In [1]:

```
#Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot  as plt
%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
import sklearn
from sklearn.cluster import KMeans
from sklearn.preprocessing import scale
from sklearn import datasets
import seaborn as sns
```

In [2]:

```
#reading data
df=pd.read_csv('MallCustomers.csv')
df
```

Out[2]:

	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 × 5 columns

In [3]:

```
df.describe()
```

Out[3]:

	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 [4]:

```
#to check if there are any null values
df.isnull().sum()
```

```
df.isnull().sum()
```

Out[4]:

```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [5]:

```
#data mapping
df_map=df.copy()
df_map['Gender']=df_map['Gender'].map({'Male':1,'Female':0})
df_map
```

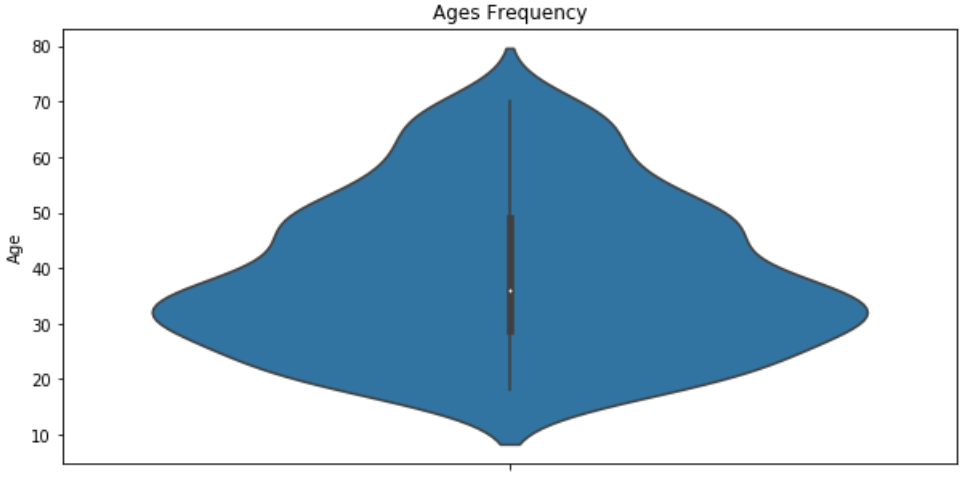
Out[5]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	1	19	15	39
1	2	1	21	15	81
2	3	0	20	16	6
3	4	0	23	16	77
4	5	0	31	17	40
...	...	...	...	...	...
195	196	0	35	120	79
196	197	0	45	126	28
197	198	1	32	126	74
198	199	1	32	137	18
199	200	1	30	137	83

200 rows × 5 columns

In [6]:

```
#age frequency of customers using voilinplot
plt.figure(figsize=(10,5))
plt.title("Ages Frequency")
sns.axes_style("dark")
sns.violinplot(y=df["Age"])
plt.show()
```



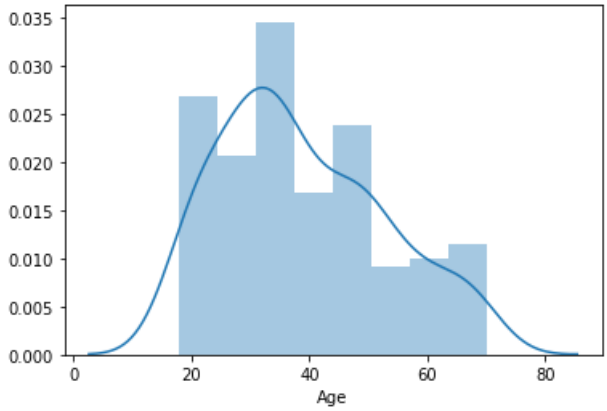
In [7]:

```
#Normal distribution plot for ages
```

```
sns.distplot(df['Age'])
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1910975e148>
```

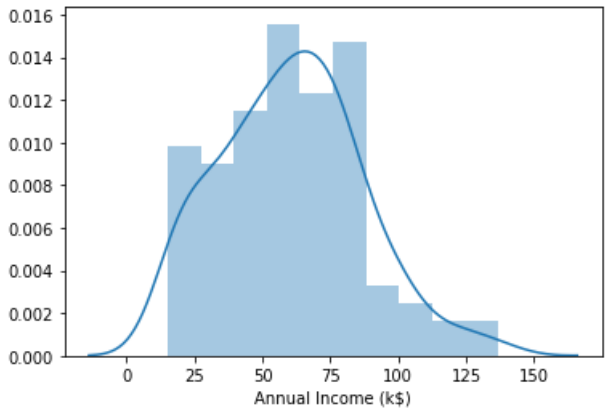


In [8]:

```
#Normal distribution plot for Annual Income
sns.distplot(df['Annual Income (k$)'])
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x19109807fc8>
```

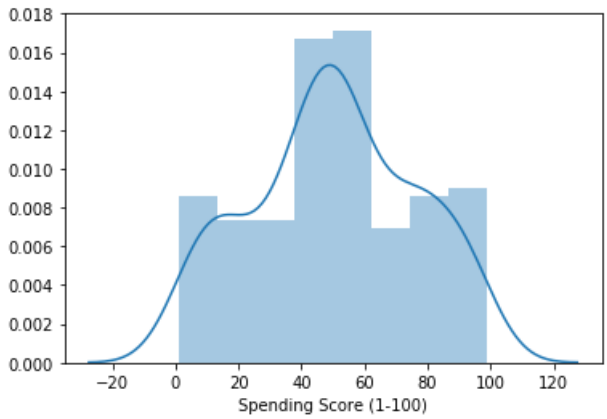


In [9]:

```
#Normal distribution plot for Spending Score
sns.distplot(df['Spending Score (1-100)'])
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1910989a908>
```

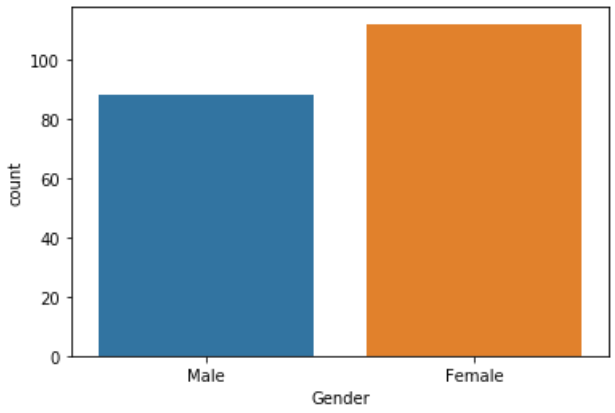


In [10]:

```
#Distribution of male and female
x=df.copy()
sns.countplot(x='Gender',data=df)
```

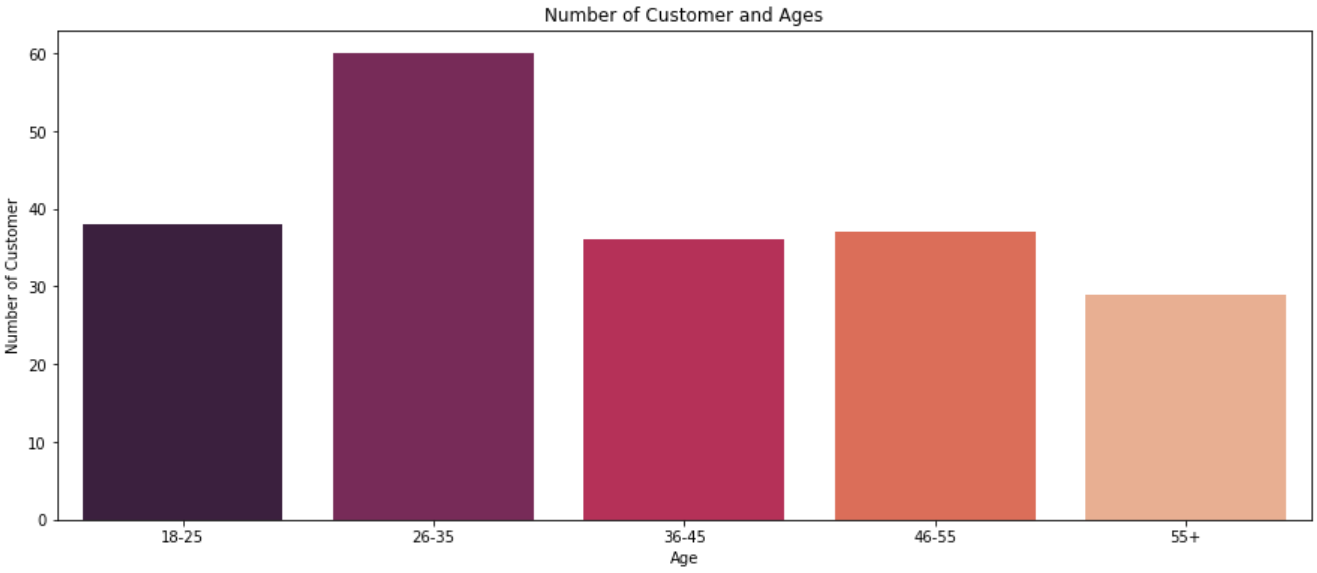
Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x19109925b48>
```



In [11]:

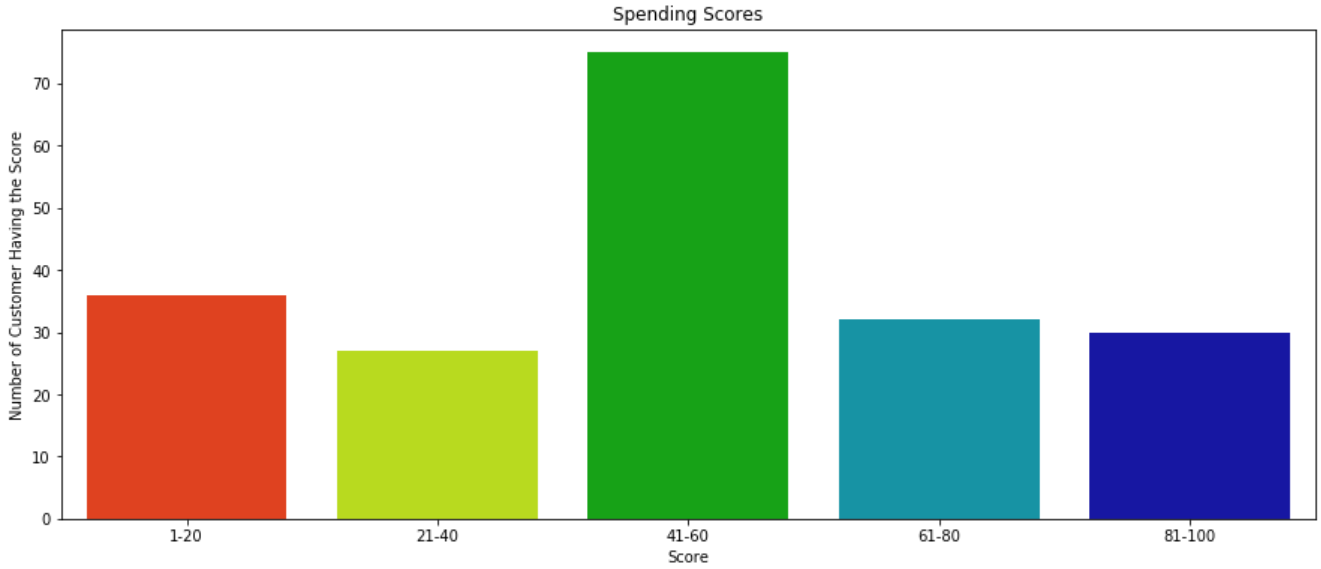
```
#Bar plot to check the distribution of number of customers in each age group
age18_25 = df.Age[(df.Age <= 25) & (df.Age >= 18)]
age26_35 = df.Age[(df.Age <= 35) & (df.Age >= 26)]
age36_45 = df.Age[(df.Age <= 45) & (df.Age >= 36)]
age46_55 = df.Age[(df.Age <= 55) & (df.Age >= 46)]
age55above = df.Age[df.Age >= 56]

x = ["18-25","26-35","36-45","46-55","55+"]
y = [len(age18_25.values),len(age26_35.values),len(age36_45.values),len(age46_55.values),len(age55a
bove.values)]

plt.figure(figsize=(15,6))
sns.barplot(x=x, y=y, palette="rocket")
plt.title("Number of Customer and Ages")
plt.xlabel("Age")
plt.ylabel("Number of Customer")
plt.show()
```



In [12]:

```
#Bar plot to visualize the number of customers according to their spending scores
```

```
ss1_20 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 1) & (df["Spending Score
(1-100)"] <= 20)]
ss21_40 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 21) & (df["Spending Score
(1-100)"] <= 40)]
ss41_60 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 41) & (df["Spending Score
(1-100)"] <= 60)]
ss61_80 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 61) & (df["Spending Score
(1-100)"] <= 80)]
ss81_100 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 81) & (df["Spending Score
(1-100)"] <= 100)]

ssx = ["1-20", "21-40", "41-60", "61-80", "81-100"]
ssy = [len(ss1_20.values), len(ss21_40.values), len(ss41_60.values), len(ss61_80.values), len(ss81_
100.values)]

plt.figure(figsize=(15,6))
sns.barplot(x=ssx, y=ssy, palette="nipy_spectral_r")
plt.title("Spending Scores")
plt.xlabel("Score")
plt.ylabel("Number of Customer Having the Score")
plt.show()
```

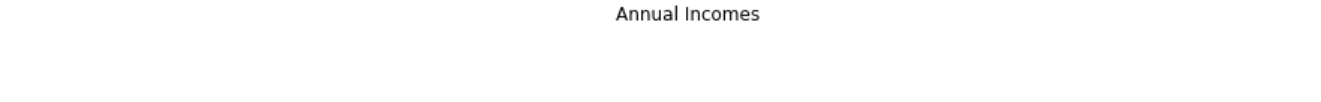


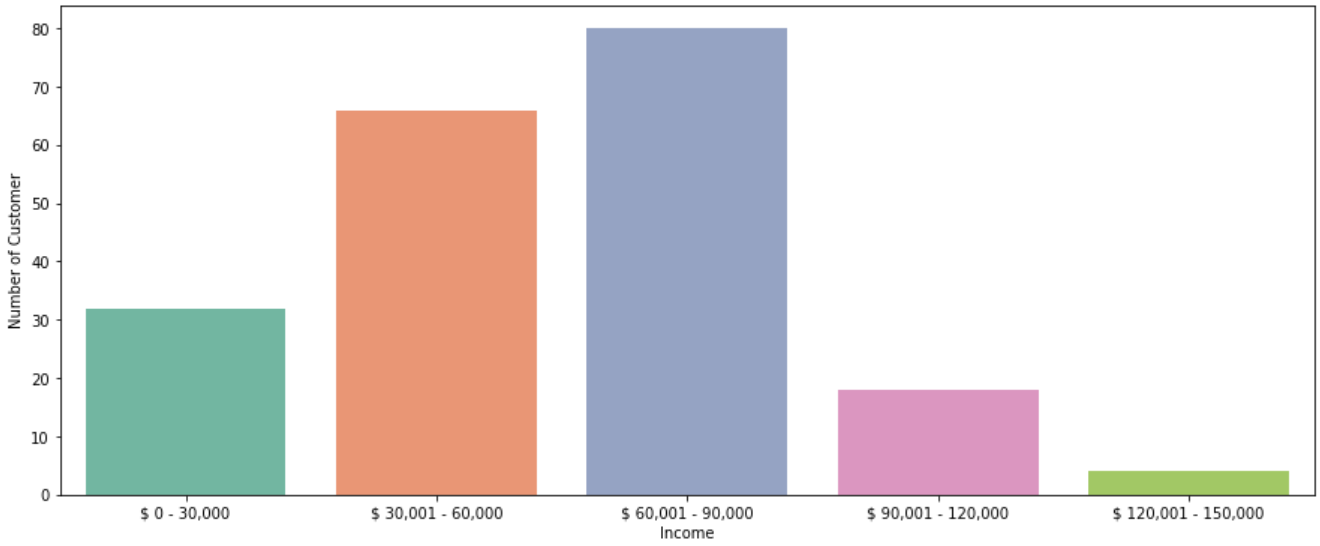
In [13]:

```
#Bar plot to visualize the number of customers according to their annual income
ai0_30 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 0) & (df["Annual Income (k$)"] <= 30
)]
ai31_60 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 31) & (df["Annual Income (k$)"] <=
60)]
ai61_90 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 61) & (df["Annual Income (k$)"] <=
90)]
ai91_120 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 91) & (df["Annual Income (k$)"] <=
120)]
ai121_150 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 121) & (df["Annual Income (k$)"]
<= 150)]

aix = ["$ 0 - 30,000", "$ 30,001 - 60,000", "$ 60,001 - 90,000", "$ 90,001 - 120,000", "$ 120,001
- 150,000"]
aiy = [len(ai0_30.values), len(ai31_60.values), len(ai61_90.values), len(ai91_120.values), len(ai12
1_150.values)]

plt.figure(figsize=(15,6))
sns.barplot(x=aix, y=aiy, palette="Set2")
plt.title("Annual Incomes")
plt.xlabel("Income")
plt.ylabel("Number of Customer")
```

Out[13]:

```
Text(0, 0.5, 'Number of Customer')
```





In [15]:

```
x = df_map.iloc[:, [1,2,3,4]].values
x
```

Out[15]:

```
array([[  1,  19,  15,  39],
       [  1,  21,  15,  81],
       [  0,  20,  16,   6],
       [  0,  23,  16,  77],
       [  0,  31,  17,  40],
       [  0,  22,  17,  76],
       [  0,  35,  18,   6],
       [  0,  23,  18,  94],
       [  1,  64,  19,   3],
       [  0,  30,  19,  72],
       [  1,  67,  19,  14],
       [  0,  35,  19,  99],
       [  0,  58,  20,  15],
       [  0,  24,  20,  77],
       [  1,  37,  20,  13],
       [  1,  22,  20,  79],
       [  0,  35,  21,  35],
       [  1,  20,  21,  66],
       [  1,  52,  23,  29],
       [  0,  35,  23,  98],
       [  1,  35,  24,  35],
       [  1,  25,  24,  73],
       [  0,  46,  25,   5],
       [  1,  31,  25,  73],
       [  0,  54,  28,  14],
       [  1,  29,  28,  82],
       [  0,  45,  28,  32],
       [  1,  35,  28,  61],
       [  0,  40,  29,  31],
       [  0,  23,  29,  87],
       [  1,  60,  30,   4],
       [  0,  21,  30,  73],
       [  1,  53,  33,   4],
       [  1,  18,  33,  92],
       [  0,  49,  33,  14],
       [  0,  21,  33,  81],
       [  0,  42,  34,  17],
       [  0,  30,  34,  73],
       [  0,  36,  37,  26],
       [  0,  20,  37,  75],
       [  0,  65,  38,  35],
       [  1,  24,  38,  92],
       [  1,  48,  39,  36],
       [  0,  31,  39,  61],
       [  0,  49,  39,  28],
       [  0,  24,  39,  65],
       [  0,  50,  40,  55],
       [  0,  27,  40,  47],
```

```
[  0,  27,  45,  47],
[  0,  29,  40,  42],
[  0,  31,  40,  42],
[  0,  49,  42,  52],
[  1,  33,  42,  60],
[  0,  31,  43,  54],
[  1,  59,  43,  60],
[  0,  50,  43,  45],
[  1,  47,  43,  41],
[  0,  51,  44,  50],
[  1,  69,  44,  46],
[  0,  27,  46,  51],
[  1,  53,  46,  46],
[  1,  70,  46,  56],
[  1,  19,  46,  55],
[  0,  67,  47,  52],
[  0,  54,  47,  59],
[  1,  63,  48,  51],
[  1,  18,  48,  59],
[  0,  43,  48,  50],
[  0,  68,  48,  48],
[  1,  19,  48,  59],
[  0,  32,  48,  47],
[  1,  70,  49,  55],
[  0,  47,  49,  42],
[  0,  60,  50,  49],
[  0,  60,  50,  56],
[  1,  59,  54,  47],
[  1,  26,  54,  54],
[  0,  45,  54,  53],
[  1,  40,  54,  48],
[  0,  23,  54,  52],
[  0,  49,  54,  42],
[  1,  57,  54,  51],
[  1,  38,  54,  55],
[  1,  67,  54,  41],
[  0,  46,  54,  44],
[  0,  21,  54,  57],
[  1,  48,  54,  46],
[  0,  55,  57,  58],
[  0,  22,  57,  55],
[  0,  34,  58,  60],
[  0,  50,  58,  46],
[  0,  68,  59,  55],
[  1,  18,  59,  41],
[  1,  48,  60,  49],
[  0,  40,  60,  40],
[  0,  32,  60,  42],
[  1,  24,  60,  52],
[  0,  47,  60,  47],
[  0,  27,  60,  50],
[  1,  48,  61,  42],
[  1,  20,  61,  49],
[  0,  23,  62,  41],
[  0,  49,  62,  48],
[  1,  67,  62,  59],
[  1,  26,  62,  55],
[  1,  49,  62,  56],
[  0,  21,  62,  42],
[  0,  66,  63,  50],
[  1,  54,  63,  46],
[  1,  68,  63,  43],
[  1,  66,  63,  48],
[  1,  65,  63,  52],
[  0,  19,  63,  54],
[  0,  38,  64,  42],
[  1,  19,  64,  46],
[  0,  18,  65,  48],
[  0,  19,  65,  50],
[  0,  63,  65,  43],
[  0,  49,  65,  59],
[  0,  51,  67,  43],
[  0,  50,  67,  57],
[  1,  27,  67,  56],
[  0,  38,  67,  40],
[  0,  40,  69,  58],
[  1,  39,  69,  91],
[  0,  23,  70,  29],
```

```
[  0,  25,  70,  29],
[  0,  31,  70,  77],
[  1,  43,  71,  35],
[  1,  40,  71,  95],
[  1,  59,  71,  11],
[  1,  38,  71,  75],
[  1,  47,  71,   9],
[  1,  39,  71,  75],
[  0,  25,  72,  34],
[  0,  31,  72,  71],
[  1,  20,  73,   5],
[  0,  29,  73,  88],
[  0,  44,  73,   7],
[  1,  32,  73,  73],
[  1,  19,  74,  10],
[  0,  35,  74,  72],
[  0,  57,  75,   5],
[  1,  32,  75,  93],
[  0,  28,  76,  40],
[  0,  32,  76,  87],
[  1,  25,  77,  12],
[  1,  28,  77,  97],
[  1,  48,  77,  36],
[  0,  32,  77,  74],
[  0,  34,  78,  22],
[  1,  34,  78,  90],
[  1,  43,  78,  17],
[  1,  39,  78,  88],
[  0,  44,  78,  20],
[  0,  38,  78,  76],
[  0,  47,  78,  16],
[  0,  27,  78,  89],
[  1,  37,  78,   1],
[  0,  30,  78,  78],
[  1,  34,  78,   1],
[  0,  30,  78,  73],
[  0,  56,  79,  35],
[  0,  29,  79,  83],
[  1,  19,  81,   5],
[  0,  31,  81,  93],
[  1,  50,  85,  26],
[  0,  36,  85,  75],
[  1,  42,  86,  20],
[  0,  33,  86,  95],
[  0,  36,  87,  27],
[  1,  32,  87,  63],
[  1,  40,  87,  13],
[  1,  28,  87,  75],
[  1,  36,  87,  10],
[  1,  36,  87,  92],
[  0,  52,  88,  13],
[  0,  30,  88,  86],
[  1,  58,  88,  15],
[  1,  27,  88,  69],
[  1,  59,  93,  14],
[  1,  35,  93,  90],
[  0,  37,  97,  32],
[  0,  32,  97,  86],
[  1,  46,  98,  15],
[  0,  29,  98,  88],
[  0,  41,  99,  39],
[  1,  30,  99,  97],
[  0,  54, 101,  24],
[  1,  28, 101,  68],
[  0,  41, 103,  17],
[  0,  36, 103,  85],
[  0,  34, 103,  23],
[  0,  32, 103,  69],
[  1,  33, 113,   8],
[  0,  38, 113,  91],
[  0,  47, 120,  16],
[  0,  35, 120,  79],
[  0,  45, 126,  28],
[  1,  32, 126,  74],
[  1,  32, 137,  18],
[  1,  30, 137,  83]], dtype=int64)
```

In [21]:

```
#considering k=3
kmeans3 = KMeans(n_clusters=3)
y_kmeans3 = kmeans3.fit_predict(x)
print(y_kmeans3)
```

```
[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 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 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 0 2 0 2 0 2
 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0]
```

In [22]:

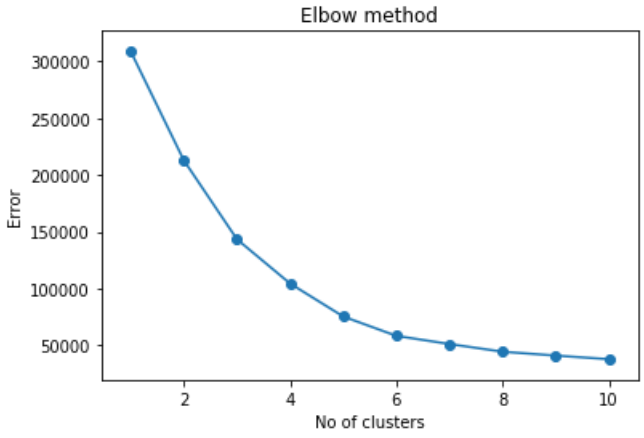
```
kmeans3.cluster_centers_
```

Out[22]:

```
array([[ 0.46153846, 32.69230769, 86.53846154, 82.12820513],
       [ 0.40650407, 40.32520325, 44.15447154, 49.82926829],
       [ 0.52631579, 40.39473684, 87.        , 18.63157895]])
```

In [27]:

```
#elbow method
wcss =[]
for i in range(1, 11):
    kmeans3 = KMeans(n_clusters = i).fit(x)
    kmeans3.fit(x)
    wcss.append(kmeans3.inertia_)
plt.plot(range(1, 11), wcss)
plt.scatter(range(1,11),wcss)
plt.title('Elbow method')
plt.xlabel('No of clusters')
plt.ylabel('Error')
plt.show()
```



In [16]:

```
#considering k=5
kmeans5 = KMeans(n_clusters=5)
y_kmeans5 = kmeans5.fit_predict(x)
print(y_kmeans5)
```

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

In [17]:

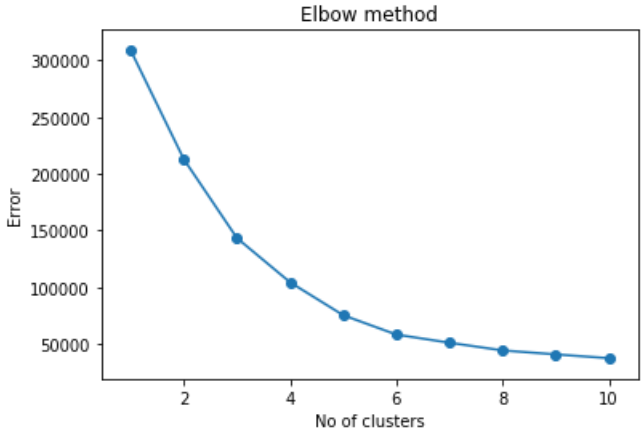
```
kmeans5.cluster_centers_
```

Out[17]:

```
array([[ 0.39130435, 25.52173913, 26.30434783, 78.56521739],
       [ 0.52777778, 40.66666667, 87.75      , 17.58333333],
       [ 0.46153846, 32.69230769, 86.53846154, 82.12820513],
       [ 0.41772152, 43.08860759, 55.29113924, 49.56962025],
       [ 0.39130435, 45.2173913 , 26.30434783, 20.91304348]])
```

In [28]:

```
#Elbow method
wcss =[]
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i).fit(x)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.scatter(range(1,11),wcss)
plt.title('Elbow method')
plt.xlabel('No of clusters')
plt.ylabel('Error')
plt.show()
```

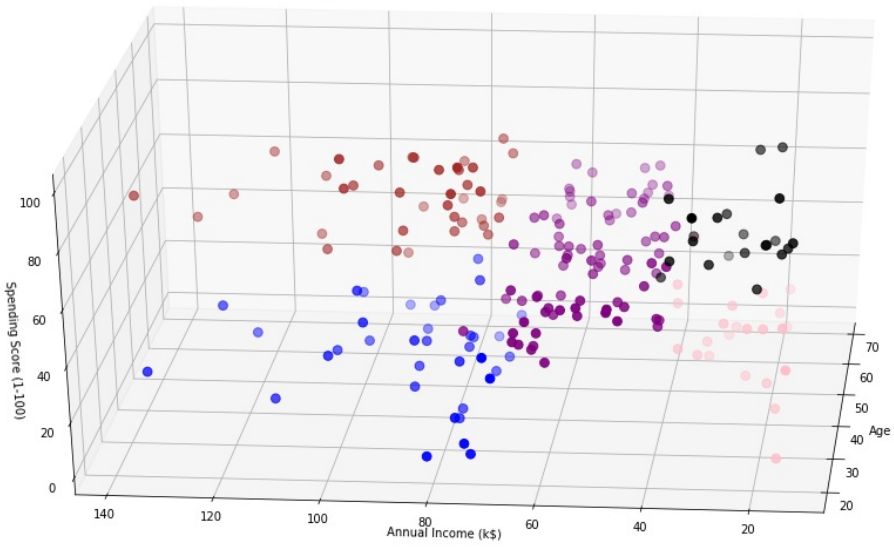


Optimum k value lies between 4 and 6 (say k=5)

In [35]:

```
x = df_map.iloc[:, [1,2,3,4]].values
km = KMeans(n_clusters=5)
clusters = km.fit_predict(df_map.iloc[:,1:])

df_map["label"] = clusters

from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df.Age[df_map.label == 0], df_map["Annual Income (k$)"][df_map.label == 0], df_map["Spen
ding Score (1-100)"][df_map.label == 0], c='black', s=60)
ax.scatter(df.Age[df_map.label == 1], df_map["Annual Income (k$)"][df_map.label == 1], df_map["Spen
ding Score (1-100)"][df_map.label == 1], c='blue', s=60)
ax.scatter(df.Age[df_map.label == 2], df_map["Annual Income (k$)"][df_map.label == 2], df_map["Spen
ding Score (1-100)"][df_map.label == 2], c='purple', s=60)
ax.scatter(df.Age[df_map.label == 3], df_map["Annual Income (k$)"][df_map.label == 3], df_map["Spen
ding Score (1-100)"][df_map.label == 3], c='brown', s=60)
ax.scatter(df.Age[df_map.label == 4], df_map["Annual Income (k$)"][df_map.label == 4], df_map["Spen
ding Score (1-100)"][df_map.label == 4], c='pink', s=60)
ax.view_init(30, 185)
```

```
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel('Spending Score (1-100)')
plt.show()
```



In [ ]:

```
```