USE CUSTOMER DATASET

In [2]:
```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

In [3]:
```
df=pd.read_csv(r"C:\Users\prajapath Arjun\Downloads\archive (1)\shopping_tr
df
```

Out[3]:

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Co
0	1	55	Male	Blouse	Clothing	53	Kentucky	L	G
1	2	19	Male	Sweater	Clothing	64	Maine	L	Mar
2	3	50	Male	Jeans	Clothing	73	Massachusetts	S	Mar
3	4	21	Male	Sandals	Footwear	90	Rhode Island	M	Mar
4	5	45	Male	Blouse	Clothing	49	Oregon	M	Turquo
...	...	...	...	...	...	...	...	...	
3895	3896	40	Female	Hoodie	Clothing	28	Virginia	L	Turquo
3896	3897	52	Female	Backpack	Accessories	49	Iowa	L	W
3897	3898	46	Female	Belt	Accessories	33	New Jersey	L	Gr
3898	3899	44	Female	Shoes	Footwear	77	Minnesota	S	Bro
3899	3900	52	Female	Handbag	Accessories	81	California	M	Be

3900 rows × 19 columns

In [4]:
```
df.head()
```

Out[4]:

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

In [5]:
```
df.tail()
```

Out[5]:

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color
3895	3896	40	Female	Hoodie	Clothing	28	Virginia	L	Turquoise
3896	3897	52	Female	Backpack	Accessories	49	Iowa	L	White
3897	3898	46	Female	Belt	Accessories	33	New Jersey	L	Green
3898	3899	44	Female	Shoes	Footwear	77	Minnesota	S	Brown
3899	3900	52	Female	Handbag	Accessories	81	California	M	Beige

In [6]:
```
df.shape
```

Out[6]:
```
(3900, 19)
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
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
```

In [8]:
```
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
```

Data Preprocessing

In [7]:
```
# Data Preprocessing
# Handling categorical variables using Label Encoding
label_cols = ['Gender', 'Category', 'Location', 'Color', 'Season', 'Payment
df_encoded = df.copy()
```

In [8]:
```
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score, calinsk
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
# Apply Label Encoding for categorical features
for col in label_cols:
    le = LabelEncoder()
    df_encoded[col] = le.fit_transform(df[col])
```

In [9]:
```
# Drop columns that are not relevant or already processed
df_encoded = df_encoded.drop(['Item Purchased', 'Promo Code Used'], axis=1)
```

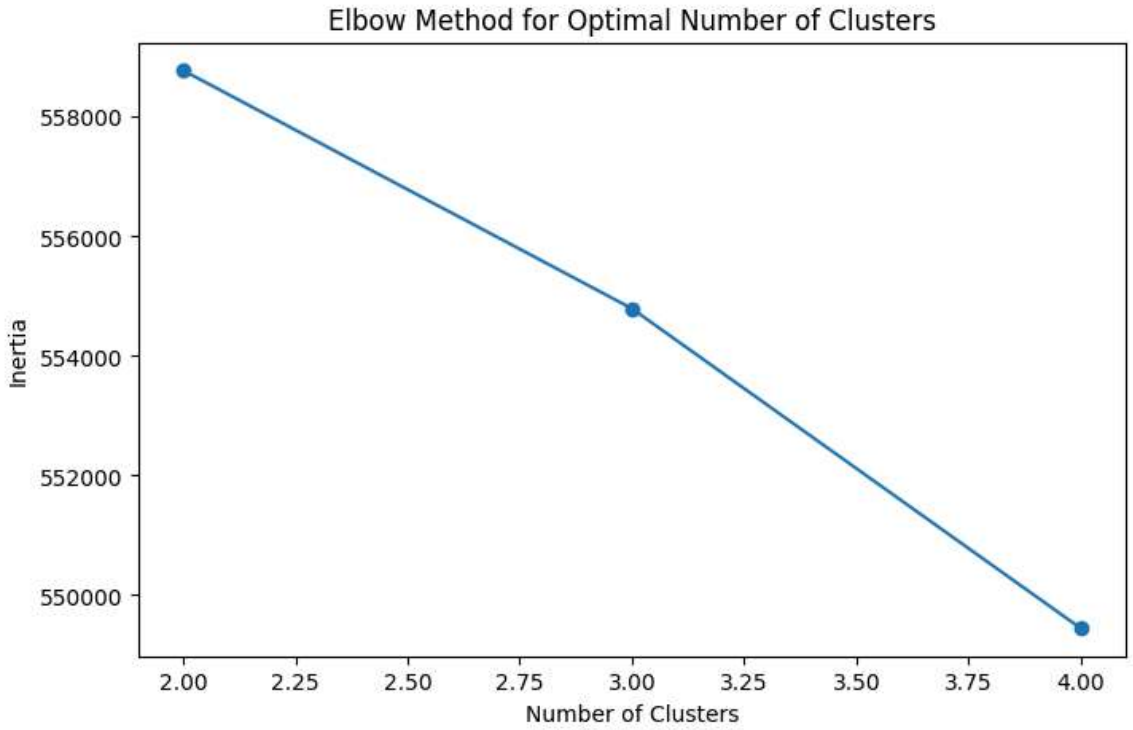
In [10]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Assuming 'df' is your dataframe
# Drop 'Customer ID' as it's not useful for clustering
df_cleaned = df.drop('Customer ID', axis=1)

# One-hot encode categorical columns (like 'Gender', 'Subscription Status',
df_encoded = pd.get_dummies(df_cleaned)

# Scale the data (important for K-means)
scaler = StandardScaler()
data_scaled = scaler.fit_transform(df_encoded)

# Check the shape of the scaled data
print(data_scaled.shape)
```

```
(3900, 149)
```

