Data Science and Machine Learning Internship
(HDLC Technologies)

Name: *B. Deva SriRama Sai Ganesh*
Email: bandarusaiganesh2003@gmail.com

MAIN PROJECT

Question1

Customer Segmentation:

Use Clustering algorithm to segment customers based on the buying behaviors.

Solutions:

Python code:

```
#Data Pre-processing
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')

# Load customer data
df = pd.read_csv('C:\\Users\\banda\\Downloads\\Mall_Customers.csv')

# Print the entire DataFrame
print(df)
```

```
# Print the DataFrame as a string
print(df.to_string())

# Get the shape of the DataFrame
print("Shape: ", df.shape)

# Get information about the DataFrame
print( df.info())

print("Displaying summary statistics: \n")
print(df.describe())

# Check the data types of columns
print(df.dtypes)

# Select relevant features for clustering
features = ['Annual Income (k$)', 'Spending Score (1-100)']
X = df[features]
print(X)

# Plot correlation matrix heatmap
plt.figure()
sns.heatmap(df.corr(), annot = True)
plt.show()

# Plot distribution plots for selected features
plt.figure(1, figsize = (15, 6))
n = 0
for x in [ 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
    n += 1
    plt.subplot(1, 3, n)
    plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
    sns.distplot(df[x], bins = 20)
    plt.title('Distplot of {}'.format(x))
plt.show()
```

```
# Plot countplot for 'Gender'
plt.figure()
sns.countplot(y = 'Gender', data = df)
plt.show()

# Plot violin plots for selected features
plt.figure(1, figsize = (15, 7))
n = 0
for cols in [ 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
    n += 1
    plt.subplot(1, 3, n)
    sns.set(style = 'whitegrid')
    plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
    sns.violinplot(x = cols, y = 'Gender', data = df)
    plt.ylabel('Gender' if n == 1 else '')
    plt.title('Violin Plot')
plt.show()

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

# Determine the optimal number of clusters using the elbow method
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, init = 'k-means++', random_state = 0)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)
```

```
# Plot the elbow curve
import matplotlib.pyplot as plt
plt.plot(range(1, 11), inertia, linewidth = 2, color = "red", marker = '8')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Curve')
plt.show()

# Choose the optimal number of clusters
k = 4

# Perform K-means clustering
kmeans = KMeans(n_clusters = k, init = 'k-means++', random_state = 0)
kmeans.fit(X_scaled)

# Add cluster labels to the original data
df['Cluster'] = kmeans.labels_

# Print the cluster centers
cluster_centers = scaler.inverse_transform(kmeans.cluster_centers_)
print('Cluster Centers:')
for i in range(k):
    print(f'Cluster {i+1}: {cluster_centers[i]}')

# Visualize the clusters
plt.scatter(X['Annual Income (k$)'], X['Spending Score (1-100)'],
c=kmeans.labels_, cmap='viridis')
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], marker='x',
color='red')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Customer Segmentation')
plt.show()
```

In [1]:
```
#Data Pre-processing
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')

# Load customer data
df = pd.read_csv('C:\\Users\\banda\\Downloads\\Mall_Customers.csv')

# Print the entire DataFrame
print(df)
```

```
     CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
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
..          ...     ...  ...                 ...                     ...
195         196  Female   35                 120                      79
196         197  Female   45                 126                      28
197         198    Male   32                 126                      74
198         199    Male   32                 137                      18
199         200    Male   30                 137                      83

[200 rows x 5 columns]
```

In [2]:
```
# Print the DataFrame as a string
print(df.to_string())
```

```
     CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
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
5             6  Female   22                  17                      76
6             7  Female   35                  18                       6
7             8  Female   23                  18                      94
8             9    Male   64                  19                       3
9            10  Female   30                  19                      72
10           11    Male   67                  19                      14
11           12  Female   35                  19                      99
12           13  Female   58                  20                      15
13           14  Female   24                  20                      77
14           15    Male   37                  20                      13
15           16    Male   22                  20                      79
16           17  Female   35                  21                      35
17           18    Male   20                  21                      66
18           19    Male   52                  23                      29
19           20  Female   35                  23                      98
20           21    Male   35                  24                      35
```

