In [3]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import KMeans
customer_dataset = pd.read_csv("C:\\Users\\Acer\\OneDrive\\Desktop\\Mall_Customers.csv")

customer_dataset.head()
```

Out[3]:

	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

In [4]:

```python
customer_dataset.tail()
```

Out[4]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
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 [5]:

```
# shape of dataset

customer_dataset.shape
(200, 5)
```

Out[5]:

```
(200, 5)
```

In [6]:

```
# information about dataset

customer_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   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
```

In [7]:

```
# datatypes of columns dataset

customer_dataset.dtypes
```

Out[7]:

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

In [8]:

```
# statistical measures of dataset
customer_dataset.describe()
```

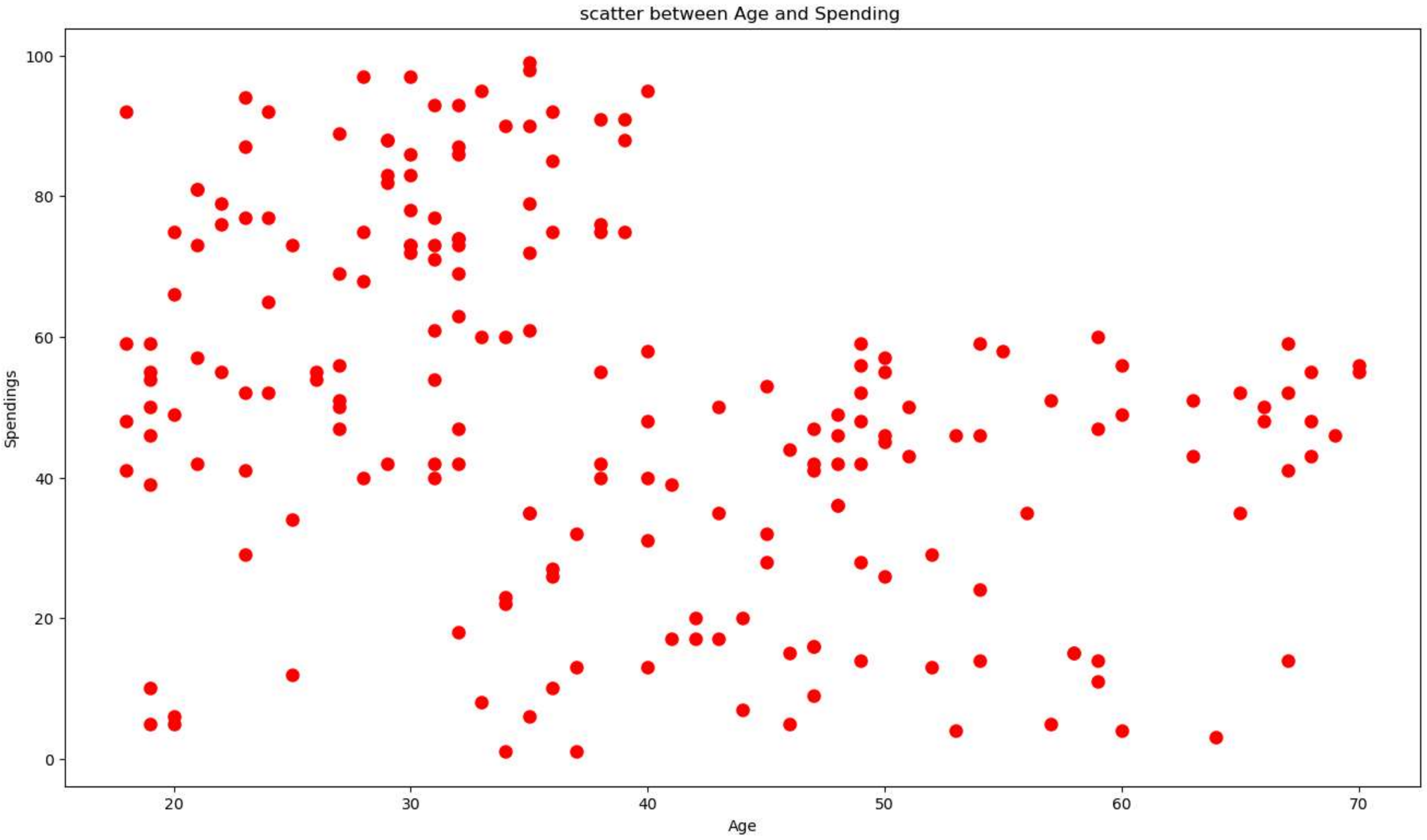
Out[8]:

	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

Data Visulalization

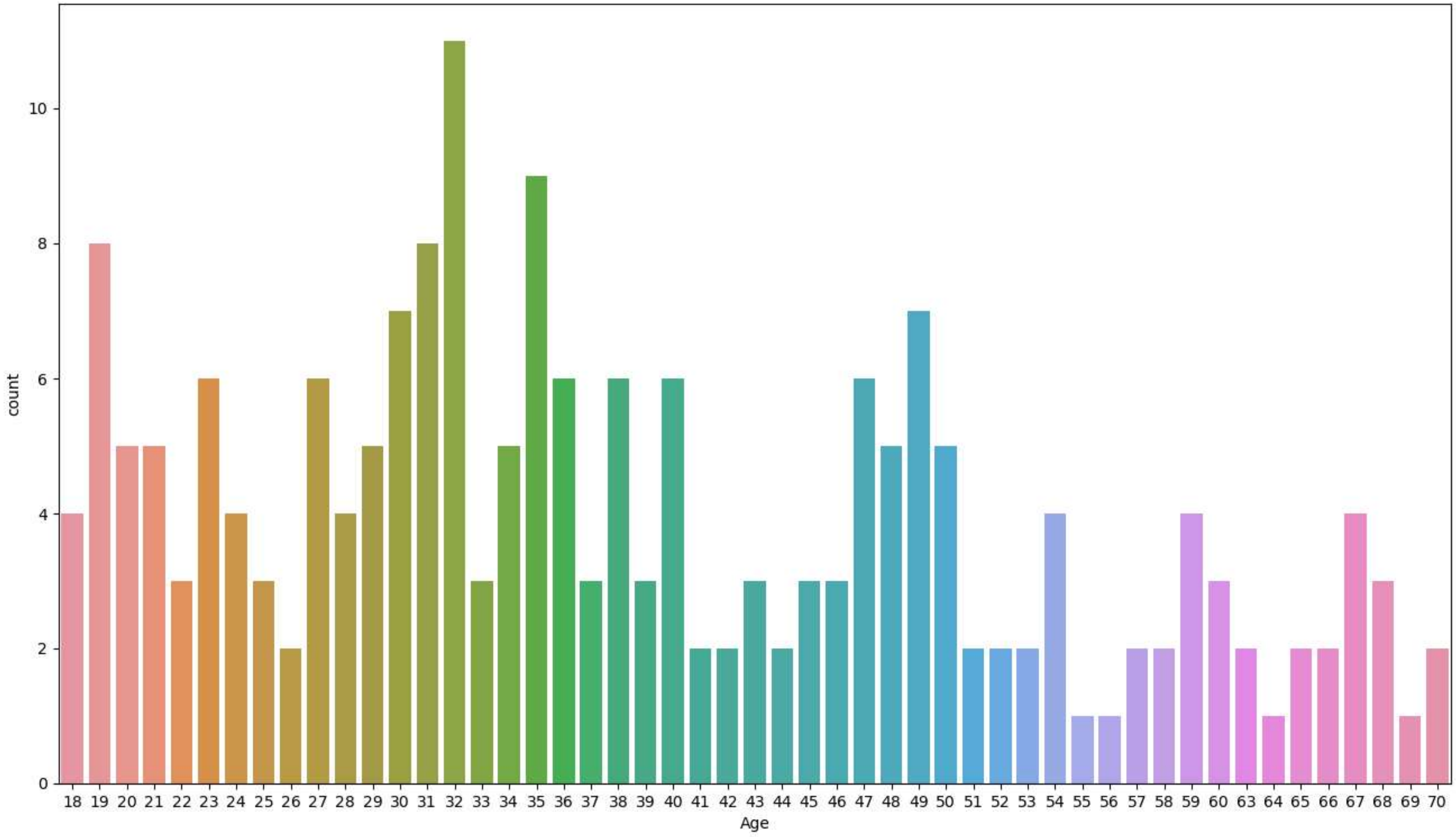
In [9]:

```
plt.figure(figsize = (16,9))
plt.scatter(x = customer_dataset.Age, y = customer_dataset['Spending Score (1-100)'], c = "red", linewidth =
plt.title('scatter between Age and Spending ')
plt.ylabel("Spendings")
plt.xlabel("Age")

plt.show()
```

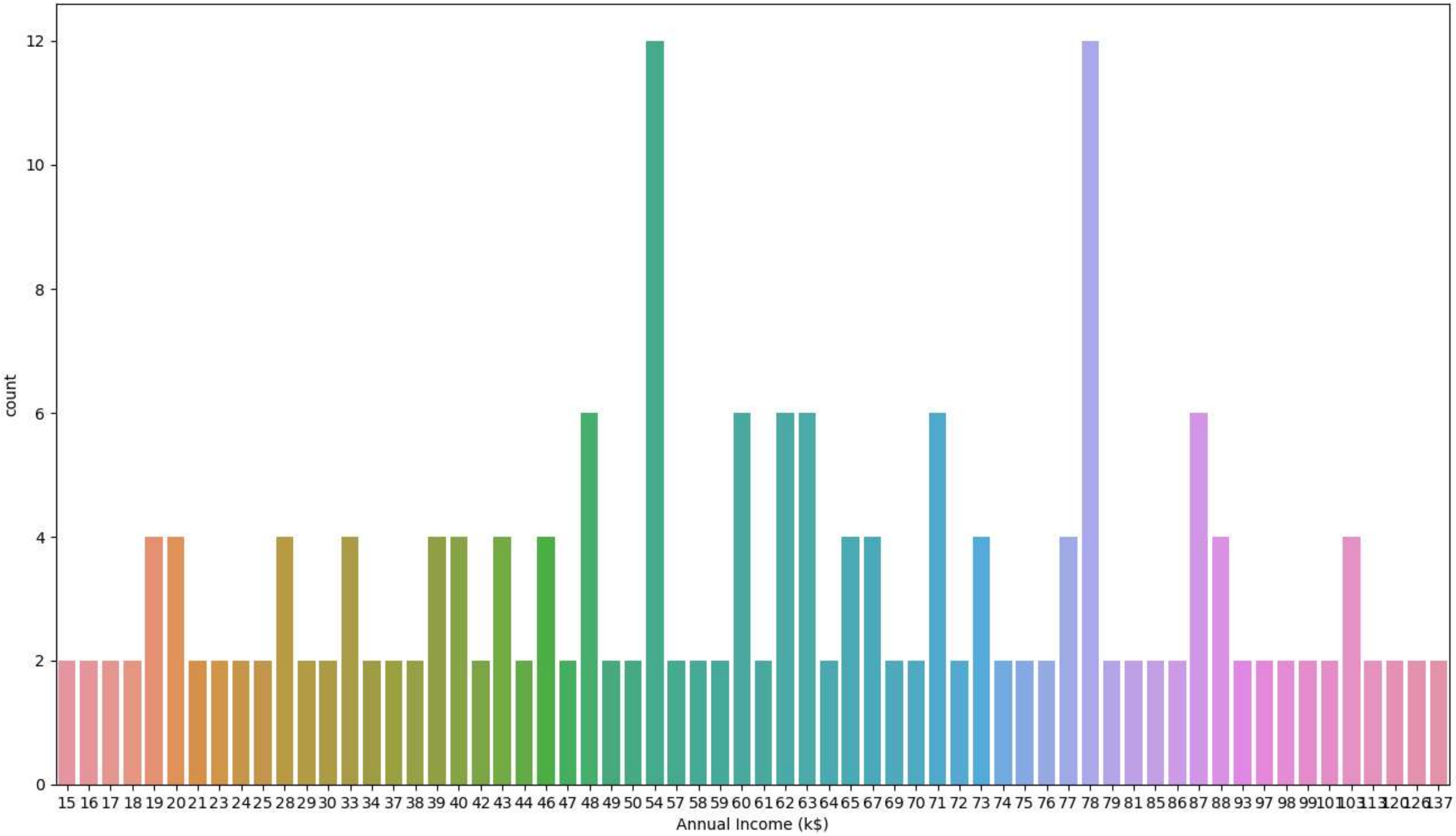


