# BookedBy Task Report

## Data Generation:

In order to create the purchases.csv by running a generate_data.py script. This script generates a synthetic customer purchase dataset with 5,000 records. The dataset includes:

- **Customer IDs**: 500 unique customer IDs are randomly selected from the range 1 to 500, and repeated across 5,000 purchase records.

- **Product IDs**: 50 unique product IDs are randomly selected from the range 1 to 50, and also repeated across the 5,000 records.

- **Product Categories**: Randomly assigned from categories such as 'Electronics', 'Fashion', 'Home Goods', 'Stationary', 'Confectionary', and 'Groceries'.

- **Purchase Amount**: A random value between $10 and $1,000 is generated for each purchase.

- **Purchase Date**: A random date from the past year is selected for each purchase.

## Data Analysis:

### Methodology:

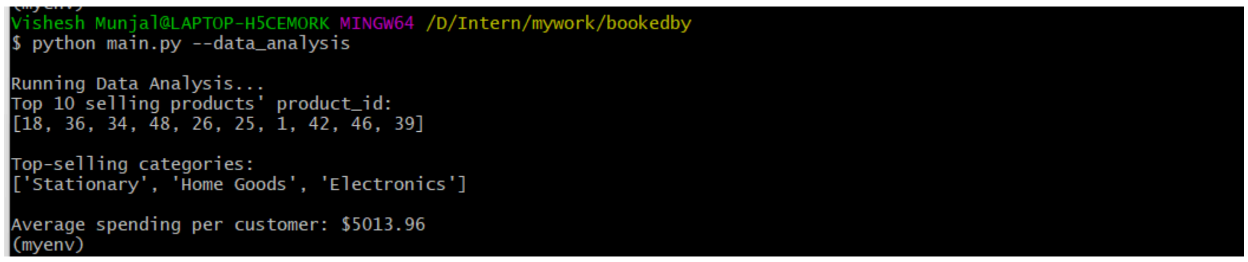
The **DataAnalysis** class performs basic analysis and visualization on customer purchase data. The methodology includes:

1. **Basic Analysis**:
   - Identifies the top 10 best-selling products based on total purchase amount.
   - Identifies the top 3 best-selling product categories.
   - Calculates the average spending per customer.
2. **Visualization**:
   - Aggregates customer data (total spent, purchase frequency, and number of unique categories purchased).
   - Creates a 3D scatter plot to visualize the relationships between total spending, purchase frequency, and the variety of product categories purchased by customers.

This approach helps in understanding customer behavior and identifying trends in purchasing patterns.

**Findings:**

**Basic Insights:**



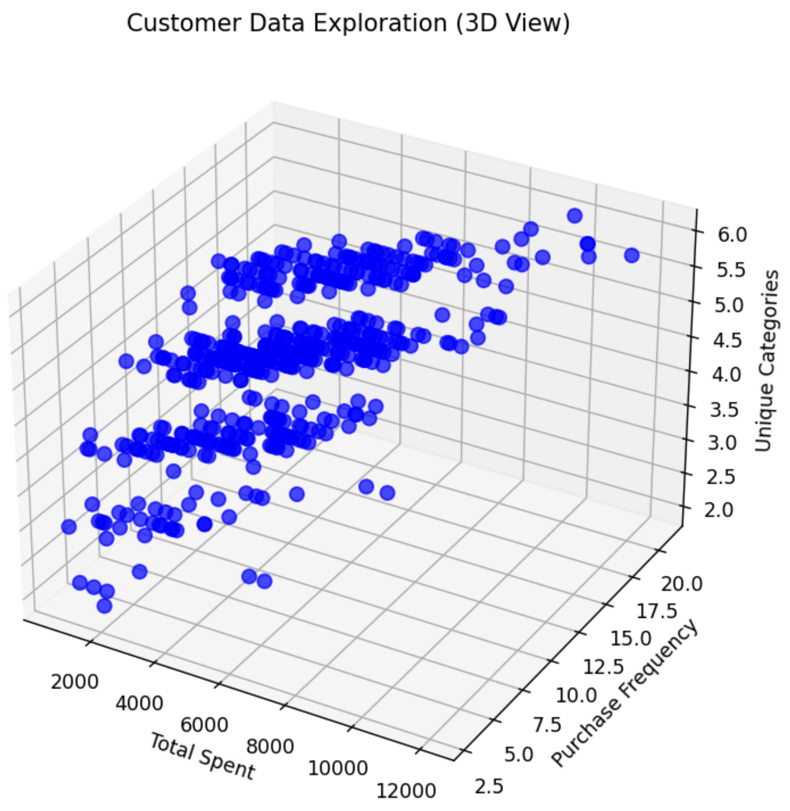
```
(myenv)
Vishesh Munjal@LAPTOP-H5CEMORK MINGW64 /D/Intern/mywork/bookedby
$ python main.py --data_analysis

Running Data Analysis...
Top 10 selling products' product_id:
[18, 36, 34, 48, 26, 25, 1, 42, 46, 39]

Top-selling categories:
['Stationary', 'Home Goods', 'Electronics']

Average spending per customer: $5013.96
(myenv)
```

**Visualization:** Based on the graph, we see that there is a uniform distribution, in terms of categories and frequency of purchases but somewhat skewed for the total amount spent by customers.



