IML Project Report - CSL2010

**Customer Segmentation and Market Basket Analysis**

Krish Patel, Harshita Pahadia , Ektedar Ahmad, Abuzer Baruchi

**Group No - 23**

**TA - Shalin Jain**

horizontal line



# INTRODUCTION

* **Customer Segmentation**: Customer segmentation aims to understand a firm’s customer base by analyzing their interactions, primarily through purchase behavior and patterns. Segmenting customers allows us to identify distinct groups and tailor strategies to their specific preferences and needs.
* **Market Basket Analysis**: Market basket analysis examines customers’ purchasing behavior at a detailed level, revealing underlying trends and associations in buying decisions. This approach uncovers patterns that may not be obvious to customers themselves, offering valuable insights for targeted marketing and personalized recommendations.

**PROJECT OBJECTIVE**

The primary objective of this project is to analyze customer purchasing behavior through customer segmentation and market basket analysis. By segmenting customers into distinct groups based on their buying patterns, we aim to better understand different customer types, their purchase frequency, and spending habits. Additionally, market basket analysis will help uncover associations between purchased items, revealing common product groupings that can be used to guide sales and promotional strategies.

**Dataset Overview**

**Data Source**: <https://www.kaggle.com/datasets/puneetbhaya/online-retail/data>

**Data Features** : The features which are provided in the dataset are :-

1. **InvoiceNo**: A unique identifier for each transaction or invoice. Each InvoiceNo corresponds to a single transaction, where multiple items may be part of the same invoice if purchased together.
2. **StockCode**: The unique code for each product in the inventory. Each represents a specific item, which helps track the item’s inventory and product details.
3. **Description**: A text description of the product, which provides details about the item purchased. For example, "WHITE HANGING HEART T-LIGHT HOLDER" describes an item as a hanging tea light holder.
4. **Quantity**: The number of units of each item purchased per transaction. For example, a quantity of 6 indicates that six units of the item were bought in that transaction.
5. **InvoiceDate**: The date and time when the transaction occurred, given in the format. This is important for analyzing purchasing patterns over time, such as peak shopping hours or seasonal trends.
6. **UnitPrice**: The price per unit of the item in the specified transaction currency (usually in GBP for this dataset). This helps in calculating the total revenue generated from each transaction.
7. **CustomerID**: A unique identifier assigned to each customer. However, in this dataset, there are missing values, and it is not unique for every row, as each customer can have multiple transactions.
8. **Country**: The country from which the customer made the purchase, e.g., "United Kingdom." This feature helps identify the geographical distribution of customers and analyze region-specific buying patterns.

## Importing the necessary libraries and Loading Dependencies

We began our analysis by importing the necessary libraries like **numpy, pandas, matplotlib, and seaborn** for data manipulation, visualization, and analysis. pandas and numpy provide essential tools for data handling and numerical operations. matplotlib and seaborn allow us to create insightful visualizations to explore the dataset.

We used libraries specifically for machine learning and clustering. **StandardScaler and PCA** from sklearn assist in standardizing and reducing dimensionality of the data, preparing it for clustering algorithms like **KMeans and AgglomerativeClustering**. **Plotly aids** in creating interactive plots for deeper exploration of clusters, while **Yellowbrick** helps in determining the optimal number of clusters using the elbow method.

To gain insights into customer purchasing behavior, we applied **apriori** from mlxtend.frequent\_patterns for market basket analysis and used association\_rules to identify frequent itemsets and rules among products. We also used **adjusted\_rand\_score** from sklearn.metrics to evaluate clustering performance, while warnings were imported to manage any warnings during the analysis.

**Data Preprocessing**

There were several challenges in using the raw data before going with the ML algorithms like K-means clustering and apriori algorithm. We have described below which were they and how we have deals with them :

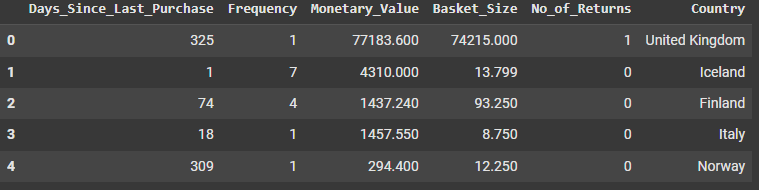
1. **NaN values in Description and CustomerID features** : We have dropped the rows in which NaN values were present in Description as it was very few in number as compared to the total rows in the dataset. While
2. **Negative Quantity with Positive Unit Price:** Negative quantities typically represent returns or cancellations, and a positive unit price suggests that the returned items have a positive monetary value. This situation could arise when customers return items for a refund or exchange. Hence, we have made a new attribute IsReturn in which negative quantity items are marked 1 otherwise 0.
3. **Entries such as 'POST' (Postage), 'D' (Discount), and 'M' (Manual)** don’t represent typical products customers would purchase or return. Thus, we opted to remove these entries to maintain a more product-focused analysis. These entries were minimal in count, so their removal has little impact on the dataset's overall integrity.
4. **Unit Price of 0:** These transactions likely represent promotional items, freebies, or items included in special deals where the monetary value is set to zero. Since these entries are not critical to our analysis, and they make up only a small fraction of the dataset, we decided to remove these rows to focus on transactions with a direct impact on revenue and purchasing behavior.

