Customer Segmentation with K-Means

Import Libraries

```
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
from tabulate import tabulate
from prettytable import PrettyTable
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

Load Dataset

```
df = pd.read_csv(r'C:\Users\Saba\Documents\Semester - 04\Itauma\Directories\Machine_Learning\
```

```
df.shape
```

```
(3900, 19)
```

The dataset contains 3,900 records with 19 columns.

Display Columns and data types

```
# Get columns and their data types
columns = df.columns
data_types = df.dtypes

# Create a DataFrame for better formatting
```

```python
summary_df = pd.DataFrame({'Column Name': columns, 'Data Type': data_types})

# Print the summary in a beautiful table format
print(tabulate(summary_df, headers='keys', tablefmt='psql', showindex=False))
```

```
+--------------------------+-------------+
| Column Name              | Data Type   |
|--------------------------+-------------|
| Customer ID              | int64       |
| Age                      | int64       |
| Gender                   | object      |
| Item Purchased           | object      |
| Category                 | object      |
| Purchase Amount (USD)    | int64       |
| Location                 | object      |
| Size                     | object      |
| Color                    | object      |
| Season                   | object      |
| Review Rating            | float64     |
| Subscription Status      | object      |
| Payment Method           | object      |
| Shipping Type            | object      |
| Discount Applied         | object      |
| Promo Code Used          | object      |
| Previous Purchases       | int64       |
| Preferred Payment Method | object      |
| Frequency of Purchases   | object      |
+--------------------------+-------------+
```

Below are the key features:

- **Customer ID**: Unique identifier for each customer.
- **Age**: Customer's age.
- **Gender**: Customer's gender.
- **Item Purchased**: Type of item bought.
- **Category**: Product category (e.g., Clothing, Footwear).
- **Purchase Amount (USD)**: Total amount spent.
- **Location**: Customer's location.

- **Previous Purchases**: Number of prior purchases.
- **Frequency of Purchases**: Purchase frequency (e.g., Weekly, Fortnightly, Annually).

```
df.describe()
```

	Customer ID	Age	Purchase Amount (USD)	Review Rating	Previous Purchases
count	3900.000000	3900.000000	3900.000000	3900.000000	3900.000000
mean	1950.500000	44.068462	59.764359	3.749949	25.351538
std	1125.977353	15.207589	23.685392	0.716223	14.447125
min	1.000000	18.000000	20.000000	2.500000	1.000000
25%	975.750000	31.000000	39.000000	3.100000	13.000000
50%	1950.500000	44.000000	60.000000	3.700000	25.000000
75%	2925.250000	57.000000	81.000000	4.400000	38.000000
max	3900.000000	70.000000	100.000000	5.000000	50.000000

- The data describes a customer base of **3900** individuals with an average age of **44** and an age range from **18 to 70**.
- The **Review Rating** ranges from 2.5 to 5, with an average of about 3.75, suggesting that customers generally rate their experience positively but with some room for improvement.
- The average number of **Previous Purchases** per customer is approximately 25, indicating frequent shopping behavior among the customers.
- **Count** indicates the number of non-null entries in each column. For instance, there are 3900 entries for Customer ID, Age, Review Rating, and Previous Purchases.
- **Mean** is the average value for each column. For example, the average Age of customers is approximately 44.07 years, while the average Review Rating is about 3.75.
- **Std** (Standard Deviation)measures the amount of variation or dispersion of a set of values. A higher standard deviation indicates that the values are spread out over a wider range. For example, the standard deviation of Age is about 15.21, indicating that ages vary significantly from the mean.
- **Min** is the minimum value in each column. For example, the youngest customer is 18 years old.
- **25% (First Quartile)** is the value below which 25% of the data falls. For Age, 25% of the customers are 31 years old or younger.
- **50% (Median)** is the middle value when the data is sorted. For Age, the median is 44 years, meaning half of the customers are older than this age.

- **75% (Third Quartile)** is the value below which 75% of the data falls. For example, 75% of customers are 57 years old or younger.
- **Max** is the maximum value in each column. For Age, the oldest customer is 70 years old.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3900 entries, 0 to 3899
Data columns (total 19 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Customer ID               3900 non-null   int64
 1   Age                       3900 non-null   int64
 2   Gender                    3900 non-null   object
 3   Item Purchased            3900 non-null   object
 4   Category                  3900 non-null   object
 5   Purchase Amount (USD)     3900 non-null   int64
 6   Location                  3900 non-null   object
 7   Size                      3900 non-null   object
 8   Color                     3900 non-null   object
 9   Season                    3900 non-null   object
 10  Review Rating             3900 non-null   float64
 11  Subscription Status       3900 non-null   object
 12  Payment Method            3900 non-null   object
 13  Shipping Type             3900 non-null   object
 14  Discount Applied          3900 non-null   object
 15  Promo Code Used           3900 non-null   object
 16  Previous Purchases        3900 non-null   int64
 17  Preferred Payment Method  3900 non-null   object
 18  Frequency of Purchases    3900 non-null   object
dtypes: float64(1), int64(4), object(14)
memory usage: 579.0+ KB
```

