Modified_date)
values
(@store_id,@store_name,@hub_key,@store_segment,@store_plan_price,GETDATE(),GETDATE()) END;
if exists (select Store_SK
from dbo.DimStores
where Store_ID = @store_id) BEGIN
update dbo.DimStores
set Store_name = @store_name,Hub_key=@hub_key,Store_segment=@store_segment,Store_plan_price=@store_plan_price,Modified_date = GETDATE()
where Store_ID = @store_id END;
END;
```

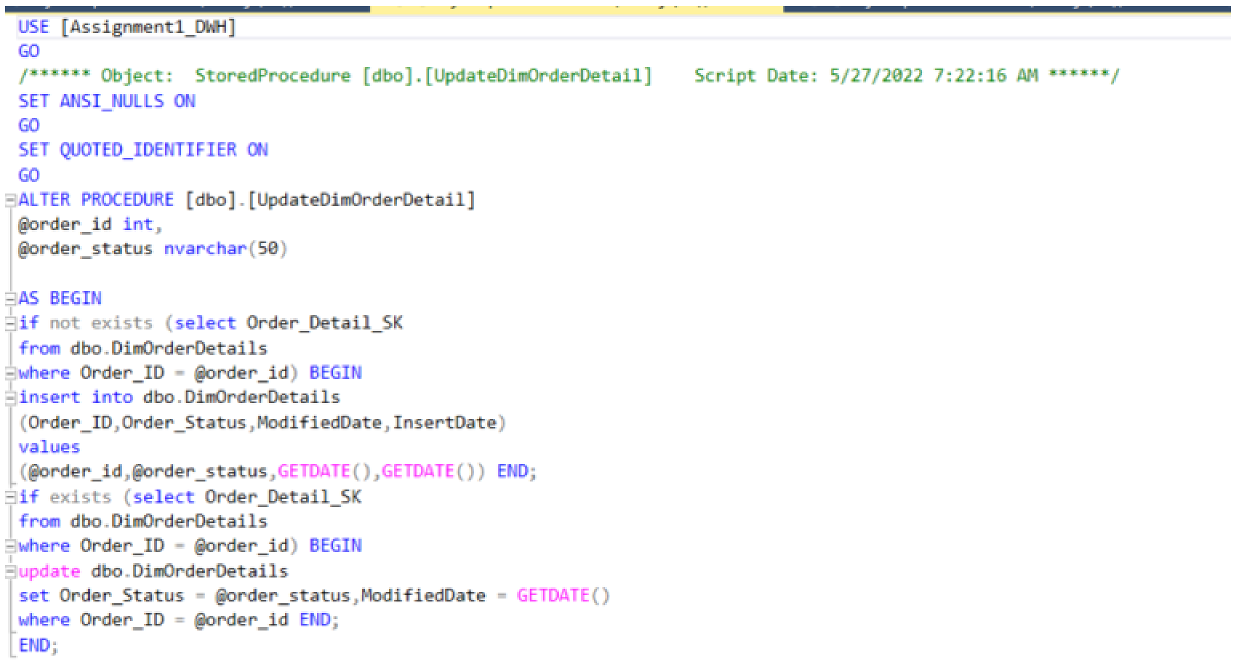
Figure 28 :DimStore Procedure



```
USE [Assignment1_DWH]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimPayment]    Script Date: 5/27/2022 7:22:17 AM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimPayment]
@payment_id int,
@payment_method nvarchar(50),
@payment_status nvarchar(50),
@payment_fee money,
@payment_order_id int,
@payment_amount money

AS BEGIN
if not exists (select Payment_SK
from dbo.DimPayments
where Payment_ID = @payment_id) BEGIN
insert into dbo.DimPayments
(Payment_ID,Payment_Method,Payment_Status,Payment_Fee,Payment_order_ID,Payment_amount,ModifiedDate,InsertDate)
values
(@payment_id,@payment_method,@payment_status,@payment_fee,@payment_order_id,@payment_amount,GETDATE(),GETDATE()) END;
if exists (select Payment_SK
from dbo.DimPayments
where Payment_ID = @payment_id) BEGIN
update dbo.DimPayments
set Payment_Method = @payment_method,Payment_Status=@payment_status,Payment_Fee=@payment_fee,Payment_amount=@payment_amount,ModifiedDate = GETDATE()
where Payment_ID = @payment_id END;
END;
```

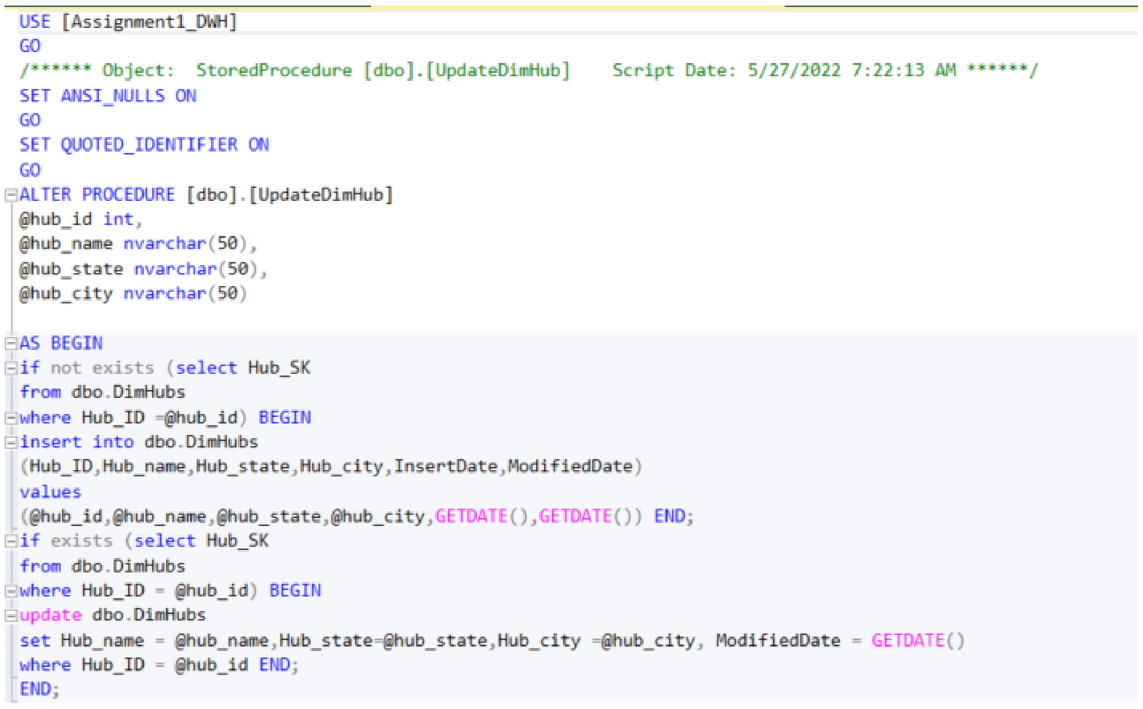
Figure 29 :DimPayment Procedure



```
USE [Assignment1_DWH]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimOrderDetail]    Script Date: 5/27/2022 7:22:16 AM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimOrderDetail]
