Sri Lanka Institute of Information Technology



DWBI – Assignment Report

Data warehousing and Business Intelligence-IT3021

**DWBI –** **Assignment 01**

Submitted by:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name with Initials** | **Registration Number** | **Contact Number** | **Email** |
| BANDARANAYAKE B.G.W.T.N. | IT22247018 | 0785622656 | it22247018@my.sliit.lk |

Contents

1. Data Set Selection ..................................................................................................... 03
2. Preparation of data sources......................................................................................... 11
3. Solution architecture..................................................................................................................12
4. Data warehouse design & development............................................................................................................... 17
5. ETL development............................................................................................................... 18
6. ETL development – Accumulating fact tables........................................................... 19

# **Data Set Selection**

# 

1. **Dataset Description – Olist E-commerce Transactions Dataset**

* The dataset represents a real-world online retail operation by Olist, an ecommerce platform in Brazil. It captures the full order lifecycle — from customer information, product listing, and seller data, to orders, payments, logistics, and geolocation making it a rich source for developing a data warehouse and business intelligence solution. The dataset spans from 2016 to 2018, providing over two years of historical data. This ensures sufficient temporal data for time-based analysis, seasonal trends, and hierarchical aggregations
* **Business Context:** Olist connects small businesses to customers through a marketplace platform. This dataset simulates how an online retailer collects and stores transactional information in a relational OLTP database, making it ideal for transformation into a dimensional model for data warehousing and analytics.
* **Business Logic**: **Impact of Shipping Cost on Product Profitability:**

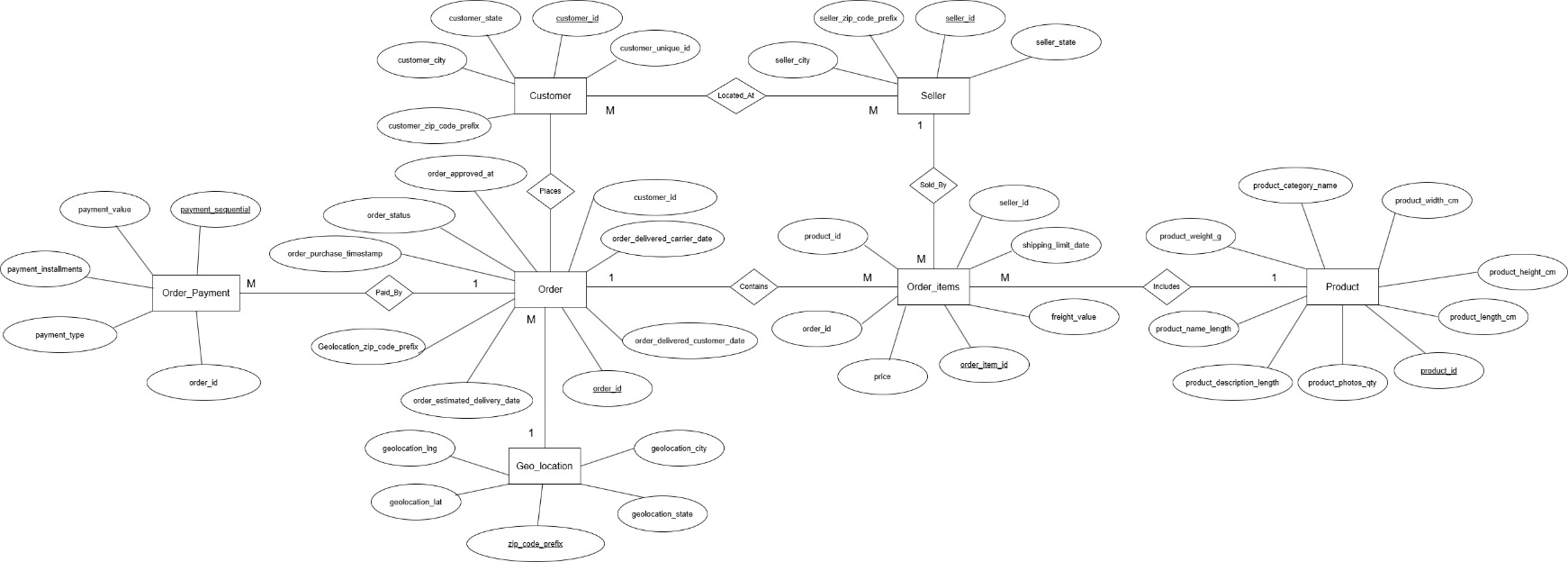
To measure the **net profit per product** by accounting for **shipping cost (FreightValue)** as a deduction from the product's sale price (ProductPrice), providing insight into how freight impacts the overall profitability of each sale.

.

1. **Entities and Attributes**

* Customers
  + customer\_id
  + customer\_unique\_id
  + customer\_zip\_code\_prefix
  + customer\_city
  + customer\_state
* Sellers
  + seller\_id
  + seller\_zip\_code\_prefix
  + seller\_city
  + seller\_state
* Orders
  + order\_id
  + customer\_id
  + order\_status
  + order\_purchase\_timestamp
  + order\_approved\_at
  + order\_delivered\_carrier\_date
  + order\_delivered\_customer\_date
  + order\_estimated\_delivery\_date
  + Geolocation\_zip\_code\_prefix
* Order Items
  + order\_id
  + order\_item\_id
  + product\_id
  + seller\_id
  + shipping\_limit\_date
  + price
  + freight\_value
* Order Payments
  + order\_id
  + payment\_sequential
  + payment\_type
  + payment\_installments
  + payment\_value
* Products
  + product\_id
  + product\_category\_name
  + product\_name\_length
  + product\_description\_length
  + product\_photos\_qty
  + product\_weight\_g
  + product\_length\_cm
  + product\_height\_cm
  + product\_width\_cm
* GeoLocation
  + zip\_code\_prefix
  + geolocation\_lat
  + geolocation\_lng
  + geolocation\_city
  + geolocation\_state

1. **ER Diagram**

****

https://drive.google.com/file/d/1AvwNlzZqPPt1ojdqOc-8vzzDkIKx1Lga/view?usp=sharing

