ML Capstone 1 - Part 1 E-Commerce Customer Segmentation

Dataset: Download CSV file from [here (https://drive.google.com/file/d/1Kyi1Akx299BFhdo77T2MmWg7fLRtMaXm/view?usp=sharing)](https://drive.google.com/file/d/1Kyi1Akx299BFhdo77T2MmWg7fLRtMaXm/view?usp=sharing)

Context & Problem statement:

In this project, we delve deep into the thriving sector of online retail by analyzing a transactional dataset from a UK-based retailer, available at the UCI Machine Learning Repository. This dataset documents all transactions between 2010 and 2011. Our primary objective is to amplify the efficiency of marketing strategies and boost sales through customer segmentation. We aim to transform the transactional data into a customer-centric dataset by creating new features that will facilitate the segmentation of customers into distinct groups using the K-means clustering algorithm. This segmentation will allow us to understand the distinct profiles and preferences of different customer groups. Building upon this, we intend to develop a recommendation system that will suggest top-selling products to customers within each segment who haven't purchased those items yet, ultimately enhancing marketing efficacy and fostering increased sales.

Objectives

- Data Cleaning & Transformation: Clean the dataset by handling missing values, duplicates, and outliers, preparing it for effective clustering.
- Feature Engineering: Develop new features based on the transactional data to create a customer-centric dataset, setting the foundation for customer segmentation.
- Data Preprocessing: Undertake feature scaling and dimensionality reduction to streamline the data, enhancing the efficiency of the clustering process.
- Customer Segmentation using K-Means Clustering: Segment customers into distinct groups using K-means, facilitating targeted marketing and personalized strategies.
- Cluster Analysis & Evaluation: Analyze and profile each cluster to develop targeted marketing strategies and assess the quality of the clusters formed.
- Recommendation System: Implement a system to recommend best-selling products to customers within the same cluster who haven't purchased those products, aiming to boost sales and marketing effectiveness.

TODO: Please make use of Python, Pandas, Numpy, Matplotlib and relevant libraries to do the following:

Data Retrieval (1 pt)

- Extracting the dataset from the source (e.g., CSV file)
- Exploring the dataset structure, features
- Understanding the context and significance of each feature

Data preprocessing (2 pts)

- Cleaning the dataset to handle missing values, duplicates, and outliers
- Encoding categorical variables and transforming data types as necessary

Feature Engineering & EDA (3 pts)

- Feature engineering to create new variables(eg Date_since_last_purchase) and do Exploratory Data Analysis (EDA)
- Identifying correlations and patterns in the data
- Make use of 1-d and 2-d explorations to know your data better.

Effective Communication (2 pts)

- Please make use of markdown cells to communicate your thought process, why did you think of performing a step? what was the observation from the visualization? etc.
- Make sure the plots are correctly labelled.
- The code should be commented so that it is readable for the reviewer.

Grading and Important Instructions

- Each of the above steps are mandatory and should be completed in good faith
- Make sure before submitting that the code is in fully working condition
- It is fine to make use of ChatGPT, stackoverflow type resources, just provide the reference links from where you got it
- Debugging is an art, if you find yourself stuck with errors, take help of stackoverflow and ChatGPT to resolve the issue and if it's still unresolved, reach out to me for help.
- You need to score atleast 7/10 to pass the project, anything less than that will be marked required, needing resubmission.

- Feedback will be provided on 3 levels (Awesome, Suggestion, & Required). Required changes are mandatory to be made.

Write your code below and do not delete the above instructions

In [2]:
```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from datetime import datetime
```

In [3]:
```
# Set visualization style
sns.set(style="whitegrid")
```

In [4]:
```
# Data Retrieval
# Load the dataset
data = pd.read_csv('/Users/SaiKiran/downloads/ecommerce_data.csv', encoding='ISO-8859-1')
```

In [5]:
```
# Initial Data Exploration
print("\nDataset Head:\n", data.head())
print("\nDataset Info:\n")
data.info()
print("\nDataset Description:\n", data.describe())
```

```
Dataset Head:
   InvoiceNo StockCode                          Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

      InvoiceDate  UnitPrice  CustomerID         Country
0  12/1/2010 8:26       2.55     17850.0  United Kingdom
1  12/1/2010 8:26       3.39     17850.0  United Kingdom
2  12/1/2010 8:26       2.75     17850.0  United Kingdom
3  12/1/2010 8:26       3.39     17850.0  United Kingdom
4  12/1/2010 8:26       3.39     17850.0  United Kingdom

Dataset Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    541909 non-null  object 
 1   StockCode    541909 non-null  object 
 2   Description  540455 non-null  object 
 3   Quantity     541909 non-null  int64  
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB

Dataset Description:
             Quantity      UnitPrice     CustomerID
count  541909.000000  541909.000000  406829.000000
mean        9.552250       4.611114   15287.690570
std       218.081158      96.759853    1713.600303
min    -80995.000000  -11062.060000   12346.000000
25%         1.000000       1.250000   13953.000000
50%         3.000000       2.080000   15152.000000
75%        10.000000       4.130000   16791.000000
max     80995.000000   38970.000000   18287.000000
```

In [6]:
```
# Data Preprocessing
# Remove duplicates
print(f"\nDuplicates before: {data.duplicated().sum()}")
data.drop_duplicates(inplace=True)
print(f"Duplicates after: {data.duplicated().sum()}")
```

```
Duplicates before: 5268
Duplicates after: 0
```

In [7]:
```
# Handle missing values
print("\nMissing values per column:\n", data.isnull().sum())
data.dropna(subset=['CustomerID'], inplace=True)
```

```
Missing values per column:
 InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135037
Country             0
dtype: int64
```

In [8]:
```
# Remove negative quantities and prices
data = data[(data['Quantity'] > 0) & (data['UnitPrice'] > 0)]
```

In [9]:
```
# Convert InvoiceDate to datetime
data['InvoiceDate'] = pd.to_datetime(data['InvoiceDate'])
```

In [10]:
```
# Feature Engineering
# Create TotalPrice
data['TotalPrice'] = data['Quantity'] * data['UnitPrice']
```

In [11]:
```
# Create new features for RFM Analysis
snapshot_date = data['InvoiceDate'].max() + pd.Timedelta(days=1)
rfm = data.groupby('CustomerID').agg({
    'InvoiceDate': lambda x: (snapshot_date - x.max()).days,
    'InvoiceNo': 'nunique',
    'TotalPrice': 'sum'
})
```

In [12]:
```
rfm.rename(columns={'InvoiceDate': 'Recency',
                    'InvoiceNo': 'Frequency',
                    'TotalPrice': 'Monetary'}, inplace=True)
```

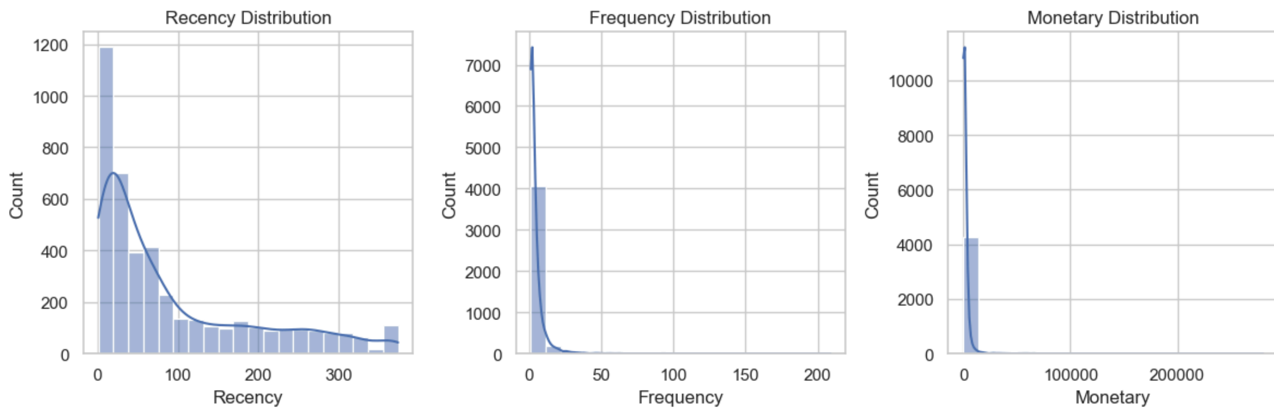
In [13]:
```
# Exploratory Data Analysis (EDA)
plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)
sns.histplot(rfm['Recency'], bins=20, kde=True)
plt.title('Recency Distribution')

plt.subplot(1, 3, 2)
sns.histplot(rfm['Frequency'], bins=20, kde=True)
plt.title('Frequency Distribution')

plt.subplot(1, 3, 3)
sns.histplot(rfm['Monetary'], bins=20, kde=True)
plt.title('Monetary Distribution')

plt.tight_layout()
plt.show()
```



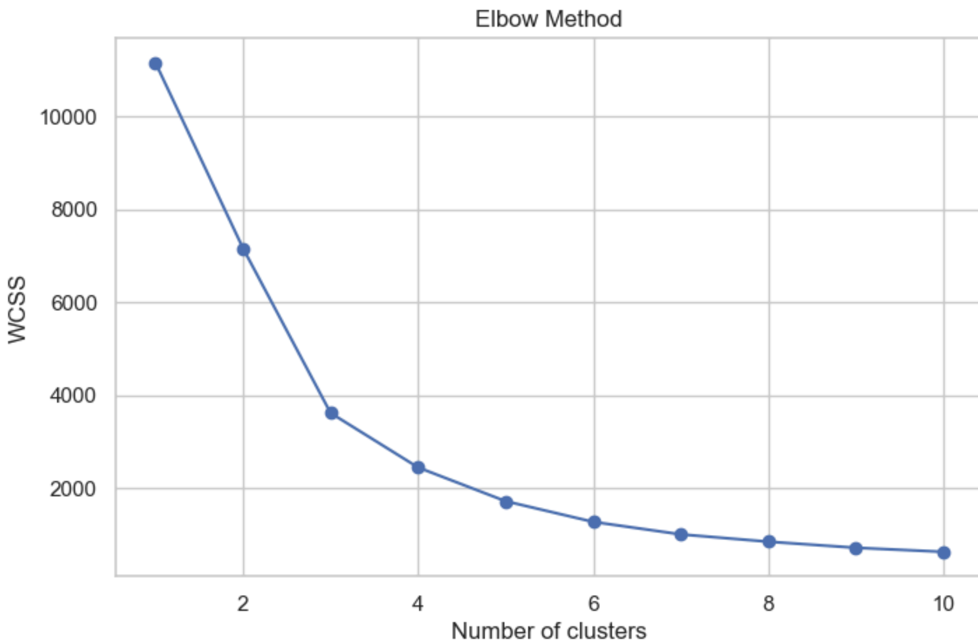
In [14]:
```
# Data Scaling
scaler = StandardScaler()
rfm_scaled = scaler.fit_transform(rfm)
```

In [15]:
```

# Dimensionality Reduction
pca = PCA(n_components=2)
rfm_pca = pca.fit_transform(rfm_scaled)
```

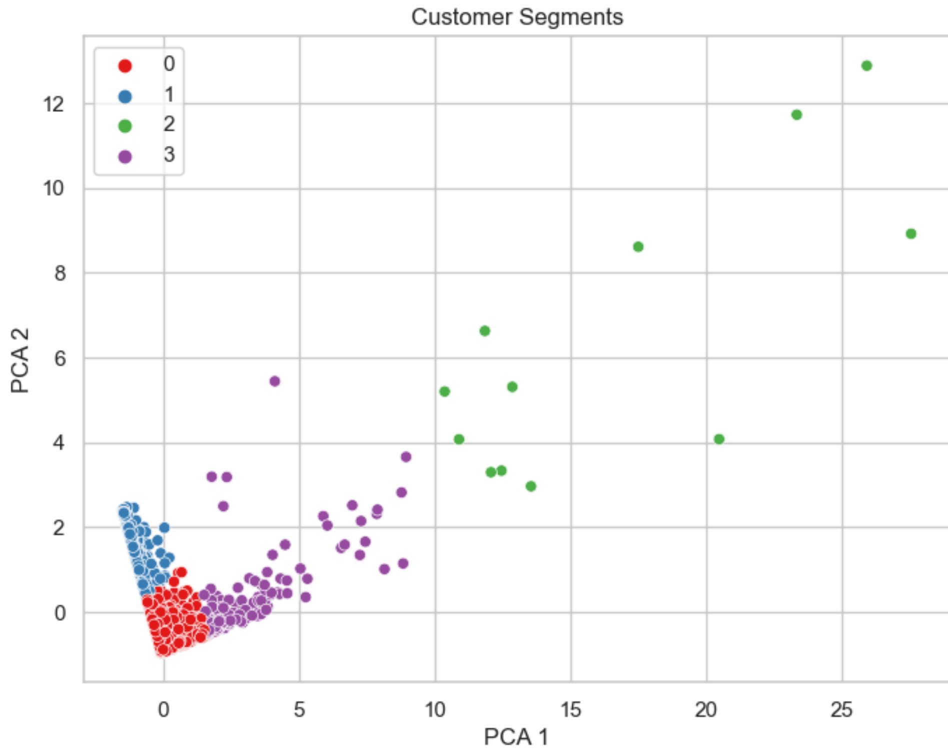
In [19]:
```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42, n_init=10)
    kmeans.fit(rfm_pca)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [20]:
```
# K-Means Clustering (choose k=4 based on elbow method)
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=42, n_init=10)
kmeans.fit(rfm_pca)
rfm['Cluster'] = kmeans.labels_
```

In [21]:
```python
# Visualize clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(x=rfm_pca[:, 0], y=rfm_pca[:, 1], hue=rfm['Cluster'], palette='Set1')
plt.title('Customer Segments')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.legend()
plt.show()
```



In [22]:
```python
# Cluster Profiling
print("\nCluster Summary:\n", rfm.groupby('Cluster').mean())
```

```
Cluster Summary:
             Recency  Frequency       Monetary
Cluster
0         43.287520   3.818152    1385.470325
1        247.756075   1.542056     477.436898
2          7.750000  86.833333  130198.424167
3         15.725146  23.748538   14852.598655
```

In [23]:
```python
# Recommendation System
# Find top 10 best-selling products
top_products = data.groupby('StockCode')['Quantity'].sum().sort_values(ascending=False).head(10)
```

In [24]:
```python
# Create a recommendation dataframe
recommendations = {}
```

In [25]:
```python
for cluster in rfm['Cluster'].unique():
    customers_in_cluster = rfm[rfm['Cluster'] == cluster].index
    purchases = data[data['CustomerID'].isin(customers_in_cluster)]
    bought_products = purchases.groupby('CustomerID')['StockCode'].apply(set)

    for customer in customers_in_cluster:
        not_bought = set(top_products.index) - bought_products.get(customer, set())
        recommendations[customer] = list(not_bought)
```

In [26]:
```
# Show sample recommendations
sample_recommendations = dict(list(recommendations.items())[:5])
print("\nSample Recommendations:\n", sample_recommendations)
```

```
Sample Recommendations:
 {12346.0: ['85123A', '85099B', '84077', '22492', '21212', '84879', '23084', '22197', '23843'], 12437.0: ['85
123A', '85099B', '23166', '84879', '23843'], 12471.0: ['85123A', '85099B', '22492', '84077', '23166', '8487
9', '22197', '23843'], 12474.0: ['85123A', '85099B', '84077', '21212', '23166', '84879', '22197', '23843'], 1
2540.0: ['85099B', '22492', '23084', '23166', '23843']}
```

In [27]:
```
#Summary of findings

Removed 5,268 duplicate entries from the dataset

Dropped rows with missing CustomerID (about 135k rows)

Filtered out negative quantities and prices to keep only valid transactions

Created RFM features: Recency (days since last purchase), Frequency (number of purchases), and Monetary (total

Segmented customers into 4 clusters using KMeans after applying PCA

Cluster 0: moderately active customers with decent spending

Cluster 1: inactive customers with low frequency and low spend

Cluster 2: highly active and valuable customers (very frequent and high spenders)

Cluster 3: recent buyers with decent frequency and strong monetary value

Visualizations showed skewed distributions, especially in Monetary

Identified top 10 most purchased products overall

Built a simple recommendation system that suggests top products a customer hasn’t bought yet

Generated sample recommendations for a few customers based on what they haven’t purchased from the top product
```