@order_id int,
@order_status nvarchar(50)

AS BEGIN
if not exists (select Order_Detail_SK
from dbo.DimOrderDetails
where Order_ID = @order_id) BEGIN
insert into dbo.DimOrderDetails
(Order_ID,Order_Status,ModifiedDate,InsertDate)
values
(@order_id,@order_status,GETDATE(),GETDATE()) END;
if exists (select Order_Detail_SK
from dbo.DimOrderDetails
where Order_ID = @order_id) BEGIN
update dbo.DimOrderDetails
set Order_Status = @order_status,ModifiedDate = GETDATE()
where Order_ID = @order_id END;
END;
```

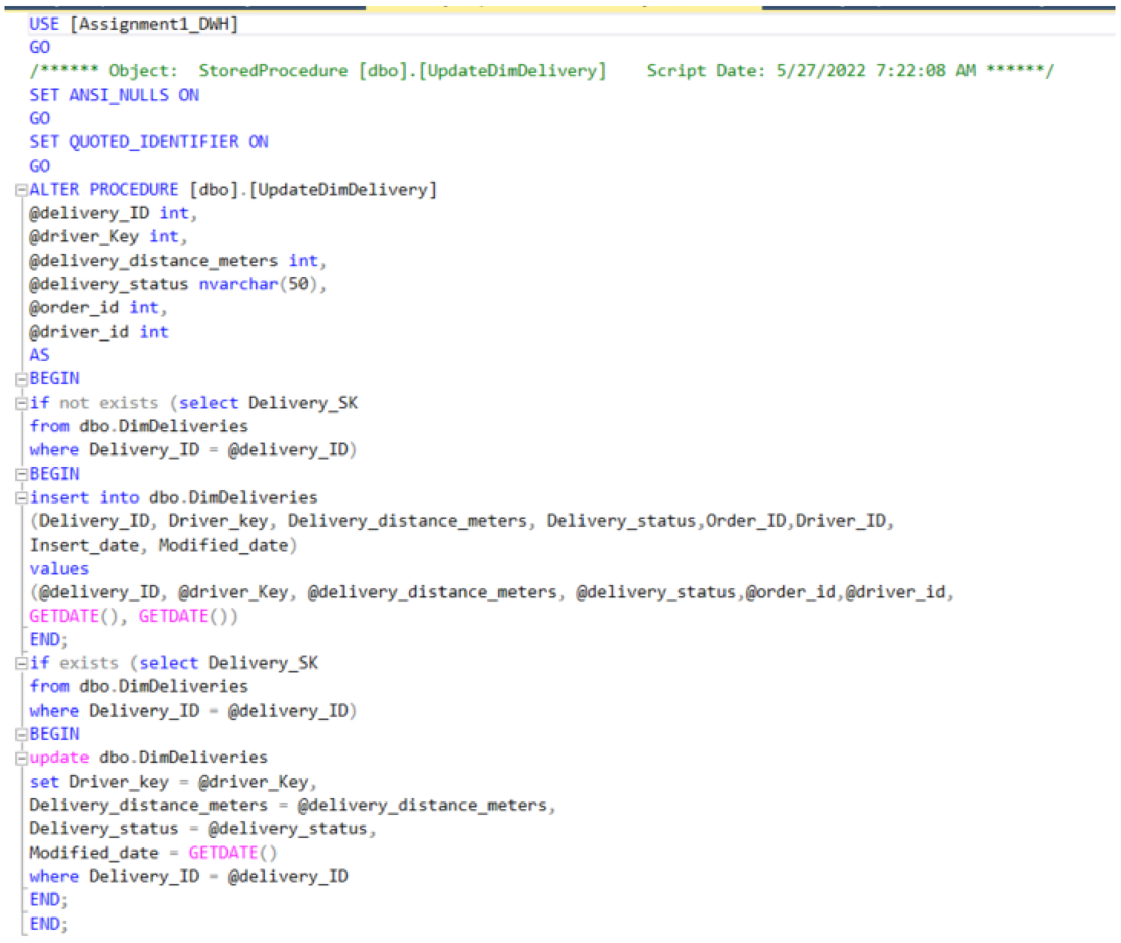
Figure 30 :DimOrderDetails Procedure



```sql
USE [Assignment1_DWH]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimHub]    Script Date: 5/27/2022 7:22:13 AM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimHub]
@hub_id int,
@hub_name nvarchar(50),
@hub_state nvarchar(50),
@hub_city nvarchar(50)

AS BEGIN
if not exists (select Hub_SK
from dbo.DimHubs
where Hub_ID =@hub_id) BEGIN
insert into dbo.DimHubs
(Hub_ID,Hub_name,Hub_state,Hub_city,InsertDate,ModifiedDate)
values
(@hub_id,@hub_name,@hub_state,@hub_city,GETDATE(),GETDATE()) END;
if exists (select Hub_SK
from dbo.DimHubs
where Hub_ID = @hub_id) BEGIN
update dbo.DimHubs
set Hub_name = @hub_name,Hub_state=@hub_state,Hub_city =@hub_city, ModifiedDate = GETDATE()
where Hub_ID = @hub_id END;
END;
```

Figure 31 :DimHubs Procedure

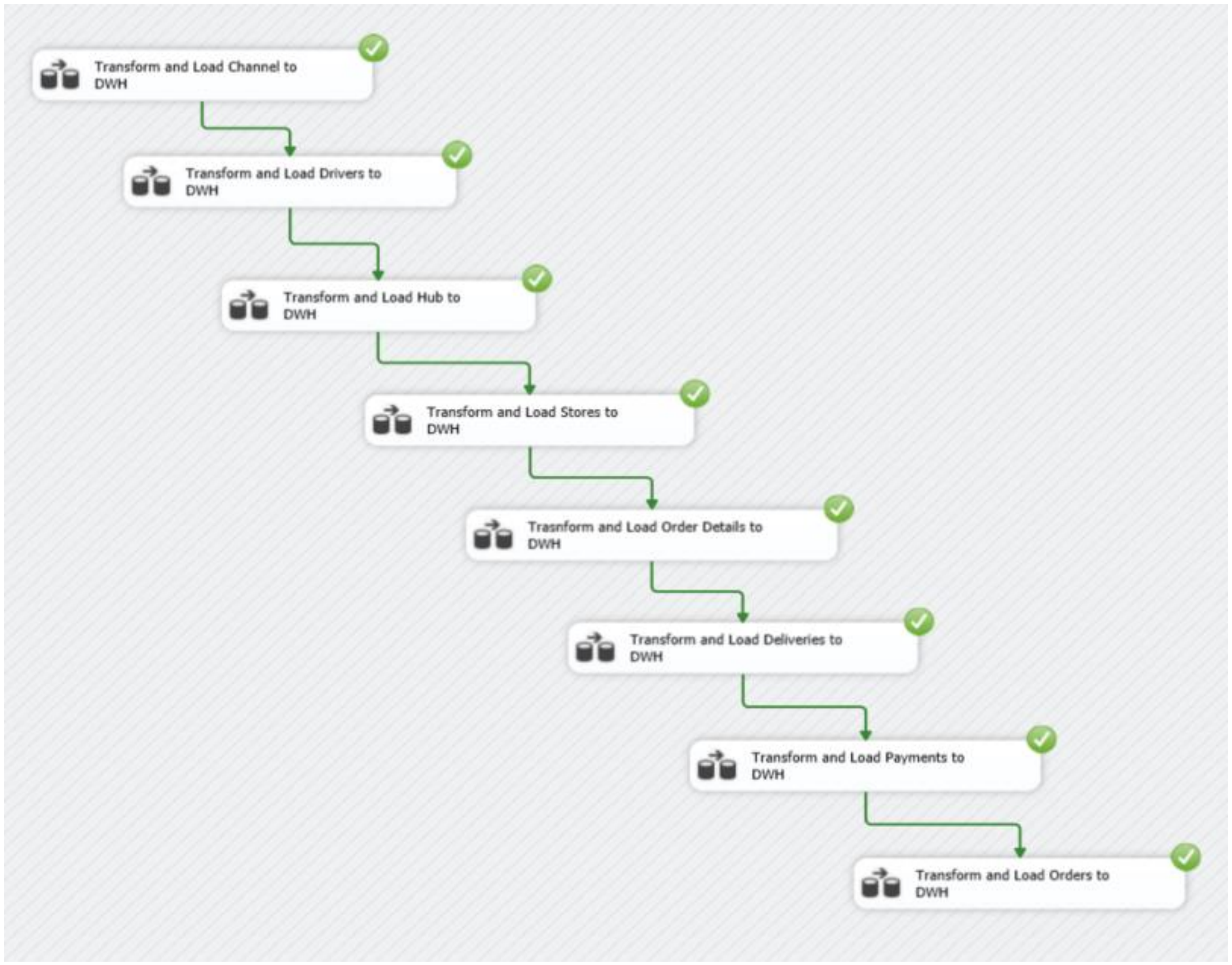


```sql
USE [Assignment1_DWH]
GO
/****** Object:  StoredProcedure [dbo].[UpdateDimDelivery]    Script Date: 5/27/2022 7:22:08 AM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdateDimDelivery]
@delivery_ID int,
@driver_Key int,
