VIETNAMESE GERMAN UNIVERISTY
Department of Computer Science and Engineering

"Current Topic in Computer Science" Module

# BUSINESS INTELLIGENCE SYSTEM - ANALYSIS

Instructor: Dr. Phan Trong Nhan

| | | |
|---|---|---|
| *Student 1:* | Luu Minh Khang | 10421024 |
| *Student 2:* | Bui Van Hoang Vu | 10421127 |
| *Student 3:* | Nguyen Tien Khoa | 10421085 |
| *Student 4:* | Le Nguyen Quynh An | 10421118 |
| *Student 5:* | Nguyen Xuan Tien | 10421058 |

December 22, 2024

# Content

# 1 Introduction

## 1.1 Project Overview

The primary objective of this project is to design and implement a Business Intelligence (BI) system for product mix analysis. A Business Intelligence System covers the process of collecting, analyzing and processing data into usable information and presentation. This project focuses on the integration of business intelligence systems by analyzing the sales performance of product line over time and segmenting the customer base on their purchase behavior. We aim to use data-driven insights to improve product mix strategies and efficiently target different customer segments by building an efficient data warehouse. The major stages of the projects may be listed as follows:

1. Data source analysis and understanding
2. Gather business requirements
3. Data warehouse design
4. Extract, Transform, Loading (Pipeline) implementation
5. Near real-time updating
6. Visualization for actionable insights

## 1.2 Objectives

From engineering aspect, this project aims for the following achievement:

- Building a data warehouse by combining relevant business data.
- Presenting the information consistently and in a timely way.
- Adapting to changes related to new data or requirements.

Furthermore, from business perspective, the following targets are expected:

- Constructing an easily accessible Business Intelligence system.
- Providing actionable insights that help business in improving decision-making.

# 2 Topic Investigation

## 2.1 Related Concepts

### 2.1.1 Product Mix

Product mix, also known as product assortment, refers to a company's entire range of products and services available to its clients. It includes all product lines, categories, and individual items that a business advertises to fulfill a variety of customer desires and preferences. A well-structured product mix is critical for companies to cater to numerous market segments, maximize revenue, and maintain a competitive advantage.
The product mix has four key dimensions:

1. Width: The amount of distinct product lines that a business provides.
2. Length: The sum of the items in each product line.
3. Depth: A product's range of variations or versions within a single product line.
4. Consistency: The degree of similarity between product lines in terms of use, production, and distribution.

By strategically managing product mix, companies can address customer demands, adapt to market trends, and optimize profitability. Understanding the concept of the product mix is crucial for creating effective marketing strategies, aligning offerings with organizational goals, and ensuring long-term business success.

### 2.1.2 Data Warehouse

This is a centralized repository where data from different sources is consolidated, cleaned, and stored for analysis. It enables aggregation of sales, customer feedback, inventory, market trend data, etc., and also facilitates tracking performance across the product mix over time.

### 2.1.3 Business Intelligence - Analytics

Business Intelligence (BI) is a technology-driven process for analyzing data and delivering actionable information that helps executives, managers and workers make informed business decisions.
Business intelligence system is essentail for most of business today. It helps businesses not only visualize their data, measure their performance of all kinds, but also optimize their operations.
Other than business intelligence, business analytics facilitates businesses with true insights

hidden from their operational data and supports decision making. In order to achieve the goals, a series of system designs and solutions have to be examined and developed from the business transactional data.

#### 2.1.4 Data Mining

This is a process of discovering patterns, correlations, and trends within large datasets using statistical and machine learning techniques. It helps predict customer behavior and identifies cross-selling opportunities.
In this project, we stay focus on using clustering - a data mining technique that - according to IBM - the International Business Machines Corporation- which is the largest industrial research organization in the world - "Clustering is an unsupervised machine learning algorithm that organizes and classifies different objects, data points, or observations into groups or clusters based on similarities or patterns."
Clustering can be used to explore data, find underlying trends, and reduce the complexity of large datasets. There are many different types of clustering algorithms, including centroid-based, hierarchical, distribution-based, density-based, and grid-based. Each algorithm has its own strengths and weaknesses, and the best algorithm for a particular dataset will depend on the characteristics of the data.

## 2.2 Business Metrics

To evaluate and optimize a good product mix, companies rely on many key metrics that provide insights into performance, profitability, customer behavior, and market dynamics. Below are some of the most relevant business metrics and their significance:

### 2.2.1 Sales Revenue

Sales revenue is the income received by a company from its sales of goods or the provision of services. This is the total income generated from selling specific products or services.
Formula:

$$\textbf{Sales Revenue} = \textstyle\sum_{allproducts}(\textbf{Quantity Sold} \times \textbf{Selling Price})$$

### 2.2.2 Gross Profit

Gross profit or gross income, is a company's profit after deducting the costs associated with producing and selling its products or services.This helps identify which products are the most profitable and worth retaining, or else they are highly discontinued.
Formula:

$$\textbf{Gross Profit} = \textbf{Sales Revenue - Cost of Goods Sold (COGS)}$$

*where*

$$COGS = QuantitySold \times UnitCost$$

### 2.2.3 Sales Volume Growth Rate

Sales volume growth rate measures your company's ability to generate revenue through sales over a fixed period of time. This indicates the popularity of the product line and reveals market demand trends. Rapid growth in sales volume can signal high demand, while a declining rate may suggest the need for changes in marketing or product strategy.
Formula:

$$\textbf{SVGR (\%)} = \frac{\textbf{(Current Period Sales}-\textbf{Prior Period Sales)}}{\textbf{Prior Period Sales}} \times \textbf{100}$$

### 2.2.4 Average Selling Price

The average selling price (ASP) is a term that refers to the average price a good or service is sold for. This is used to highlight pricing trends and their impact on profitability, and also determine whether price adjustments are needed to stay competitive or increase margins.
Formular:

$$\textbf{ASP} = \frac{\textbf{Product Revenue}}{\textbf{Total Units Sold}}$$

### 2.2.5 RFM Metrics

On the customer side, we will use the RFM model (Recency, Frequency, Monetary Value) for this purpose. This model can analyze customer behavior based on how recently they purchased (Recency), how often they purchase (Frequency), and how much they spend (Monetary Value). Customers are segmented into multiple groups based on their RFM scores, and we can apply these to identify which products are popular with each segment and also to adjust the perfect product mix.

- For High RFM Customers: This focuses on high-margin or premium products that customers already buy or are likely to buy.
- For Low RFM Customers: This offers entry-level or discounted products for customers who may respond better to discounts or essentials.
- For Specific RFM Segments: This will tailor the product mix to align with each segment's preferences and behaviors (e.g., prioritize fast-moving products for frequent purchasers).

RFM score is calculated based upon recency, frequency, monetary value normalize ranks. Based upon this score we divide our customers. Here we rate them on a scale of 5. Formula used for calculating RFM score is:

$$\mathbf{0.15 \times Recency + 0.28 \times Frequency\ Score + 0.57 \times Monetary\ Score}$$

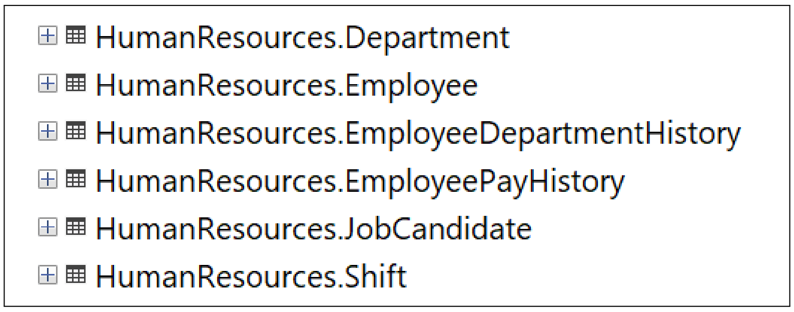
By using the RFM model, companies can maximize their sales revenues from prioritizing the right products for specific buyers which can lead to improved customer satisfaction and loyalty. Moreover, businesses can avoid wasting resources promoting low-margin or irrelevant products to segments unlikely to buy them, and also predict stocking and inventory status to make better decisions.

# 3 Data Source Analysis

## 3.1 Data Source Collection

The provided dataset "CompanyX" consists of 5 major groups of tables. The tables that belong to one group are named after with the same prefix name of that group. The five groups are:

1. **HumanResources**
   Tables in this group relate to employee information, such as hiring, job roles, payroll data, staff shift and job candidates,... Since the topic is product mix, after some careful consideration, we determined that this group is not related to the topic, therefore no tables in this group are selected for the implementation.



HumanResources.Department
HumanResources.Employee
HumanResources.EmployeeDepartmentHistory
HumanResources.EmployeePayHistory
HumanResources.JobCandidate
HumanResources.Shift

Figure 1: List of HumanResources tables

2. **Person**
   This group focuses on individual details, such as customer, store contact, sales person, vendor contact and general contact's profiles and contacts. There are also tables that contain information about business entities, which have connection to other Person's tables.

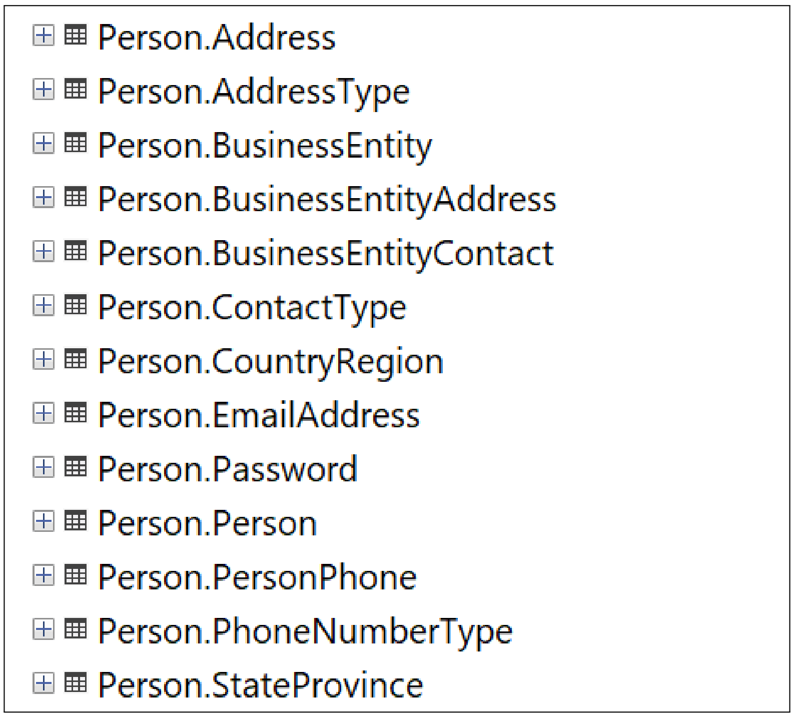

Person.Address
Person.AddressType
Person.BusinessEntity
Person.BusinessEntityAddress
Person.BusinessEntityContact
Person.ContactType
Person.CountryRegion
Person.EmailAddress
Person.Password
Person.Person
Person.PersonPhone
Person.PhoneNumberType
Person.StateProvince

Figure 2: List of Person tables

3. **Production**

   These tables cover aspects of manufacturing, inventory, and product specifications. This group contains a huge amount of information which is partly proved by the much larger number of tables compared to other groups. Since the topic "Product mix" is extremely related, the group should be examined carefully. It is necessary to identify how the business defines their own product mix. To do so, every column in specific tables is required to be explored. It is not easy to construct the company's product mix since all we have is just the OLTP dataset, but no description from the company's department. Noticeably, the following tables are strictly involved in the product mix: *Production.Product*, *Production.ProductCategory*, *Production.Subcategory*, and *Production.ProductModel*. Further investigation is mentioned in the next section.

4. **Purchasing**

   Tables in this group manage procurement processes, vendor information, and purchase orders. These tables show the data from the business to business side, and because we need only the data from the perspective of business to customers, the exploration into this group is later neglected.

5. **Sales**

   This group is centered on transactional data, customer orders, and sales-performance-related data. The group consists of necessary tables for the analysis, therefore, it is

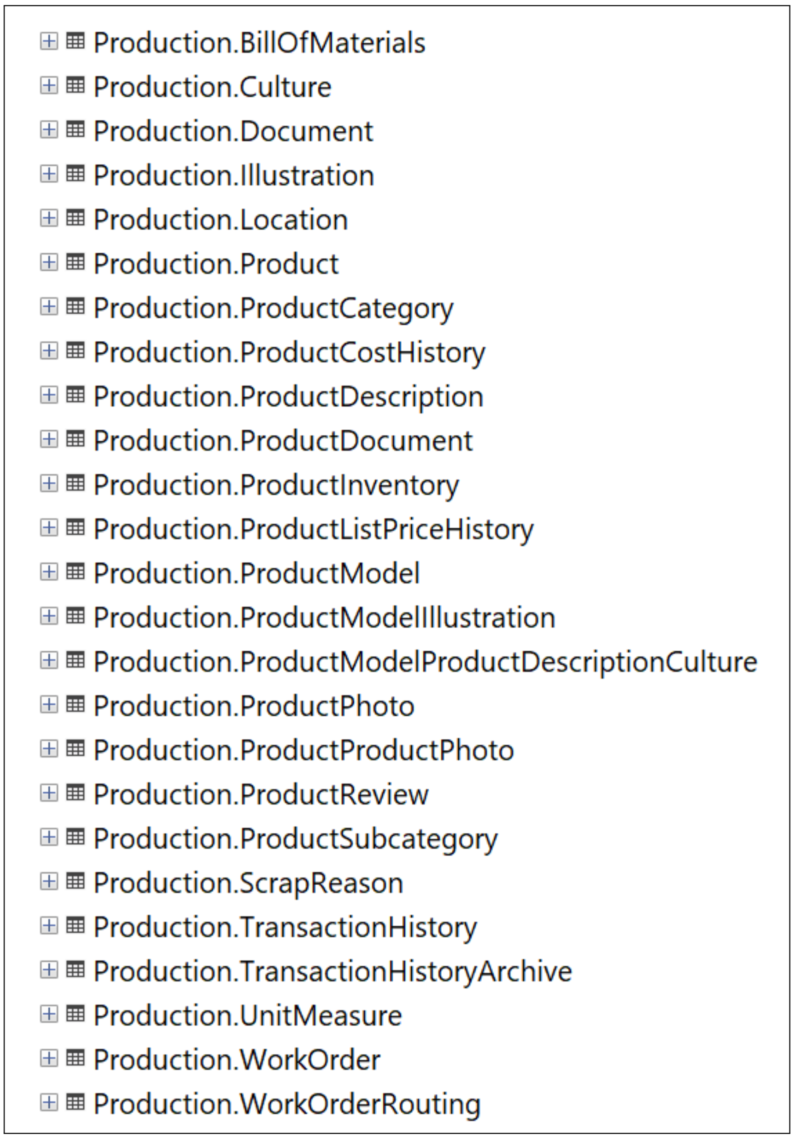

Production.BillOfMaterials
Production.Culture
Production.Document
Production.Illustration
Production.Location
Production.Product
Production.ProductCategory
Production.ProductCostHistory
Production.ProductDescription
Production.ProductDocument
Production.ProductInventory
Production.ProductListPriceHistory
Production.ProductModel
Production.ProductModelIllustration
Production.ProductModelProductDescriptionCulture
Production.ProductPhoto
Production.ProductProductPhoto
Production.ProductReview
Production.ProductSubcategory
Production.ScrapReason
Production.TransactionHistory
Production.TransactionHistoryArchive
Production.UnitMeasure
Production.WorkOrder
Production.WorkOrderRouting

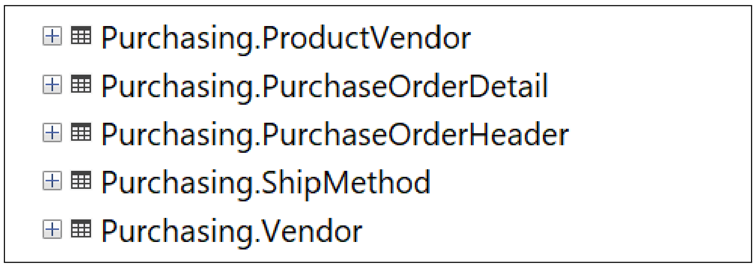
Figure 3: List of Production tables

considered one of the most important groups for this project.