### 

### Feature Engineering:

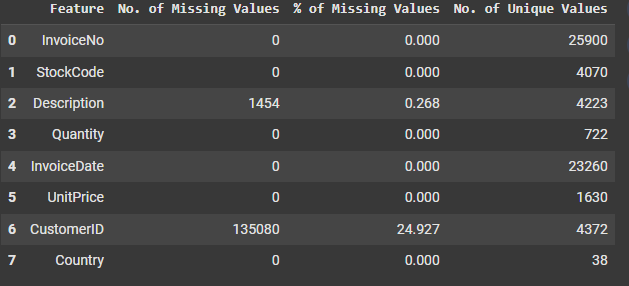
We also have added many other attributes to our dataset to find more relative buying patterns in the customers. They are :

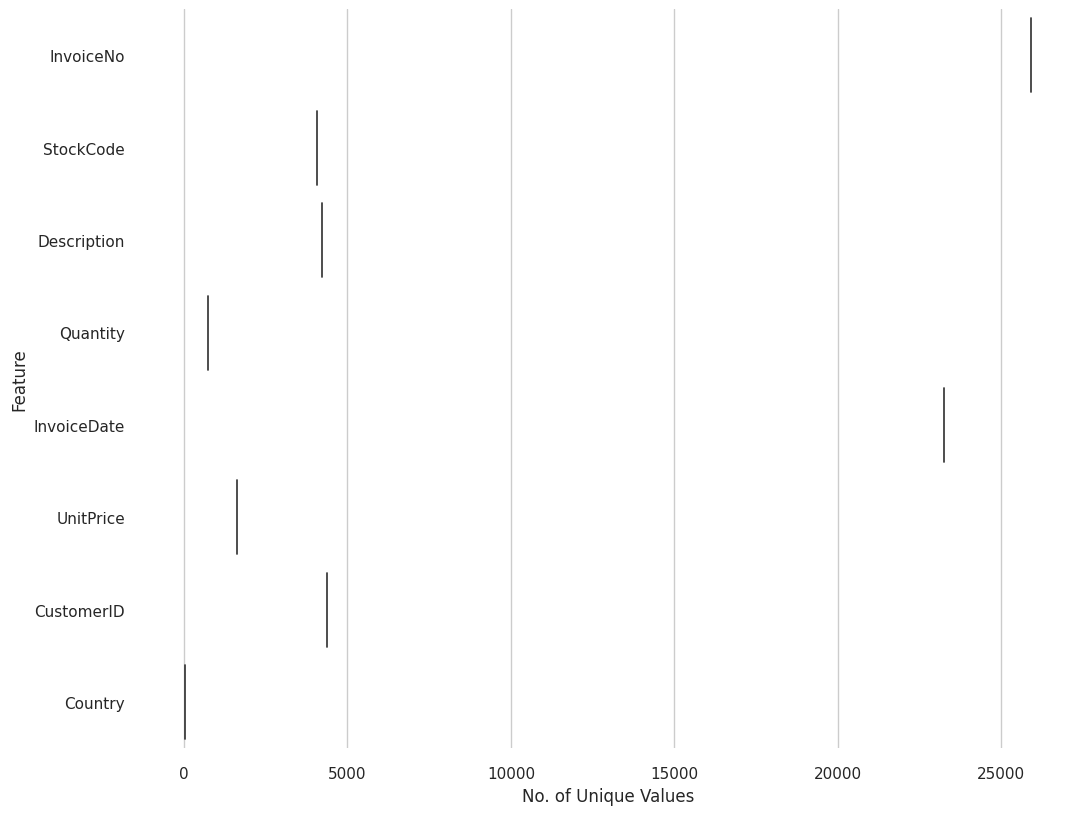
1. **Addition of 'TotalAmount' :** Defined as Quantity \* Unit Price. It captures the financial aspect of each transaction.
2. **Days Since Last Purchase (Recency):** This feature captures the recency of customer transactions, excluding returns. It's derived by calculating the number of days between the latest transaction date and the current date.
3. **Frequency :** This attribute represents the frequency of customer purchases, determined by counting the unique invoices for each customer, excluding returns.
4. **Monetary Value :** This attribute captures the total amount spent by each customer, derived by summing up the Total Amount values for all their transactions, excluding returns.
5. **Basket\_Size :** This attribute represents the average size of a customer's shopping basket and is calculated by considering the mean quantity of items per invoice.
6. **No\_of\_Returns :** This attribute represents the total number of returns made by each customer, providing valuable insights into their engagement and potential challenges.