Determining Number of Clusters

In [11]:
```
# Determine the optimal number of clusters using the Elbow Method
inertia = []
K = range(2, 5)  # Adjusted to avoid clusters greater than the number of sa
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init='auto')  # Set n_
    kmeans.fit(data_scaled)
    inertia.append(kmeans.inertia_)
```

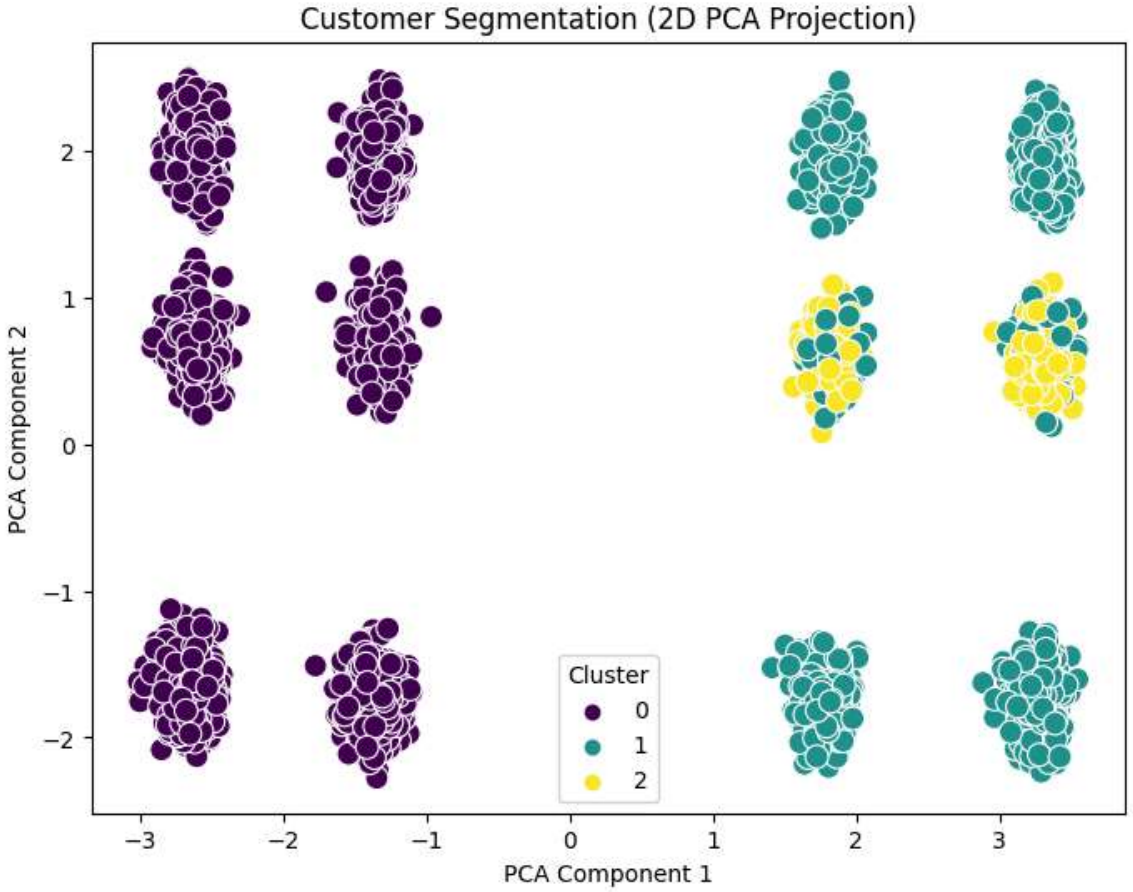
In [12]:
```
# Plot the Elbow Curve
plt.figure(figsize=(8, 5))
plt.plot(K, inertia, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.show()
```