21	22	Male	25	24	73
22	23	Female	46	25	5
23	24	Male	31	25	73
24	25	Female	54	28	14
25	26	Male	29	28	82
26	27	Female	45	28	32
27	28	Male	35	28	61
28	29	Female	40	29	31
29	30	Female	23	29	87
30	31	Male	60	30	4
31	32	Female	21	30	73
32	33	Male	53	33	4
33	34	Male	18	33	92
34	35	Female	49	33	14
35	36	Female	21	33	81
36	37	Female	42	34	17
37	38	Female	30	34	73
38	39	Female	36	37	26
39	40	Female	20	37	75
40	41	Female	65	38	35
41	42	Male	24	38	92
42	43	Male	48	39	36
43	44	Female	31	39	61
44	45	Female	49	39	28
45	46	Female	24	39	65
46	47	Female	50	40	55
47	48	Female	27	40	47
48	49	Female	29	40	42
49	50	Female	31	40	42
50	51	Female	49	42	52
51	52	Male	33	42	60
52	53	Female	31	43	54
53	54	Male	59	43	60
54	55	Female	50	43	45
55	56	Male	47	43	41
56	57	Female	51	44	50
57	58	Male	69	44	46
58	59	Female	27	46	51
59	60	Male	53	46	46
60	61	Male	70	46	56
61	62	Male	19	46	55
62	63	Female	67	47	52
63	64	Female	54	47	59
64	65	Male	63	48	51
65	66	Male	18	48	59

66	67	Female	43	48	50
67	68	Female	68	48	48
68	69	Male	19	48	59
69	70	Female	32	48	47
70	71	Male	70	49	55
71	72	Female	47	49	42
72	73	Female	60	50	49
73	74	Female	60	50	56
74	75	Male	59	54	47
75	76	Male	26	54	54
76	77	Female	45	54	53
77	78	Male	40	54	48
78	79	Female	23	54	52
79	80	Female	49	54	42
80	81	Male	57	54	51
81	82	Male	38	54	55
82	83	Male	67	54	41
83	84	Female	46	54	44
84	85	Female	21	54	57
85	86	Male	48	54	46
86	87	Female	55	57	58
87	88	Female	22	57	55
88	89	Female	34	58	60
89	90	Female	50	58	46
90	91	Female	68	59	55
91	92	Male	18	59	41
92	93	Male	48	60	49
93	94	Female	40	60	40
94	95	Female	32	60	42
95	96	Male	24	60	52
96	97	Female	47	60	47
97	98	Female	27	60	50
98	99	Male	48	61	42
99	100	Male	20	61	49
100	101	Female	23	62	41
101	102	Female	49	62	48
102	103	Male	67	62	59
103	104	Male	26	62	55
104	105	Male	49	62	56
105	106	Female	21	62	42
106	107	Female	66	63	50
107	108	Male	54	63	46
108	109	Male	68	63	43
109	110	Male	66	63	48
110	111	Male	65	63	52

111	112	Female	19	63	54
112	113	Female	38	64	42
113	114	Male	19	64	46
114	115	Female	18	65	48
115	116	Female	19	65	50
116	117	Female	63	65	43
117	118	Female	49	65	59
118	119	Female	51	67	43
119	120	Female	50	67	57
120	121	Male	27	67	56
121	122	Female	38	67	40
122	123	Female	40	69	58
123	124	Male	39	69	91
124	125	Female	23	70	29
125	126	Female	31	70	77
126	127	Male	43	71	35
127	128	Male	40	71	95
128	129	Male	59	71	11
129	130	Male	38	71	75
130	131	Male	47	71	9
131	132	Male	39	71	75
132	133	Female	25	72	34
133	134	Female	31	72	71
134	135	Male	20	73	5
135	136	Female	29	73	88
136	137	Female	44	73	7
137	138	Male	32	73	73
138	139	Male	19	74	10
139	140	Female	35	74	72
140	141	Female	57	75	5
141	142	Male	32	75	93
142	143	Female	28	76	40
143	144	Female	32	76	87
144	145	Male	25	77	12
145	146	Male	28	77	97
146	147	Male	48	77	36
147	148	Female	32	77	74
148	149	Female	34	78	22
149	150	Male	34	78	90
150	151	Male	43	78	17
151	152	Male	39	78	88
152	153	Female	44	78	20
153	154	Female	38	78	76
154	155	Female	47	78	16
155	156	Female	27	78	89

