VIETNAMESE GERMAN UNIVERISTY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEER

Current Topic in Computer Science

PRODUCT MIX AND CLUSTERING

Instructor: Dr. Phan Trong Nhan

Student 1:	Huynh Dang Trinh	10421064
Student 2:	Phan Khanh Nghi	10421094
Student 3:	Nguyen Hoang Nguyen	10421096
Student 4:	Huynh Minh Tuong	10421088
Student 5:	Trinh The Hao	10421017

December 22, 2024

Abstract

This project explores the design and implementation of a data warehouse tailored to analyze the Product Mix of Company X. The objective is to leverage data warehouse techniques to provide strategic insights into the product offerings and their performance across various categories. By organizing product data in a star schema, the system enables efficient querying and reporting, facilitating better decision-making in inventory management, marketing strategies, and sales forecasting. Key elements include the central Fact Table tracking product sales and associated metrics, linked to Dimension Tables detailing product categories, time, promotion, and store demographics. The findings demonstrate how a well-structured data warehouse can enhance data-driven decision-making in modern enterprises.

Content

1 Introduction **3**
1.1 Product Mix Strategy . . . 3
1.2 Components of a Product Mix Strategy . . . 3
1.3 Benefits of a Well-Planned Product Mix Strategy . . . 4
1.4 Product Mix Diagram . . . 4
1.5 Problem Statement . . . 6

2 Topic Overview **6**
2.1 Background on Product Mix Strategy . . . 6
2.2 Elbow Method & Applications of K-Means Clustering . . . 7
2.3 Use Cases . . . 8

3 Star Schema Design **8**
3.1 Star Schema . . . 8
3.2 Relationship Between Fact Table & Dimension Tables . . . 9

4 Bussiness Metrics **10**
4.1 Return on Investment . . . 10
4.2 Revenue . . . 10
4.3 Net Profit Margin . . . 11

5 ETL Pipeline Using SSIS **12**
5.1 Data source collection . . . 12
5.2 Dimension initialization . . . 17

6 Data Mining **29**
6.1 The importance of Data Mining . . . 29
6.2 Clustering Techniques . . . 29
6.2.1 Introduction of KMeans and DBSCAN . . . 29
6.2.2 Clustering Algorithm Implementation . . . 29
6.2.3 Comparison between KMeans and DBSCAN . . . 53

7 Power BI presentation **54**

8 Member's Contribution **55**

1 Introduction

Company X's data is immensely large and is unable to handle its already growing product portfolio which includes different product categories like bikes, accessories, clothing, and components. The problems are missing the ability to easily recognize the best-performing product, be comprehensible on the sales performance in different regions as well as be able to target stocking and marketing decisions based on data. Such flaws compromise the company's chances of making maximum revenue, an optimized return on investment as well as increasing the net profit margin. This project denotes the discrepancy that the data can yield trends, insights, and correlations if a purpose-designed data warehouse and specific business intelligence applications such as clustering techniques and power BI are utilized. Through efficient analysis of clusters based on key performance indicators, Company X would know the right volumes to stock at the right time making it easy to plan marketing campaigns and stock the most profitable items. The project expects that mapping the trends and financial metrics of every region and every category will provide better information in order to ensure development and business penetration.

1.1 Product Mix Strategy

A product mix strategy is a thorough plan that a business creates to maximize its product offers by taking several aspects like resources, competition, and market demand into account. It is an essential component of product management and holds significant importance in the overall marketing strategy of a business. In order to satisfy the demands of their target market and optimize profitability and growth, companies can use the product mix strategy to help them choose the combination of items to offer.

1.2 Components of a Product Mix Strategy

A comprehensive product mix strategy includes four primary components that help businesses shape and refine their offerings to meet market demands and enhance profitability. These components are:

- **Product Line Length:** This aspect involves the total number of products within a specific product line. Companies can opt for a shorter product line with fewer products or a longer one with many products, depending on factors like target market needs, production capabilities, and resource availability.
- **Product Line Depth:** Depth refers to the variety of versions or variations of a single product within a product line. Companies can maintain a shallow product line depth with limited variations or a deep product line depth with numerous options, allowing them to cater to specific customer preferences and needs.
- **Product Line Width:** The width of a product mix refers to the number of distinct product lines a company offers. A wider product mix allows a company to diversify its offerings,

cater to a broader range of customer needs, and reduce business risks by not relying heavily on a single product line. For instance, a company with product lines in electronics, clothing, and home appliances has a broader product mix width compared to one focused solely on electronics.

- **Product Line Consistency:** Product mix consistency measures how closely related the various product lines are in terms of use, production, and distribution. High consistency ensures operational efficiency and a unified brand identity, as seen in companies like Apple, which maintains consistent design, technology, and ecosystem across its product lines. In contrast, low consistency can reflect a more diverse portfolio targeting entirely different markets, such as a conglomerate offering both financial services and consumer goods.

By carefully considering these components, businesses can create a balanced and customer-focused product mix strategy that drives growth and enhances their competitive edge.

1.3 Benefits of a Well-Planned Product Mix Strategy

A thoughtfully implemented product mix strategy can offer several key benefits to a business, such as:

- Expanding market share by appealing to a wider range of customers.
- Boosting profitability by offering products at varied price levels, and accommodating different customer segments.
- Increasing customer satisfaction through a diverse selection of products that meet varying needs and preferences.
- Boosting brand reputation by providing high-quality products and keeping pace with industry trends.
- Lowering risk by diversifying the product portfolio, reducing dependency on any single product for revenue.

1.4 Product Mix Diagram

```sql
SELECT DISTINCT
    c.[ProductCategoryID] AS CategoryID,
    c.[Name] AS CategoryName,
    s.[ProductSubcategoryID] AS SubcategoryID,
    s.[Name] AS SubcategoryName
FROM [CompanyX].[Production].[ProductCategory] c
LEFT JOIN [CompanyX].[Production].[ProductSubcategory] s
    ON c.[ProductCategoryID] = s.[ProductCategoryID];
```

SQL Query to Retrieve Product Categories and Subcategories

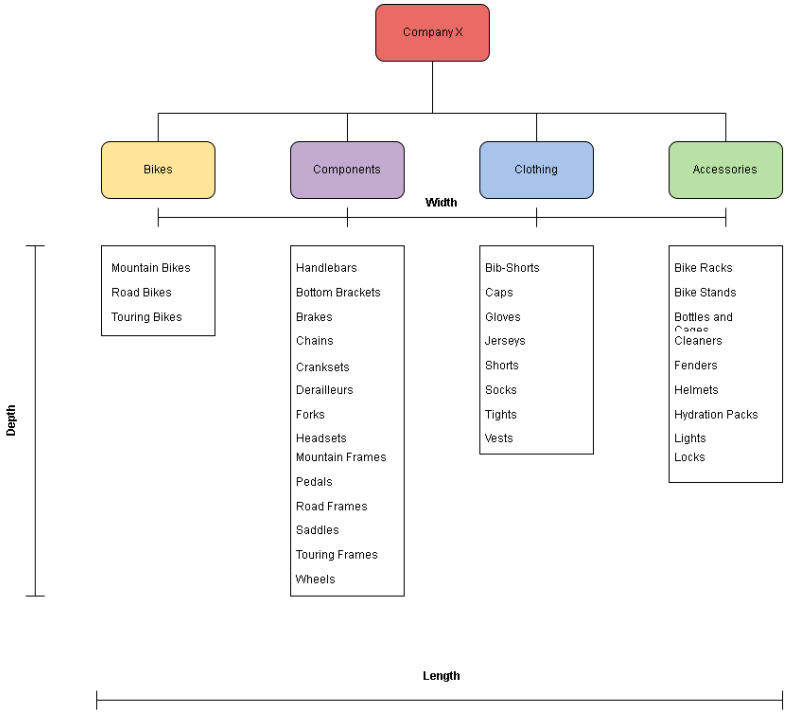


Figure 1: Product Mix Diagram of Company X

The diagram represents the **product mix** of Company X, illustrating the range of products it offers. The product mix is organized into the following dimensions:

1. **Width:**
   This refers to the number of product lines offered by the company. Company X's product mix includes four broad categories:
   - **Bikes**
   - **Components**
   - **Clothing**
   - **Accessories**
2. **Depth:**
   This indicates the variety within each product line. Each main category is further broken down into subcategories:
   - **Bikes:** Mountain Bikes, Road Bikes, Touring Bikes.
   - **Components:** Handlebars, Chains, Pedals, Wheels, and more.
   - **Clothing:** Bib-Shorts, Jerseys, Gloves, Socks, etc.
   - **Accessories:** Bike Racks, Helmets, Locks, Lights, etc.
3. **Length:**
   This reflects the total number of items offered across all categories and subcategories. It demonstrates the diversity and comprehensiveness of the product offerings.

This structure highlights the company's strategic approach to meeting a wide range of customer needs by providing both breadth (product categories) and depth (variety within categories). It ensures comprehensive coverage of products, from specialized bikes to supporting components, apparel, and essential accessories.

1.5 Problem Statement

The variety and scope of product mix data present a considerable problem for companies to effectively extract relevant insights and develop data-driven decisions. However, conventional approaches limit their ability to manage inventories, marketing, and strategic designs because they are unable to offer significant information on key performance measures like revenue, profit margin, and return on investment. These problems are addressed in this project by dividing products into sense-based clusters utilizing clustering techniques, such as K-Means and DBSCAN. Through the process of uncovering patterns and connections in the data, this method helps us find product groups with high performance and product groups with low performance, which Company X can use to remedy its product mix strategy, and that can lead to an overall improvement of business performance.

2 Topic Overview

2.1 Background on Product Mix Strategy

A product mix strategy involves managing a company's entire range of products to meet market demands, maximize profits, and strengthen brand positioning. It includes decisions related to product lines, product items, and the overall product portfolio.

- **Types of Product Mix Strategies:**

  1. **Expansion:**

     Adding new product lines or extending existing ones to attract new markets or increase market share.

  2. **Contraction:**

     Reducing the number of products by discontinuing underperforming items to focus on profitable ones.

  3. **Modification:**

     Improving or updating existing products to maintain competitiveness.

  4. **Diversification:**

     Introducing entirely new product categories to reduce risk and increase market opportunities.

  5. **Branding and Positioning:**

     Creating a strong brand identity and aligning products with specific market segments.

2.2 Elbow Method & Applications of K-Means Clustering

One of the partitioned algorithms is the K-means algorithm, which defines the initial centroid value to determine the starting number of groups.(Eltibi & Ashour, 2011) The K-means algorithm, which uses the process repeatedly, is one clustering strategy. The most straightforward and widely used clustering technique is K-Means. Large volumes of data can be grouped using K-means in comparatively quick and effective computation time.(Syakur et al., 2018)Implementing the elbow method is uncomplicated if you look at the ideal k-value graph that shows the elbow position and an SSE (Sum of Square Error) of less than 1. The basis for clustering will be the best cluster k result. The better the cluster results, the lower the elbow graph and the smaller the SSE value.(Syakur et al., 2018)Since the initial cluster center may shift, the K-Means technique needs exact figures to determine the number of clusters, k, as this could lead to unstable data grouping.(Likas et al., 2003)

The steps to implement the Elbow Method are as follows:

Step 1: Run K-Means for Multiple K Values

Apply K-means clustering for a range of K values (e.g., 1 to 10). For each K, compute the Within-Cluster Sum of Squares (WCSS), which measures the total distance of data points from their respective cluster centroids.

The formula for WCSS:

$$\text{WCSS} = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$$

Where:

- x = data point
- μ_i = centroid of cluster C_i
- k = number of clusters

Figure 2: Formula

Step 2: Plot K vs. WCSS

- On the x-axis, plot the number of clusters K.
- On the y-axis, plot the corresponding WCSS values.

Step 3: Identify the "Elbow" Point

- The graph will show a decreasing curve of WCSS as K increases.
- Look for the "elbow" point where the rate of decrease sharply slows down.
- This point indicates the optimal K, where adding more clusters gives diminishing returns.

2.3 Use Cases

The data warehouse and clustering methods proposed in this project, combined with Power BI visualizations, can be utilized in several scenarios:

- **Revenue Analysis:** By employing Power BI's cluster-based pie charts and revenue breakdown. The company is able to know which of its product categories is the most profitable and which is least profitable allowing it to make necessary and strategic moves to better the overall business financials.
- **Category Trends:** Vertical Line Graphs comparing revenues, net profit margins, and even ROI for various time periods would enable the company to make projections about the future performance of specific product categories and revise their marketing strategies accordingly.
- **Regional Insights:** The slicer function in Power BI enables regional filtering which helps to understand and gain insights on regional markets and also helps in customizing products and promotions offered to specific regions.
- **Cluster-Specific Strategies:** The clustering breakdown provides a useful guide on the proportions of the revenue added by different clusters, thus helping to plan targeted marketing or operational refinements for underperforming clusters.
- **Marketing and Promotions:** By analyzing ROI trends, the company is able to suitably plan their promotion as well as budget efforts across the various product categories and regions more efficiently.

3 Star Schema Design

Star schema is a database design commonly used in data warehousing to optimize query performance and simplify the structure of the data. It organizes data into a central fact table connected to multiple-dimension tables, forming a star-like pattern.

3.1 Star Schema

In order to show the structure of the data utilized when building the data warehouse, the Star Schema was created. The contents of the schema :

1. **Fact Table:**
    - Fact: Contains the foreign keys of others' dimensions and 3 chosen metrics.
    - Columns:
    - Foreign keys: Time ID, Product ID, Store ID, Promotion ID (linking to dimension tables).
    - Measures: Total revenue, Return on investment, Profit margin.

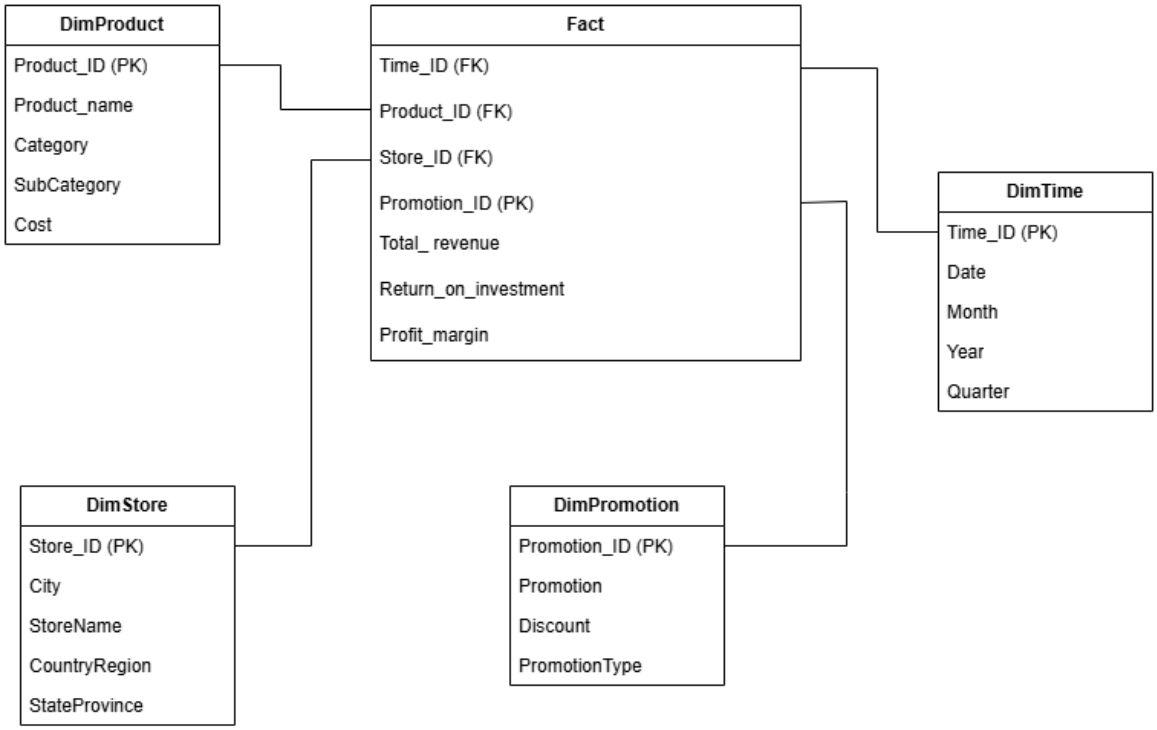


Figure 3: Star Schema

2. **Dimension Tables:**

   - **DimProduct:** Attributes: Product ID (PK), Product name, Category, SubCategory, Cost.
   - **DimStore:** Attributes: Store ID (PK), City, StoreName, CountryRegion, StateProvince.
   - **DimTime:** Attributes: Time ID (PK), Date, Month, Year, Quarter.
   - **DimPromotion:** Attributes: Promotion ID (PK), Promotion, Discount, PromotionType.

3.2 Relationship Between Fact Table & Dimension Tables

- **The Fact Table** contains the quantitative data or measures that are being analyzed. It represents the business metrics or transactional data of an organization, such as sales, profit, revenue, or other key performance indicators (KPIs).
- **The Dimension Tables** provide descriptive context to the facts in the Fact Table. They answer the who, what, where, and when questions in relation to the measures in the Fact Table. Each Dimension Table contains attributes that can be used to slice, dice, filter, or group the data for analysis.

4 Bussiness Metrics

Among the various business performance indicators, three key metrics have been selected: Return on Investment (ROI), Revenue, and Net Profit Margin. These metrics provide valuable insight into the company's financial health, profitability, and its overall efficiency. ROI evaluates the profitability of investments, Revenue measures the total income generated from business operations, and Net Profit Margin indicates the company's ability to convert revenue into profit after covering all expenses. Together, these metrics offer a comprehensive overview of a business's performance and growth potential.