In [13]:
```
# Choosing an optimal number of clusters (from the elbow curve)
optimal_k = 3  # Adjust as needed based on the elbow plot

# Apply KMeans clustering with the chosen number of clusters
kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init='auto')  # Se
clusters = kmeans.fit_predict(data_scaled)
```

In [14]:
```
# Add the clusters to the original dataset
df['Cluster'] = clusters
```

In [15]:
```
# Visualize clusters using PCA for dimensionality reduction
pca = PCA(n_components=2)
data_pca = pca.fit_transform(data_scaled)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=data_pca[:, 0], y=data_pca[:, 1], hue=df['Cluster'], pale
plt.title('Customer Segmentation (2D PCA Projection)')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.show()
```



In [16]:
```
# 7. Interpret Results
# Compute statistics only for numeric columns
numeric_columns = df_encoded.select_dtypes(include=[np.number]).columns
```

In [17]:

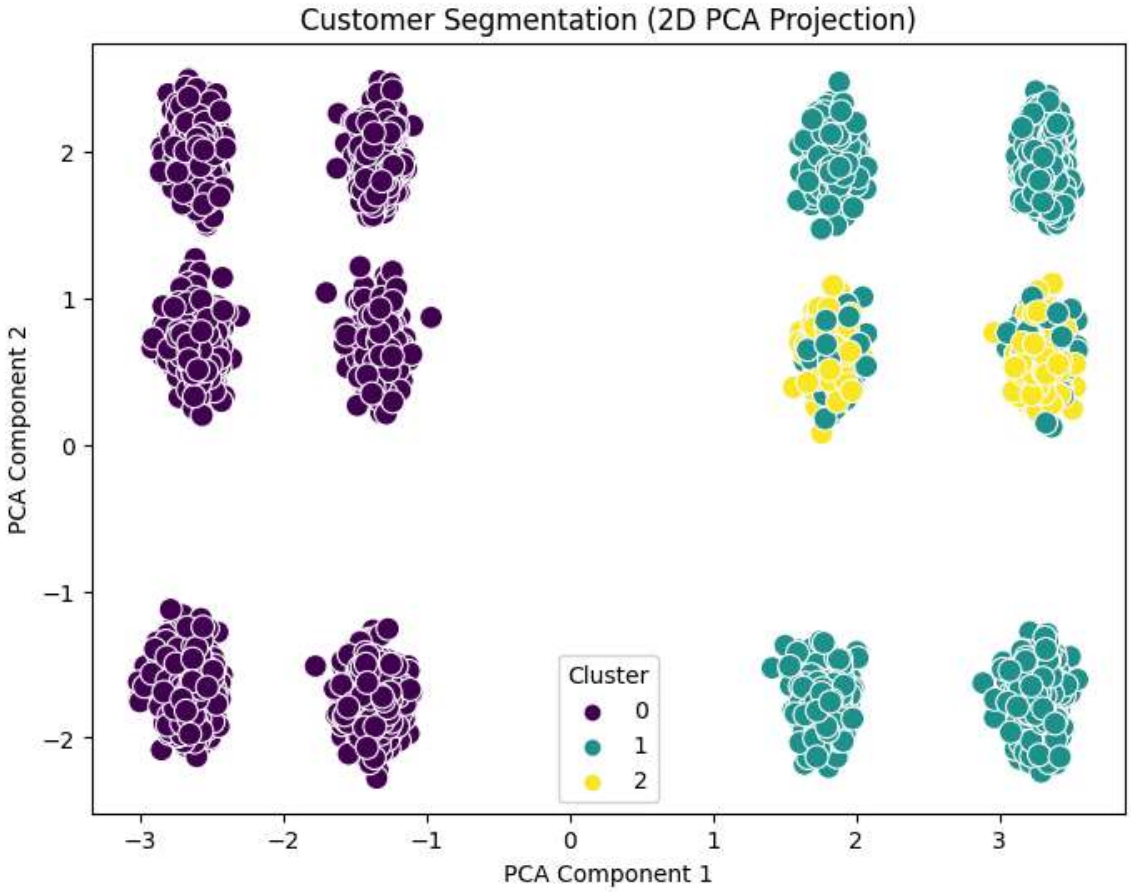
```python
# 8. Apply Insights
# Example: Print insights for each cluster
for cluster in df['Cluster'].unique():
    print(f"Cluster {cluster} statistics:")
    cluster_data = df[df['Cluster'] == cluster][numeric_columns]
    print(cluster_data.describe())
```

```
Cluster 1 statistics:
                Age  Purchase Amount (USD)  Review Rating  Previous Purchas
es
count  1418.000000            1418.000000    1418.000000         1418.0000
00
mean     44.150212              58.818759       3.729619           25.7447
11
std      15.381184              23.683404       0.715301           14.4670
45
min      18.000000              20.000000       2.500000            1.0000
00
25%      31.000000              38.000000       3.100000           14.0000
00
50%      44.000000              59.000000       3.700000           25.0000
00
75%      58.000000              79.000000       4.400000           38.0000
00
max      70.000000             100.000000       5.000000           50.0000
00
Cluster 2 statistics:
               Age  Purchase Amount (USD)  Review Rating  Previous Purchase
s
count  260.000000             260.000000     260.000000          260.00000
0
mean    44.076923              61.738462       3.793462           25.77692
3
std     15.075417              23.067319       0.725325           14.28948
7
min     18.000000              20.000000       2.500000            1.00000
0
25%     31.000000              42.000000       3.200000           13.00000
0
50%     46.000000              63.000000       3.900000           26.00000
0
75%     56.000000              81.250000       4.400000           37.00000
0
max     70.000000             100.000000       5.000000           50.00000
0
Cluster 0 statistics:
                Age  Purchase Amount (USD)  Review Rating  Previous Purchas
es
count  2222.000000            2222.000000    2222.000000         2222.0000
00
mean     44.015302              60.136814       3.757831           25.0508
55
std      15.117899              23.743777       0.715682           14.4519
15
min      18.000000              20.000000       2.500000            1.0000
00
25%      31.000000              39.000000       3.100000           13.0000
00
50%      44.000000              60.000000       3.800000           25.0000
00
75%      57.000000              81.000000       4.400000           37.7500
00
max      70.000000             100.000000       5.000000           50.0000
00
```

