**High-Level Design: Customer Segmentation for Online Retail**

**1. Problem Statement**

The goal is to segment customers of an online retail business based on their purchasing behavior. By identifying distinct customer groups with similar characteristics (e.g., purchase frequency, spending habits, engagement duration), the business can tailor marketing strategies, improve customer retention, and optimize resource allocation for targeted campaigns. This project aims to apply unsupervised machine learning techniques to achieve this segmentation using historical transaction data.

**2. Understanding the Data**

* Dataset: online\_retail.csv
* Source: Contains transactional data for a UK-based online retailer.
* Initial Features (8):
  + InvoiceNo: Unique identifier for each transaction. Categorical (treated as object).
  + StockCode: Unique product code. Categorical (treated as object).
  + Description: Product description. Categorical (treated as object). *Dropped during preprocessing.*
  + Quantity: Number of units purchased per transaction. Numerical (int64).
  + InvoiceDate: Date and time of the transaction. Categorical (object), converted to datetime.
  + UnitPrice: Cost of a single unit of the product. Numerical (float64).
  + CustomerID: Unique identifier for each customer. Numerical (float64), treated as categorical ID.
  + Country: Country where the customer resides. Categorical (object).
* This is an unsupervised learning problem therefore, there is no predefined target variable.

**3. Assumptions:**

* If a customer ID is missing, we can’t track that customer’s behavior so we excluded those rows.
* Negative quantities usually mean returns, which don’t reflect regular buying habits, so we left those out too.
* We capped very high values in spending and frequency to avoid having extreme outliers skew the results.
* We assume that our engineered features like how recently someone shopped, how often, how much they spend, and how long they’ve been around capture enough of their behavior to build meaningful segments.
* Lastly, we used clustering methods like K-Means and DBSCAN, which group customers based on how similar they are after cleaning and transforming the data.

**4. Data Cleaning Strategy:**

* Missing Values:
* Description: Column dropped due to irrelevance for behavioral segmentation.
* CustomerID: Rows with missing CustomerID dropped, as customer identification is essential for the analysis.
* Duplicates: Transaction-level duplicates (based on key identifiers like InvoiceNo, StockCode, CustomerID, etc.) were handled by grouping and aggregating Quantity (sum) and UnitPrice (mean). Full row duplicates were verified as removed.
* Irrelevant Data: Rows with negative Quantity (likely returns) were filtered out.
* Outlier Handling:
* Transaction Level: Quantity and UnitPrice outliers were capped at the 90th percentile based on IQR limits.
* Customer Level: Engineered features (Recency, Frequency, Monetary, ProductVariety, AOV) had outliers capped at the 90th percentile based on IQR limits (Tenure excluded as it showed no major outliers).
* Data Transformation & Feature Selection:
* InvoiceDate converted to datetime format.
* TotalCost calculated (Quantity \* UnitPrice).
* Customer-level features (Recency, Frequency, Monetary, ProductVariety, Tenure, AOV) engineered via aggregation.
* Skewness in customer-level features addressed using Yeo-Johnson transformation, excluding Tenure.
* Features scaled using Standard Scaler.
* Multicollinearity addressed by calculating VIF and dropping the Monetary feature due to high VIF and correlation with Frequency and AOV.

**5. Exploratory Data Analysis (EDA):**

* A large portion of customers didn’t have an ID, so removed the rows, that has null/missing values.
* The features of the dataset : Quantity, UnitPrice, and feature engineered customer metrics (Recency, Frequency, Monetary, ProductVariety, Tenure, AOV) were heavily right-skewed with numerous outliers, visualized via histograms and boxplots.
* After outlier capping and Yeo-Johnson transformation, distributions became approximately symmetrical and closer to normal.
* Correlation Analysis: A heatmap of scaled customer features
* Moderate positive correlations between Frequency, ProductVariety, and Tenure.
* Moderate negative correlation between Recency and Frequency, ProductVariety, Tenure.
* AOV showed weaker correlations with other metrics after Monetary was removed.
* Multicollinearity: VIF analysis identified high multicollinearity involving Frequency, Monetary, and AOV, leading to the removal of Monetary. Final VIF scores were all below 5.