@delivery_distance_meters int,
@delivery_status nvarchar(50),
@order_id int,
@driver_id int
AS
BEGIN
if not exists (select Delivery_SK
from dbo.DimDeliveries
where Delivery_ID = @delivery_ID)
BEGIN
insert into dbo.DimDeliveries
(Delivery_ID, Driver_key, Delivery_distance_meters, Delivery_status,Order_ID,Driver_ID,
Insert_date, Modified_date)
values
(@delivery_ID, @driver_Key, @delivery_distance_meters, @delivery_status,@order_id,@driver_id,
GETDATE(), GETDATE())
END;
if exists (select Delivery_SK
from dbo.DimDeliveries
where Delivery_ID = @delivery_ID)
BEGIN
update dbo.DimDeliveries
set Driver_key = @driver_Key,
Delivery_distance_meters = @delivery_distance_meters,
Delivery_status = @delivery_status,
Modified_date = GETDATE()
where Delivery_ID = @delivery_ID
END;
END;
```

Figure 32 :DimDeliveries procedure

5.2.4.2. Data Flow

After the execution of the stored procedure, a new SSIS package "Assignment_Load_DWH" was created for this process in Data tools. Comparable to Step 01, control flows were created for each table extraction from staging layer, and within the control task the data flow was designed.


Transform and Load Channel to DWH
Transform and Load Drivers to DWH
Transform and Load Hub to DWH
Transform and Load Stores to DWH
Trasnform and Load Order Details to DWH
Transform and Load Deliveries to DWH
Transform and Load Payments to DWH
Transform and Load Orders to DWH

a) Transform and Load Channel to DWH

Data Conversion flow convert varchar to nvarchar and integer values.

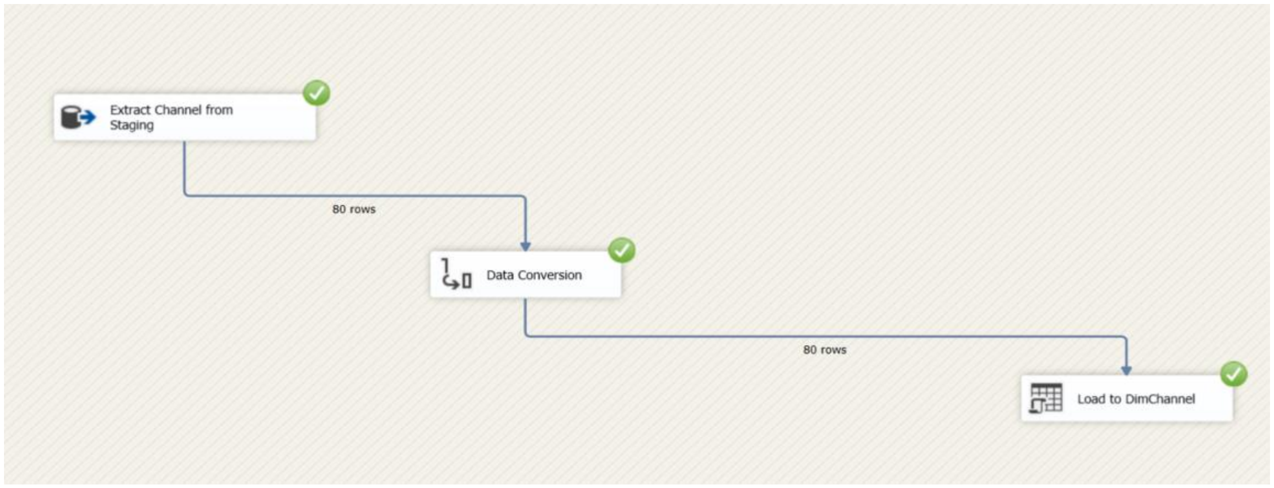
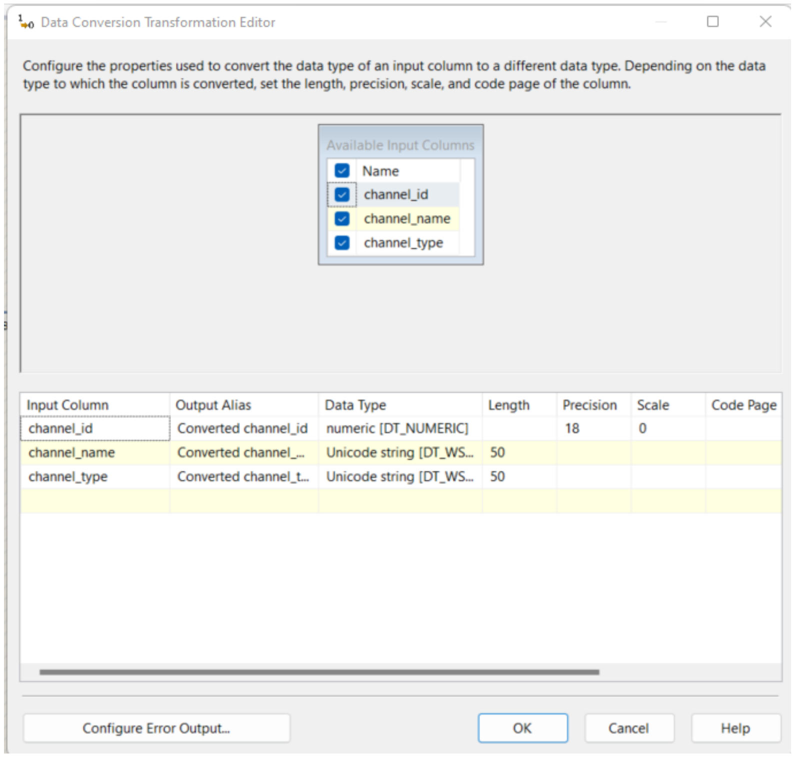


Figure 34 : DimChannel Data flow task


Data Conversion Transformation Editor
Configure Error Output...