In [18]:
```
# 9. Model Evaluation
# Evaluation Metrics
# Silhouette Score
sil_score = silhouette_score(data_scaled, clusters)
print(f'Silhouette Score: {sil_score}')

# Davies-Bouldin Index
db_index = davies_bouldin_score(data_scaled, clusters)
print(f'Davies-Bouldin Index: {db_index}')

# Calinski-Harabasz Index
ch_score = calinski_harabasz_score(data_scaled, clusters)
print(f'Calinski-Harabasz Index: {ch_score}')

# Inertia (Within-Cluster Sum of Squares)
inertia_final = kmeans.inertia_
print(f'Inertia (Within-Cluster Sum of Squares): {inertia_final}')
```

```
Silhouette Score: 0.026911378149880854
Davies-Bouldin Index: 5.306001815623909
Calinski-Harabasz Index: 92.39060651257172
Inertia (Within-Cluster Sum of Squares): 554793.7485658784
```

In [19]:
```
# 10. Documentation and Reporting
# Example of documentation
with open('clustering_report.txt', 'w') as f:
    f.write("Clustering Report\n")
    f.write("================\n")
    f.write(f"Optimal Number of Clusters: {optimal_k}\n")
    f.write(f"Silhouette Score: {sil_score}\n")
    f.write(f"Davies-Bouldin Index: {db_index}\n")
    f.write(f"Calinski-Harabasz Index: {ch_score}\n")
    f.write(f"Inertia: {inertia_final}\n\n")
    f.write("Cluster Statistics:\n")
    for cluster in df['Cluster'].unique():
        f.write(f"Cluster {cluster} statistics:\n")
        cluster_data = df[df['Cluster'] == cluster][numeric_columns]
        f.write(f"{cluster_data.describe()}\n\n")
    f.write("Visualization of Clusters:\n")
    # Save plot to file
    plt.figure(figsize=(8, 6))
    sns.scatterplot(x=data_pca[:, 0], y=data_pca[:, 1], hue=df['Cluster'],
    plt.title('Customer Segmentation (2D PCA Projection)')
    plt.xlabel('PCA Component 1')
    plt.ylabel('PCA Component 2')
    plt.savefig('clusters_visualization.png')
    f.write("See'clusters_visualization.png'for the cluster visualization."
```



In [ ]:

In [ ]: