Sri Lanka Institute of Information Technology

Data warehousing and Business Intelligence

(**IT3021**)

Continuous Assignment – 2025, Semester 1

Assignment 1

Completed By:

Himsara P V D

IT22367112

Table of Contents

1. Introduction to the project

Data plays a pivotal role today in the corporate world and ranks as one of the most important assets. With millions of records coming up each day, dealing with huge volumes of data and extracting the meaningful insights from such data is proving to be a tough job. The data warehouse is a concept that is important for handling large-scale corporate data efficiently. This article discusses the architecture, development, as well as the ETL process of the "Instacart dataset." The main goal of this project is to plan and provide a data warehouse solution for corporate utilization, enabling efficient storage, maintenance, as well as analytics of organizational data.

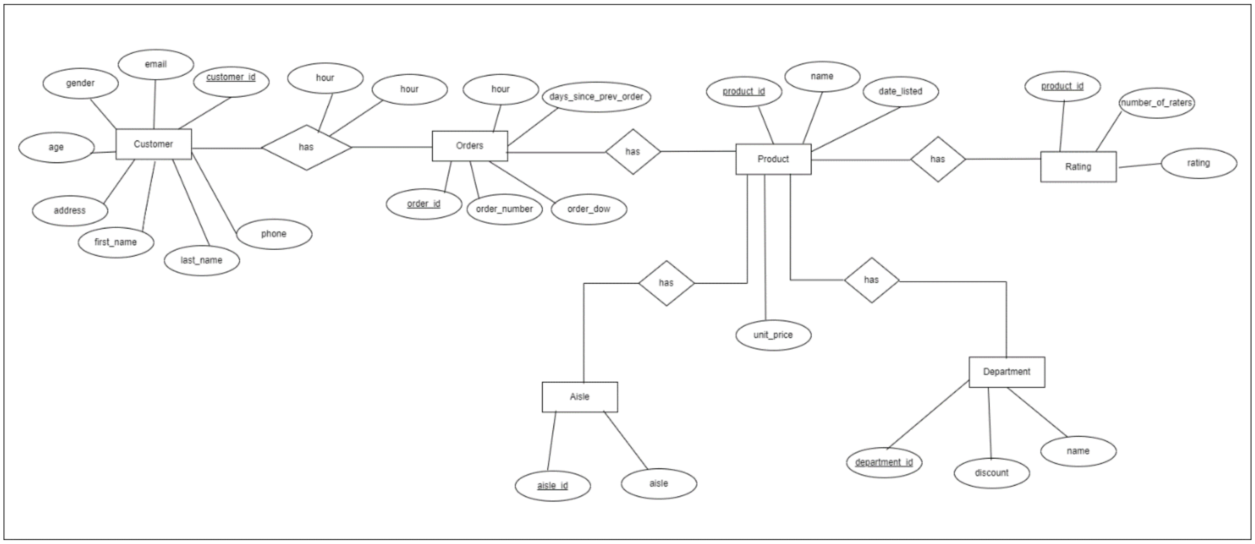
2. Data Selection

2.1 Overview

Data is the foundation of all decision-making techniques in the commercial world. In order to obtain reliable results, businesses must have sufficient data to process. The Instacart DataMart Analysis dataset found on Kaggle is used in this project. To adapt to the given context, I added additional datasets to increase the complexity of the Datawarehouse.

➢ Main Dataset – https://www.kaggle.com/c/instacart-market-basket-analysis/data

➢ Ratings dataset was developed using Excel



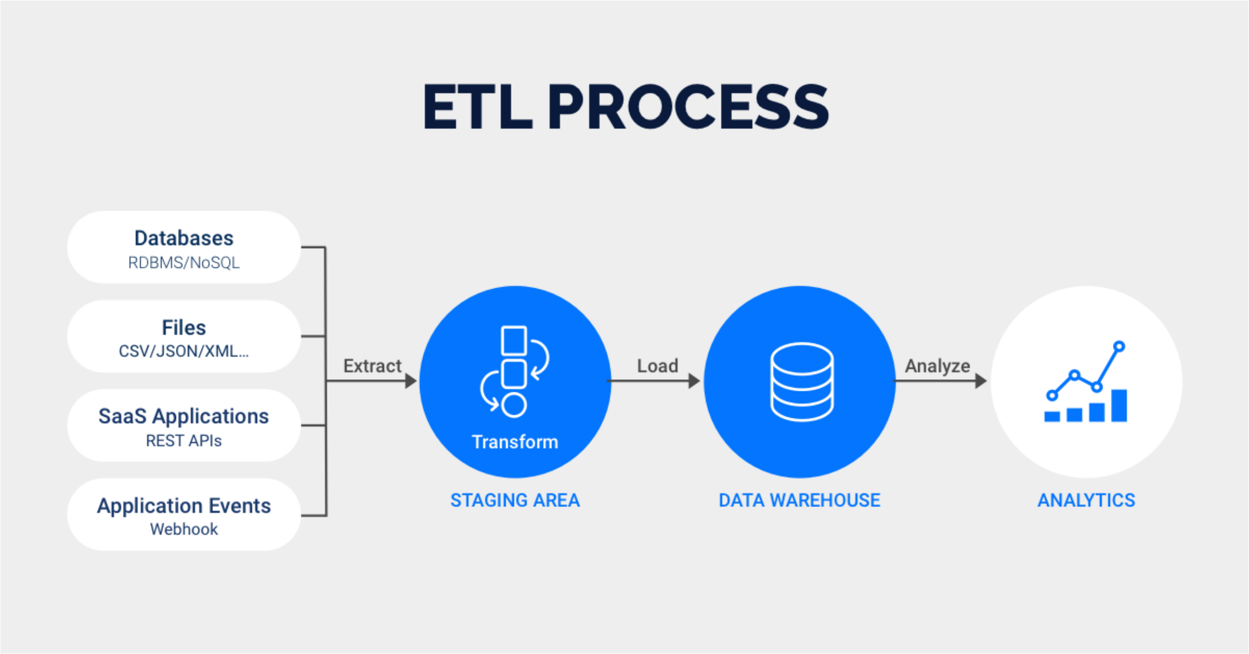
3.Data Preparation

Initially, the original data files were available in different formats: Excel, CSV, TXT, and SQL. These files were then processed and organized into several tables for further use.

Four types of data sources were utilized: excel, csv, txt, and database.

1. **.xlsx (Excel)** –
   The Product data were kept in an Excel source file.
   This table contains detailed information about the products available for customers, including product names, categories, and related attributes.
2. **.csv** –
   Multiple datasets were saved in CSV format, including Orders, Aisle, Department, and OrderDetails.
   These files contain information about customer orders, the categorization of products into aisles and departments, and detailed breakdowns of each order transaction.
3. **.txt** –
   Specific supplementary data were saved in a text file format.
   (You can mention here if you had a specific file, otherwise this line can be omitted.)
4. **Database (SQL)** –
   A source database was created by importing customer-related data from a SQL file.
   a. **Customer** – This table contains customer profiles, including demographic details such as age, gender, address, and contact information, used to manage and analyze customer interactions.