## Clustering:

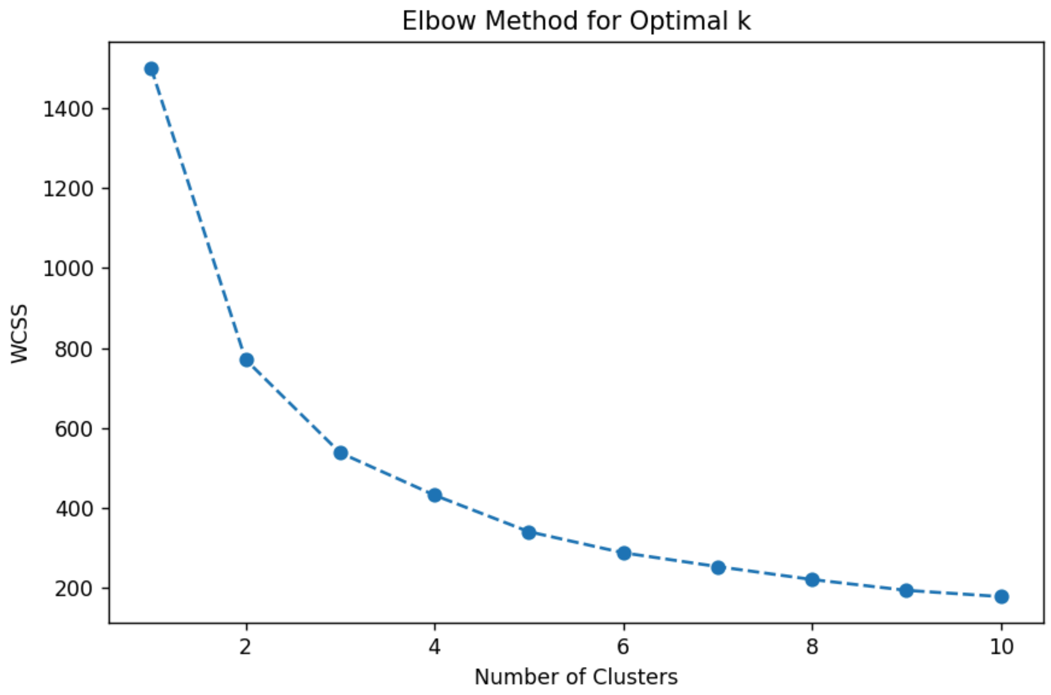
**Methodology:**

The Clustering class created uses K-Means clustering to segment customers based on their purchasing behavior. First, the data is aggregated by customer, capturing total spending, purchase frequency, and the number of unique categories purchased. To determine the optimal number of clusters (k), the WCSS (Within-Cluster Sum of Squares) method is used, where the "elbow" point helps identify the ideal k.

Since spending is often skewed, each parameter is weighted before applying the K-Means algorithm to ensure that spending appropriately influences clustering. The clusters are then visualized in a 3D plot with each segment color-coded for clarity. After clustering, the labels for each segment (e.g., "High Spenders", "Frequent Shoppers") are assigned by analyzing the cluster characteristics, as K-Means alone don't provide meaningful labels. Finally, the customer segments are saved to a CSV file for further analysis.