In [16]:

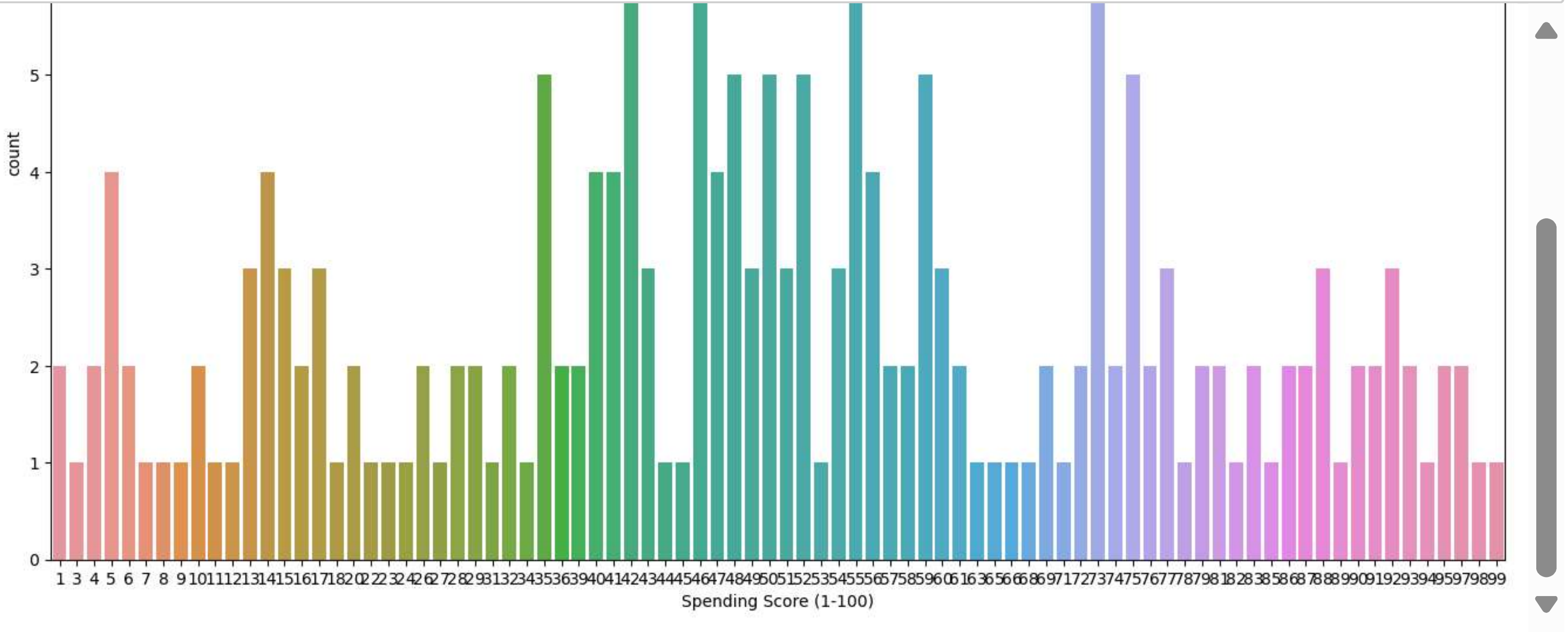
```
plt.figure(figsize=(16,9))
sns.countplot(x='Age', data=customer_dataset)  # This is the plotting command
plt.show()  # This will display the plot
```



In [18]:
```
# Plot configuration
plt.figure(figsize=(16,9))

# Create the count plot
sns.countplot(x='Annual Income (k$)', data=customer_dataset)

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

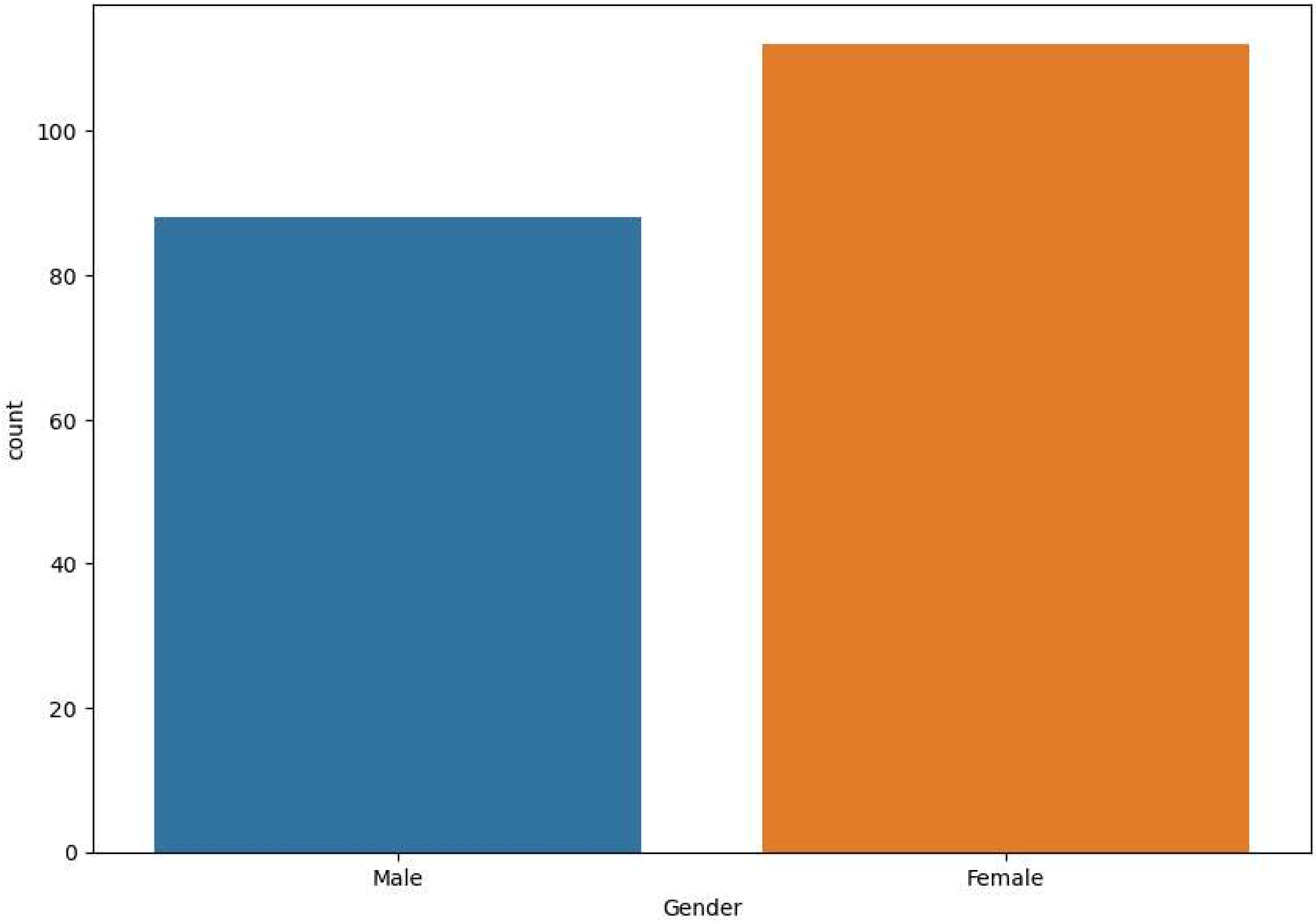


In [20]:
```
# Set the figure size
plt.figure(figsize=(16,9))

# Create the count plot
sns.countplot(x='Spending Score (1-100)', data=customer_dataset)

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



In [21]:

```
# Configure the plot size
plt.figure(figsize=(10,7))

# Create the count plot with explicit keyword arguments
sns.countplot(x='Gender', data=customer_dataset)

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



Choosing the Annual Income Column and Spending Columns

In [22]:

```
customer_dataset.head()
```

Out[22]:

	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

In [24]:

```
customer_dataset.columns
```

Out[24]:

```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

```
In [25]:
# using iloc for picking 3rd and 4th columns
X = customer_dataset.iloc[:,[3,4]].values
print(X)
```

```
 [ 97  32]
 [ 97  86]
 [ 98  15]
 [ 98  88]
 [ 99  39]
 [ 99  97]
 [101  24]
 [101  68]
 [103  17]
 [103  85]
 [103  23]
 [103  69]
 [113   8]
 [113  91]
 [120  16]
 [120  79]
 [126  28]
 [126  74]
 [137  18]
 [137  83]]
```

choosing the number of clusters

```
WCSS : within Clusters Sum of Squares
```

In [26]:

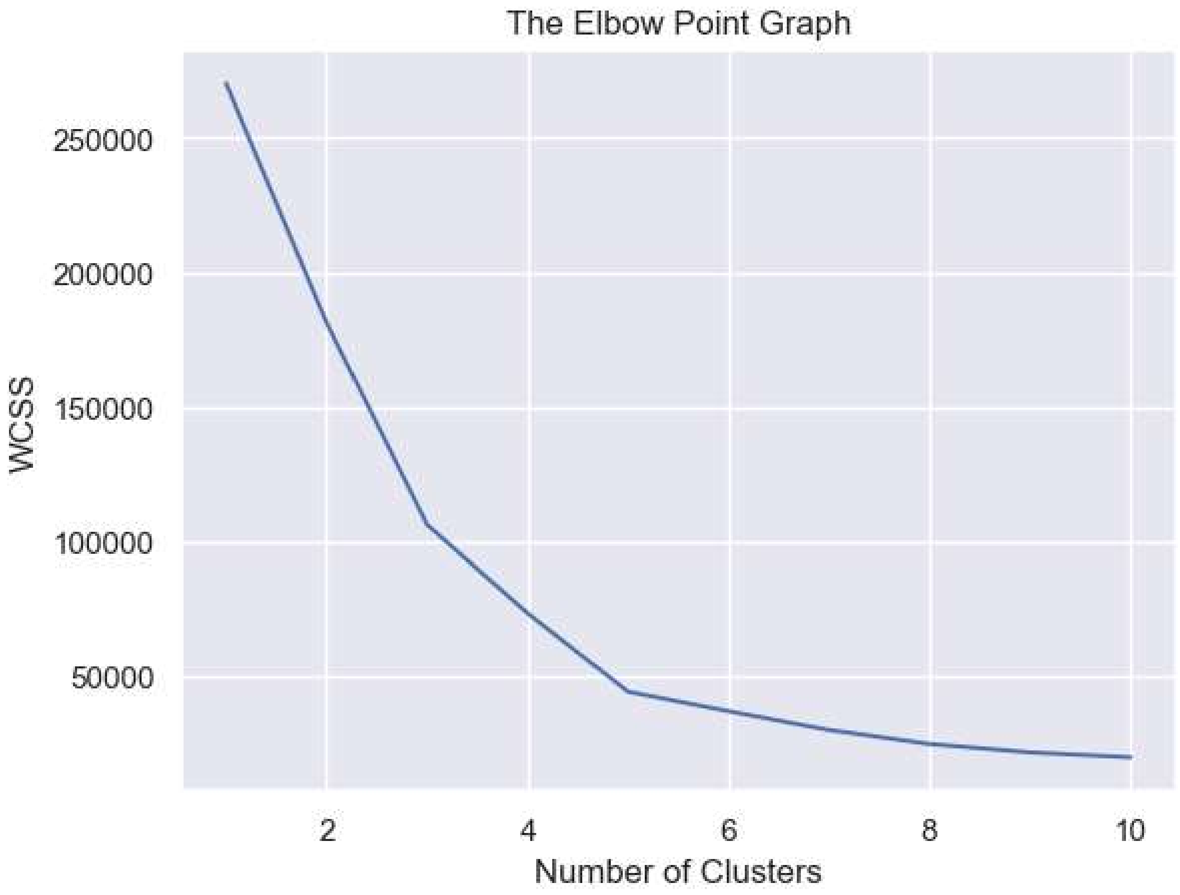
```
# finding WCSS values for different number of clusters

wcss = []

for i in range(1,11):
    # 1 and 11 will be excluded
    kmeans = KMeans(n_clusters=i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

```
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory lea
k on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the en
vironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory lea
k on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the en
vironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory lea
k on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the en
vironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory lea
k on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the en
vironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory lea
k on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the en
vironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory lea
k on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the en
vironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will
```

```
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory lea
k on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the en
vironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory lea
k on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the en
vironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory lea
k on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the en
vironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory lea
k on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the en
vironment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [28]:

```
# plot an elbow graph

sns.set()
plt.plot(range(1,11), wcss)
plt.title("The Elbow Point Graph")
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS")
plt.show()
```



```
Optimum number of clusters = 5
```

Training the K-Means Clustering model

Training the K-Means Clustering model

```
In [30]: kmeans = KMeans(n_clusters =5, init = 'k-means++', random_state =0)

         # return a label for each data point based on their clusters
         Y = kmeans.fit_predict(X)

         print(Y)
```

```
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will
change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\pyth\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory lea
k on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the en
vironment variable OMP_NUM_THREADS=1.
  warnings.warn(

[3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3
 4 3 4 3 4 3 0 3 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 2 1 0 1 2 1 2 1 0 1 2 1 2 1 2 1 2 1 0 1 2 1 2 1
 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2
 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1]
```

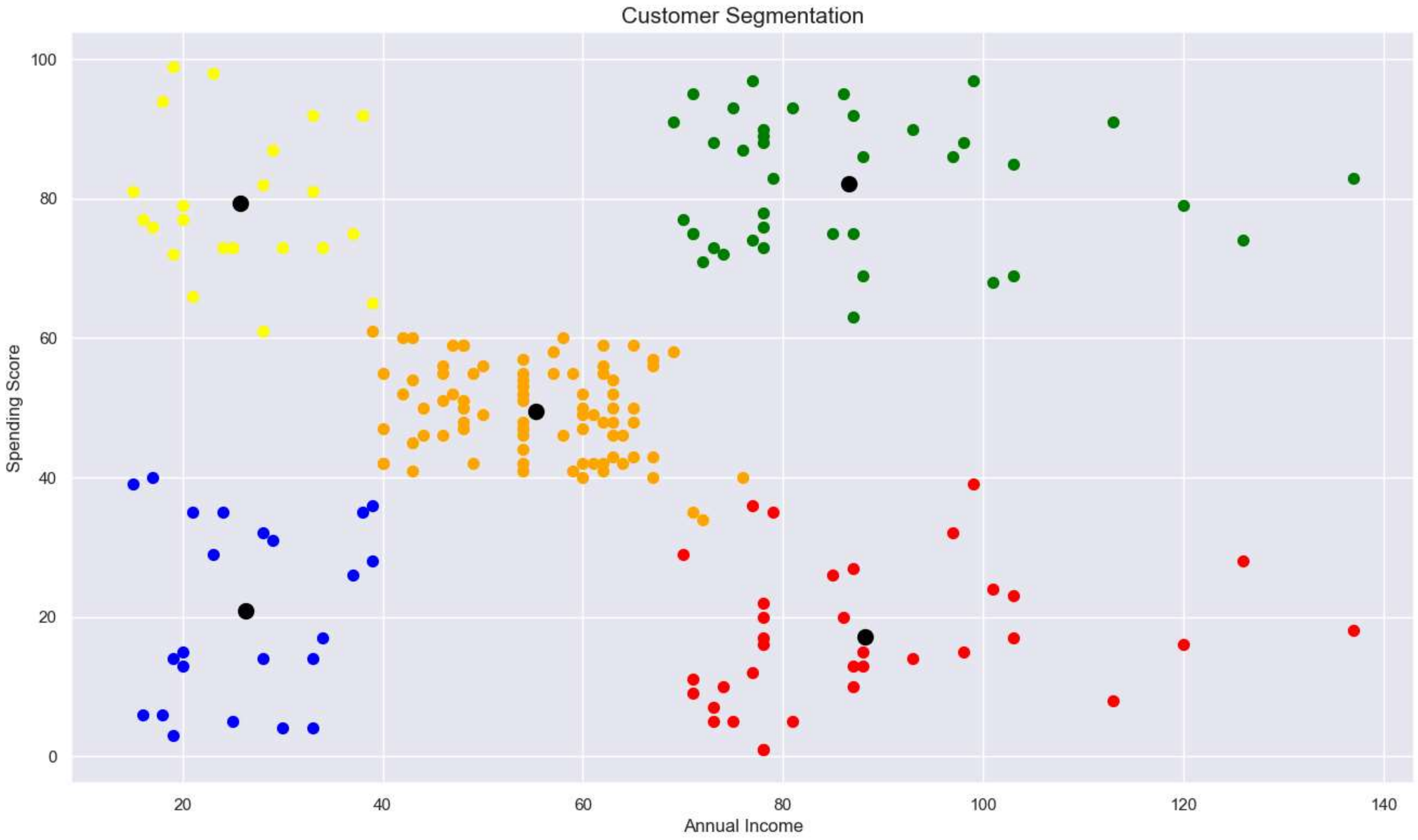
Plotting data in graph

In [31]:

```
# plotting all the clusters and their centroids

plt.figure(figsize = (16,9))
plt.title("Customer Segmentation", fontsize = 15)
plt.scatter(X[Y==0,0], X[Y==0,1], s= 50, c = 'orange', label = 'Cluster 1')
plt.scatter(X[Y==1,0], X[Y==1,1], s= 50, c = 'green', label = 'Cluster 1')
plt.scatter(X[Y==2,0], X[Y==2,1], s= 50, c = 'red', label = 'Cluster 2')
plt.scatter(X[Y==3,0], X[Y==3,1], s= 50, c = 'blue', label = 'Cluster 3')
plt.scatter(X[Y==4,0], X[Y==4,1], s= 50, c = 'yellow', label = 'Cluster 4')

# plot the centroids
plt.xlabel("Annual Income")
plt.ylabel("Spending Score")
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s =100, c='black', label = 'Centroids
```

Out[31]:

```
<matplotlib.collections.PathCollection at 0x20890cfe610>
```


Customer Segmentation
Spending Score
Annual Income
0
20
40
60
80
100
20
40
60
80
100
120
140

```
In [ ]:
```