4. Solution Architecture

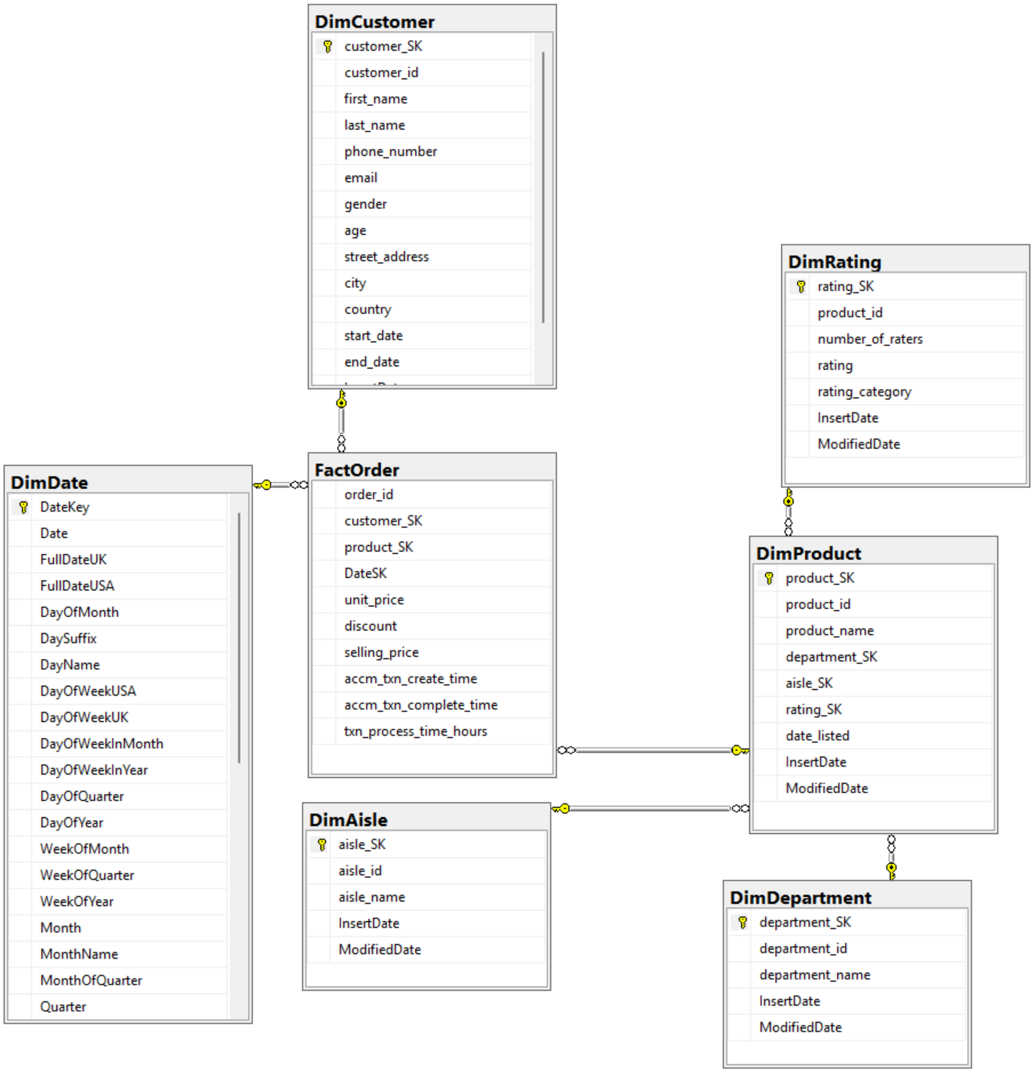


1. **Data Sources –**
   The initial step towards setting a good architecture involves gathering information from a number of different sources of data including CRM, ERP, databases, files, or APIs based on the purpose and the available resources.
   In the situation provided, there are three principal sources of data: a source database, Excel files, and flat files (CSV).
   Source database (Customer.sql) – includes the Customer table that stores detailed customer information.
   Excel document (Product.xlsx) – includes the Product table, wherein there exists detailed product information including names, categories, as well as attributes.
   Plain text files –
   - Orders.csv (customers' orders information)
   - Aisle.csv - aisle categorization of products
   - Department.csv (categorization of products by departments)
   - OrderDetails.csv (order detailed transaction data)
2. **Staging Area –**
   The data staging area serves as a temporary storage facility between the data sources and the data warehouse. The main purpose of a staging area is for quick extractions of

the data from the data sources without creating much of an impact on the sources.
In this situation, instacart_Staging serves as a database that acts as a data staging area.

3. **Data Warehouse –**
   A data warehouse is a vast reservoir of company data that supports improved internal decision-making . It contains a lot of historical data.
   A database file called instacart_DW serves as the data warehouse within this context. The data warehouse includes a number of dimension tables (Product, Customer, Aisle, Department, Date) as well as a FactOrder table that stores the transactional sales details.
4. **Business Intelligence Solution (Consumption)**
   This uses technology and services to turn information into actionable insights that inform improved decision-making by organizations.
   Within this scenario's business solution, the information might be examined, displayed, and reported in order to gain insights into customer buying habits, sales trends, product demand, and other key business information.
5. **ETL –**
   ETL (Extract, Transform, and Load) is a data integration process that brings together data from multiple sources into a common, coherent data repository that can then be fed into a data warehouse or another target system.
   ETL operations in this scenario extract the data from Excel, CSV, and SQL sources, transforming the information into a standardized form and populating the instacart_DW database and eventually the instacart_DW data warehouse.

5. Data Warehouse Design and Development



The following is the dimensional model applied within the provided scenario. Generally speaking, the dimensional model consists of 5 dimensional tables (such as the date dimension) and one fact table.

- **Schema used – Snowflake Schema**

The snowflake schema was used within the dimensional modeling to minimize redundancy through normalization. You can see the customer dimension table was normalized.

- **Fact and Dimension Tables**

They set up five tables of dimensions and a fact table:

1. **DimCustomer** – The customer dimension holds customer details. The surrogate key is Customer_SK.
2. **DimProduct** – The product dimension holds detailed product information. Product_SK is the surrogate key.
3. **DimAisle** – The aisle dimension holds data on the product categorization by aisle. The surrogate key is Aisle_SK.
4. **DimDepartment** – This dimension holds data regarding the product categorization by departments. Department_SK serves as the surrogate key.
5. **DimDate** – It's a typical date dimension accommodating time-based analysis. The surrogate key is DateKey. The date dimension was created using an SQL script.
6. **FactOrder** – Holds all of the sales information of the transactions. It uses the dimension tables through foreign keys.

- **Slowly Changing Dimensions**

Customer information was presumed to evolve over time and thus was perceived as a Slowly Changing Dimension.

Assumption –

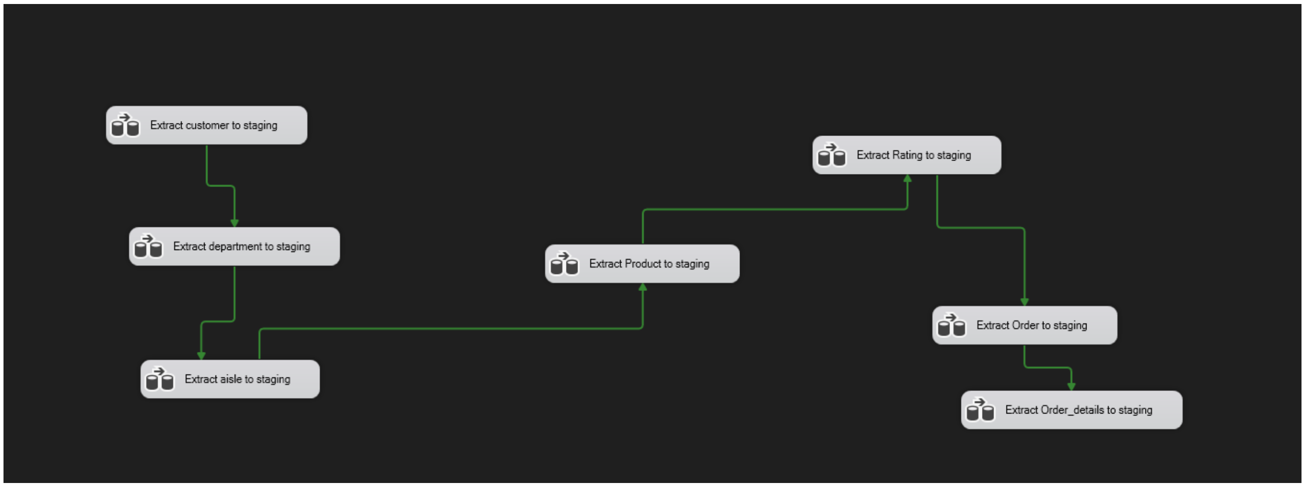
Because the customer's city, address, and phone number are the types of attributes that the customer may eventually update over a period of time, the DimCustomer table as a slowly changing dimension was created to track the changes.