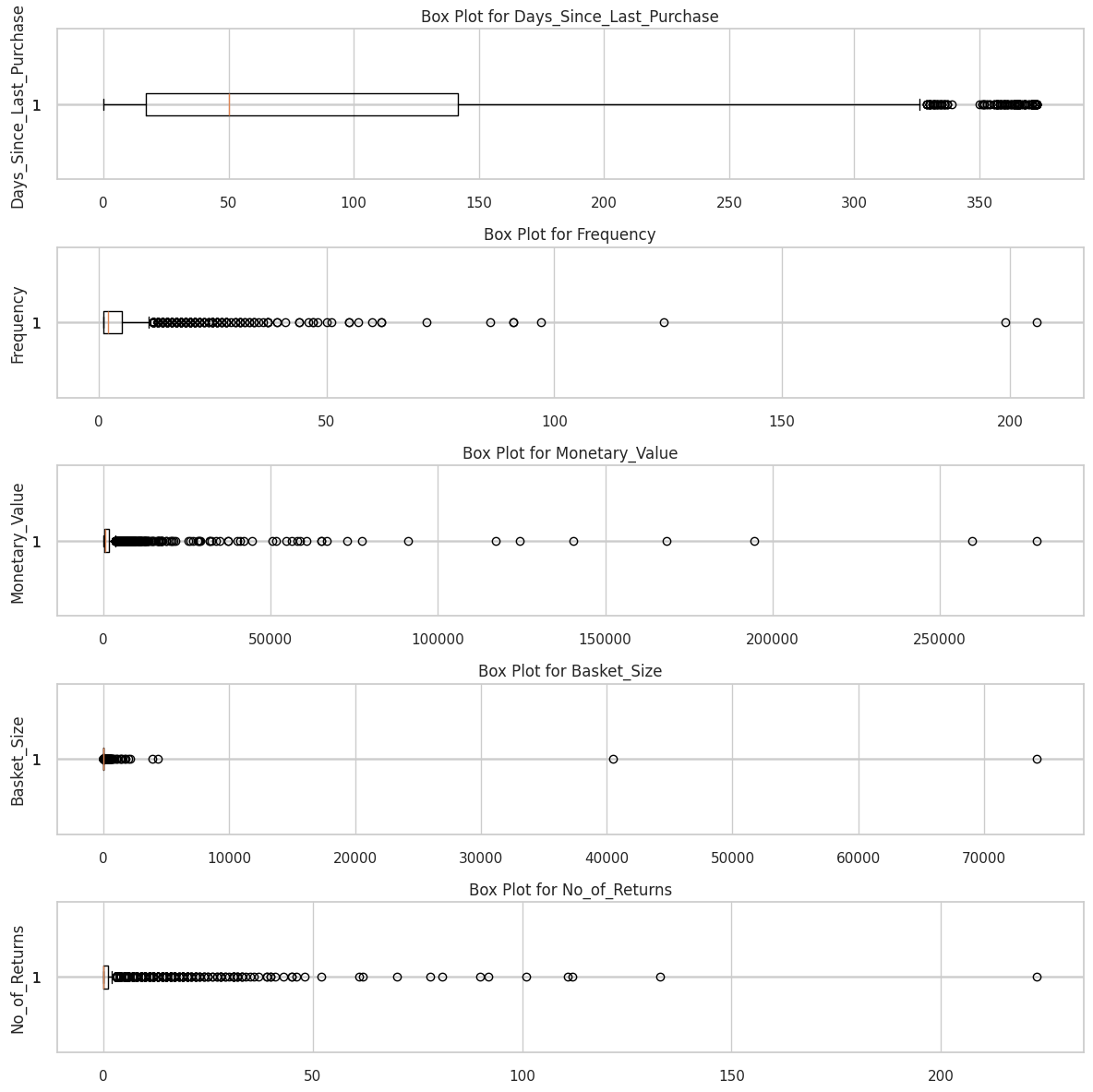


**Exploratory Data Analysis :** EDA provides a preliminary understanding of the dataset's structure, distributions, and relationships among key features. By visualizing and summarizing important variables, we can identify meaningful insights about customer purchasing behavior and pinpoint any necessary data transformations before moving forward with model development.

1. Describe, Info, Shape, Value\_Counts for each column, % of missing values, No. of Unique Values, datatypes of features etc