# **Preparation of data sources**

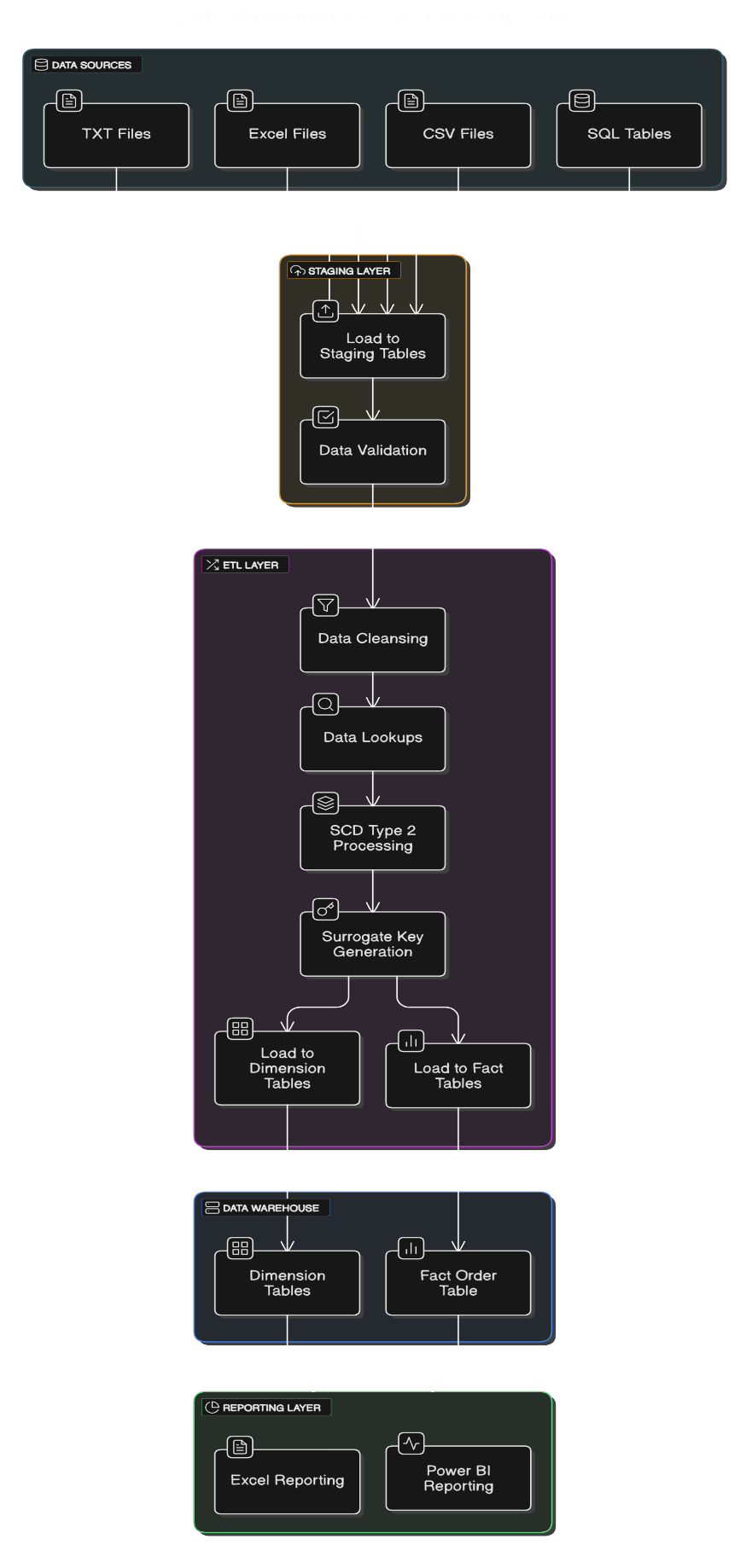
1. **Dataset Breakdown by Source Type**
   * + **Flat Files (CSV)**
       - olist\_orders\_dataset.csv
       - olist\_order\_items\_dataset.csv
       - olist\_order\_payments\_dataset.csv
       - olist\_products\_dataset.csv
       - olist\_geolocation\_dataset.csv
       - product\_category\_name\_translation.csv
     + **Flat Files (TXT)**
       - olist\_sellers\_dataset.txt
     + **Excel Worksheet**
       - olist\_customers\_dataset.xlsx
2. **Data Source Load To Staging DB Types Used:**

|  |  |  |
| --- | --- | --- |
| **File Name** | **Format** | **Content Description** |
| olist\_customers\_dataset.xlsx | Microsoft Excel Worksheet | Contains customer-level data, such as customer ID, city, state, and ZIP code prefix. |
| Geolocation Table(Olist SourseDB) | SQL DataBase | Provides latitude and longitude for ZIP code prefixes along with city/state information. |
| olist\_order\_items\_dataset.csv | CSV | Contains detailed order item data (product ID, seller ID, price, freight) |
| olist\_order\_payments\_dataset.csv | CSV | Includes payment types, installment details, and payment values per order |
| Orders Table(Olist SourseDB) | SQL DataBase | Captures order data including status and timestamps |
| olist\_products\_dataset.csv | CSV | Describes products, including category, dimensions, and weights. |
| olist\_sellers\_dataset.txt | Text (TXT) | Contains seller details including ZIP prefix, city, and state. Simulates export from vendor or merchant registry. |

1. **Why This Structure?**

* **Excel**: Used to simulate structured exports from customer-facing systems
* **CSV**: Represents system-generated exports from order and transaction management systems.
* **TXT**: Simulates raw exports from legacy or unstructured sources, like seller registration files.

# **3.** **Solution architecture**



1. **Overview**

* This solution is structured in a layered manner and mainly aims to use Data Warehouse (DW) and Business Intelligence (BI) platform to DI-ETL through the Olist dataset and insights to be discovered through the data since we are dealing with e-commerce transaction data. It supports scalable reporting, time based analysis and fast data querying for strategic business decisions.

1. **Architectural Components and Description**

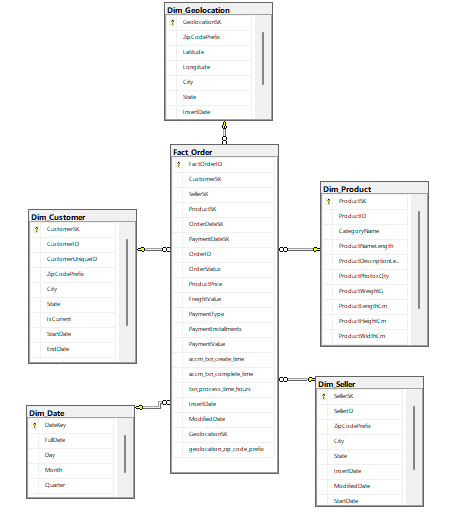
|  |  |
| --- | --- |
| **Component** | **Description** |
| Source Systems | Consist of structured files in CSV, Excel, and TXT format, simulating data exports from CRM, ERP, and transaction systems. |
| Staging Area | Temporary holding area where raw data is loaded and pre-processed |
| ETL Layer | Extracts data from various sources, applies transformation logic (data cleaning, enrichment, mapping), and loads it into the DW schema. |
| Data Warehouse | Central analytical repository modeled using a star schema. Contains fact and dimension tables optimized for OLAP and business reporting. |
| OLAP Layer | Constructs cubes for multidimensional analysis. Enables drill-downs, aggregations, and hierarchy navigation. |
| BI Tools (Power BI ) | Provides dashboards, reports, KPIs, and ad hoc analysis capabilities to end-users |

|  |  |
| --- | --- |

# **4.** **Data warehouse design & development**

**1.Dimensional Model Type: Star Schema**

We have implemented a star schema for the Olist e-commerce dataset, centered around a single fact table (Fact\_Order) and multiple dimension tables. This model enables efficient analytical queries and supports reporting across various business perspectives.



* **Fact Table: Fact\_Order:**

This table stores all transactional data related to customer orders, payments, shipping, and product-level metrics.(StgOrder,StgOrderItems,StgPayments)

* **Measures:**
* ProductPrice
* FreightValue
* PaymentInstallments
* PaymentValue
* **Foreign Keys**
* CustomerSK → Dim\_Customer
* SellerSK → Dim\_Seller
* ProductSK → Dim\_Product
* OrderDateKey and PaymentDateKey → Dim\_Date
* GeolocationSK → Dim\_Geolocation
* **Dimensions Used:**
  + - 1. **Dim\_Seller(SCD Type 2)**
  + SellerID (Business Key)
  + ZipCodePrefix
  + City
  + State
  + InsertDate/ModifiedDate
  + StartDate/EndDate
    - 1. **Dim\_Customers(SCD Type 2)**
  + CustomerID (Business Key)
  + ZipCodePrefix
  + City
  + State
  + IsCurrent
  + InsertDate/ModifiedDate
  + StartDate/EndDate
    - 1. **Dim\_Geolocation**
  + ZipCodePrefix (Business Key)
  + Latitude / Longitude
  + City / State
  + InsertDate/ModifiedDate
    - 1. **Dim\_Product**
  + ProductID (Business Key)
  + CategoryName
  + ProductNameLength
  + ProductDescriptionLength
  + ProductWeightG
  + ProductLengthCm / HeightCm / WidthCm
  + InsertDate/ModifiedDate
    - 1. **Dim\_Date**
  + DateKey
  + FullDate
  + Day / Month / Quarter / Year
* **Slowly Changing Dimensions:**

In this data warehouse design, we implemented both Dim\_Customer and Dim\_Seller as Slowly Changing Dimensions (SCD Type 2) to preserve historical changes in customer and seller location data.

We chose SCD Type 2 because the following attributes are considered historical and may change over time:

* City
* State
* ZipCodePrefix

Each time one of these attributes changes for a given CustomerID or SellerID, a new version of the record is inserted into the dimension table. We also track:

* InsertDate: when this version was added
* ModifiedDate: when changes occurred.
* StartDate: beginning of this version's validity,
* EndDate: when this version expired.
* IsCurrent: a flag indicating whether the record is the latest version (1 = active, 0 = old)

This approach enables accurate reporting on customer and seller activity over time, even if they have changed locations.

* **Design Assumptions**
* Seller and customer locations can change over time and are tracked using SCD.
* Date dimension is prepopulated to cover order and payment dates.
* Payment is handled at the order level, not at item level.
* Fact\_Order at the center.
* Dimension tables arranged around it with foreign key relationships.
* **Dimension Table SQL Definitions**
  + Create Dimension Tables

-- Customer Dimension

CREATE TABLE Dim\_Customer (

CustomerSK INT PRIMARY KEY IDENTITY(1,1), -- Surrogate Key

CustomerID VARCHAR(50), -- Business (Natural) Key

CustomerUniqueID VARCHAR(50),

ZipCodePrefix VARCHAR(50),

City VARCHAR(50),

State CHAR(2),

IsCurrent BIT DEFAULT 1,

StartDate DATETIME DEFAULT GETDATE(),

EndDate DATETIME,

InsertDate DATETIME DEFAULT GETDATE(),

ModifiedDate DATETIME

);

-- Seller Dimension

CREATE TABLE Dim\_Seller (

SellerSK INT PRIMARY KEY IDENTITY(1,1),

SellerID VARCHAR(50), -- Business Key

ZipCodePrefix VARCHAR(50),

City VARCHAR(50),

State CHAR(2),

InsertDate DATETIME DEFAULT GETDATE(),

ModifiedDate DATETIME,

StartDate DATETIME DEFAULT GETDATE(),

EndDate DATETIME,

);

-- Product Dimension

CREATE TABLE Dim\_Product (

ProductSK INT PRIMARY KEY IDENTITY(1,1),

ProductID VARCHAR(50), -- Business Key

CategoryName VARCHAR(100),

ProductNameLength INT,

ProductDescriptionLength INT,

ProductPhotosQty INT,

ProductWeightG INT,

ProductLengthCm INT,

ProductHeightCm INT,

ProductWidthCm INT,

InsertDate DATETIME DEFAULT GETDATE(),

ModifiedDate DATETIME

);

-- Geolocation Dimension

CREATE TABLE Dim\_Geolocation ( GeolocationSK INT PRIMARY KEY IDENTITY(1,1),

ZipCodePrefix VARCHAR(50), Latitude VARCHAR(50),

Longitude VARCHAR(50),

City VARCHAR(50),

State VARCHAR(50),

InsertDate DATETIME DEFAULT GETDATE(),

ModifiedDate DATETIME );

-- Date Dimension (you will need to create Date dimension separately usually)

CREATE TABLE Dim\_Date (

DateKey INT PRIMARY KEY, -- Format: YYYYMMDD

FullDate DATE,

Day INT,

Month INT,

Quarter INT,

Year INT,

InsertDate DATETIME DEFAULT GETDATE()

);

-- Fact Table: Orders / Sales

CREATE TABLE Fact\_Order (

FactOrderID INT PRIMARY KEY IDENTITY(1,1), -- Surrogate Key

CustomerSK INT,

SellerSK INT,

ProductSK INT,

OrderDateSK INT, -- Surrogate Key from Dim\_Date

PaymentDateSK INT,

GeolocationSK INT, -- Surrogate Key from Dim\_Date

OrderID VARCHAR(50), -- Business Key from Source

OrderStatus VARCHAR(20),

ProductPrice DECIMAL(10,2),

FreightValue DECIMAL(10,2),

PaymentType VARCHAR(50),

PaymentInstallments INT,

PaymentValue DECIMAL(10,2),

accm\_txn\_create\_time DATETIME,

accm\_txn\_complete\_time DATETIME,

txn\_process\_time\_hours DECIMAL(10,2),

InsertDate DATETIME DEFAULT GETDATE(),

ModifiedDate DATETIME,

-- Foreign Key Constraints

FOREIGN KEY (CustomerSK) REFERENCES Dim\_Customer(CustomerSK),

FOREIGN KEY (SellerSK) REFERENCES Dim\_Seller(SellerSK),

FOREIGN KEY (ProductSK) REFERENCES Dim\_Product(ProductSK),

FOREIGN KEY (OrderDateSK) REFERENCES Dim\_Date(DateKey),

FOREIGN KEY (PaymentDateSK) REFERENCES Dim\_Date(DateKey),

FOREIGN KEY (GeolocationSK) REFERENCES Dim\_Geolocation(GeolocationSK);

);

-- Date Dimension Sql query for generate data

DECLARE @StartDate DATE = '2010-01-01';

DECLARE @EndDate DATE = '2030-12-31';

WHILE @StartDate <= @EndDate

BEGIN

INSERT INTO Dim\_Date

(

DateKey,

FullDate,

Day,

Month,

Quarter,

Year,

InsertDate

)

VALUES

(

CONVERT(INT, FORMAT(@StartDate, 'yyyyMMdd')),

@StartDate,

DAY(@StartDate),

MONTH(@StartDate),

DATEPART(QUARTER, @StartDate),

YEAR(@StartDate),

GETDATE()

);

SET @StartDate = DATEADD(DAY, 1, @StartDate);

END;

* Stored Procedure: UpdateDimProduct

CREATE OR ALTER PROCEDURE dbo.UpdateDimProduct

@ProductID VARCHAR(50),

@CategoryName VARCHAR(100),

@ProductNameLength INT,

@ProductDescriptionLength INT,

@ProductPhotosQty INT,

@ProductWeightG INT,

@ProductLengthCm INT,

@ProductHeightCm INT,

@ProductWidthCm INT

AS

BEGIN

SET NOCOUNT ON;

-- Check if Product exists

IF NOT EXISTS (

SELECT 1

FROM dbo.Dim\_Product

WHERE ProductID = @ProductID

)

BEGIN

-- Insert new record

INSERT INTO dbo.Dim\_Product (

ProductID,

CategoryName,

ProductNameLength,

ProductDescriptionLength,

ProductPhotosQty,

ProductWeightG,

ProductLengthCm,

ProductHeightCm,

ProductWidthCm,

InsertDate,

ModifiedDate

)

VALUES (

@ProductID,

@CategoryName,

@ProductNameLength,

@ProductDescriptionLength,

@ProductPhotosQty,

@ProductWeightG,

@ProductLengthCm,

@ProductHeightCm,

@ProductWidthCm,

GETDATE(), -- InsertDate

NULL -- ModifiedDate (new insert, no modification yet)

);