▪ **DimCustomer** – Such dimensions as city, country, street_address, and phone_number are historical dimensions that may vary over time.

6. ETL Development

6.1 Extract Data from Source to Staging

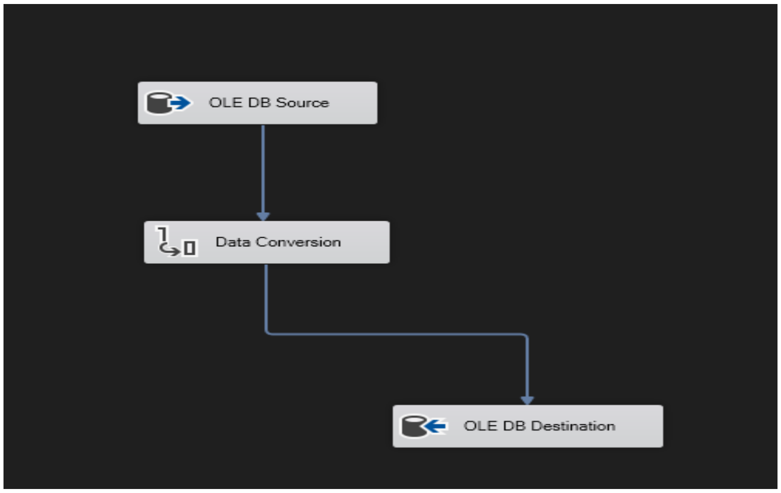
- Order of Execution


Extract customer to staging
Extract department to staging
Extract aisle to staging
Extract Product to staging
Extract Rating to staging
Extract Order to staging
Extract Order_details to staging

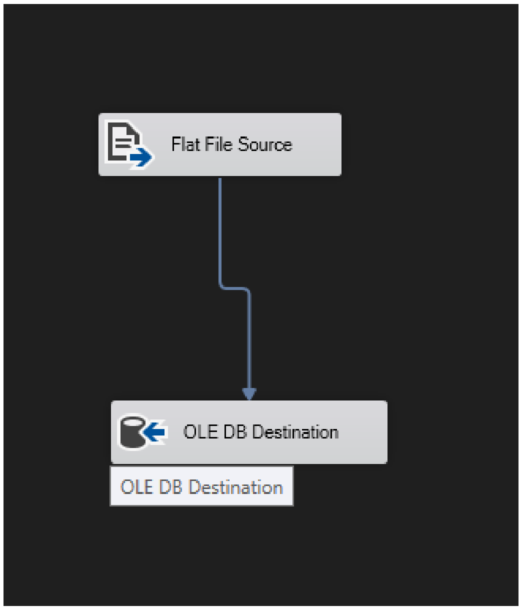
Data is extracted from various data sources and is staged in an intermediate location until being loaded into the data warehouse. Individual extractions into the staging database happens as below images.

6.1.1 Customer data are extracted from a source database and is loaded into staging area

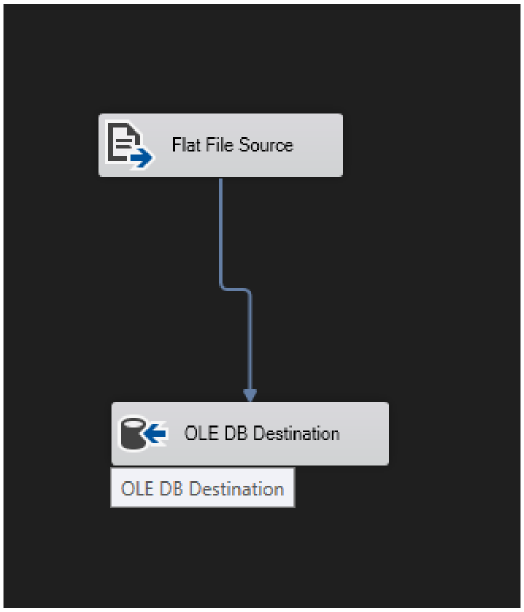
A Data Conversion transformation is applied to convert columns like first_name, last_name, phone_number, email, etc., into the required Unicode string (DT_WSTR) format to match the schema of the staging table.



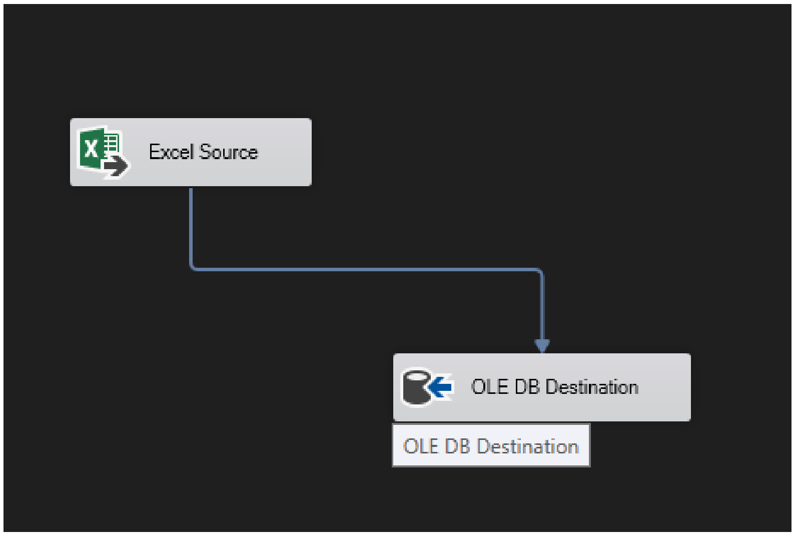
6.1.2 Department data are extracted from a flat file source and is loaded into staging area



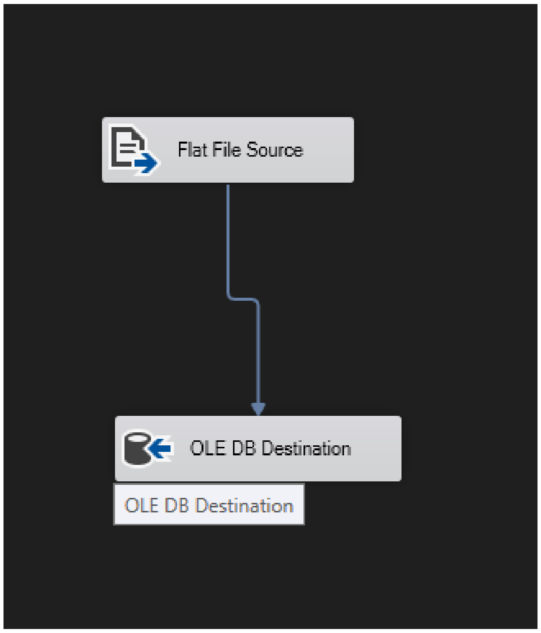
6.1.3 Aisle data are extracted from a flat file source and is loaded into staging area