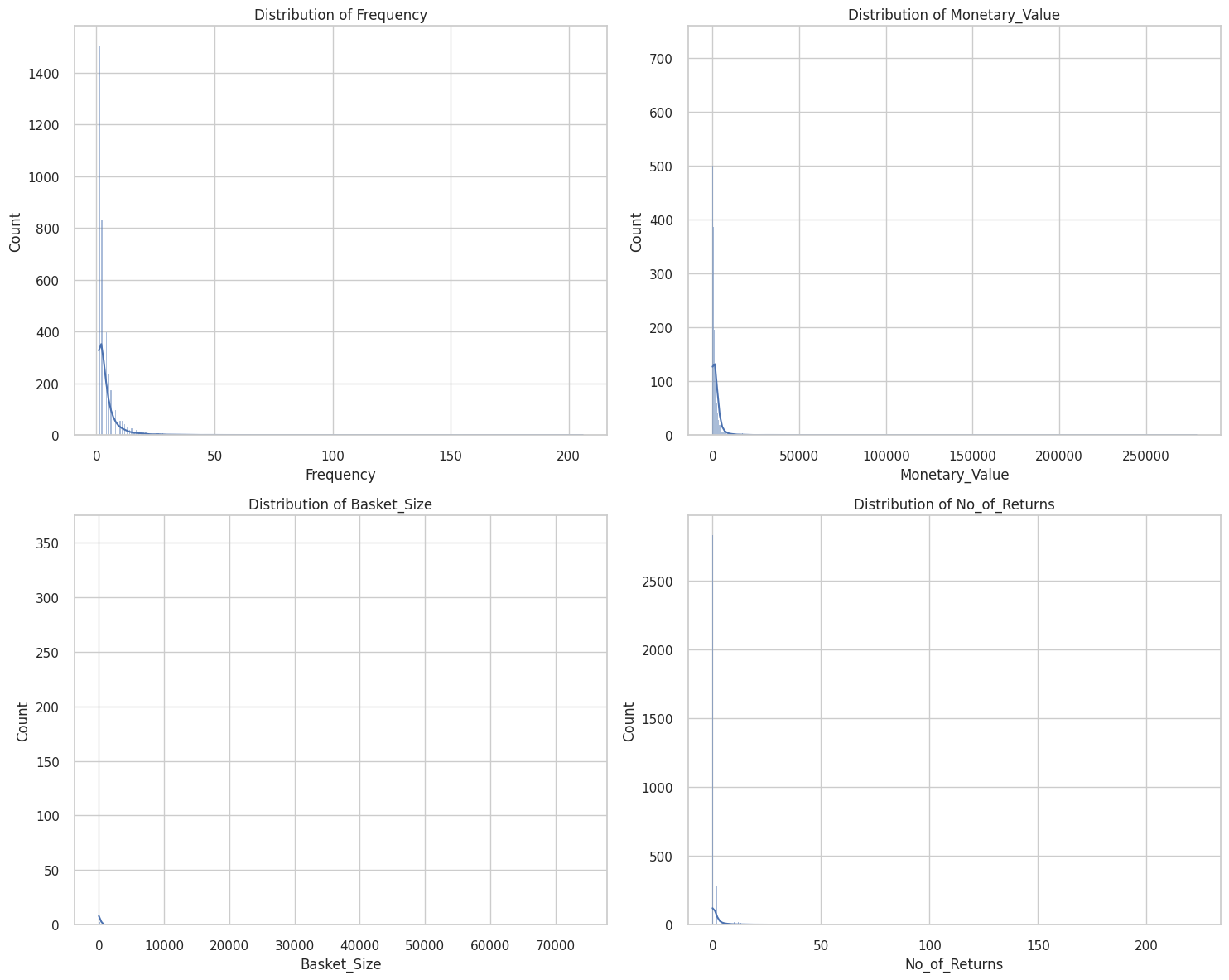


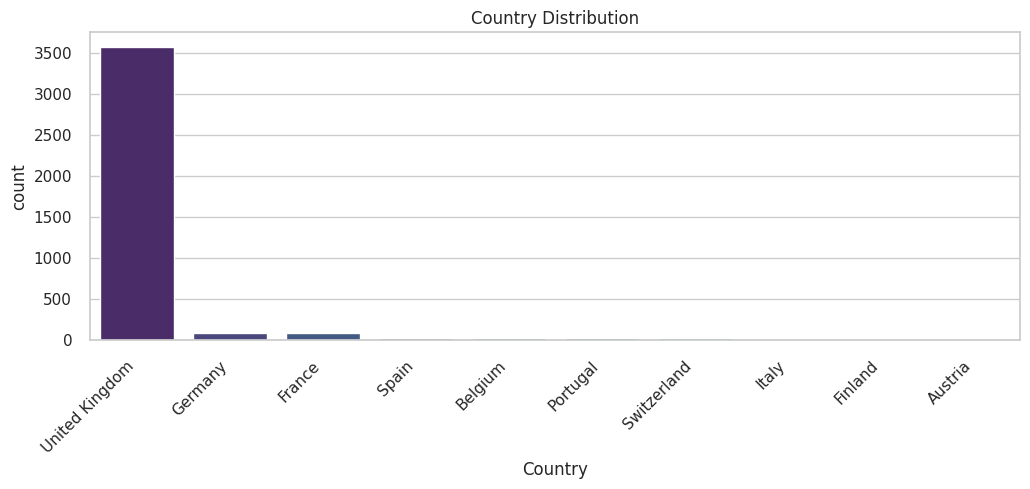
1. Plot for Unique Values   
   
2. Boxplot



Encountering extreme outliers in our box plot has obscured the visibility of the box itself, hindering our ability to interpret the distribution of our numerical columns. To remedy this, we'll be calculating upper and lower bounds to identify and remove these outliers. This process aims to enhance the clarity of our visualizations and ensure a more accurate representation of the underlying data distribution.

1. Histogram with Kernel Density Estimation (KDE)

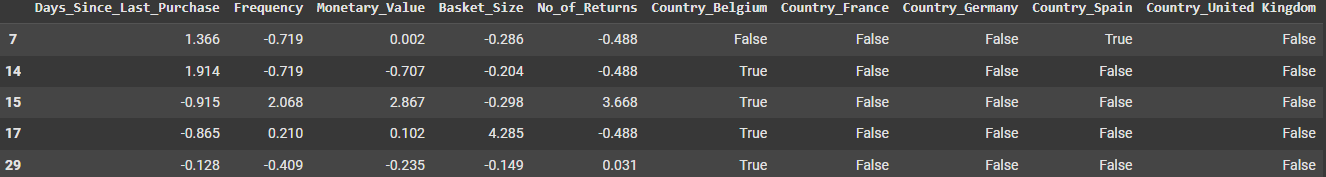


1. Bar Graph for Countries   
   
2. Correlation (Heatmaps) between features like Days\_Since\_Last\_Purchase,Frequency,Monetary\_Value, Basket\_Size,No\_of\_Returns



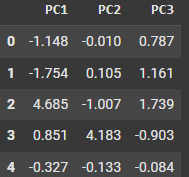
Highly Correlated features : Monetary value and Frequency

Least correlated features : Basket Size and Days since last Purchase

**Standard Scaling :** For applying PCA and K-means Clustering we are Standard Scaling our below final dataset 

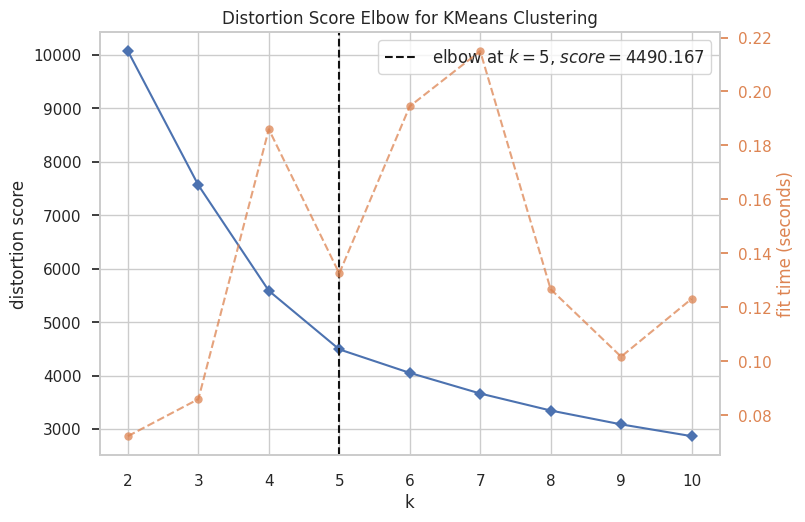
We have performed PCA on our data because it :