4.1 Return on Investment

Return on Investment (ROI) is a common measure for judging the economic performance of a project. It is a measure of the net output of a project, presented as a percentage, determined by dividing its total inputs (total cost) by its net budget. The net output is calculated from cost (and/or revenue) savings/new revenue brought by a project, compared to total project cost (Jeffery, 2008). ROI offers a direct and unambiguous quantification to see if a project generates enough financial return as expenditure, a tool to evaluate a project's overall profitability.

In the domain of information technology projects, ROI is a very important aspect of investment decision-making. As found in a survey of 179 Chief Information Officers (CIOs), 59% of their firms routinely forecasted the ROI of IT projects before capitalizing on a purchase(Jeffery, 2008). This illustrates the widespread use of ROI as a means of financial evaluation and planning. Furthermore, 45% of the surveyed CIOs acknowledged that ROI was an essential component of the decision-making process, emphasizing its importance in aligning IT investments with organizational goals and financial accountability. Return on investment is presented as

$$ROI = \frac{\text{Net Profit} - \text{Investment Cost}}{\text{Investment Cost}} \times 100$$

Figure 4: Return on Investment (ROI)

where the project inputs are all of the project's expenses, and the project outputs are all of the benefits of the project, measured in terms of revenue production and cost savings.

4.2 Revenue

Revenue refers to the total amount of money generated from a company's business activities, typically through the sale of goods or services. It represents the top line of the income statement and serves as a critical indicator of a company's ability to generate sales and sustain operation (Hayes, 2024). Revenue is calculated before deducting expenses, taxes, and other costs, making it a key metric for assessing a business's market performance and growth potential.

The formula for calculating revenue is:

$$\text{Revenue} = \text{Units Sold} \times \text{Sales Price per Unit}$$

Figure 5: Revenue

Revenue reflects how well a company performs and grows. For example, Starbucks achieved $14.89 billion in revenue in the fiscal year 2013 due to its strategic product mix, which included high-quality coffee, tea, fresh food items, and beverages offered at 19,767 outlets worldwide (Geereddy, 2013). Incorporating similar concepts regarding company X, it is observable that turnover by product categories and clusters can also be useful. For example, concentrating on high-revenue clusters as a result of clustering analysis can assist the allocation of funds towards the products. On the other hand, comprehension of the low performers can assist in the readjusting of the product mix. Additionally, introduction of new products or development of existing ones in high demand clusters can help in revenue growth by extending the range of clients and satisfying the needs of the unsatisfied ones.

4.3 Net Profit Margin

Net Profit Margin is a key financial metric that measures a company's profitability by indicating how much net income is generated as a percentage of total revenue. It shows how efficiently a business manages its expenses relative to its sales. A higher net profit margin indicates strong financial health and operational efficiency, while a lower margin may signal cost management issues or declining sales performance. The proportion of sales that remains after all costs and expenses, including interest, taxes, and preferred stock dividends, have been subtracted is known as the net profit margin(Gitman, 2012). Therefore, a corporation might be deemed good if its Net Profit Margin (NPM) is high.

The formula for calculating Net Profit Margin is:

$$Net\ Profit\ Margin = \frac{EAT}{Net\ Sales} \times 100\%$$

Figure 6: Net Profit Margin

The profitability ratio known as net profit margin compares net income to sales and illustrates how well a business can control its operating expenses over a certain time frame(Sutrisno, 2013).NPM positively and significantly impacts the growth of a company's profits since it shows how well the business can control operating expenses(Puspasari et al., 2017). Based on the findings of the mentioned research NPM is expected to perform well in this project. Net profit margin

can demonstrate to the investor how well the company is doing based on the outcome. When businesses maintain a high NPM, investors can anticipate a larger return on their investment because this shows operational effectiveness and profitability(Gitman, 2012).

5 ETL Pipeline Using SSIS

5.1 Data source collection

1. **Product Dimension**

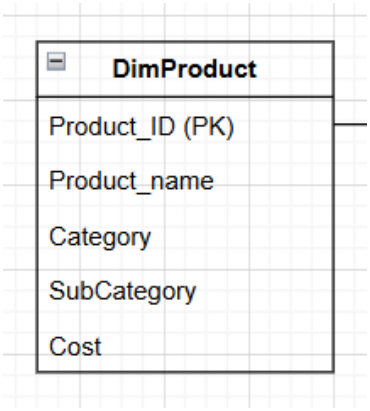


Figure 7: Product Dimension

DimProduct was created to get all the information about the targeted product. Gather details about the product is important for clustering and this dimension contains all the data gathered from our database by joining tables of Compan like the following figure:

By joining the tables the necessary data were found to construct the desired dimension. Through foreign keys, the tables were connected to reach the desired table for the data needed:

- **The name** of the product lies in the product table with the cost. However, the subcategory & category is located in 2 others which were connected using foreign keys.
- **Subcategory&category** also has the **name** attribute but that is not the name of the product in search, by collecting that name false data would have been collected.
- **Price** was labeled as **Standardcost** in the product table but the data got collected and renamed to **cost** to convince of reading.

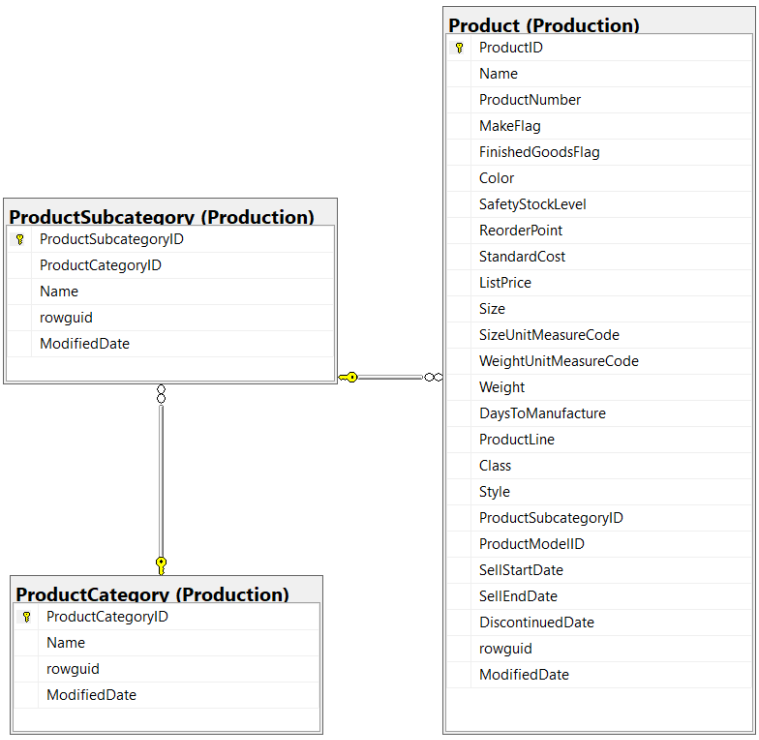


Figure 8: Product Diagram

2. **Store Dimension**

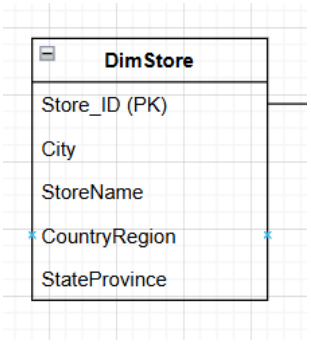


Figure 9: Store Dimension

The **store dimension** was created to contain the location of the chosen store. Knowing its position helps aid in determining which location has what sales and what product line should be advertised at the location.

Because multiple attributes were pulled out of the different tables as the figure shown below the store dimension have the most join in the course of our work:

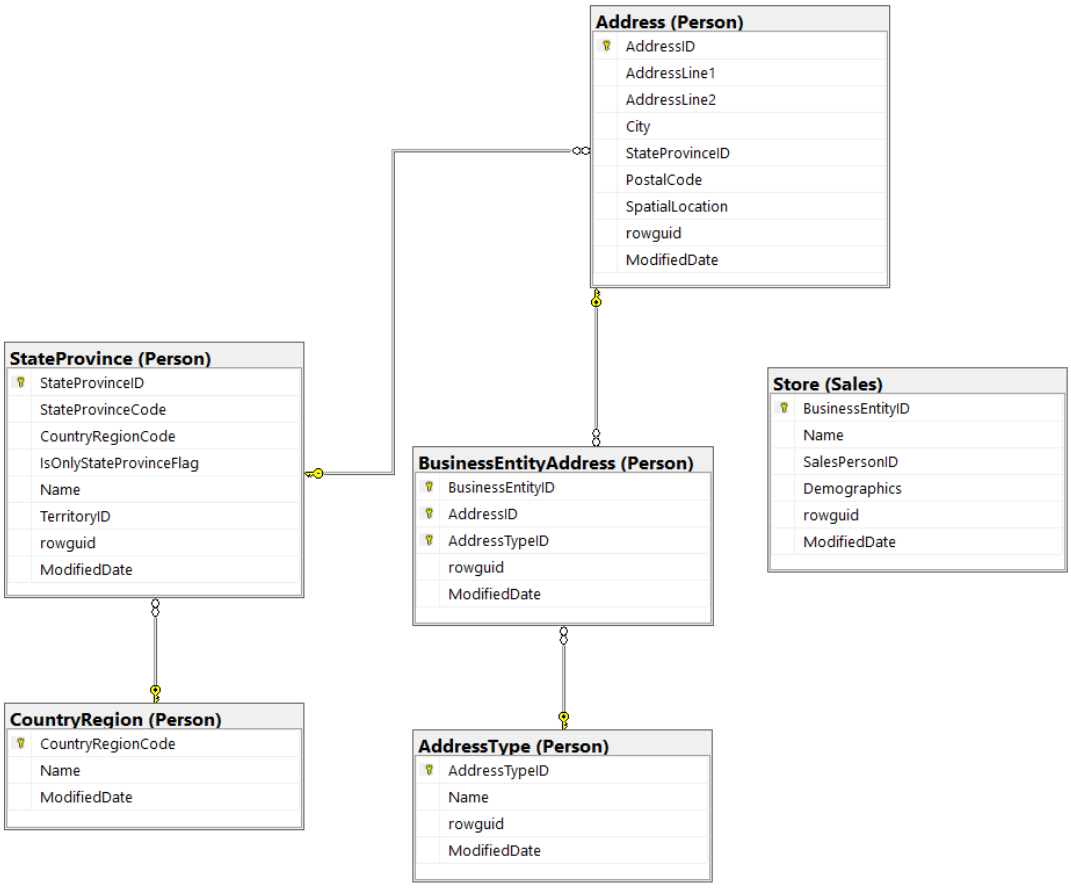


Figure 10: Store Diagram

- **City** located in the address table as shown. The targeted data is linked using foreign keys. Along with City another data also got collected in this table is **StateProvince**
- **Storename** Is the attribute found in the store table and was linked to BussinessEntityAddress using BussinessEntityID as its foreign key.
- **CountryRegion** data is needed at the CountryRegion table and was labeled as **Name** tracked by foreign key CountryRegionCode.
- **StateProvince** found in address .However it code is in the StateProvince table and the foreign key of the table is AddressID.

3. **Time Dimension**

   The time dimension was created to contain the details of the temporal aspects related to the data. Knowing the time attributes helps determine when specific events occurred, which time periods had the highest activity, and how performance trends evolve over time. These attributes allow for better analysis and reporting of data across various time periods, such as specific dates, months, quarters, and years.

   - **OrderDate** is an attribute located in the **SalesOrderHeader** table, and it contains the full date when the sales order was created. This data can be broken down into **Day**,

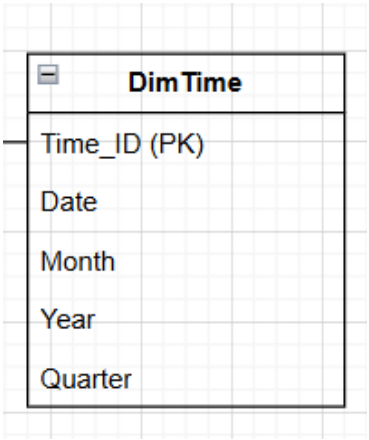


Figure 11: Time Dimension

Month, **Year**, and **Quarter** for analysis.

- **DueDate** is another date attribute in the table, which specifies the deadline for the order to be fulfilled. This can also be split into **Month**, **Year**, and **Quarter**.
- **ShipDate** indicates the date when the order was shipped and similarly can be divided into **Month**, **Year**, and **Quarter**.

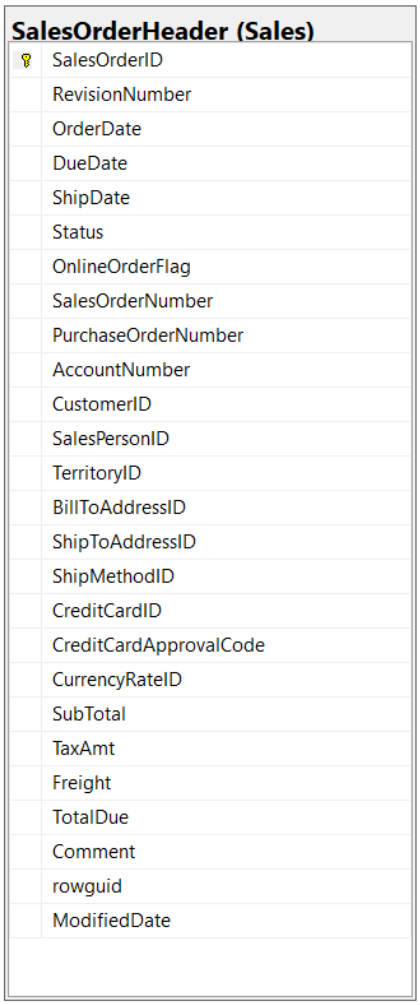


Figure 12: Time Diagram

4. **Promotion Dimension**

   The promotion dimension was created to store details about promotional activities and related discounts. Understanding these attributes helps identify which promotions drive the highest sales and how various promotional types influence customer behavior.

   Because promotions play a crucial role in marketing strategies, the figure shown below highlights the key attributes of this dimension, such as specific promotion names, discounts applied, and promotion types. This enables better analysis of promotional effectiveness and aids in planning future marketing campaigns.

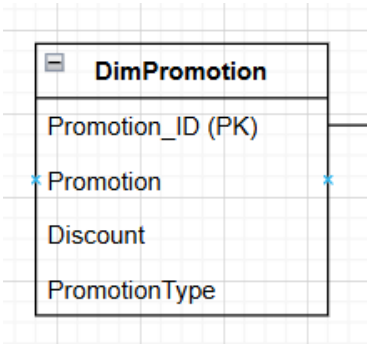


Figure 13: Promotion Dimension

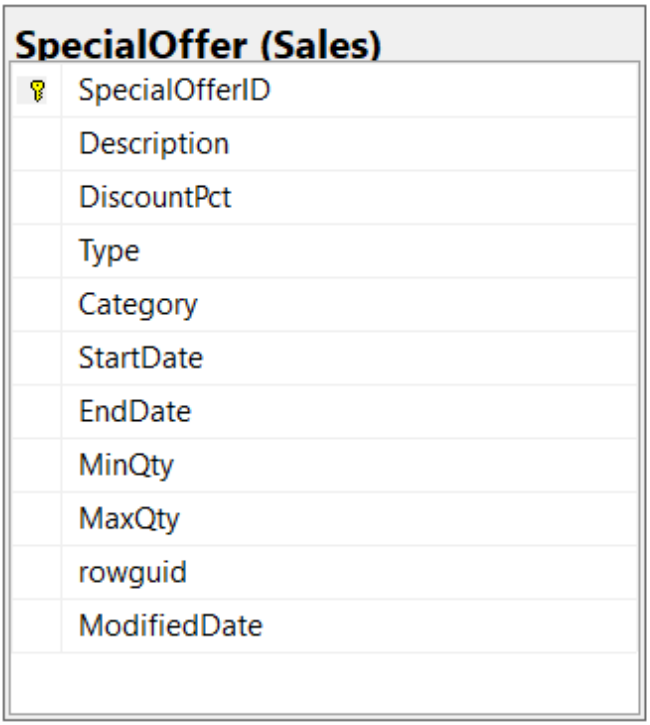


Figure 14: Promotion Diagram

- **Promotion** is represented in the table as the **Description** attribute, providing details about the specific promotion being offered.
- **Discount** is captured under the **DiscountPct** attribute, which specifies the percentage of the discount applied for the promotion.
- **PromotionType** is represented by the **Type** attribute, which classifies the promotion based on its nature or category.

5.2 Dimension initialization

- **Creating Blank Dimension Tables**

  The figure illustrates the process of creating the blank dimension table for our star schema The first 3 are for DimProduct, DimPromotion, and DimStore because most of the at-

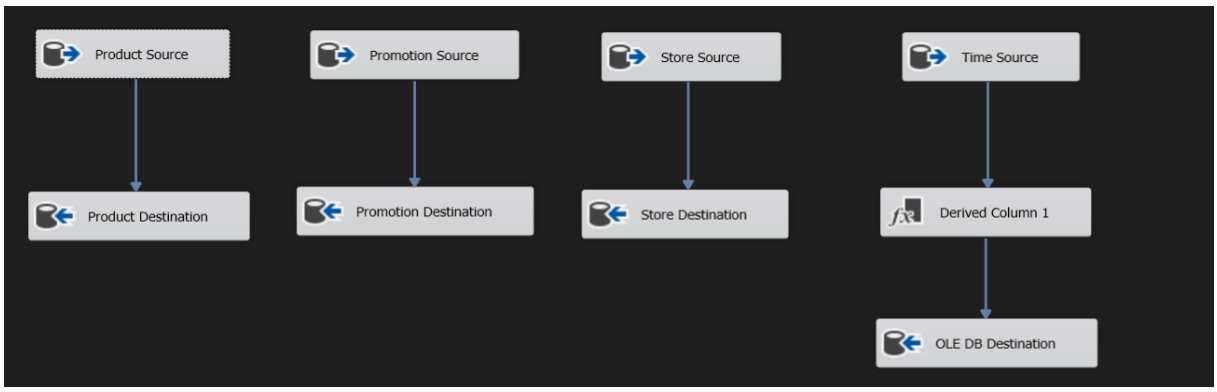


Figure 15: Creating Blank Dimension Tables

tributes needed to remain the same for these dimensions.