- **Column** lists each column in the DataFrame, along with its index number (from 0 to 18).
- **Non-Null Count** indicates how many non-null (non-missing) values are present in each column. In this case, all columns have 3900 non-null entries, meaning there are no missing values in the dataset.
- Each column has an associated data type:

- **int64** is the Integer type, used for numeric data. Columns like Customer ID, Age, Purchase Amount (USD), and Previous Purchases are of this type.
- **float64** is the Floating-point number, used for decimal values. In this case, Review Rating is a float.
- **object** is typically used for text or mixed data types. Many columns (like Gender, Item Purchased, Category, etc.) are of this type, indicating they contain categorical data.

1. Data Preprocessing

Encode categorical variable:

```
# Encode categorical variable if needed (e.g., 'Frequency of Purchases')
freq_encoder = LabelEncoder()
df['Frequency of Purchases'] = freq_encoder.fit_transform(df['Frequency of Purchases'])
```

Drop missing values

```
# Drop any rows with missing values
df.dropna(inplace=True)
```

Identify missing values:

```
df.isnull().sum()
```

```
Customer ID               0
Age                       0
Gender                    0
Item Purchased            0
Category                  0
Purchase Amount (USD)     0
Location                  0
Size                      0
Color                     0
Season                    0
Review Rating             0
Subscription Status       0
Payment Method            0
Shipping Type             0
```

```
Discount Applied           0
Promo Code Used            0
Previous Purchases         0
Preferred Payment Method   0
Frequency of Purchases     0
dtype: int64
```

All columns show 0, indicating that there are no missing values in any of the columns of the DataFrame. So we dont have to handle any null values in this analysis.

To see the first five rows.

```
df.head()
```

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	S
0	1	55	Male	Blouse	Clothing	53	Kentucky	L
1	2	19	Male	Sweater	Clothing	64	Maine	L
2	3	50	Male	Jeans	Clothing	73	Massachusetts	S
3	4	21	Male	Sandals	Footwear	90	Rhode Island	M
4	5	45	Male	Blouse	Clothing	49	Oregon	M

Select features for clustering

```
# Select features for clustering
features = ['Age', 'Purchase Amount (USD)', 'Previous Purchases', 'Frequency of Purchases']
X = df[features]
```

