```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

```
# Load the dataset
# Dataset example file: customer_data.csv
dataset_url = "/content/Mall_Customers.csv"
df = pd.read_csv(dataset_url)

# Display the first few rows of the dataset
print("First 5 rows of the dataset:")
print(df.head())

# Display basic information about the dataset
print("\nDataset Info:")
print(df.info())

# Check for missing values
print("\nMissing Values in Dataset:")
print(df.isnull().sum())
```

```
First 5 rows of the dataset:
   CustomerID   Genre  Age  Annual_Income_(k$)  Spending_Score
0           1    Male   19                  15              39
1           2    Male   21                  15              81
2           3  Female   20                  16               6
3           4  Female   23                  16              77
4           5  Female   31                  17              40

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   CustomerID          200 non-null    int64
 1   Genre               200 non-null    object
 2   Age                 200 non-null    int64
 3   Annual_Income_(k$)  200 non-null    int64
 4   Spending_Score      200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None

Missing Values in Dataset:
CustomerID            0
Genre                 0
Age                   0
Annual_Income_(k$)    0
Spending_Score        0
dtype: int64
```

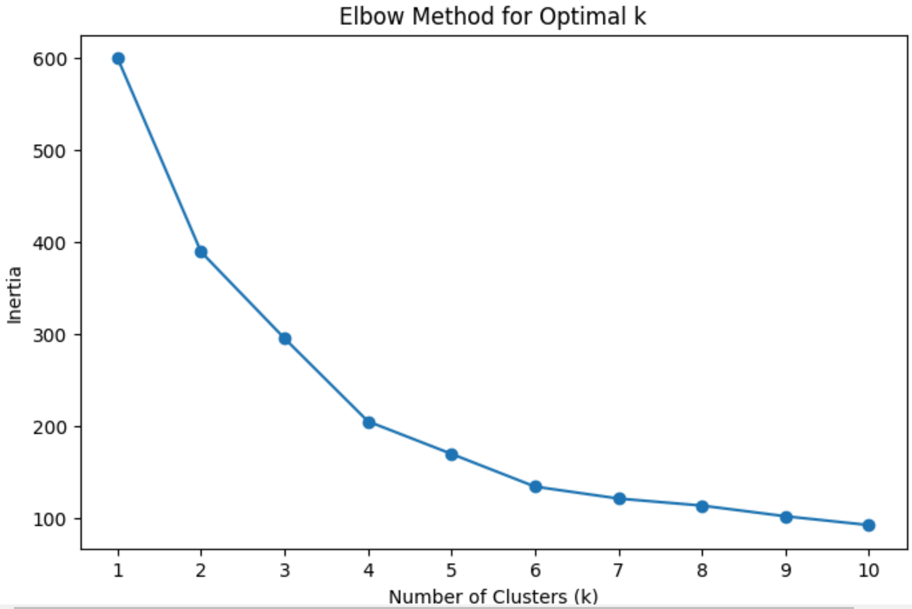
Start coding or generate with AI.

```
# Select relevant columns (e.g., Age, Annual Income, Spending Score)
features = df [['Age', 'Annual_Income_(k$)', 'Spending_Score']]

# Standardize the data
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# Display the first few rows of the standardized data
print("\nFirst 5 rows of scaled features: ")
print(scaled_features[:5])
```

```
First 5 rows of scaled features:
[[-1.42456879 -1.73899919 -0.43480148]
 [-1.28103541 -1.73899919  1.19570407]
 [-1.3528021  -1.70082976 -1.71591298]
 [-1.13750203 -1.70082976  1.04041783]
 [-0.56336851 -1.66266033 -0.39597992]]
```

```
# Elbow Method to find the optimal number of clusters
inertia = []
```

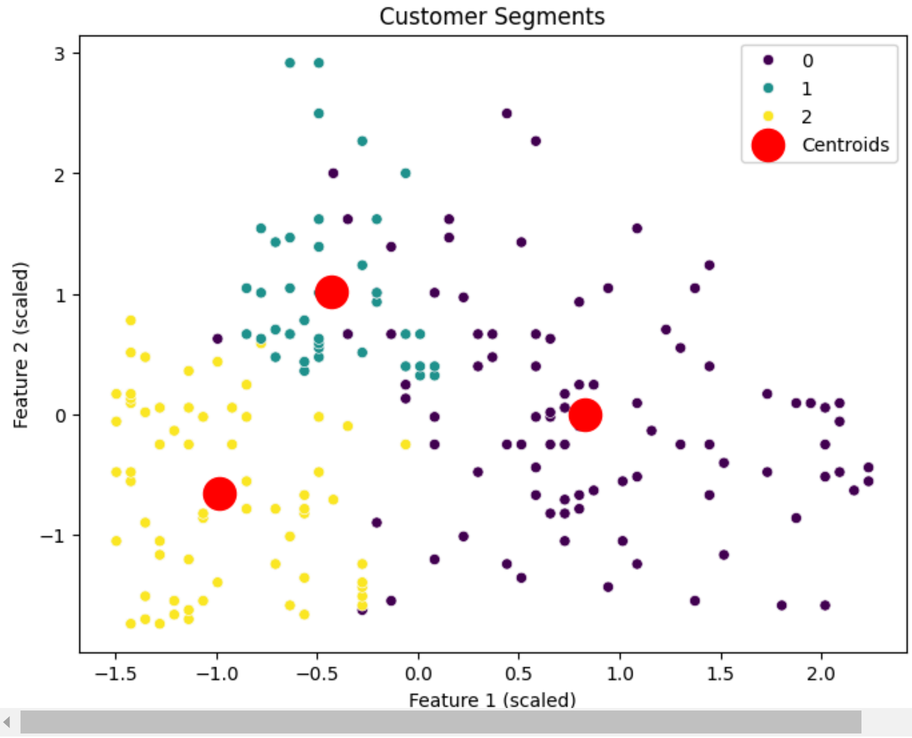
```
k_range = range(1, 11)
for k in k_range:
 kmeans = KMeans (n_clusters=k, random_state=42)
 kmeans.fit(scaled_features)
 inertia.append(kmeans. inertia_)

# Plot the Elbow Method graph
plt.figure(figsize=(8, 5))
plt.plot(k_range, inertia, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.xticks(k_range)
plt.show()
```




```
# Perform K-Means clustering with the optil k from elbow method, assume k=3 here)
optimal_k = 3
kmeans = KMeans (n_clusters=optimal_k, random_state=42)
cluster_labels = kmeans.fit_predict (scaled_features)

# Add cluster Labels to the original dataset
df['Cluster'] = cluster_labels

# Display the first few rows with cluster Labels
print("\nFirst 5 rows with cluster labels:")
print (df.head())
```

```
First 5 rows with cluster labels:
   CustomerID   Genre  Age  Annual_Income_(k$)  Spending_Score  Cluster
0           1    Male   19                  15              39        2
1           2    Male   21                  15              81        2
2           3  Female   20                  16               6        2
3           4  Female   23                  16              77        2
4           5  Female   31                  17              40        2
```

```
# Visualize clusters (using the first two features for plotting)
plt.figure(figsize=(8, 6))
sns.scatterplot (x=scaled_features[:, 0], y=scaled_features[:, 1], hue=cluster_labels, palette='viridis',s=35)
plt.scatter (kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300, c='red', label='Centroids')
plt.title('Customer Segments')
plt.xlabel('Feature 1 (scaled)')
plt.ylabel('Feature 2 (scaled)')
plt.legend()
plt.show()
```


Customer Segments
0
1
2
Centroids
Feature 2 (scaled)
Feature 1 (scaled)
3
2
1
0
−1
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0