156	157	Male	37	78	1
157	158	Female	30	78	78
158	159	Male	34	78	1
159	160	Female	30	78	73
160	161	Female	56	79	35
161	162	Female	29	79	83
162	163	Male	19	81	5
163	164	Female	31	81	93
164	165	Male	50	85	26
165	166	Female	36	85	75
166	167	Male	42	86	20
167	168	Female	33	86	95
168	169	Female	36	87	27
169	170	Male	32	87	63
170	171	Male	40	87	13
171	172	Male	28	87	75
172	173	Male	36	87	10
173	174	Male	36	87	92
174	175	Female	52	88	13
175	176	Female	30	88	86
176	177	Male	58	88	15
177	178	Male	27	88	69
178	179	Male	59	93	14
179	180	Male	35	93	90
180	181	Female	37	97	32
181	182	Female	32	97	86
182	183	Male	46	98	15
183	184	Female	29	98	88
184	185	Female	41	99	39
185	186	Male	30	99	97
186	187	Female	54	101	24
187	188	Male	28	101	68
188	189	Female	41	103	17
189	190	Female	36	103	85
190	191	Female	34	103	23
191	192	Female	32	103	69
192	193	Male	33	113	8
193	194	Female	38	113	91
194	195	Female	47	120	16
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

In [3]:
```
# Get the shape of the DataFrame
print("Shape: ", df.shape)
```

```
Shape:  (200, 5)
```

In [4]:
```
# Get information about the DataFrame
print( df.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   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
```

In [5]:
```
print("Displaying summary statistics: \n")
print(df.describe())
```

```
Displaying summary statistics:

       CustomerID         Age  Annual Income (k$)  Spending Score (1-100)
count  200.000000  200.000000          200.000000              200.000000
mean   100.500000   38.850000           60.560000               50.200000
std     57.879185   13.969007           26.264721               25.823522
min      1.000000   18.000000           15.000000                1.000000
25%     50.750000   28.750000           41.500000               34.750000
50%    100.500000   36.000000           61.500000               50.000000
75%    150.250000   49.000000           78.000000               73.000000
max    200.000000   70.000000          137.000000               99.000000
```