OK
Cancel
Help

Configure the properties used to convert the data type of an input column to a different data type. Depending on the data type to which the column is converted, set the length, precision, scale, and code page of the column.

Available Input Columns

	Name
✓	channel_id
✓	channel_name
✓	channel_type

Input Column	Output Alias	Data Type	Length	Precision	Scale	Code Page
channel_id	Converted channel_id	numeric [DT_NUMERIC]		18	0	
channel_name	Converted channel_...	Unicode string [DT_WS...	50			
channel_type	Converted channel_t...	Unicode string [DT_WS...	50			

Figure 33 :DimChannel Data Conversion

b) Transform and Load Drivers to DWH

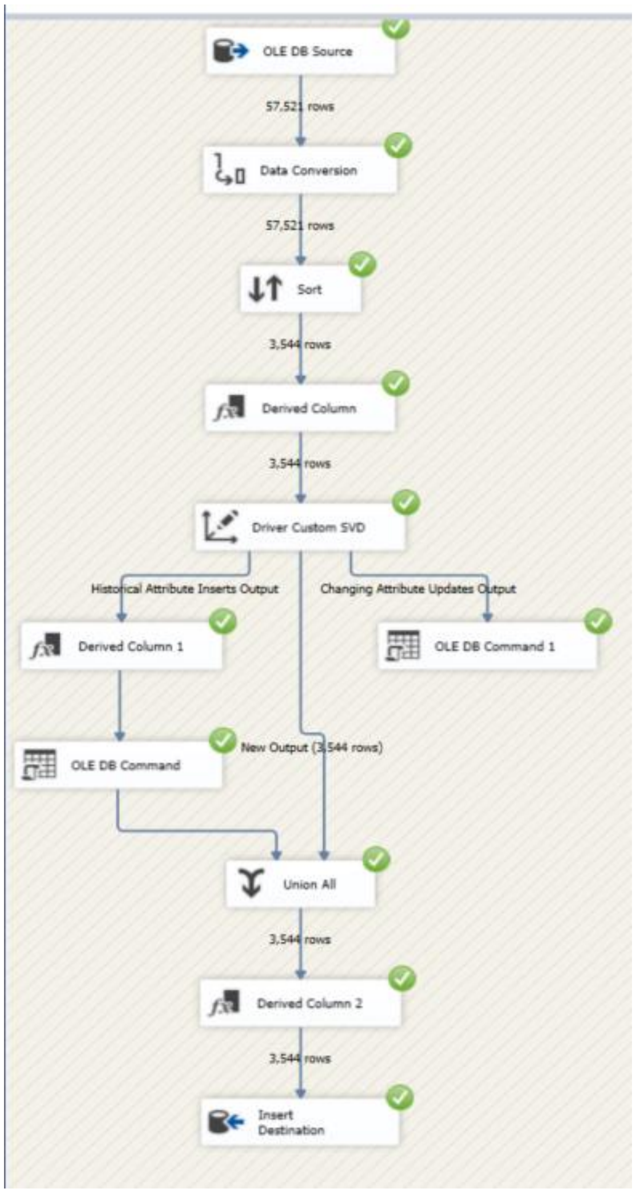


Figure 35 :DimDrivers SVD Data FLOW

c) Transform and Load Hub to DWH

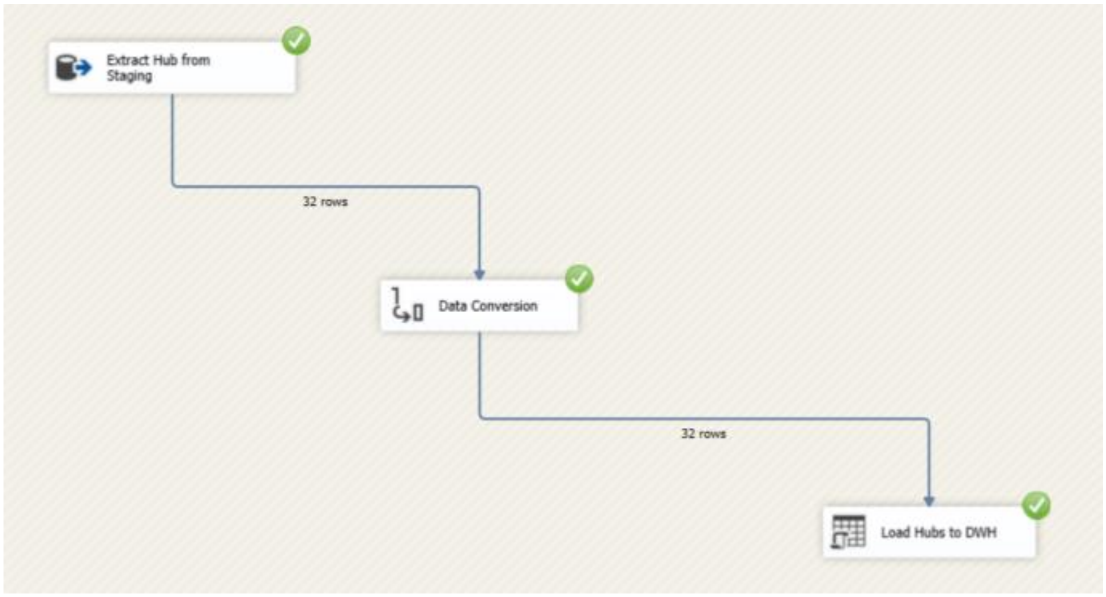


Figure 36:DimHub Data Flow Task

d) Transform and Load Stores to DWH

According to the data profile, additionally cleaned and replaced the null values to the “0” value in Store_plan_price

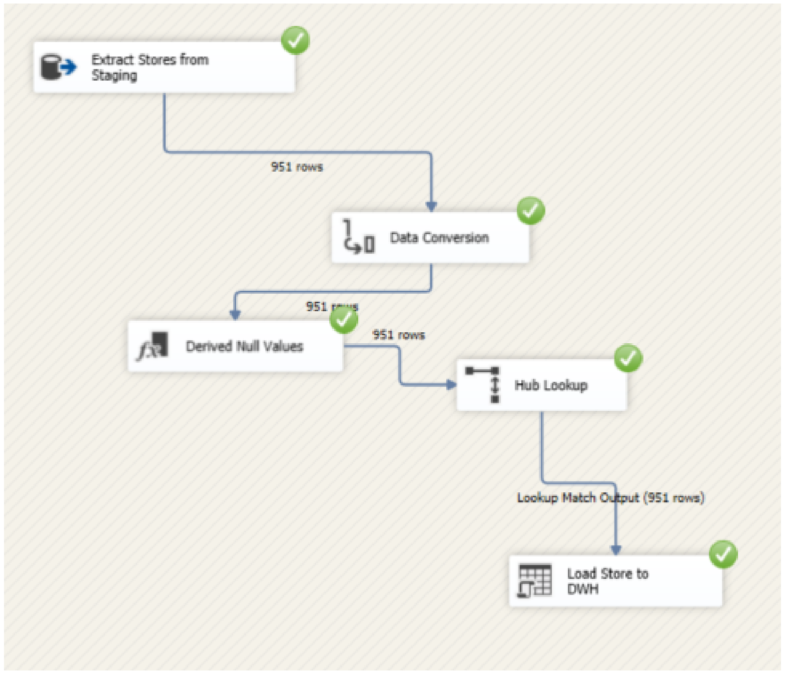
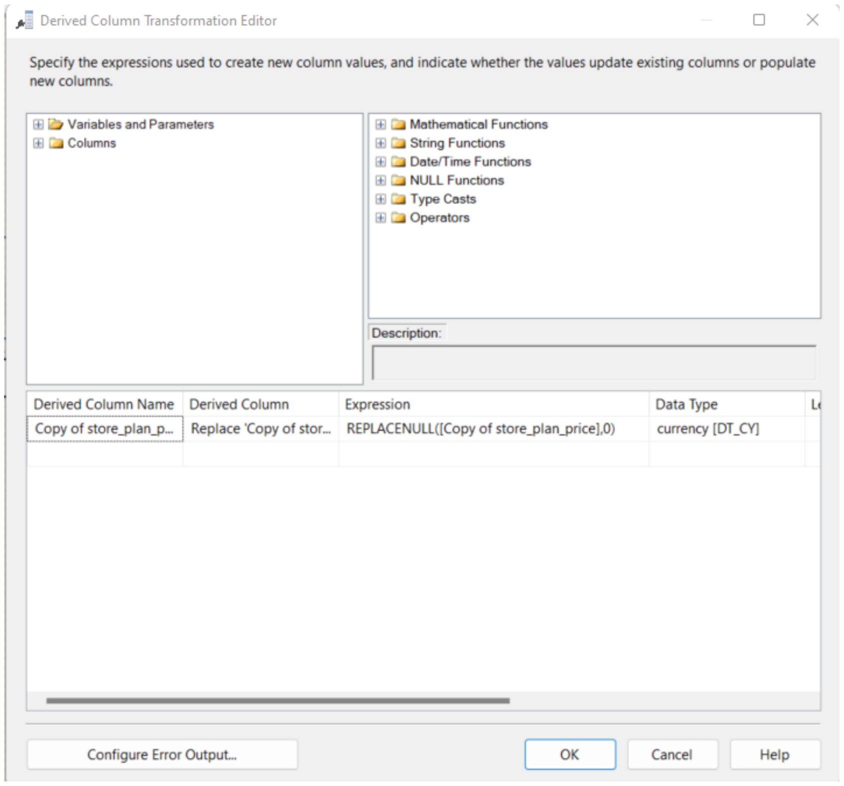


Figure 37 :Dim Stores Data flow task


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.

Derived Column Name	Derived Column	Expression	Data Type
Copy of store_plan_p...	Replace 'Copy of stor...	REPLACENULL([Copy of store_plan_price],0)	currency [DT_CY]

Figure 38 :Dim Stores Filling Nulls

e) Transform and Load Order Details to DWH

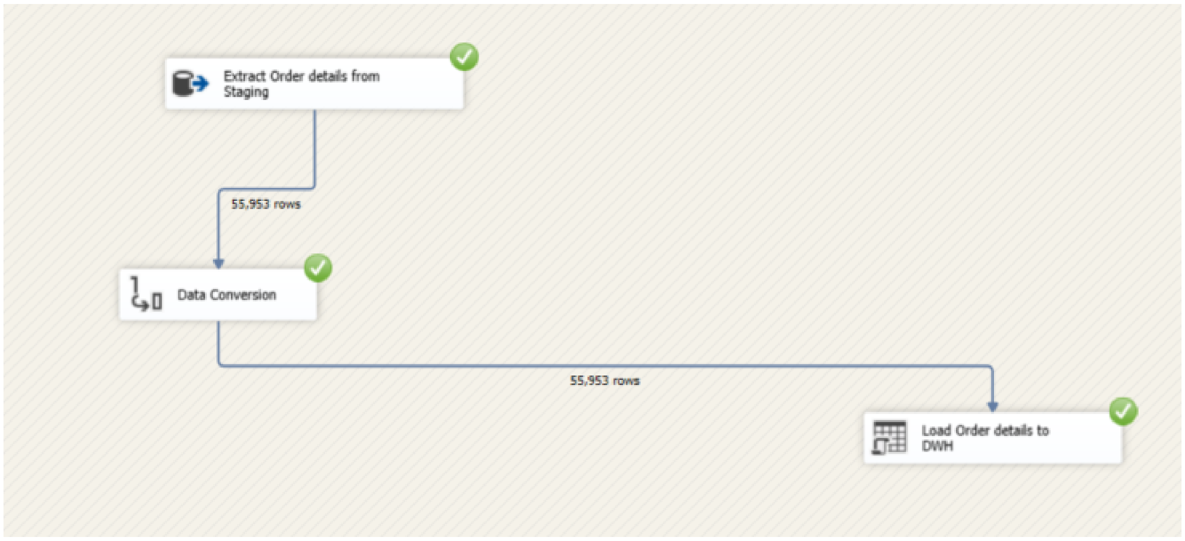


Figure 39 :Dim Order details data flow task

f) Transform and Load Deliveries to DWH

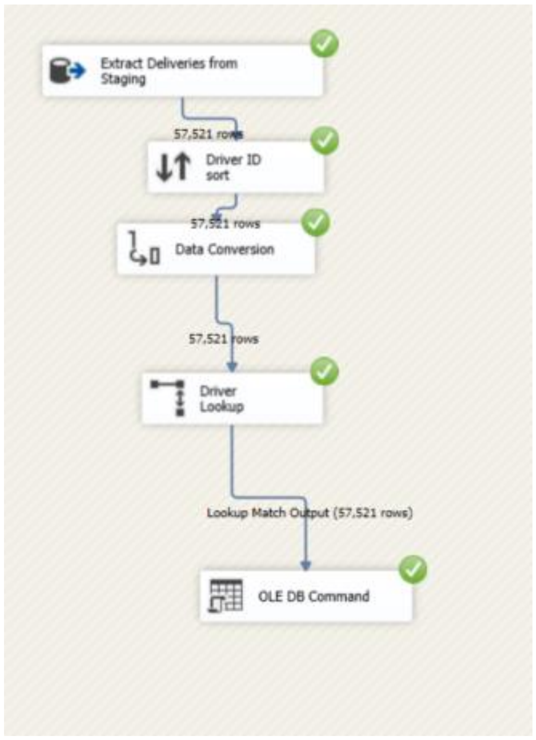


Figure 40 :Dim Deliveries Data flow task

g) Transform and Load Payments to DWH

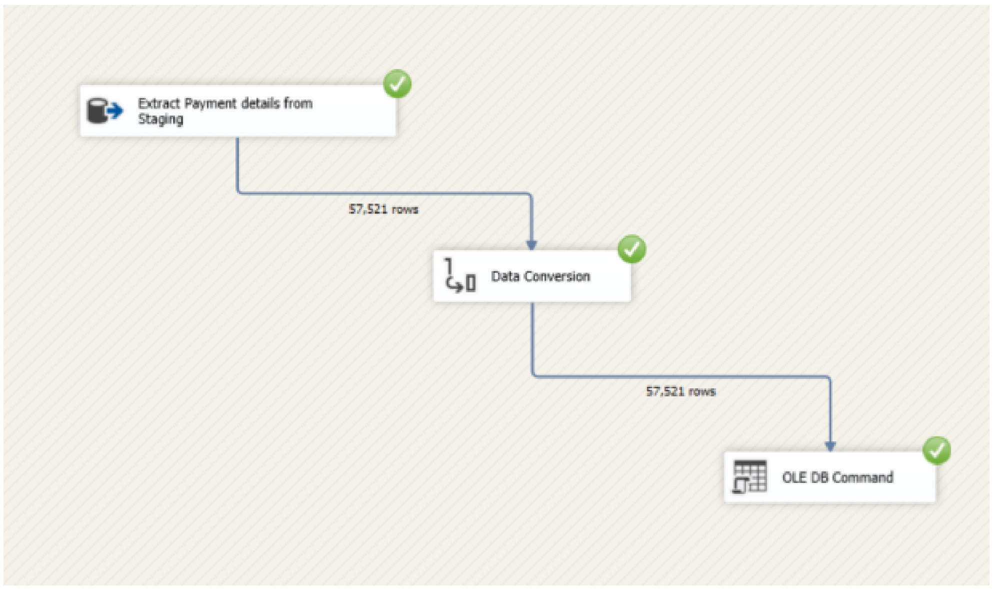


Figure 41 :Dim payments data flow task

h) Transform and Load Orders to DWH

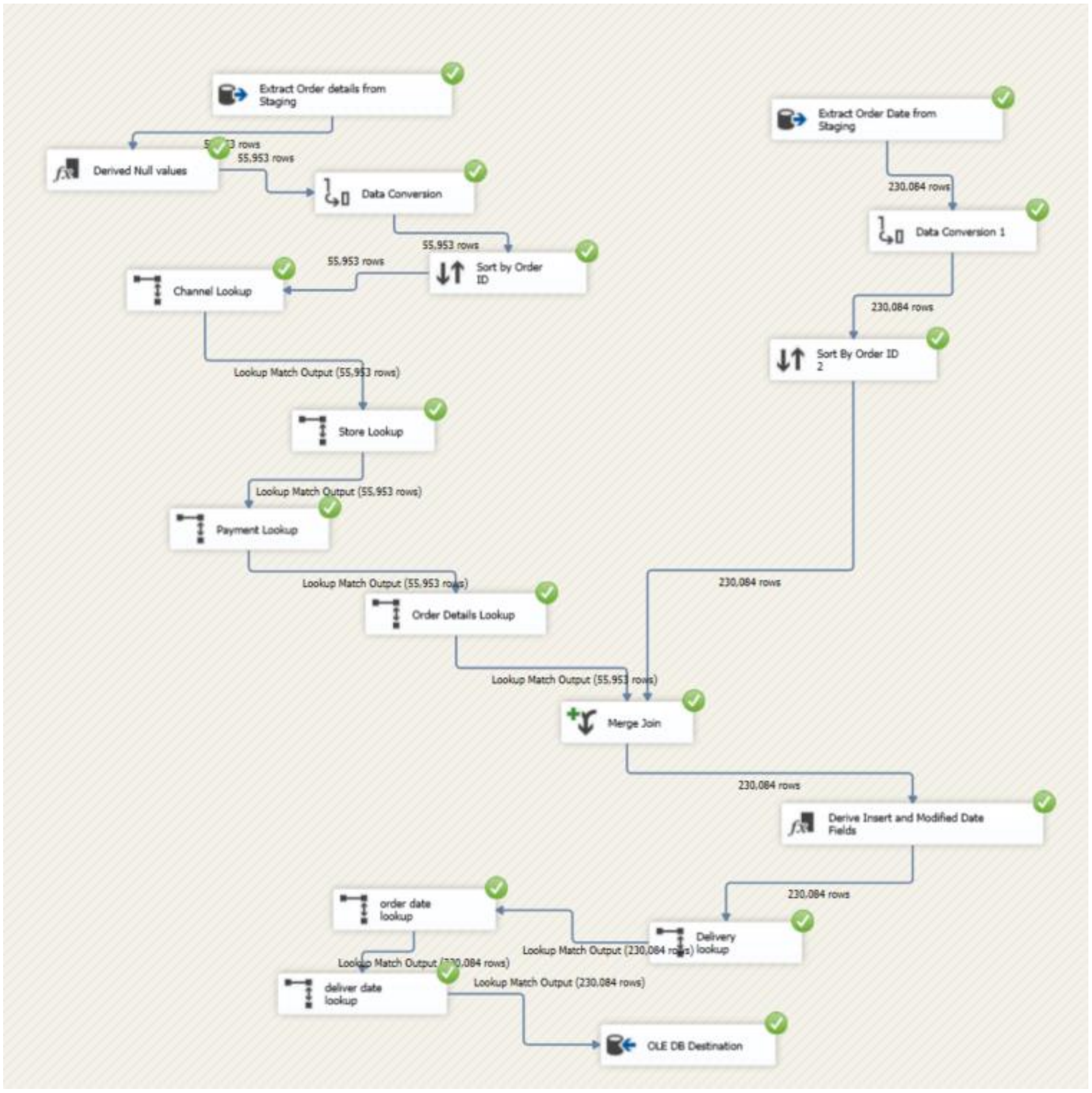
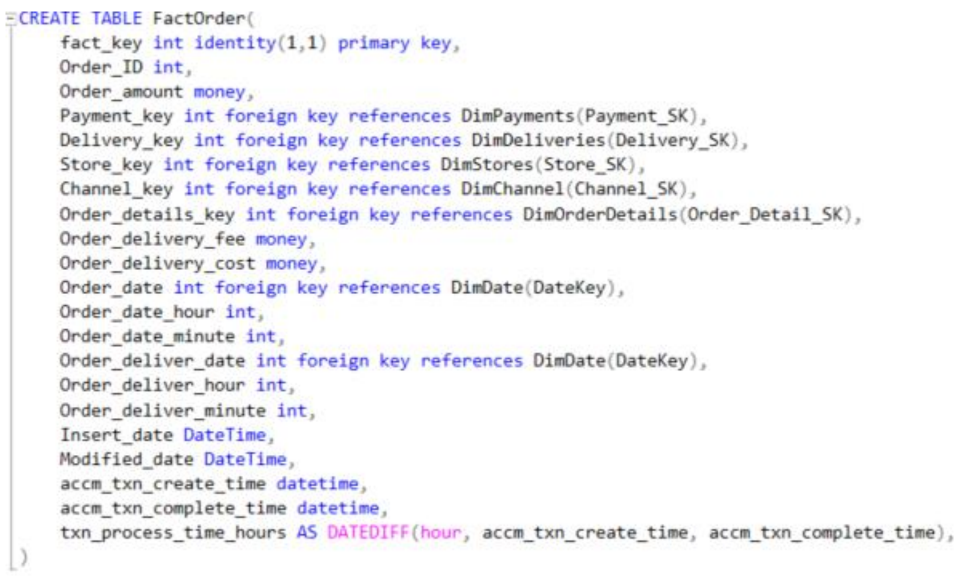

Extract Order details from Staging
Derived Null values
55,953 rows
55,953 rows
Data Conversion
55,953 rows
Sort by Order ID
55,953 rows
Channel Lookup
Lookup Match Output (55,953 rows)
Store Lookup
Lookup Match Output (55,953 rows)
Payment Lookup
Lookup Match Output (55,953 rows)
Order Details Lookup
Lookup Match Output (55,953 rows)
Extract Order Date from Staging
230,084 rows
Data Conversion 1
230,084 rows
Sort By Order ID 2
230,084 rows
Merge Join
230,084 rows
Derive Insert and Modified Date Fields
230,084 rows
Delivery lookup
Lookup Match Output (230,084 rows)
order date lookup
Lookup Match Output (230,084 rows)
deliver date lookup
Lookup Match Output (230,084 rows)
OLE DB Destination

Figure 42 :Fact Orders data flow task

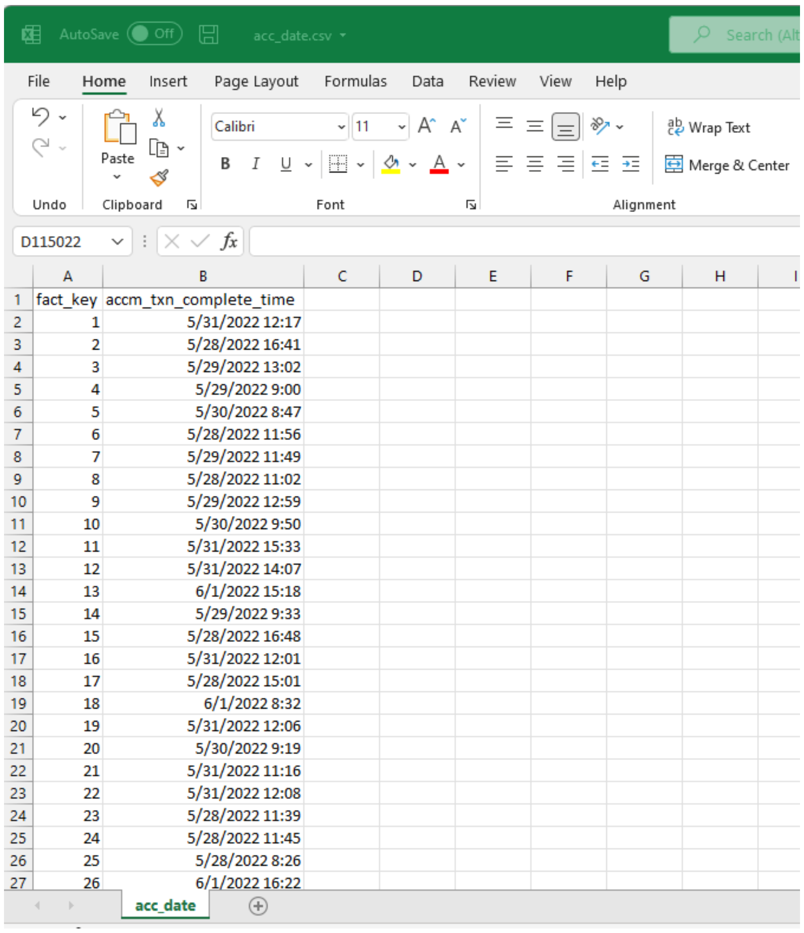
6. Accumulating fact tables

6.1. Extended Fact Table



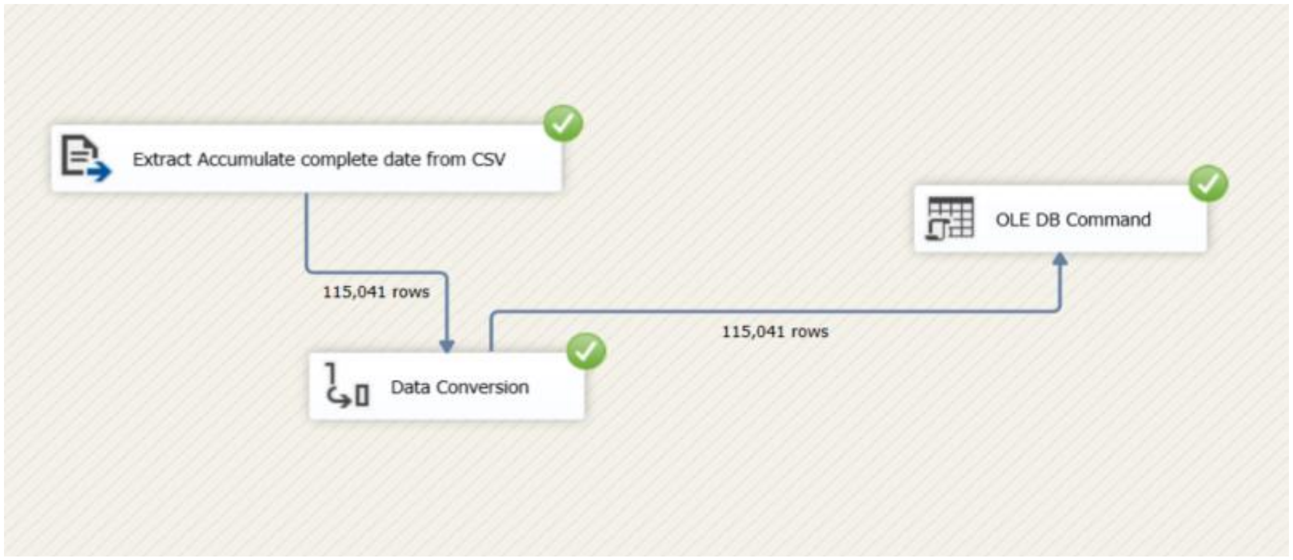
```
CREATE TABLE FactOrder(
    fact_key int identity(1,1) primary key,
    Order_ID int,
    Order_amount money,
    Payment_key int foreign key references DimPayments(Payment_SK),
    Delivery_key int foreign key references DimDeliveries(Delivery_SK),
    Store_key int foreign key references DimStores(Store_SK),
    Channel_key int foreign key references DimChannel(Channel_SK),
    Order_details_key int foreign key references DimOrderDetails(Order_Detail_SK),
    Order_delivery_fee money,
    Order_delivery_cost money,
    Order_date int foreign key references DimDate(DateKey),
    Order_date_hour int,
    Order_date_minute int,
    Order_deliver_date int foreign key references DimDate(DateKey),
    Order_deliver_hour int,
    Order_deliver_minute int,
    Insert_date DateTime,
    Modified_date DateTime,
    accm_txn_create_time datetime,
    accm_txn_complete_time datetime,
    txn_process_time_hours AS DATEDIFF(hour, accm_txn_create_time, accm_txn_complete_time),
