**Customer Segmentation Using RFM Analysis and K-Means Clustering**

## **Abstract**

This project provides a comprehensive customer segmentation approach by integrating **Recency, Frequency, Monetary (RFM)** analysis with **K-Means clustering**, enhanced by thorough **outlier detection and handling** strategies. Using a retail transaction dataset, the project filters and cleans the data, identifies and treats outliers based on IQR, and segments customers into meaningful groups. The system includes visual analytics and a **Flask-based web app** for exploring segmentation results.

## **1. Introduction**

Understanding customer behavior is key to personalized marketing. This project builds a data-driven pipeline using RFM analysis and clustering to segment customers. It specifically addresses the limitations of simplistic segmentation by:

* Cleaning and standardizing data
* Removing outliers that distort analysis
* Applying clustering to reveal hidden customer patterns

## **2. System Analysis**

### **Existing System**

* Manual rule-based segmentation
* Ignores extreme or inconsistent customer behavior (outliers)
* Limited use of visualization or ML-based methods

### **Proposed System**

* Preprocessing with rigorous outlier handling
* Clustering using K-Means (and DBSCAN for comparison)
* Visual exploration of clusters
* 3D & PCA-based representation of customer segments

## **3. Objectives**

* Load and clean retail customer data
* Calculate RFM values per customer
* Remove statistical outliers to ensure clean clustering
* Apply and evaluate clustering (K-Means, DBSCAN)
* Develop visual reports and a Flask-based app

## **4. System Specification**

|  |  |
| --- | --- |
| **Component** | **Details** |
| **Language** | Python |
| **Libraries** | pandas, seaborn, matplotlib, sklearn, Flask |
| **Visualization** | 3D plots, violin plots, PCA |
| **Platform** | Jupyter Notebook, Flask Web App |

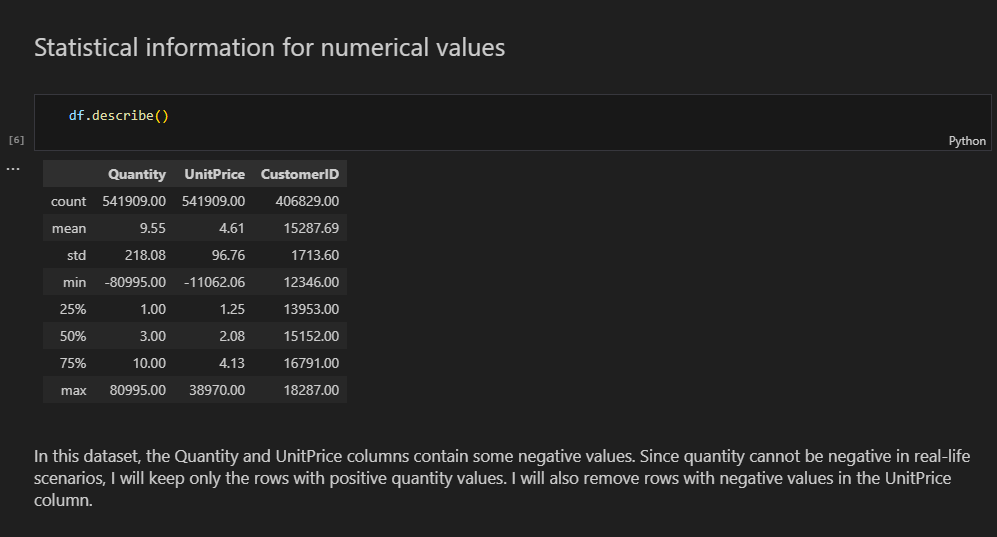
## **5. Methodology**

### **5.1 Data Overview**

* The dataset used for this project is sourced from an online retail store and contains historical transaction data from December 2010. The data includes transactional records for various customers across multiple countries. Each row represents a single product purchased as part of an invoice.

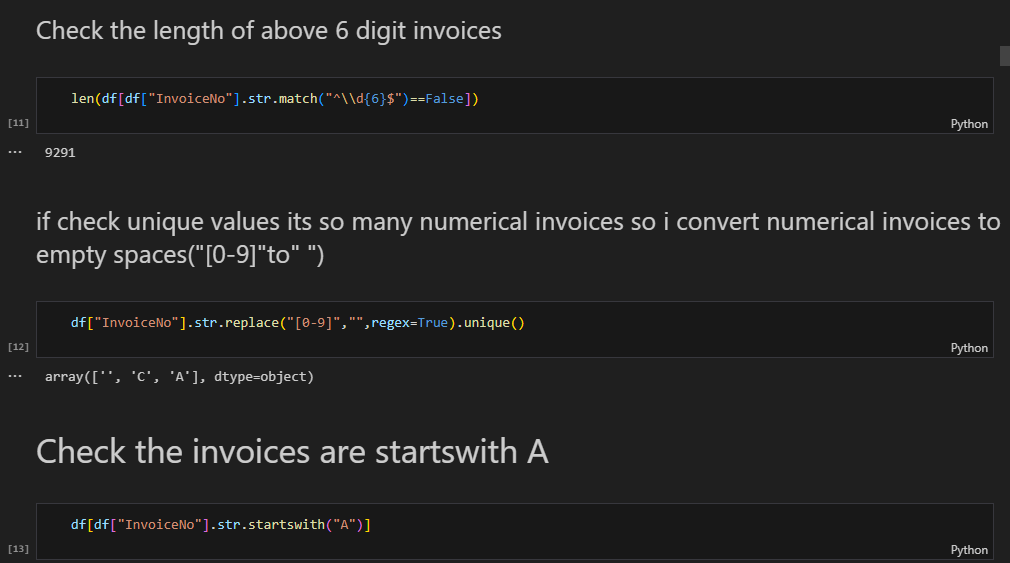
**Key Features in the Dataset**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **InvoiceNo** | A unique identifier for each transaction (invoice). |
| **StockCode** | A unique code for each product (not used in this analysis). |
| **Description** | Product name/description (not used for segmentation). |
| **Quantity** | The number of units of the product purchased. |
| **InvoiceDate** | Date and time when the transaction was generated. |
| **UnitPrice** | Price per unit of the product. |
| **CustomerID** | Unique identifier for each customer. |
| **Country** | Country name of the customer. |

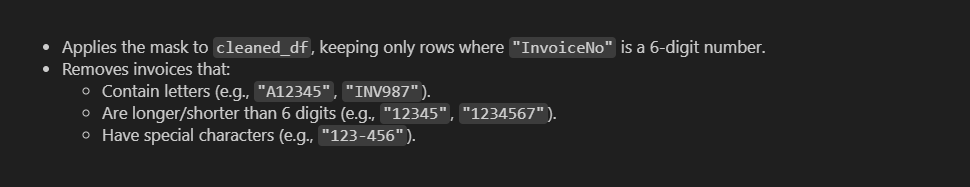


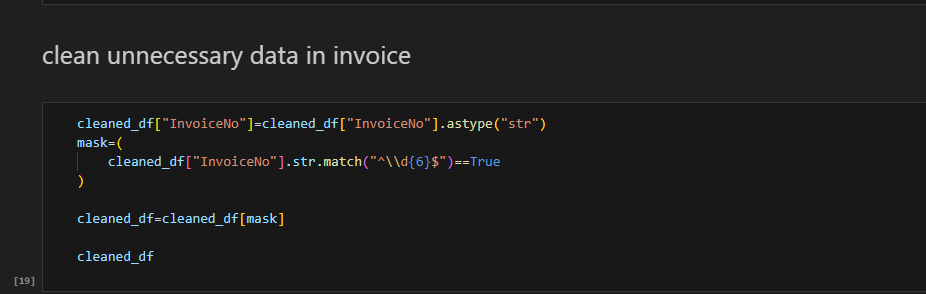
#### **Preprocessing Highlights:**

* **Filtering Data for United Kingdom**:
  + To maintain regional consistency and reduce noise, only customers from the **United Kingdom** are retained in the final dataset.
* **Handling Missing Values**:
  + Transactions with **missing Customer IDs** are dropped since segmentation requires customer identification.

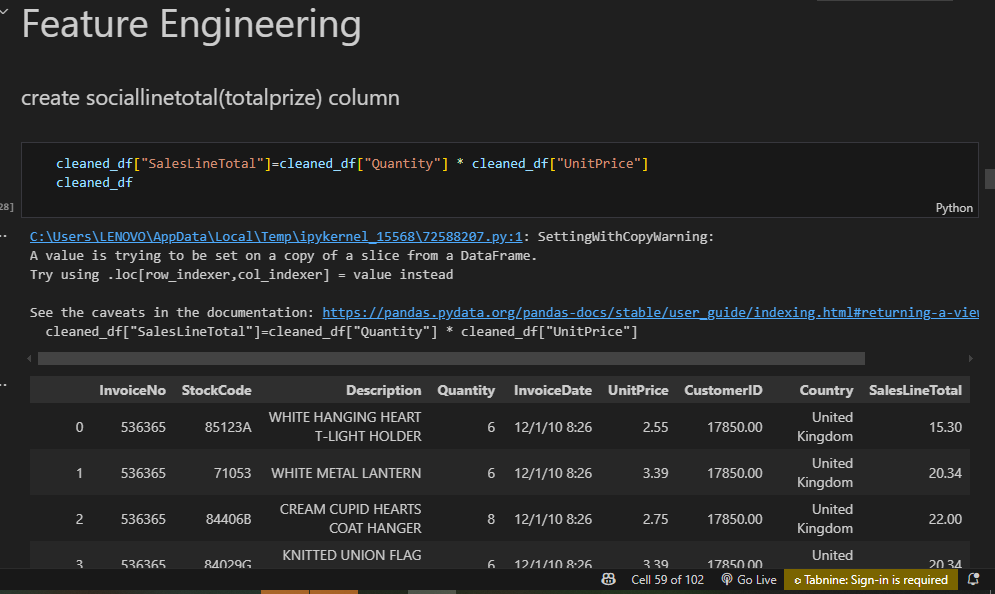


* **Filtering Valid Purchases**:
  + Only transactions with **positive Quantity and UnitPrice** are considered. Refunds or incorrect entries (negative values) are removed.

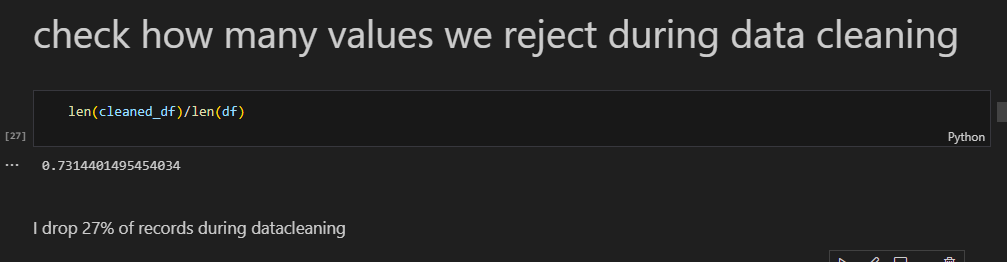




* **Creating the TotalPrice Column**:
  + A new column TotalPrice = Quantity × UnitPrice is calculated to determine the total value of each transaction.

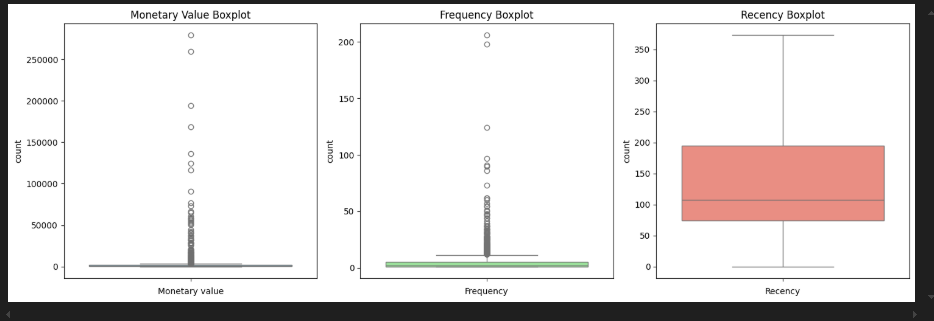


* **Aggregation by Customer**:
  + Transactions are grouped by CustomerID to derive the following:
    - **MonetaryValue**: Sum of TotalPrice per customer
    - **Frequency**: Number of unique invoices per customer
    - **Recency**: Days since the most recent purchase from a fixed reference date



### **5.2 Outlier Detection & Handling**

Outliers are customers whose purchasing behavior significantly deviates from the majority. Instead of removing these valuable customer data points, this project takes a more strategic and inclusive approach by detecting, labeling, and analyzing outliers as **separate segments**.



#### **Detection of Outliers Using IQR Method**

The project applies the **Interquartile Range (IQR)** method to detect outliers for each RFM (Recency, Frequency, MonetaryValue) metric:

* **Q1** and **Q3** (the 25th and 75th percentiles) are computed.
* The **IQR** is calculated as Q3 - Q1.
* Outliers are defined as values that fall below Q1 - 1.5 × IQR or above Q3 + 1.5 × IQR.

This method is applied separately to identify:

* **Monetary Outliers**
* **Frequency Outliers**
* **Recency Outliers**

#### **Categorization and Labeling of Outliers**

Instead of removing the detected outliers, the project separates and categorizes them into distinct groups:

* **Monetary Outliers Only**: Assigned to Cluster -1 (labeled as *PAMPER*)
* **Frequency Outliers Only**: Assigned to Cluster -2 (labeled as *UPSELL*)
* **Customers who are both Monetary and Frequency Outliers**: Assigned to Cluster -3 (labeled as *DELIGHT*)

Each of these outlier groups is stored in a dedicated DataFrame and later merged with the main clustering output.

#### **Segregated Visualization and Analysis**

To provide better insights into the behavior of outlier customers:

* **Violin plots** are used to compare Recency, Frequency, and MonetaryValue across the outlier clusters.
* Each group is analyzed separately to understand their potential business value.

#### **Reintegration with Main Dataset**

After performing K-Means clustering on the **non-outlier customers**, the labeled outlier groups are:

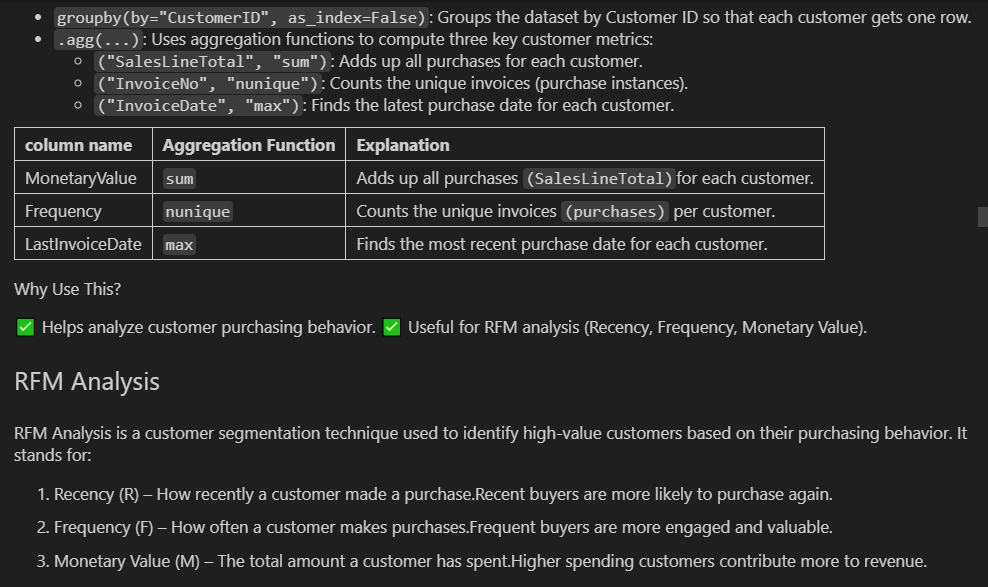
* **Merged back** with the main dataset
* Included in the **final customer segmentation output**

This ensures that no customer is excluded from the analysis and that **high-value or unusual customers** are given proper strategic attention.