**6. Approach Taken:**

* Load & Initial Prep: Load data, create a working copy, perform initial inspection.
* Transaction-Level Cleaning: Handle missing Description & CustomerID, aggregate duplicates, remove negative quantities, cap Quantity/UnitPrice outliers.
* Feature Engineering: Calculate TotalCost, group by CustomerID, compute Recency, Frequency, Monetary, ProductVariety, Tenure, AOV.
* Customer-Level Cleaning: Capped outliers in engineered features, transform skewed data using Yeo-Johnson tranformer, scaled features using Standard Scaler.
* Feature Analysis: Analyze correlations and checked multicollinearity (VIF check ) and dropped Monetary.
* K-Means Clustering:
* Determine k value using Elbow Method (WCSS).
* Evaluate cluster separation using Silhouette Scores.
* Fit K-Means k=2 and calculated cluster characteristics using mean on features.
* DBSCAN Clustering:
* Estimate eps using Nearest Neighbors k-distance plot.
* Fit DBSCAN models for selected eps values = 0.6 with min\_samples=10.
* Analyze cluster characteristics, including noise points.

**7. Insights**

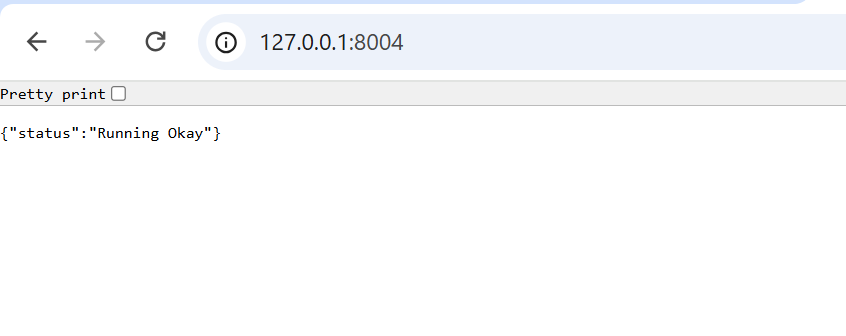
Both K-Means and DBSCAN helped identify clear customer groups:

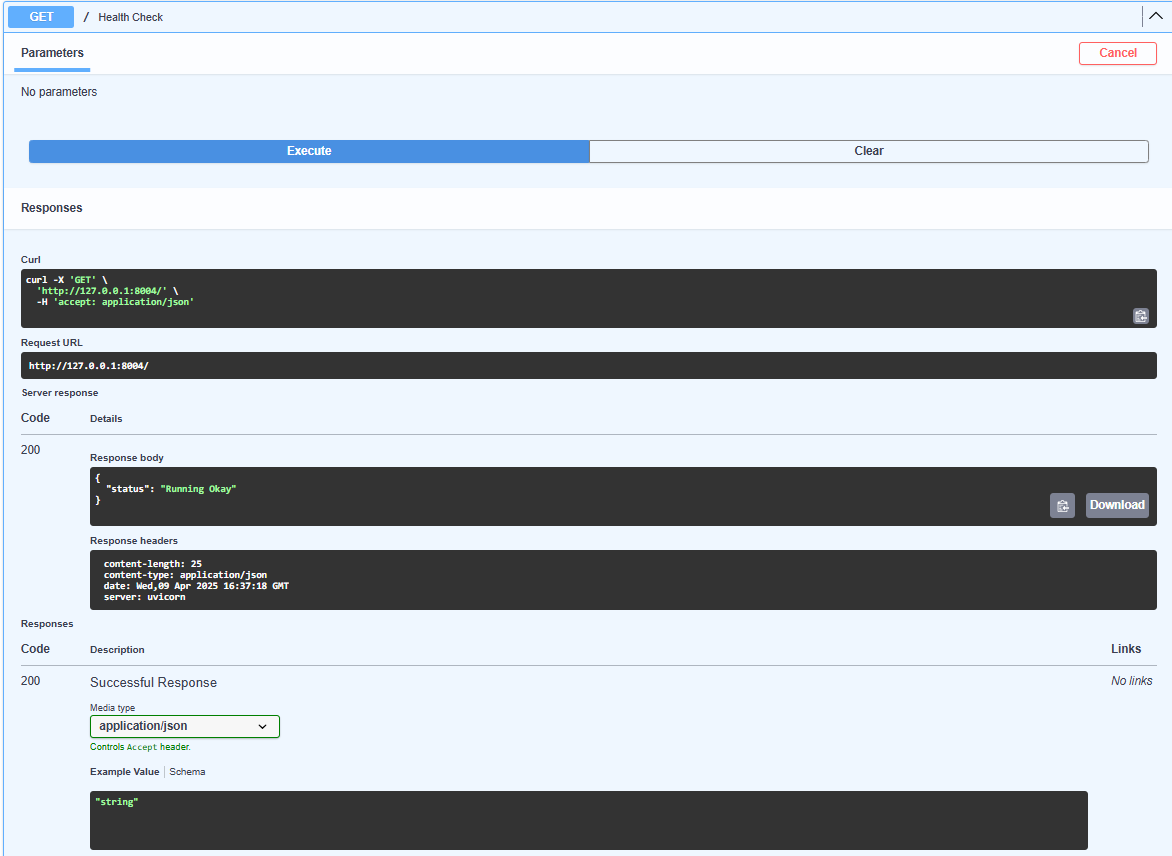
* One group included customers who shop often, spend more, and have been with the store longer shows these are highly engaged customers.
* Another group had people who haven’t shopped in a while, doesn’t buy much, and haven’t been around for long.
* A third group was somewhere in the middle, showing signs of potential growth.
  1. **Business Strategy**
* Reward loyal customers with early access or special discounts to keep them coming back.
* For those slipping away, you could send reminder emails or limited-time offers.
* Customers showing growth potential might benefit from product recommendations or bundled deals.