However, in Dimtime attributes such as Date, Month, Year, and Quater were used in another step. Derived column 1 is used to calculate the needed information for the dimension table

Derived Column Name	Derived Column	Expression	Data Type	L
Date	<add as new column>	DAY(SellDate)	four-byte signed inte...	
Month	<add as new column>	MONTH(SellDate)	four-byte signed inte...	
Year	<add as new column>	YEAR(SellDate)	four-byte signed inte...	
Quarter	<add as new column>	(MONTH(SellDate) - 1) / 3 + 1	four-byte signed inte...	

Figure 16: Derived column 1

To map the attribute from the database to the dimension tables the following SQL code were written in the Product Store for example.

```
SELECT DISTINCT
    p.ProductID,
    p.Name AS ProductName,
    ps.Name AS Subcategory,
    pc.Name AS Category,
    p.StandardCost AS Cost,
    p.ModifiedDate
FROM
    [CompanyX].[Production].[Product] AS p

LEFT JOIN
    [CompanyX].[Production].[ProductSubcategory] AS ps
    ON p.ProductSubcategoryID = ps.ProductSubcategoryID
LEFT JOIN
    [CompanyX].[Production].[ProductCategory] AS pc
    ON ps.ProductCategoryID = pc.ProductCategoryID
WHERE p.[FinishedGoodsFlag] = '1'
ORDER BY
```

```
19      p.ProductID;
```

SQL Mapping

The last step of completing this is to check the mapping for any error that may occur

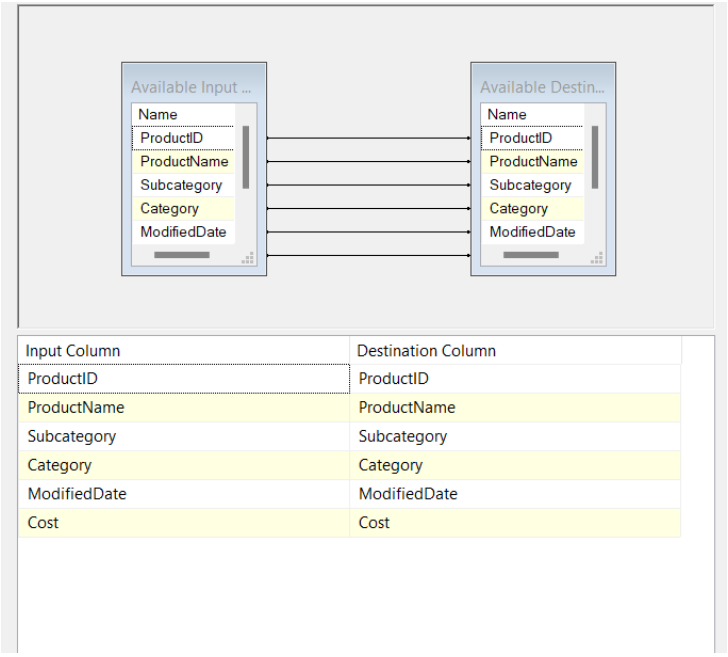


Figure 17: Mapping

After the initialization setup, these blank tables can be found in database

Figure 18: Blank Table Example

These tables have not yet had any data since reaching this step no data were loaded into them.

- **Execute SQL task**

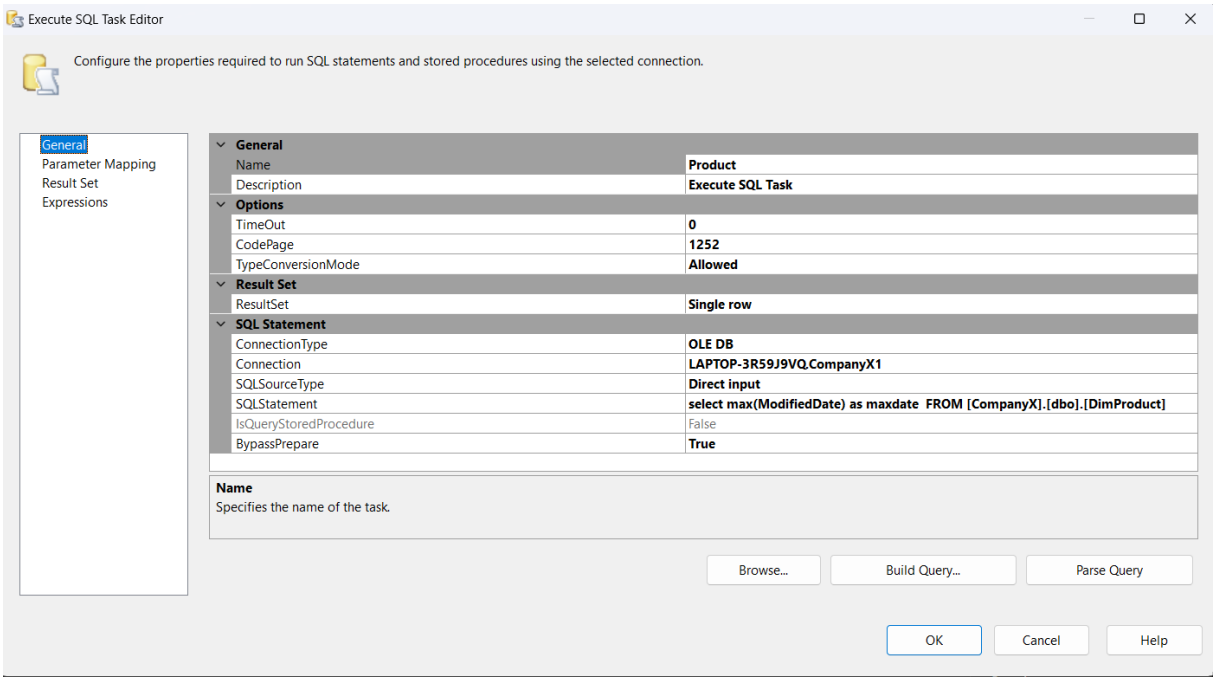


Figure 19: Maxdate generation

The execute SQL task Were used to generate maxdate for tracking which data is the latest.

```
1 select max(ModifiedDate) as maxdate
2   FROM [CompanyX].[dbo].[DimProduct]
```

Maxdates

Before loading data to the dimension table, data need to run through 2 gateways to identify which data is new, which data is old, and which data does not need to be replaced.

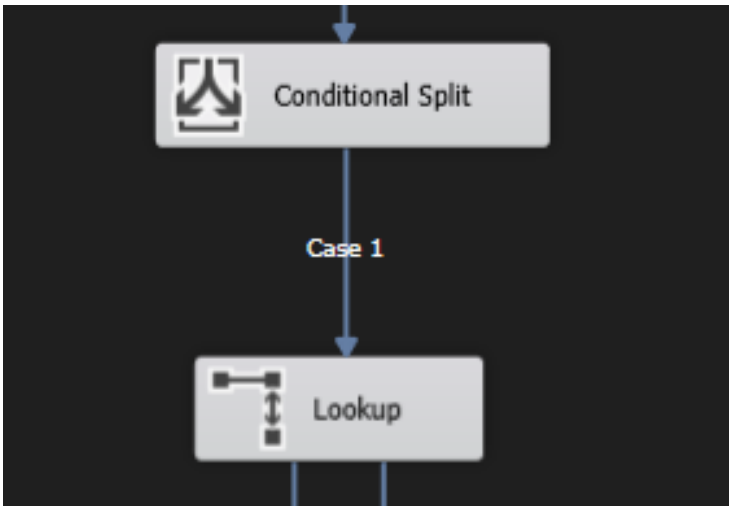


Figure 20: Data flow

The conditional split acts as a condition to identify which is the new data using the maxdate that have been generated before

Order	Output Name	Condition
1	Case 1	ISNULL(@[User::maxdate]) || (DT_DATE)ModifiedDate > (DT_DATE)@[User::maxdate]

Figure 21: Condition

This condition lay in the condition split This condition translated to the target data would be the one that the modified date is greater than the maxdate

Lookup is used to identify which data fit the condition above, 2 of them combine to target all the new updates for the next step

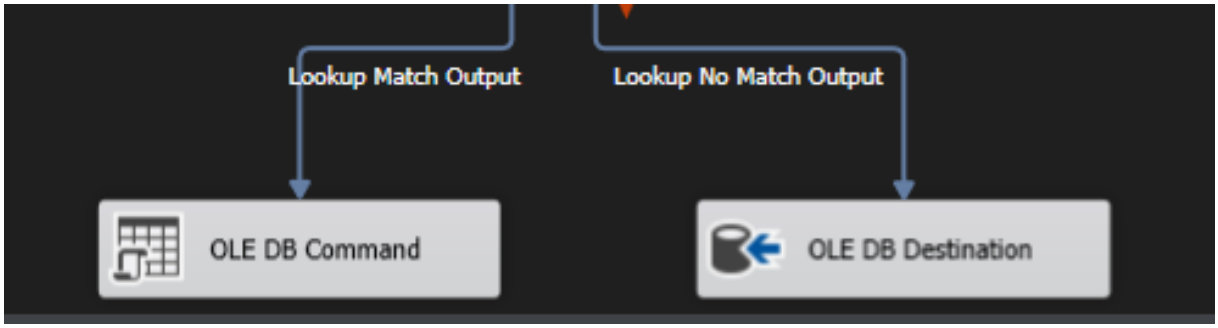


Figure 22: Path

If the data existed before and there is a newer version of it, it would go to the left If the data is completely new and needs to be loaded in it would go to the right These actions aim to aid the problem faced before which is duplication of the amount of data every time is ran .

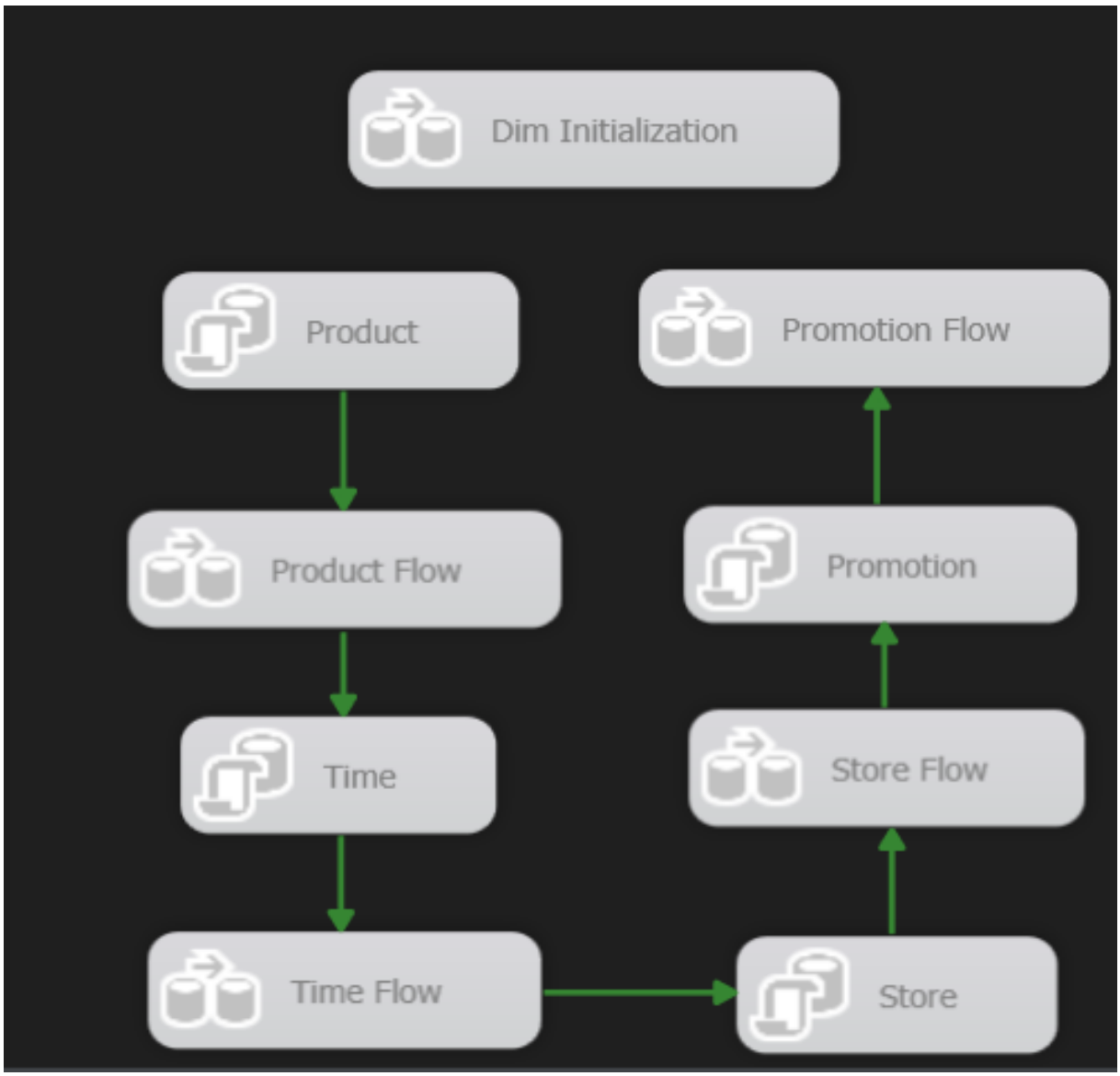


Figure 23: Data flow

After this dataflow is ran all the dimension table should be loaded with data

- **Source Initialization**

  To feed the necessary data for the fact table, the init step was repeated to create the Source. Unlike the dimension data that can be fetched from the database of CompanyX. The Source init has 3 derived columns each serving a purpose

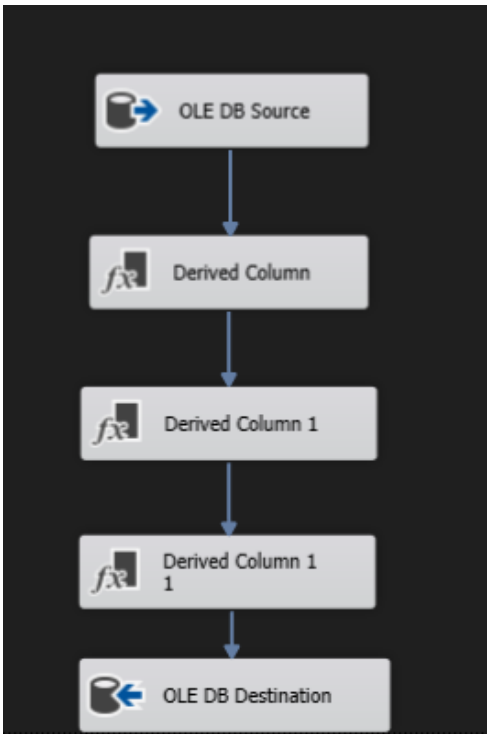


Figure 24: Source init

The first derived column was meant for calculating the data needed for the calculation of our 3 metrics

Derived Column Name	Derived Column	Expression	Data Type	L
Revenue	<add as new column>	SellingPrice * SellQty	numeric [DT_NUMERI...	
ImportCost	<add as new column>	Cost * SellQty	numeric [DT_NUMERI...	
InvestCost	<add as new column>	Cost * InventoryQty	numeric [DT_NUMERI...	
NetProfit	<add as new column>	(SellingPrice * SellQty) - (Cost * SellQty)	numeric [DT_NUMERI...	

Figure 25: Derived Column

The second derived column is for calculating 2 out of 3 of the selected metrics

Derived Column Name	Derived Column	Expression	Data Type
ROI	<add as new column>	(NetProfit / InvestCost) * 100	numeric [DT_NUMERI...
NetProfitMargin	<add as new column>	(NetProfit / Revenue) * 100	numeric [DT_NUMERI...

Figure 26: Derived Column

The last derived column is to get the date, month, and year for the fact table

Date	<add as new column>	DAY(OrderDate)	four-byte signed inte...
Month	<add as new column>	MONTH(OrderDate)	four-byte signed inte...
Year	<add as new column>	YEAR(OrderDate)	four-byte signed inte...
Quarter	<add as new column>	(MONTH(OrderDate) - 1) / 3 + 1	four-byte signed inte...