1. Dimensionality Reduction
2. Improved Computational Efficiency
3. Visualization
4. Variance Maximization
5. Noise Reduction



**Why K-Means Algorithm ?**

K-Means is effective for customer segmentation because it **identifies distinct customer groups** based on similarities in behavior and demographics. Its simplicity and interpretability allow for easy understanding of the resulting clusters, facilitating data-driven decision-making. The **algorithm is scalable**, making it suitable for large datasets, and helps optimize resources by tailoring marketing strategies to specific segments. Additionally, K-Means informs product development by highlighting customer needs, enhances customer relationship management through personalized engagement, and can adapt dynamically to changing behaviors, ensuring ongoing relevance in segmentation efforts.

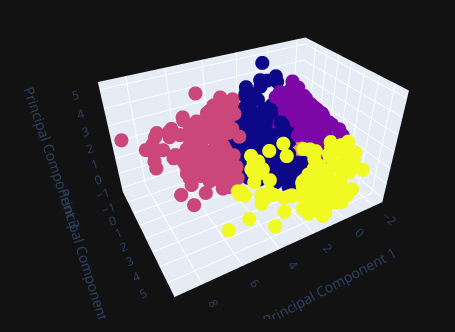


Elbow Point : k = 5

The Distortion Score Elbow suggests us that 5 clusters are best for K- means Clustering.

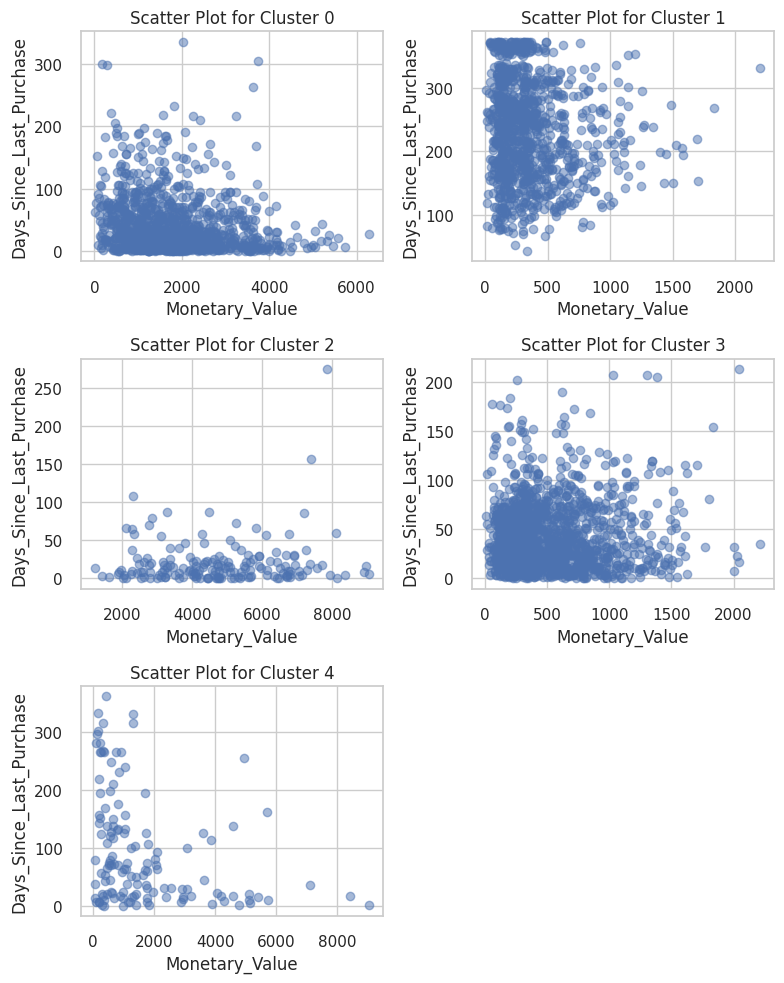
The elbow point suggests a balance between having enough clusters to capture the data's structure without overfitting.

Scatter Plot of Clusters :

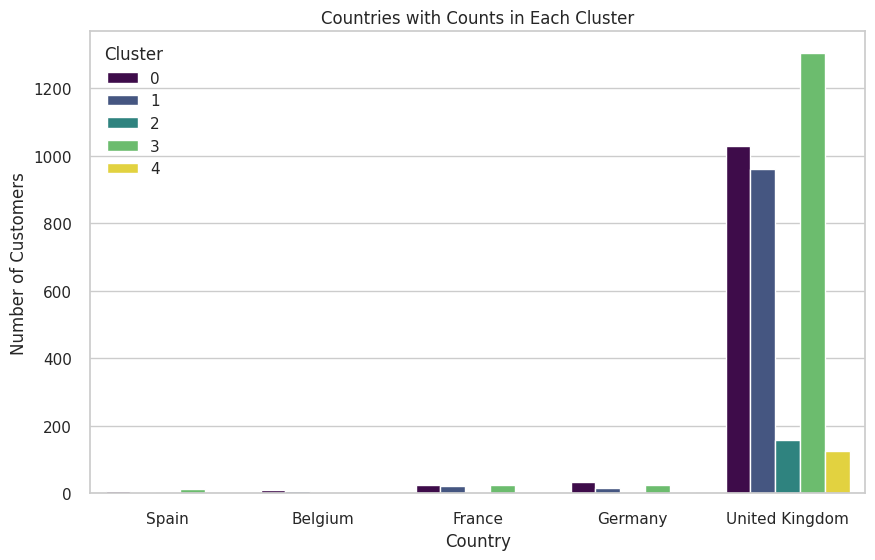


Distribution Plot of 5 Clusters :