In [6]:
```
# Check the data types of columns
print(df.dtypes)
```

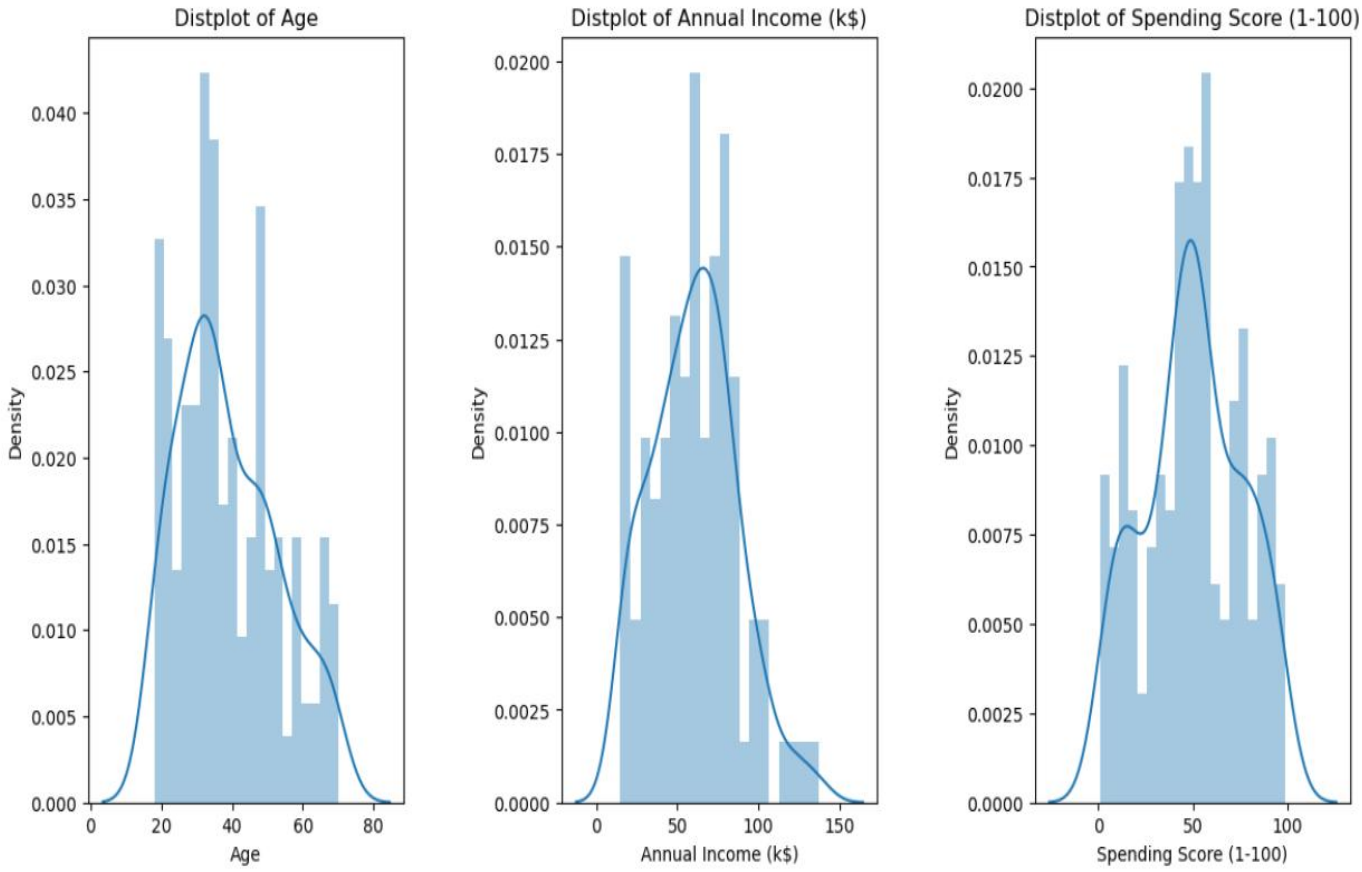
```
CustomerID                int64
Gender                   object
Age                       int64
Annual Income (k$)        int64
Spending Score (1-100)    int64
dtype: object
```

In [7]:
```
# Select relevant features for clustering
features = ['Annual Income (k$)', 'Spending Score (1-100)']
X = df[features]
print(X)
```