END

ELSE

BEGIN

-- Only update if any actual data changed

IF EXISTS (

SELECT 1

FROM dbo.Dim\_Product

WHERE ProductID = @ProductID

AND (

ISNULL(CategoryName, '') <> ISNULL(@CategoryName, '') OR

ISNULL(ProductNameLength, -1) <> ISNULL(@ProductNameLength, -1) OR

ISNULL(ProductDescriptionLength, -1) <> ISNULL(@ProductDescriptionLength, -1) OR

ISNULL(ProductPhotosQty, -1) <> ISNULL(@ProductPhotosQty, -1) OR

ISNULL(ProductWeightG, -1) <> ISNULL(@ProductWeightG, -1) OR

ISNULL(ProductLengthCm, -1) <> ISNULL(@ProductLengthCm, -1) OR

ISNULL(ProductHeightCm, -1) <> ISNULL(@ProductHeightCm, -1) OR

ISNULL(ProductWidthCm, -1) <> ISNULL(@ProductWidthCm, -1)

)

)

BEGIN

-- Update record

UPDATE dbo.Dim\_Product

SET

CategoryName = @CategoryName,

ProductNameLength = @ProductNameLength,

ProductDescriptionLength = @ProductDescriptionLength,

ProductPhotosQty = @ProductPhotosQty,

ProductWeightG = @ProductWeightG,

ProductLengthCm = @ProductLengthCm,

ProductHeightCm = @ProductHeightCm,

ProductWidthCm = @ProductWidthCm,

ModifiedDate = GETDATE()

WHERE ProductID = @ProductID;

END

-- else ➔ do nothing

END

END

* Stored Procedure: UpdateDimGeoLocation

CREATE OR ALTER PROCEDURE dbo.UpdateDimGeolocation

@ZipCodePrefix VARCHAR(50),

@Latitude VARCHAR(50),

@Longitude VARCHAR(50),

@City VARCHAR(50),

@State VARCHAR(50)

AS

BEGIN

SET NOCOUNT ON;

IF NOT EXISTS (

SELECT GeolocationSK

FROM dbo.Dim\_Geolocation

WHERE ZipCodePrefix = @ZipCodePrefix

)

BEGIN

INSERT INTO dbo.Dim\_Geolocation (

ZipCodePrefix,

Latitude,

Longitude,

City,

State,

InsertDate,

ModifiedDate

)

VALUES (

@ZipCodePrefix,

@Latitude,

@Longitude,

@City,

@State,

GETDATE(),

GETDATE()

);

END;

IF EXISTS (

SELECT GeolocationSK

FROM dbo.Dim\_Geolocation

WHERE ZipCodePrefix = @ZipCodePrefix

)

BEGIN

UPDATE dbo.Dim\_Geolocation

SET Latitude = @Latitude,

Longitude = @Longitude,

City = @City,

State = @State,

ModifiedDate = GETDATE()

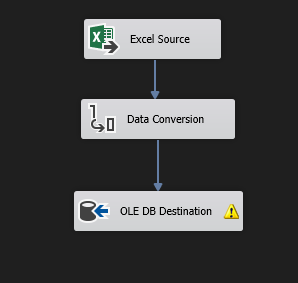
WHERE ZipCodePrefix = @ZipCodePrefix;

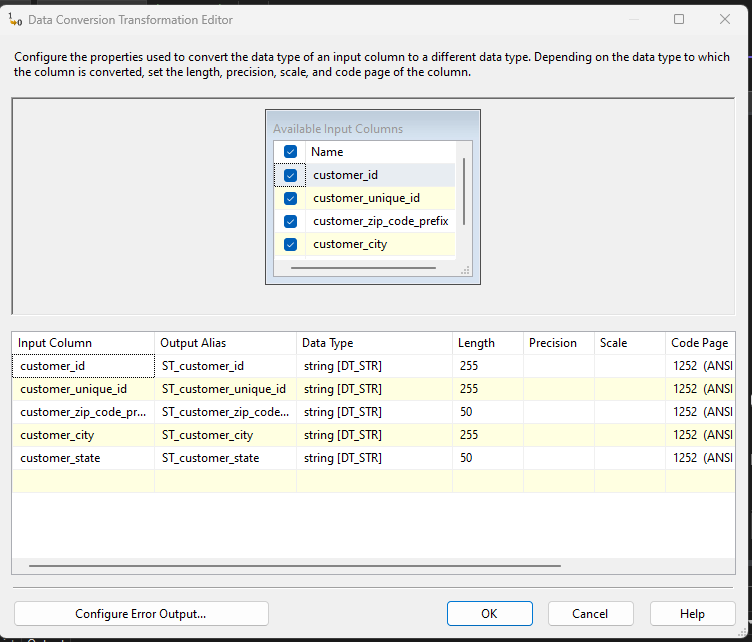
END

END;

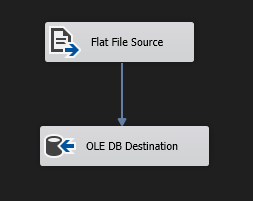
# **5.** **ETL development**

* **ETL WorkFlow Overview**
  1. **Staging Phase**
     + Data is loaded into matching staging tables using OLE DB Destination
     + This phase prepares the data for transformation and validation
* **Stg\_Customers-Extract Customer Data To Staging**

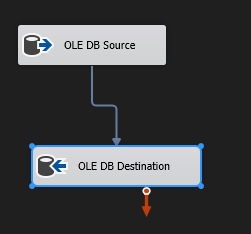




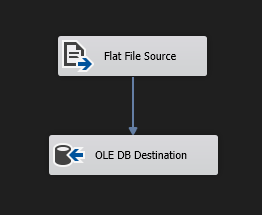
* **Stg\_Sellers- Extract Seller Data To Staging**



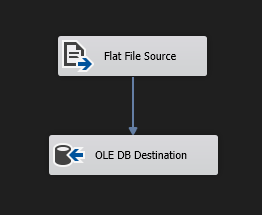
* Stg\_Orders-Extract Customer Data To Staging



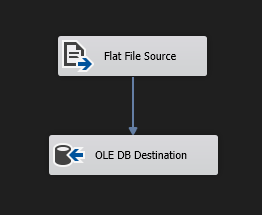
* Stg\_Products-Extract Product Data To Staging



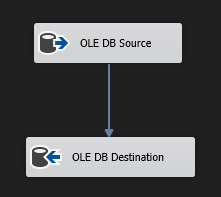
* Stg\_OrderItems-Extract OrderItem Data To Staging



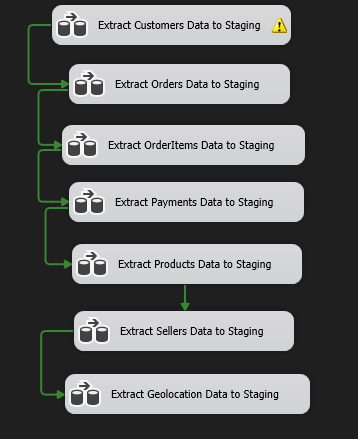
* Stg\_Payments-Extract Payments Data To Staging



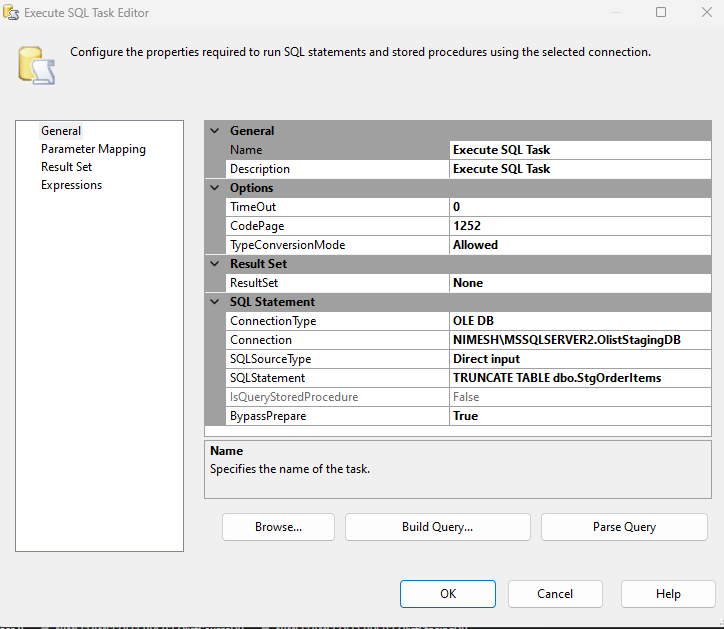
* Stg\_Geolocation Extract Locations Data To Staging

