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

I. 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.
  .

II. 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

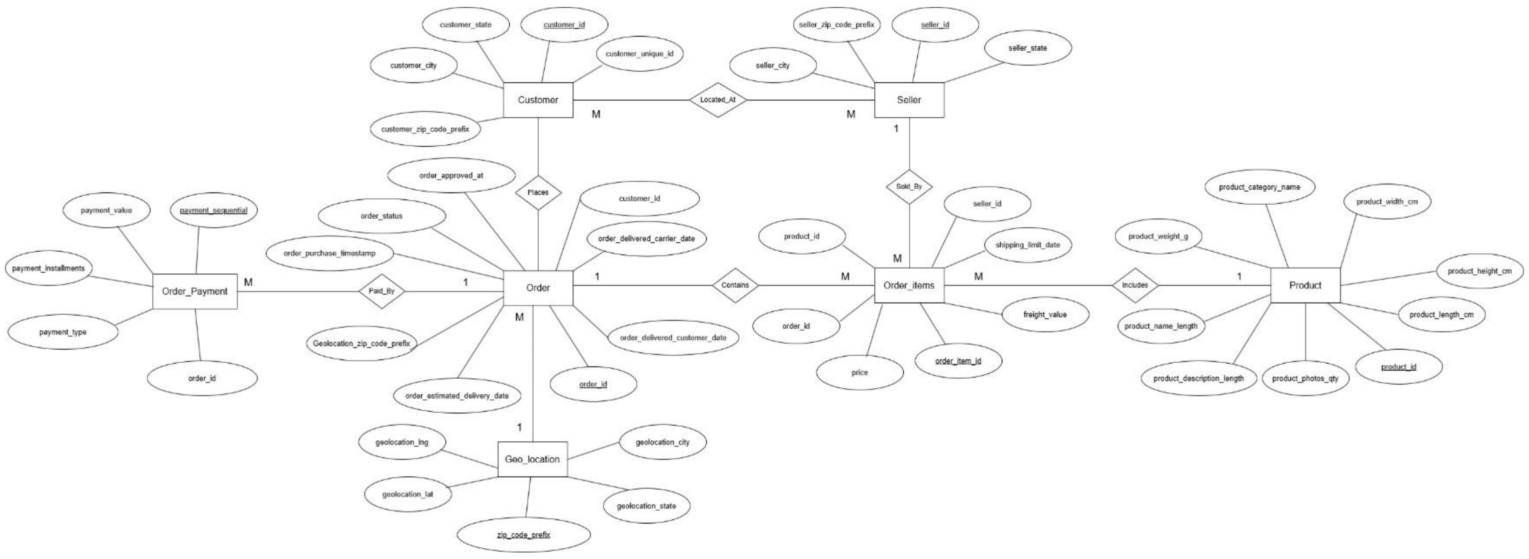
- Order Items
  - order_id
  - order_item_id
  - product_id
  - seller_id
  - shipping_limit_date
  - price
  - freight_value

- 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

III. ER Diagram



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

2. 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

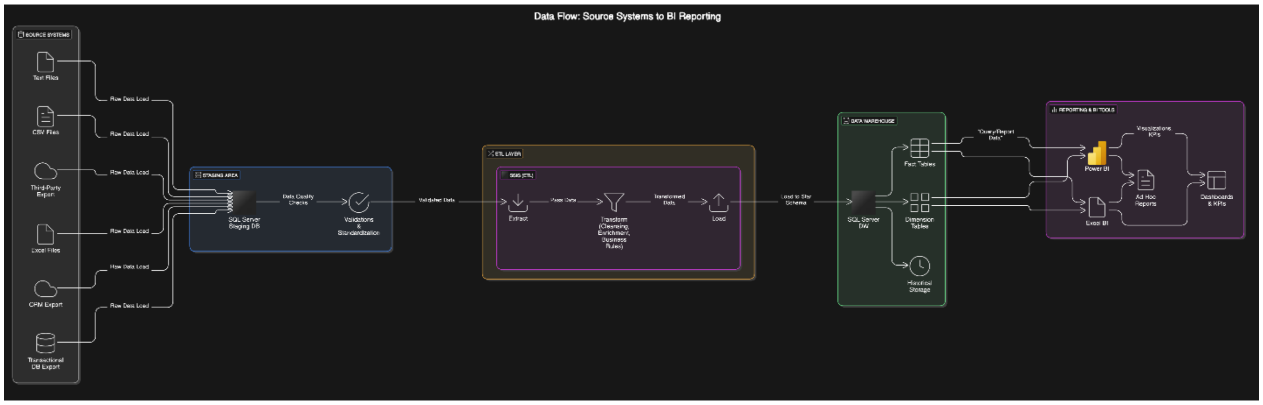
Orders Table(Olist SourseDB)	SQL DataBase	Captures order data including status and timestamps
olist_products_dataset.csv	CSV	Describes products, including category, dimensions, and weights.
olist_sellers_dataset.txt	Text (TXT)	Contains seller details including ZIP prefix, city, and state. Simulates export from vendor or merchant registry.

---

3. 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.

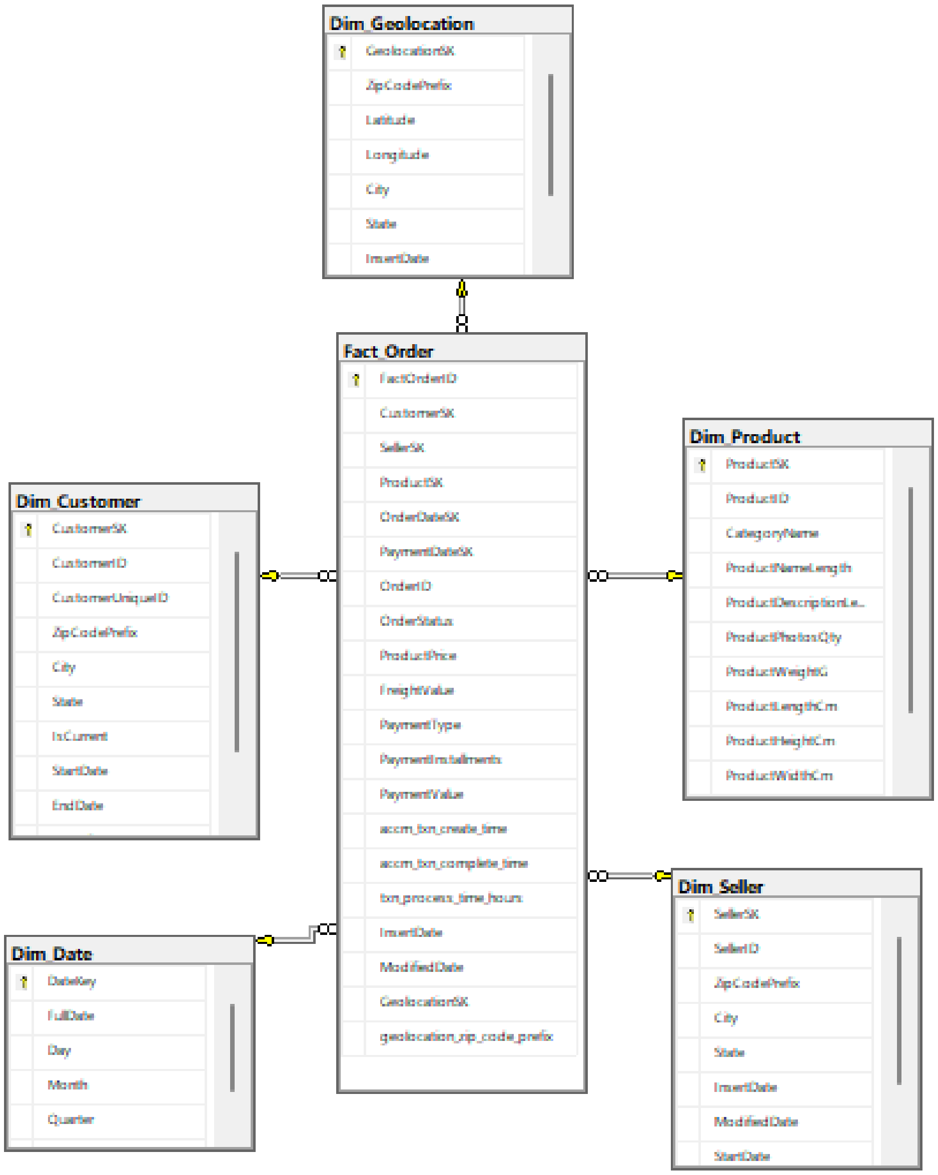
2. Architectural Components and Description

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

2. **Dim_Customers(SCD Type 2)**
   - ❖ CustomerID (Business Key)

- ZipCodePrefix
- City
- State
- IsCurrent
- InsertDate/ModifiedDate
- StartDate/EndDate

3. **Dim_Geolocation**

- ZipCodePrefix (Business Key)
- Latitude / Longitude
- City / State
- InsertDate/ModifiedDate

4. **Dim_Product**

- ProductID (Business Key)
- CategoryName
- ProductNameLength
- ProductDescriptionLength
- ProductWeightG
- ProductLengthCm / HeightCm / WidthCm
- InsertDate/ModifiedDate

5. **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 Type 2.
  - 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),
```

```sql
   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;
```

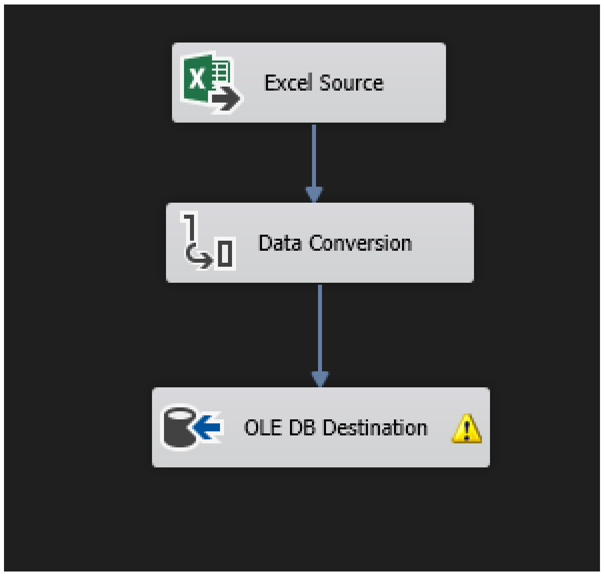
5. ETL development

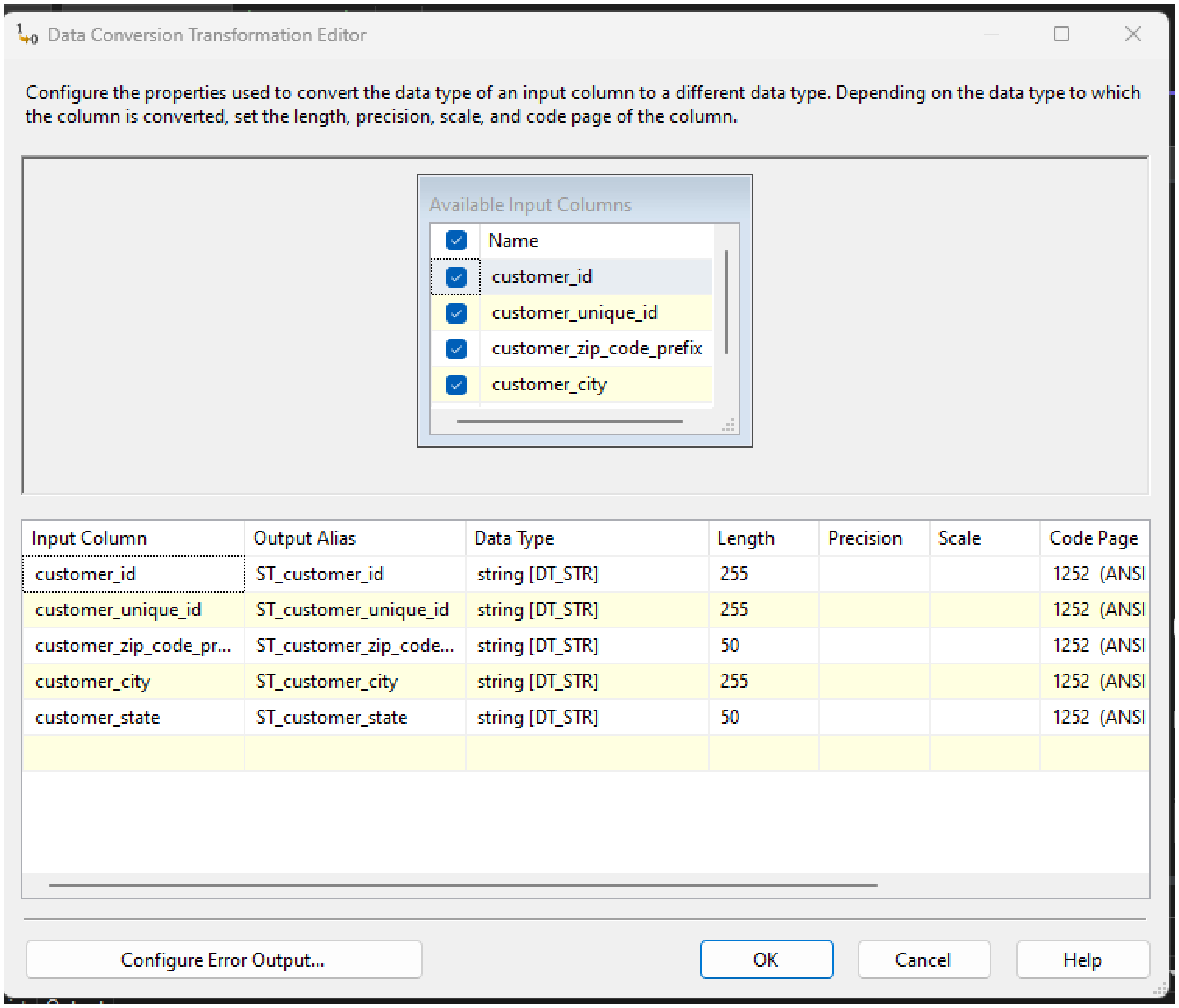
- **ETL WorkFlow Overview**

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**





Data Conversion Transformation Editor

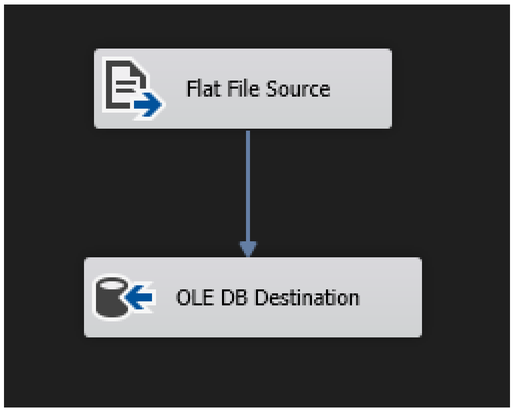
Configure the properties used to convert the data type of an input column to a different data type. Depending on the data type to which the column is converted, set the length, precision, scale, and code page of the column.

Available Input Columns

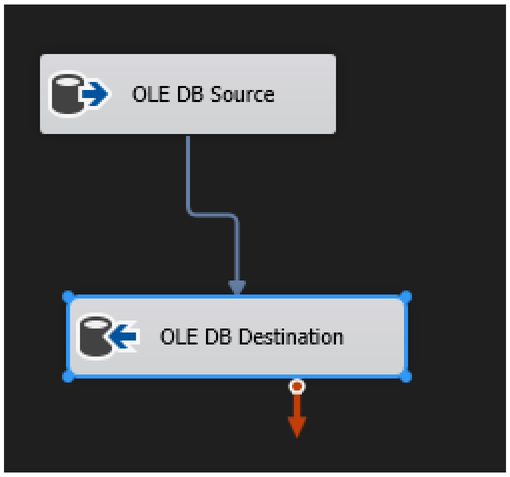
✓	Name
✓	customer_id
✓	customer_unique_id
✓	customer_zip_code_prefix
✓	customer_city

Input Column	Output Alias	Data Type	Length	Precision	Scale	Code Page
customer_id	ST_customer_id	string [DT_STR]	255			1252 (ANSI
customer_unique_id	ST_customer_unique_id	string [DT_STR]	255			1252 (ANSI
customer_zip_code_pr...	ST_customer_zip_code...	string [DT_STR]	50			1252 (ANSI
customer_city	ST_customer_city	string [DT_STR]	255			1252 (ANSI
customer_state	ST_customer_state	string [DT_STR]	50			1252 (ANSI

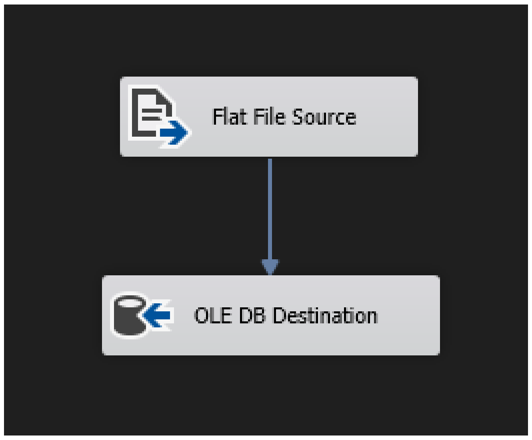
- **Stg_Sellers- Extract Seller Data To Staging**



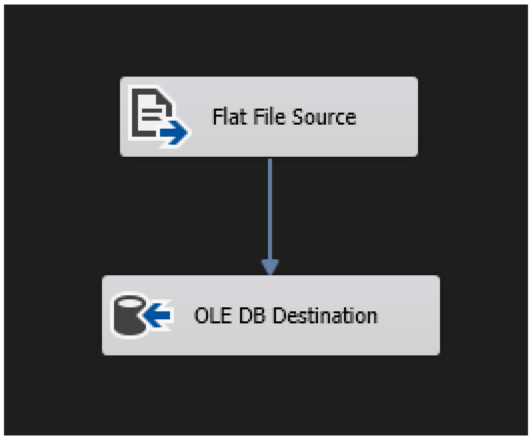
- Stg_Orders-Extract Customer Data To Staging



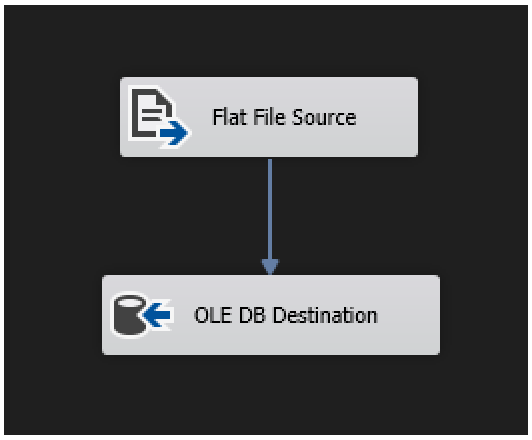
- Stg_Products-Extract Product Data To Staging



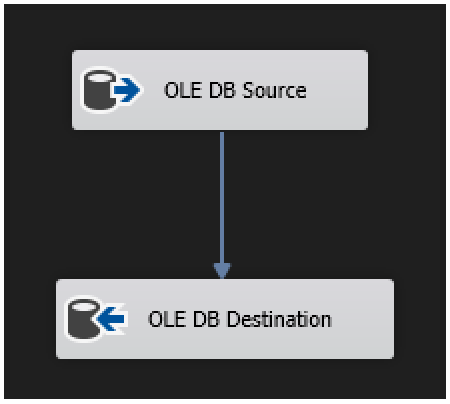
- Stg_OrderItems-Extract OrderItem Data To Staging



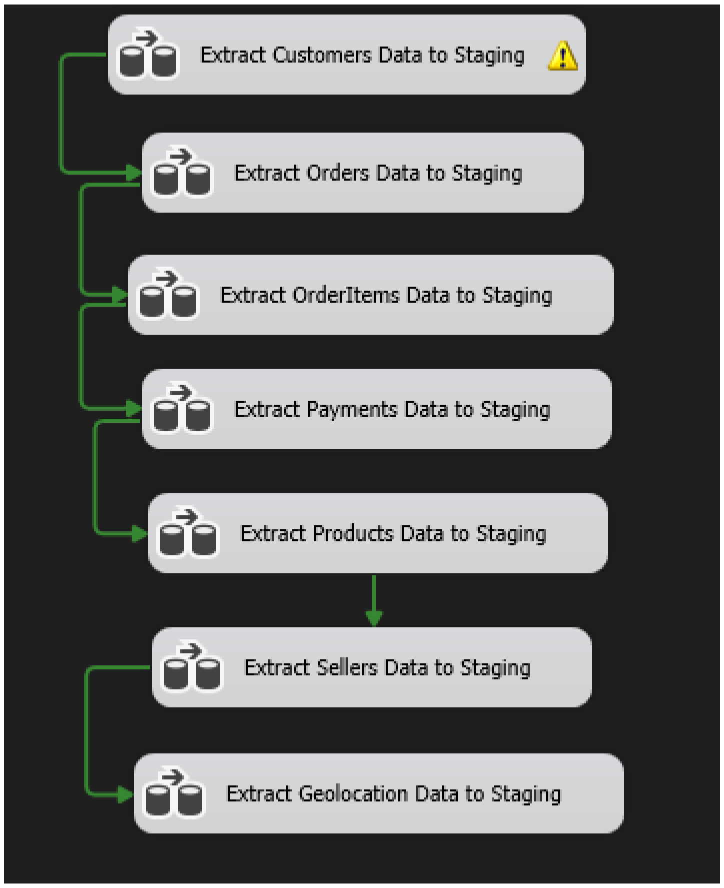
- Stg_Payments-Extract Payments Data To Staging



- Stg_Geolocation Extract Locations Data To Staging



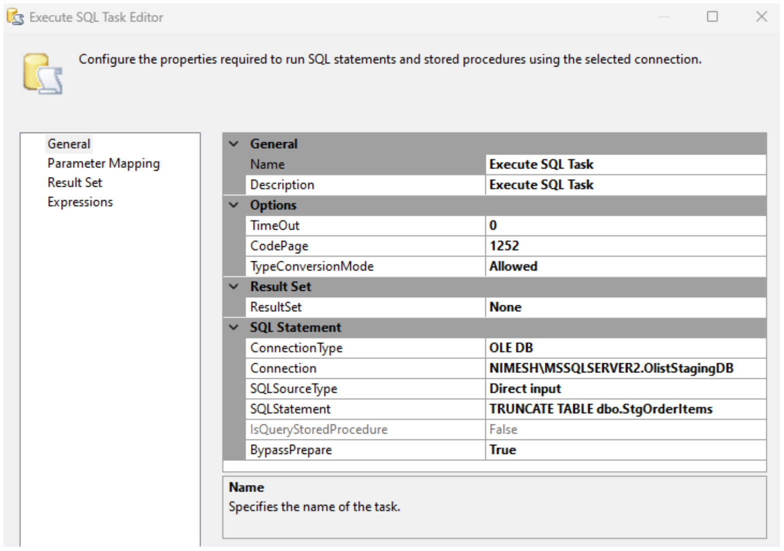
- Staging WorkFlow -Extract All Data To Staging


Extract Customers Data to Staging
Extract Orders Data to Staging
Extract OrderItems Data to Staging
Extract Payments Data to Staging
Extract Products Data to Staging
Extract Sellers Data to Staging
Extract Geolocation Data to Staging

- **Truncating Staging Tables**

→ **TRUNCATE and LOAD strategy** was applied.

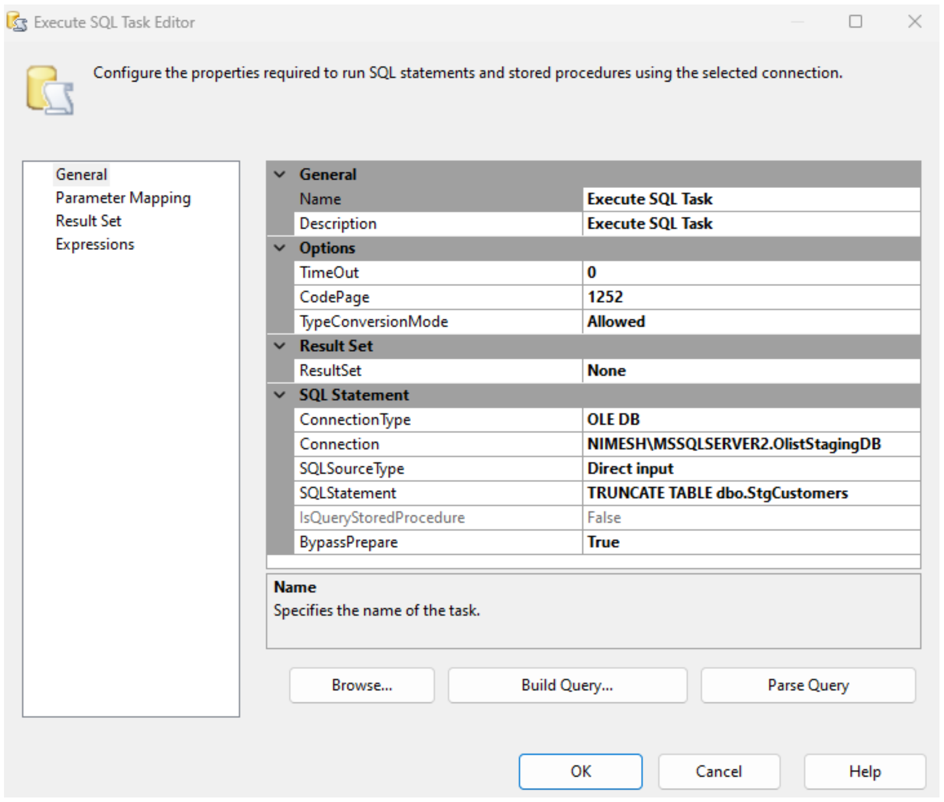
This means the staging tables were **cleared (truncated) before each load**, ensuring no residual or duplicate data remained from previous ETL runs. This simplifies data handling and ensures consistency.



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

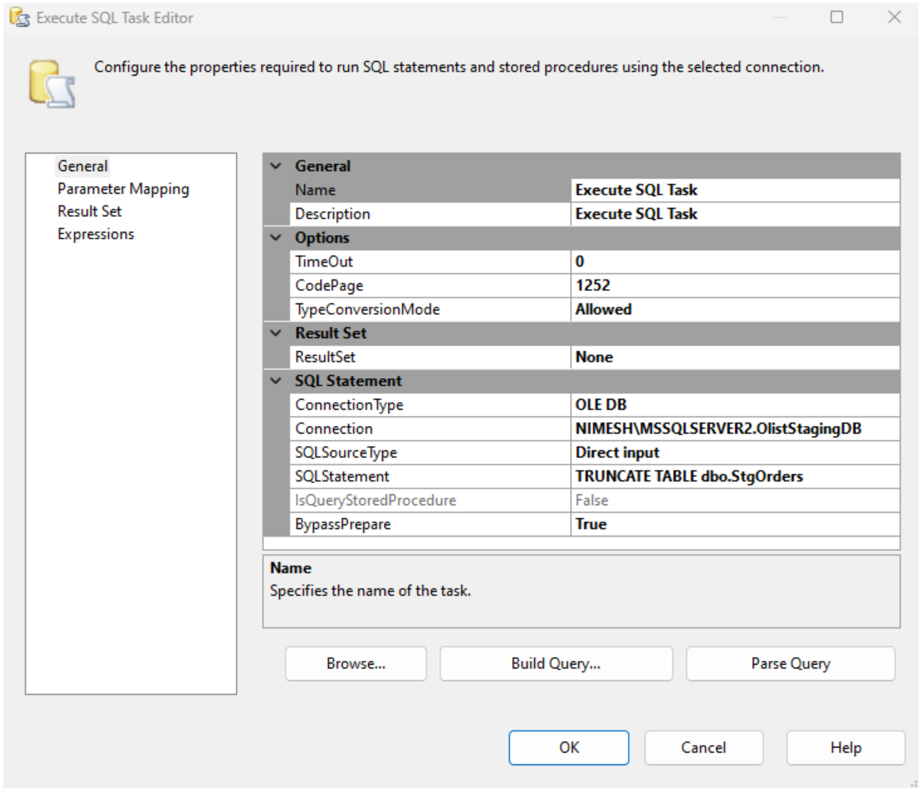
General	
Name	Execute SQL Task
Description	Execute SQL Task
Options	
TimeOut	0
CodePage	1252
TypeConversionMode	Allowed
Result Set	
ResultSet	None
SQL Statement	
ConnectionType	OLE DB
Connection	NIMESH\MSSQLSERVER2.OlistStagingDB
SQLSourceType	Direct input
SQLStatement	TRUNCATE TABLE dbo.StgOrderItems
IsQueryStoredProcedure	False
BypassPrepare	True

Name
Specifies the name of the task.



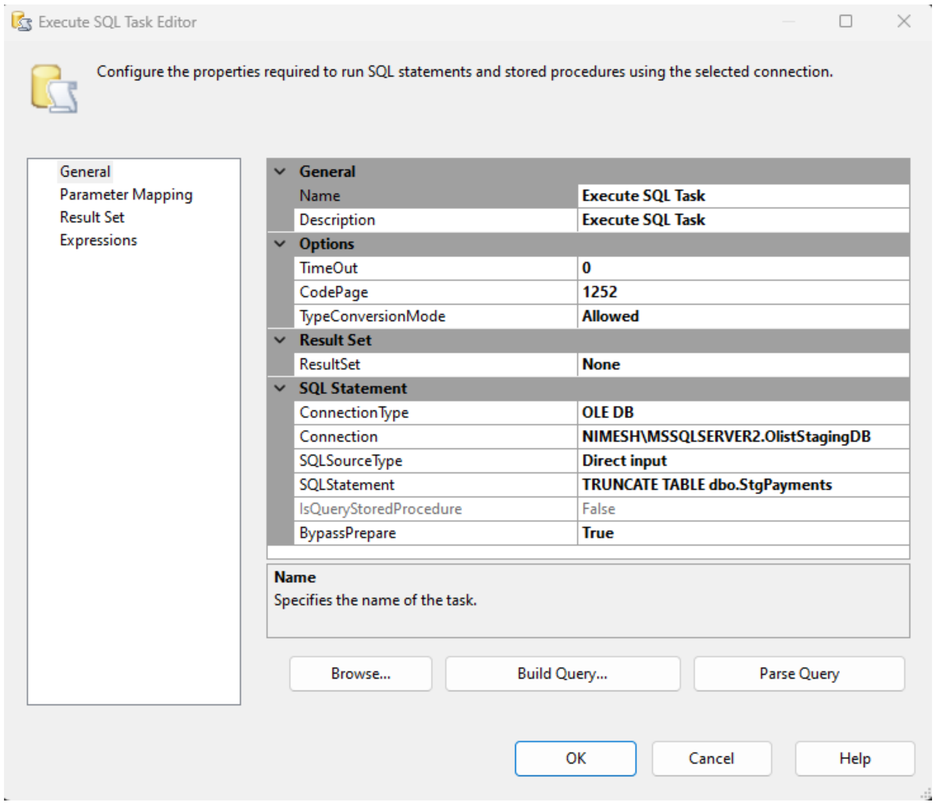
General	
Name	Execute SQL Task
Description	Execute SQL Task
Options	
TimeOut	0
CodePage	1252
TypeConversionMode	Allowed
Result Set	
ResultSet	None
SQL Statement	
ConnectionType	OLE DB
Connection	NIMESH\MSSQLSERVER2.OlistStagingDB
SQLSourceType	Direct input
SQLStatement	TRUNCATE TABLE dbo.StgCustomers
IsQueryStoredProcedure	False
BypassPrepare	True

Name
Specifies the name of the task.



General	
Name	Execute SQL Task
Description	Execute SQL Task
Options	
TimeOut	0
CodePage	1252
TypeConversionMode	Allowed
Result Set	
ResultSet	None
SQL Statement	
ConnectionType	OLE DB
Connection	NIMESH\MSSQLSERVER2.OlistStagingDB
SQLSourceType	Direct input
SQLStatement	TRUNCATE TABLE dbo.StgOrders
IsQueryStoredProcedure	False
BypassPrepare	True

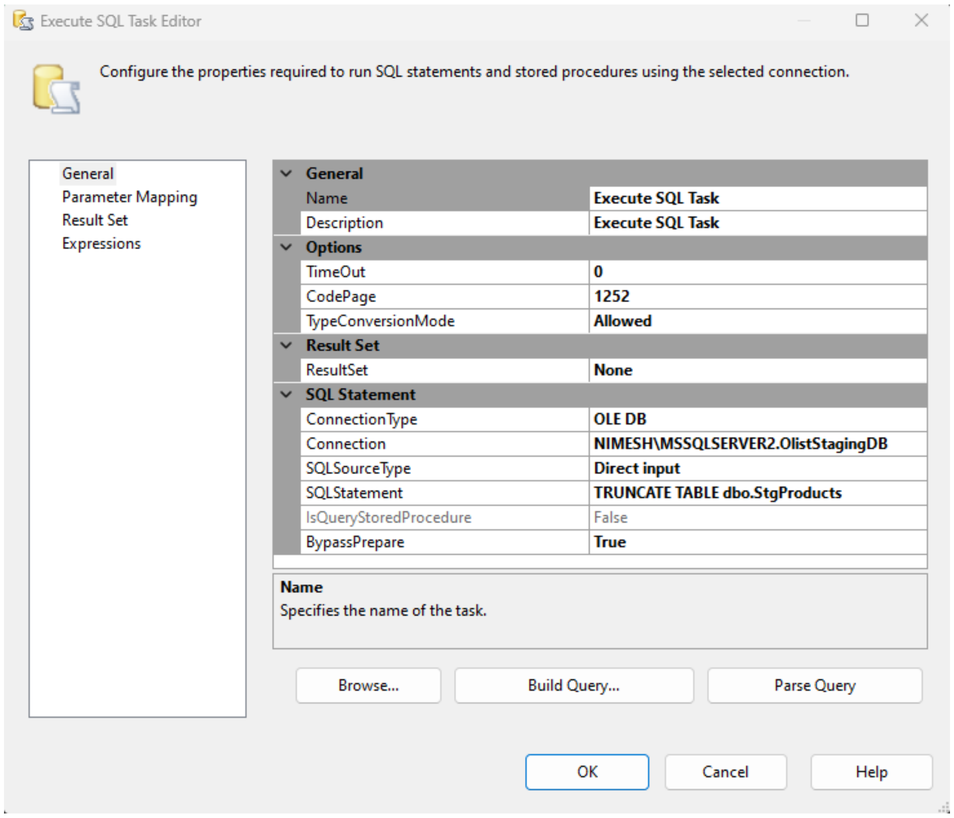
Name
Specifies the name of the task.



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

General	
Name	Execute SQL Task
Description	Execute SQL Task
Options	
TimeOut	0
CodePage	1252
TypeConversionMode	Allowed
Result Set	
ResultSet	None
SQL Statement	
ConnectionType	OLE DB
Connection	NIMESH\MSSQLSERVER2.OlistStagingDB
SQLSourceType	Direct input
SQLStatement	TRUNCATE TABLE dbo.StgPayments
IsQueryStoredProcedure	False
BypassPrepare	True

Name
Specifies the name of the task.



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

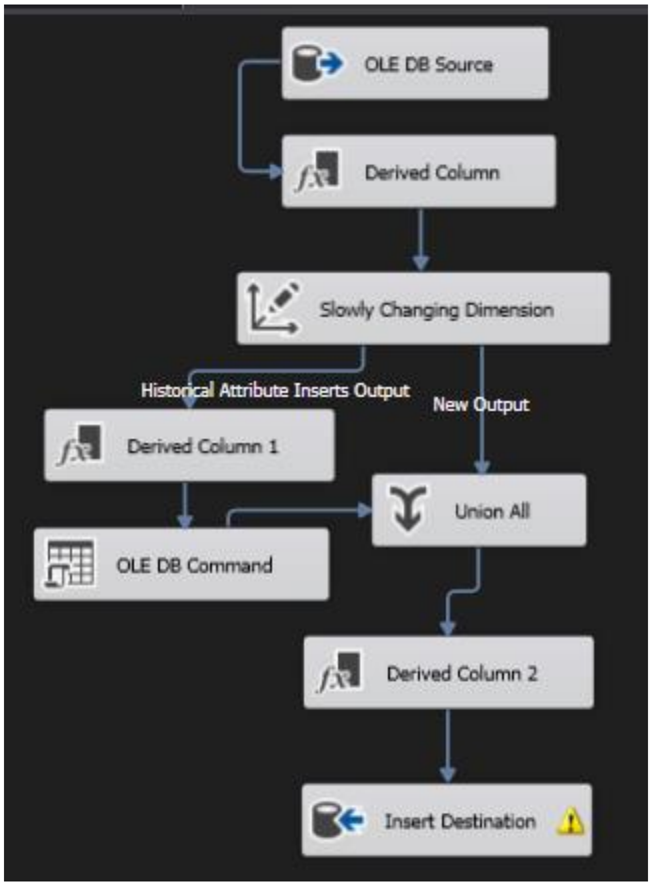
General	
Name	Execute SQL Task
Description	Execute SQL Task
Options	
TimeOut	0
CodePage	1252
TypeConversionMode	Allowed
Result Set	
ResultSet	None
SQL Statement	
ConnectionType	OLE DB
Connection	NIMESH\MSSQLSERVER2.OlistStagingDB
SQLSourceType	Direct input
SQLStatement	TRUNCATE TABLE dbo.StgProducts
IsQueryStoredProcedure	False
BypassPrepare	True

Name
Specifies the name of the task.

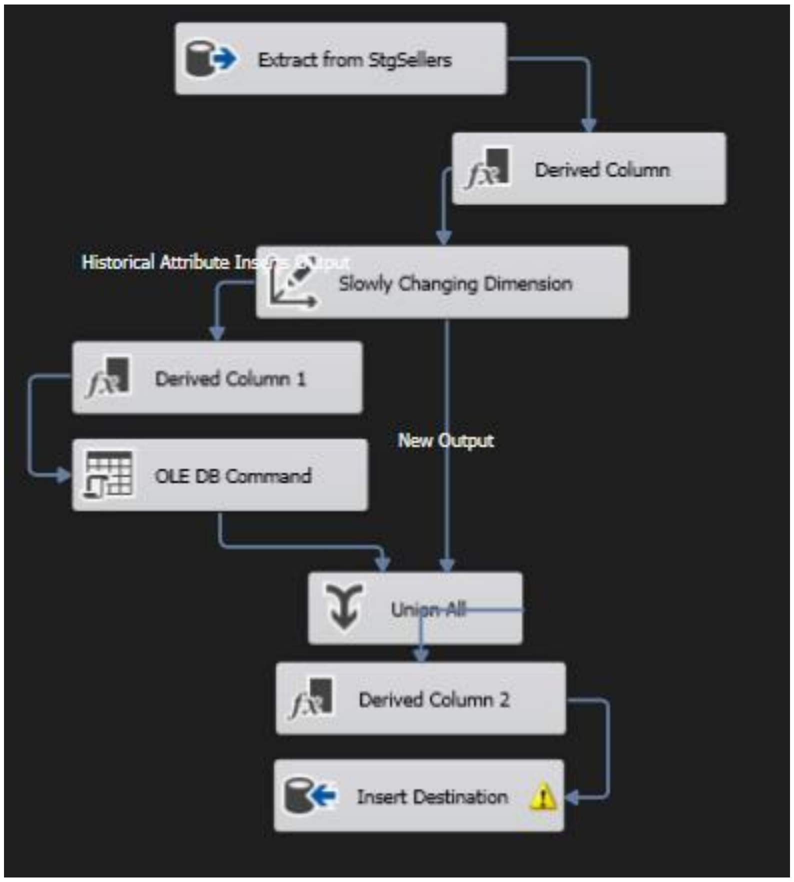
2. Transformation Phase

- 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

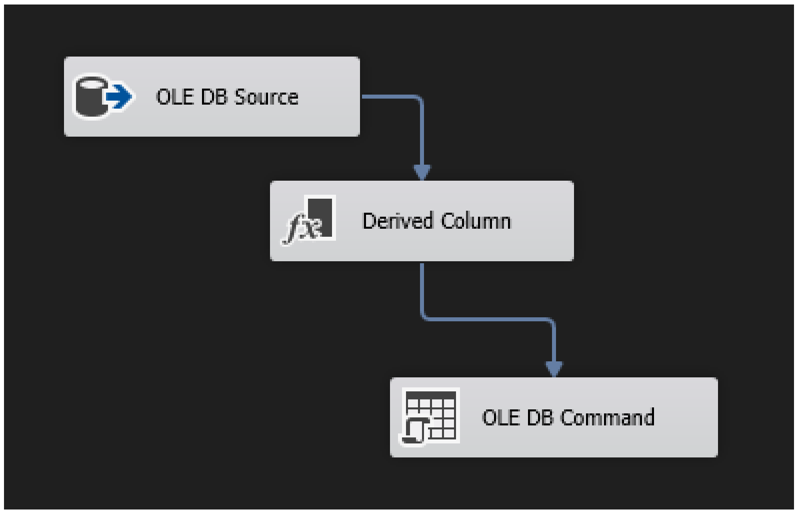
- **Dim_Customers- Transform and Load Dim_Customers**

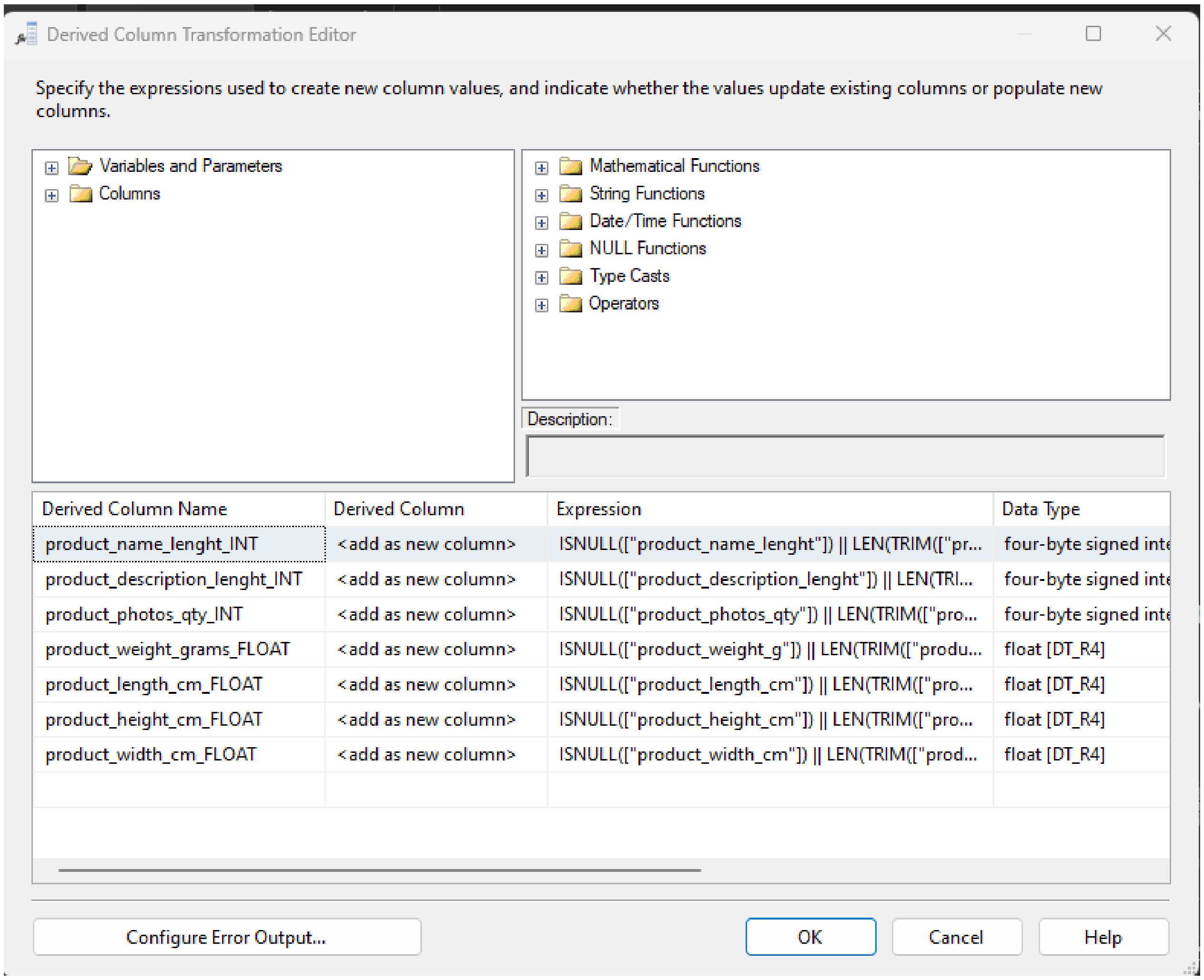


- **Dim_Seller- Transform and Load Dim_Seller**



- **Dim_Product- Transform and Load Dim_Product**



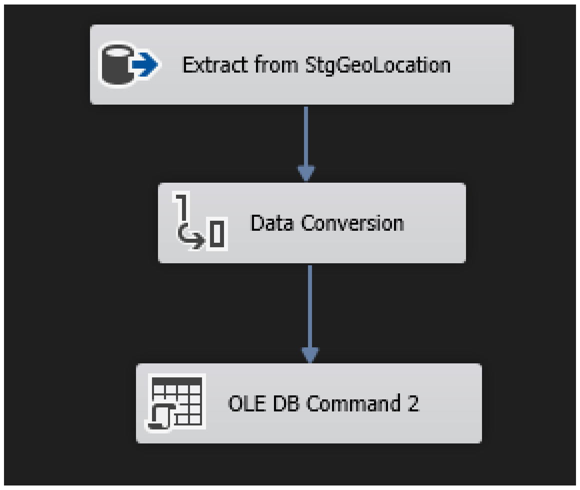


Derived Column Transformation Editor

Specify the expressions used to create new column values, and indicate whether the values update existing columns or populate new columns.

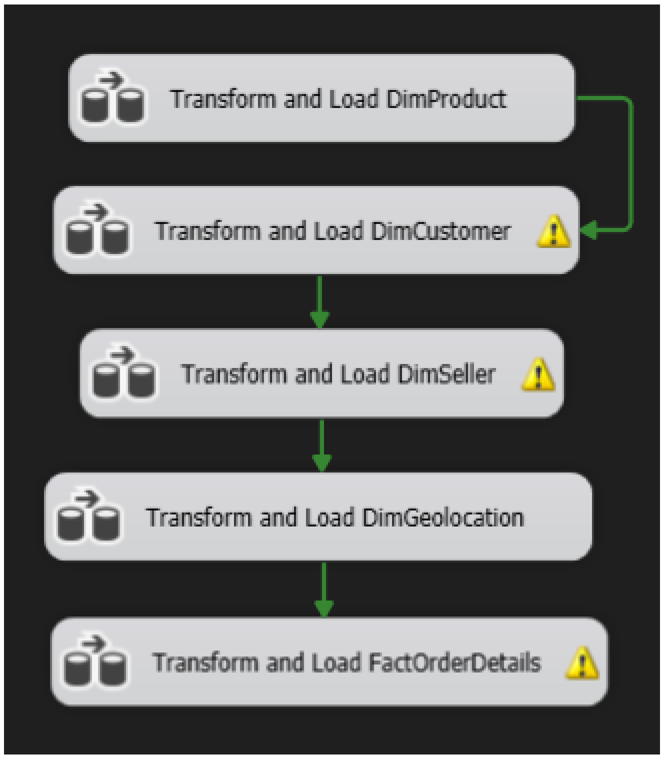
Derived Column Name	Derived Column	Expression	Data Type
product_name_lenght_INT	<add as new column>	ISNULL(["product_name_lenght"]) || LEN(TRIM(["pr...	four-byte signed inte
product_description_lenght_INT	<add as new column>	ISNULL(["product_description_lenght"]) || LEN(TRI...	four-byte signed inte
product_photos_qty_INT	<add as new column>	ISNULL(["product_photos_qty"]) || LEN(TRIM(["pro...	four-byte signed inte
product_weight_grams_FLOAT	<add as new column>	ISNULL(["product_weight_g"]) || LEN(TRIM(["produ...	float [DT_R4]
product_length_cm_FLOAT	<add as new column>	ISNULL(["product_length_cm"]) || LEN(TRIM(["pro...	float [DT_R4]
product_height_cm_FLOAT	<add as new column>	ISNULL(["product_height_cm"]) || LEN(TRIM(["pro...	float [DT_R4]
product_width_cm_FLOAT	<add as new column>	ISNULL(["product_width_cm"]) || LEN(TRIM(["prod...	float [DT_R4]

- **Dim_Geolocation - Transform and Load Dim_Geolocation**

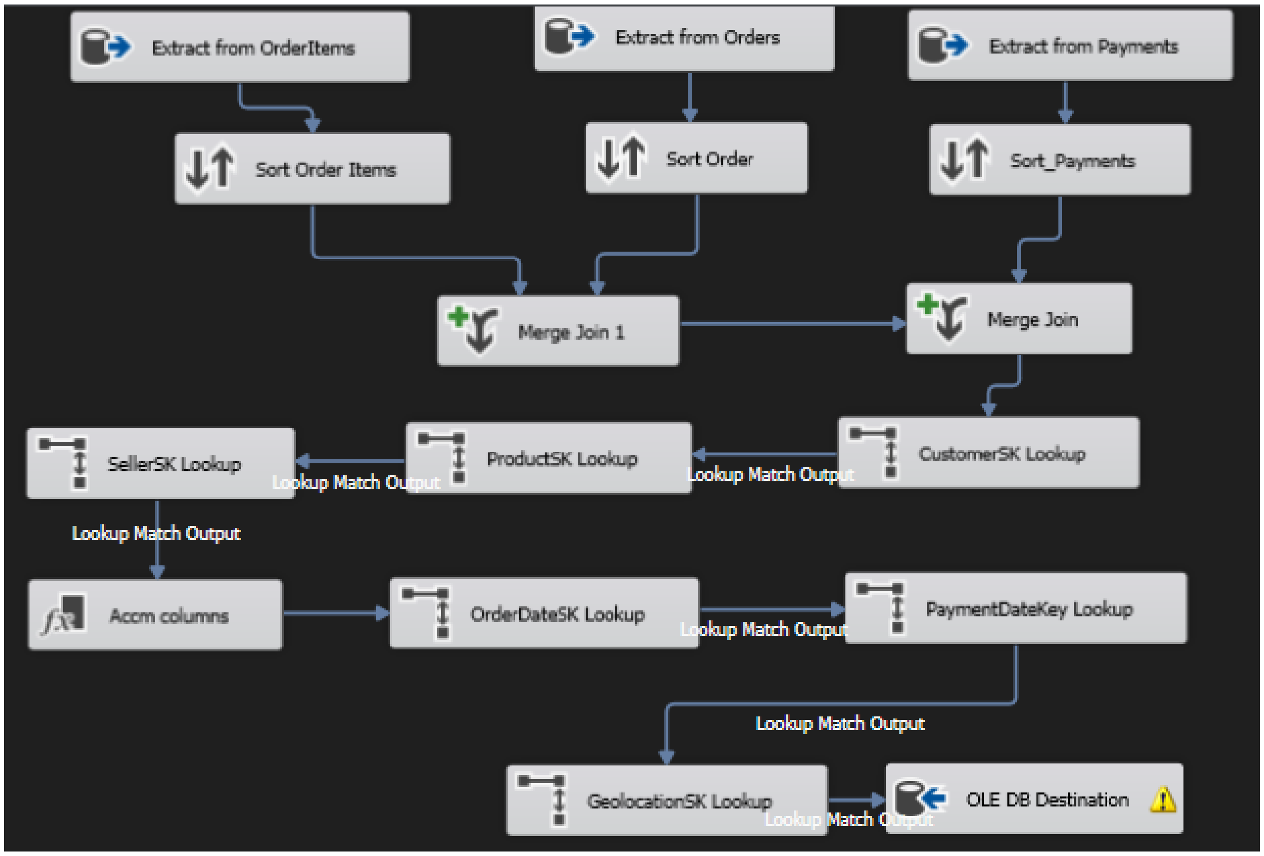


3. Loading Phase

- Load **Fact_Order** using resolved surrogate keys
- Ensure all referential integrity is preserved
- Load dimension tables


Transform and Load DimProduct
Transform and Load DimCustomer
Transform and Load DimSeller
Transform and Load DimGeolocation
Transform and Load FactOrderDetails

- **Fact_Order - Transform and Load Fact_Order**



6. ETL development – Accumulating fact tables

- **Extended Fact Table Columns**

The following columns were added to Fact_Order:

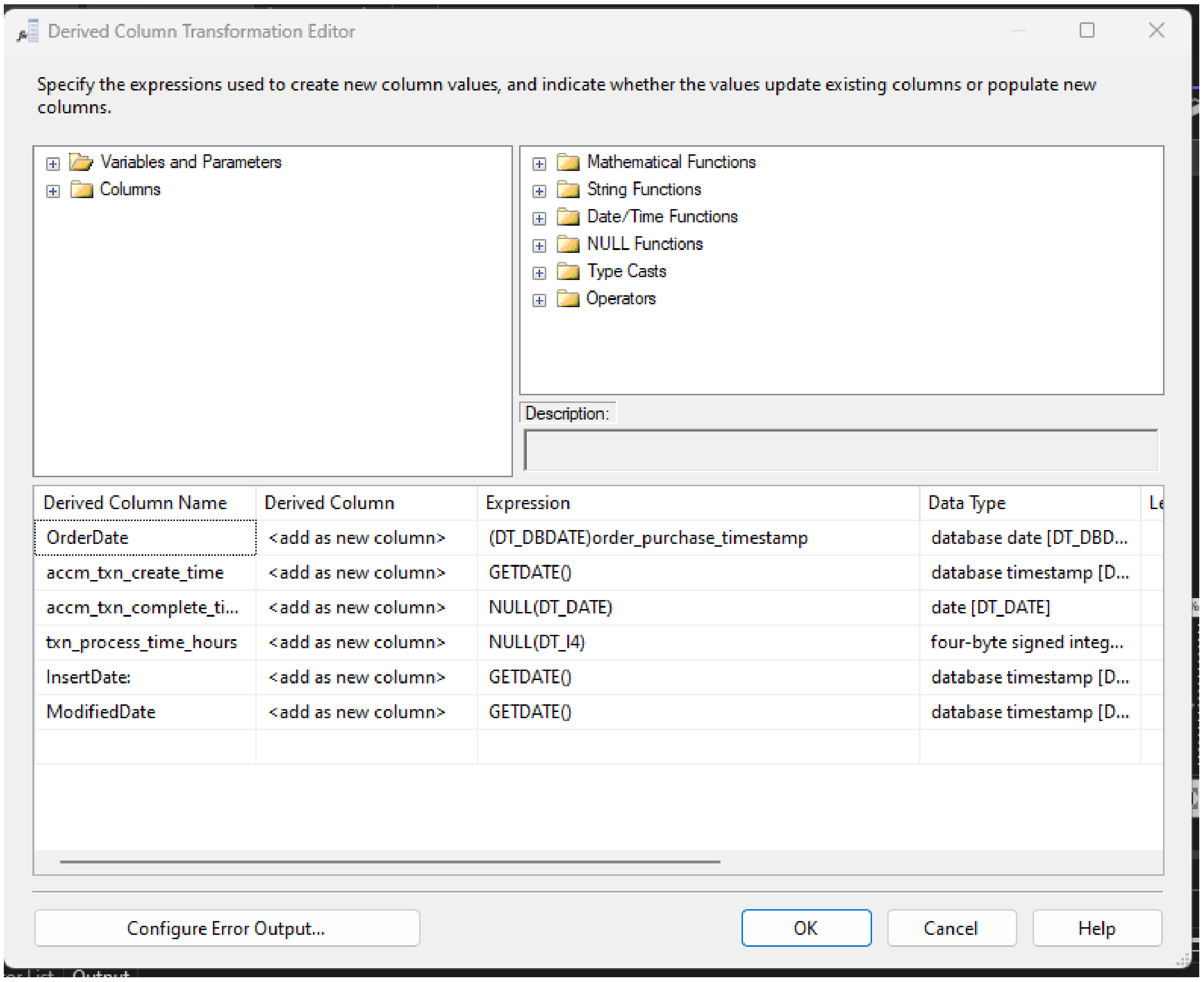
- accm_txn_create_time DATETIME
- accm_txn_complete_time DATETIME NULL
- txn_process_time_hours INT NULL

- **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)



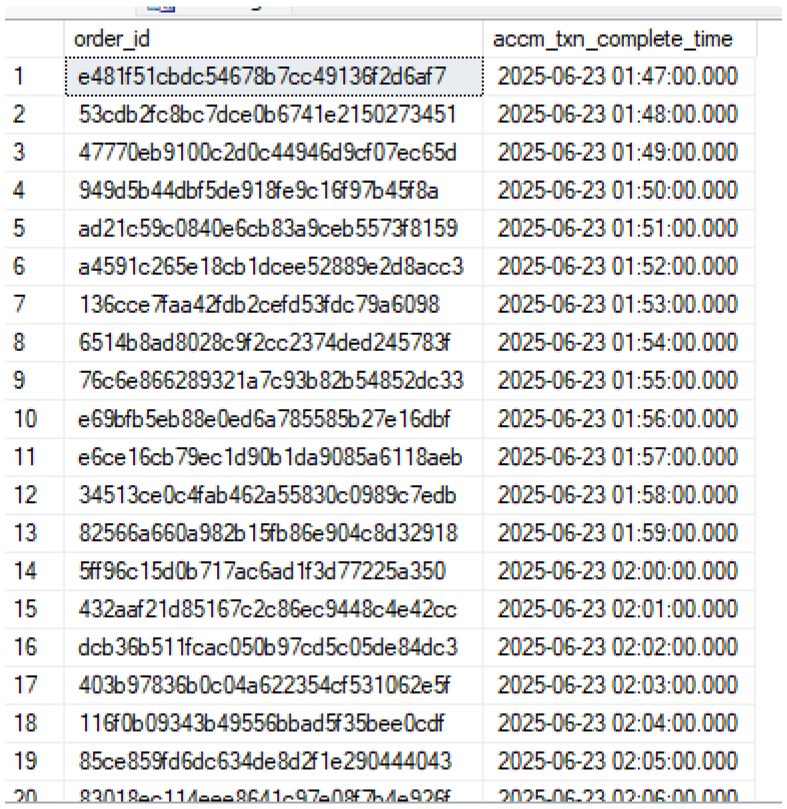
Derived Column Transformation Editor

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	Le
OrderDate	<add as new column>	(DT_DBDATE)order_purchase_timestamp	database date [DT_DBD...	
accm_txn_create_time	<add as new column>	GETDATE()	database timestamp [D...	
accm_txn_complete_ti...	<add as new column>	NULL(DT_DATE)	date [DT_DATE]	
txn_process_time_hours	<add as new column>	NULL(DT_I4)	four-byte signed integ...	
InsertDate:	<add as new column>	GETDATE()	database timestamp [D...	
ModifiedDate	<add as new column>	GETDATE()	database timestamp [D...	

- **Completion Time Data Source**

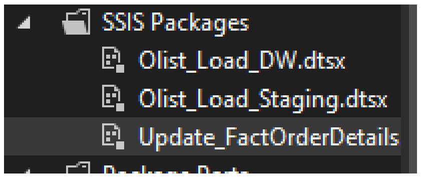
The update data is retrieved from a SQL table named **fact_order_updates**, which contains the order IDs and their respective transaction completion timestamps.



	order_id	accm_txn_complete_time
1	e481f51cbdc54678b7cc49136f2d6af7	2025-06-23 01:47:00.000
2	53cdb2fc8bc7dce0b6741e2150273451	2025-06-23 01:48:00.000
3	47770eb9100c2d0c44946d9cf07ec65d	2025-06-23 01:49:00.000
4	949d5b44dbf5de918fe9c16f97b45f8a	2025-06-23 01:50:00.000
5	ad21c59c0840e6cb83a9ceb5573f8159	2025-06-23 01:51:00.000
6	a4591c265e18cb1dcee52889e2d8acc3	2025-06-23 01:52:00.000
7	136cce7faa42fdb2cefd53fdc79a6098	2025-06-23 01:53:00.000
8	6514b8ad8028c9f2cc2374ded245783f	2025-06-23 01:54:00.000
9	76c6e866289321a7c93b82b54852dc33	2025-06-23 01:55:00.000
10	e69bfb5eb88e0ed6a785585b27e16dbf	2025-06-23 01:56:00.000
11	e6ce16cb79ec1d90b1da9085a6118aeb	2025-06-23 01:57:00.000
12	34513ce0c4fab462a55830c0989c7edb	2025-06-23 01:58:00.000
13	82566a660a982b15fb86e904c8d32918	2025-06-23 01:59:00.000
14	5ff96c15d0b717ac6ad1f3d77225a350	2025-06-23 02:00:00.000
15	432aaf21d85167c2c86ec9448c4e42cc	2025-06-23 02:01:00.000
16	dcb36b511fcac050b97cd5c05de84dc3	2025-06-23 02:02:00.000
17	403b97836b0c04a622354cf531062e5f	2025-06-23 02:03:00.000
18	116f0b09343b49556bbad5f35bee0cdf	2025-06-23 02:04:00.000
19	85ce859fd6dc634de8d2f1e290444043	2025-06-23 02:05:00.000
20	83018ec114eee8641c97e08f7b4e926f	2025-06-23 02:06:00.000

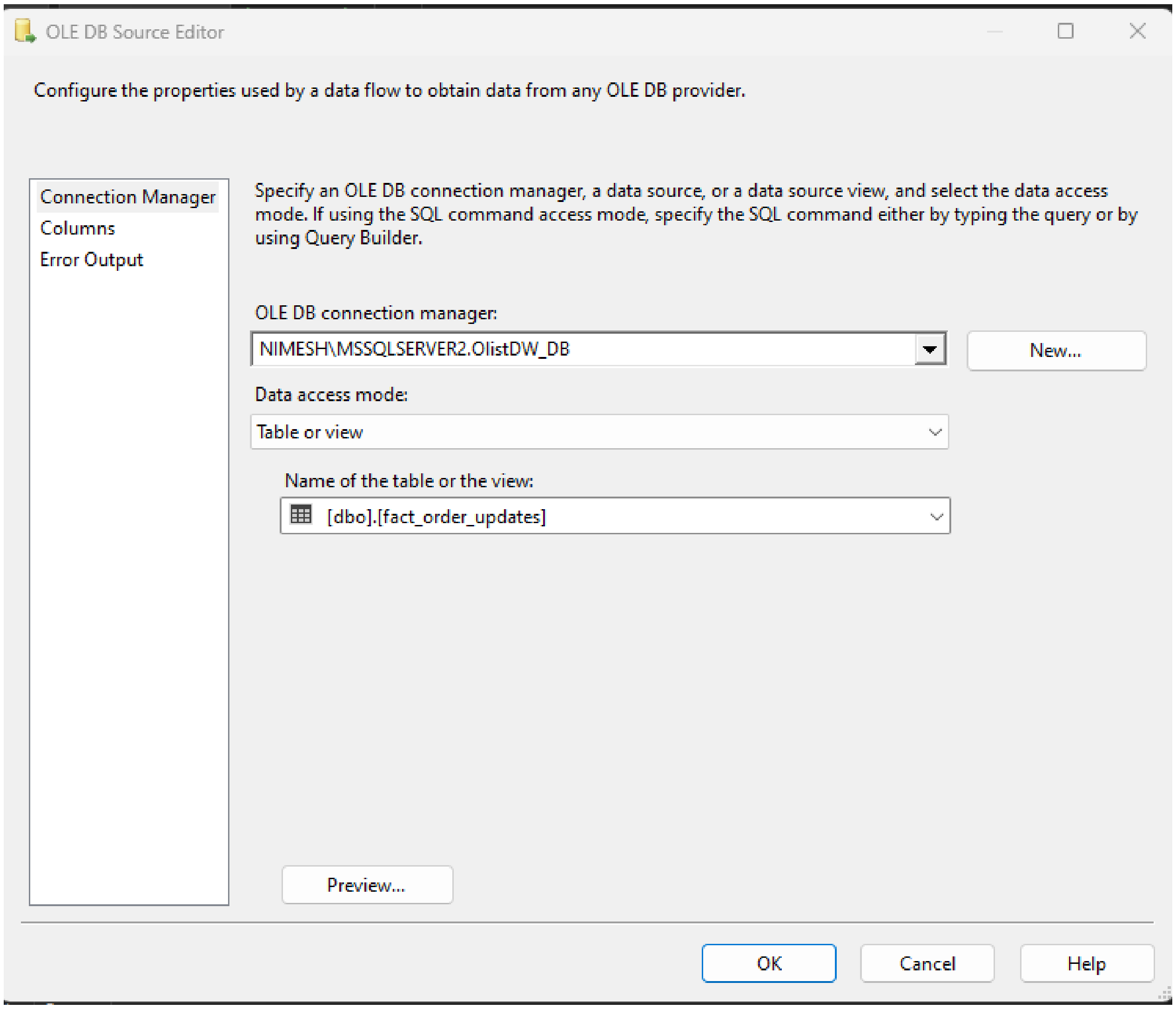
- **Separate SSIS Update Package**

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



SSIS Packages
Olist_Load_DW.dtsx
Olist_Load_Staging.dtsx
Update_FactOrderDetails

1. OLE DB Source for SQL
   - Reads the completion update dataset



OLE DB Source Editor

Configure the properties used by a data flow to obtain data from any OLE DB provider.

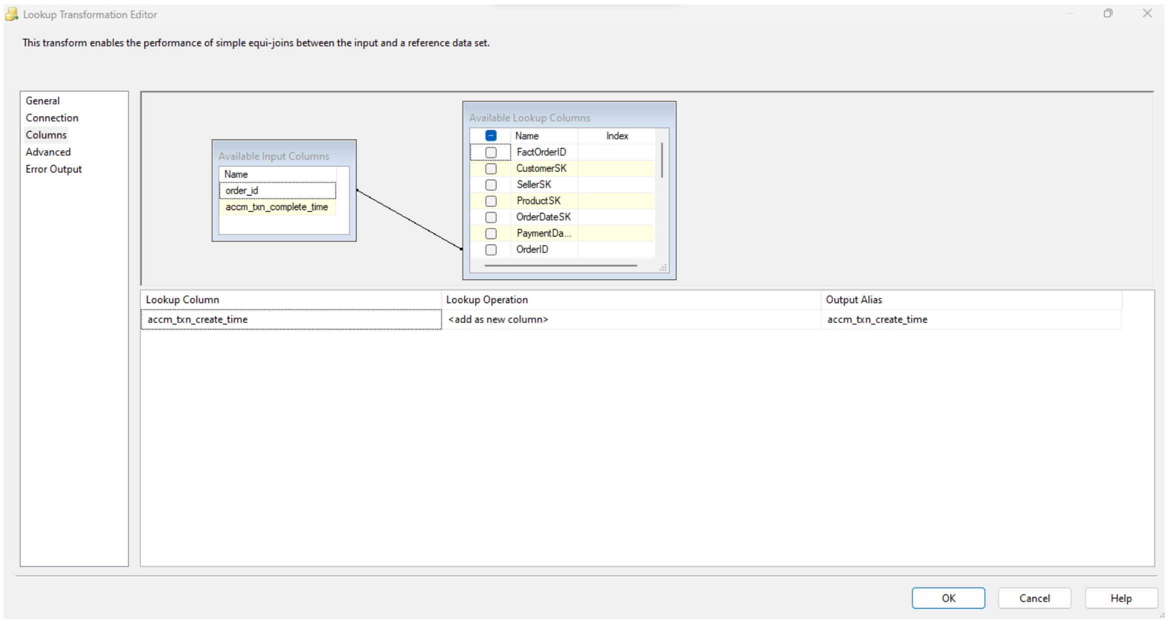
Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder.

OLE DB connection manager: NIMESH\MSSQLSERVER2.OlistDW_DB

Data access mode: Table or view

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

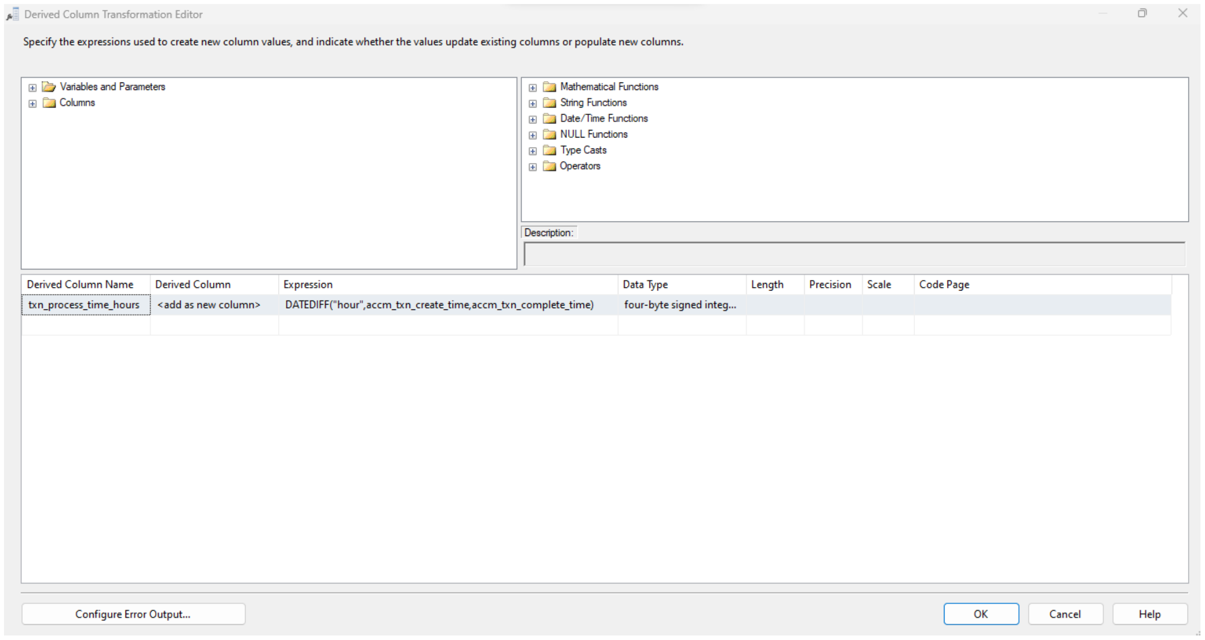
2. Lookup Transformation
   - Match on order_id from Fact_Order
3. Derived Column


Lookup Transformation Editor

This transform enables the performance of simple equi-joins between the input and a reference data set.

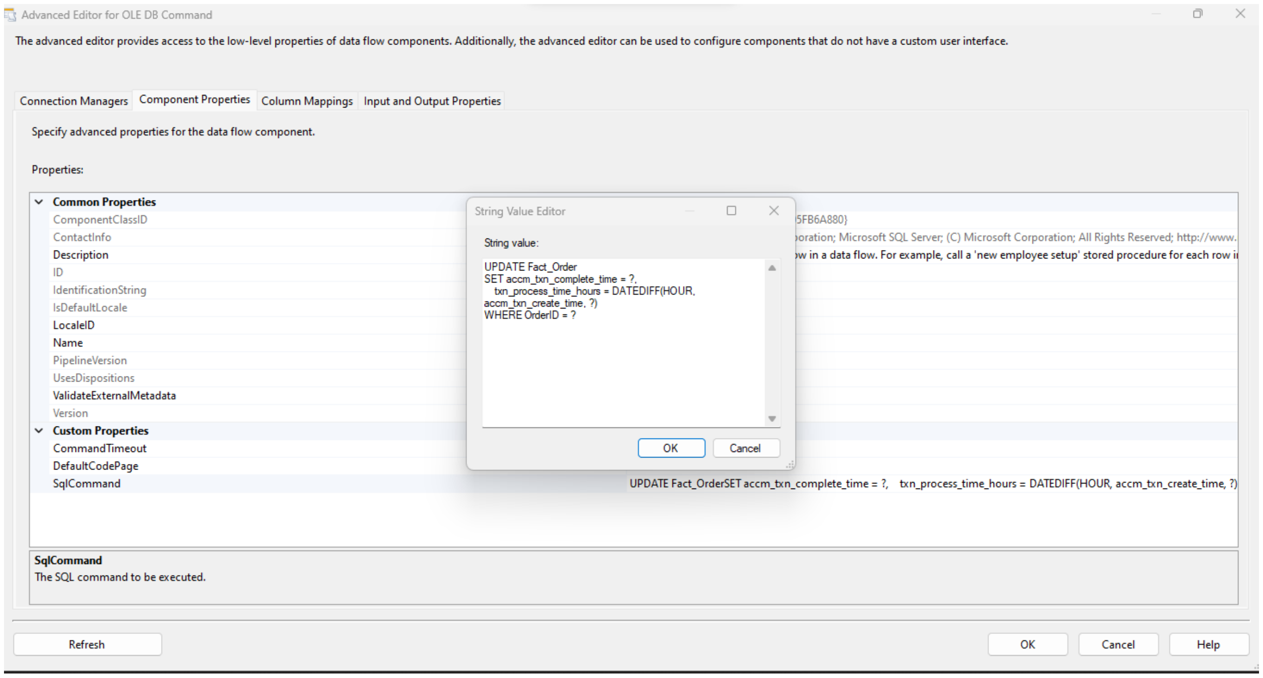
Lookup Column	Lookup Operation	Output Alias
accm_txn_create_time	<add as new column>	accm_txn_create_time

4. OLE DB Command
   - SQL statement to update complete time and duration


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	Length	Precision	Scale	Code Page
txn_process_time_hours	<add as new column>	DATEDIFF("hour",accm_txn_create_time,accm_txn_complete_time)	four-byte signed integ...				



Advanced Editor for OLE DB Command

The advanced editor provides access to the low-level properties of data flow components. Additionally, the advanced editor can be used to configure components that do not have a custom user interface.

Specify advanced properties for the data flow component.

Properties:

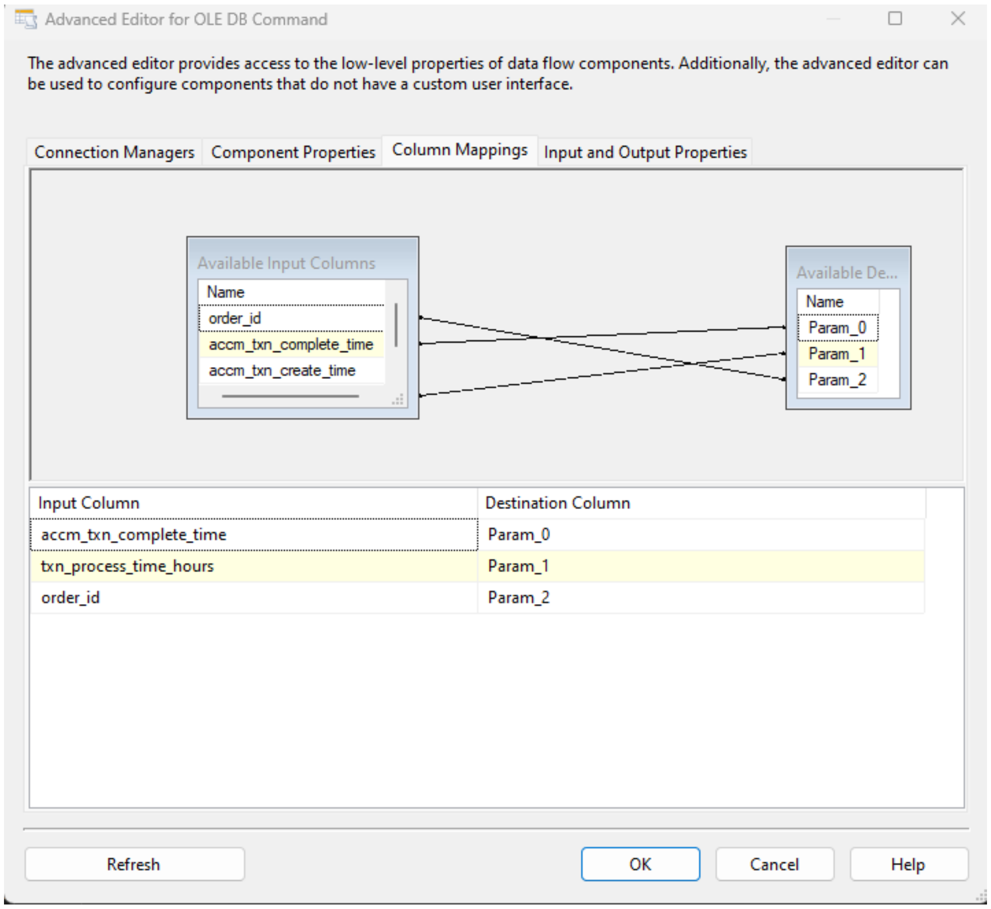
Common Properties	
ComponentClassID	5FB6A880}
ContactInfo	oration; Microsoft SQL Server; (C) Microsoft Corporation; All Rights Reserved; http://www.
Description	w in a data flow. For example, call a 'new employee setup' stored procedure for each row i
ID	
IdentificationString	
IsDefaultLocale	
LocaleID	
Name	
PipelineVersion	
UsesDispositions	
ValidateExternalMetadata	
Version	
Custom Properties	
CommandTimeout	
DefaultCodePage	
SqlCommand	UPDATE Fact_OrderSET accm_txn_complete_time = ?, txn_process_time_hours = DATEDIFF(HOUR, accm_txn_create_time, ?)

String Value Editor

String value:

```
UPDATE Fact_Order
SET accm_txn_complete_time = ?,
   txn_process_time_hours = DATEDIFF(HOUR,
accm_txn_create_time, ?)
WHERE OrderID = ?
```

SqlCommand
The SQL command to be executed.



Advanced Editor for OLE DB Command

The advanced editor provides access to the low-level properties of data flow components. Additionally, the advanced editor can be used to configure components that do not have a custom user interface.

Input Column	Destination Column
accm_txn_complete_time	Param_0
txn_process_time_hours	Param_1
order_id	Param_2