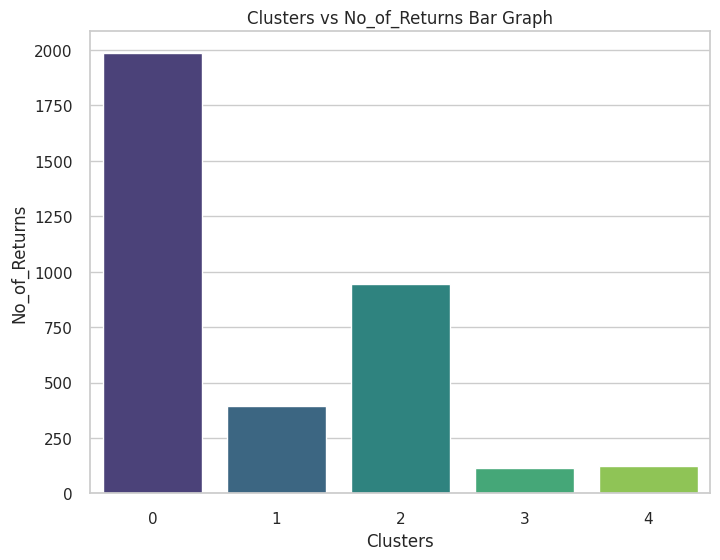


Scatter Plot of Clusters   


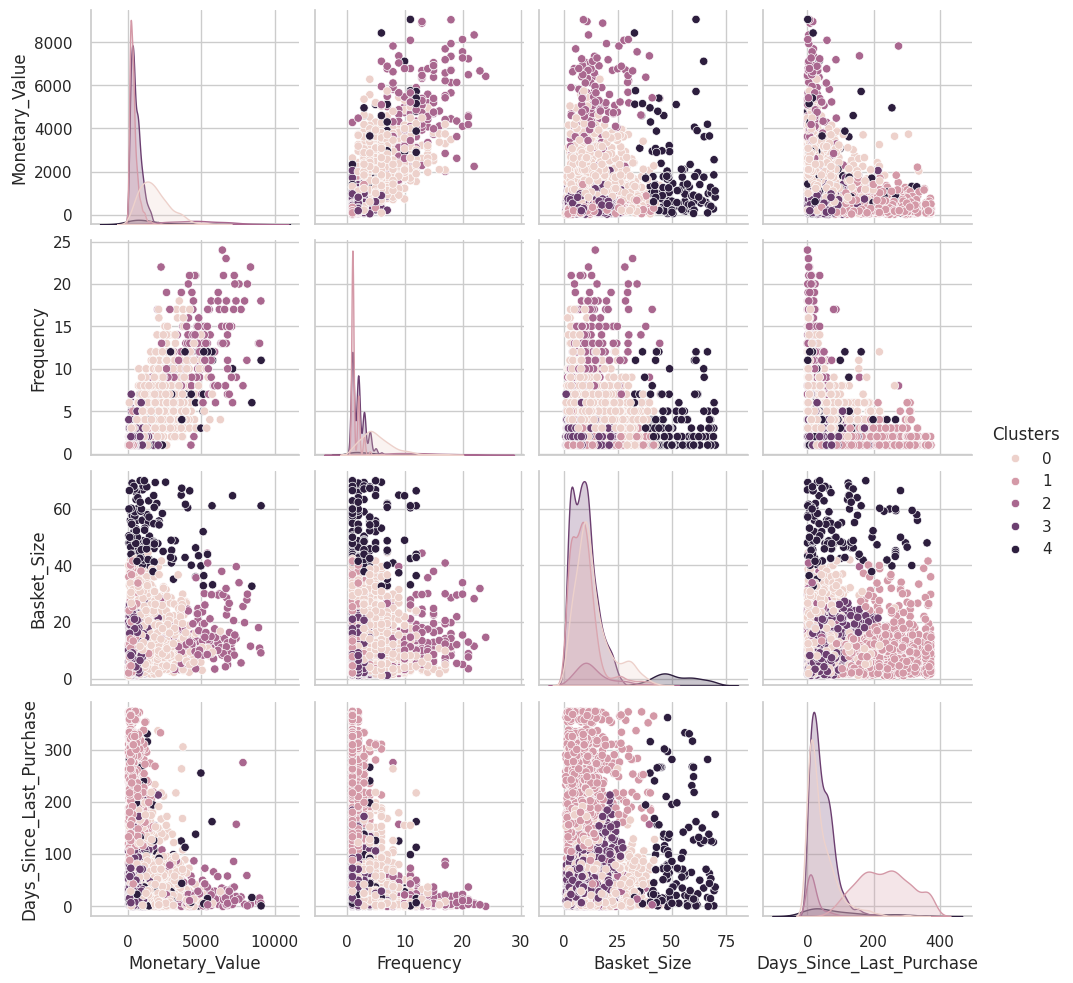
Count plot of Countries with Clusters :



Bar Plot for No of Returns vs Clusters



Pairwise plot between the features



**CONCLUSION :**

### Cluster 0: Premium Spenders

1. Monetary Value: High spending customers with a median value around £1603.99.
2. Frequency: Frequent shoppers with a median frequency of 5 transactions.
3. Days Since Last Purchase: Active customers who make purchases relatively frequently (median of 26 days).
4. Basket Size: Typically larger basket sizes, with a median around 10.90 items per transaction.
5. No. of Returns: Some customers have made returns, but it's not a predominant behavior.
6. Recommendation: Target this group with exclusive offers and loyalty programs to enhance their brand loyalty.