)
```

6.2. Prepared Dataset

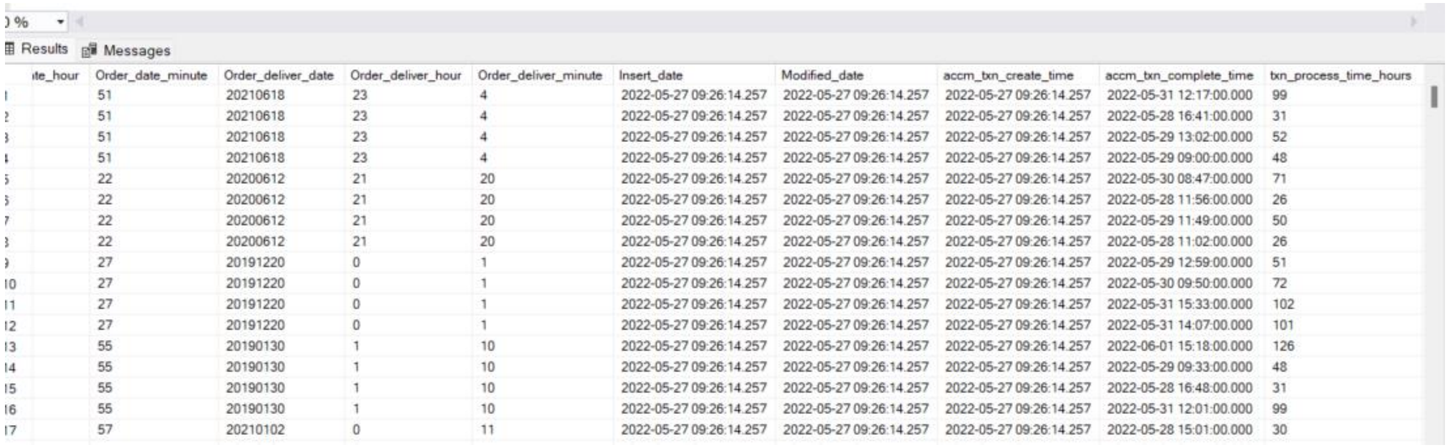


	A	B
1	fact_key	accm_txn_complete_time
2	1	5/31/2022 12:17
3	2	5/28/2022 16:41
4	3	5/29/2022 13:02
5	4	5/29/2022 9:00
6	5	5/30/2022 8:47
7	6	5/28/2022 11:56
8	7	5/29/2022 11:49
9	8	5/28/2022 11:02
10	9	5/29/2022 12:59
11	10	5/30/2022 9:50
12	11	5/31/2022 15:33
13	12	5/31/2022 14:07
14	13	6/1/2022 15:18
15	14	5/29/2022 9:33
16	15	5/28/2022 16:48
17	16	5/31/2022 12:01
18	17	5/28/2022 15:01
19	18	6/1/2022 8:32
20	19	5/31/2022 12:06
21	20	5/30/2022 9:19
22	21	5/31/2022 11:16
23	22	5/31/2022 12:08
24	23	5/28/2022 11:39
25	24	5/28/2022 11:45
26	25	5/28/2022 8:26
27	26	6/1/2022 16:22

6.3. ETL Pipeline



6.4. Result



	te_hour	Order_date_minute	Order_deliver_date	Order_deliver_hour	Order_deliver_minute	Insert_date	Modified_date	accm_txn_create_time	accm_txn_complete_time	txn_process_time_hours
1		51	20210618	23	4	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-31 12:17:00.000	99
2		51	20210618	23	4	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-28 16:41:00.000	31
3		51	20210618	23	4	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-29 13:02:00.000	52
4		51	20210618	23	4	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-29 09:00:00.000	48
5		22	20200612	21	20	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-30 08:47:00.000	71
6		22	20200612	21	20	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-28 11:56:00.000	26
7		22	20200612	21	20	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-29 11:49:00.000	50
8		22	20200612	21	20	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-28 11:02:00.000	26
9		27	20191220	0	1	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-29 12:59:00.000	51
10		27	20191220	0	1	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-30 09:50:00.000	72
11		27	20191220	0	1	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-31 15:33:00.000	102
12		27	20191220	0	1	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-31 14:07:00.000	101
13		55	20190130	1	10	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-06-01 15:18:00.000	126
14		55	20190130	1	10	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-29 09:33:00.000	48
15		55	20190130	1	10	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-28 16:48:00.000	31
16		55	20190130	1	10	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-31 12:01:00.000	99
17		57	20210102	0	11	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-27 09:26:14.257	2022-05-28 15:01:00.000	30