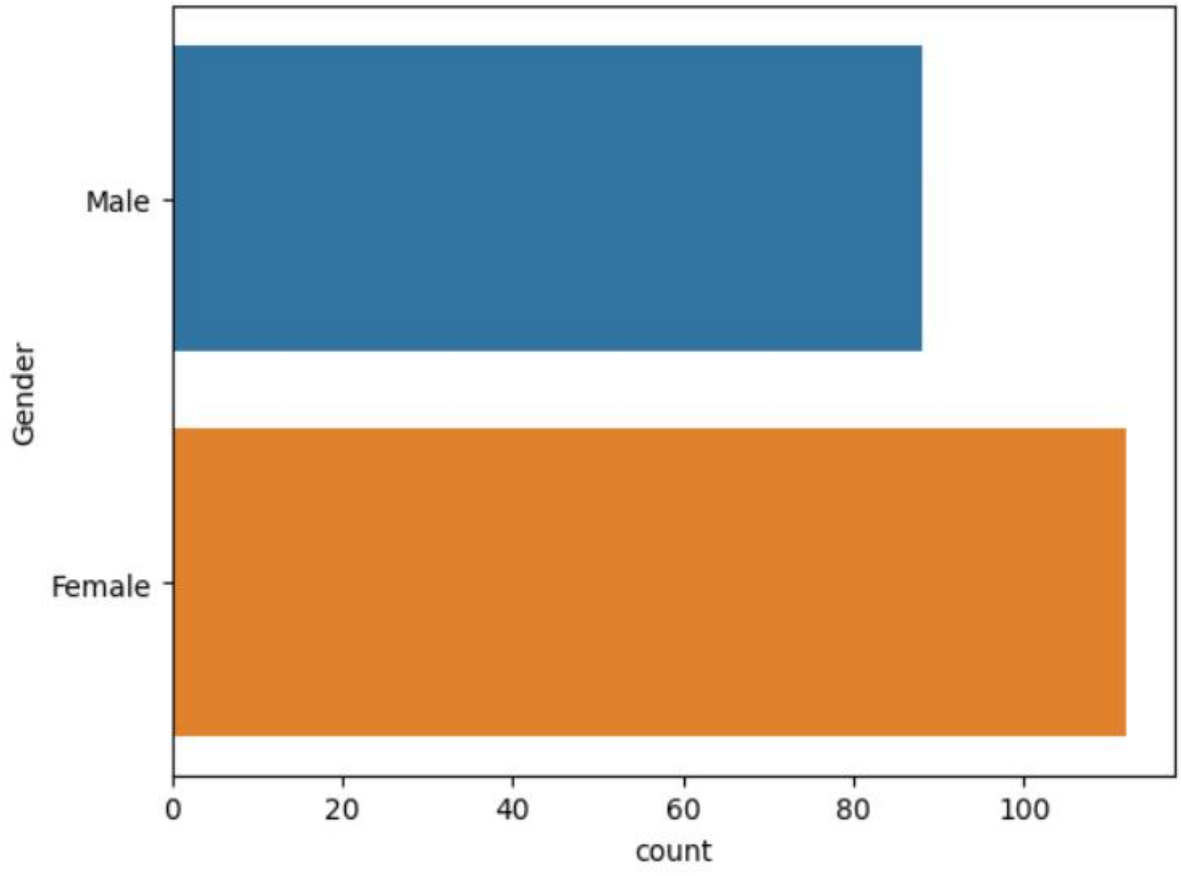
```
     Annual Income (k$)  Spending Score (1-100)
0                    15                      39
1                    15                      81
2                    16                       6
3                    16                      77
4                    17                      40
..                  ...                     ...
195                 120                      79
196                 126                      28
197                 126                      74
198                 137                      18
199                 137                      83

[200 rows x 2 columns]
```

In [8]:
```
# Plot correlation matrix heatmap
plt.figure()
sns.heatmap(df.corr(), annot = True)
plt.show()
```