### Cluster 1: Low Activity Customers

1. Monetary Value: Low spending customers with a median value around £291.76.
2. Frequency: Low transaction frequency, mostly 1 transaction per customer.
3. Days Since Last Purchase: Generally infrequent buyers, with a median of 234 days between purchases.
4. Basket Size: Smaller basket sizes compared to other clusters.
5. No. of Returns: Rarely make returns.
6. Recommendation: Implement strategies to re-engage and attract these customers, perhaps through special promotions.

### Cluster 2: High-Value Loyal Customers

1. Monetary Value**:** High spending customers with a median value around £4684.76.
2. Frequency**:** Regular shoppers with a median frequency of 12 transactions.
3. Days Since Last Purchase**:** Active and engaged customers with a median of 11 days between purchases.
4. Basket Size**:** Generally larger basket sizes, with a median around 11.45 items per transaction.
5. No. of Returns**:** Some customers make returns, but it's relatively low compared to their overall activity.
6. **Recommendation** : Reward loyalty with exclusive perks, personalized offers, and loyalty programs.

### Cluster 3: Emerging Customers

1. Monetary Value**:** Customers with moderate spending, median around £430.50.
2. Frequency**:** Moderate transaction frequency, with a median of 2 transactions.
3. Days Since Last Purchase: Varied engagement, but a significant number have recently made a purchase.
4. Basket Size: Varies, with a median basket size of around 8.82 items.
5. No. of Returns: Mostly don't make returns.
6. Recommendation**:** Encourage continued engagement with targeted promotions and personalized incentives.

### Cluster 4: Diverse Shoppers

1. Monetary Value**:** Diverse spending patterns, with a median around £1017.50.
2. Frequency**:** Varied transaction frequency, ranging from 1 to 5 transactions.
3. Days Since Last Purchase: Diverse engagement, from relatively recent to more infrequent purchases.
4. Basket Size: Varied basket sizes, showing a range of shopping behaviors.
5. No. of Returns: Some customers make returns.
6. Recommendation: Tailor marketing strategies to accommodate the diverse preferences within this group.

# Market Basket Analysis

We have implemented a market basket analysis using the Apriori algorithm to identify purchasing patterns and generate product recommendations.

**1. Data Preparation:** prepare\_data(data)

This function prepares the transaction data for analysis by performing several cleaning steps:

* Dropping Missing Values: Removes rows with any missing values to ensure data integrity.
* Filtering Out Specific Rows: Excludes rows with "POST" in StockCode and invoices containing "C" to focus on relevant transactions.
* Filtering Negative Values: Ensures that only valid purchases (positive quantities and prices) are included.
* Handling Outliers: Applies thresholds to the Quantity and UnitPrice columns to replace outlier values, which helps in maintaining the reliability of the data.

Dataset used for analysis is clean, accurate, and representative of actual purchasing behavior.

## 2. Outlier Detection: calculate\_outlier\_thresholds(data, column)

This function calculates the lower and upper thresholds for outlier detection using the Interquartile Range (IQR) method.

* It computes the first and third quartiles (Q1 and Q3), calculates the IQR, and determines the thresholds for outliers.

This function helps to enhance the quality of the data, preventing skewed results in the analysis.

## 3. Creating Product Matrix: create\_product\_matrix(data, by\_stockcode=True)

This function creates a binary matrix representation of the transactions:

* When by\_stockcode is True, it groups the data by InvoiceNo and StockCode, summing the quantities to create a matrix where a value of 1 indicates that an item was purchased in a transaction.
* Alternatively, it can group by product descriptions.

The product matrix is essential for the Apriori algorithm, as it provides a clear representation of item purchases across transactions.

## 4. Product Name Retrieval: get\_product\_name(data, stock\_code)

This function retrieves the product name corresponding to a given StockCode from the dataset.

This is useful for displaying human-readable product names in recommendations, enhancing the user experience.

## 5. Generating Association Rules: generate\_association\_rules(data, min\_support=0.01)

This function generates frequent itemsets and association rules:

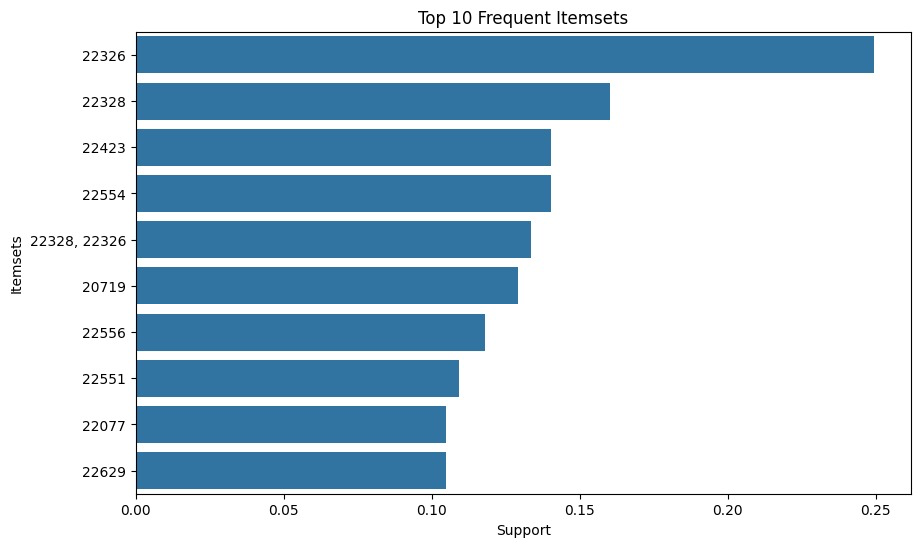
* It first creates the product matrix and then applies the Apriori algorithm to identify frequent itemsets based on the specified minimum support.
* It generates association rules from these itemsets.