### **5.3 RFM Feature Engineering**

Calculated per customer:

* **Recency**: days since last purchase
* **Frequency**: number of purchases
* **Monetary**: total spending

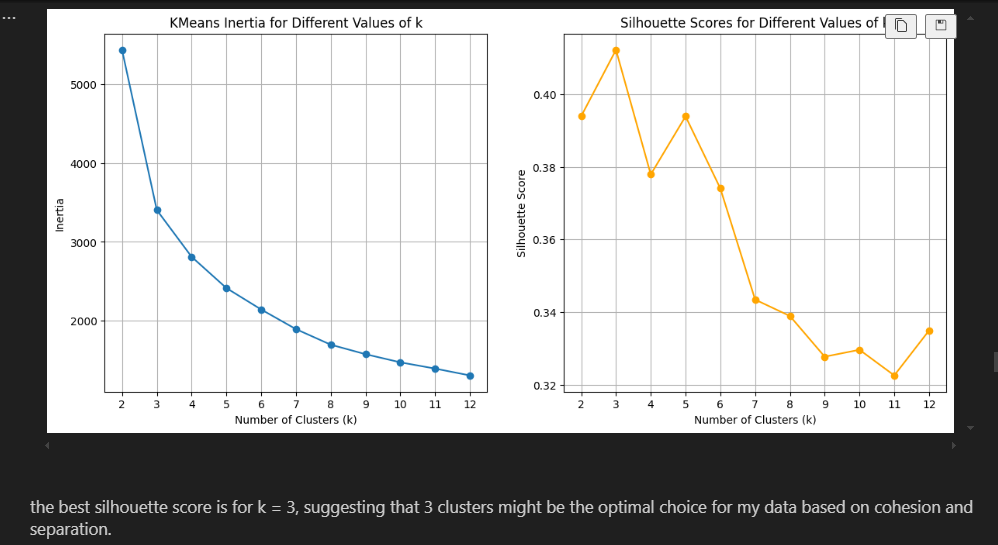


### **5.4 Feature Scaling**

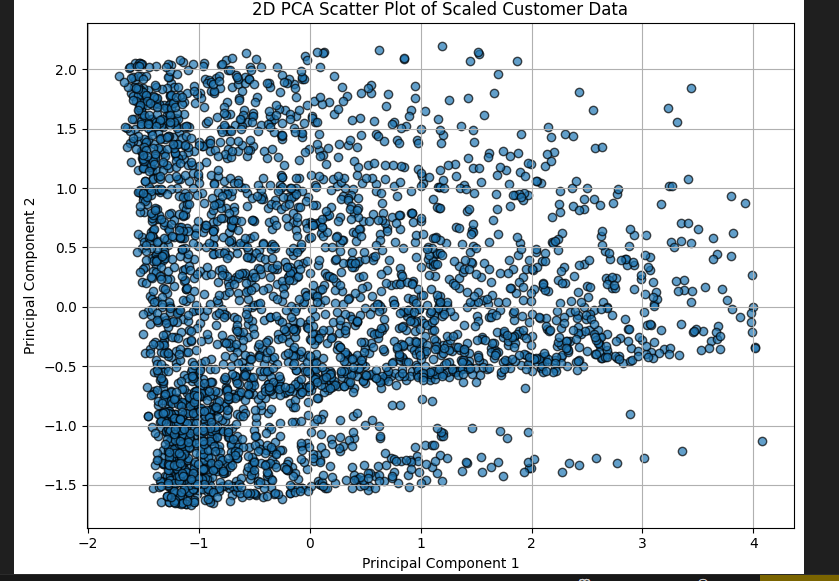
StandardScaler standardizes RFM features for clustering.

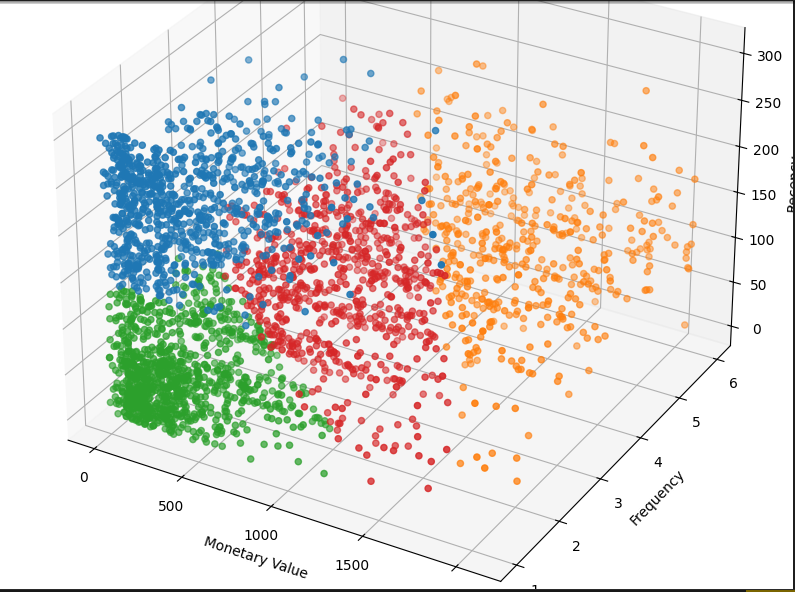
### **5.5 Clustering Algorithms**

* **K-Means Clustering**:
  + Optimal K chosen via Elbow method



* + Visualized in 3D and PCA-reduced 2D





* **DBSCAN** (for comparison):
  + Clustered based on density
  + Silhouette scores computed if valid

## **6. Module Description**

### **6.1 Data Loading Module**

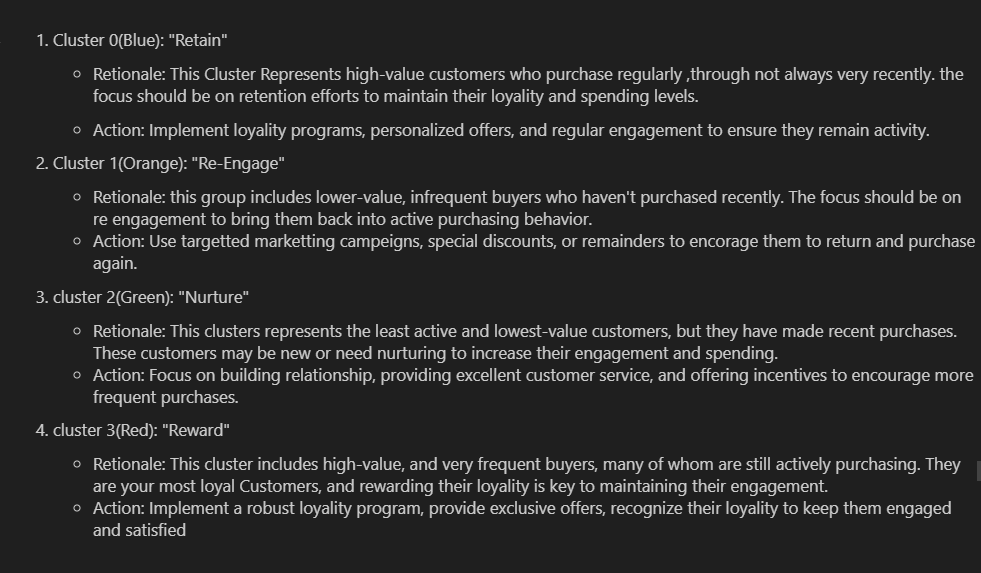
* Loads CSV, inspects types, handles missing values