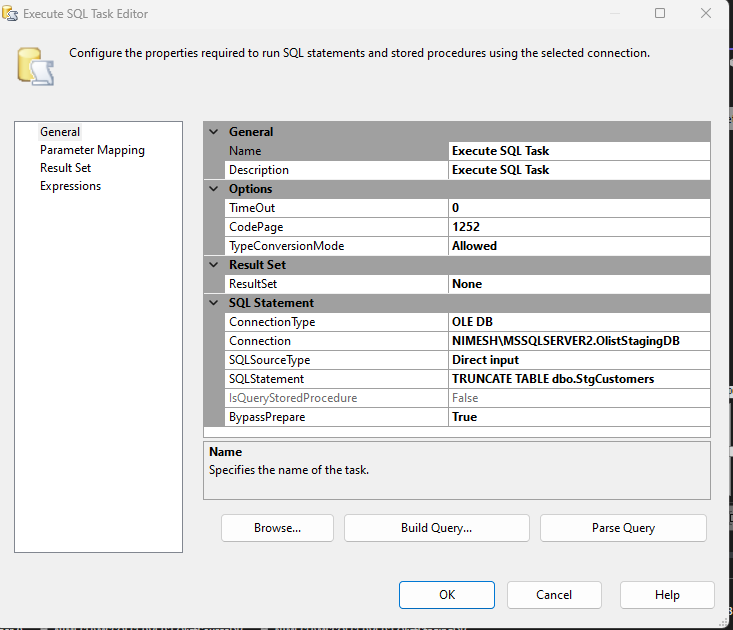


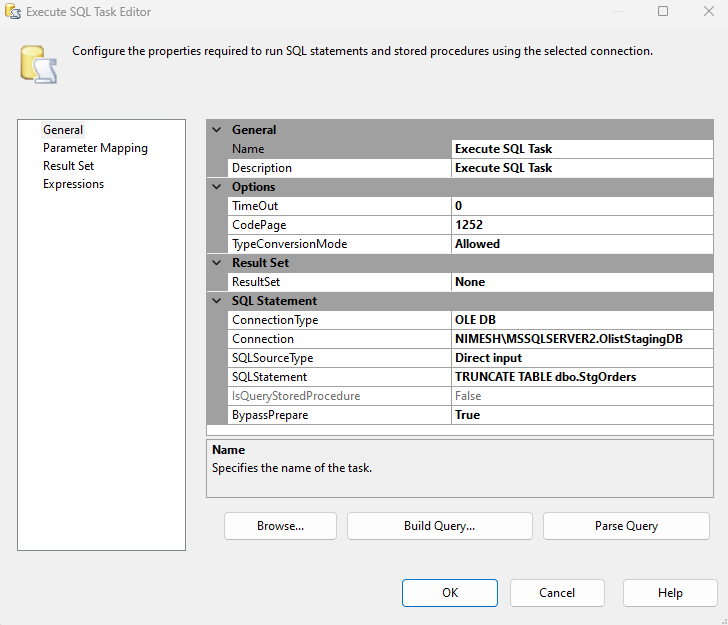
* Staging WorkFlow -Extract All Data To Staging

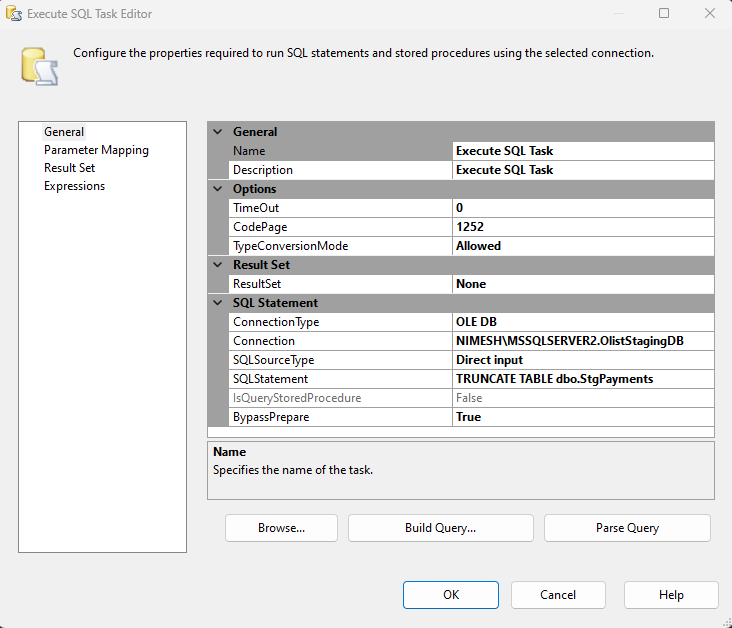


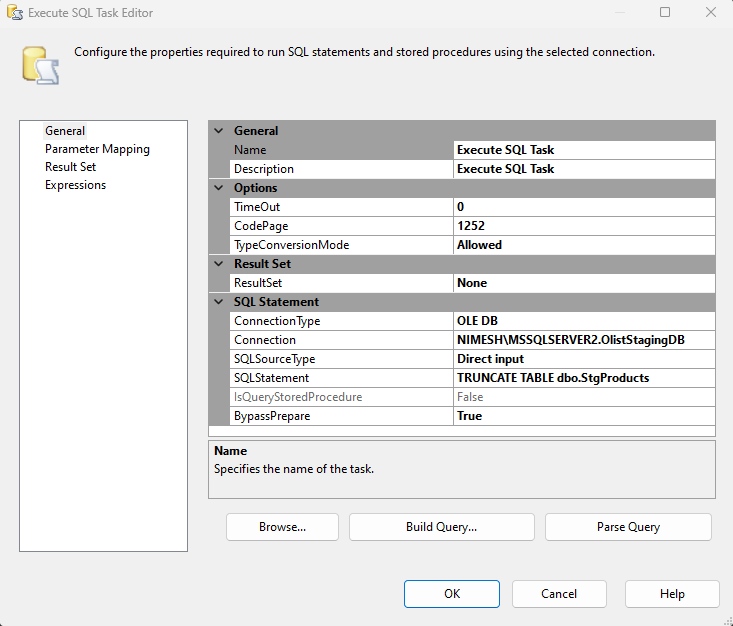
* **Truncating Staging Tables**
* **TRUNCATE and LOAD strategy** was applied.  
  This means the staging tables were **cleared (truncated) before each load**, ensuring no residual or duplicate data remained from previous ETL runs. This simplifies data handling and ensures consistency.