## 3.2 Feasibility Analysis

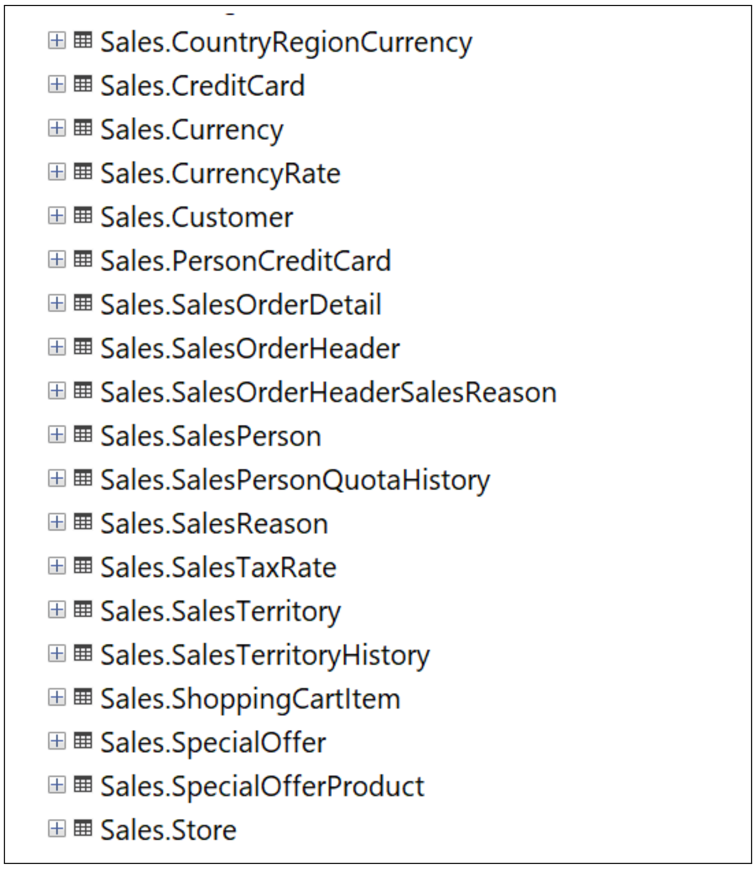
From the database of Company X, we have checked and selected the necessary tables which include the required attributes for analysis and calculations. We temporarily divided the tables into two groups: Product and Sales. All of the tables can be seen below:

- **Production.Product**
  This table contains detailed information about each product, such as its name, product number, cost, price, size, color, manufacturing time, and categorization.
  Reason for selection: The table serves as the main dataset for understanding the characteristics and pricing of individual products. It is crucial for analyzing which product lines perform well and how they contribute to the overall product mix.



Purchasing.ProductVendor
Purchasing.PurchaseOrderDetail
Purchasing.PurchaseOrderHeader
Purchasing.ShipMethod
Purchasing.Vendor

Figure 4: List of Purchasing tables



Sales.CountryRegionCurrency
Sales.CreditCard
Sales.Currency
Sales.CurrencyRate
Sales.Customer
Sales.PersonCreditCard
Sales.SalesOrderDetail
Sales.SalesOrderHeader
Sales.SalesOrderHeaderSalesReason
Sales.SalesPerson
Sales.SalesPersonQuotaHistory
Sales.SalesReason
Sales.SalesTaxRate
Sales.SalesTerritory
Sales.SalesTerritoryHistory
Sales.ShoppingCartItem
Sales.SpecialOffer
Sales.SpecialOfferProduct
Sales.Store

Figure 5: List of Sales tables

It also has data of which customer buy what products, the quantity and revenue generated by each.

- **Production.ProductSubcategory**
  This table organizes products into subcategories, providing a higher-level grouping of related products.
  Reason for selection: It allows for grouping and analyzing products based on shared traits, which is essential for identifying trends within subcategories in the product mix.

- **Production.ProductCategory**
  This table further categorizes products at a broader level, grouping related subcat-

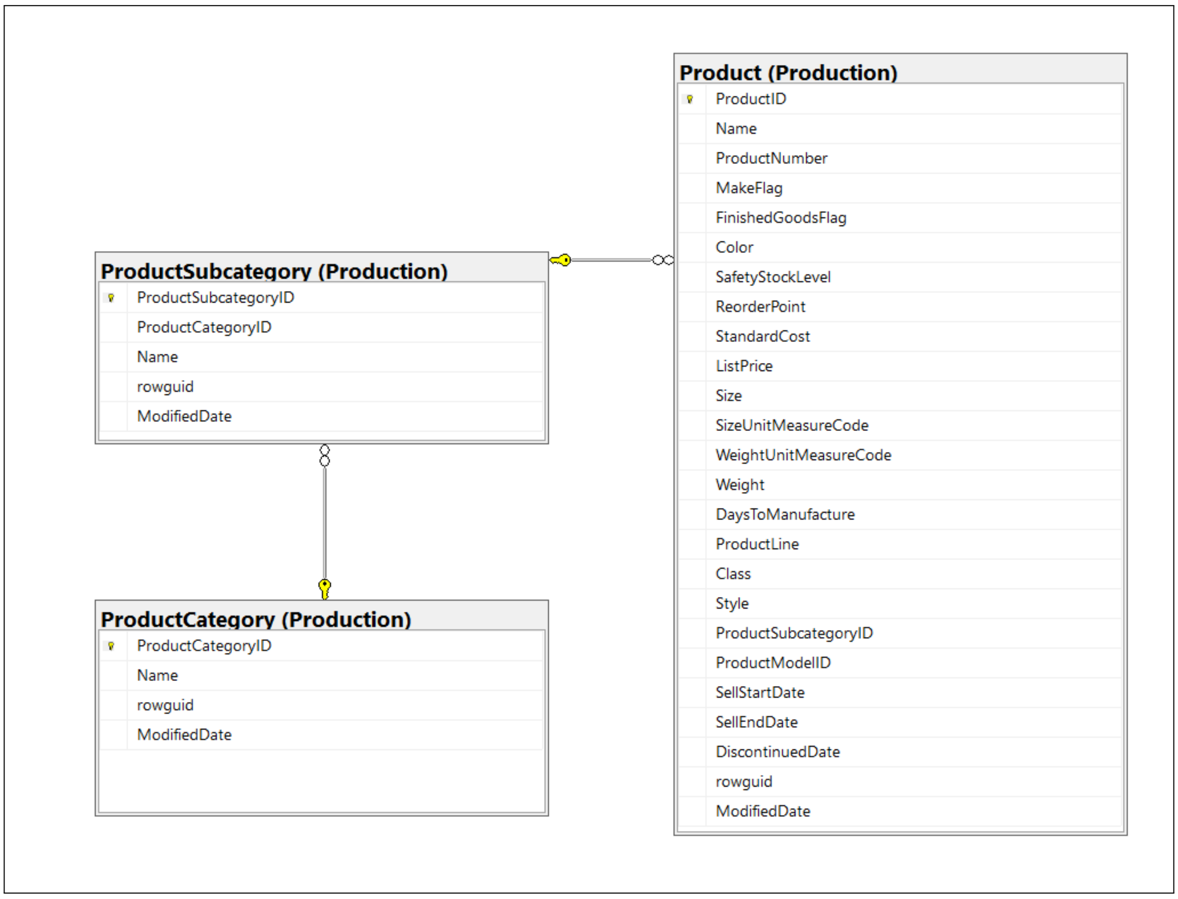


Figure 6: Production-related tables

egories.
Reason for selection: By analyzing categories, you can assess the overall performance of broad product lines and guide decisions on which categories to expand or reduce.

- **Sales.SalesOrderDetail**
  This table records transaction details, such as the quantity sold, unit price, discount applied, and total amount for each product in an order.
  Reason for selection: This table directly ties products to their sales performance, enabling the evaluation of how different products and categories contribute to revenue.

- **Sales.SalesOrderHeader**
  This table provides higher-level information about sales orders, including customer ID, order date, and shipping details.
  Reason for selection: It connects individual sales details to customer and order data, enabling insights into customer behavior and how different orders contribute to the overall sales mix.

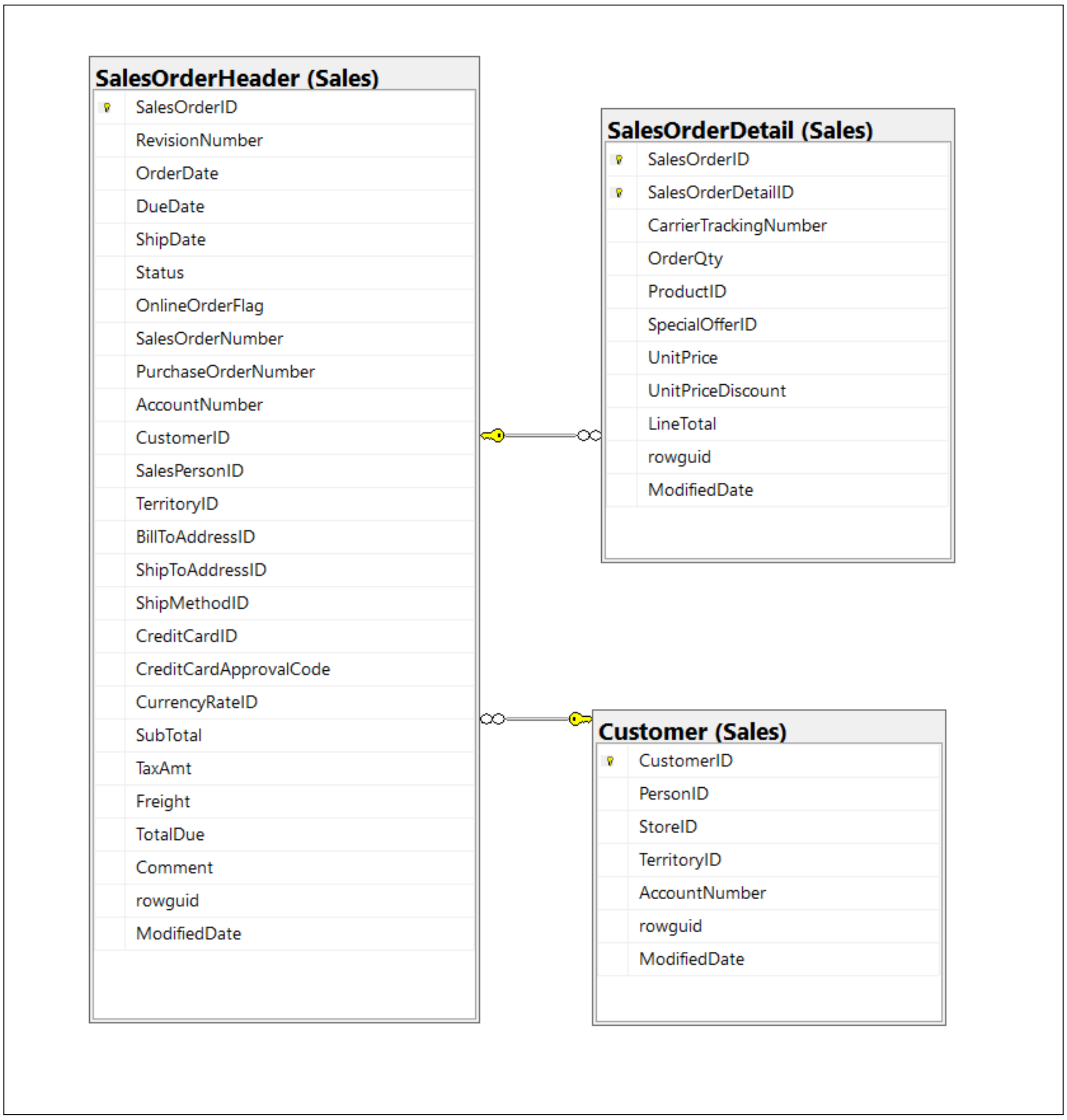


Figure 7: Sales-related tables

- **Sales.Customer**

  This table shows the CustomerID as long as the store where they made transaction, and which territory they belong to.

  Reason for selection: It connects individual sales details to customer and order data. The data is then used for calculating the RFM score for each of the customer.

# 4 Problem Statement

## 4.1 Business Context

From the CompanyX database, it is not infeasible to identify the company product mix. We observe that the products are categorized into 4 main categories: Bikes, Components, Clothing, and Accessories. This is shown in the *Production.ProductCategory* table, which comprises four rows.

There is also a table called *Production.ProductSubcategory*, in which each category is further divided into subcategories. There are a total of 37 subcategories, and each subcategory is assigned to one of the four main categories.

The mapping of categories and subcategories might be visualized in the figure below.