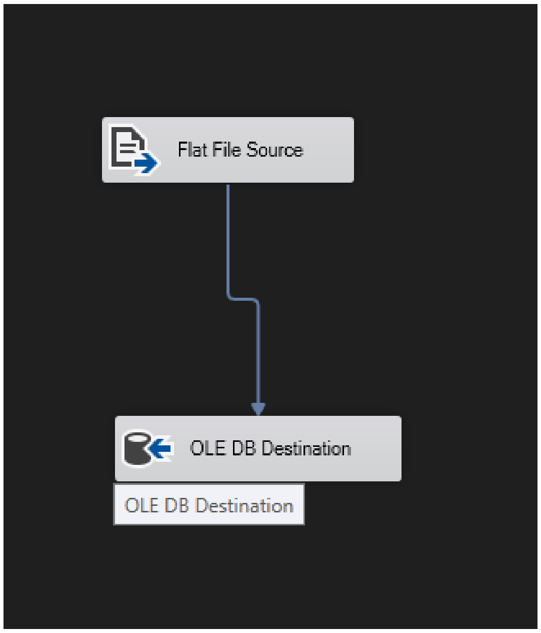
6.1.4 Product data are extracted from a Excel file source and is loaded into staging area



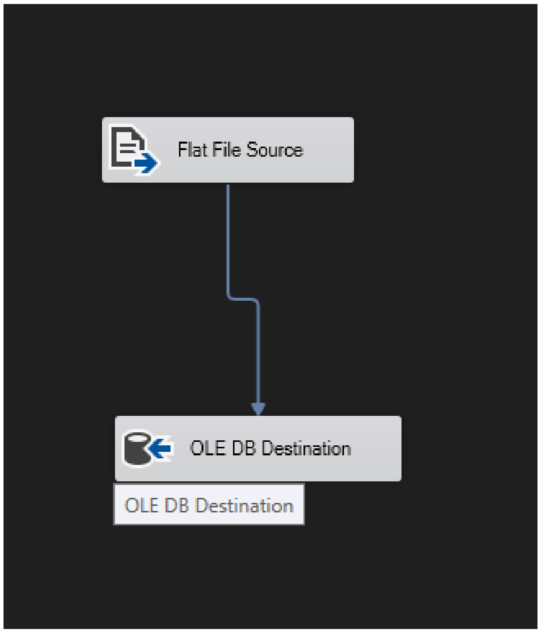
6.1.5 Rating data are extracted from a flat file source and is loaded into staging area