This function is central to market basket analysis, as it uncovers relationships between items, enabling the discovery of purchase patterns.

## 6. Plotting Top Itemsets: plot\_top\_itemsets(frequent\_itemsets)

This function plots the top 10 frequent itemsets based on support values.

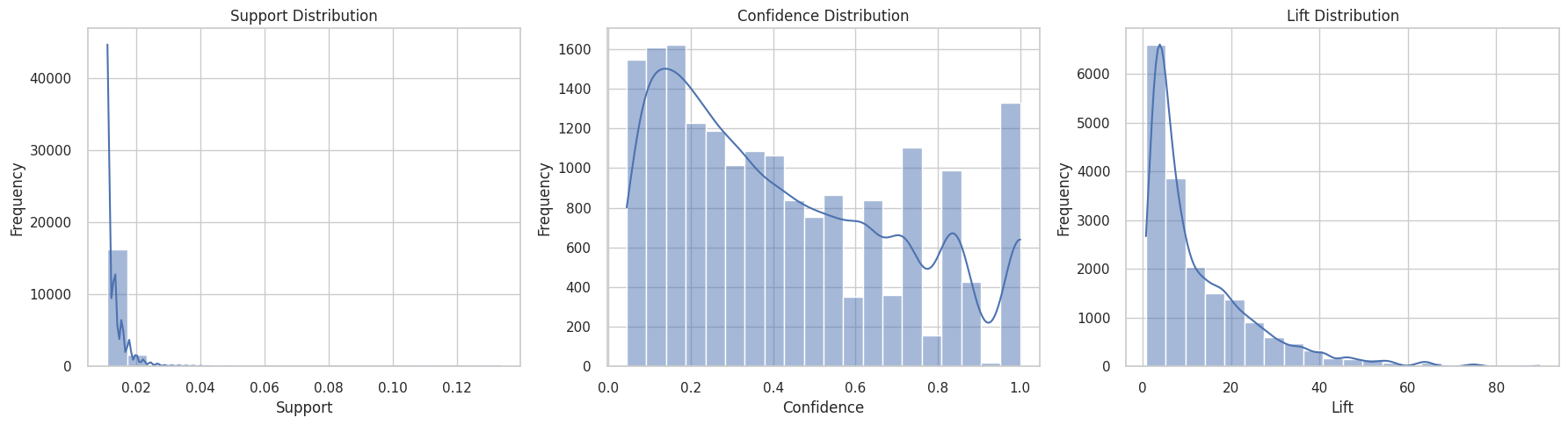
By highlighting the most commonly purchased item combinations, this visualization provides insights into popular purchasing patterns.



## 7. Plotting Confidence vs. Lift: plot\_confidence\_lift(rules)

This function creates a scatter plot to visualize the relationship between confidence and lift of the generated association rules.

This plot helps in evaluating the effectiveness of the rules, providing a visual means to assess their strength and relevance.



## 8. Product Recommendation: recommend\_products(data, target\_product\_id, support\_threshold=0.01, max\_recommendations=5)

This function generates product recommendations based on a target product ID:

* It retrieves association rules and identifies products that are frequently purchased together with the target product.

This functionality enhances customer experience by providing tailored product recommendations, potentially increasing sales through cross-selling.

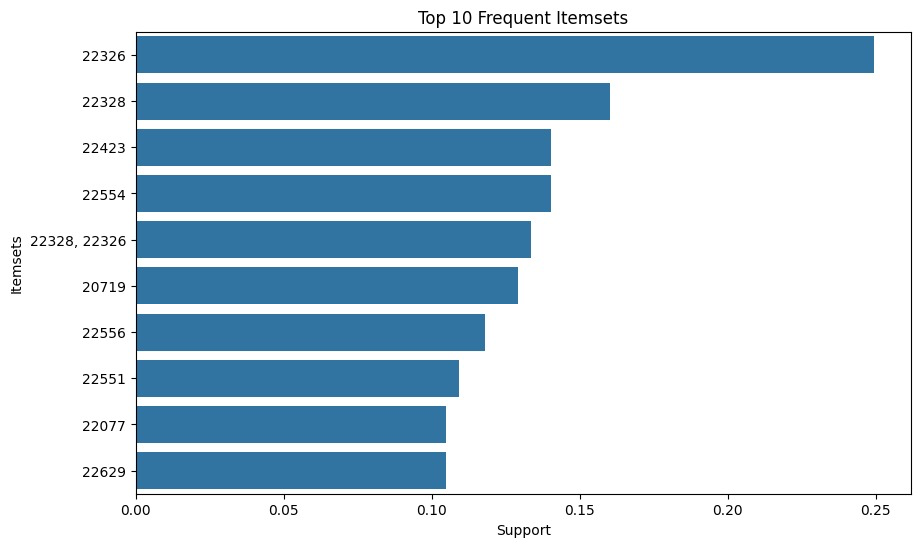
## 9. Product Recommendation System: product\_recommendation\_system(data, support\_threshold=0.01, max\_recommendations=5)

This function prompts the user to input a product ID and retrieves recommendations based on that input.

* It validates the input and returns a list of recommended products along with their names.

This interactive component makes the analysis user-friendly, allowing users to engage with the data and receive actionable insights based on their input.

Overall, it provides a comprehensive approach to market basket analysis, from data preprocessing to generating actionable product recommendations, enabling businesses to leverage customer purchasing behavior effectively.

Support Plot of Frequent Itemsets   


RESULT : After entering the stock code, it recommends the other products also.