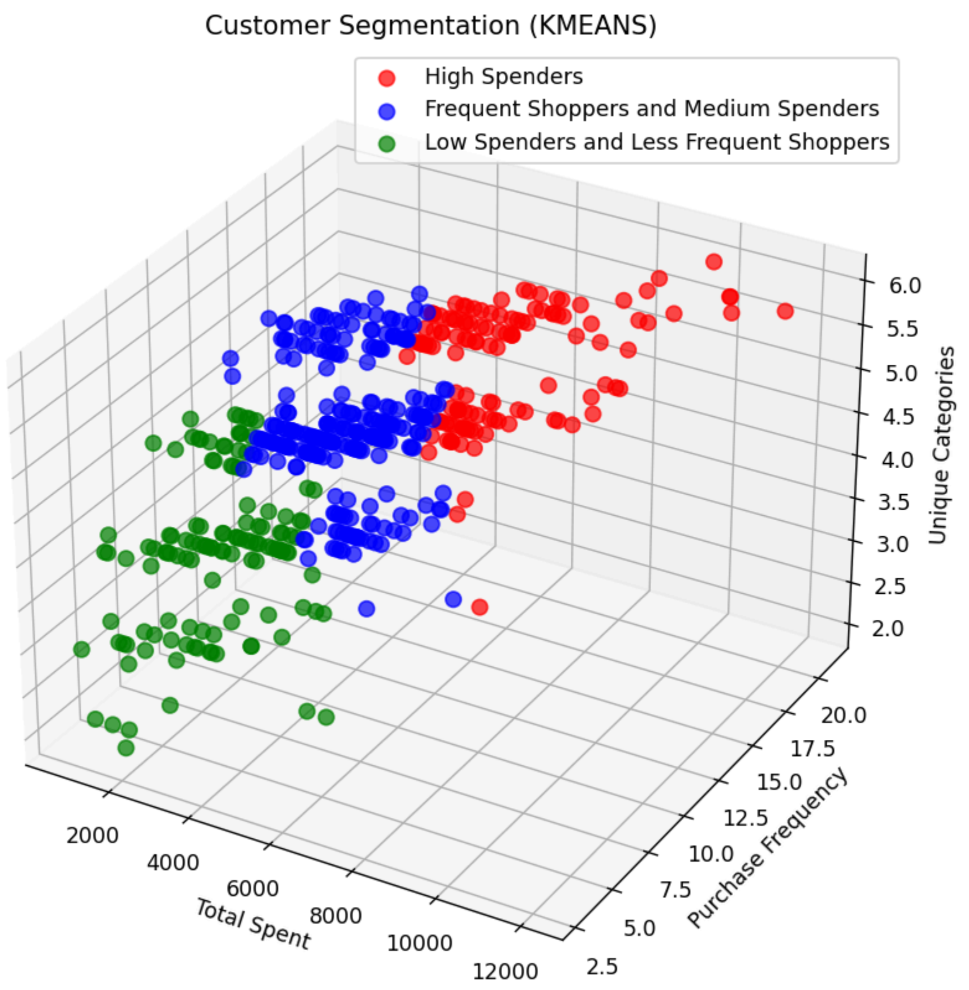
**Findings:**

**WCSS:**



Based on the above graph, I choose the cluster numbers to be 3.

**Kmeans:**



After Kmeans clustering we see that the 3 clusters divide the customers into 3 categories as shown in the graph. It can be noted that certain buyers (customers) usually are higher spenders who buy certain unique categories in terms of product categories(4 and above).

## Recommendations:

### Methodology:

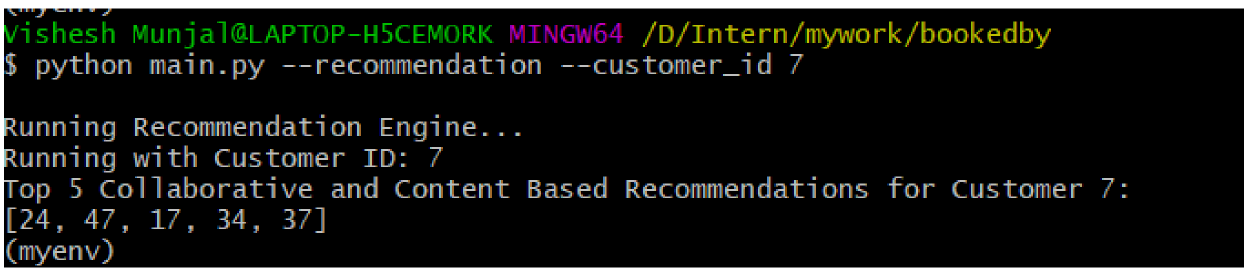
The Recommender class generates product recommendations using a hybrid approach combining Collaborative Filtering (CF) and Content-Based Filtering (CB). Initially, it creates a utility matrix for CF by aggregating purchase data, and normalizes it by subtracting the mean spending per customer. For CB, it encodes product categories using LabelEncoder to facilitate category-based recommendations.

In Collaborative Filtering, recommendations are made by finding similar customers using cosine similarity, and then suggesting products that are popular among those similar customers. Content-Based Filtering recommends products within the same categories that the customer has previously shown interest in.

The Hybrid Recommendation combines CF and CB by weighting their contributions. It filters out products the customer has already purchased, calculates a combined score for each product based on both CF and CB, and returns the top N products with the highest combined score.

This hybrid method allows for more personalized recommendations by blending both customer similarity (CF) and category relevance (CB), with customizable weightings for each method.

**Findings:**



```
Vishesh Munjal@LAPTOP-H5CEMORK MINGW64 /D/Intern/mywork/bookedby
$ python main.py --recommendation --customer_id 7

Running Recommendation Engine...
Running with Customer ID: 7
Top 5 Collaborative and Content Based Recommendations for Customer 7:
[24, 47, 17, 34, 37]
(myenv)
```

The recommendation for customer 7 is shown above, we do collaborative filtering and find the top 10 similar users to customer 7 and find the products they bought, then we do category-based recommendation i.e. content filtering, then calculate the score for all products that are not already bought by customer 7 among the ones we just choose from content and collaborative filtering.
We find a weighted score and sort the products in descending order, then get the top 5 product id’s.