Figure 27: Derived Column

- **Source execute SQL task**

  Creating a Maxdate is as important as before to prevent data duplication

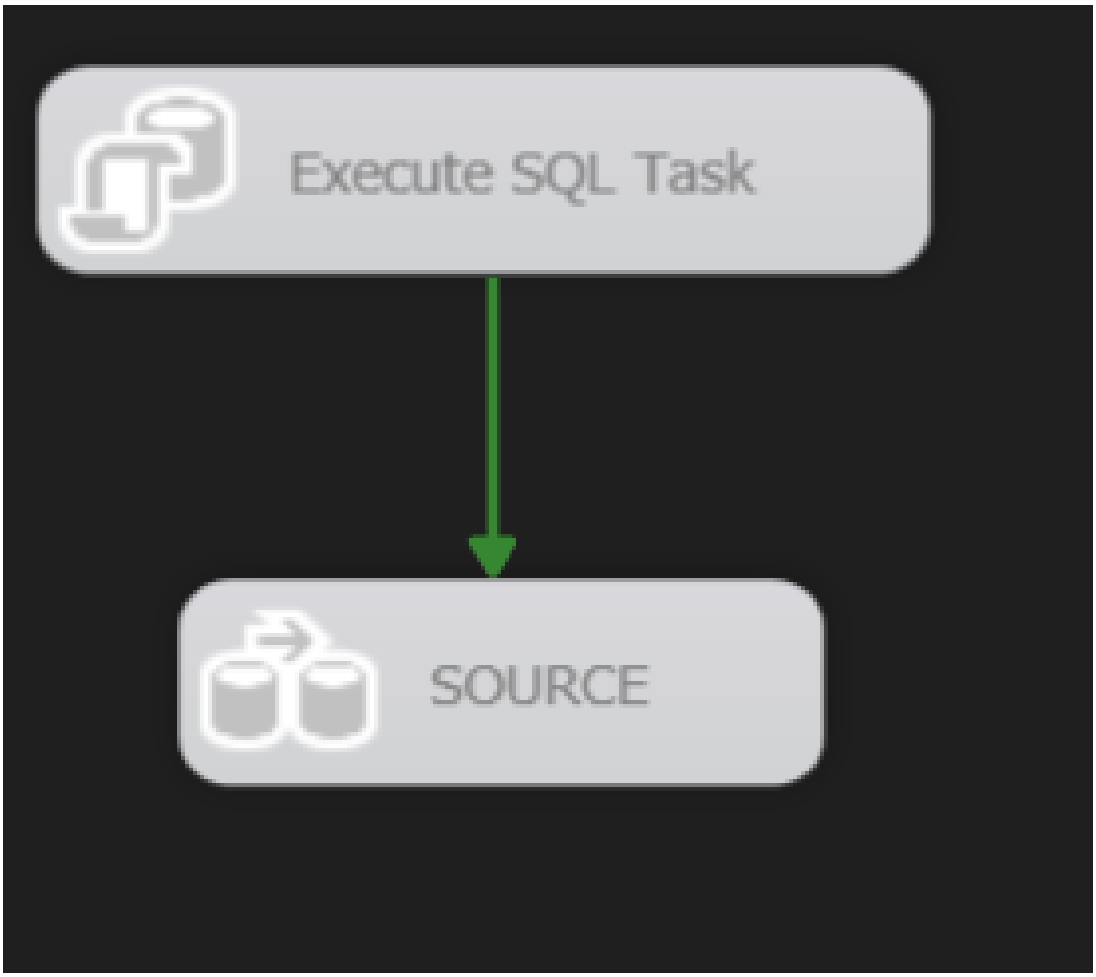


Figure 28: Data flow

The execute SQL task contains the SQL for generating Maxdate

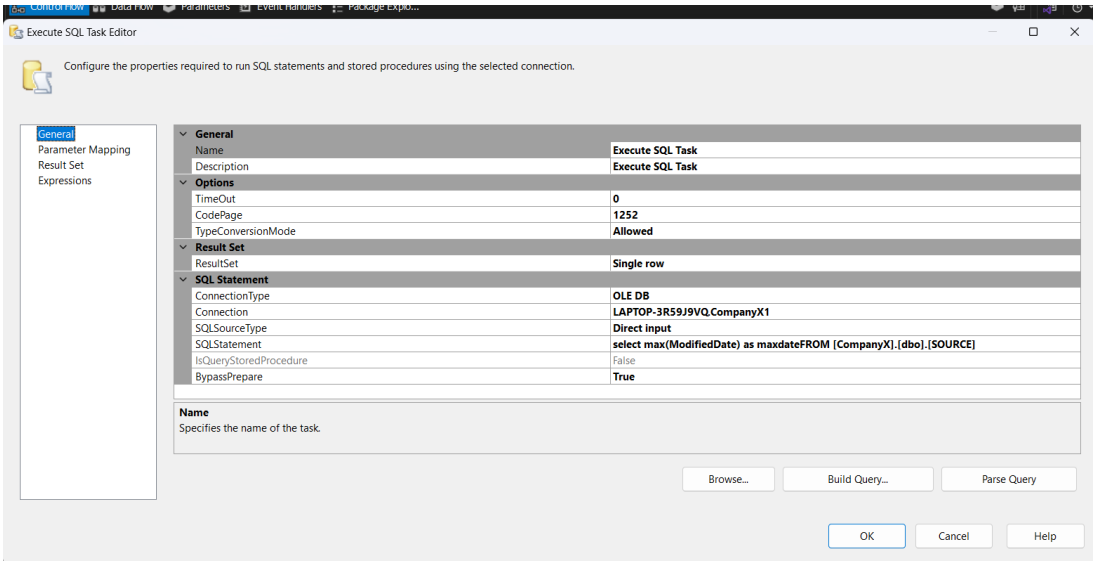


Figure 29: Maxdate

Finally, the step to load data to the source table

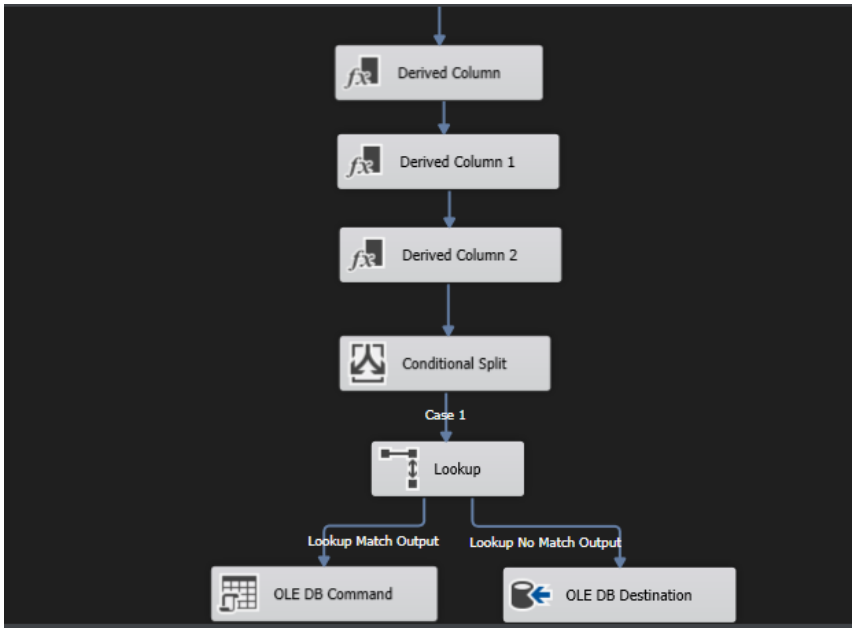


Figure 30: Update data

The logic used was the same to cluster data. If the data ID is detected it will only update the data if it is new, if no ID was found simply add new.

- **Fact table initialization& execute SQL task**

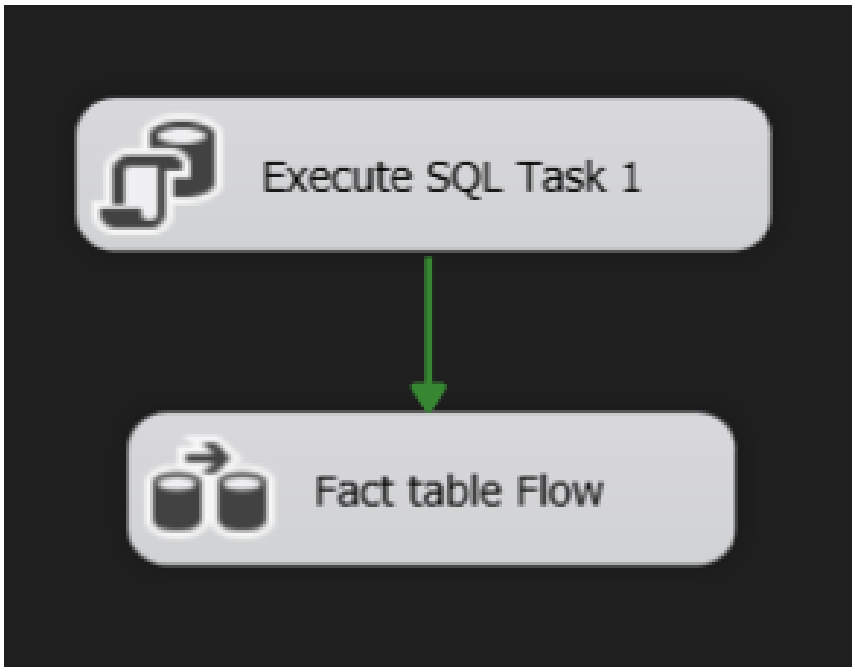


Figure 31: Data flow

Repeating the step generates maxdate

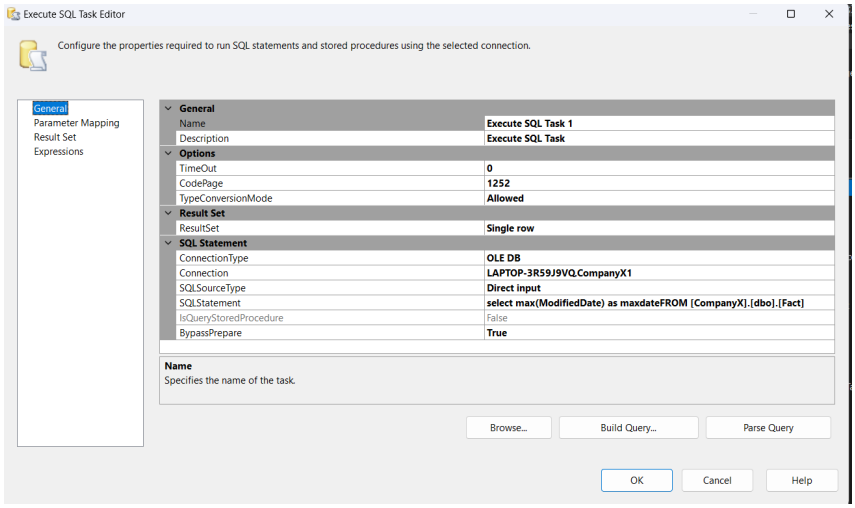


Figure 32: Maxdate

Then check the data before load

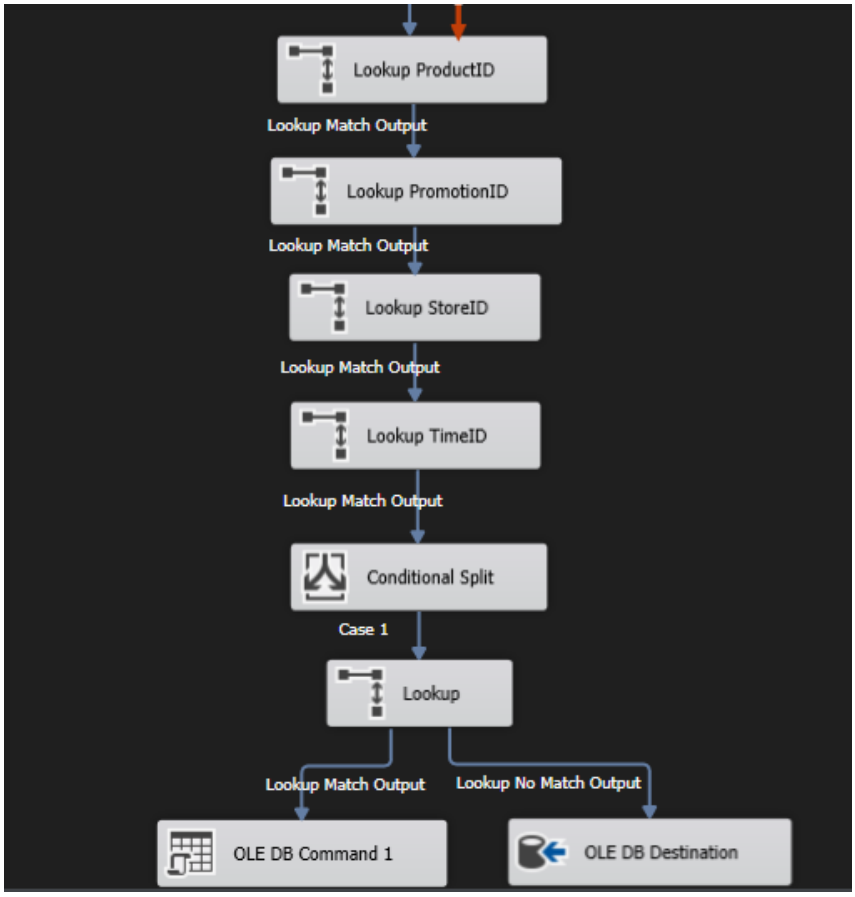


Figure 33: Path

Finish loading the data, the Fact table is successfully created.

6 Data Mining

6.1 The importance of Data Mining

In our team's Product Mix topic, Data mining is essential for understanding and optimizing the product mix as it supports us in uncovering patterns, relationships, and insights that are hidden in the huge CompanyX dataset. By applying data mining process into this project, we can:

- Identify Customer Preferences by analyzing purchasing behaviors to determine which products are frequently bought together.
- Understand the demand for each product category to avoid overstocking or understocking
- Discover the products that create the highest profit and focus on promoting them.

6.2 Clustering Techniques

6.2.1 Introduction of KMeans and DBSCAN

KMeans and DBSCAN are all powerful unsupervised machine learning method that groups data points based on their similarities. In this project, we want to try to implement both techniques to identify clusters of products based on metrics such as sales revenue, net profit margin, and return on investment(ROI) to determine which technique may give better clustering outputs for our product mix data.

6.2.2 Clustering Algorithm Implementation

A) KMeans

A.1 Data Loading and Preprocessing

In the first part of the Data Mining process,we are going to do the Data Preprocessing part. It is an important step in Data Mining as using raw data format is almost impossible to do analysis or modeling. By applying Data Preprocessing, we expect to handle missing data, ensure consistency and increase the accuracy of the model.
The majority of our data preprocessing and clustering parts are done in Jupyter Notebook. We try to connect to the database stored in SQL Server and query the crucial data that we are going to analyze later.

```
# SQL query to fetch data
query = """
SELECT
    p.Productname,
    p.ProductID,
    p.Category,
    f.Revenue,
```

```
    f.ROI,
    f.NetProfitMargin
FROM
    Fact f
JOIN
    DimProduct p ON f.ProductID = p.ProductID
"""

# Load data into a pandas DataFrame
data = pd.read_sql(query, engine)
print("Data loaded successfully.")
```

Database connection

Dropping rows with null values and duplicates is an essential step in the data preprocessing before implementing KMeans. Null values can cause disruptions in the calculations of Euclidean distance between data points and the centroids. Duplicate rows can bias the clustering process as duplicated rows give unecessary weight to some data points.

```
# Drop rows with missing values and check for duplicates
data = data.dropna().drop_duplicates()
```

We define a function named **preprocess_category** to handle the data preprocessing task. In that function, we first filter the dataset to only rows corresponding to the specified categories such as Bikes, Clothing, Components, and Accessories.

```
def preprocess_category(data, category_name):
    category_data = data[data['Category'] == category_name].copy()
    if category_data.empty:
        print(f"Warning: No data found for category '{category_name}'.
            Skipping.")
        return None, None, None
```

The next part is to extract the most relevant columns and outlier detection. The clustering task will occur mostly on the three metrics **Reveneue, ROI and NetProfitMargin** so we will extract the data from these 3 columns and focus on them. Some revenue values are large and dominant compared with other values. So, we will remove rows where **Revenue** values are above the 99th percentile.

```
# Feature selection
features = category_data[['Revenue', 'ROI', 'NetProfitMargin']].copy()

# Outlier detection: Filter rows with extreme values
revenue_threshold = features['Revenue'].quantile(0.99)  # 99th
    percentile
features_filtered = features[features['Revenue'] < revenue_threshold]
```

```
7
8 # Synchronize category_data to match filtered features
9 category_data = category_data.loc[features_filtered.index]
```

The last part in the Data Preprocessing part is to normalize the data. We want to ensure that all features contribute equally to the model by scaling them to a common range. In this case, our team uses MinMaxScaler to perform the Min-Max normalization.

```
1 # Scaling using MinMaxScaler
2 scaler = MinMaxScaler()
3 features_scaled = scaler.fit_transform(features_filtered)
4 return category_data, features_scaled, scaler
```

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Figure 34: MinMax Normalization Formula

A.2 Clustering

When implementing KMeans Clustering, we need to know the number of clusters that we are going to need. In our project, we determine the optimal number of clusters **K** by using the **Elbow Method** and **Silhouette Scores**.

We use the standardization technique here by applying **StandardScaler** on the scaled features above.

```
1 def find_optimal_clusters(features, max_clusters=15):
2     # Scale the features for better clustering
3     scaler = StandardScaler()
4     scaled_features = scaler.fit_transform(features)
```

In the next part of defining the optimal value of clusters, we need to calculate **Inertia** and **Silhouette Score**. In elbow method, Inertia measures the sum of squared distances between each point and its assigned cluster centroid. As usual, the inertia decreases as the number of K increases. The lower the inertia value is, the tighter clusters are. For Silhouette Method, Silhouette Score measures how well points are clustered. A high silhouette score illustrates well-separated and cohesive clusters.

```
1 inertia = []
2 silhouette_scores = []
3
4 # Calculate inertia and silhouette scores
5 for k in range(2, max_clusters + 1):
6     kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
7     labels = kmeans.fit_predict(scaled_features)
```

```
    inertia.append(kmeans.inertia_)
    silhouette = silhouette_score(scaled_features, labels)
    silhouette_scores.append(silhouette)
```

We determine the elbow in the elbow method graph by applying **KneeLocator** method from **kneed** package.

```
# Find knee point for optimal K
kneedle = KneeLocator(range(2, max_clusters + 1), inertia, curve="convex", direction="decreasing")
optimal_k = kneedle.knee
```

The visualization of the the elbow method and silhouette method is implemented as below:

```
# Plot Elbow Curve and Silhouette Scores
fig, ax = plt.subplots(1, 2, figsize=(12, 5))

# Elbow Curve
ax[0].plot(range(2, max_clusters + 1), inertia, 'o--', label='Inertia')
ax[0].axvline(x=optimal_k, color='r', linestyle='--', label=f'Optimal K: {optimal_k}')
ax[0].set_title('Elbow Method')
ax[0].set_xlabel('Number of Clusters')
ax[0].set_ylabel('Inertia')
ax[0].legend()

# Silhouette Scores
ax[1].plot(range(2, max_clusters + 1), silhouette_scores, 'o--', color='green', label='Silhouette Score')
ax[1].axvline(x=optimal_k, color='r', linestyle='--', label=f'Optimal K: {optimal_k}')
ax[1].set_title('Silhouette Scores for Clusters')
ax[1].set_xlabel('Number of Clusters')
ax[1].set_ylabel('Silhouette Score')
ax[1].legend()

plt.show()

return optimal_k
```

In the final step, we will perform the clustering by combining all the above functions. The whole process can be summarized in these main steps: preprocessing the data, clustering using KMeans, visualizing the results and summarizing cluster characteristics.

```
# Perform clustering and visualize
def perform_clustering(data, category_name, n_clusters=None):
category_data, features_scaled, scaler = preprocess_category(data, category_name)
```

```
if category_data is None or features_scaled is None:
    return None

# Determine number of clusters if not provided
if n_clusters is None:
    n_clusters = find_optimal_clusters(features_scaled)

# K-Means clustering
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
category_data['Cluster'] = kmeans.fit_predict(features_scaled)

# Retrieve centroids
centroids = scaler.inverse_transform(kmeans.cluster_centers_)

# Visualization
plt.figure(figsize=(10, 6))
sns.scatterplot(data=category_data, x='Revenue', y='ROI', hue='Cluster',
    palette='viridis', s=100)

# Plot centroids
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=300, marker='X'
    , label='Centroids')
plt.title(f'Clustering Results for {category_name} Category')
plt.xlabel('Revenue')
plt.ylabel('Return on Investment')
plt.legend(title='Cluster')
plt.show()

# Display cluster statistics
cluster_summary = category_data.groupby('Cluster')[['Revenue', 'ROI', '
    NetProfitMargin']].mean()
print(f"Cluster Summary for {category_name}:")
print(cluster_summary)

return category_data
```

The data preprocessing prepared for the clustering process is illustrated as below:

```
# Example: Preprocess the categories and perform clustering
bikes_data, bikes_features, bikes_scaler = preprocess_category(data, '
    Bikes')
clothing_data, clothing_features, clothing_scaler = preprocess_category(
    data, 'Clothing')
components_data, components_features, components_scaler =
    preprocess_category(data, 'Components')
accessories_data, accessories_features, accessories_scaler =
```

```
    preprocess_category(data, 'Accessories')


print("Bike data: ")
print(bikes_data.describe())
print("Clothings data: ")
print(clothing_data.describe())
print("Components data: ")
print(components_data.describe())
print("Accessories data: ")
print(accessories_data.describe())
```

```
Bike data:
         ProductID       Revenue          ROI  NetProfitMargin
count  1034.000000   1034.000000  1034.000000      1034.000000
mean    851.918762   2915.845625    -0.128704        -7.199583
std      96.058233   2734.493462     0.581104        23.397620
min     749.000000    112.998000    -4.222552      -172.764031
25%     776.000000    939.588000    -0.279099        -6.066663
50%     793.000000   1879.176000    -0.096976        -3.600003
75%     965.000000   4079.988000     0.091627         6.266665
max     999.000000  13215.657400     2.587926         9.078667
Clothings data:
         ProductID     Revenue         ROI  NetProfitMargin
count  210.000000  210.000000  210.000000       210.000000
mean   819.400000  186.937563    0.852198         9.293679
std     69.946131  166.132835    2.362474        31.924466
min    709.000000    5.186500   -9.595960       -45.600825
25%    715.000000   57.680800   -0.437375       -28.333333
50%    859.000000  144.089000    0.519547        31.249931
75%    865.000000  269.964000    2.081331        35.517241
max    884.000000  769.890000   12.626230        40.415789
```

Figure 35: Data Preprocessing Overview

Components data:

	ProductID	Revenue	ROI	NetProfitMargin
count	717.000000	717.000000	717.000000	717.000000
mean	862.596932	456.151169	0.292948	16.953930
std	77.937273	579.923855	0.364145	11.708841
min	717.000000	12.144000	-0.202193	-14.584067
25%	810.000000	97.176000	0.000000	8.733326
50%	886.000000	209.256000	0.197945	25.999835
75%	937.000000	599.496000	0.413352	25.999945
max	996.000000	2944.582000	2.702685	26.000055

Accessories data:

	ProductID	Revenue	ROI	NetProfitMargin
count	92.000000	92.000000	92.000000	92.000000
mean	787.576087	110.653427	1.705646	34.954760
std	80.035464	123.201181	2.849675	4.723436
min	707.000000	1.374000	0.000000	16.888635
25%	708.000000	35.334000	0.436245	35.173012
50%	777.000000	73.482000	0.839268	37.665331
75%	873.750000	127.467000	1.495808	37.666476
max	880.000000	648.000000	16.785502	37.666667

Figure 36: Data Preprocessing Overview

We are going to show the results of the clusters for 4 different categories: Bikes, Components, Clothing and Accessories consecutively. We both use the visualization of elbow method and silhouette method to determine the number of clusters. Based on our observation, the result of the elbow method is quite accurate and good for analysis process. The silhouette method result is quite close to that of elbow method but when applying it, the clusters are not well separated. The reason is possibly due to the different criteria. Silhouette method usually focus on cluster cohesion and may lead to fewer clusters.

BIKES CATEGORY

```
print(f"\n--- Clustering for Bikes Category ---")
# Apply clustering with a predefined number of clusters
bikes_clustered_data = perform_clustering(bikes_data,'Bikes')
```

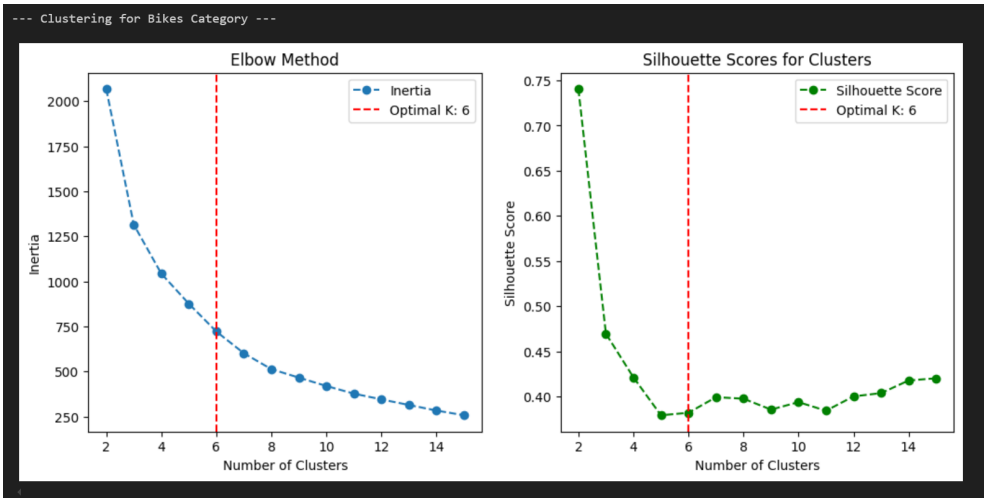


Figure 37: Bikes Category 1

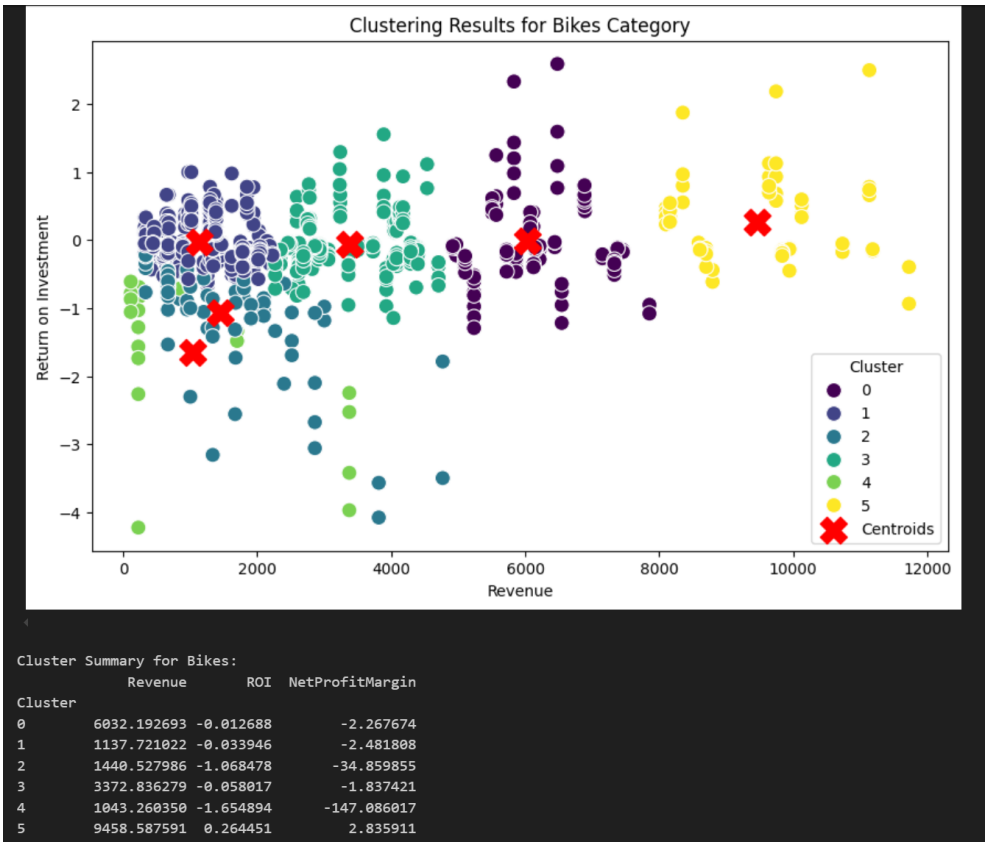


Figure 38: Bikes Category 2

COMPONENTS CATEGORY

```
print(f"\n--- Clustering for Components Category ---")
# Apply clustering with a predefined number of clusters
components_clustered_data = perform_clustering(components_data,'
    Components')
```

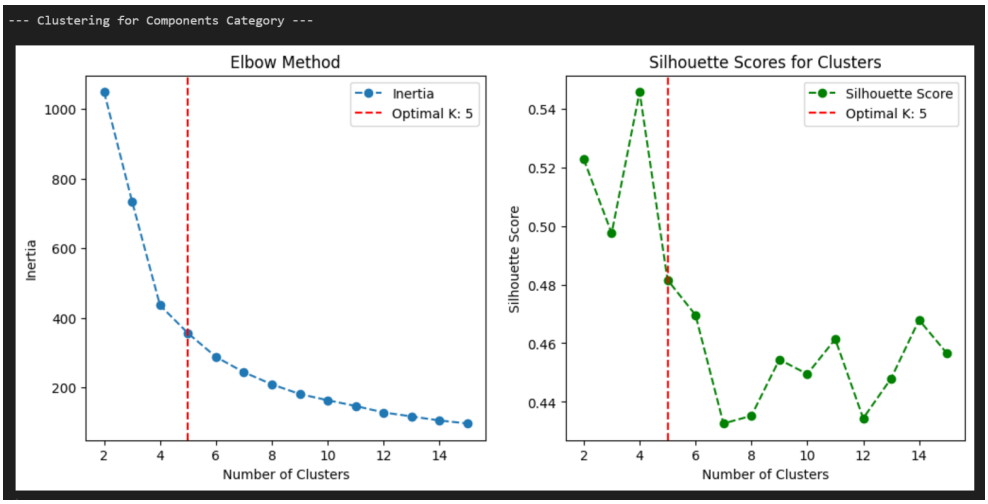


Figure 39: Components Category 1

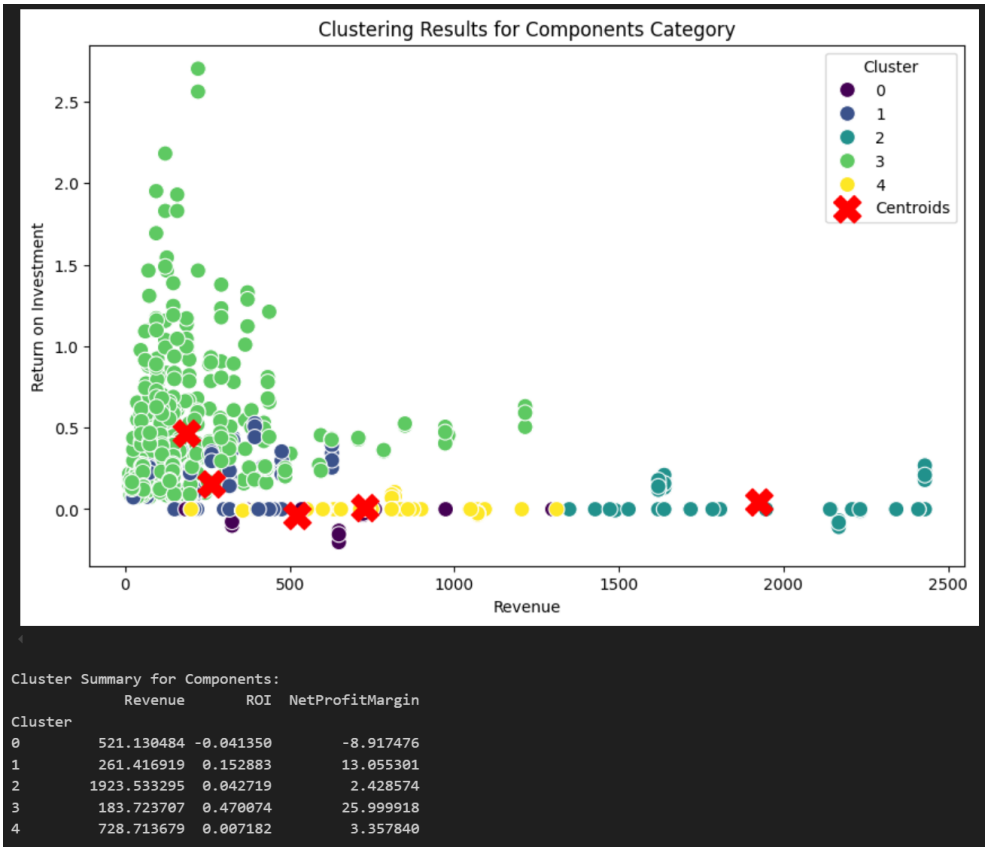


Figure 40: Components Category 2

CLOTHING CATEGORY

```
print(f"\n--- Clustering for Clothing Category ---")
# Apply clustering with a predefined number of clusters
clothing_clustered_data = perform_clustering(clothing_data,'Clothing')
```

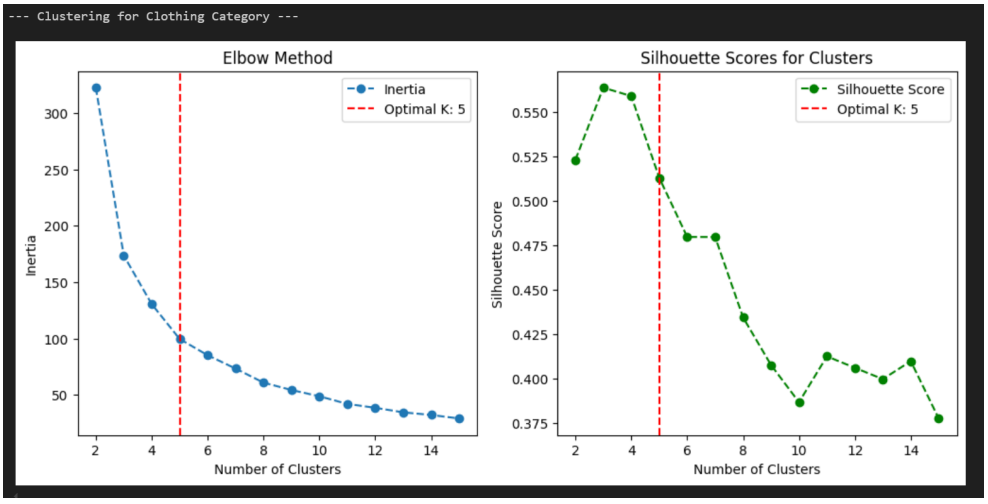


Figure 41: Clothing Category 1

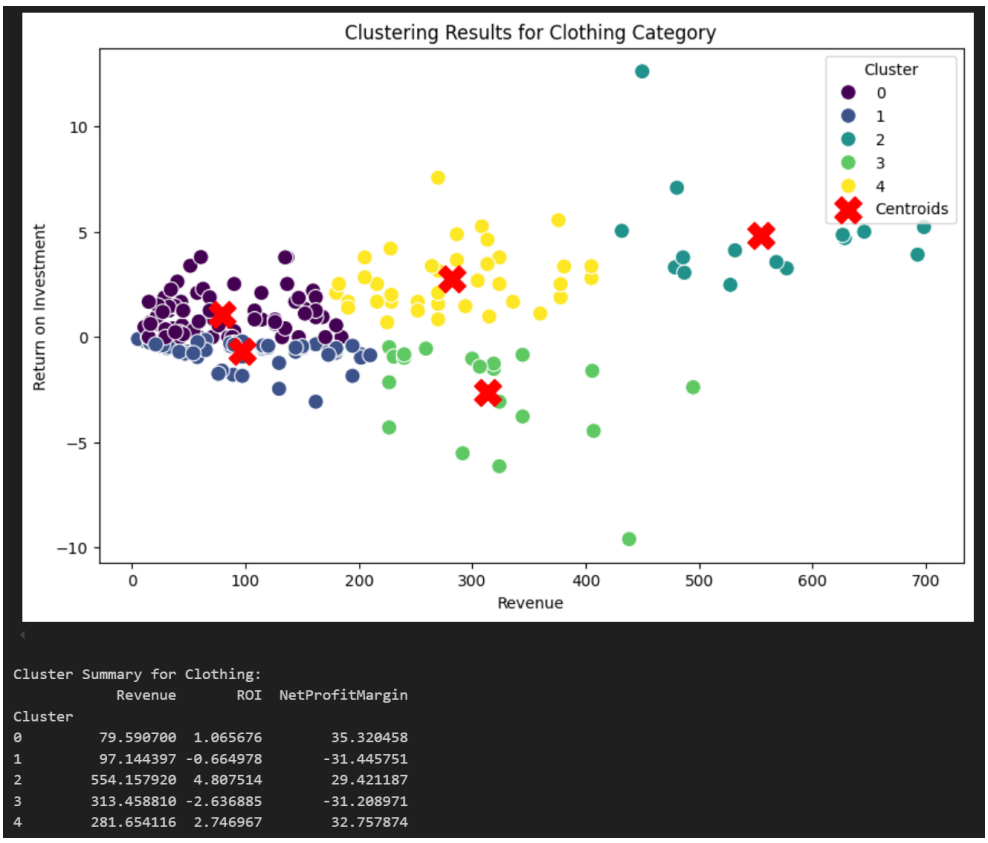


Figure 42: Clothing Category 2

ACCESSORIES CATEGORY

```
print(f"\n--- Clustering for Accessories Category ---")
# Apply clustering with a predefined number of clusters
accessories_clustered_data = perform_clustering(accessories_data,'
    Accessories')
```

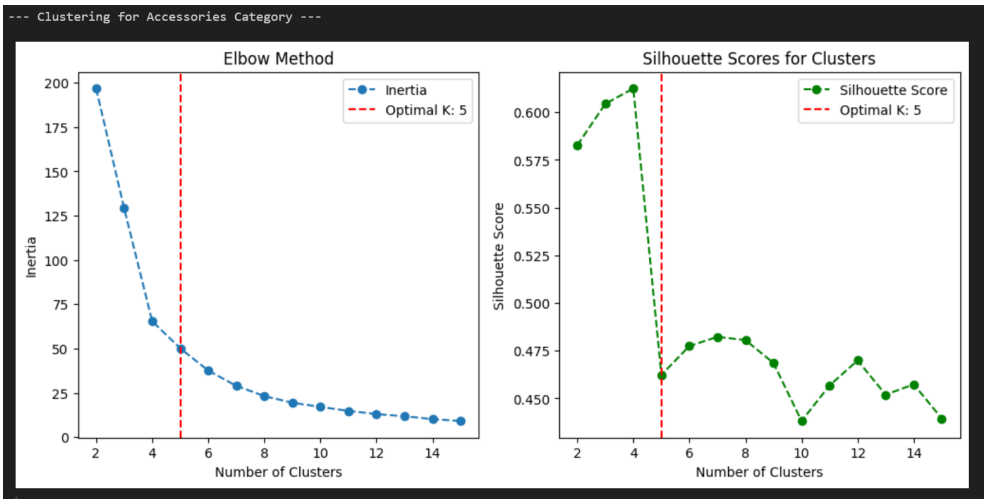


Figure 43: Accessories Category 1

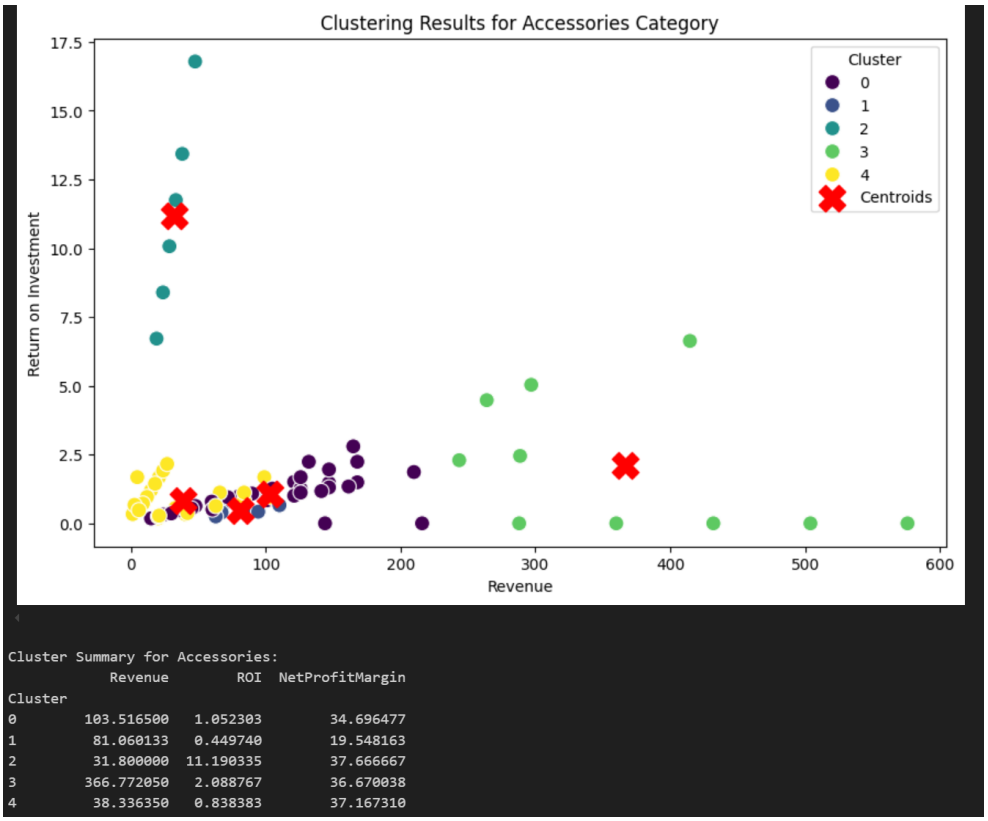


Figure 44: Accessories Category 2

B) DBSCAN

B.1 Data Preprocessing

The data preprocessing section in DBSCAN is implemented almost similarly to that in KMeans. We focus on the 3 most important metrics: Revenue, ROI and NetProfitMargin and standardize the data using the StandardScaler().

```
# Define function to preprocess data for each category
def preprocess_category(data, category_name):
    # Filter data for the given category
    category_data = data[data['Category'] == category_name]

    # Select relevant features for clustering
    features = category_data[['Revenue', 'ROI', 'NetProfitMargin']]

    # Standardize the features
    scaler = StandardScaler()
    features_scaled = scaler.fit_transform(features)

    return category_data, features_scaled
```

$$z = \frac{x - \mu}{\sigma}$$

$$\mu = \text{Mean}$$

$$\sigma = \text{Standard Devia}$$

Figure 45: StandardScaler Formula

B.2 Clustering

In DBSCAN, we need to know the optimal values of epsilon and minimum samples before doing the clustering process. In our project, we try to define these parameters by applying k-distance graph and silhouette score evaluation.Below is how we implement the process of optimizing these variables.

```python
def find_optimal_dbscan_params(features, k_neighbors=5, min_samples_range=range(5, 30, 5), eps_search_margin=0.5):
    # """
    # Combines the process of finding optimal eps and min_samples for DBSCAN clustering.

    # Parameters:
    # - features (numpy.ndarray): The standardized features for clustering.
    # - k_neighbors (int): The number of neighbors to use in the k-distance graph.
    # - min_samples_range (range): The range of min_samples values to evaluate.
    # - eps_search_margin (float): The margin around the initial eps to refine the search.

    # Returns:
    # - dict: A dictionary containing optimal eps, min_samples, and the corresponding silhouette score.
    # """
    # Step 1: Find the optimal eps using the k-distance graph
    neighbors = NearestNeighbors(n_neighbors=k_neighbors)
    neighbors_fit = neighbors.fit(features)
    distances, indices = neighbors_fit.kneighbors(features)

    # Sort distances for the k-th nearest neighbor
    distances = np.sort(distances[:, -1], axis=0)

    # Find the elbow point in the k-distance graph
    kneedle = KneeLocator(range(len(distances)), distances, curve="convex", direction="increasing")
    optimal_knee_point = kneedle.knee
    optimal_eps = distances[optimal_knee_point]

    # Plot the k-distance graph with the elbow point
    plt.figure(figsize=(8, 6))
    plt.plot(distances, label='k-distances')
    plt.axvline(x=optimal_knee_point, color='r', linestyle='--', label=f'Elbow Point: {optimal_eps:.4f}')
    plt.title("K-distance Graph with Elbow Point")
    plt.xlabel("Data Points")
    plt.ylabel("Distance to k-th Nearest Neighbor")
    plt.legend()
    plt.show()

```

```
    print(f"Optimal epsilon from K-distance graph (Elbow Method): {optimal_eps:.4f}")

    # Step 2: Refine the search for optimal eps using silhouette scores
    eps_range = np.arange(optimal_eps - eps_search_margin, optimal_eps + eps_search_margin, 0.05)
    silhouette_scores_eps = []

    for eps in eps_range:
        dbscan = DBSCAN(eps=eps, min_samples=min_samples_range[0])  # Use the smallest min_samples initially
        labels = dbscan.fit_predict(features)

        if len(set(labels)) > 1:  # Ensure at least 2 clusters
            score = silhouette_score(features, labels)
        else:
            score = -1

        silhouette_scores_eps.append(score)

    optimal_eps_refined = eps_range[np.argmax(silhouette_scores_eps)]

    # Plot silhouette scores for eps
    plt.figure(figsize=(8, 6))
    plt.plot(eps_range, silhouette_scores_eps, 'o--', color='green', label='Silhouette Score')
    plt.axvline(x=optimal_eps_refined, color='r', linestyle='--', label=f'Optimal eps: {optimal_eps_refined:.2f}')
    plt.title('Silhouette Scores for Different eps Values')
    plt.xlabel('eps')
    plt.ylabel('Silhouette Score')
    plt.legend()
    plt.show()

    print(f"Optimal epsilon from Silhouette Score Method: {optimal_eps_refined:.4f}")

    # Step 3: Find the optimal min_samples using the refined eps
    silhouette_scores_min_samples = []

    for min_samples in min_samples_range:
        dbscan = DBSCAN(eps=optimal_eps_refined, min_samples=min_samples)
        labels = dbscan.fit_predict(features)

```

```
        if len(set(labels)) > 1:  # Ensure at least 2 clusters
            score = silhouette_score(features, labels)
        else:
            score = -1

        silhouette_scores_min_samples.append(score)

    optimal_min_samples = min_samples_range[np.argmax(
        silhouette_scores_min_samples)]

    # Plot silhouette scores for min_samples
    plt.figure(figsize=(8, 6))
    plt.plot(min_samples_range, silhouette_scores_min_samples, 'o--',
        color='blue', label='Silhouette Score')
    plt.axvline(x=optimal_min_samples, color='r', linestyle='--', label=
        f'Optimal min_samples: {optimal_min_samples}')
    plt.title('Silhouette Scores for Different min_samples Values')
    plt.xlabel('min_samples')
    plt.ylabel('Silhouette Score')
    plt.legend()
    plt.show()

    print(f"Optimal min_samples from Silhouette Score Method: {
        optimal_min_samples}")

    # Return the optimal parameters and their silhouette score
    return {
        "optimal_eps": optimal_eps_refined,
        "optimal_min_samples": optimal_min_samples,
        "silhouette_score": max(silhouette_scores_min_samples)
    }
```

This process can be divided into 3 main steps.

Step 1: Find the optimal epsilon using the k-distance graph

- Use NearestNeighbors to compute distances to the k-nearest neighbors for all points in the dataset
- Create the distance graph by sorting the distances of the k-neighbors in ascending order.
- Identify the elbow point which is the finding epsilon by using KneeLocator.
- Plot the k-distance graph with the identified elbow point.

Step 2: Refine the epsilon value using Silhouette scores

- Define the range of the potential epsilon values around the previously obtained epsilon value in step 1.
- For each epsilon value in the range, apply the DBSCAN algorithm with the smallest min sample value in the previously defined min sampel range. Calculate the silhouette score if at least 2 clusters are formed.
- Identify the epsilon value that has the greatest silhouette score
- Plot the silhouette scores for each epsilon value.

Step 3: Optimize the min samples using the optimal epsilon value

- For each min sample in the min sample ranges, apply the DBSCAN algorithm using the optimal epsilon value and current min sample. Compute the silhouette score if there are at least 2 clusters formed.
- Identify the min sample that has the greatest silhouette score.
- Plot the silhouette score for each min sample.

With the optimal epsilon and min samples achieved from the above part, we are going to perform clustering on our dataset and visualize the results.

```
# Function to perform DBSCAN clustering and visualize results
def perform_dbscan_clustering(data, category_name):
    # Preprocess the data
    category_data, features_scaled = preprocess_category(data,
        category_name)

    # Dynamically find the optimal eps and min_samples
    # Define parameters
    k_neighbors = 15
    min_samples_range = range(5, 20, 5)
    eps_search_margin = 0.4
    # Call the function
    optimal_params = find_optimal_dbscan_params(features_scaled,
        k_neighbors, min_samples_range, eps_search_margin)

    # Extract optimal values
    eps = optimal_params["optimal_eps"]
    min_samples = optimal_params["optimal_min_samples"]


    # Perform DBSCAN clustering
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)
```

```
    category_data['Cluster'] = dbscan.fit_predict(features_scaled)

    # Visualize clustering results
    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=category_data, x='Revenue', y='ROI',
                    hue='Cluster', palette='viridis', s=100, marker='o')

    plt.title(f'DBSCAN Clustering Results for {category_name} Category')
    plt.xlabel('Total Revenue')
    plt.ylabel('Return on Investment')
    plt.legend(title='Cluster')
    plt.show()

    # Analyze cluster statistics
    cluster_summary = category_data.groupby('Cluster')[['Revenue', 'ROI'
        , 'NetProfitMargin']].mean()
    print(f"Cluster Summary for {category_name} Category:")
    print(cluster_summary)

    return category_data
```

This process implements all the above created functions followed these steps

Step 1: Preprocess the Data

Step 2: Define Optimal Parameters for DBSCAN

Step 3: Perform DBSCAN Clustering

Step 4: Visualize Clustering Results

Step 5: Analyze Cluster Statistics based on the 3 metrics: Revenue, ROI and NetProfitMargin

Below are the clustering results for each category using DBSCAN.

BIKES CATEGORY

```
1 print(f"\n--- Clustering for Bikes Category ---")
2 bikes_clustered_data = perform_dbscan_clustering(bikes_data, 'Bikes')
```

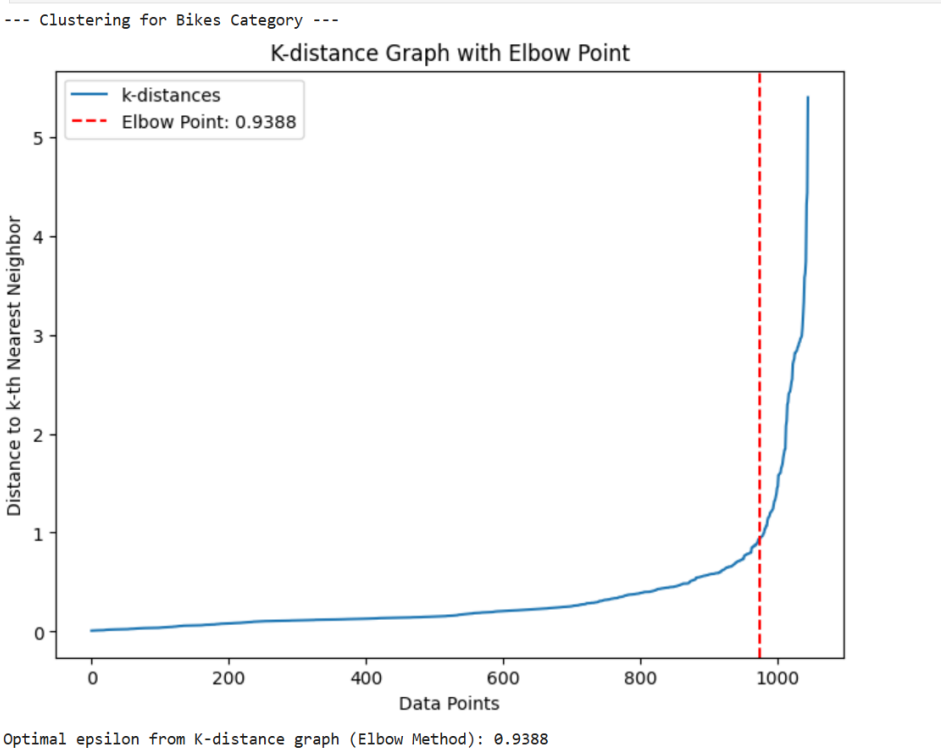


Figure 46: Bikes Category 1

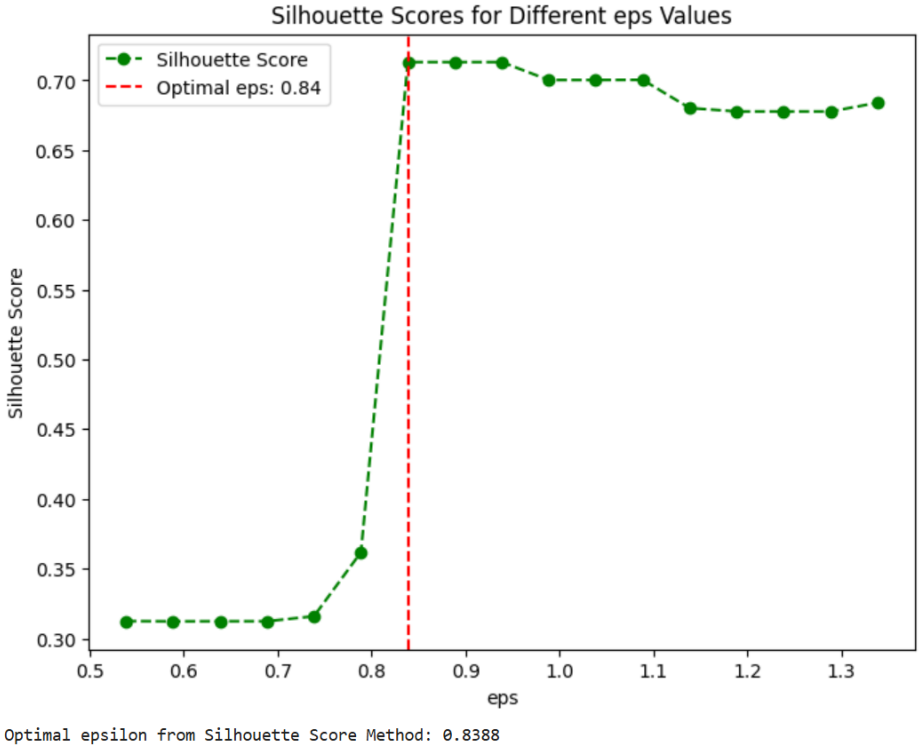
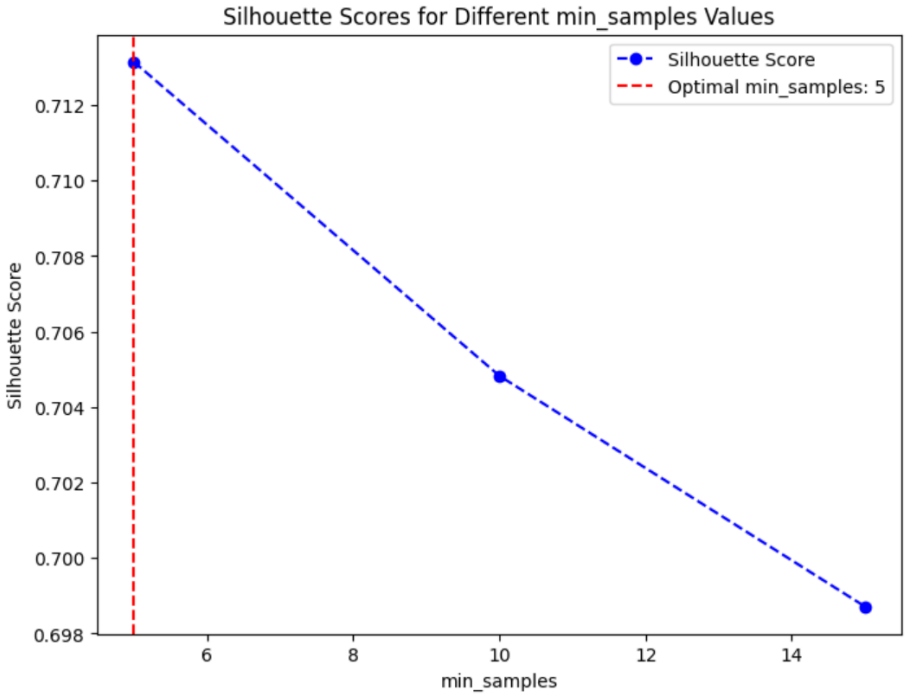
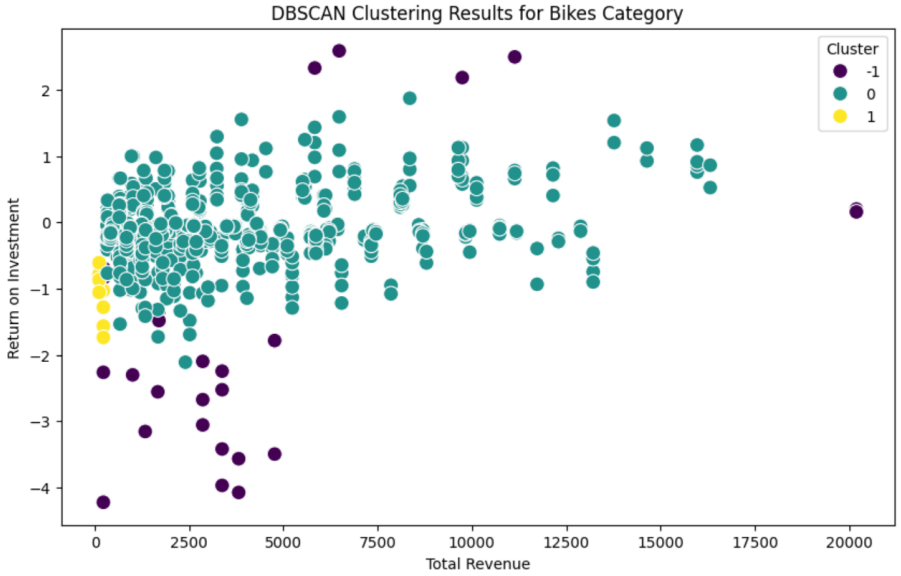


Figure 47: Bikes Category 2




```
Optimal min_samples from Silhouette Score Method: 5
```

Figure 48: Bikes Category 3

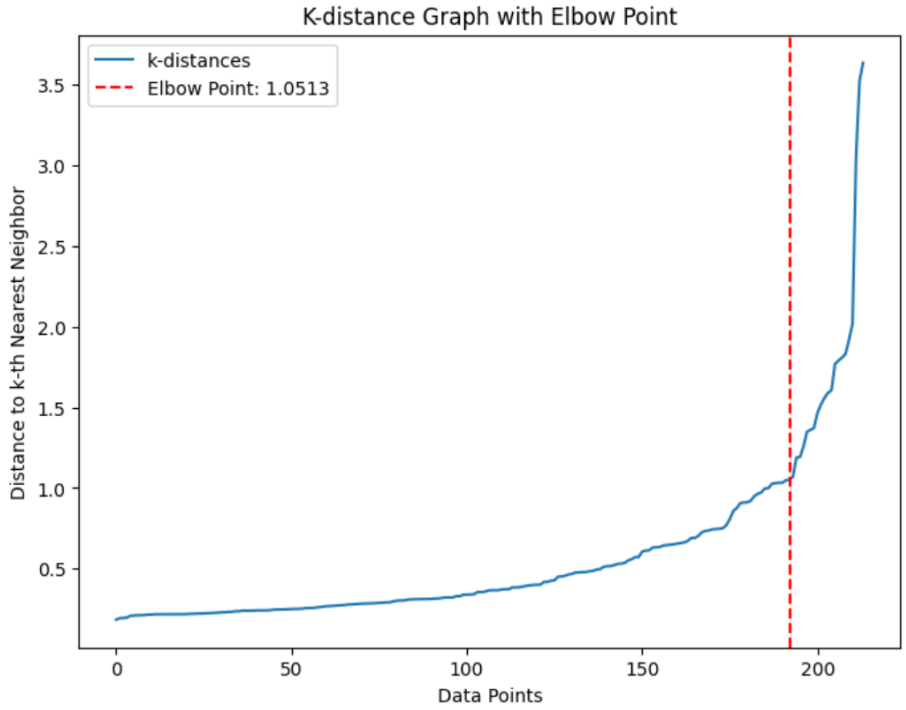



```
Cluster Summary for Bikes Category:
              Revenue       ROI  NetProfitMargin
Cluster
-1       4387.482036 -1.534734       -72.383164
 0       3053.987233 -0.070247        -3.928267
 1        169.497000 -1.115365      -172.764031
```

Figure 49: Bikes Category 4

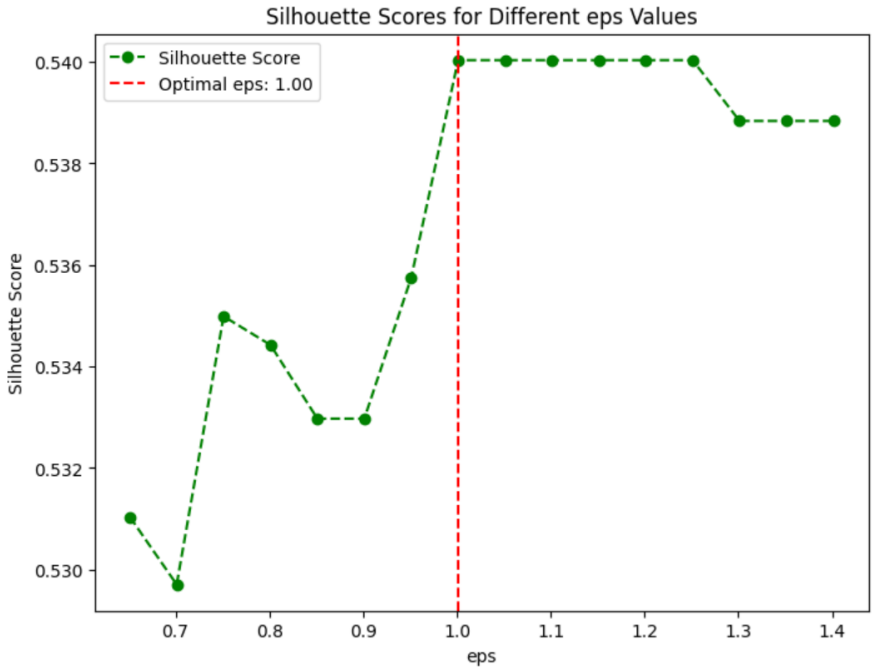
CLOTHING CATEGORY

```
print(f"\n--- Clustering for Clothing Category ---")
clothing_clustered_data = perform_dbscan_clustering(clothing_data, 'Clothing')
```



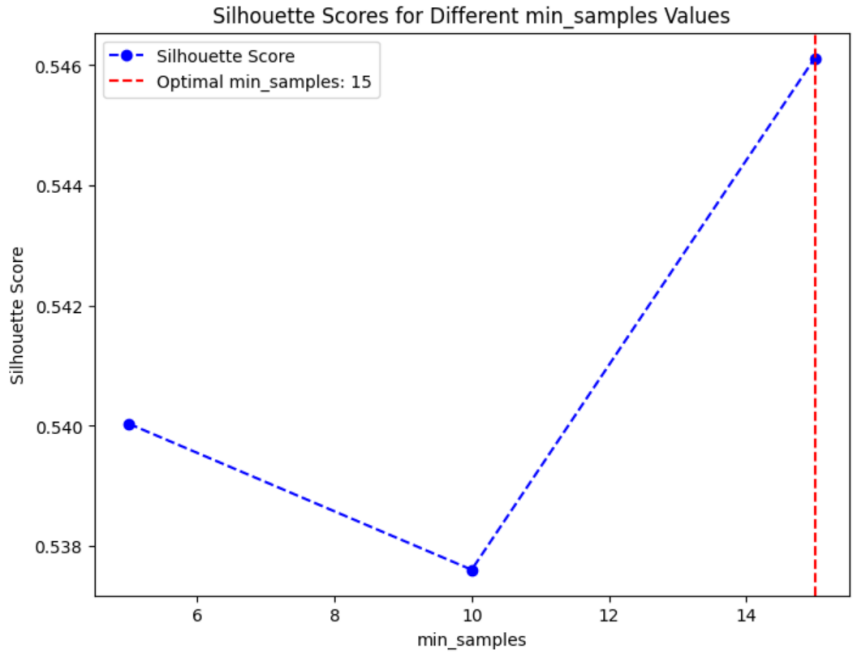

```
Optimal epsilon from K-distance graph (Elbow Method): 1.0513
```

Figure 50: Clothing Category 1



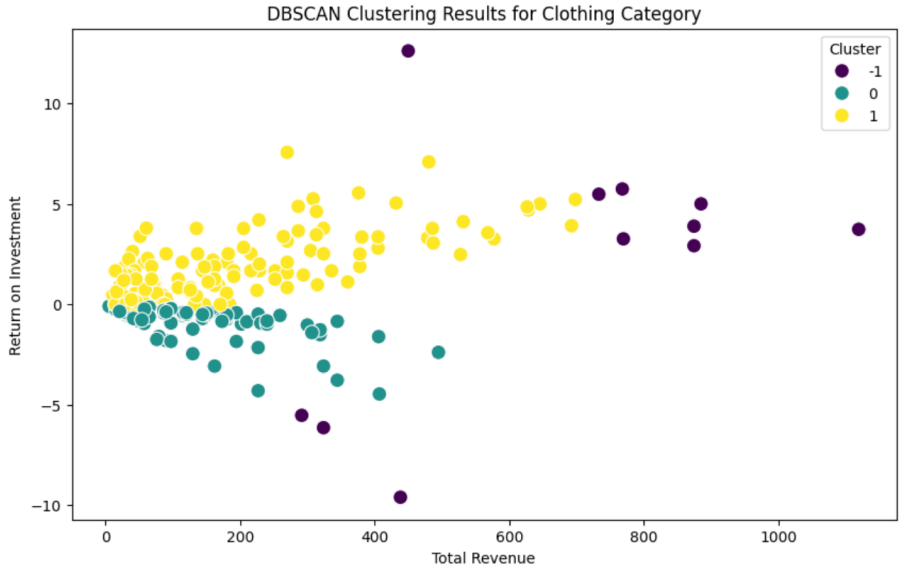

```
Optimal epsilon from Silhouette Score Method: 1.0013
```

Figure 51: Clothing Category 2




```
Optimal min_samples from Silhouette Score Method: 15
```

Figure 52: Clothing Category 3

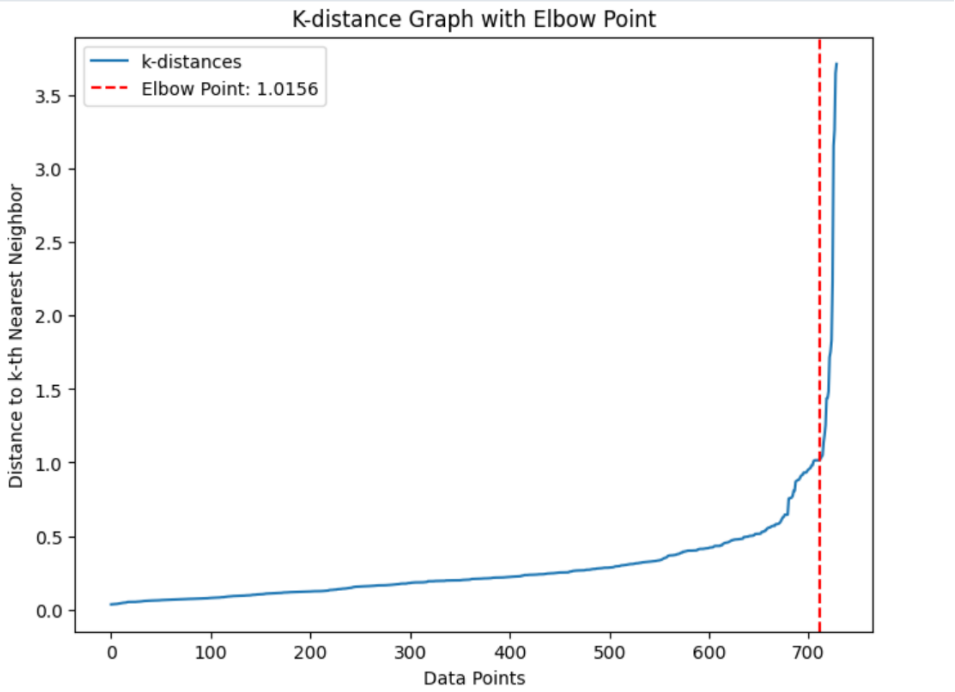



```
Cluster Summary for Clothing Category:
            Revenue       ROI  NetProfitMargin
Cluster
-1       684.586664  1.951427        13.220373
 0       144.036984 -0.930568       -31.448072
 1       193.185730  1.919661        33.888988
```

Figure 53: Clothing Category 4

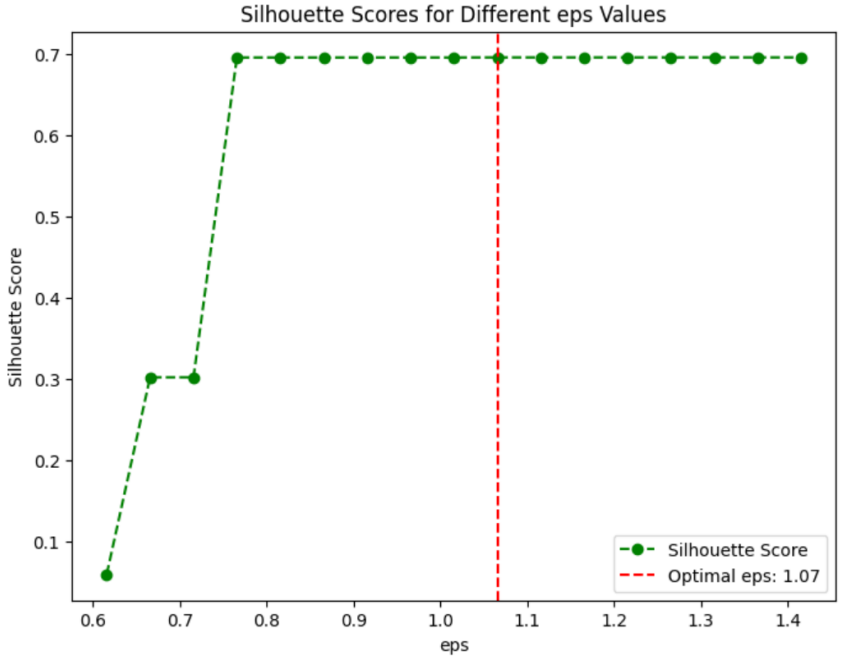
COMPONENTS CATEGORY

```
print(f"\n--- Clustering for Components Category ---")
components_clustered_data = perform_dbscan_clustering(components_data, '
    Components')
```



