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', 'Country').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()

data['Country'] = data['Country'].astype('category').cat.codes

country_mapping =
dict(enumerate(data3['Country'].astype('category').cat.categories))
print(country_mapping)


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['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 MinMaxScaler
from sklearn.decomposition import PCA


scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(x[features])

pca = PCA(n_components=2)
x_pca = pca.fit_transform(X_scaled)

X_scaled_ = pd.DataFrame(x_pca, columns=['PCA1', 'PCA2'])
```

---

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 = 6** 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(n_clusters=i, random_state=42, n_init=10)
    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=6, random_state=42, n_init=10)
identified_clusters = kmeans.fit_predict(X_scaled_)
x['Cluster'] = identified_clusters
X_scaled_['Cluster'] = identified_clusters
```

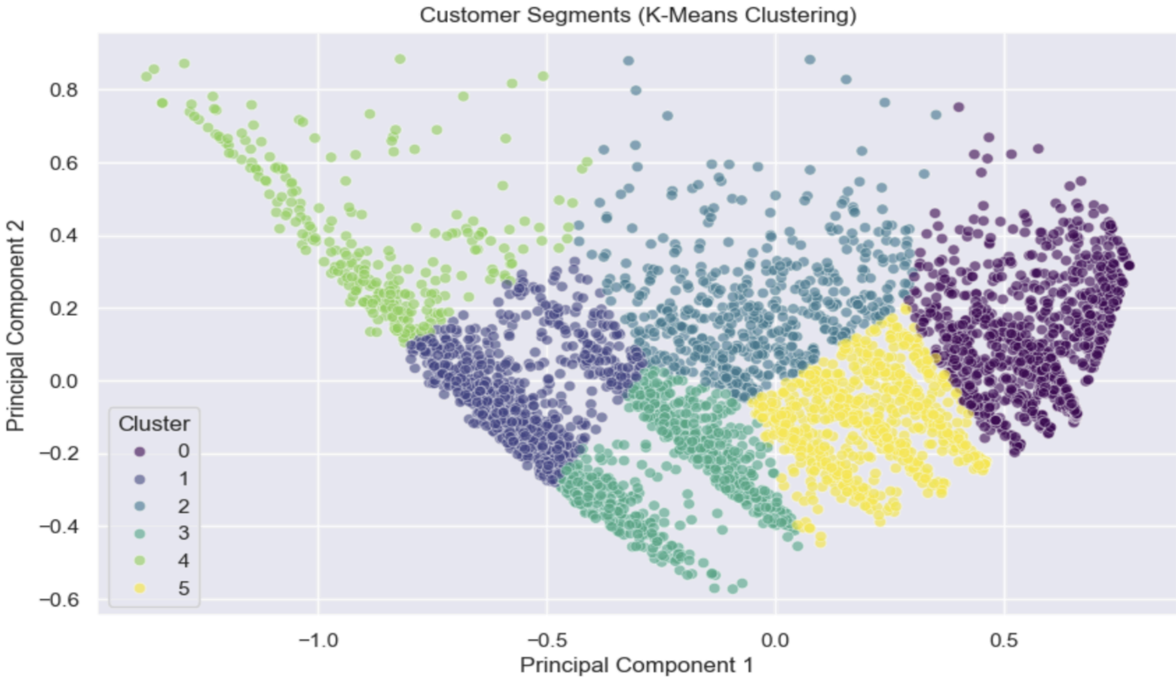
---

4. Results & Analysis

```
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.scatterplot(
    x='PCA1', y='PCA2', hue='Cluster', palette='viridis',
    data=X_scaled_, 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()
```




```
pca_components = pd.DataFrame(pca.components_, columns=features,
index=['PCA1', 'PCA2'])
print(pca_components)
```

```
       Total_Spending  Total_Orders  Total_Quantity  Avg_Order_Value  \
PCA1         0.261082      0.739095        0.350267         0.113122
PCA2        -0.115810      0.491368       -0.237805        -0.256477

       Avg_Items_Per_Order  Distinct_Items   Recency
PCA1              0.166701        0.454306 -0.126127
PCA2             -0.481551       -0.433906 -0.450164
```

```
cluster_summary = x.groupby("Cluster")[features].mean()
print(cluster_summary)
```

```
         Total_Spending  Total_Orders  Total_Quantity  Avg_Order_Value  \
Cluster
0              8.800782      2.131775        7.211839         6.730780
1              6.181943      0.766413        4.794264         6.017247
2              7.246448      1.512477        5.689206         5.977972
3              7.357215      0.937040        5.846686         6.737338
4              5.117665      0.769881        3.092582         4.948367
5              8.122883      1.559927        6.602010         6.717955

         Avg_Items_Per_Order  Distinct_Items   Recency
Cluster
0                   5.178742        4.569234  4.656341
1                   4.647576        2.630371  5.664856
2                   4.430807        3.086962  4.649159
3                   5.262523        3.665384  5.733197
4                   2.977382        1.350471  5.546614
5                   5.203265        4.222889  5.482236
```

Key Insights:

Cluster 0: "Frequent Big Spenders"

High Spending, High Orders, High Variety

Low Recency (very recent customers)

Cluster 1: "Occasional Buyers"

Average Spendings, Low Orders, Low Variety(Low Distinct Items)

High Recency (older customers)

Cluster 2: "Balanced or Regular Buyers"

Average Spendings, Average Orders, Average Variety(Average Distinct Items)

Low Recency (Recentcustomers)

Cluster 3: "Loyal but Rare Buyers"

Average Spendings, Low Orders, Average Variety(Average Distinct Items)

High Recency (Old customers)

spend well when they buy

Cluster 4: "Inactive Customers"

Low Spending, Low Orders, Very Low Variety

Highest Recency (Oldest customers)

Cluster 5: "High-Value Customers"

High Spending, Average Orders, High Variety

High Recency (not very recent customers)

Spend a lot, buy different items, but not as often as Cluster 0.

4. Conclusion & Business Recommendations

- **Target High-Value Customers (Cluster 0 and 5)**: Implement exclusive promotions and personalized discounts.
- **Re-engage Low-Activity Customers (Cluster 1 & 3)**: Utilize email marketing, limited-time offers, and reminders to encourage purchases.
- **Reactivate Inactive Customers(Cluster 4)**: free shipping offers, or surveys to understand why they left.
- **Nurture Average Customers (Cluster 2)**: Offer a subscription model or loyalty rewards to increase their frequency.