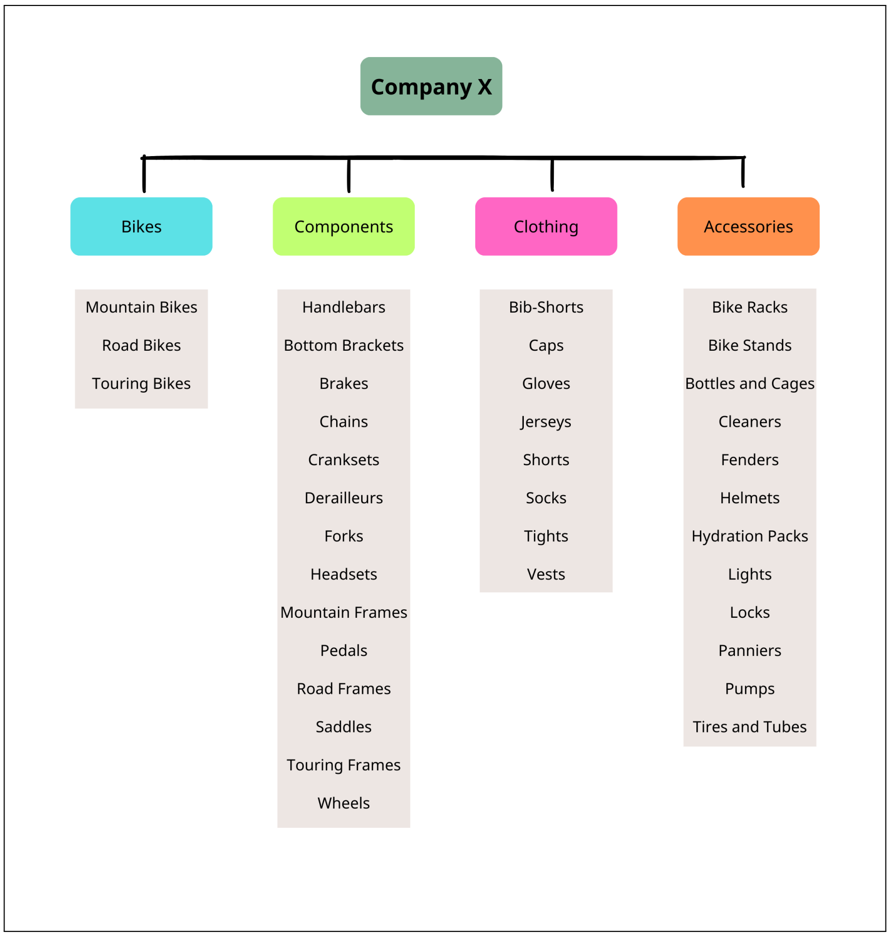


Figure 8: CompanyX product lines visualization

Within each subcategory, individual products are uniquely defined. Because the volume of companyX total products is large, it is not practical to illustrate the entire companyX product mix within a single graph. Product line may be determined by using query language to retrieve relevant data from the companyX dataset. The data of each product along with its labeled subcategory lies in the Production.Product table.

For instance, products belonging to the Handlebars subcategory are just a fraction of the company's total product mix. With 36 additional subcategories like Handlebars, and some of them containing significantly more products, the complexity of analyzing the product mix increases exponentially.

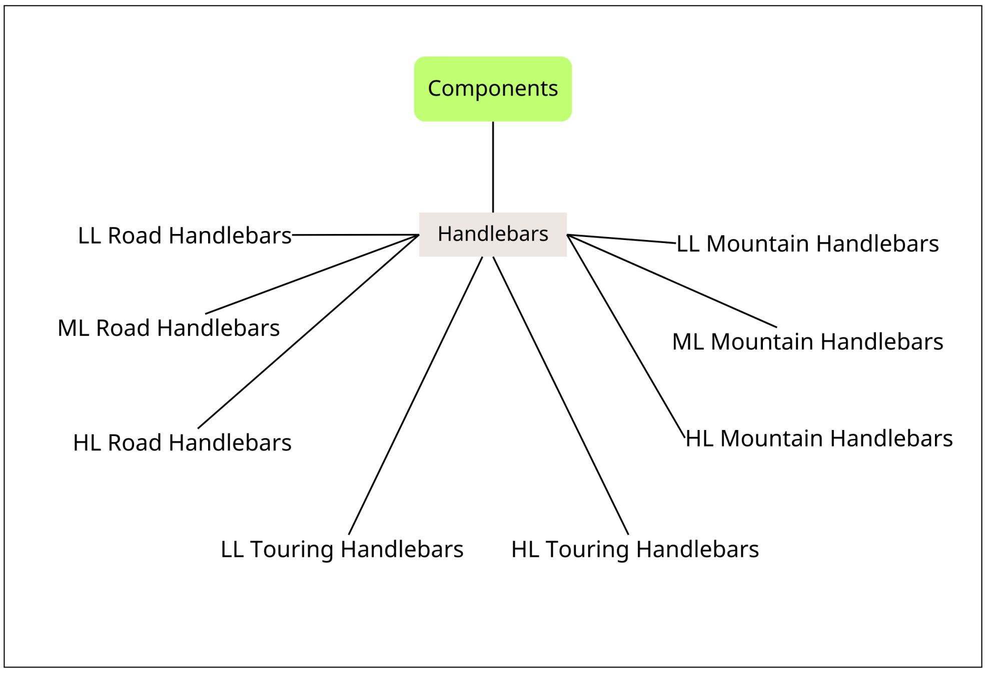


Figure 9: A CompanyX product line: Components

## 4.2 Challenges

From the inspection, we identified that in order to improve the business, we need to focus on two aspects. First is the sales performance, and second, the customer behavior, both regarding each of the product lines. There are certain metrics that we have defined earlier in section *2. Topic investigation*, and for the sales performances, the key performance indicators are: Gross Profit, Revenue, Sales Volume Growth Rate and Average Selling Price. As for the Customer Behavior, we implement the RFM Business Model, which implies the following metrics: Recency, Frequency and Moneytary. For each of the attributes, customers are assigned a score from 1 to 5 based on their observed purchasing behavior which is called RFM Score.

## 4.3 Goals and Expected Outcomes

To address the identified challenges, this project aims to achieve the following goals and outcomes:

1. **Sales Performance Optimization**

- Create insights of sales trends by analyzing metrics as mentioned before: gross profit, revenue, sales volume growth, and average selling price for each product line.
- Determine which product categories sells well, and which sells poorly, from that insights, other departments in company make decision, for example: inventory planning or sales promotion.

2. **Customer Behavior Segmentation**
   - Implement the RFM model to segment customers based on their purchasing behavior.
   - From the model, apply RFM score to classified customers into actionable groups relating to the customer loyalty.
3. **Products clustering based on product mix**
   - Cluster the products that belong to a single subcategory.
   - Compare all the subcategories of each category to identify good and bad groups of products according to the metrics.
   - From that, business can make decisions.
4. **Visualizing Insights**
   - Create an interactive Business Intelligence (BI) dashboard to visualize sales performance and customer behavior across product lines.
   - Present insights in a way that supports real-time decision-making.

# 5 Data Warehouse Design

## 5.1 Requirement Analysis

From the problem statement, we defined the two main approaches for the project:

- Product Line Sales Performance
- Customer Purchasing Behavior Segmentation

The main task to do in this section is to determine the essential attributes and where to obtain them. To do this, we need to look back at the metrics formulas, and compare it to the data set. There are tables to be discovered first and then the columns (attributes) from which the relevant data is collected.

- **Sales Revenue**: according to the formula, these figures are essential: Unit price, Quantity Sold for each product in the line.

Table 1: Sales Revenue required data from data source

| **Table** | **Attributes** |
|---|---|
| Sales.SalesOrderDetail | SalesOrderID, ProductID, LineTotal |
| Sales.SalesOrderHeader | SalesOrderID, Status, OrderDate |
| Production.Product | ProductID, ProductSubcategoryID |
| Production.Subcategory | ProductSubcategoryID, ProductCategoryID, Name |
| Production.Category | ProductCategoryID, Name |

- **Gross Profit**: to calculate this metrics, we need these figures: Sales Revenue, Quantity Sold and Unit Cost. In this case, Sold Quantity equals Unit Sold that is used to calculate Sales Revenue.

Table 2: Gross Profit required data from data source

| **Table** | **Attributes** |
|---|---|
| Sales.SalesOrderDetail | SalesOrderID, ProductID, LineTotal, OrderQty |
| Sales.SalesOrderHeader | SalesOrderID, Status, OrderDate |
| Production.Product | ProductID, ProductSubcategoryID, StandardCost, ListPrice |
| Production.Subcategory | ProductSubcategoryID, ProductCategoryID, Name |
| Production.Category | ProductCategoryID, Name |

- **Sales Volume Growth Rate**: it is necessary to retrieve Total Sold Quantity in a specific time period within a product category.

Table 3: Sales Volume Growth Rate required data from the data source)

| **Table** | **Attributes** |
|---|---|
| Sales.SalesOrderDetail | SalesOrderID, ProductID, OrderQty |
| Sales.SalesOrderHeader | SalesOrderID, Status, OrderDate |
| Production.Product | ProductID, ProductSubcategoryID |
| Production.Subcategory | ProductSubcategoryID, ProductCategoryID, Name |
| Production.Category | ProductCategoryID, Name |

- **Average Selling Price**: it is requisited to compute Sales Revenue, Order Quantity and Unit Cost. In this case, Order Quantity equals Unit Sold that is used to calculate Sales Revenue.

Table 4: Average Selling Price required data from the data source

| **Table** | **Attributes** |
|---|---|
| Sales.SalesOrderDetail | SalesOrderID, OrderQty, ProductID, LineTotal |
| Sales.SalesOrderHeader | SalesOrderID, Status, OrderDate |
| Production.Product | ProductID, ProductSubcategoryID, ListPrice. |
| Production.Subcategory | ProductSubcategoryID, ProductCategoryID, Name |
| Production.Category | ProductCategoryID, Name |

4

## 5.2 Dimensional Modeling

Facts are the numerical data that represent business measurements or KPIs (Key Performance Indicators) and are stored in the fact table. The fact table contains quantitative data that can be aggregated or analyzed.
In the project, the fact table, which is named "Product Mix Fact" includes the following metrics:

- **TotalQtySold**
  The total quantity sold of each product represents the total number of units sold to customers accumulating to different time interval.

- **SalesRevenue**
  Sales revenue is the income received by a company from its sales of goods or the provision of services. This is the total income generated from selling specific products or services.

- **GrossProfit**
  Gross profit or gross income, is a company's profit after deducting the costs associated with producing and selling its products or services.This helps identify which products are the most profitable and worth retaining, or else they are highly discontinued.

- **SalesGrowthRate**
  Sales volume growth rate measures your company's ability to generate revenue through sales over a fixed period of time. This indicates the popularity of the product line and reveals market demand trends. Rapid growth in sales volume can

signal high demand, while a declining rate may suggest the need for changes in marketing or product strategy.

- **AverageSellPrice**
  The average selling price (ASP) is a term that refers to the average price a good or service is sold for. This is used to highlight pricing trends and their impact on profitability, and also determine whether price adjustments are needed to stay competitive or increase margins.

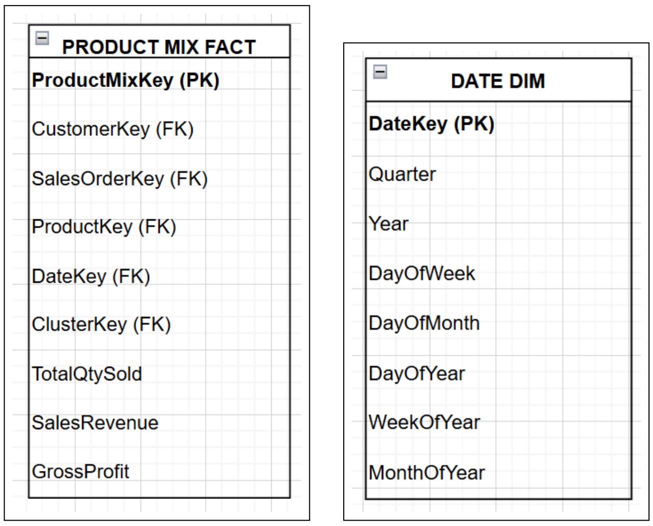


Figure 10: Fact and Dimension Table Designs

## 5.3 Schema Design

### 5.3.1 Fact Table

**Product Mix Fact**:
Central table that records sales and product-related metrics, connecting all dimensions for analysis. Key attributes include:

- **ProductMixKey (PK)**: Unique identifier for each fact record.
- Foreign keys linking to dimensions: *CustomerKey*, *SalesOrderKey*, *ProductKey*, *DateKey*, *ClusterKey*, and *SubcategoryKey*.
- Measures: *TotalQtySold*, *SalesRevenue*, *GrossProfit*, *SalesGrowthRate*, and *AverageSellPrice*.

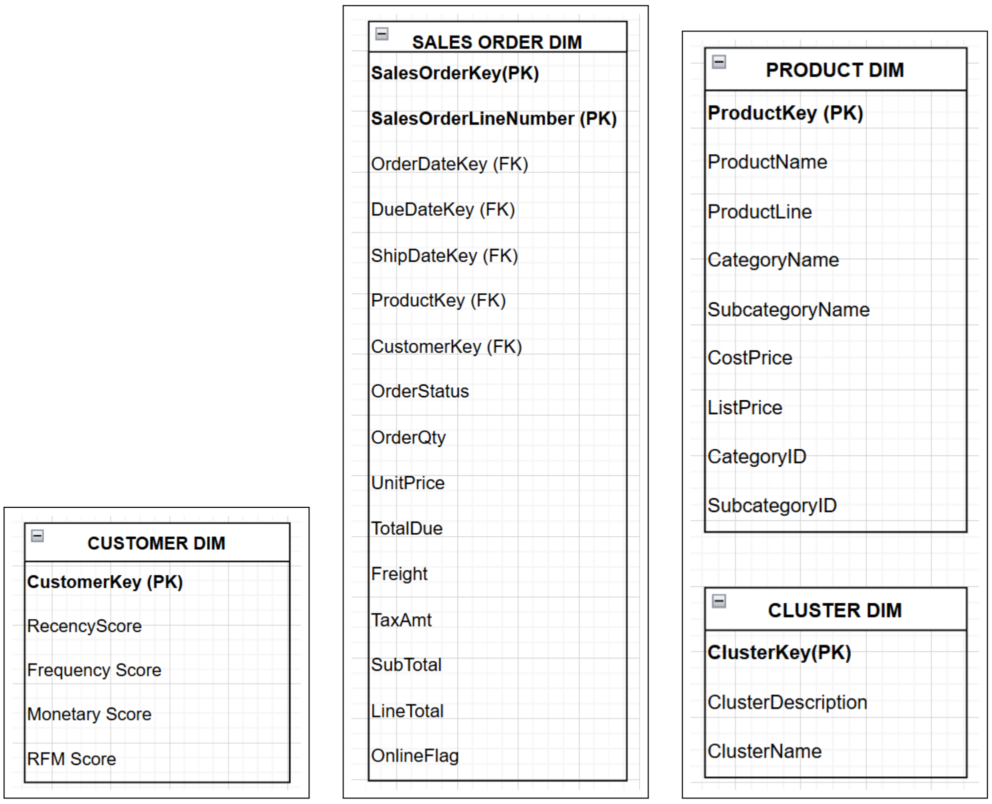


Figure 11: Fact and Dimension Table Designs (cont.)