**9. REST API using Fast API**

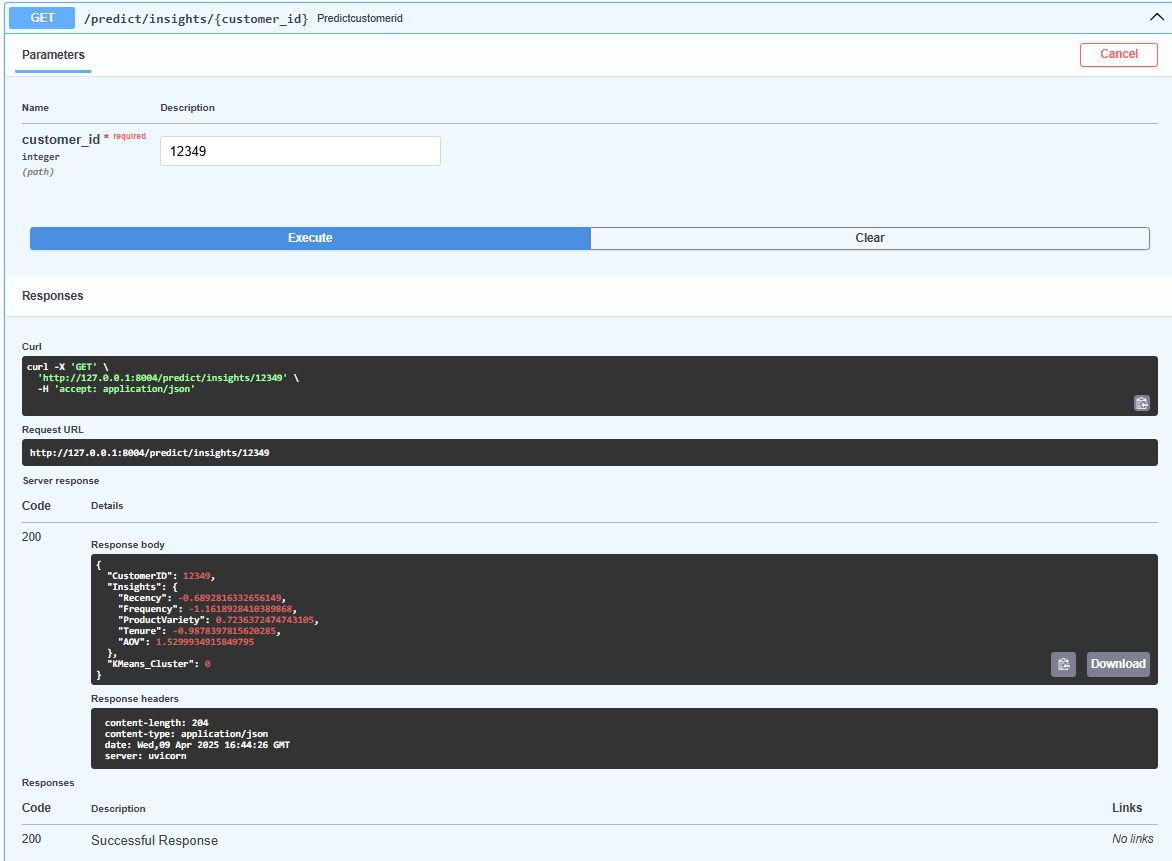
**Steps:**

* Save the trained model to disk using joblib using models/kmeans\_model.pkl for later use by the API.
* Create a copy of the relevant dataset, set CustomerID as index and save the dataset to a file.
* Initialize API by creating a FastAPI instance.
* Load the K-Means model and the csv file that contains the dataset.
* API Endpoints:
  + **Health Check (GET /):** Implementing an endpoint to check if the API is running properly.

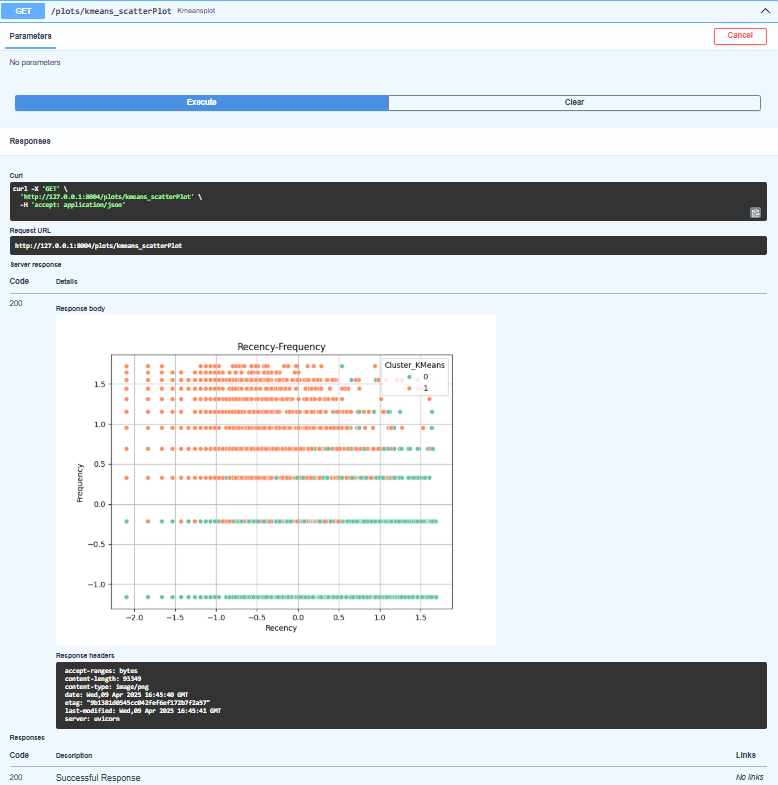




* + **Prediction & Insights (GET /predict/insights/{customer\_id}):**
* Accept a customer\_id as input.
* Check the customer\_id against the loaded data.
* Retrieve the customer's features.
* Use the K-Means model to predict the cluster for that customer.
* Return the predictions and insights as a JSON response.



* + **Plot Generation (GET /plots/kmeans\_scatterPlot):**
* Generate a scatter plot Recency vs. Frequency using the loaded data.
* Save the plot temporarily as an image file.
* Return the image file directly using FileResponse, specifying the correct media type.



**Insights :** The scatter plot displays Recency versus Frequency with dots of K-Means cluster assignment (Cluster 0 = Green, Cluster 1 = Orange).

* Cluster 0 : Customers who bought less recently inactive.
* Cluster 1 : Customers who bought more recently, often more frequently active and loyal.

**Strategy :** Focus on reactivating Cluster 0 and rewarding Cluster 1.

* **Deployment**
* Run the FastAPI application using server Uvicorn to make the use the endpoints over HTTP.