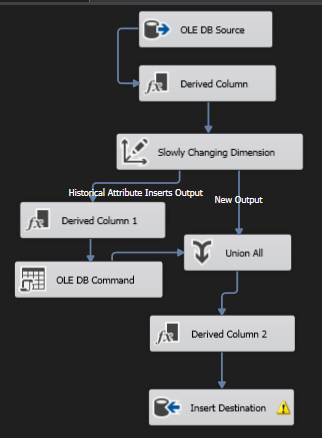




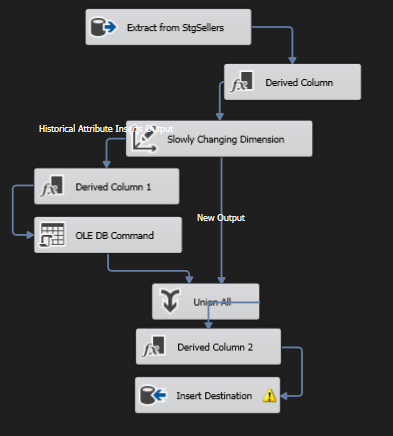




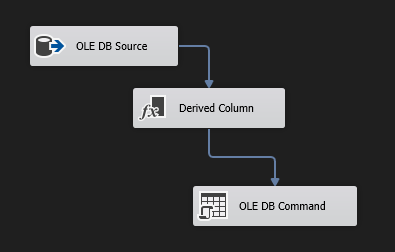
* 1. **Transformation Phase**
* Apply data cleaning: remove nulls, trim spaces, convert formats
* Use Derived Column transforms to compute new fields
* Use Lookup transforms to fetch surrogate keys for dimensions
* Apply Slowly Changing Dimension logic on Dim\_Customer and Dim\_Seller
* **Dim\_Customers- Transform and Load Dim\_Customers**

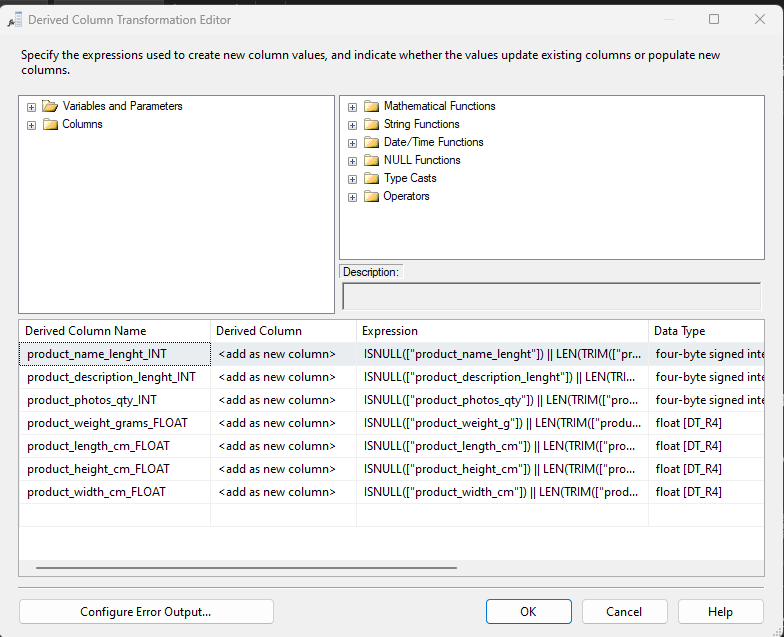


* **Dim\_Seller- Transform and Load Dim\_Seller**

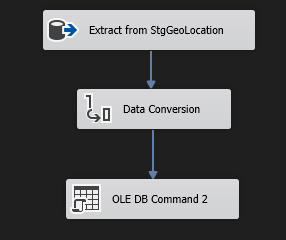


* **Dim\_Product- Transform and Load Dim\_Product**

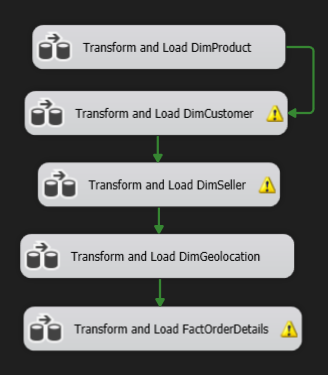


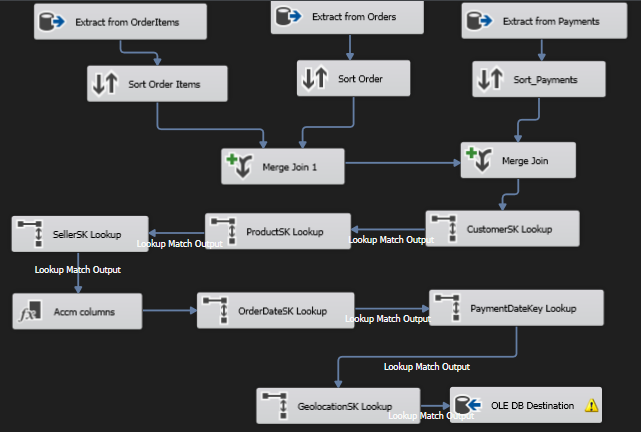


* **Dim\_Geolocation - Transform and Load Dim\_Geolocation**



* 1. **Loading Phase**
     + Load **Fact\_Order** using resolved surrogate keys
     + Ensure all referential integrity is preserved
     + Load dimension tables



* **Fact\_Order** **- Transform and Load Fact\_Order**

The Fact\_Order table serves as the central fact table in the Olist Data Warehouse, capturing transactional information related to customer orders, product prices, shipping costs, payment details, and timestamps for accumulating fact logic. The SSIS data flow for Fact\_Order is designed to integrate and transform data from multiple dimension lookups and compute profit-related metrics before loading into the warehouse.

Source Data Extraction

* Data is sourced from a prepared staging that consolidates key fields: OrderID, ProductID, SellerID, CustomerID, ProductPrice, FreightValue, PaymentType, PaymentInstallments, PaymentValue, OrderDate, and PaymentDate.

Lookup Transformations

* Lookup: Dim\_Product → returns ProductSK using ProductID
* Lookup: Dim\_Seller → returns SellerSK using SellerID
* Lookup: Dim\_Customer → returns CustomerSK using CustomerID
* Lookup: Dim\_Date →for OrderDateSK and PaymentDateSK using full date
* Lookup: Dim\_Geolocation → for ZipCodePrefix

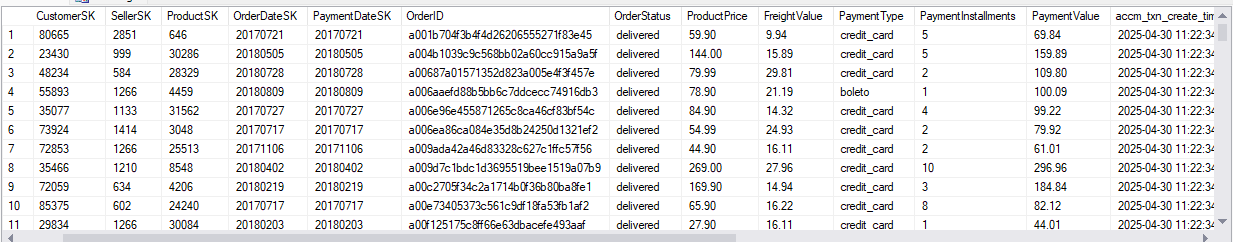
Derived Columns

* Adds accm\_txn\_create\_time using -GETDATE()
* Calculates initial profit- [ProductPrice] - [FreightValue]

OLE DB Destination

* Final output is loaded into Fact\_Order with all surrogate keys and measures.

Fact\_Order Table



# **6.** **ETL development – Accumulating fact tables**

### **Extended Fact Table Columns**

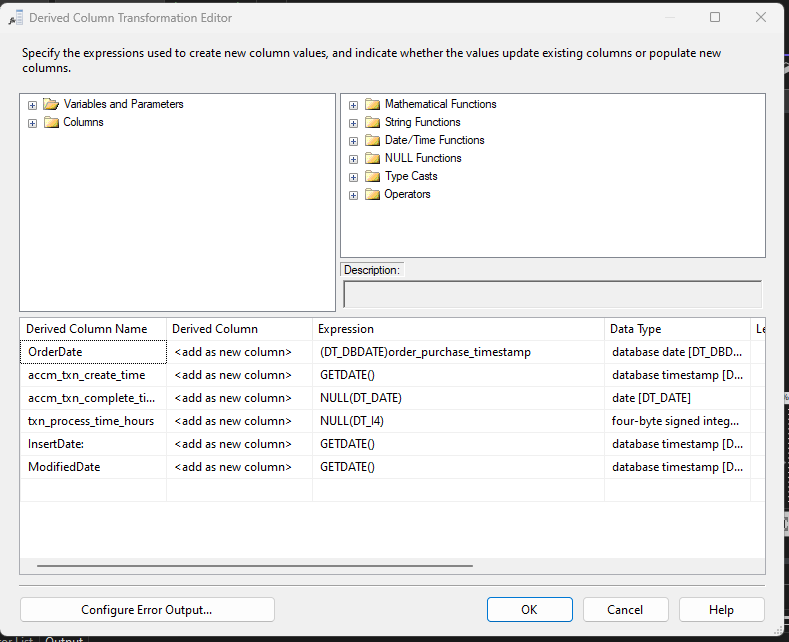
The following columns were added to Fact\_Order:

* accm\_txn\_create\_time DATETIME
* accm\_txn\_complete\_time DATETIME NULL
* txn\_process\_time\_hours INT NULL
* **Initial Load-Create Time**

During the initial data load via SSIS, the accm\_txn\_create\_time is populated with the current system date and time (GETDATE()). This simulates the moment when the transaction event occurred.

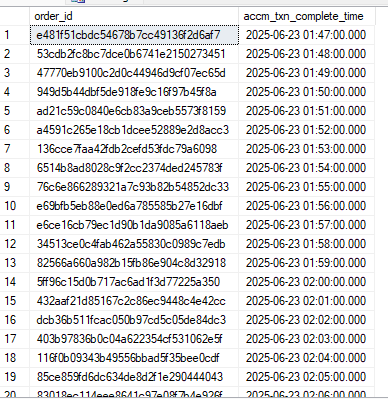
In the Data Flow for Fact\_Order:

* + Add a Derived Column transformation
  + Create a new columns:
    - accm\_txn\_create\_time = GETDATE()
    - accm\_txn\_complete\_time = NULL(DT\_DATE)
    - accm\_txn\_process\_hours = NULL(DT\_14)

****

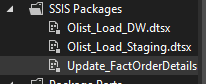
### **Completion Time Data Source**

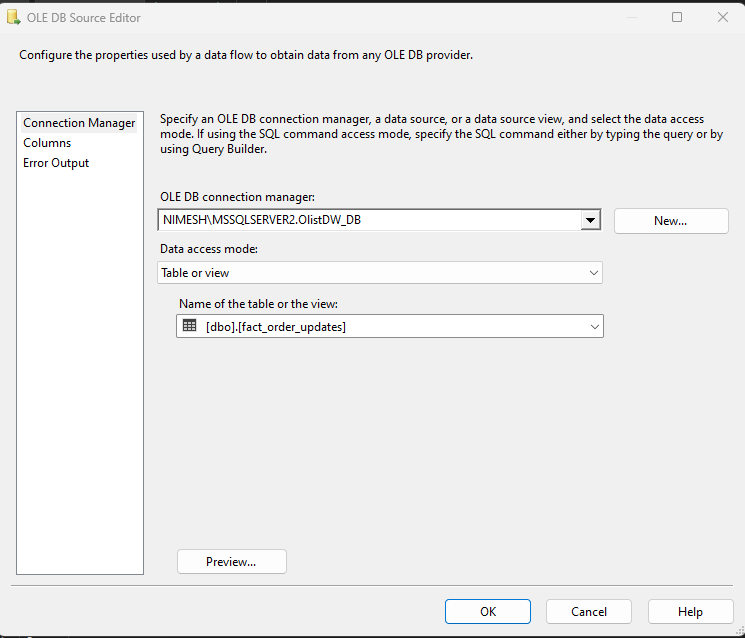
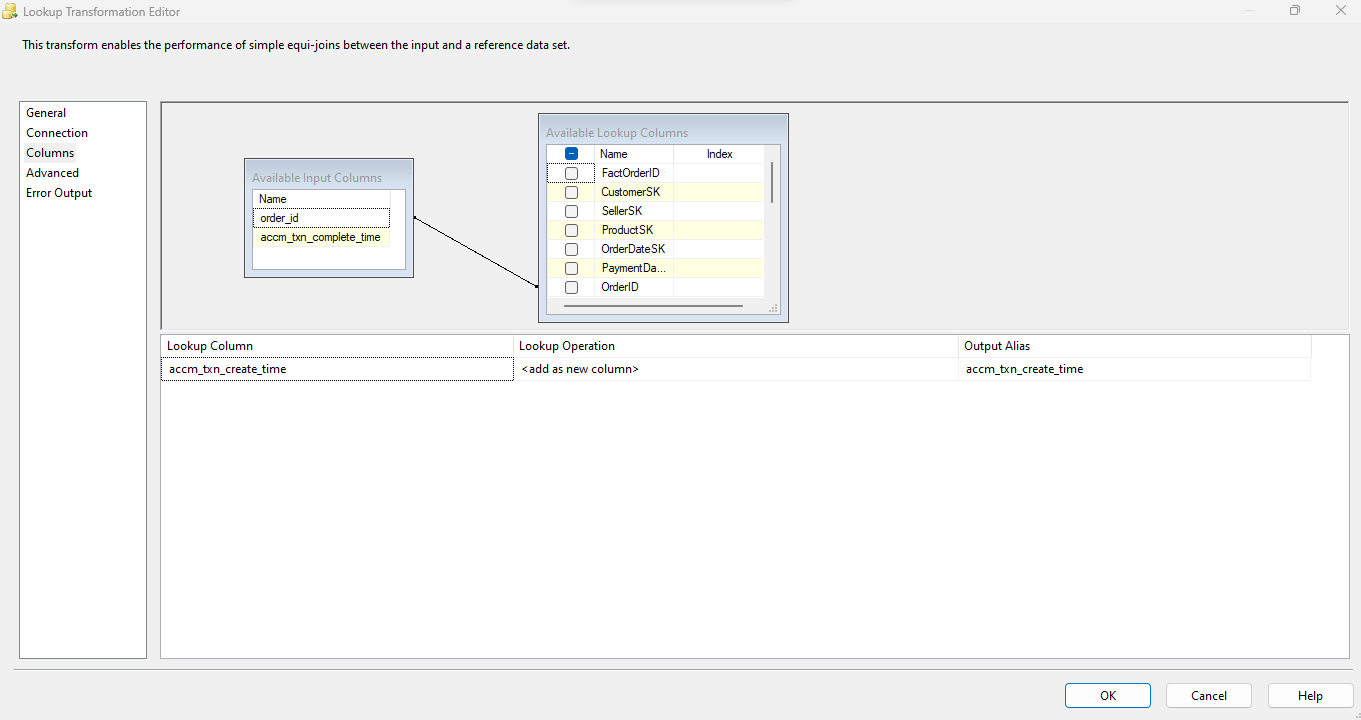
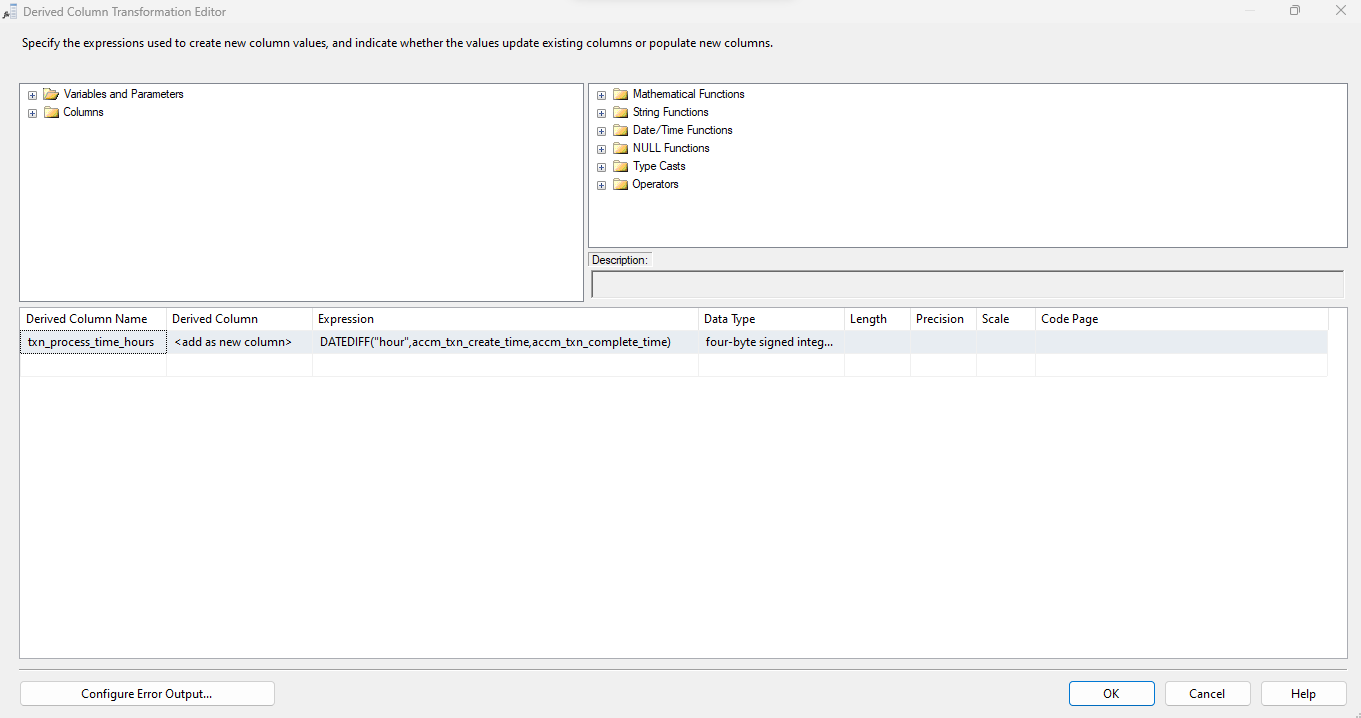
The update data is retrieved from a SQL table named **fact\_order\_updates**, which contains the order IDs and their respective transaction completion timestamps.