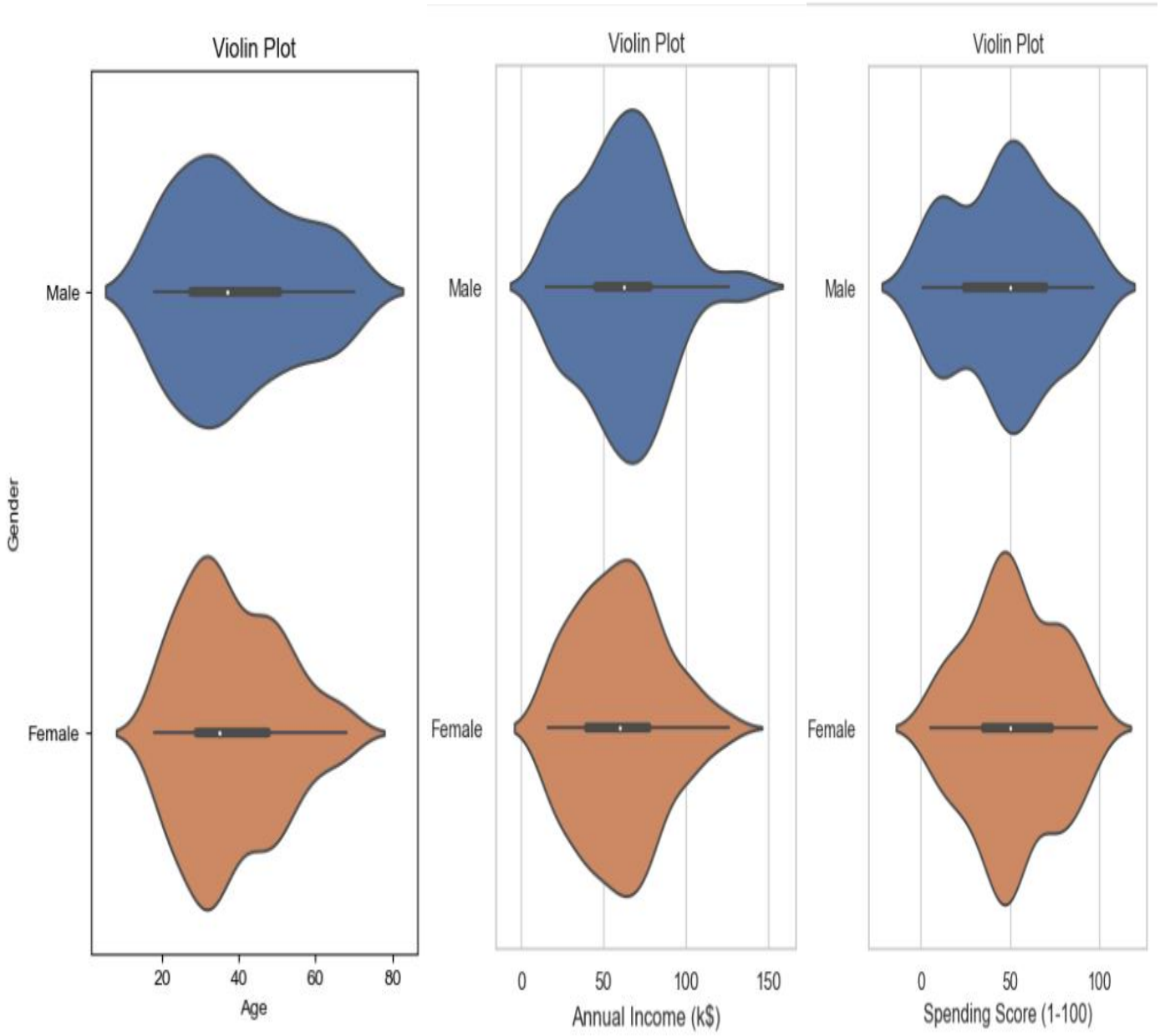
In [9]:
```
# Plot distribution plots for selected features
plt.figure(1, figsize = (15, 6))
n = 0
for x in [ 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
    n += 1
    plt.subplot(1, 3, n)
    plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
    sns.distplot(df[x], bins = 20)
    plt.title('Distplot of {}'.format(x))
plt.show()
```



In [13]:
```
# Plot countplot for 'Gender'
plt.figure()
sns.countplot(y = 'Gender', data = df)
plt.show()
```



In [14]:
```
# Plot violin plots for selected features
plt.figure(1, figsize = (15, 7))
n = 0
for cols in [ 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
    n += 1
    plt.subplot(1, 3, n)
    sns.set(style = 'whitegrid')
    plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
    sns.violinplot(x = cols, y = 'Gender', data = df)
    plt.ylabel('Gender' if n == 1 else '')
    plt.title('Violin Plot')
plt.show()
```