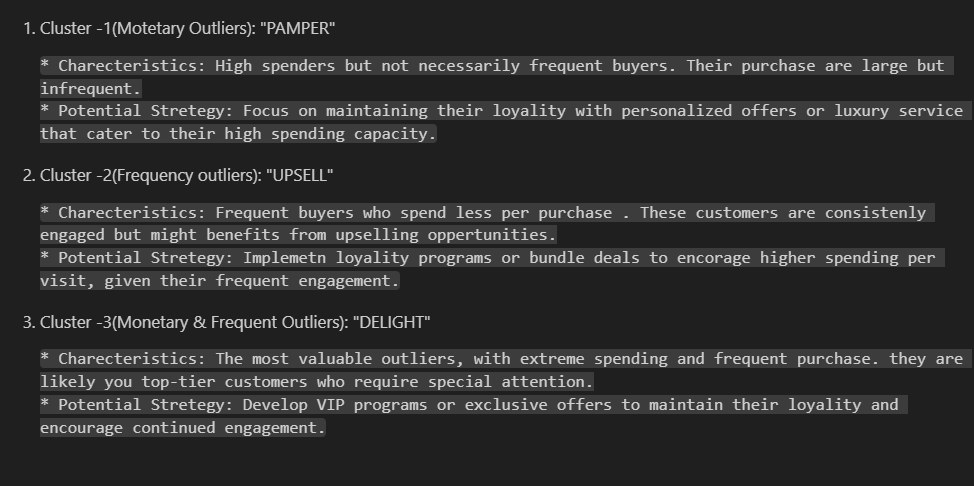
### **6.2 Outlier Analysis Module**

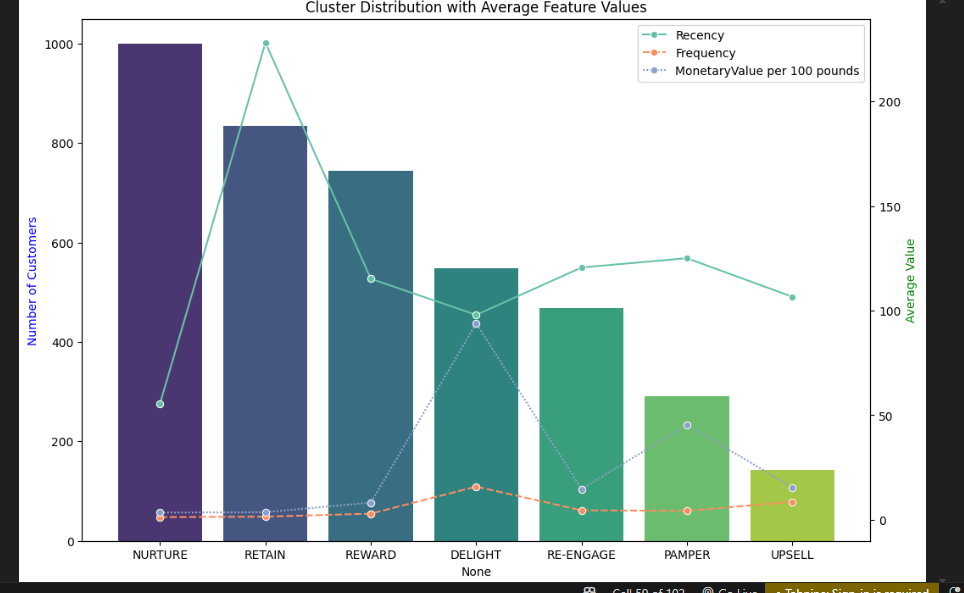
* IQR analysis for Recency, Frequency, Monetary
* Violin plots for visual verification
* Outliers excluded from core dataset

### **6.3 RFM & Clustering Module**

* Computes RFM scores
* Standardizes and clusters using K-Means
* Visualizes clusters with 3D and PCA







\=

### **6.4 Web App Module**

* Flask app with forms for uploading new data
* Visualizes cluster predictions

## **7. Conclusion**

Your project goes beyond basic segmentation by **thoughtfully removing outliers**, which drastically improves clustering accuracy. K-Means and DBSCAN are used for meaningful grouping, and visualizations offer deep insight. The web app completes the solution for real-world application.

## **8. Future Enhancements**

* Integrate more demographic/behavioral features
* Automate cluster evaluation metrics (silhouette, Davies–Bouldin)
* Allow real-time user uploads in the web app
* Explore advanced models like GMM, HDBSCAN