### 5.3.2 Dimension Tables

- **Customer Dimension:**
  Stores customer data for segmentation and behavior analysis.
  Attributes include:
  - **CustomerKey (PK)**: Unique customer identifier.
  - Metrics like *Recency*, *Frequency*, *Monetary* scores, and *RFM Score*.

- **Cluster Dimension:**
  Groups products and line of products into clusters for targeted purposes or analysis.
  Attributes include:
  - **ClusterKey (PK)**: Unique cluster identifier.
  - *ClusterDescription* and *ClusterName*.

- **Product Dimension:**
  Contains product information for detailed product-level analysis.
  Attributes include:
  - **ProductKey (PK)**: Unique product identifier.

- *ProductName*, *ProductLine*, *CategoryName*, and *SubcategoryName*.
- Cost-related attributes: *CostPrice* and *ListPrice*.

- **Date Dimension:**
  Provides a calendar structure for time-based analysis.
  Attributes include:
  - **DateKey (PK)**: Unique date identifier.
  - *DayName*, *Quarter*, *Year*, and other calendar *breakdowns*.

- **Sales Order Dimension:**
  Tracks sales transactions and order details.
  Attributes include:
  - **SalesOrderKey (PK)** and **SalesOrderLineNumber (PK)**: Unique identifiers for orders and lines.
  - Composite primary keys that combine the order identifier with its line number of each purchase in the bill.
  - Foreign keys: *OrderDateKey*, *DueDateKey*, *ShipDateKey*, and *ProductKey*.
  - Measures: *OrderQty*, *UnitPrice*, *UnitPriceDiscountPct*, *TotalDue*, and others.

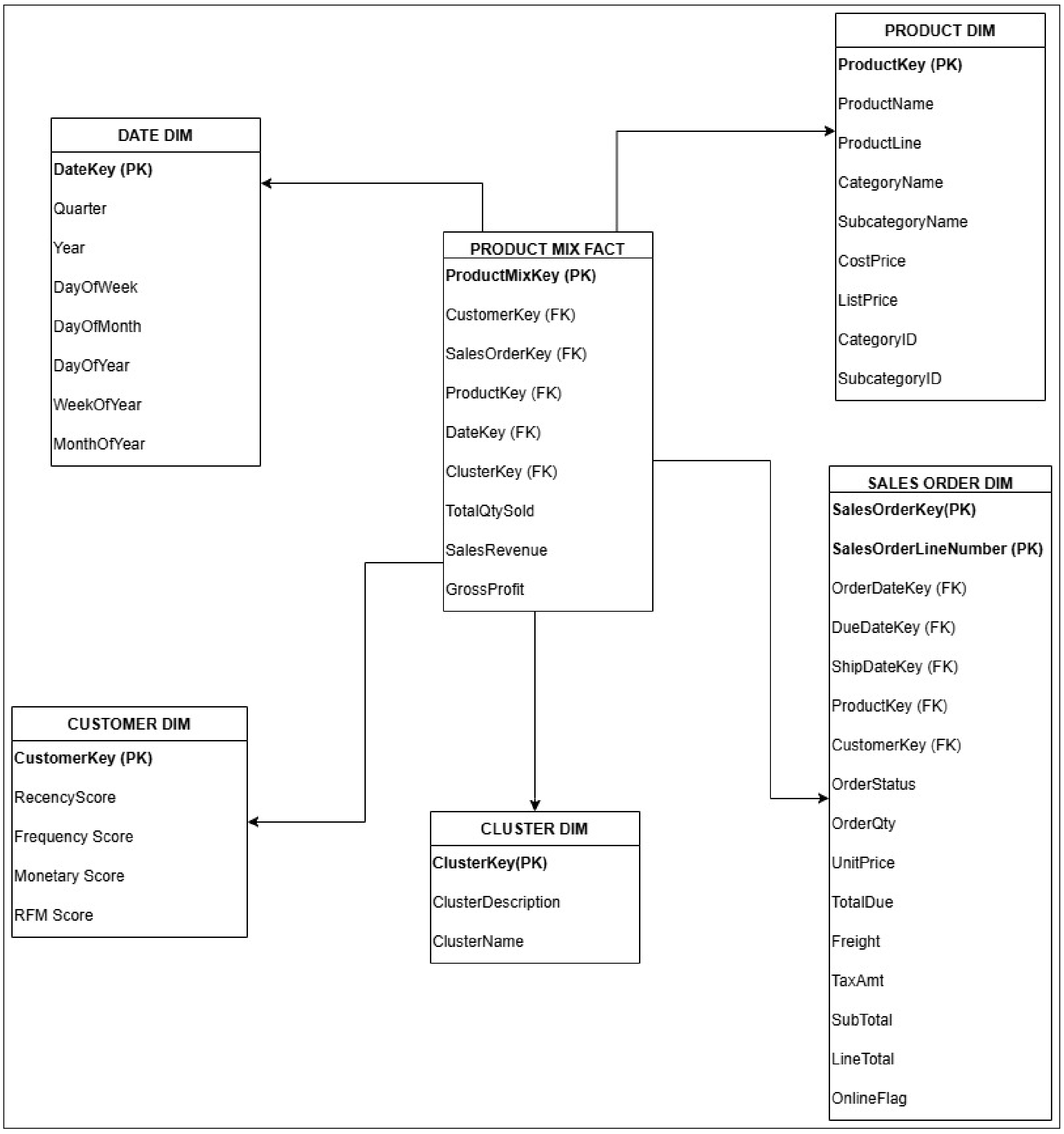


Figure 12: Star Schema of The Project

# 6 ETL Pipeline Implementation

## 6.1 ETL Overview

To implement the ETL pipeline, a combination of tools was used to facilitate data integration and transformation. The tools were chosen based on scalability, ease of use, and compatibility with the dataset and project requirements. The SQL Server Integration Service (SSIS) was trustfully used and we develop it using Microsoft Visual Studio 2022. The database management systems of SQL Server Management System 20 were employed for structured data storage, with the connection to the SQL Server 17 using Database Engine.
After creating the data warehouse, we use python as a ETL pipeline to connect data to the analysis service. Python were also used for data mining process, which acts as a tool to support data analyst.

## 6.2 Data Extraction

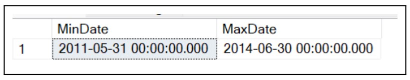
The data extraction phase involved retrieving raw data from the CompanyX dataset, using SQL Server Integration Service. For the connection, we use the OLE DB Source mainly, since it extracts data from an OLE DB-compliant relational database using SQL Command or other derived function. The source is easily found in the SSIS Toolbox in a Integration Service Project in Visual Studio 22.
As proposed, the star schema consists of one fact table, and five dimension tables. The four tables: ProductDim, SalesOrderDim, CustomerDim, and DateDim are created via extracting and aggregating data from the CompanyX dataset, whilst the ClusterDim is created after everytime the clustering part in python runs. The fact table used data from the four mentioned dimension tables, then calculated, computed and displayed with foreign keys and metrics.
The following part explain how the tables are queried and acquired.

**1. DateDim**

DateDim is retrieved using the time marks of the data of customer orders. From simple query in table SalesOrderHeader, we receive the time interval of the order data.

```sql
    SELECT  MIN (OrderDate) AS MinDate,
            MAX (OrderDate) AS MaxDate
    FROM [CompanyX].[Sales].[SalesOrderHeader]
```

The dimension table is defined and the source of data for the table is determined, with the column of OrderDate in Sales.SalesOrderHeader table in the relational database. The steps are presented in the coming figures.



| | MinDate | MaxDate |
|---|---|---|
| 1 | 2011-05-31 00:00:00.000 | 2014-06-30 00:00:00.000 |

Figure 13: Time interval of orders

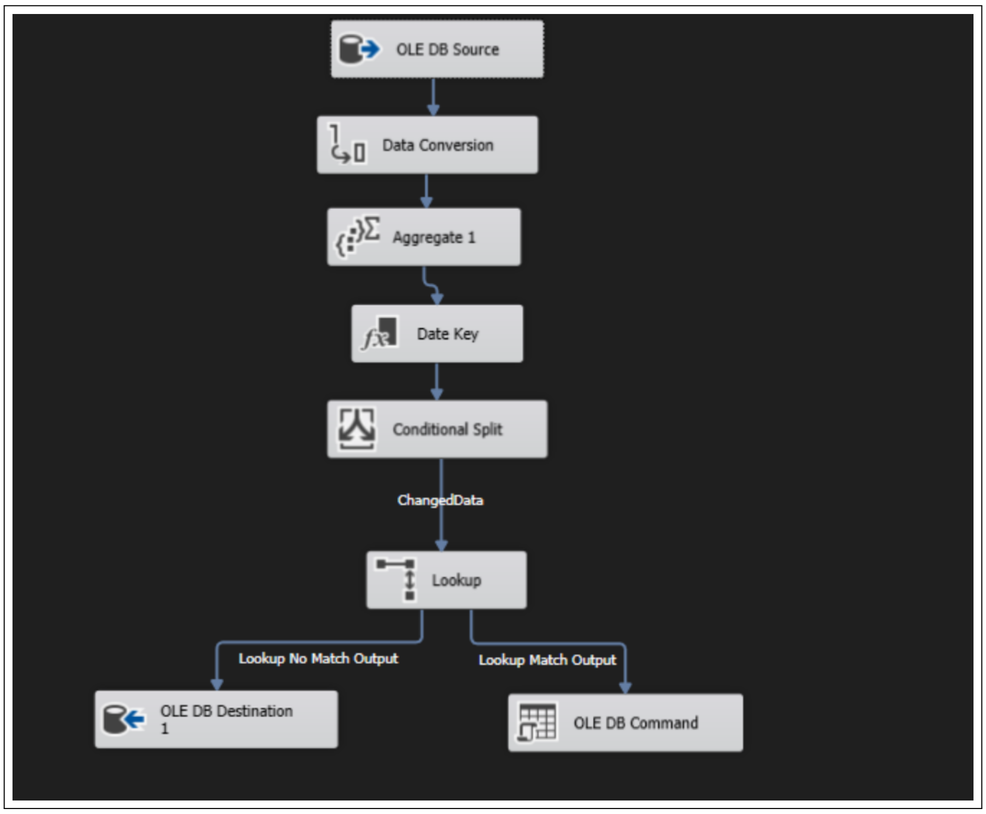


Figure 14: DimDate data flow

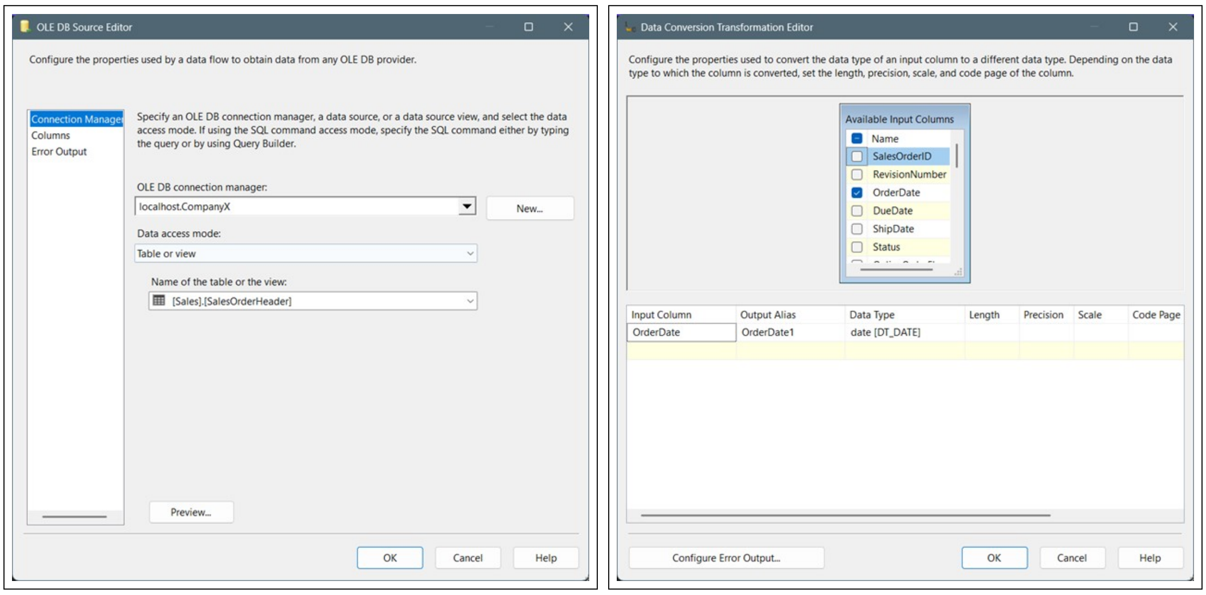
**2. ProductDim:**

The dimension is primarily derived from the Production.Product, Production.Subcategory and Production.Category tables.
Attribute Extraction: Relevant attributes are extracted from the Production.Product table and joined with related tables:

- Attributes: ProductKey (derived from ProductID), ProductName, CostPrice, List-Price.
- Product Subcategory: ProductSubcategory (Name from Production.ProductSubcategory joined based on ProductSubcategoryID).
- Product Category: ProductCategory (Name from Production.ProductCategory joined through Production.ProductSubcategory).

Here is the data flow of the creation of this dimension.

First, we need to join every subcategories with its category, then join the product with both its subcategory and category. to do this, we use merge join in SSIS. The result will


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: localhost.CompanyX

Data access mode: Table or view

Name of the table or the view: [Sales].[SalesOrderHeader]

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.

| Input Column | Output Alias | Data Type | Length | Precision | Scale | Code Page |
|---|---|---|---|---|---|---|
| OrderDate | OrderDate1 | date [DT_DATE] | | | | |

Figure 15: Step 1: Choose the source column

be pushed back to data warehouse as a newly created dimension table.

**3. SalesOrderDim:**

The dimension is primarily derived from the Sales.SalesOrderHeader table and also incorporates data from the Sales.SalesOrderDetail table.

SalesOrderKey (PK) and SalesOrderLineNumber (PK): Unique identifiers for orders and lines. Composite primary keys that combine the order identifier with its line number of each purchase in the bill.

