1. Introduction

I applied K-means clustering to identify patterns in customer behavior, ultimately aiding in targeted marketing strategies and business decision-making.

---

2. Data Preprocessing

Before clustering, I took the following steps:

- **Concatenate Both Datasets**: The excel file had datasets for both 2009-1020 and 2010-2011. So, I used `pd.concat` to join both into one dataframe.
- **Handling Missing Values**: Approximately 107,927 Customer ID values were missing, and these rows were removed.

```
data1 = pd.read_csv("2009_2010_retail.csv")
data1 = data1.dropna()
data2 = pd.read_csv("2010_2011_retail.csv")
dataframes = [data1, data2]
data3 = pd.concat(dataframes)
print(data3.shape)
```

- **Feature Engineering**: I derived key features such as:
    - **Total Spending** = Sum of all purchases per customer.
    - **Total Orders** = Count of unique invoices per customer.(number of orders they've made)
    - **Total Quantity** = Sum of all items purchased per customer.
    - **Average Order Value** = Total Spending / Total Orders.(Amount spent per order made)
    - **Average Items Per Order** = Total Quantity / Total Orders. (Average items per order)
    - **Distinct Items Purchased** = Number of unique products bought by a customer.
    - **Recency** = Days since the last purchase.

```
data3['Customer ID'] = data3['Customer ID'].astype(str)
data3['InvoiceData'] = pd.to_datetime(data3['InvoiceDate'])
data3 = data3[data3['Quantity'] >0 ]
data = data3.groupby('Customer ID').agg(
    Total_Spending=('Price', lambda x: (x * data3.loc[x.index,
'Quantity']).sum()),
    Total_Orders=('Invoice', 'nunique'),
    Total_Quantity=('Quantity', 'sum'),
    Distinct_Items=('StockCode', 'nunique'),
    First_Purchase=('InvoiceDate', 'min'),
    Last_Purchase=('InvoiceDate', 'max')
).reset_index()

latest_purchase = data3['InvoiceDate'].max()



data['Last_Purchase'] = pd.to_datetime(data['Last_Purchase'],
errors='coerce')
data['First_Purchase'] = pd.to_datetime(data['First_Purchase'],
errors='coerce')
latest_purchase = pd.to_datetime(latest_purchase, errors='coerce')

data['Recency'] = (latest_purchase - data['Last_Purchase']).dt.days
data.drop(columns=['First_Purchase', 'Last_Purchase'], inplace=True)
data['Avg_Order_Value'] = data['Total_Spending'] / data['Total_Orders']
data['Avg_Items_Per_Order'] = data['Total_Quantity'] /
data['Total_Orders']
```

- **Standardization**: Since the dataset contained numerical variables on different scales, I applied **StandardScaler, Log Values and Winsorization** to normalize the data. Winsorization was used to remove outliers in the 80th percentile.

```
data_log = data.copy()
data_log['Total_Spending'] = np.log1p(data_log['Total_Spending'])
data_log['Total_Quantity'] = np.log1p(data_log['Total_Quantity'])
data_log['Distinct_Items'] = np.log1p(data_log['Distinct_Items'])
data_log['Avg_Order_Value'] = np.log1p(data_log['Avg_Order_Value'])
data_log['Avg_Items_Per_Order'] =
np.log1p(data_log['Avg_Items_Per_Order'])
data_log['Total_Orders'] = np.log1p(data_log['Total_Orders'])
data_log['Recency'] = np.log1p(data_log['Recency'])
cols_to_winsorize = ['Total_Spending', 'Total_Orders', 'Total_Quantity',
                      'Distinct_Items', 'Avg_Items_Per_Order',
'Avg_Order_Value']

data_log['Recency'] = data_log['Recency'].replace(-np.inf, np.nan)

# Ensure all columns are numeric
for col in cols_to_winsorize:
    data_log[col] = pd.to_numeric(data_log[col], errors='coerce')

# Winsorize: Cap values at the 80th percentile
for col in cols_to_winsorize:
    q8 = data_log[col].quantile(0.80)
    data_log[col] = np.where(data_log[col] > q8, q8, data_log[col])

x = data_log[['Total_Spending', 'Total_Orders', 'Total_Quantity',
                          'Avg_Order_Value', 'Avg_Items_Per_Order',
'Distinct_Items', 'Recency']]

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(x)
```

3. Clustering Methodology

I used **K-Means Clustering** to segment the customers. The process included:

- **Choosing the Optimal K**:
  - The **Elbow Method** was used to determine the best number of clusters.
  - Based on the analysis, I selected **K = 2** as the optimal number of clusters.
- **Applying K-Means**: The algorithm was implemented on the PCA-transformed data, ensuring efficiency and interpretability.

```
wcss = []

for i in range(1,10):
    kmeans = KMeans(i)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

wcss

plt.plot(range(1,10), wcss)
plt.xlabel("number of clusters")
plt.ylabel('wcss')



kmeans = KMeans(n_clusters=2, random_state=42)
data_log['Cluster'] = kmeans.fit_predict(X_scaled)

identified_clusters = kmeans.fit_predict(X_scaled)
identified_clusters

data_with_clusters = data_log.copy()
data_with_clusters['Cluster'] = identified_clusters
data_with_clusters
```

---

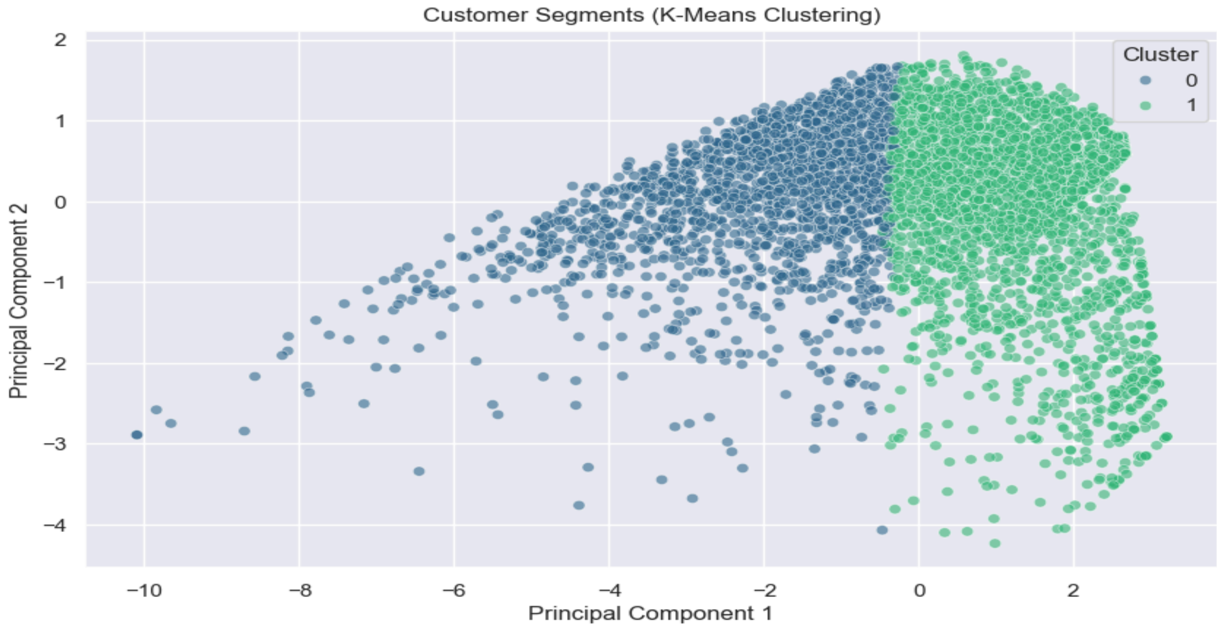
4. Results & Analysis

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA

# Reduce to 2D for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Add PCA components to the DataFrame
data_with_clusters['PCA1'] = X_pca[:, 0]
data_with_clusters['PCA2'] = X_pca[:, 1]

# Scatter plot of clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(x='PCA1', y='PCA2', hue='Cluster', palette='viridis',
data=data_with_clusters, alpha=0.6)
plt.title('Customer Segments (K-Means Clustering)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title="Cluster")
plt.show()
```




```
data_with_clusters['Cluster'] = identified_clusters

cluster_summary =
data_with_clusters.groupby("Cluster").mean(numeric_only=True)
print(cluster_summary)
```

Cluster Characteristics

Metric	Cluster 0	Cluster 1
Total Spending	6.25	8.37
Total Orders	0.93	1.74
Total Quantity	4.78	6.77
Distinct Items	2.63	4.26
Recency	5.57 days	4.99 days
Avg Order Value	5.84	6.75
Avg Items Per Order	4.39	5.19

Key Insights

- **Cluster 1** customers tend to purchase more items and spend more per order.
- **Cluster 0** customers exhibit lower spending
- Customers in Cluster 1 made purchases more recently (lower Recency value), indicating more engagement with the store.
- The business can target Cluster 1 customers with loyalty programs, rewards and others to encourage them to keep coming, while Cluster 0 customers may require re-engagement strategies, discounts, sales and targeted ads to make them spend more.

---

4. Conclusion & Business Recommendations

- **Target High-Value Customers (Cluster 1)**: Implement exclusive promotions and personalized discounts.
- **Re-engage Low-Activity Customers (Cluster 0)**: Utilize email marketing, limited-time offers, and reminders to encourage purchases.
- **Monitor Customer Behavior**: Regularly update clustering models to reflect changing consumer habits.