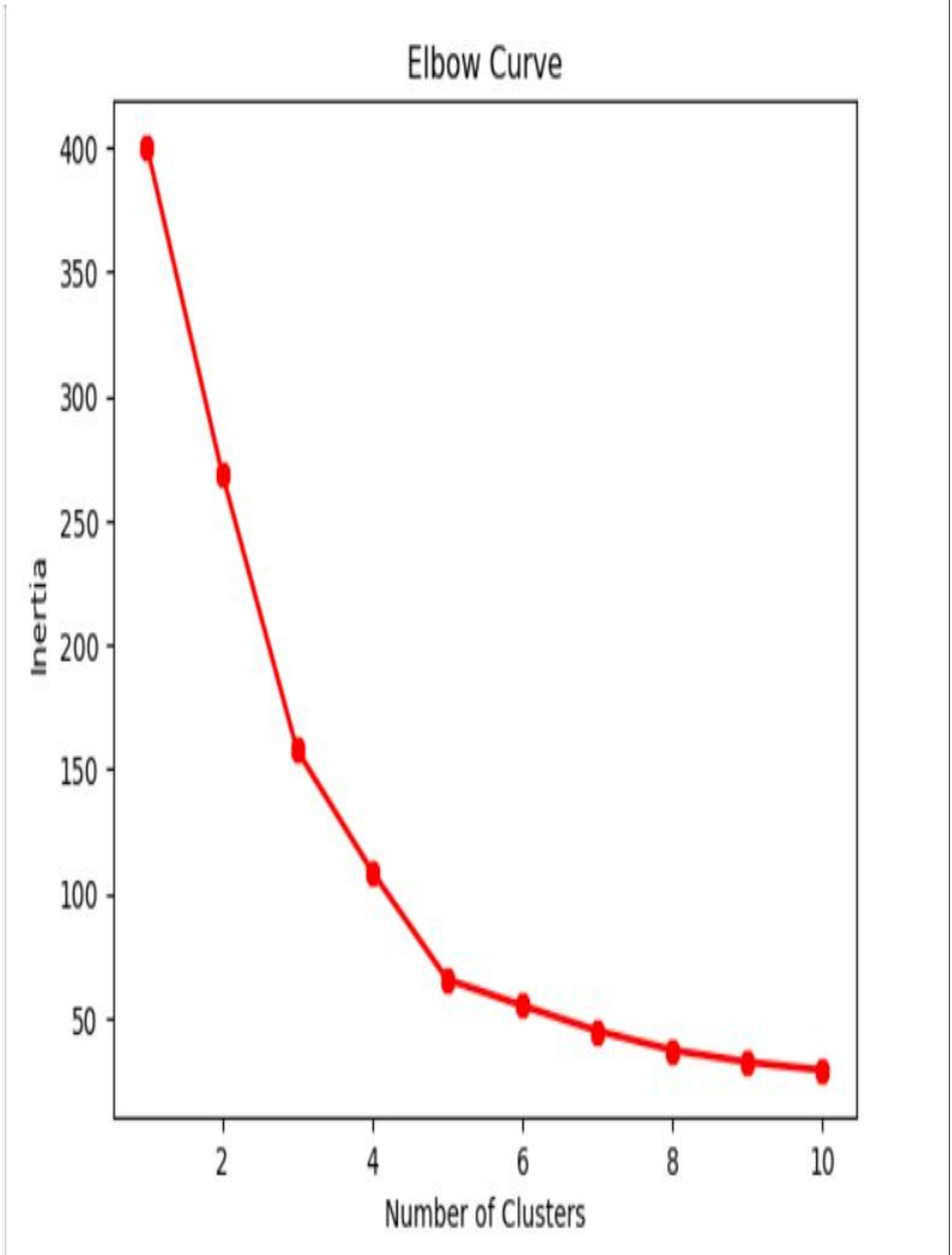


In [16]:
```
# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
print(X_scaled[:10])
```

```
[[-1.73899919 -0.43480148]
 [-1.73899919  1.19570407]
 [-1.70082976 -1.71591298]
 [-1.70082976  1.04041783]
 [-1.66266033 -0.39597992]
 [-1.66266033  1.00159627]
 [-1.62449091 -1.71591298]
 [-1.62449091  1.70038436]
 [-1.58632148 -1.83237767]
 [-1.58632148  0.84631002]]
```

In [19]:
```
# Determine the optimal number of clusters using the elbow method
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, init = 'k-means++', random_state = 0)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)
```

In [20]:
```
# Plot the elbow curve
import matplotlib.pyplot as plt
plt.plot(range(1, 11), inertia, linewidth = 2, color = "red", marker = '8')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Curve')
plt.show()
```