**4. CustomerDim**:

The dimension is primarily derived from the Sales.SalesOrderHeader table. Key Generation: A unique CustomerKey is generated for each customer, using the CustomerID from the SalesOrderHeader table.

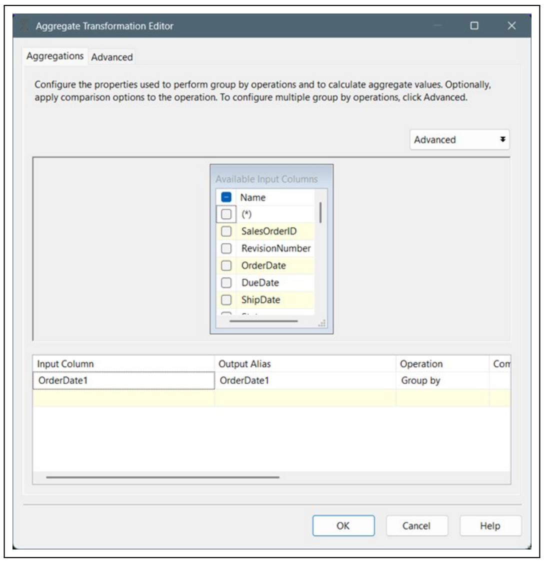
Customer Behavior Metrics Calculation:

- Recency: Calculated as the number of days since the customer's last order.
- Frequency: Calculated as the total number of orders placed by the customer.
- Monetary: Calculated as the total amount spent by the customer across all orders.
- RFM score: Calculated follow the formula:

$$\mathbf{0.15} \times \textbf{Recency} + \mathbf{0.28} \times \textbf{Frequency Score} + \mathbf{0.57} \times \textbf{Monetary Score}$$

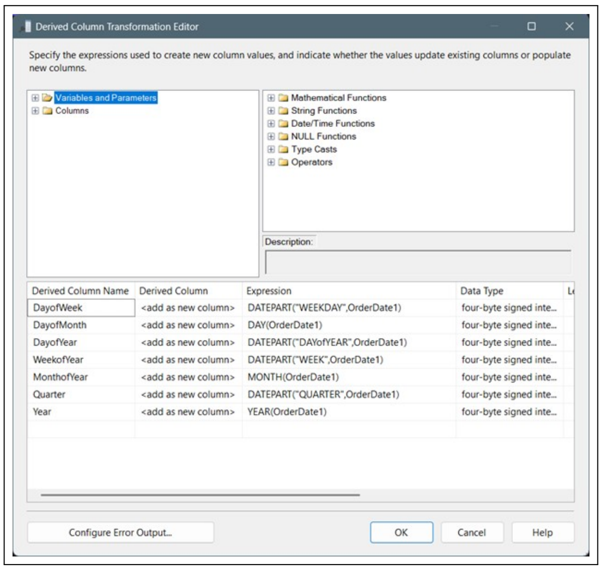
**5. Fact table:**

Central table that records sales and product-related metrics, connecting all dimensions for analysis. Key attributes include:



Configure the properties used to perform group by operations and to calculate aggregate values. Optionally, apply comparison options to the operation. To configure multiple group by operations, click Advanced.

| Input Column | Output Alias | Operation | Com |
|---|---|---|---|
| OrderDate1 | OrderDate1 | Group by | |



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 |
|---|---|---|---|---|
| DayofWeek | <add as new column> | DATEPART("WEEKDAY",OrderDate1) | four-byte signed inte... | |
| DayofMonth | <add as new column> | DAY(OrderDate1) | four-byte signed inte... | |
| DayofYear | <add as new column> | DATEPART("DAYofYEAR",OrderDate1) | four-byte signed inte... | |
| WeekofYear | <add as new column> | DATEPART("WEEK",OrderDate1) | four-byte signed inte... | |
| MonthofYear | <add as new column> | MONTH(OrderDate1) | four-byte signed inte... | |
| Quarter | <add as new column> | DATEPART("QUARTER",OrderDate1) | four-byte signed inte... | |
| Year | <add as new column> | YEAR(OrderDate1) | four-byte signed inte... | |

Figure 16: Step 2: Create the columns in dimension table

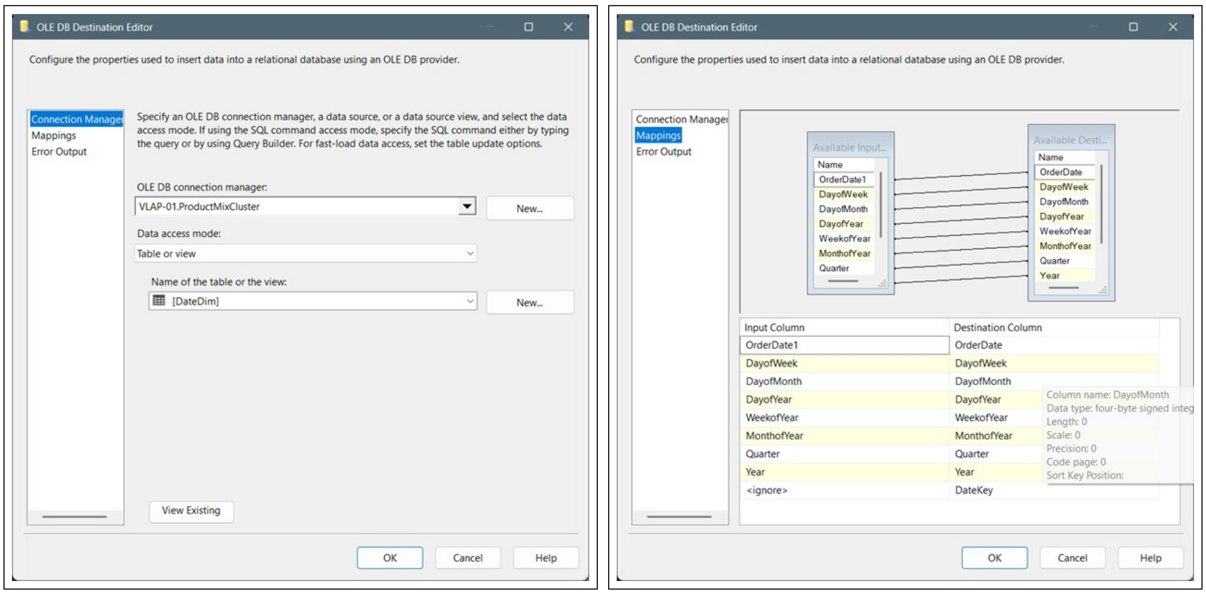
- **ProductMixKey (PK)**: Unique identifier for each fact record.
- Foreign keys linking to dimensions: *CustomerKey*, *SalesOrderKey*, *ProductKey*, *DateKey*, *ClusterKey*, and *SubcategoryKey*.
- Measures: *TotalQtySold*, *SalesRevenue*, *GrossProfit*, *SalesGrowthRate*, and *AverageSellPrice*.

## 6.3 Data Transformation

### 6.3.1 Data Preprocessing

Data preprocessing involved handling missing values, correcting inconsistencies, and standardizing data formats. Techniques like imputation for missing values, outlier detection, and type casting were applied to ensure the dataset was clean and usable for subsequent steps.
During inspection, some major issues in the raw dataset are discovered. Those issues and the solutions are:

1. Null values:
   - Production.product, column SubcategoryID of rows 1 to 209 is NULL.
     Solution: Assign -1, as unknown subcategory
   - Sales.SalesOrderDetail, rows of OrderID = 43678 has column UnitPrice of NULL.
     Solution: Fill the value manually using product list price and discount


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: VLAP-01.ProductMixCluster

Data access mode: Table or view

Name of the table or the view: [DateDim]

OLE DB Destination Editor

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

| Input Column | Destination Column |
|---|---|
| OrderDate1 | OrderDate |
| DayofWeek | DayofWeek |
| DayofMonth | DayofMonth |
| DayofYear | DayofYear |
| WeekofYear | WeekofYear |
| MonthofYear | MonthofYear |
| Quarter | Quarter |
| Year | Year |
| <ignore> | DateKey |

Column name: DayofMonth
Data type: four-byte signed integ
Length: 0
Scale: 0
Precision: 0
Code page: 0
Sort Key Position:

Figure 17: Step 3: Save the results to new dimension table

- Some other similar problems, the solution is to cross check other tables in the dataset and manually fill them.

2. Missvalues:

   - Sales.SalesOrderDetail, rows of OrderID = 43677 has column UnitPrice of 0.
     Solution: Fill the value manually using product list price and discount
   - Sales.SalesOrderDetail, rows of OrderID = 43682 has column OrderQty below 0.
     Solution: Calculate and guess the product, achieve the correct quantity and fill the value manually.

### 6.3.2 Normalization and Aggregation

Normalization techniques were employed to scale numerical data to a consistent range, improving the performance of machine learning models and ensuring uniformity across the dataset. Aggregation was applied to consolidate data points into meaningful summaries, such as monthly sales totals or customer purchase frequency.
This process is mostly applied in the data mining, hence detailed information to be discussed in later sections.

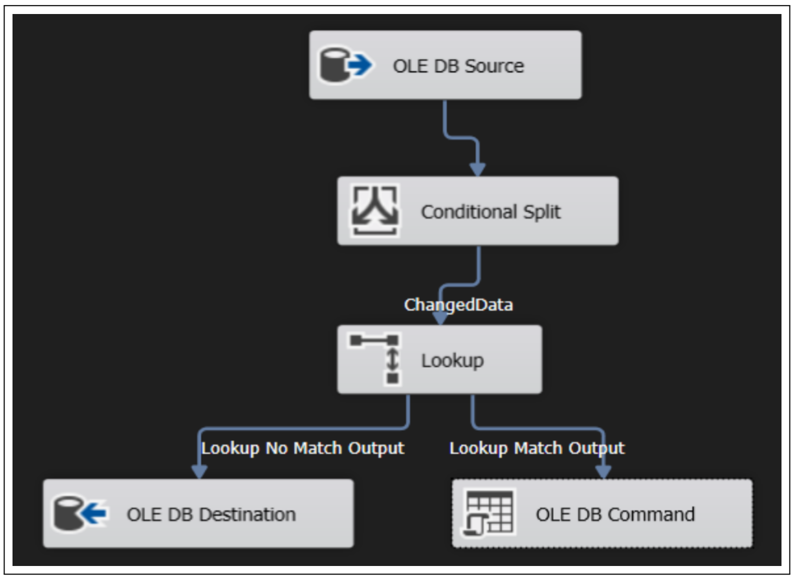


Figure 18: Data flow of dimension Product

## 6.4 Data Loading

### 6.4.1 Incremental Data Loading Process

### 6.4.2 Near Real-time Updates

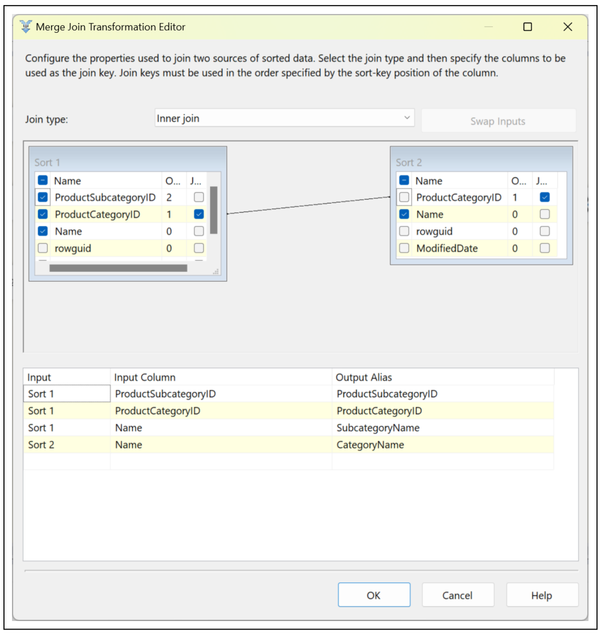
# 7 Data Mining Technique

## 7.1 Technique Selection

### 7.1.1 Choice explanation

There are many different types of clustering algorithms, including centroid-based, hierarchical, distribution-based, density-based, and grid-based. Each algorithm has its own strengths and weaknesses, and the best algorithm for a particular dataset will depend on the characteristics of the data. The selected technique is k-Means clustering, a centroid-model for clustering.

The option of k-Means clustering for the CompanyX dataset because it is an efficient and scalable algorithm suitable for analyzing large datasets like this one. The dataset includes numerical features such as customer purchase history and product preferences, which are ideal for K-Means' distance-based clustering. By grouping customers into distinct clusters, the algorithm helps identify patterns and customer segments, enabling targeted marketing strategies. Additionally, K-Means is straightforward to implement and interpret



Merge Join Transformation Editor

Configure the properties used to join two sources of sorted data. Select the join type and then specify the columns to be used as the join key. Join keys must be used in the order specified by the sort-key position of the column.

Join type: Inner join

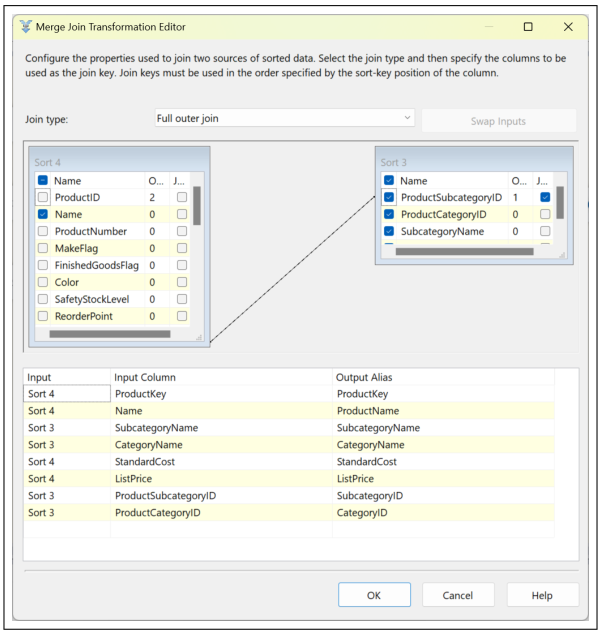
Sort 1

|  | Name | O... | J... |
|---|---|---|---|
| ✓ | ProductSubcategoryID | 2 | ☐ |
| ✓ | ProductCategoryID | 1 | ✓ |
| ✓ | Name | 0 | ☐ |
| ☐ | rowguid | 0 | ☐ |

Sort 2

|  | Name | O... | J... |
|---|---|---|---|
| ☐ | ProductCategoryID | 1 | ✓ |
| ✓ | Name | 0 | ☐ |
| ☐ | rowguid | 0 | ☐ |
| ☐ | ModifiedDate | 0 | ☐ |

| Input | Input Column | Output Alias |
|---|---|---|
| Sort 1 | ProductSubcategoryID | ProductSubcategoryID |
| Sort 1 | ProductCategoryID | ProductCategoryID |
| Sort 1 | Name | SubcategoryName |
| Sort 2 | Name | CategoryName |



Merge Join Transformation Editor

Configure the properties used to join two sources of sorted data. Select the join type and then specify the columns to be used as the join key. Join keys must be used in the order specified by the sort-key position of the column.

Join type: Full outer join

Sort 4

|  | Name | O... | J... |
|---|---|---|---|
| ☐ | ProductID | 2 | ☐ |
| ✓ | Name | 0 | ☐ |
| ☐ | ProductNumber | 0 | ☐ |
| ☐ | MakeFlag | 0 | ☐ |
| ☐ | FinishedGoodsFlag | 0 | ☐ |
| ☐ | Color | 0 | ☐ |
| ☐ | SafetyStockLevel | 0 | ☐ |
| ☐ | ReorderPoint | 0 | ☐ |

Sort 3

|  | Name | O... | J... |
|---|---|---|---|
| ✓ | ProductSubcategoryID | 1 | ✓ |
| ✓ | ProductCategoryID | 0 | ☐ |
| ✓ | SubcategoryName | 0 | ☐ |

| Input | Input Column | Output Alias |
|---|---|---|
| Sort 4 | ProductKey | ProductKey |
| Sort 4 | Name | ProductName |
| Sort 3 | SubcategoryName | SubcategoryName |
| Sort 3 | CategoryName | CategoryName |
| Sort 4 | StandardCost | StandardCost |
| Sort 4 | ListPrice | ListPrice |
| Sort 3 | ProductSubcategoryID | SubcategoryID |
| Sort 3 | ProductCategoryID | CategoryID |

Figure 19: Join each product with it subcategory and category

### 7.1.2 Clustering explanation

Given a set of observations $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, where each observation is a $d$-dimensional real vector, $k$-means clustering aims to partition the $n$ observations into $k \leq n$ sets $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS), i.e., variance. Formally, the objective is to find:

$$\arg\min_{\mathcal{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathcal{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

where $\boldsymbol{\mu}_i$ is the mean (also called centroid) of points in $S_i$, i.e.,

$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{\mathbf{x} \in S_i} \mathbf{x},$$

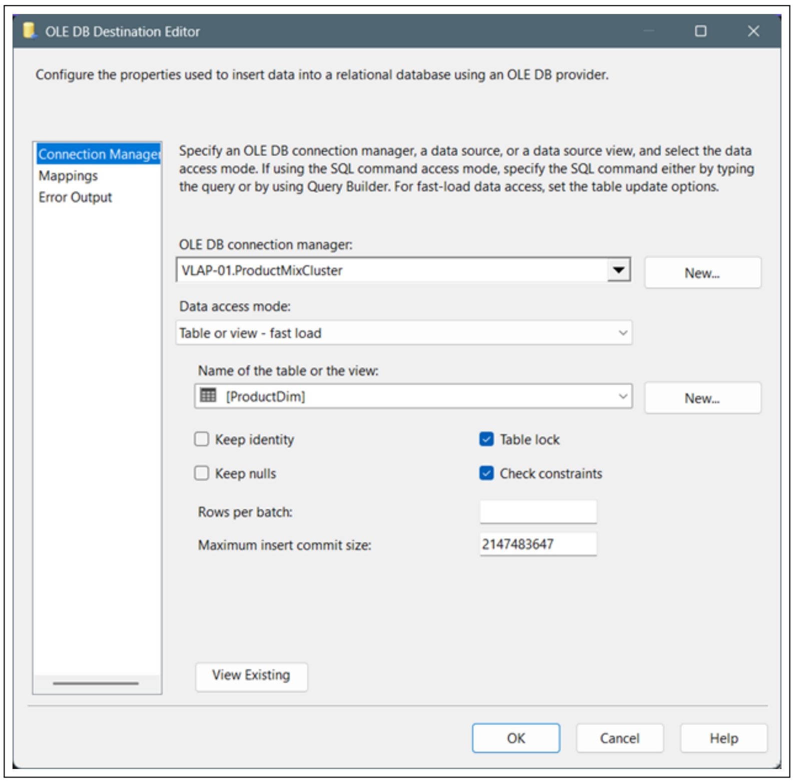
$|S_i|$ is the size of $S_i$, and $\|\cdot\|$ is the usual $L^2$ norm. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

$$\arg\min_{\mathcal{S}} \sum_{i=1}^{k} \frac{1}{|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2$$

The equivalence can be deduced from the identity:

$$|S_i| \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \frac{1}{2} \sum_{\mathbf{x},\mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2.$$

Since the total variance is constant, this is equivalent to maximizing the sum of squared deviations between points in *different* clusters (between-cluster sum of squares, BCSS).


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: VLAP-01.ProductMixCluster

Data access mode: Table or view - fast load

Name of the table or the view: [ProductDim]

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

Rows per batch:

Maximum insert commit size: 2147483647

Figure 20: Destination of dimension Product

This deterministic relationship is also related to the law of total variance in probability theory.

**Algorithm:**

The most common algorithm for $k$-means uses an iterative refinement technique. Due to its ubiquity, it is often called "the $k$-means algorithm," also referred to as Lloyd's algorithm in the computer science community. The algorithm proceeds by alternating between two steps:

1. **Assignment step:** Assign each observation to the cluster with the nearest mean, using the least squared Euclidean distance. Mathematically, this means partitioning the observations according to the Voronoi diagram generated by the means:

$$S_i^{(t)} = \left\{ \mathbf{x}_p : \|\mathbf{x}_p - \mathbf{m}_i^{(t)}\|^2 \leq \|\mathbf{x}_p - \mathbf{m}_j^{(t)}\|^2,\ \forall j,\ 1 \leq j \leq k \right\},$$

   where each $\mathbf{x}_p$ is assigned to exactly one $S^{(t)}$, even if it could be assigned to two or more of them.

2. **Update step:** Recalculate the means (centroids) for observations assigned to each cluster:

$$\mathbf{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j.$$

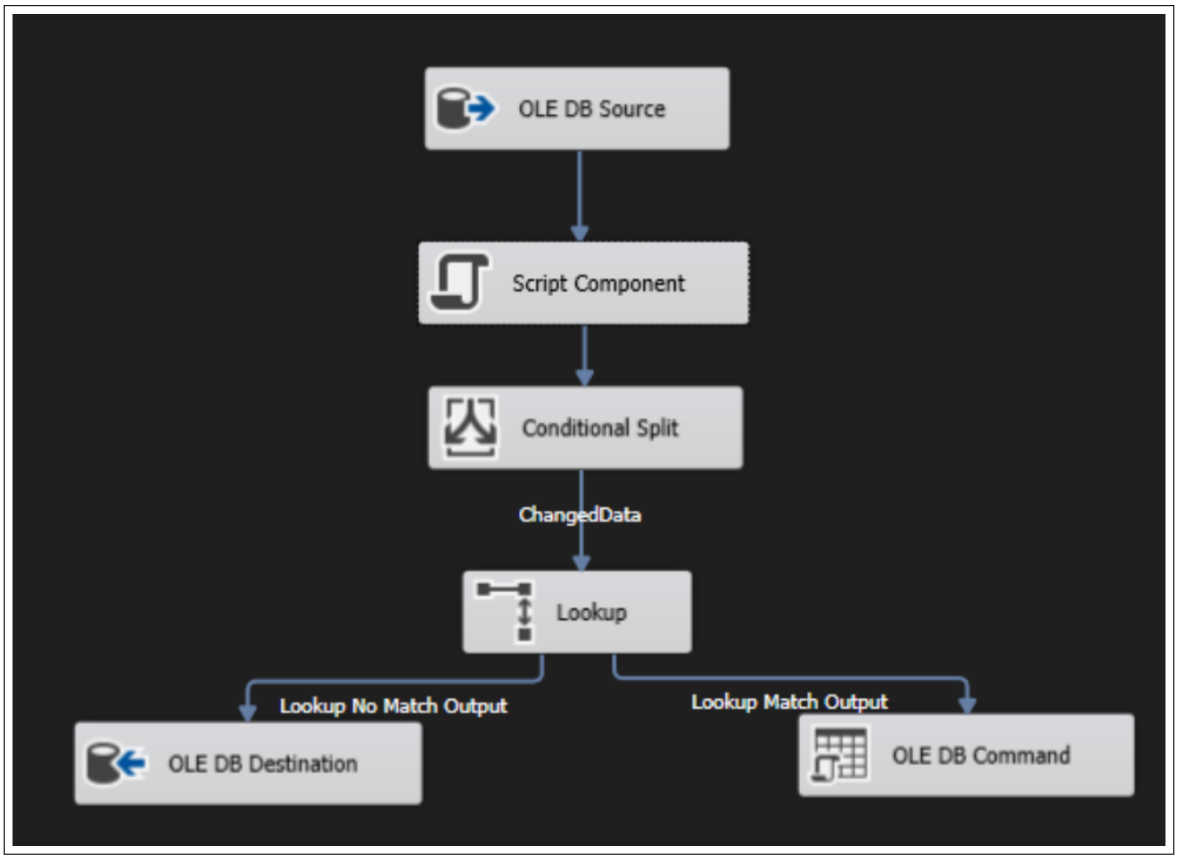


Figure 21: Data flow of Sales Order dimension

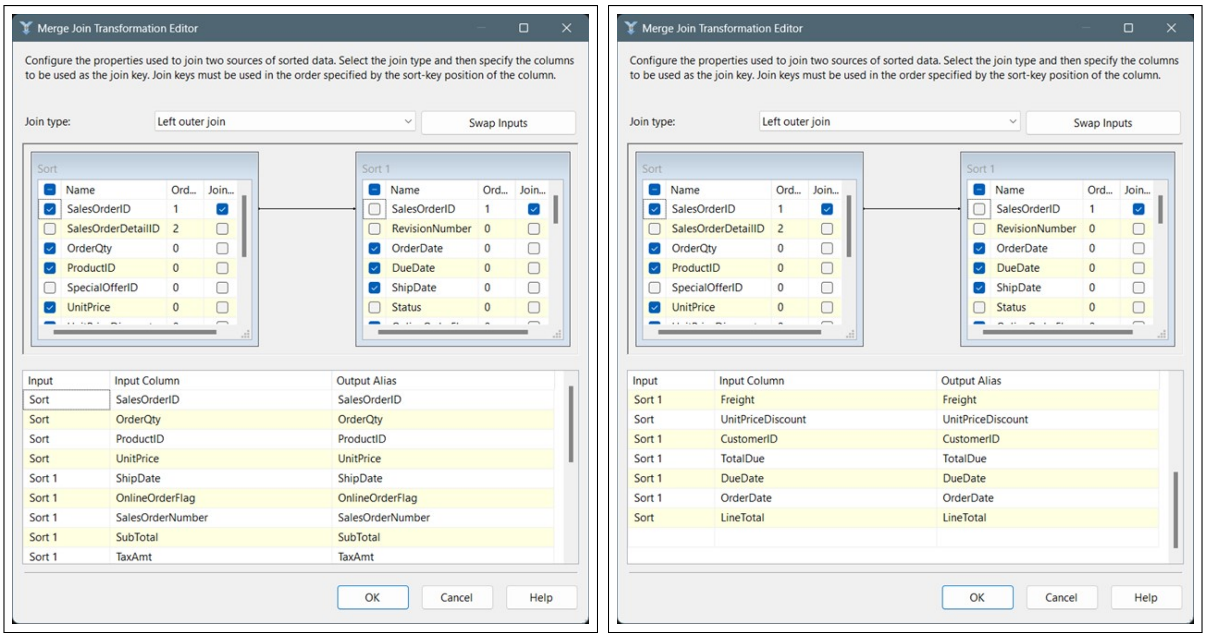
The objective function in $k$-means is the WCSS (within-cluster sum of squares). After each iteration, the WCSS decreases, ensuring that the algorithm always converges, though not necessarily to the global optimum. The algorithm has converged when the assignments no longer change, or equivalently, when the WCSS becomes stable.

## 7.2 Implementation

**Step 1: Import required libraries**

To load the necessary Python libraries for the operations performed in the script. Key libraries include:

- pyodbc for SQL database connection.
- pandas for data manipulation and querying.
- sklearn for clustering and preprocessing.
- matplotlib and seaborn for data visualization.
- sqlalchemy for connecting to SQL databases.



Merge Join Transformation Editor

Configure the properties used to join two sources of sorted data. Select the join type and then specify the columns to be used as the join key. Join keys must be used in the order specified by the sort-key position of the column.

Join type: Left outer join

Sort

| ✓ | Name | Ord... | Join... |
|---|---|---|---|
| ✓ | SalesOrderID | 1 | ✓ |
| ☐ | SalesOrderDetailID | 2 | ☐ |
| ✓ | OrderQty | 0 | ☐ |
| ✓ | ProductID | 0 | ☐ |
| ☐ | SpecialOfferID | 0 | ☐ |
| ✓ | UnitPrice | 0 | ☐ |

Sort 1

| ✓ | Name | Ord... | Join... |
|---|---|---|---|
| ☐ | SalesOrderID | 1 | ✓ |
| ☐ | RevisionNumber | 0 | ☐ |
| ✓ | OrderDate | 0 | ☐ |
| ✓ | DueDate | 0 | ☐ |
| ✓ | ShipDate | 0 | ☐ |
| ☐ | Status | 0 | ☐ |

| Input | Input Column | Output Alias |
|---|---|---|
| Sort | SalesOrderID | SalesOrderID |
| Sort | OrderQty | OrderQty |
| Sort | ProductID | ProductID |
| Sort | UnitPrice | UnitPrice |
| Sort 1 | ShipDate | ShipDate |
| Sort 1 | OnlineOrderFlag | OnlineOrderFlag |
| Sort 1 | SalesOrderNumber | SalesOrderNumber |
| Sort 1 | SubTotal | SubTotal |
| Sort 1 | TaxAmt | TaxAmt |

Merge Join Transformation Editor

Configure the properties used to join two sources of sorted data. Select the join type and then specify the columns to be used as the join key. Join keys must be used in the order specified by the sort-key position of the column.

Join type: Left outer join

| Input | Input Column | Output Alias |
|---|---|---|
| Sort 1 | Freight | Freight |
| Sort | UnitPriceDiscount | UnitPriceDiscount |
| Sort 1 | CustomerID | CustomerID |
| Sort 1 | TotalDue | TotalDue |
| Sort 1 | DueDate | DueDate |
| Sort 1 | OrderDate | OrderDate |
| Sort | LineTotal | LineTotal |

Figure 22: Merge join of header and detail in sales orders

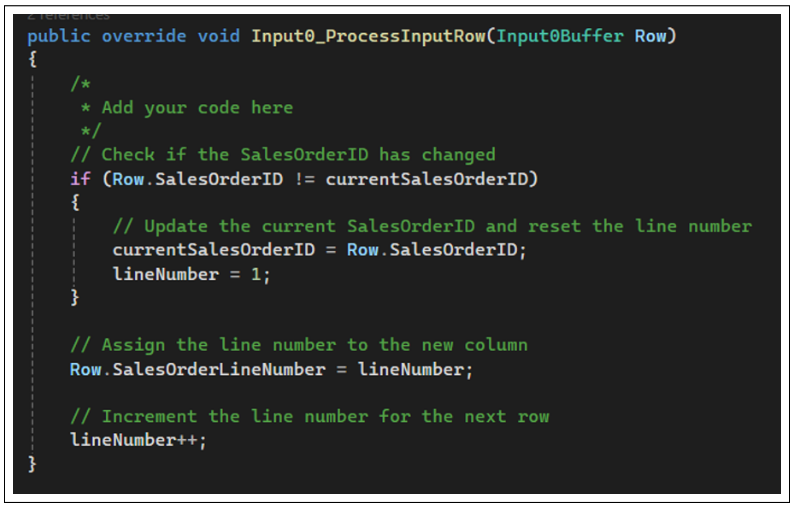
```python
# Import required libraries
import pyodbc
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
import seaborn as sns
from sqlalchemy import inspect
from sqlalchemy import create_engine
```

**Step 2: Establish database connection**

To set up a connection to the SQL Server database CompanyX.

- Configuration details: DRIVER_NAME specifies the SQL Server driver.
  SERVER_NAME is the name of the server.
  DATABASE_NAME is the database name.
  conn_str creates a connection string using the create_engine() function.
- Authentication: A trusted connection is used (Windows Authentication).
- Validation: A test connection is made to confirm the connection is successful.

```python
# Database connection details
DRIVER_NAME = 'SQL Server'
SERVER_NAME = 'DESKTOP-QU8EF69'  #alternative
DATABASE_NAME = 'CompanyX'       #alternative
```



```
public override void Input0_ProcessInputRow(Input0Buffer Row)
{
    /*
     * Add your code here
     */
    // Check if the SalesOrderID has changed
    if (Row.SalesOrderID != currentSalesOrderID)
    {
        // Update the current SalesOrderID and reset the line number
        currentSalesOrderID = Row.SalesOrderID;
        lineNumber = 1;
    }

    // Assign the line number to the new column
    Row.SalesOrderLineNumber = lineNumber;

    // Increment the line number for the next row
    lineNumber++;
}
```

Figure 23: C# script to assign Line Number to each order

```

# Trusted connection (Windows Authentication)
conn_str = f"mssql+pyodbc://{SERVER_NAME}/{DATABASE_NAME}?driver={
    DRIVER_NAME}&trusted_connection=yes"

# Create the engine
engine = create_engine(conn_str)

# Test connection
try:
    with engine.connect() as connection:
        print("Connection successful!")
except Exception as e:
    print("Connection failed:", e)
```

**Step 3: Query data from data warehouse**

To fetch data from the database for analysis and clustering using a SQL query. Query Details:

- The data is aggregated in the ProductMetrics.
- Key columns are: TotalQtySold, SalesRevenue, GrossProfit, and SalesGrowthRate.
- Products are matched within the same subcategory for clustering purposes.
- The query output is loaded into a DataFrame (df) using pd.read_sql().

```
query = """
WITH ProductMetrics AS (
    SELECT
        P.ProductKey,
```

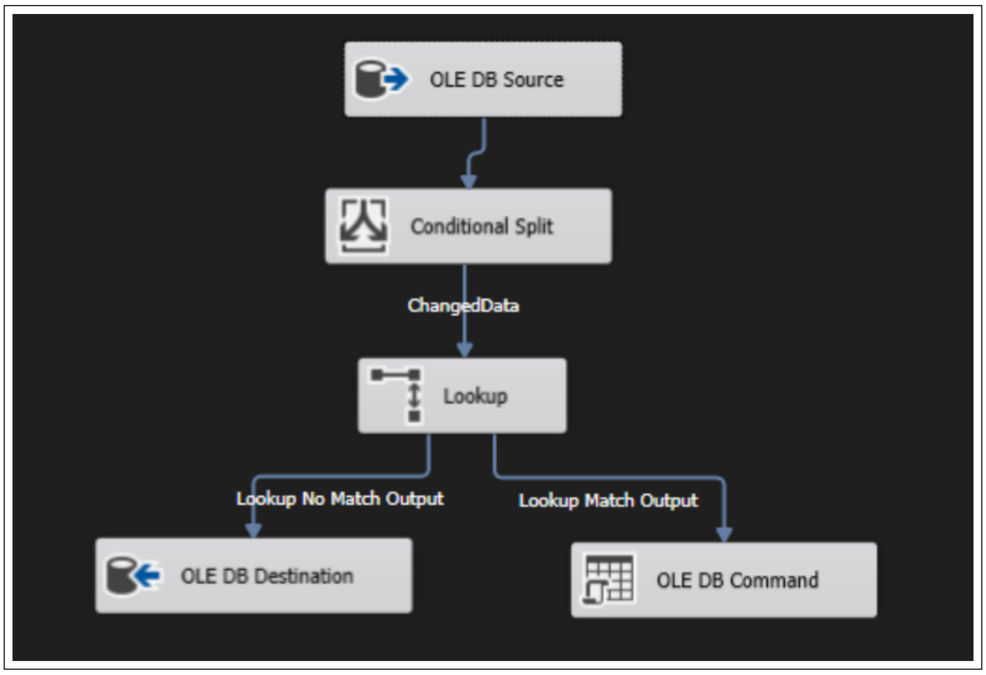
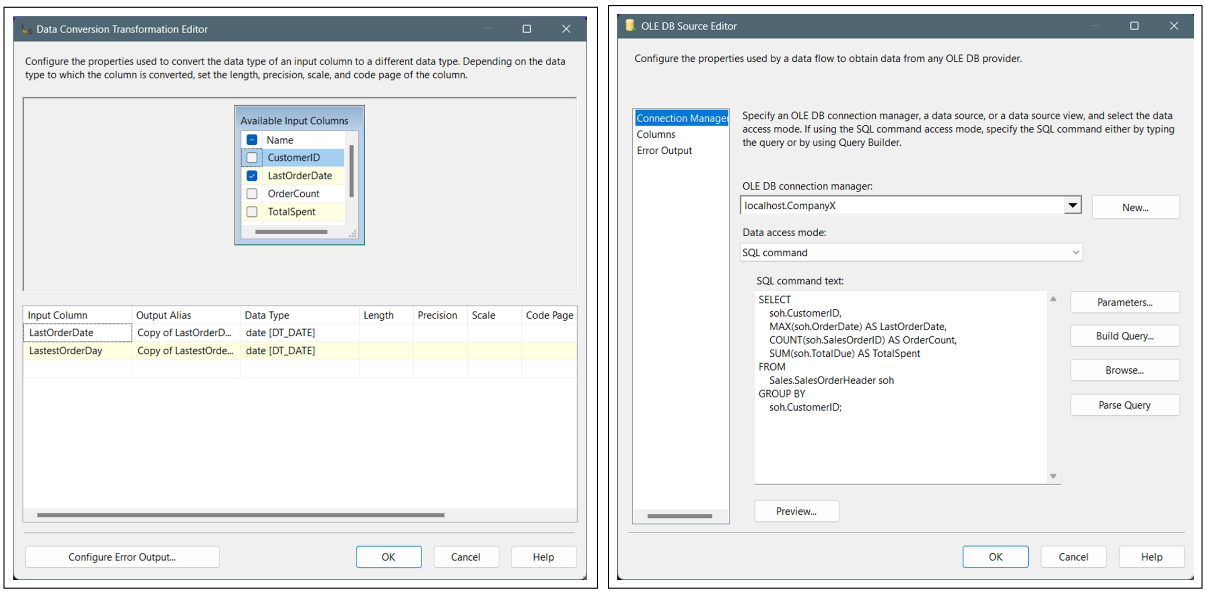


Figure 24: Data flow of Customer dimension

```
            P.ProductName,
            P.SubcategoryID,
            P.SubcategoryName,
            SUM(F.TotalQtySold) AS TotalQtySold,
            SUM(F.SalesRevenue) AS SalesRevenue,
            SUM(F.GrossProfit) AS GrossProfit,
            AVG(F.SalesGrowthRate) AS SalesGrowthRate
        FROM
            PRODUCT_DIM P
        INNER JOIN
            PRODUCT_MIX_FACT F
            ON P.ProductKey = F.ProductKey
        GROUP BY
            P.ProductKey, P.ProductName, P.SubcategoryID
    )
    SELECT
        PM1.ProductKey AS ProductKey,
        PM1.ProductName AS ProductName,
        PM1.SubcategoryName AS SubcategoryName,
        PM1.SubcategoryID AS SubcategoryID,
        PM1.TotalQtySold,
        PM1.SalesRevenue,
        PM1.GrossProfit,
        PM1.SalesGrowthRate,
        PM2.ProductKey AS SameSubcategoryProductKey,
        PM2.ProductName AS SameSubcategoryProductName
```



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
- [ ] CustomerID
- [✓] LastOrderDate
- [ ] OrderCount
- [ ] TotalSpent

| Input Column | Output Alias | Data Type | Length | Precision | Scale | Code Page |
|---|---|---|---|---|---|---|
| LastOrderDate | Copy of LastOrderD... | date [DT_DATE] | | | | |
| LastestOrderDay | Copy of LastestOrde... | date [DT_DATE] | | | | |

Configure Error Output... | OK | Cancel | Help

OLE DB Source Editor

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

Connection Manager
Columns
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.

OLE DB connection manager: localhost.CompanyX

Data access mode: SQL command

SQL command text:

```
SELECT
    soh.CustomerID,
    MAX(soh.OrderDate) AS LastOrderDate,
    COUNT(soh.SalesOrderID) AS OrderCount,
    SUM(soh.TotalDue) AS TotalSpent
FROM
    Sales.SalesOrderHeader soh
GROUP BY
    soh.CustomerID;
```

New... | Parameters... | Build Query... | Browse... | Parse Query | Preview... | OK | Cancel | Help

Figure 25: Data extraction and conversion for Customer dimension

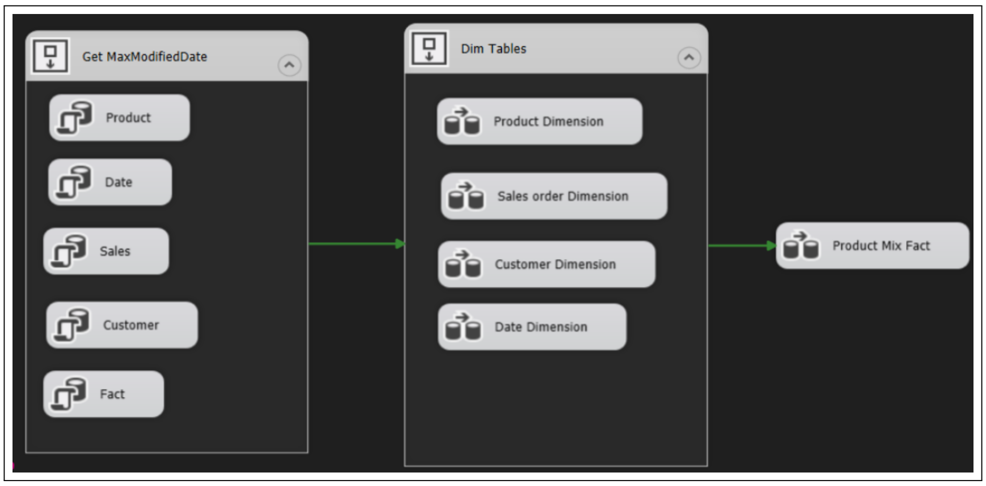


Figure 26: Fact table control flow

```
    FROM
        ProductMetrics PM1
    INNER JOIN
        ProductMetrics PM2
        ON PM1.SubcategoryID = PM2.SubcategoryID
        AND PM1.ProductKey != PM2.ProductKey
    ORDER BY
        PM1.SubcategoryID, PM1.ProductKey;
    """
    df = pd.read_sql(query, conn)
```

**Step 4: Preprocessing**

To clean and prepare the data for clustering by handling missing values, removing outliers, and standardizing numerical features.

