Customer Segmentation (Clustering)

K-means clustering model

Importing Needed Packages

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import KMeans
```

Data Collection and Analysis

In [2]:
```
customer_dataset = pd.read_csv(r"D:\Arivu\Desktop\projects\mall_customers.csv")
```

In [3]:
```
customer_dataset.head()
```

Out[3]:

	customer_id	gender	age	annual_income	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

In [6]:
```
customer_dataset.tail()
```

Out[6]:

	customer_id	gender	age	annual_income	spending_score
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 [7]:
```
#shape of the dataset

customer_dataset.shape
```

Out[7]:
```
(200, 5)
```

In [8]:
```
#information about dataset

customer_dataset.info
```

Out[8]:
```
<bound method DataFrame.info of      customer_id  gender  age  annual_income  spen
ding_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
..           ...     ...  ...            ...             ...
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 [9]:
```
#datatype of columns

customer_dataset.dtypes
```

Out[9]:
```
customer_id        int64
gender            object
age                int64
annual_income      int64
spending_score     int64
dtype: object
```

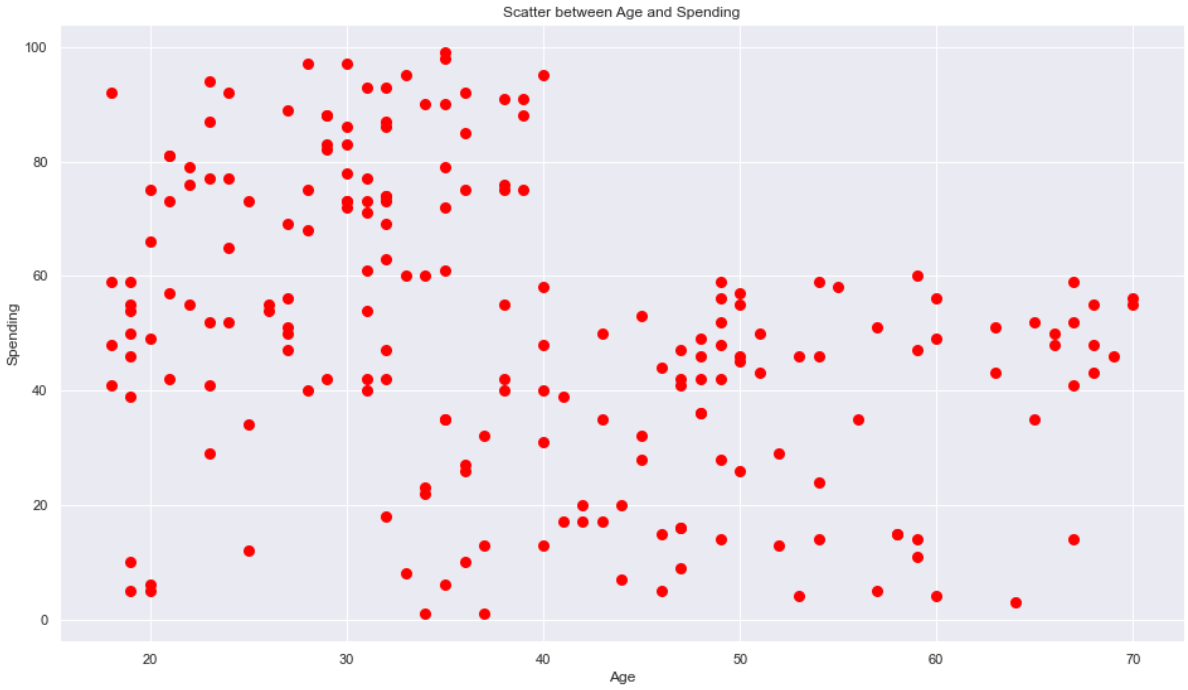
In [10]:
```
#Statistical measures of dataset

customer_dataset.describe()
```

Out[10]:

	customer_id	age	annual_income	spending_score
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 Visualization

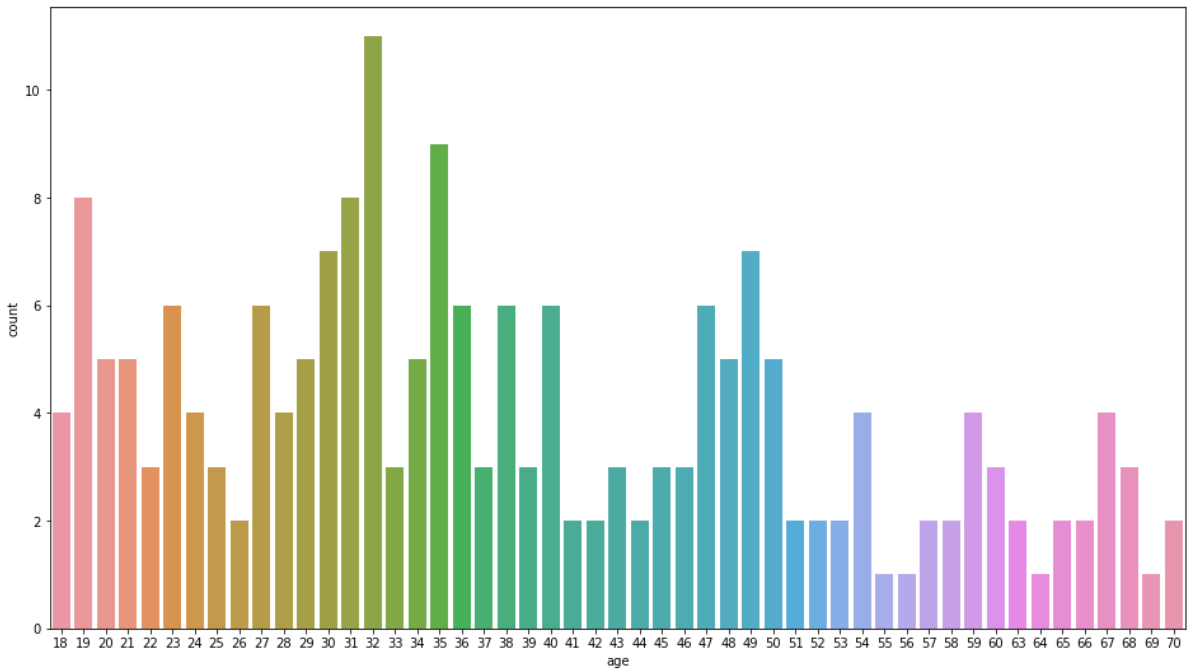
In [37]:
```
plt.figure(figsize = (16,9))
plt.scatter(x=customer_dataset.age, y= customer_dataset.spending_score, c ='red', l
plt.title('Scatter between Age and Spending')
plt.xlabel("Age")
plt.ylabel("Spending")
#xlable-Age
#ylable-Spendings
plt.show()
```



In [16]:
```
plt.figure(figsize = (16,9))
sns.countplot(customer_dataset.age)
#range of age
```

```
C:\Users\arivu\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[16]:
```
<AxesSubplot:xlabel='age', ylabel='count'>
```

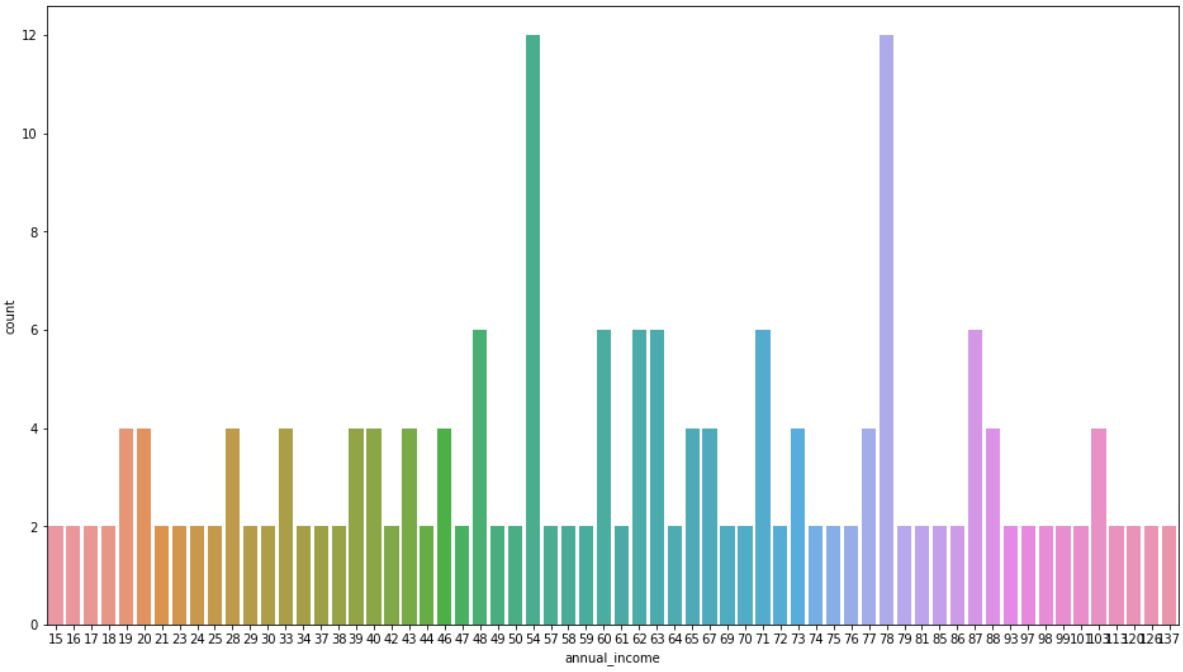


In [17]:
```
plt.figure(figsize = (16,9))
sns.countplot(customer_dataset.annual_income)
```

```
C:\Users\arivu\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_
decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and passing
other arguments without an explicit keyword will result in an error or misinterpre
tation.
  warnings.warn(
```

Out[17]:

```
<AxesSubplot:xlabel='annual_income', ylabel='count'>
```



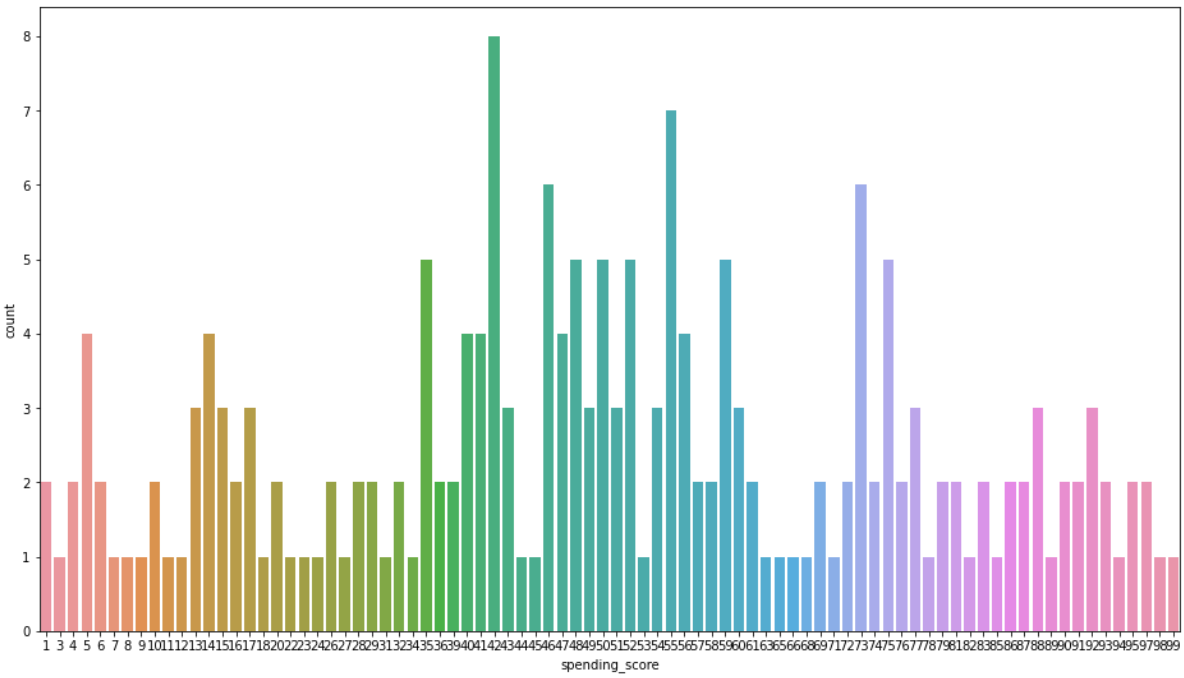
In [18]:

```
plt.figure(figsize =(16,9))
sns.countplot(customer_dataset.spending_score)
```

```
C:\Users\arivu\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_
decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and passing
other arguments without an explicit keyword will result in an error or misinterpre
tation.
  warnings.warn(
```

Out[18]:

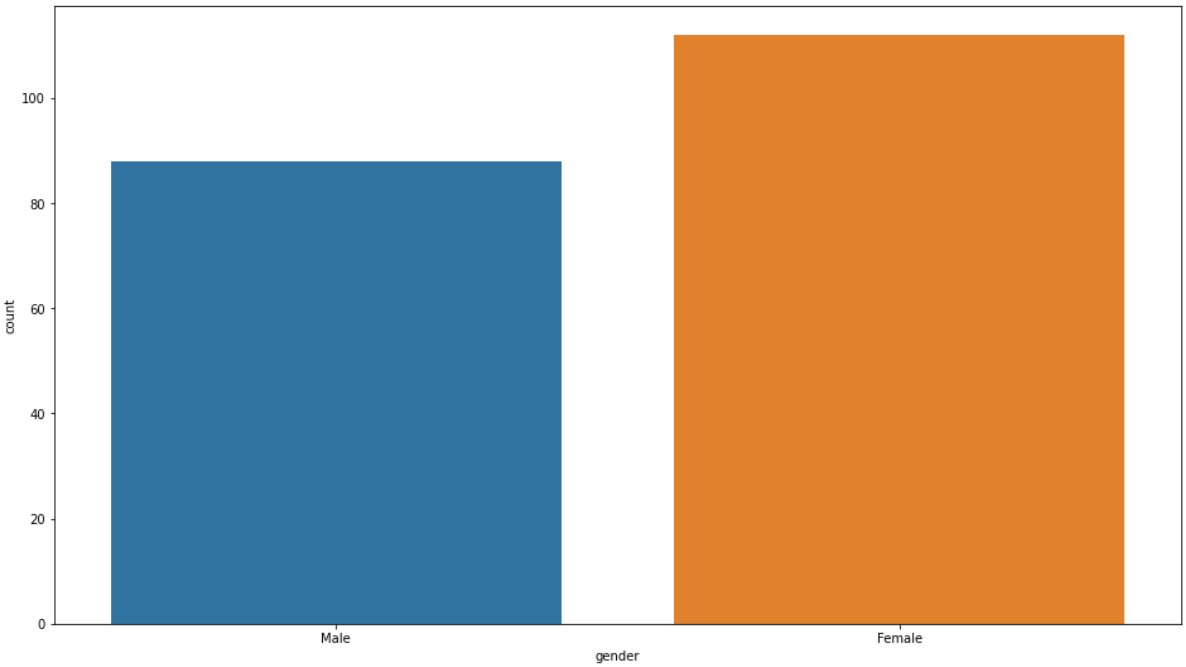
```
<AxesSubplot:xlabel='spending_score', ylabel='count'>
```



In [19]:
```
plt.figure(figsize= (16,9))
sns.countplot(customer_dataset.gender)
```

```
C:\Users\arivu\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[19]:
```
<AxesSubplot:xlabel='gender', ylabel='count'>
```



Choosing the Annual Income Column and Spending Columns

In [20]:
```
customer_dataset.head()
```

Out[20]:

	customer_id	gender	age	annual_income	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

In [21]:
```
customer_dataset.columns
```

Out[21]:
```
Index(['customer_id', 'gender', 'age', 'annual_income', 'spending_score'], dtype='object')
```

In [22]:
```
#using iloc for picking 3rd and 4th columns

x=customer_dataset.iloc[:,[3,4]].values
```

In [23]:
```
print(x)
```

```
[[ 15  39]
 [ 15  81]
 [ 16   6]
 [ 16  77]
 [ 17  40]
 [ 17  76]
 [ 18   6]
 [ 18  94]
 [ 19   3]
 [ 19  72]
 [ 19  14]
 [ 19  99]
 [ 20  15]
 [ 20  77]
 [ 20  13]
 [ 20  79]
 [ 21  35]
 [ 21  66]
 [ 23  29]
 [ 23  98]
 [ 24  35]
 [ 24  73]
 [ 25   5]
 [ 25  73]
 [ 28  14]
 [ 28  82]
 [ 28  32]
 [ 28  61]
 [ 29  31]
 [ 29  87]
 [ 30   4]
 [ 30  73]
 [ 33   4]
 [ 33  92]
 [ 33  14]
 [ 33  81]
 [ 34  17]
 [ 34  73]
 [ 37  26]
 [ 37  75]
 [ 38  35]
 [ 38  92]
 [ 39  36]
 [ 39  61]
 [ 39  28]
 [ 39  65]
 [ 40  55]
 [ 40  47]
 [ 40  42]
 [ 40  42]
 [ 42  52]
 [ 42  60]
 [ 43  54]
 [ 43  60]
 [ 43  45]
 [ 43  41]
 [ 44  50]
 [ 44  46]
 [ 46  51]
 [ 46  46]
 [ 46  56]
 [ 46  55]
 [ 47  52]
 [ 47  59]
```

```
[ 48  51]
[ 48  59]
[ 48  50]
[ 48  48]
[ 48  59]
[ 48  47]
[ 49  55]
[ 49  42]
[ 50  49]
[ 50  56]
[ 54  47]
[ 54  54]
[ 54  53]
[ 54  48]
[ 54  52]
[ 54  42]
[ 54  51]
[ 54  55]
[ 54  41]
[ 54  44]
[ 54  57]
[ 54  46]
[ 57  58]
[ 57  55]
[ 58  60]
[ 58  46]
[ 59  55]
[ 59  41]
[ 60  49]
[ 60  40]
[ 60  42]
[ 60  52]
[ 60  47]
[ 60  50]
[ 61  42]
[ 61  49]
[ 62  41]
[ 62  48]
[ 62  59]
[ 62  55]
[ 62  56]
[ 62  42]
[ 63  50]
[ 63  46]
[ 63  43]
[ 63  48]
[ 63  52]
[ 63  54]
[ 64  42]
[ 64  46]
[ 65  48]
[ 65  50]
[ 65  43]
[ 65  59]
[ 67  43]
[ 67  57]
[ 67  56]
[ 67  40]
[ 69  58]
[ 69  91]
[ 70  29]
[ 70  77]
[ 71  35]
[ 71  95]
```

```
[ 71  11]
[ 71  75]
[ 71   9]
[ 71  75]
[ 72  34]
[ 72  71]
[ 73   5]
[ 73  88]
[ 73   7]
[ 73  73]
[ 74  10]
[ 74  72]
[ 75   5]
[ 75  93]
[ 76  40]
[ 76  87]
[ 77  12]
[ 77  97]
[ 77  36]
[ 77  74]
[ 78  22]
[ 78  90]
[ 78  17]
[ 78  88]
[ 78  20]
[ 78  76]
[ 78  16]
[ 78  89]
[ 78   1]
[ 78  78]
[ 78   1]
[ 78  73]
[ 79  35]
[ 79  83]
[ 81   5]
[ 81  93]
[ 85  26]
[ 85  75]
[ 86  20]
[ 86  95]
[ 87  27]
[ 87  63]
[ 87  13]
[ 87  75]
[ 87  10]
[ 87  92]
[ 88  13]
[ 88  86]
[ 88  15]
[ 88  69]
[ 93  14]
[ 93  90]
[ 97  32]
[ 97  86]
[ 98  15]
[ 98  88]
[ 99  39]
[ 99  97]
[101  24]
[101  68]
[103  17]
[103  85]
[103  23]
[103  69]
```

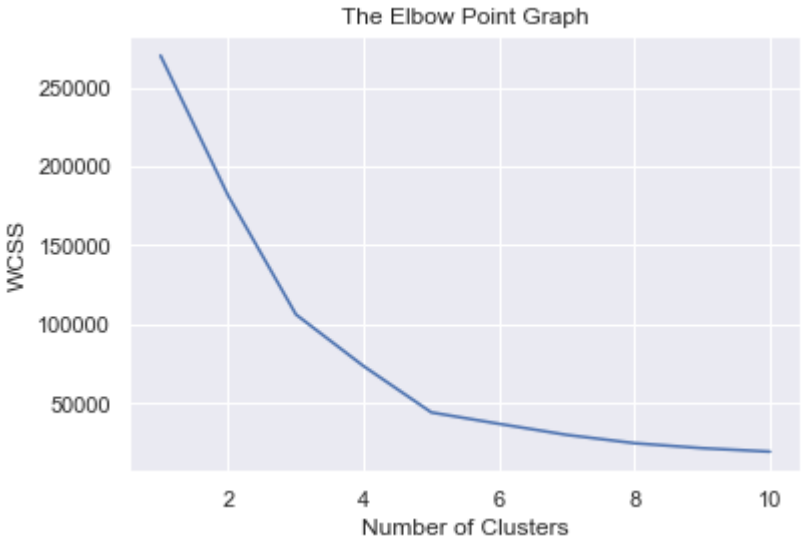
```
[113   8]
[113  91]
[120  16]
[120  79]
[126  28]
[126  74]
[137  18]
[137  83]]
```

Choosing the number of clusters

WCSS : within Clusters Sum of Squares

In [25]:
```
#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_)
```

In [36]:
```
#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

In [28]:
```
kmeans = KMeans(n_clusters=5, init = 'k-means++', random_state=0)

#retun a lable for each data point based on their clusters
```

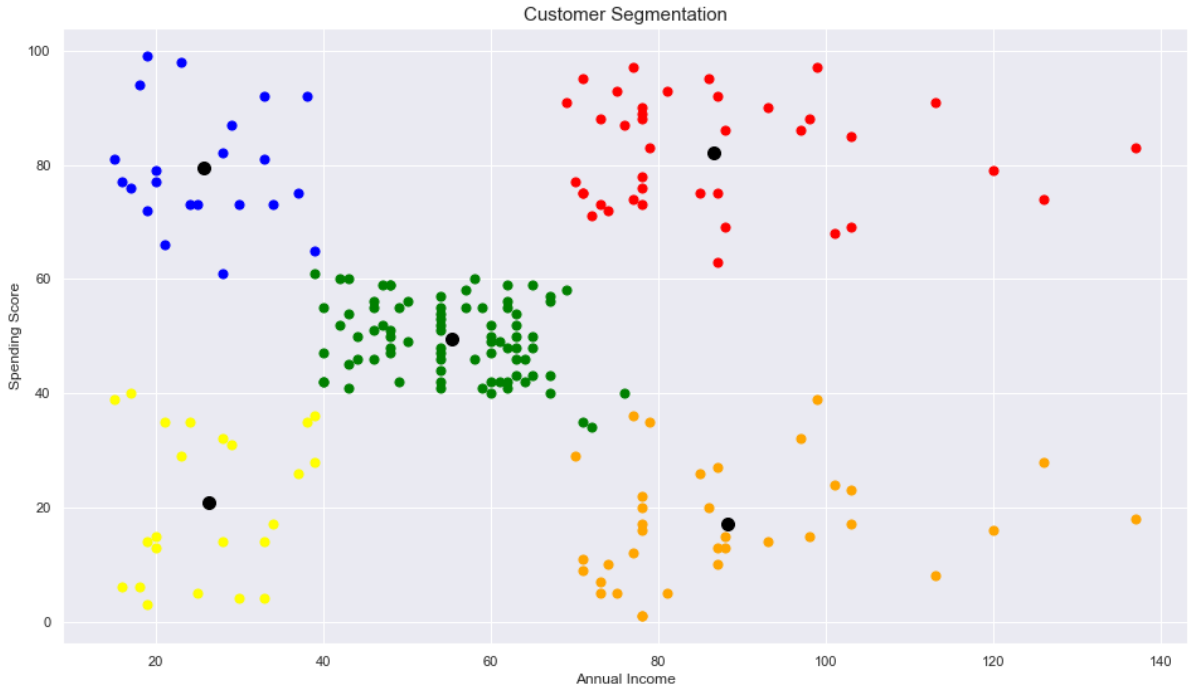
In [29]:
```
y = kmeans.fit_predict(x)

print(y)
```

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

Plotting data in graph

In [35]:
```
#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 2')
plt.scatter(x[y==2,0], x[y==2,1], s = 50, c = 'red', label = 'Cluster 3')
plt.scatter(x[y==3,0], x[y==3,1], s = 50, c = 'blue', label = 'Cluster 4')
plt.scatter(x[y==4,0], x[y==4,1], s = 50, c = 'yellow', label = 'Cluster 5')

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

Out[35]:
```
<matplotlib.collections.PathCollection at 0x178467833a0>
```



In [ ]:
```
```