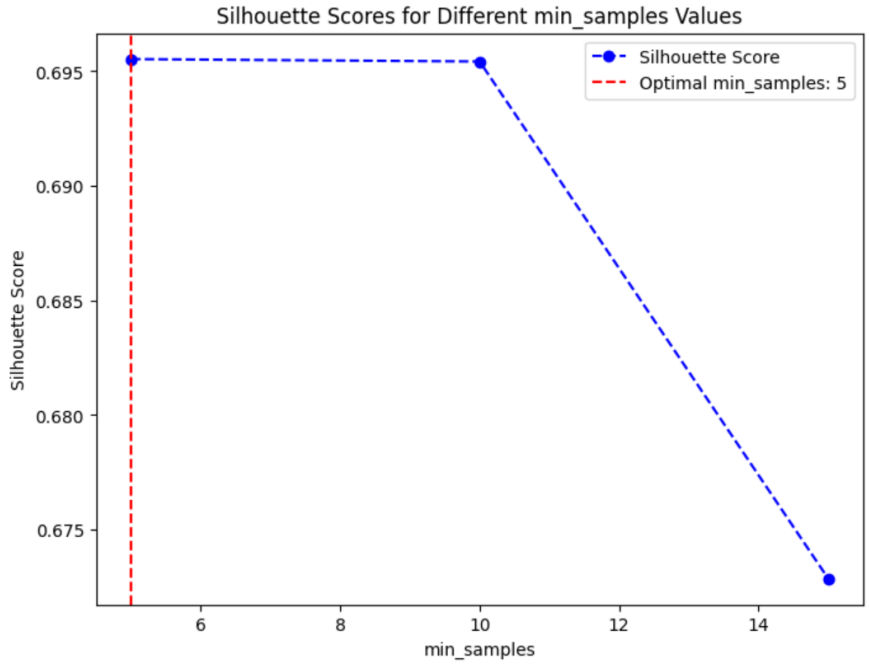

```
Optimal epsilon from K-distance graph (Elbow Method): 1.0156
```

Figure 54: Components Category 1



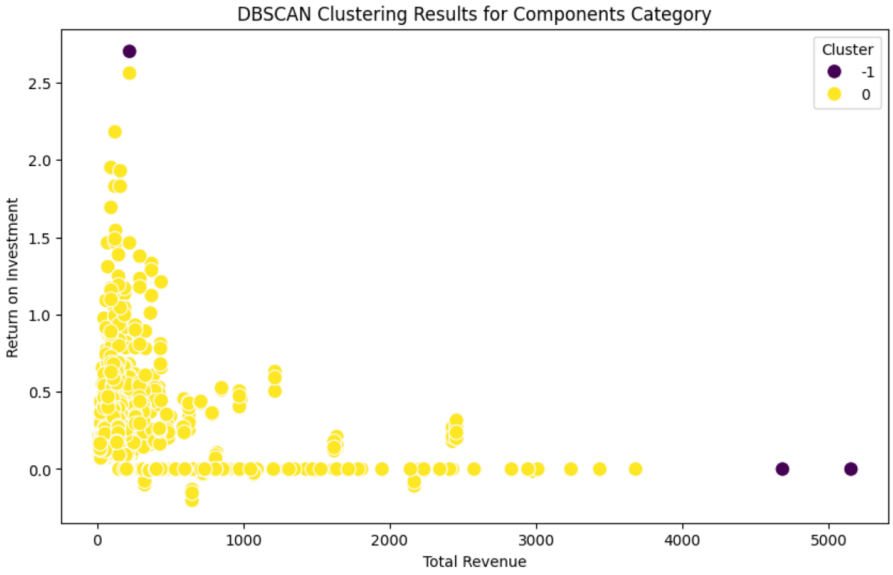

```
Optimal epsilon from Silhouette Score Method: 1.0656
```

Figure 55: Components Category 2




```
Optimal min_samples from Silhouette Score Method: 5
```

Figure 56: Components Category 3

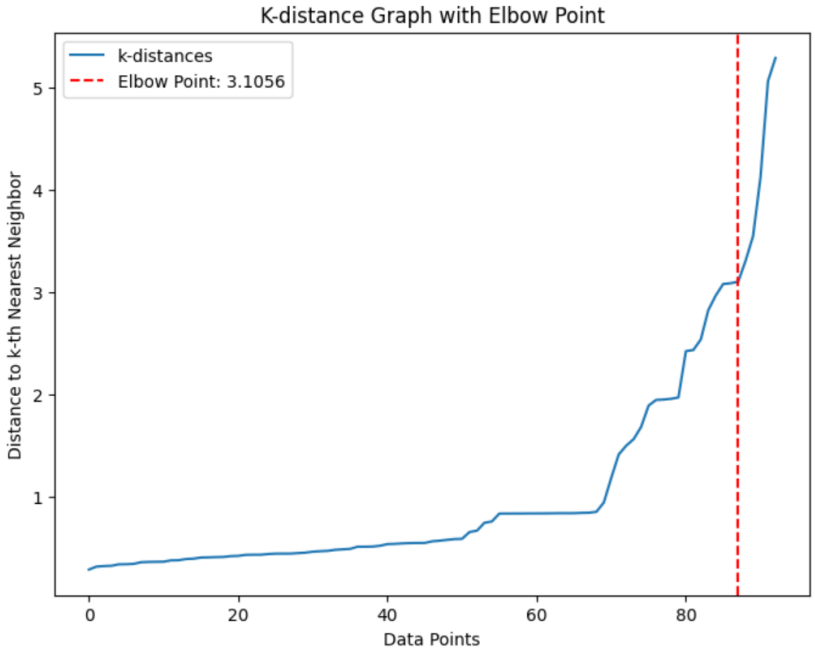



```
Cluster Summary for Components Category:
              Revenue       ROI  NetProfitMargin
Cluster
-1       3353.005233  0.900895         4.786625
 0        493.207910  0.285485        16.714759
```

Figure 57: Components Category 4

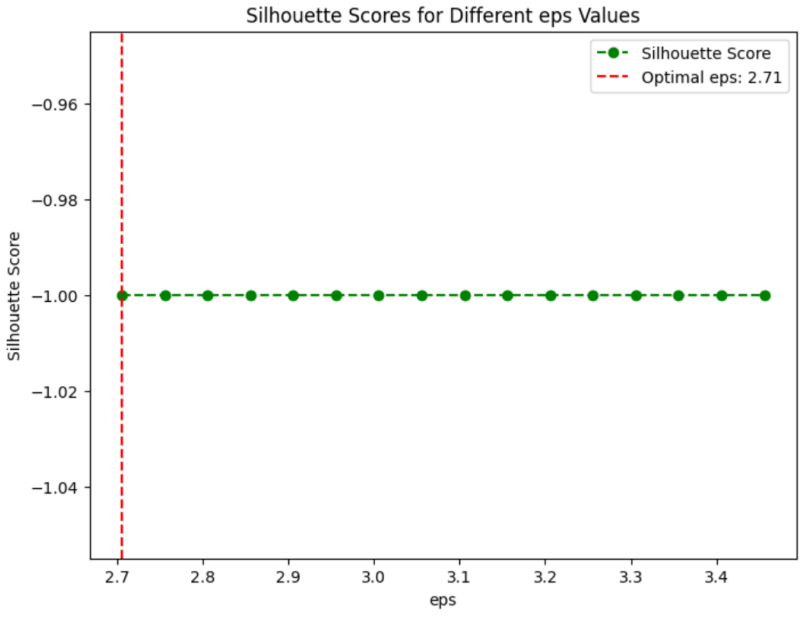
ACCESSORIES CATEGORY

```
1 print(f"\n--- Clustering for Accessories Category ---")
2 accessories_clustered_data=perform_dbscan_clustering(accessories_data,'
    Accessories')
```




```
Optimal epsilon from K-distance graph (Elbow Method): 3.1056
```

Figure 58: Accessories Category 1




```
Optimal epsilon from Silhouette Score Method: 2.7056
```

Figure 59: Accessories Category 2

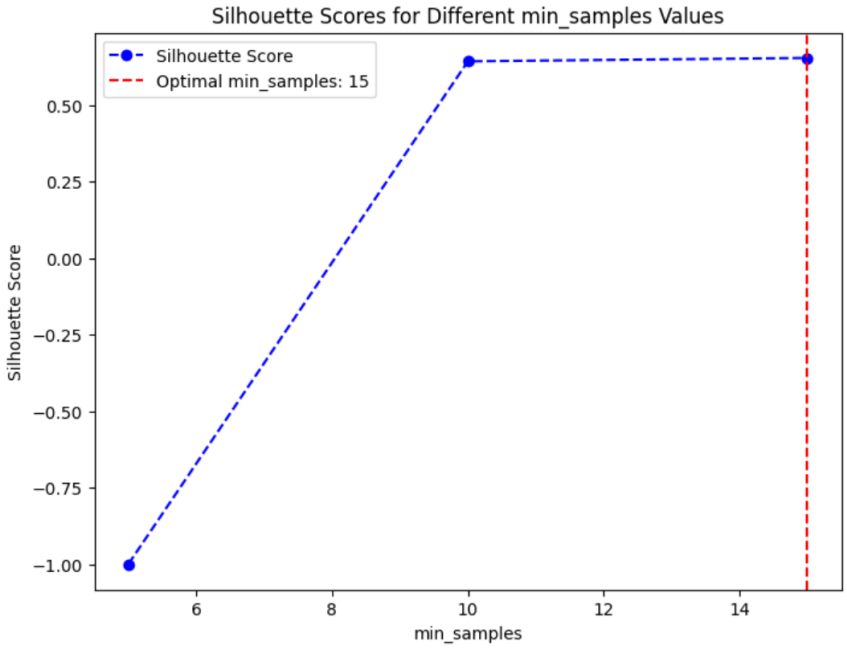


Figure 60: Accessories Category 3

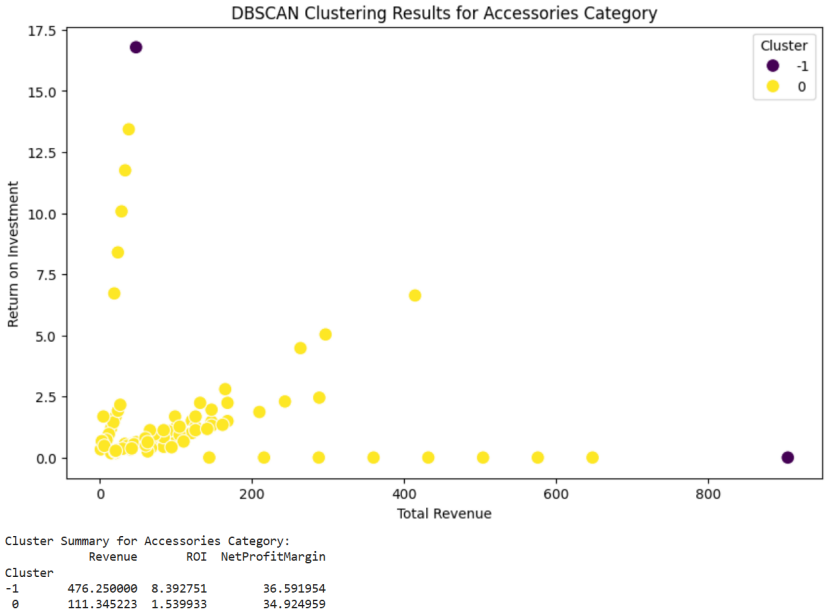


Figure 61: Accessories Category 4

6.2.3 Comparison between KMeans and DBSCAN

In this project, our team implemented both KMeans and DBSCAN clustering algorithms to analyze the product mix data. After reviewing clustering results from both methods, we observed that KMeans outperformed DBSCAN in this context. The clusters generated by KMeans are more clearly defined and better separatedm making it easier for interpretation. With this experience, we realize the importance of evaaluating multiple clustering techniques to identify the most effective approach for a certain dataset and analysis objective.

7 Power BI presentation

The dashboard provides a comprehensive analysis of key financial metrics for CompanyX that are Revenue, Return on Investment (ROI), and Net Profit Margin for each product category in each country in Europe.

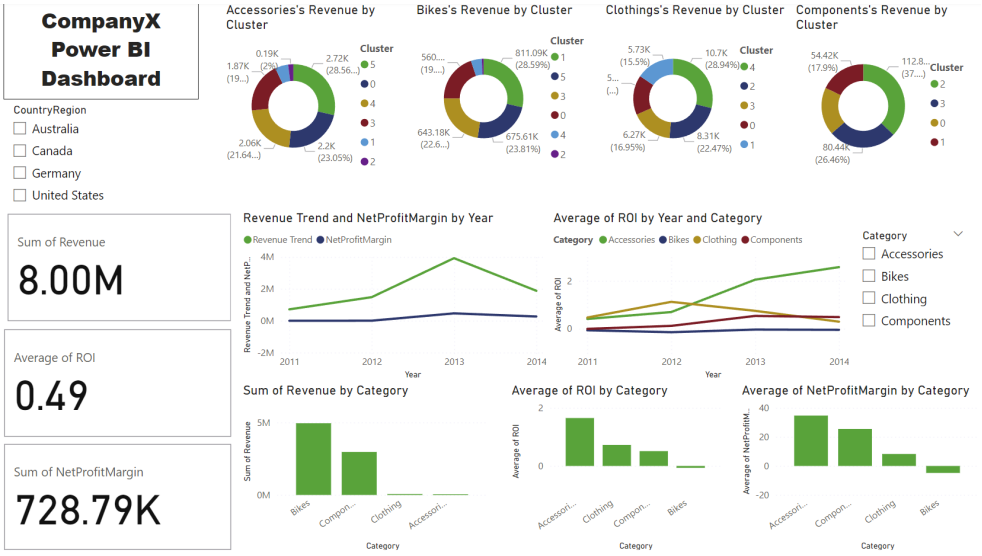


Figure 62: Power BI dashboard

Key Performance Indicators (KPIs)

- **Sum of Revenue**: Displays the total revenue across all regions ($8.00M).
- **Average ROI**: Highlights the average return on investment (0.49).
- **Sum of Net Profit Margin**: Summarizes the total net profit margin ($728.79K).

Cluster-Based Analysis

- Pie charts provide a breakdown of revenue by clusters, revealing proportional contributions of each cluster to total revenue.

Yearly Trends

- A line chart compares revenue and net profit margin trends over the years.
- Another line chart shows ROI progression across categories like Accessories, Bikes, Clothing, and Components.

Category-Based Analysis

- Bar charts display revenue, ROI, and net profit margin comparisons for different product categories.

Region-Specific Insights

- A slicer allows filtering the dashboard by region (Australia, Canada, Germany, United States) to provide region-specific insights.

This dashboard ensures clarity and depth in financial analysis but can be improved by consolidating redundant visuals, refining layout, and applying consistent color schemes to enhance readability and interactivity.

8 Member's Contribution

Our project was a team effort where each member played a vital role in contributing to the overall success. Let me walk you through the specific responsibilities and tasks assigned to each member

Member	Task
Phan Khanh Nghi	Contribute to star schema design, ETL, Data mining and Power Bi
Huynh Dang Trinh	Contribute to star schema design, Data Mining, Power BI
Huynh Minh Tuong	Contribute star schema design, ETL, Report writing
Nguyen Hoang Nguyen	Contribute to star schema design, ETL
Trinh The Hao	Contribute to star schema design, Data Mining

Figure 63: Task Division

- **Phan Khanh Nghi:**
  - Contributed to the *star schema design*, ensuring the database structure was optimized for analysis.
  - Worked on the *ETL (Extract, Transform, Load) process*, managing the movement of data from raw sources into the warehouse.
  - Participated in *data mining*, extracting meaningful insights from the data.
  - Created and visualized insights using *Power BI dashboards*.
- **Huynh Dang Trinh:**
  - Contributed to the *star schema design*, collaborating closely with the team to finalize the data model.
  - Engaged in *data mining*, focusing on analyzing and identifying trends in the data.
  - Designed visuals and reports using *Power BI*, creating charts and graphs to communicate findings effectively.
- **Huynh Minh Tuong:**

  - Played a key role in the *star schema design*, ensuring the database structure met analytical requirements.
  - Managed the *ETL process*, transforming raw data into a structured format for analysis.
  - Contributed to *report writing*, compiling and documenting the project's methodologies, findings, and conclusions.

- **Nguyen Hoang Nguyen:**
  - Contributed to the *star schema design*, ensuring data was organized logically and efficiently.
  - Worked on the *ETL process*, ensuring smooth data transformation and integration.

- **Trinh The Hao:**
  - Contributed to the *star schema design*, aligning it with the project goals.
  - Engaged in *data mining*, analyzing patterns and trends to support the project's key insights.

Each member's role contributed to a different stage of the research process, ensuring a comprehensive and well-structured investigation into the ProductMix with clustering in Datawarehouse.

References

Eltibi, M., & Ashour, W. (2011). Initializing k-means clustering algorithm using statistical information. *International Journal of Computer Applications (IJCA)*, *29*, 51–55. https://doi.org/10.5120/3573-4930

Geereddy, N. (2013). Strategic analysis of starbucks corporation [Harvard University Case Study].

Gitman, L. J. (2012). *Principles of managerial finance* (13th). Addison-Wesley Publishing Company.

Hayes, A. (2024). Revenue definition, formula, calculation, and examples [Investopedia, https://www.investopedia.com/terms/r/revenue.asp].

Jeffery, M. (2008). *Return on investment analysis* [Chapter from the "mjeffery Handbook ROI Chapter v3.0"]. Northwestern University.

Likas, A., Vlassis, N., & Verbeek, J. (2003). The global k-means clustering algorithm. *Pattern Recognition*, *36*(2), 451–461. https://doi.org/10.1016/S0031-3203(02)00060-2

Puspasari, M. F., Suseno, Y. D., & Sriwidodo, U. (2017). Pengaruh current ratio, debt to equity ratio, total asset turnover, net profit margin dan ukuran perusahaan terhadap pertumbuhan laba. *Jurnal Manajemen Sumber Daya Manusia*, *11*(1), 121–133. http://www.ejurnal.unisri.ac.id/index.php/Manajemen/article/viewFile/1601/1407

Sutrisno. (2013). *Manajemen keuangan: Teori, konsep dan aplikasi*. Ekonisia.

Syakur, M. A., Khotimah, B. K., Rochman, E. M. S., & Satoto, B. D. (2018). Integration k-means clustering method and elbow method for identification of the best customer profile cluster. *IOP Conference Series: Materials Science and Engineering*, *336*(1), 012017. https://doi.org/10.1088/1757-899X/336/1/012017