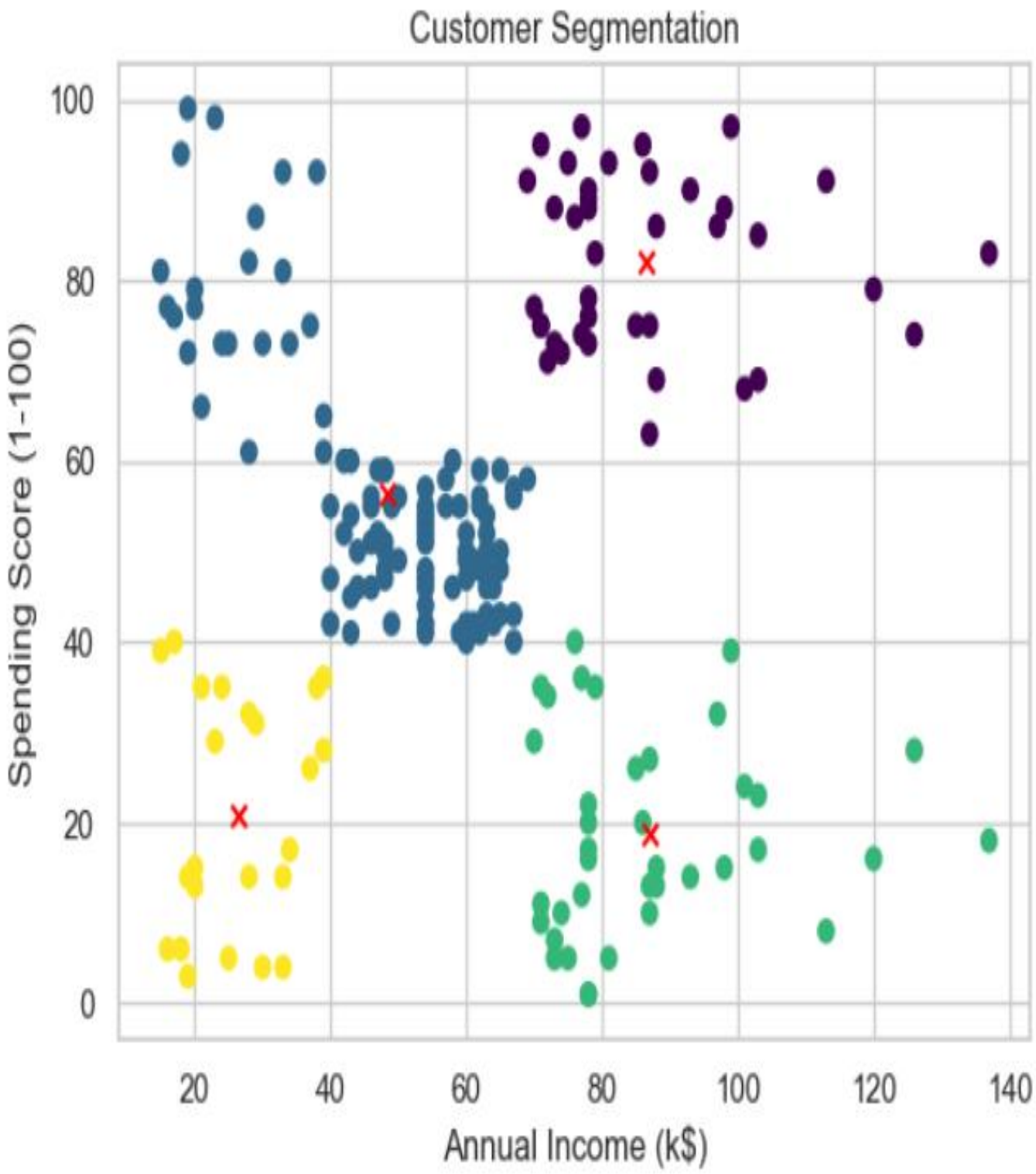

Elbow Curve
Inertia
400
350
300
250
200
150
100
50
2
4
6
8
10
Number of Clusters

In [21]:
```
# Choose the optimal number of clusters
k = 4

# Perform K-means clustering
kmeans = KMeans(n_clusters = k, init = 'k-means++', random_state = 0)
kmeans.fit(X_scaled)

# Add cluster labels to the original data
df['Cluster'] = kmeans.labels_
```

In [22]:
```
# Print the cluster centers
cluster_centers = scaler.inverse_transform(kmeans.cluster_centers_)
print('Cluster Centers:')
for i in range(k):
    print(f'Cluster {i+1}: {cluster_centers[i]}')
```

```
Cluster Centers:
Cluster 1: [86.53846154 82.12820513]
Cluster 2: [48.26 56.48]
Cluster 3: [87.         18.63157895]
Cluster 4: [26.30434783 20.91304348]
```

In [23]:
```
# Visualize the clusters
plt.scatter(X['Annual Income (k$)'], X['Spending Score (1-100)'], c=kmeans.labels_, cmap='viridis')
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], marker='x', color='red')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Customer Segmentation')
plt.show()
```



From these analyses, we can draw conclusions about the customer segmentation based on their annual income and spending score. We can identify distinct customer groups or segments and analyze their characteristics to make data-driven decisions and tailor marketing strategies accordingly.