6.1.6 Order data are extracted from a flat file source and is loaded into staging area

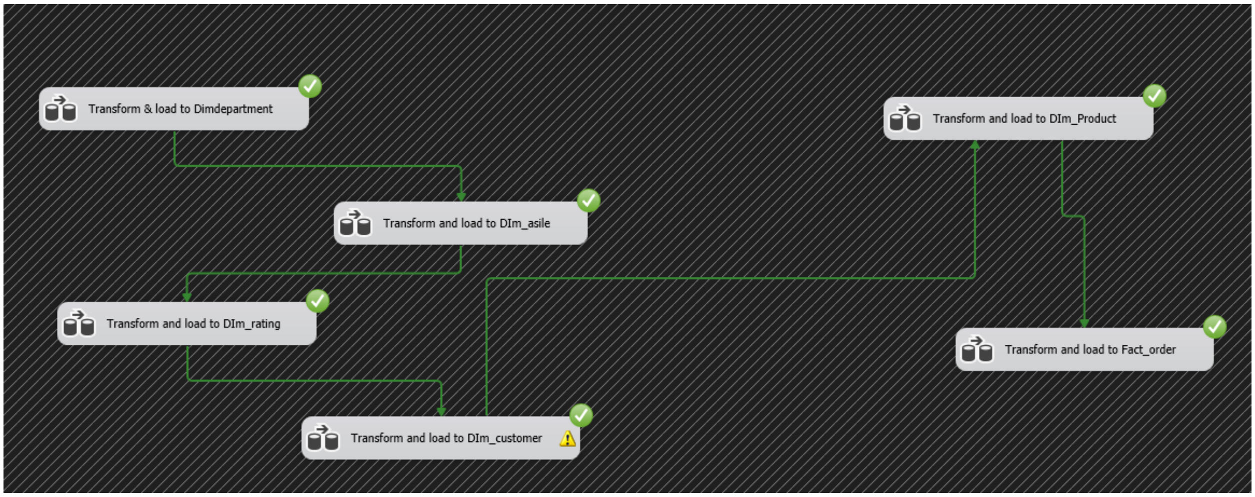


6.1.7 Order Details data are extracted from a flat file source and is loaded into staging area

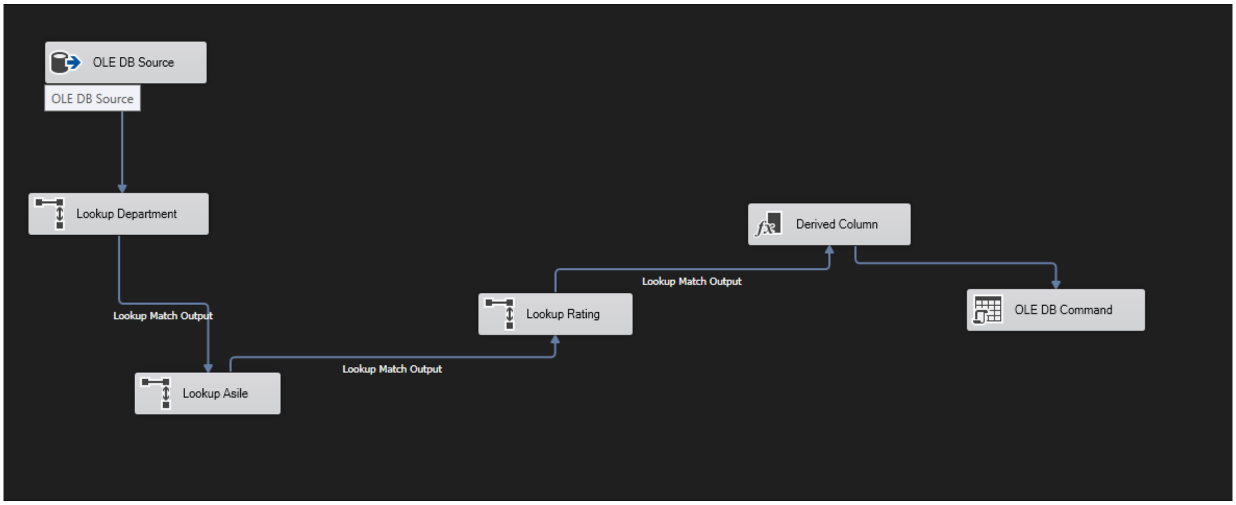


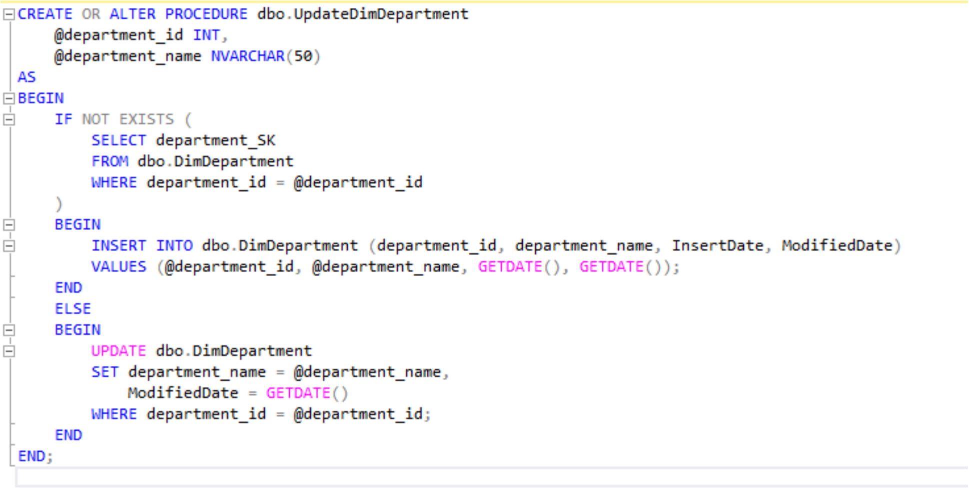
6.2 Transforming and Loading to the Datawarehouse

Once the data was staged, it was transformed and loaded into the data warehouse utilizing a Snowflake Schema design. The schema contains five dimension tables and a fact table. The structure was applied in SQL Server, and stored procedures were created to maintain data integrity and avoid redundancy. The Date Dimension was also created and populated by a custom SQL procedure. The SSIS package shown above depicts the transformation and load process from the staging tables to the warehouse.


Transform & load to Dimdepartment
Transform and load to DIm_asile
Transform and load to DIm_rating
Transform and load to DIm_customer
Transform and load to DIm_Product
Transform and load to Fact_order

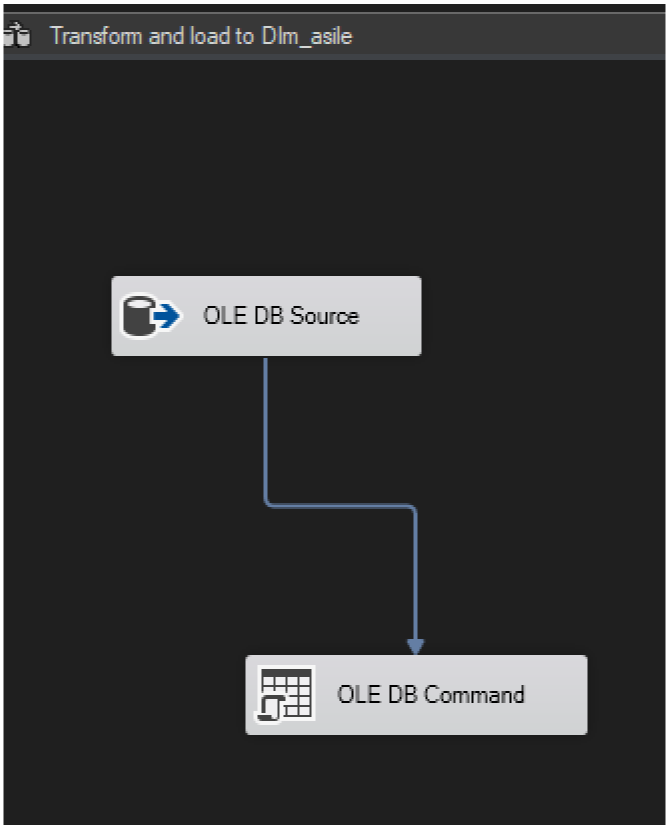
6.2.1 Transform & Load to DimDepartment

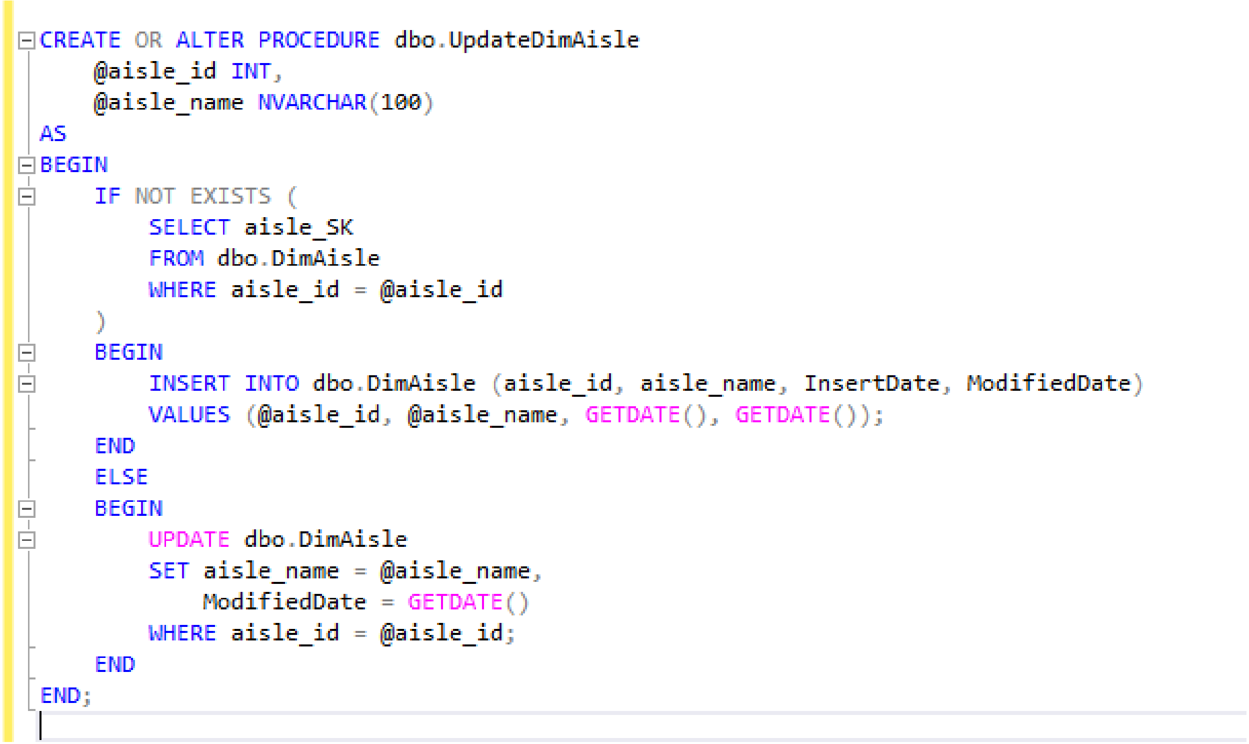




```
CREATE OR ALTER PROCEDURE dbo.UpdateDimDepartment
    @department_id INT,
    @department_name NVARCHAR(50)
AS
BEGIN
    IF NOT EXISTS (
        SELECT department_SK
        FROM dbo.DimDepartment
        WHERE department_id = @department_id
    )
    BEGIN
        INSERT INTO dbo.DimDepartment (department_id, department_name, InsertDate, ModifiedDate)
        VALUES (@department_id, @department_name, GETDATE(), GETDATE());
    END
    ELSE
    BEGIN
        UPDATE dbo.DimDepartment
        SET department_name = @department_name,
            ModifiedDate = GETDATE()
        WHERE department_id = @department_id;
    END
END;
```

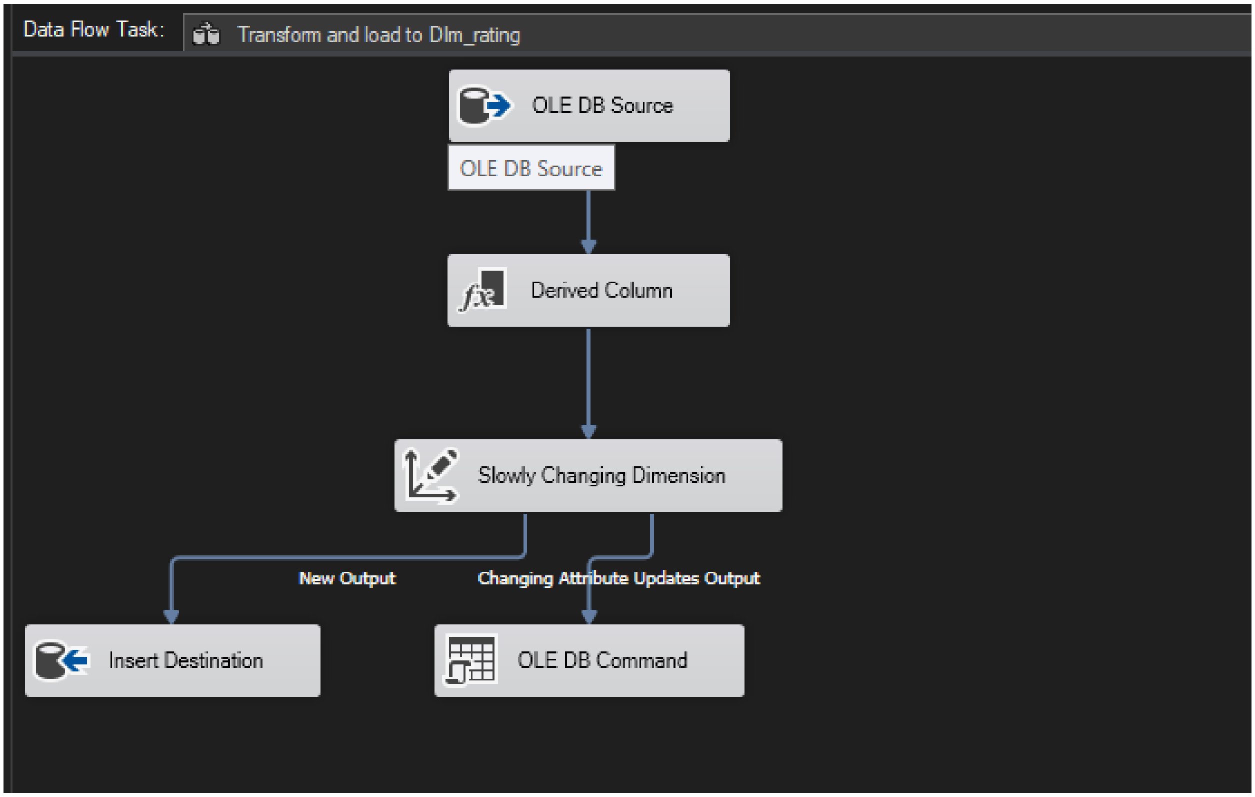
6.2.2 Transform & Load to Dim Aisle





```
CREATE OR ALTER PROCEDURE dbo.UpdateDimAisle
    @aisle_id INT,
    @aisle_name NVARCHAR(100)
AS
BEGIN
    IF NOT EXISTS (
        SELECT aisle_SK
        FROM dbo.DimAisle
        WHERE aisle_id = @aisle_id
    )
    BEGIN
        INSERT INTO dbo.DimAisle (aisle_id, aisle_name, InsertDate, ModifiedDate)
        VALUES (@aisle_id, @aisle_name, GETDATE(), GETDATE());
    END
    ELSE
    BEGIN
        UPDATE dbo.DimAisle
        SET aisle_name = @aisle_name,
            ModifiedDate = GETDATE()
        WHERE aisle_id = @aisle_id;
    END
END;
```

6.2.3 Transform & Load to Dim Rating



The SCD transformation and load process of the DimRating table are implemented for handling Slowly Changing Dimensions through the use of SSIS.

Derived Column – rating_category

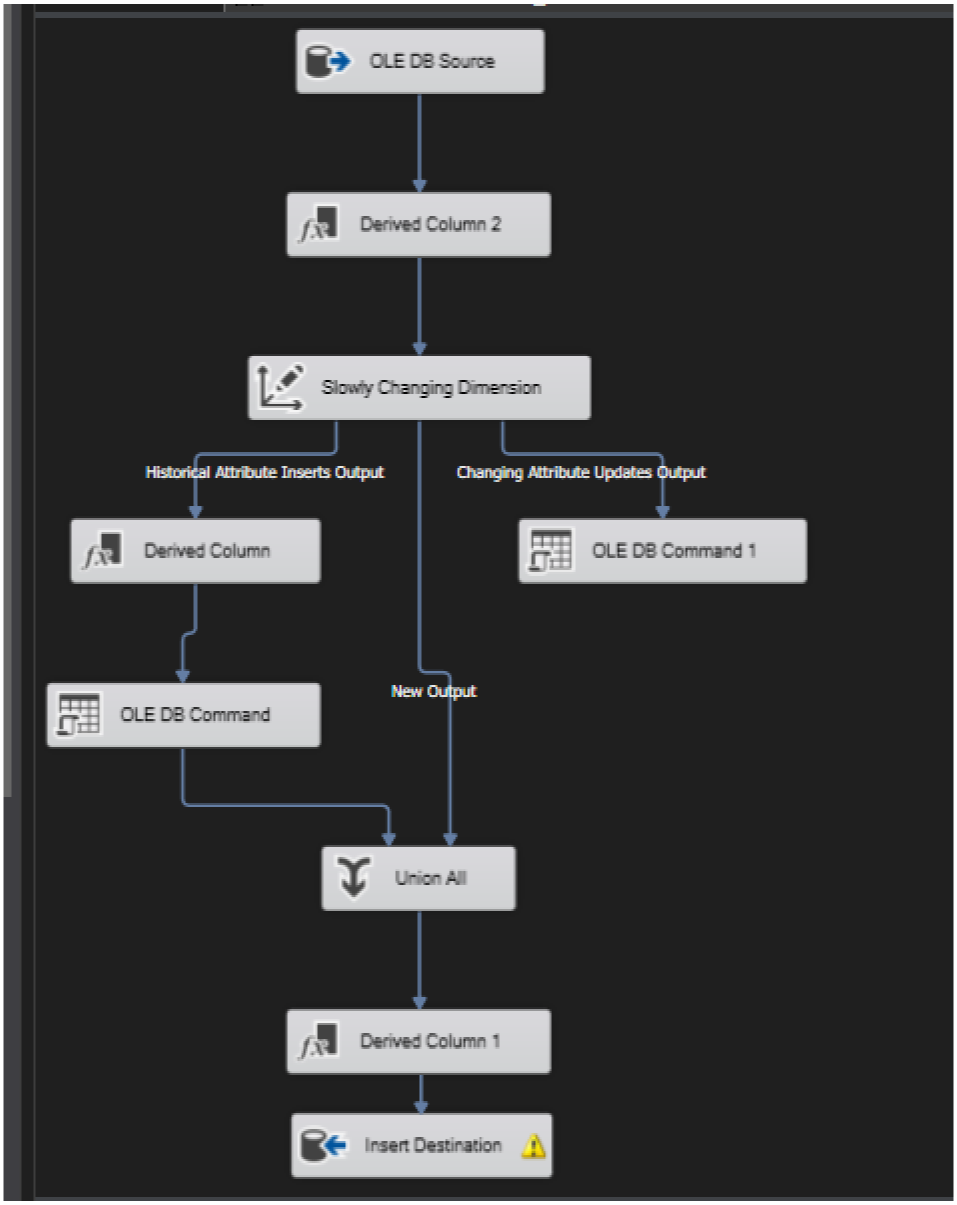
A new column called *rating_category* was created using an SSIS expression to categorize customer ratings. The logic applied is:

- If the rating is 4.5 or higher, it is categorized as **"Excellent"**.
- If the rating falls between 3 (inclusive) and 4.5, it is categorized as **"Average"**.
- Ratings below 3 are classified as **"Poor"**.

Slowly Changing Dimension (SCD) Component

- Manages the updates to dimension data.
- Both *rating* and *rating_category* columns are defined as changing attributes (Type 1 SCD), meaning that any updates to these fields overwrite the existing values.

6.2.4 Transform & Load to Dim Customer



The load and transformation process for the DimCustomer table supports both changing and historical attributes using the Slowly Changing Dimension (SCD) feature of SSIS.

The following configurations were applied through the Slowly Changing Dimension Wizard:

- **Changing Attributes (Type 1 SCD)**

These fields are directly updated when modifications occur:

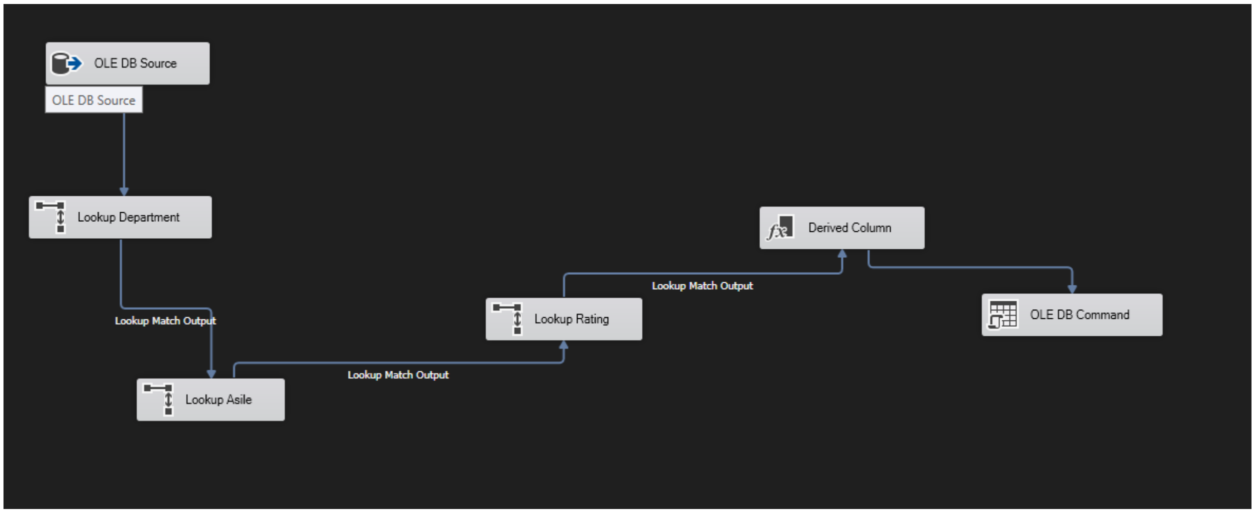
- age
- first_name
- last_name
- phone_number

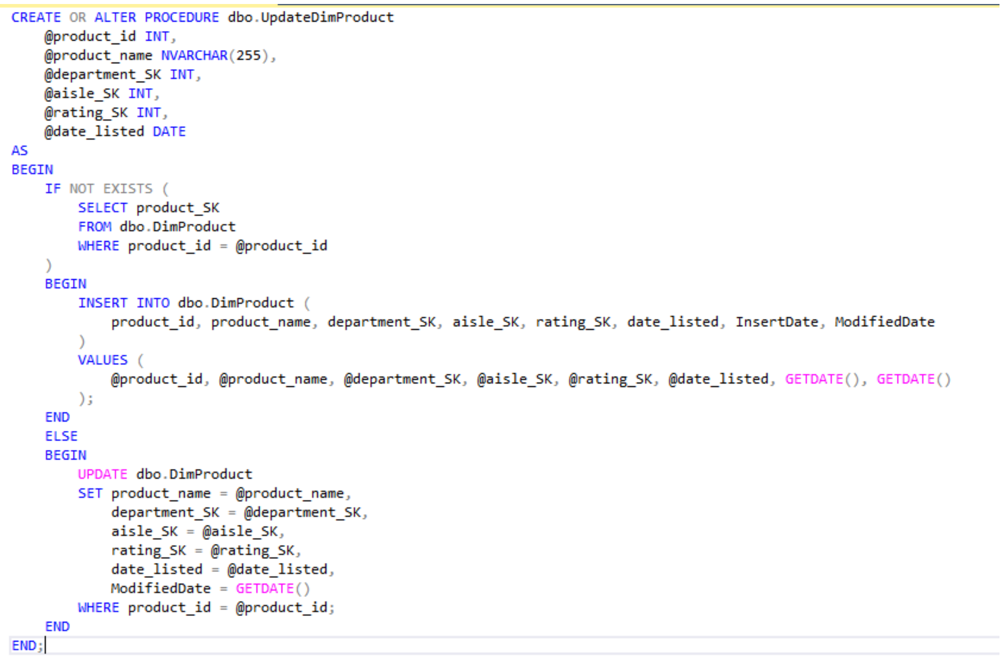
- **Historical Attributes (Type 2 SCD)**

These fields are tracked by inserting new records to maintain a historical record:

- city
- country
- email
- street_address

6.2.5 Transform & Load to Dim Product





```
CREATE OR ALTER PROCEDURE dbo.UpdateDimProduct
    @product_id INT,
    @product_name NVARCHAR(255),
    @department_SK INT,
    @aisle_SK INT,
    @rating_SK INT,
    @date_listed DATE
AS
BEGIN
    IF NOT EXISTS (
        SELECT product_SK
        FROM dbo.DimProduct
        WHERE product_id = @product_id
    )
    BEGIN
        INSERT INTO dbo.DimProduct (
            product_id, product_name, department_SK, aisle_SK, rating_SK, date_listed, InsertDate, ModifiedDate
        )
        VALUES (
            @product_id, @product_name, @department_SK, @aisle_SK, @rating_SK, @date_listed, GETDATE(), GETDATE()
        );
    END
    ELSE
    BEGIN
        UPDATE dbo.DimProduct
        SET product_name = @product_name,
            department_SK = @department_SK,
            aisle_SK = @aisle_SK,
            rating_SK = @rating_SK,
            date_listed = @date_listed,
            ModifiedDate = GETDATE()
        WHERE product_id = @product_id;
    END
END;
```

6.2.6 Transform & Load to Fact Order

The SSIS package illustrated above manages the transformation and loading of data into the **FactOrders** table. This process involves several lookups, derived columns, and data formatting steps to maintain accuracy and consistency within the data warehouse.

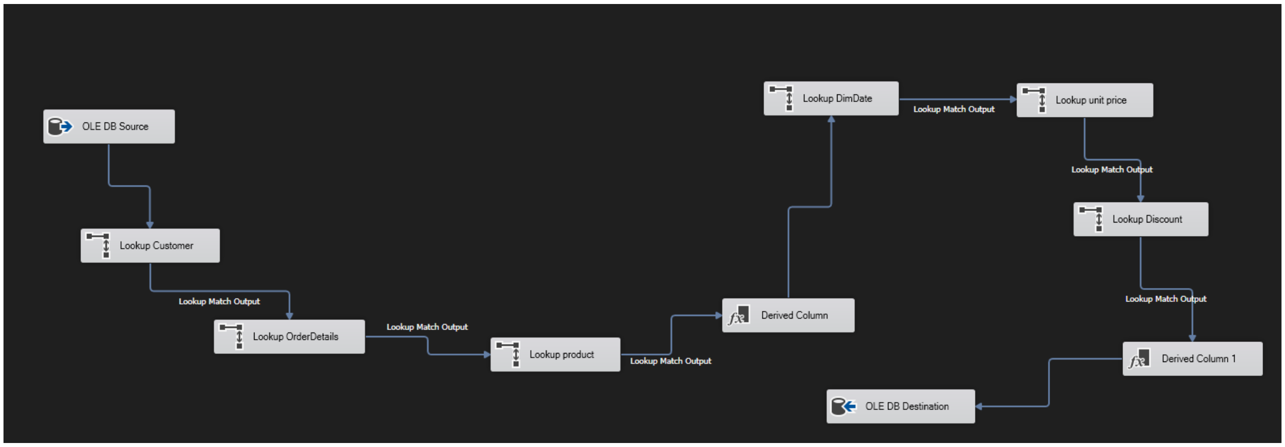
Lookup Transformations:

- **Order Detail Lookup:** Matches each record with the correct transactional order data.
- **Customer Lookup:** Fetches the appropriate surrogate key from the **DimCustomer** table.
- **Product Lookup:** Connects each order to its related product found in **DimProduct**.
- **Unit Price Lookup:** Retrieves the unit price for each product.
- **Discount Lookup:** Collects any applicable discount percentages.
- **DimDate Lookup:** Converts the order date to its corresponding surrogate key in the **DimDate** table.

Derived Columns:

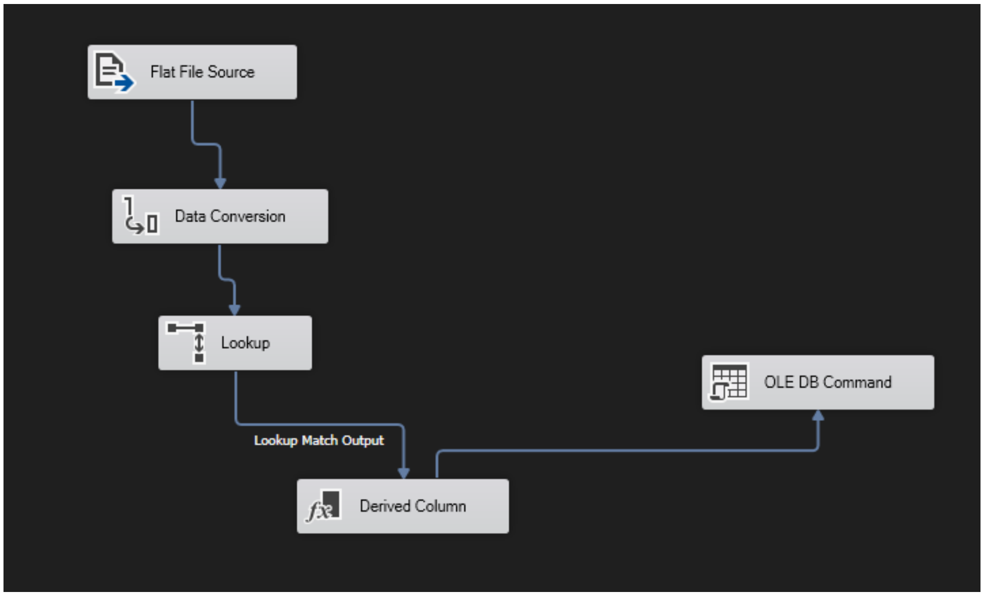
1. **Selling Price Calculation:**
   A new column named *selling_price* was created by applying a specific formula.
2. **Formatted Order Date:**
   A transformation was applied to format the *order_date* to match the format used in the **DimDate** table.

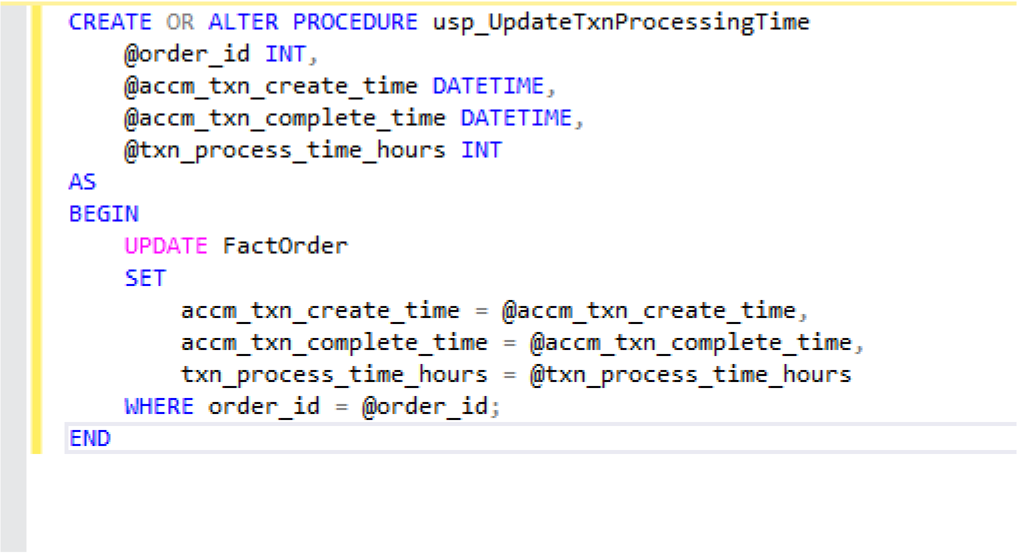
Purpose:
This ensures accurate matching by adding leading zeros to single-digit months and days (for example, changing "4/2/2025" to "04/02/2025").


OLE DB Source
Lookup Customer
Lookup Match Output
Lookup OrderDetails
Lookup Match Output
Lookup product
Lookup Match Output
Derived Column
Lookup DimDate
Lookup Match Output
Lookup unit price
Lookup Match Output
Lookup Discount
Lookup Match Output
Derived Column 1
OLE DB Destination

7.Accumilating Fact Table

The process time in hours is determined by calculating the difference between the create time and complete time, and it is then updated in the current fact table. This operation is handled in a separate package where a derived column computes the process time and updates both the **txn_process_time_hours** and **accm_txn_complete_time** columns using the calculated values along with data from an external flat file.





```sql
CREATE OR ALTER PROCEDURE usp_UpdateTxnProcessingTime
    @order_id INT,
    @accm_txn_create_time DATETIME,
    @accm_txn_complete_time DATETIME,
    @txn_process_time_hours INT
AS
BEGIN
    UPDATE FactOrder
    SET
        accm_txn_create_time = @accm_txn_create_time,
        accm_txn_complete_time = @accm_txn_complete_time,
        txn_process_time_hours = @txn_process_time_hours
    WHERE order_id = @order_id;
END
```