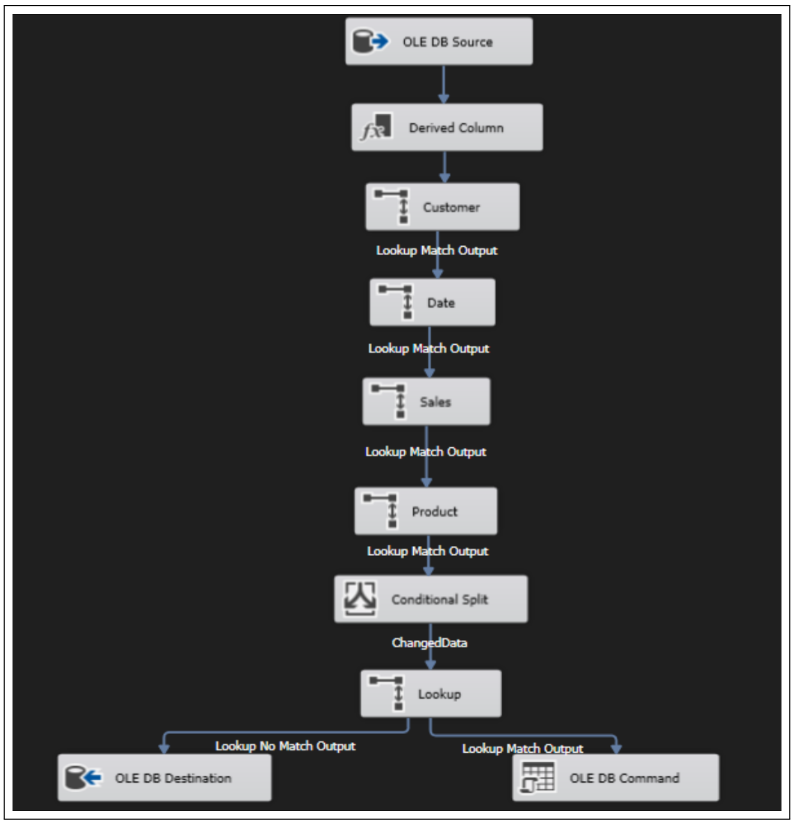


Figure 27: Fact table data flow

- Check missing values: df.isnull().sum() identifies missing data.
  Handle missing values: missing values are manually handled or filled as mentioned before in this report.
- Outlier Removal: Outliers are removed using the Interquartile Range (IQR) method.
- Feature Scaling: Numerical columns (TotalQtySold, SalesRevenue, GrossProfit) are standardized using StandardScaler.
- Output: Preprocessed data (df_scaled) ready for clustering.

```python
# Check for missing values
print(df.isnull().sum())
#data[metrics] = data[metrics].fillna(0)

# Fill or drop missing values
#NOTE: Missing values are manually handled

# data_scaled = scaler.fit_transform(data[metrics])
# df.fillna(method='ffill', inplace=True)  # Example: forward fill

# Remove outliers (outside 1.5 * IQR)
Q1 = df[numeric_features].quantile(0.25)
```

```python
    Q3 = df[numeric_features].quantile(0.75)
    IQR = Q3 - Q1
    df = df[~((df[numeric_features] < (Q1 - 1.5 * IQR)) | (df[
        numeric_features] > (Q3 + 1.5 * IQR))).any(axis=1)]

    # Normalize/standardize numerical data
    scaler = StandardScaler()
    numeric_features = ['TotalQtySold', 'SalesRevenue', 'GrossProfit']
    df_scaled = scaler.fit_transform(df[numeric_features])
```

**Step 5: Determine the optimal number of clusters**

```python
    # Define the range of potential cluster numbers
    k_values = range(1, 11)  # Try clusters from 1 to 10
    inertia = []  # Store the inertia (sum of squared distances) for
        each value of k

    # Loop over each value of k and calculate inertia
    for k in k_values:
        kmeans = KMeans(n_clusters=k, random_state=42)  # Initialize
            KMeans
        kmeans.fit(df_scaled)  # Fit to scaled data
        inertia.append(kmeans.inertia_)  # Append the inertia to the
            list

    # Plot the elbow graph
    plt.figure(figsize=(8, 5))
    plt.plot(k_values, inertia, marker='o', linestyle='--')
    plt.title('Elbow Method for Optimal Number of Clusters')
    plt.xlabel('Number of Clusters (k)')
    plt.ylabel('Inertia (Sum of Squared Distances)')
    plt.xticks(np.arange(1, 11, 1))
    plt.grid(True)
    plt.show()
```

- k_values: The range of cluster numbers is defined (e.g., 1 to 10).
- inertia: Inertia is the sum of squared distances of samples to their nearest cluster center. It measures how well data points are grouped within clusters.
- Loop Over k: For each k, fit the K-Means model and compute the inertia.
- Plot Results: The plot visualizes k against inertia. The "elbow" point indicates the optimal number of clusters where inertia starts to decrease at a slower rate. Identify the "elbow" point in the graph (where the rate of inertia reduction sharply decreases). This is the ideal number of clusters to use.

**Step 6: Apply k-Means clustering**

```python
kmeans = KMeans(n_clusters=3, random_state=42)  # Define number of clusters
df['Cluster'] = kmeans.fit_predict(df_scaled)

# Step 5: Visualize clustering results
sns.pairplot(df, hue='Cluster', vars=numeric_features)
plt.title("Clustering Results")
plt.show()
```

**Step 7: Save Cluster data to DimCluster**

Inspect if the DimCluster table exists. Compute statistics for each cluster (Average Revenue, Average Quantity Sold, Average Gross Profit) for descriptive metadata.
Create a DataFrame with cluster metadata: ClusterKey: Unique identifier for each cluster. ClusterName: Name of the cluster. ClusterDescription: Description with summary statistics.
Save metadata to DimCluster using to_sql().

```python
# Check if the CLUSTER_DIM table exists
inspector = inspect(engine)
table_names = inspector.get_table_names()

if 'CLUSTER_DIM' in table_names:
    print("CLUSTER_DIM table exists. Replacing the data...")
    if_exists_option = 'replace'
else:
    print("CLUSTER_DIM table does not exist. Creating it...")
    if_exists_option = 'replace'  # The first write will create the table

# Define the cluster metadata based on metrics
cluster_metadata = []

# Iterate through each cluster and calculate basic statistics for naming and describing
for cluster_id in df['Cluster'].unique():
    cluster_data = df[df['Cluster'] == cluster_id]

    # Extract statistics for description (example: SalesRevenue and TotalQtySold)
    avg_revenue = cluster_data['SalesRevenue'].mean()
    avg_qty = cluster_data['TotalQtySold'].mean()
    avg_profit = cluster_data['GrossProfit'].mean()

```

```
        # Create dynamic names and descriptions
        cluster_name = f"Cluster_{cluster_id + 1}"
        cluster_description = (
            f"Cluster {cluster_id + 1}: "
            f"Average Revenue: {avg_revenue:.2f}, "
            f"Average Quantity Sold: {avg_qty:.2f}"
            f"Average Gross Profit: {avg_profit:.2f}"
        )

        cluster_metadata.append((cluster_id + 1, cluster_name,
            cluster_description))

    # Convert to DataFrame
    cluster_dim_df = pd.DataFrame(cluster_metadata, columns=['ClusterKey
        ', 'ClusterName', 'ClusterDescription'])

    # Save to CLUSTER_DIM table (replace or create)
    cluster_dim_df.to_sql('CLUSTER_DIM', con=engine, if_exists=
        if_exists_option, index=False)
    print("Cluster metadata saved to CLUSTER_DIM.")
```

**Step 8: Update ClusterKey to Product Mix Fact** Create a mapping of ProductKey to ClusterKey. Merge the existing fact table with the mapping. Use pd.merge() with a left join to retain existing fact data while adding the new ClusterKey column.
Save the updated fact table back to the database using to_sql().

```
    # %% Update PRODUCT_MIX_FACT table with ClusterKey
    product_cluster_mapping = df[['ProductKey', 'Cluster']].copy()
    product_cluster_mapping['ClusterKey'] = product_cluster_mapping['
        Cluster'] + 1

    # Merge or update instead of replacing the entire table
    existing_fact = pd.read_sql('SELECT * FROM PRODUCT_MIX_FACT', engine
        )
    updated_fact = pd.merge(existing_fact, product_cluster_mapping, on='
        ProductKey', how='left')
    updated_fact.to_sql('PRODUCT_MIX_FACT', con=engine, if_exists='
        replace', index=False)

    print("Updated PRODUCT_MIX_FACT with ClusterKey.")
```

# 8 Business Intelligence (BI) Dashboard

## 8.1 Dashboard Design and KPIs

After obtaining all the necessary calculated results from the analysis, we need to have a way to display all this information in an effective and clear way for other people to easily understand and can apply them for other purposes. Since this project focuses mostly on the product side and the customer side when working with product mix, some diagrams that can visualize periods and make comparisons are recommended for designing a good dashboard, and we decided to mostly use line charts for displaying periods and bar charts for comparisons.

For the key performance indicators (KPIs), we will use our predefined business metrics (Sales Revenue, Gross Profit, Sales Volume Growth Rate, and Average Selling Price) and display them throughout specific periods of time to improve better decision making for managing products and declaring suitable sale prices. On the other hand, for the customer side, we can measure their behaviour according to the RFM model around each point of time for the purpose of adapting to fulfillment and satisfaction of customers.

## 8.2 Implementation

So far, we have applied the ETL process to manage and tidy up messy data from the Company X database, by using Data Integration from Microsoft Visual Studio, in order to obtain mandatory data required for calculating the business metric of this project. Now for the visualization part, we will use Power BI, which is also from Microsoft, to collect all the data calculated and display them as comprehensible diagrams in a dashboard. This is where a large amount of data can appear human readable and support other purposes such as deep analysis, real-time comparisons, determination of future plans, foreseeing rare opportunities, etc. Moreover, these softwares are free to download.

## 8.3 Visualization and Insights

When we have a complete dashboard ready, we can describe all the research information discovered and provide actionable insights in patterns, trends, and anomalies in the product mix. After fully understanding the product mix of the company based on data collected, we can bring out specific, data-driven recommendations for optimizing the product mix, and also give suggestions on how to improve products and sales that satisfy the needs from customers for more positive feedback and useful information.

## 9 Use Case for Business Decision-Making

After the analysis, it is visible that some subcategories are underperforming while others show outstanding results. This insight can be specifically used to make informed decisions about extending a successful product line. For example, if a particular subcategory consistently demonstrates high sales revenue and profit margins, the business can prioritize developing new variations or complementary products within that subcategory. This allows the company to capitalize on existing demand, strengthen its market position, and drive additional revenue by catering to customer preferences identified through the analysis.

## 10 Business Model Application

### 10.1 RFM Model Overview

The RFM model is a customer segmentation framework that evaluates three key dimensions of customer behavior: Recency (R), Frequency (F), and Monetary value (M).

- Recency measures the time since a customer's last interaction or purchase. Customers who recently purchased are more likely to respond to marketing efforts.
- Frequency reflects how often a customer makes purchases within a given timeframe. High-frequency customers typically have greater loyalty.
- Monetary assesses the total spending of a customer. Higher spending indicates higher-value customers.

By leveraging these dimensions, the RFM model enables businesses to group customers into distinct segments, tailoring marketing strategies and enhancing product mix efficiency. This model is particularly suited to analyzing sales performance and optimizing product line offerings.

### 10.2 Application and Insights

During the process of creating Customer dimension, we implement a SQL script to compute the RFM metrics as well as the scores. The script source is listed below.

```sql
WITH RFM_Calculated AS (
    SELECT
        CustomerID,
        DATEDIFF(DAY, MAX(OrderDate), GETDATE()) AS Recency, -- Days since the last order
        COUNT(SalesOrderID) AS Frequency,                    -- Total number of orders
        SUM(TotalDue) AS Monetary,                           -- Total amount spent
        MAX(ModifiedDate) AS ModifiedDate                -- Latest ModifiedDate
    FROM Sales.SalesOrderHeader
    GROUP BY CustomerID
),
RFM_Normalized AS (
    SELECT
        CustomerID,
        NTILE(5) OVER (ORDER BY Recency ASC) AS RecencyScore,     -- Rank Recency into 5 groups (1=most recent)
        NTILE(5) OVER (ORDER BY Frequency DESC) AS FrequencyScore, -- Rank Frequency into 5 groups (5=highest frequency)
        NTILE(5) OVER (ORDER BY Monetary DESC) AS MonetaryScore,  -- Rank Monetary into 5 groups (5=highest spend)
        ModifiedDate
    FROM RFM_Calculated
)
SELECT
    CustomerID,
    CAST(RecencyScore AS VARCHAR(1)) + '-' +
    CAST(FrequencyScore AS VARCHAR(1)) + '-' +
    CAST(MonetaryScore AS VARCHAR(1)) AS RFMScore, -- Combine scores into R-F-M format
    RecencyScore,
    FrequencyScore,
    MonetaryScore,
    ModifiedDate
FROM RFM_Normalized
ORDER BY RFMScore DESC;
```

For the topic of product mix and product line sales performance analysis, the RFM model provides actionable insights by identifying customer segments with varying behaviors and preferences.

1. **Data Preparation:**
   Extract transactional data, including purchase history, product categories, and sales amounts.
   Calculate RFM scores for each customer based on standardized criteria for Recency, Frequency, and Monetary value.

2. **Segmentation Process:**
   Divide customers into quintiles or predefined bins for each RFM dimension.
   Assign RFM scores and create customer segments, such as "Champions," "Loyal Customers," "Potential Loyalists," and "At-Risk Customers."

3. **Analysis of Product Line Performance:**
   Evaluate sales contributions from each RFM segment across different product lines.
   Identify top-performing product lines among high-value customer segments.
   Highlight underperforming product lines that resonate with at-risk or infrequent customers.

4. **Strategic Recommendations:**
   Develop targeted promotions for high-value segments to further enhance product line sales.
   Cross-sell or bundle complementary products to increase average order value among medium-value segments.
   Offer to engage low-value or at-risk segments and encourage re-engagement with the brand.

# 11 Conclusion

## 11.1 Summary

This project focuss on developing a Business Intelligence (BI) system for product mix analysis. The system will enhance data-driven insights to optimize product strategies. Key stages include data source analysis, designing a data warehouse, implementing an ETL pipeline, ensuring near real-time updates, and creating visualizations for actionable insights. Key deliverables are:

- A well-structured data warehouse to consolidate critical business data.
- An efficient ETL pipeline for data integration and updates.
- A result from data mining techniques, in this case, clustering methods, to enhance the product mix.

## 11.2 Future Work

For further development in the future, we consider applying more clustering method, such as model-based clustering, density-based clustering, grid-based clustering. Evaluation and assessment of mining techniques should also be conducted. Evaluating datamining clustering results is crucial for several reasons:

1. Validation of results: determining if the identified clusters are truly distinctive and meaningful, or if they algorithm generate arbitrary groupings.
2. Selection of optimal algorithm: Different clustering algorithms (k-means, hierarchical, DBSCAN, etc.) may produce different results on the same dataset. Evaluation allow compare the performance of different algorithms and select the one that best appropriate.
3. Interpretation of results: insights into the characteristics and properties of the identified clusters, helping to interpret them and analyze them later.
4. Assessment of model quality: metrics provide a quantitative measure of the quality of the clustering solution, enable comparison between different data sets, algorithms, and parameter settings.

## 12 References

1. GeeksforGeeks. (2024). ***Clustering Metrics in Machine Learning.*** [Online].
   Available at: https://www.geeksforgeeks.org/clustering-metrics/

2. IBM. (2024). ***What is clustering?*** [Online].
   Available at: https://www.ibm.com/think/topics/clustering

3. J. Han, M. Kamber, and J. Pei, ***Data Mining: Concepts and Techniques***, 3rd edition, Morgan Kaufmann Publishers, 2012.

4. J.-T. Wei, S.-Y. Lin, and H.-H. Wu, ***A review of the application of RFM model***, African Journal of Business Management December Special Review, vol. 4, 1 no. 36, pp. 4199-4206, 2010. [Online].
   Available at:
   https://www.researchgate.net/publication/228399859-
   _A_review_of_the_application_of_RFM_model

5. Karan S. (2022). **Customer Segmentation - RFM, EDA, KNN(95%).** [Online].
   Available at: https://www.kaggle.com/code/karan842/customer-segmentation-rfm-eda-knn-95#Calculating-RFM-Score

6. R. Kimball and M. Ross, **The Data Warehouse Toolkit**, 3rd ed., Wiley Publishing, Inc., 2013.

7. T. Zwingmann, **AI-Powered Business Intelligence**, Kindle Edition, O'Reilly Press, 2022.

**THE END**