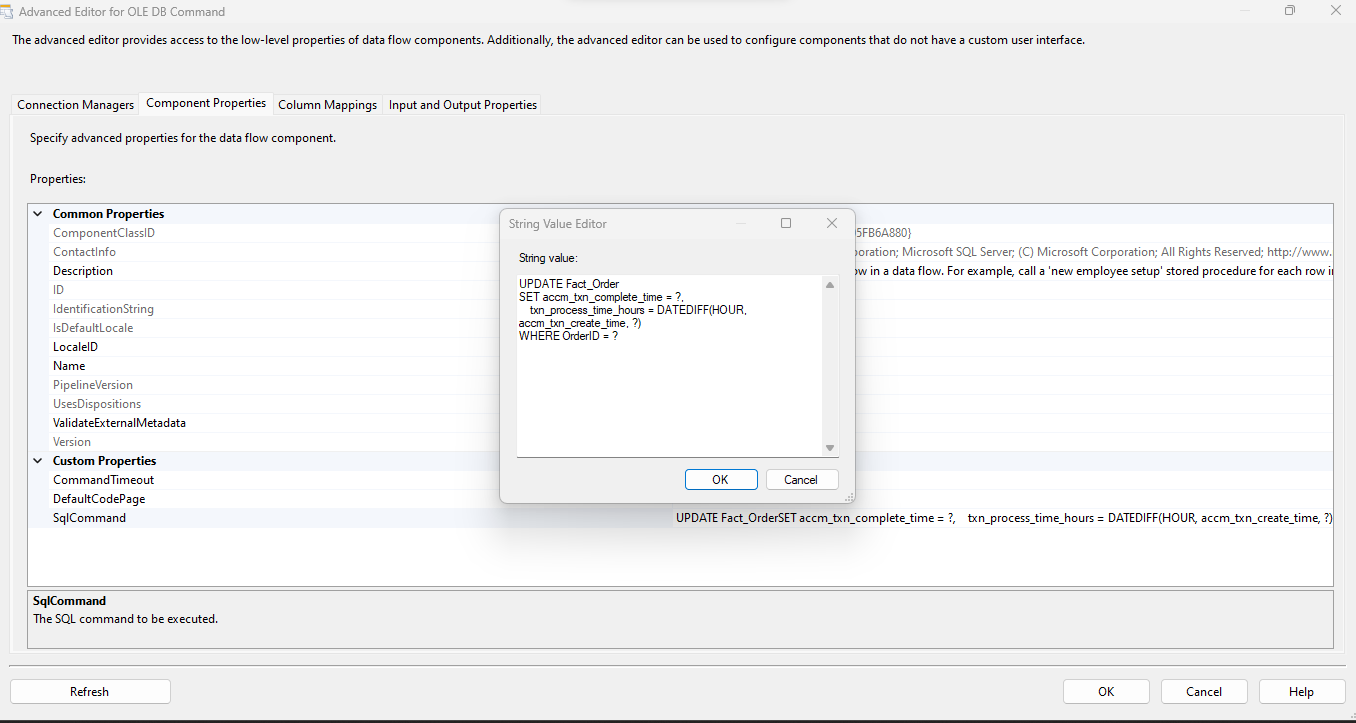


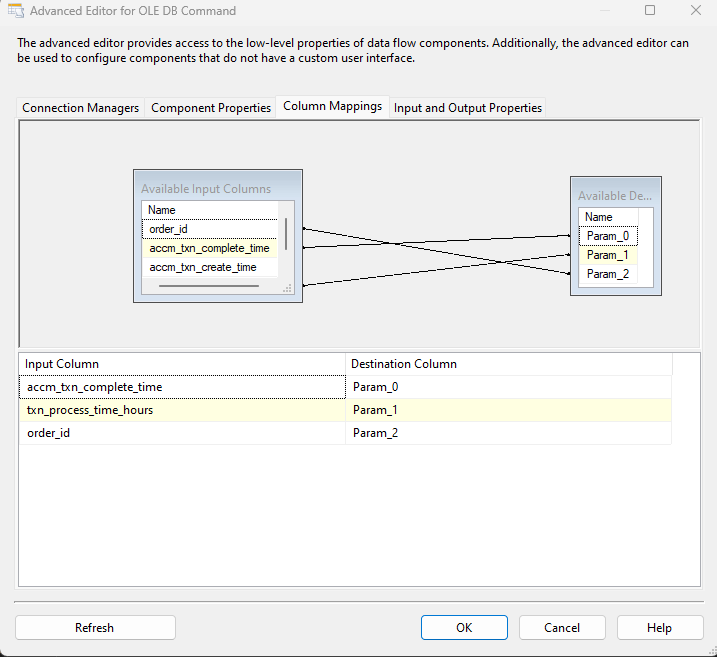
### **Separate SSIS Update Package**

A new SSIS package was created to perform the update operation:



1. OLE DB Source for SQL
   * Reads the completion update dataset
2. Lookup Transformation
   * Match on order\_id from Fact\_Order
3. Derived Column
4. OLE DB Command
   * SQL statement to update complete time and duration





1. Update\_FactOrderDetails Data Flow