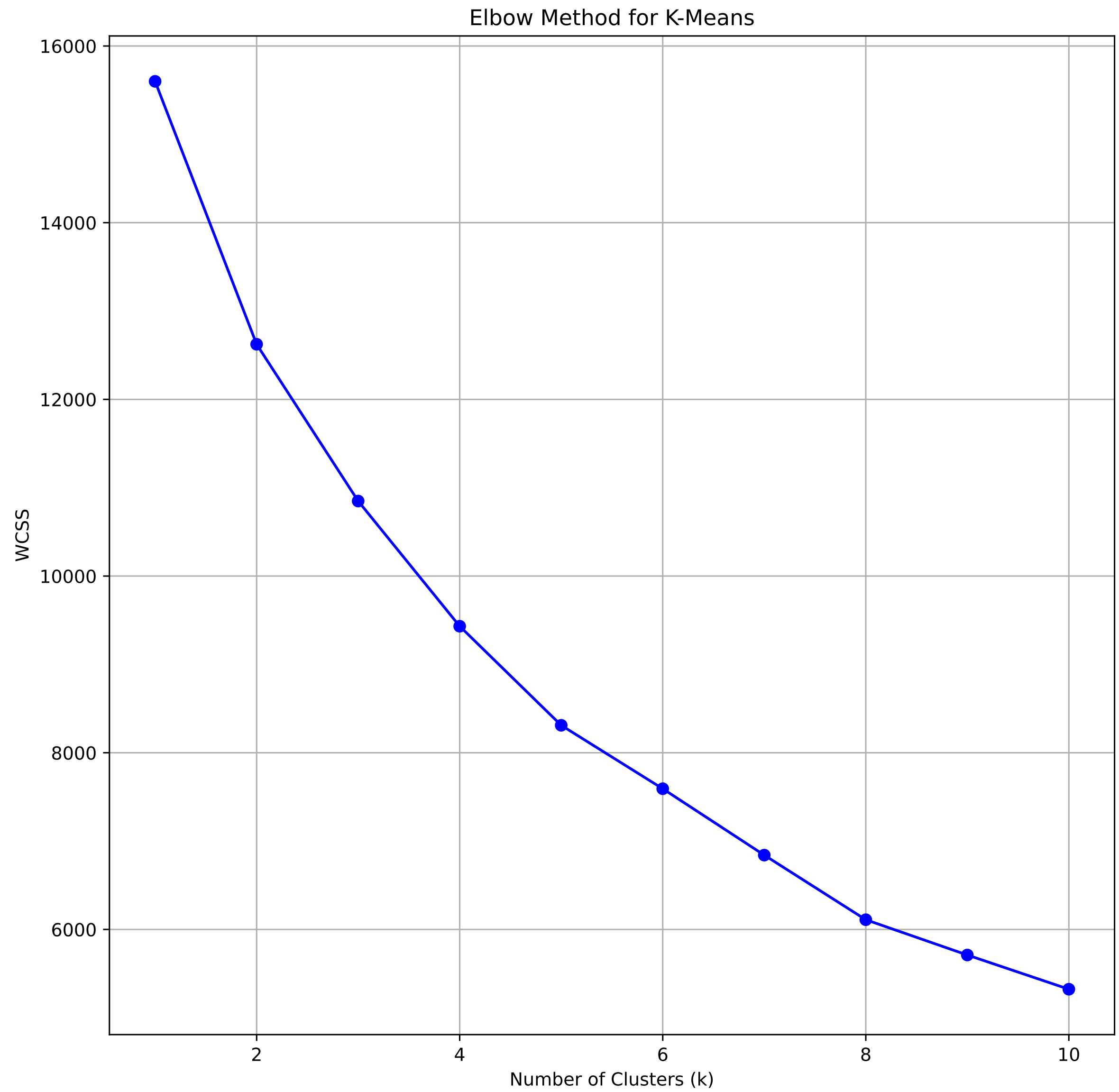
DataFrame X is typically used in clustering algorithms, such as K-Means, DBSCAN, or Hierarchical Clustering, to group similar customers based on their characteristics.

Standardize the features

```
# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

2. K-Means Clustering *(Jain, 2010)*

```
# Elbow method to determine the optimal number of clusters for K-Means
wcss = []  # Within-cluster sum of squares
for k in range(1, 11):  # Trying different values of k
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

# Plot the Elbow curve
plt.figure(figsize=(10, 10))
plt.plot(range(1, 11), wcss, marker='o', color='b')
plt.title('Elbow Method for K-Means')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()

# Optimal k from Elbow plot (for example, let's assume it's 3)
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans_labels = kmeans.fit_predict(X_scaled)

# Adding K-Means cluster labels to the DataFrame
df['KMeans_Cluster'] = kmeans_labels

# Evaluating K-Means Clustering with silhouette score
kmeans_silhouette = silhouette_score(X_scaled, kmeans_labels)
print(f"K-Means Silhouette Score: {kmeans_silhouette:.4f}")
```




```
K-Means Silhouette Score: 0.1758
```

The plot shows a downward trend in WCSS as the number of clusters increases. This indicates that as you increase the number of clusters, the WCSS decreases, meaning the clusters become more compact.

The elbow appears to be around k=3 or k=4, this is where the reduction in WCSS starts to diminish significantly.

k=3 or k=4 is the optimal number of clusters. Choosing a higher number of clusters example k=5 to 10 results in only a small reduction in WCSS, indicating that the additional clusters do not provide substantial improvements in clustering quality.

3. Applying Clustering

DBSCAN Clustering

```
# Applying DBSCAN clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)  # Adjust eps and min_samples as needed
dbscan_labels = dbscan.fit_predict(X_scaled)

# Adding DBSCAN cluster labels to the DataFrame
df['DBSCAN_Cluster'] = dbscan_labels

# Evaluating DBSCAN clustering with silhouette score (ignoring noise points: label -1)
dbscan_silhouette = silhouette_score(X_scaled[df['DBSCAN_Cluster'] != -1], dbscan_labels[df[
print(f"DBSCAN Silhouette Score (excluding noise): {dbscan_silhouette:.4f}")
```

```
DBSCAN Silhouette Score (excluding noise): -0.1691
```

Hierarchical Clustering

```
## Hierarchical Clustering ##
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score

# Applying Agglomerative (Hierarchical) clustering
hierarchical = AgglomerativeClustering(n_clusters=optimal_k, metric='euclidean', linkage='wa
hierarchical_labels = hierarchical.fit_predict(X_scaled)

# Adding Hierarchical cluster labels to the DataFrame
df['Hierarchical_Cluster'] = hierarchical_labels

# Evaluating Hierarchical Clustering with silhouette score
hierarchical_silhouette = silhouette_score(X_scaled, hierarchical_labels)
print(f"Hierarchical Clustering Silhouette Score: {hierarchical_silhouette:.4f}")
```

```
Hierarchical Clustering Silhouette Score: 0.1287
```

A Silhouette Score of 0.1287 is relatively low, suggesting that the clusters formed by the hierarchical clustering algorithm are not well-separated.This score indicates that there is a significant overlap between the clusters, and the samples within clusters are not as distinct from samples in other clusters as one might desire. Since the score is positive, it implies that, on average, the samples are assigned to the correct clusters, but the degree of separation is weak.

A Silhouette Score of 0.1287 for hierarchical clustering indicates that the clusters are not well-defined. So we explored alternative clustering methods to improve the clustering quality and achieve better segment separation.

PCA for Visualization

```
print(X_scaled.shape)
print(X_scaled[:5])  # Show the first 5 rows
```

```
(3900, 4)
[[ 0.71891344 -0.28562864 -0.78583067  0.01257477]
 [-1.64862924  0.17885219 -1.61655226  0.01257477]
 [ 0.39008807  0.55888195 -0.16278948  1.51384863]
 [-1.51709909  1.27671595  1.63710729  1.51384863]
 [ 0.0612627  -0.45453076  0.39102491 -1.48869909]]
```

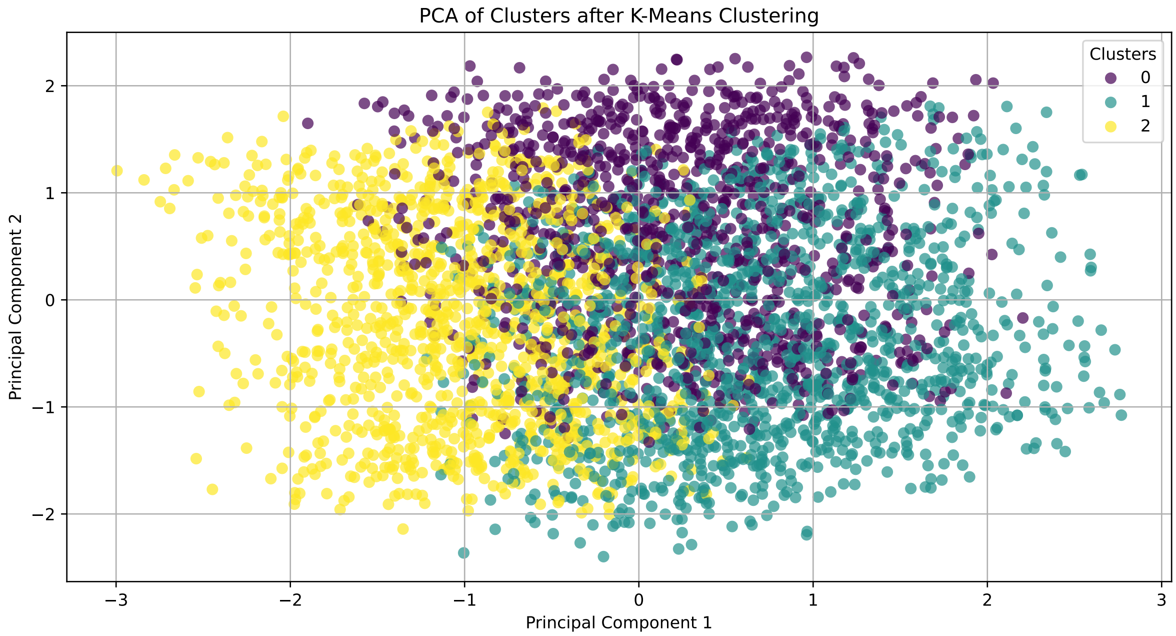
```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

print(X_pca.shape)
print(X_pca[:5])  # Show the first 5 rows of PCA output
```

```
(3900, 2)
[[-0.05809075 -0.47854117]
 [-1.91591641 -0.48884786]
 [ 0.85250637  0.17897073]
 [ 0.85227551  1.31006259]
 [-0.46909905  0.01835147]]
```

```
df['KMeans_Cluster'] = kmeans_labels

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

```python
## PCA for Visualization ##
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

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

# Check if PCA columns are added correctly
print(df[['PCA1', 'PCA2']].head())

# Scatter plot of the PCA components colored by K-Means clusters
plt.figure(figsize=(12, 6))
scatter = plt.scatter(df['PCA1'], df['PCA2'], c=df['KMeans_Cluster'], cmap='viridis', alpha=
plt.title('PCA of Clusters after K-Means Clustering')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')

# Create a legend
legend1 = plt.legend(*scatter.legend_elements(), title="Clusters")

# Show grid
plt.grid(True)

# Display the plot
plt.show()
```

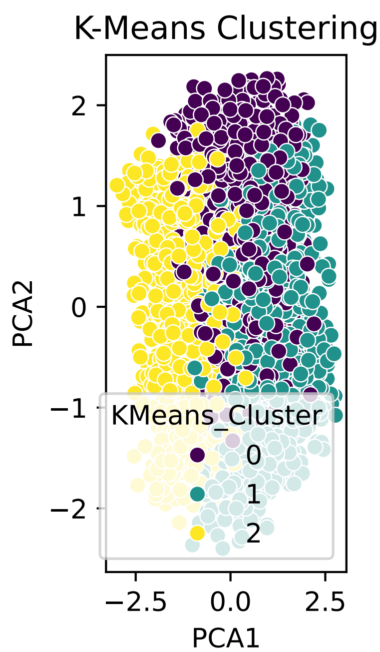
```
       PCA1      PCA2
0 -0.058091 -0.478541
1 -1.915916 -0.488848
2  0.852506  0.178971
3  0.852276  1.310063
4 -0.469099  0.018351
```



K-Means

```
# K-Means
plt.subplot(1, 3, 1)
sns.scatterplot(x='PCA1', y='PCA2', hue='KMeans_Cluster', data=df, palette='viridis', legend=
plt.title('K-Means Clustering')
```

```
Text(0.5, 1.0, 'K-Means Clustering')
```



DBSCAN

```
from sklearn.cluster import DBSCAN

# Applying DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)  # Adjust eps and min_samples as needed
df['DBSCAN_Cluster'] = dbscan.fit_predict(X_scaled)

# Check the unique values in DBSCAN_Cluster
print(df['DBSCAN_Cluster'].unique())
```

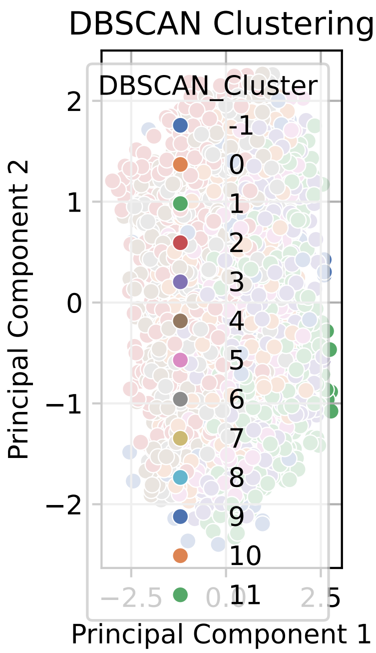
```
[ 0  1  2  3  4  5  6 -1  7  9  8 11 10]
```

```
print(df.head())  # Check the first few rows to see if DBSCAN_Cluster exists
```

```
   Customer ID  Age Gender Item Purchased  Category  Purchase Amount (USD)  \
0            1   55   Male         Blouse  Clothing                     53
1            2   19   Male        Sweater  Clothing                     64
2            3   50   Male          Jeans  Clothing                     73
3            4   21   Male        Sandals  Footwear                     90
4            5   45   Male         Blouse  Clothing                     49
```

```
        Location Size      Color  Season  ...  Discount Applied  \
0       Kentucky    L       Gray  Winter  ...               Yes
1          Maine    L     Maroon  Winter  ...               Yes
2  Massachusetts    S     Maroon  Spring  ...               Yes
3   Rhode Island    M     Maroon  Spring  ...               Yes
4         Oregon    M  Turquoise  Spring  ...               Yes

  Promo Code Used Previous Purchases Preferred Payment Method  \
0             Yes                 14                    Venmo
1             Yes                  2                     Cash
2             Yes                 23              Credit Card
3             Yes                 49                   PayPal
4             Yes                 31                   PayPal

  Frequency of Purchases KMeans_Cluster  DBSCAN_Cluster Hierarchical_Cluster  \
0                      3              2               0                    0
1                      3              2               0                    0
2                      6              1               1                    0
3                      6              1               1                    0
4                      0              0               2                    1

       PCA1      PCA2
0 -0.058091 -0.478541
1 -1.915916 -0.488848
2  0.852506  0.178971
3  0.852276  1.310063
4 -0.469099  0.018351

[5 rows x 24 columns]
```

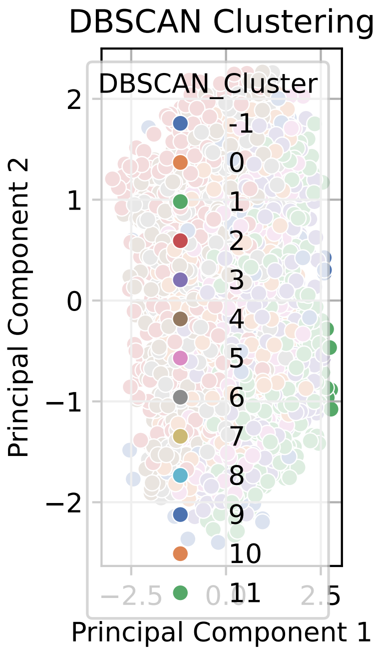
```
plt.subplot(1, 3, 2)
sns.scatterplot(x='PCA1', y='PCA2', hue='DBSCAN_Cluster', data=df, palette='deep', legend='fu
plt.title('DBSCAN Clustering')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid(True)
```




```
from sklearn.cluster import DBSCAN
import seaborn as sns
import matplotlib.pyplot as plt

# Applying DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)  # Adjust as needed
df['DBSCAN_Cluster'] = dbscan.fit_predict(X_scaled)

# Check if DBSCAN_Cluster column exists
print(df.head())  # Check the DataFrame

# Plotting DBSCAN Clusters
plt.subplot(1, 3, 2)
sns.scatterplot(x='PCA1', y='PCA2', hue='DBSCAN_Cluster', data=df, palette='deep', legend='fu
plt.title('DBSCAN Clustering')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid(True)
plt.show()
```

```
   Customer ID  Age Gender Item Purchased  Category  Purchase Amount (USD)  \
0            1   55   Male         Blouse  Clothing                     53
```

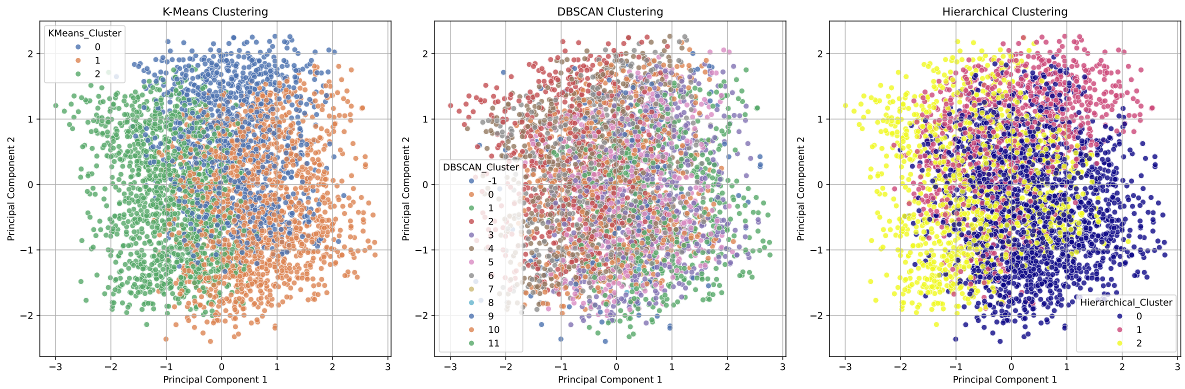
```
1           2   19   Male        Sweater  Clothing                      64
2           3   50   Male          Jeans  Clothing                      73
3           4   21   Male        Sandals  Footwear                      90
4           5   45   Male         Blouse  Clothing                      49

       Location Size      Color  Season  ... Discount Applied  \
0      Kentucky    L       Gray  Winter  ...              Yes
1         Maine    L     Maroon  Winter  ...              Yes
2 Massachusetts    S     Maroon  Spring  ...              Yes
3  Rhode Island    M     Maroon  Spring  ...              Yes
4        Oregon    M  Turquoise  Spring  ...              Yes

  Promo Code Used Previous Purchases Preferred Payment Method  \
0             Yes                 14                    Venmo
1             Yes                  2                     Cash
2             Yes                 23              Credit Card
3             Yes                 49                   PayPal
4             Yes                 31                   PayPal

  Frequency of Purchases KMeans_Cluster  DBSCAN_Cluster Hierarchical_Cluster  \
0                      3              2               0                    0
1                      3              2               0                    0
2                      6              1               1                    0
3                      6              1               1                    0
4                      0              0               2                    1

       PCA1      PCA2
0 -0.058091 -0.478541
1 -1.915916 -0.488848
2  0.852506  0.178971
3  0.852276  1.310063
4 -0.469099  0.018351

[5 rows x 24 columns]
```



Hierarchical Clustering

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

# Increase the figure size
plt.figure(figsize=(18, 6))  # Change the size as needed

# First subplot for K-Means
plt.subplot(1, 3, 1)
sns.scatterplot(x='PCA1', y='PCA2', hue='KMeans_Cluster', data=df, palette='deep', legend='fu
plt.title('K-Means Clustering')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid(True)

# Second subplot for DBSCAN
plt.subplot(1, 3, 2)
sns.scatterplot(x='PCA1', y='PCA2', hue='DBSCAN_Cluster', data=df, palette='deep', legend='fu
plt.title('DBSCAN Clustering')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
```

```
plt.grid(True)

# Third subplot for Hierarchical Clustering
plt.subplot(1, 3, 3)
sns.scatterplot(x='PCA1', y='PCA2', hue='Hierarchical_Cluster', data=df, palette='plasma', le
plt.title('Hierarchical Clustering')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid(True)

# Adjusting layout to prevent overlapping
plt.subplots_adjust(wspace=0.3)  # Adjust horizontal space between subplots

# Show the plots
plt.tight_layout()
plt.show()
```



4. Analyze Results

Insights Interpretation

```
# Select numeric columns and perform clustering analysis based on the mean for each cluster
numeric_cols = df.select_dtypes(include='number')
cluster_analysis_numeric = numeric_cols.groupby(df['KMeans_Cluster']).mean()

# Display the numeric cluster analysis result
print(cluster_analysis_numeric)

# For categorical columns, calculate the mode for each cluster
```

```python
categorical_cols = df.select_dtypes(include='object')
cluster_analysis_categorical = categorical_cols.groupby(df['KMeans_Cluster']).agg(lambda x: x

# Display the categorical cluster analysis result
print(cluster_analysis_categorical)
```

```
                Customer ID        Age  Purchase Amount (USD)  Review Rating  \
KMeans_Cluster
0               1903.842059  43.904887              64.691099       3.759948
1               1946.875085  45.983707              56.039375       3.739919
2               1996.409055  42.012490              59.640125       3.752537

                Previous Purchases  Frequency of Purchases  KMeans_Cluster  \
KMeans_Cluster
0                        39.035777                1.781850             0.0
1                        26.124915                5.038697             1.0
2                        12.220141                1.669009             2.0

                DBSCAN_Cluster  Hierarchical_Cluster      PCA1      PCA2
KMeans_Cluster
0                     3.144852              1.335079  0.230915  0.627143
1                     2.525458              0.107264  0.688521 -0.326989
2                     3.220921              1.188134 -0.998298 -0.185052
               Gender Item Purchased  Category   Location Size  Color  Season  \
KMeans_Cluster
0                Male        Jewelry  Clothing    Alabama    M   Cyan  Winter
1                Male          Pants  Clothing  Louisiana    M  Black  Spring
2                Male           Belt  Clothing   New York    M  Olive    Fall

               Subscription Status Payment Method  Shipping Type  \
KMeans_Cluster
0                               No           Cash        Express
1                               No    Credit Card  Free Shipping
2                               No    Credit Card   Next Day Air

               Discount Applied Promo Code Used Preferred Payment Method
KMeans_Cluster
0                            No              No                   PayPal
1                            No              No                     Cash
2                            No              No                   PayPal
```

Analyse Clusters

1. K-Means Cluster Analysis:

```
# Grouping by KMeans Cluster and calculating mean for each feature
kmeans_analysis = df.groupby('KMeans_Cluster')[features].mean()
print("K-Means Cluster Analysis:\n", kmeans_analysis)
```

```
K-Means Cluster Analysis:
                      Age  Purchase Amount (USD)  Previous Purchases  \
KMeans_Cluster
0               43.904887              64.691099           39.035777
1               45.983707              56.039375           26.124915
2               42.012490              59.640125           12.220141

                Frequency of Purchases
KMeans_Cluster
0                             1.781850
1                             5.038697
2                             1.669009
```

Cluster 0 (Budget-Conscious Shoppers):

Younger age group, lower purchase amounts, high frequency of purchases.

Implement frequent promotional campaigns and loyalty discounts to encourage repeat purchases. Use platforms like Instagram and TikTok for targeted ads, showcasing budget-friendly products. Send personalized emails with special offers based on past purchase behavior.

Cluster 1 (High-Value Customers):

Older age group, high purchase amounts, fewer purchases.

Create exclusive loyalty programs that offer rewards, discounts, or VIP experiences. Use personalized recommendations and high-end product offerings to appeal to their purchasing behavior. Implement a dedicated customer service approach to enhance loyalty.

Cluster 2 (Occasional Shoppers):

Mixed age group, average purchase amounts, low frequency of purchases.

Design campaigns to win back inactive customers, such as limited-time offers or special events. Create engaging content that highlights product benefits and trends to stimulate interest. Use retargeting strategies based on previous interactions with the brand to increase conversion rates.

2. DBSCAN Cluster Analysis:

```
# Grouping by DBSCAN Cluster and calculating mean for each feature, ignoring noise (-1)
dbscan_analysis = df[df['DBSCAN_Cluster'] != -1].groupby('DBSCAN_Cluster')[features].mean()
print("DBSCAN Cluster Analysis:\n", dbscan_analysis)
```

```
DBSCAN Cluster Analysis:
                        Age  Purchase Amount (USD)  Previous Purchases  \
DBSCAN_Cluster
0               43.577220              58.409266           25.208494
1               44.326214              58.751456           25.920388
2               44.912656              60.534759           24.677362
3               44.425373              59.110075           27.628731
4               43.027933              60.890130           24.653631
5               44.707930              59.396518           25.044487
6               42.950178              60.443060           25.024911
7               24.500000              23.000000           36.000000
8               68.714286              24.142857           15.571429
9               21.000000              49.333333           46.666667
10              25.000000              98.333333           42.000000
11              63.750000              90.250000            3.125000

                Frequency of Purchases
DBSCAN_Cluster
0                                  3.0
1                                  6.0
2                                  0.0
3                                  5.0
4                                  1.0
5                                  4.0
6                                  2.0
7                                  4.0
8                                  2.0
```

```
9                                   4.0
10                                  3.0
11                                  5.0
```

3. Hierarchical Cluster Analysis:

```
# Grouping by Hierarchical Cluster and calculating mean for each feature
hierarchical_analysis = df.groupby('Hierarchical_Cluster')[features].mean()
print("Hierarchical Cluster Analysis:\n", hierarchical_analysis)
```

```
Hierarchical Cluster Analysis:
                            Age  Purchase Amount (USD)  Previous Purchases  \
Hierarchical_Cluster
0                     43.358201              55.665387           24.612178
1                     56.538136              74.294492           24.621822
2                     34.821712              54.253310           27.149162

                      Frequency of Purchases
Hierarchical_Cluster
0                                   4.564454
1                                   1.872881
2                                   1.335393
```

5. Suggestion for marketing strategies *(Kotler & Keller, 2016)*:

- Launch tailored marketing campaigns based on the identified strategies for each segment.
- Utilize different marketing channels (email, social media, etc.) based on the segment's characteristics.
- Track key performance indicators (KPIs) for each campaign, such as conversion rates, customer retention, and engagement levels.
- Adjust strategies as necessary based on feedback and performance data.

Conclusion:

By understanding the characteristics of each customer segment, businesses can create targeted marketing strategies that resonate with their specific needs and behaviors. This approach will not only enhance customer satisfaction but also increases the effectiveness of marketing efforts, ultimately leading to improved business outcomes.

References:

1. **Reference on Clustering Techniques:**

   Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8), 651-666. https://doi.org/10.1016/j.patrec.2009.09.011

2. **Reference on Marketing Strategies:**

   Kotler, P., & Keller, K. L. (2016). *Marketing management* (15th ed.). Pearson.