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

In [2]:
```
#Read Dataset
df = pd.read_csv("Mall_Customers.csv")
df
```

Out[2]:

	CustomerID	Genre	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

Exploratory Data Analysis (EDA)

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

Out[3]:

	CustomerID	Genre	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]:
```
#Renaming the genre column as gender as it is mispelled
df.rename(columns={'Genre' : 'Gender'},inplace= True)
```

In [5]:
```
df.head()
```

Out[5]:

	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 [6]:
```
df.shape
```

Out[6]:
```
(200, 5)
```

In [7]:
```
#Descriptive statistics
df.describe()
```

Out[7]:

	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 [8]:
```
df.dtypes
```

Out[8]:
```
CustomerID                int64
Gender                   object
Age                       int64
Annual Income (k$)        int64
Spending Score (1-100)    int64
dtype: object
```

In [9]:
```
#Checking for null values
df.isnull().sum()
```

Out[9]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [10]:
```
df.drop(['CustomerID'],axis =1,inplace=True)
```

In [11]:
```
df
```

Out[11]:

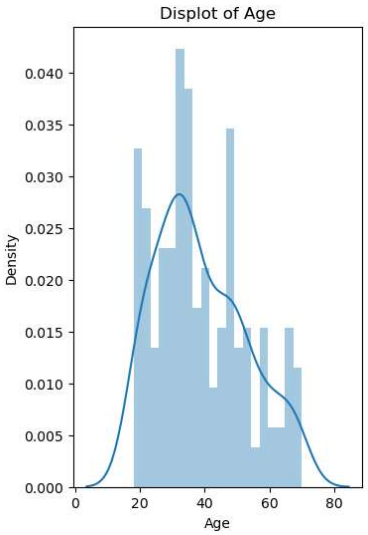
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40
...	...	...	...	...
195	Female	35	120	79
196	Female	45	126	28
197	Male	32	126	74
198	Male	32	137	18
199	Male	30	137	83

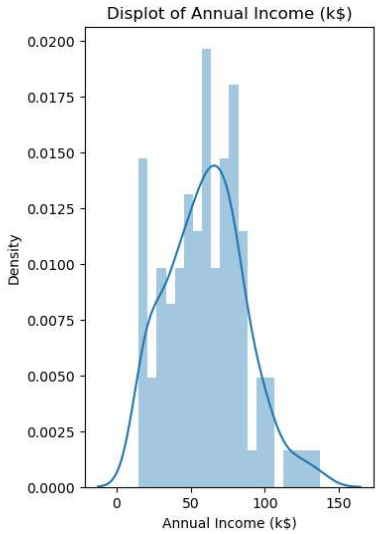
200 rows × 4 columns

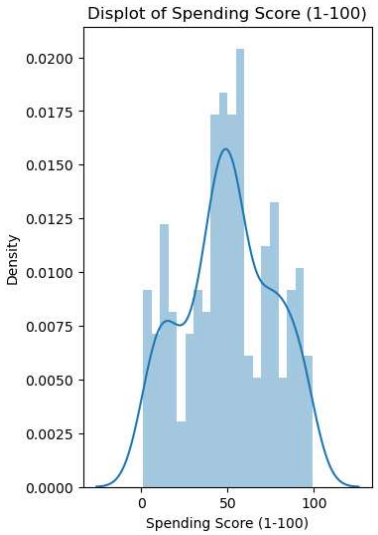
In [12]:

```
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("Displot of {}".format(x))
plt.show()
```

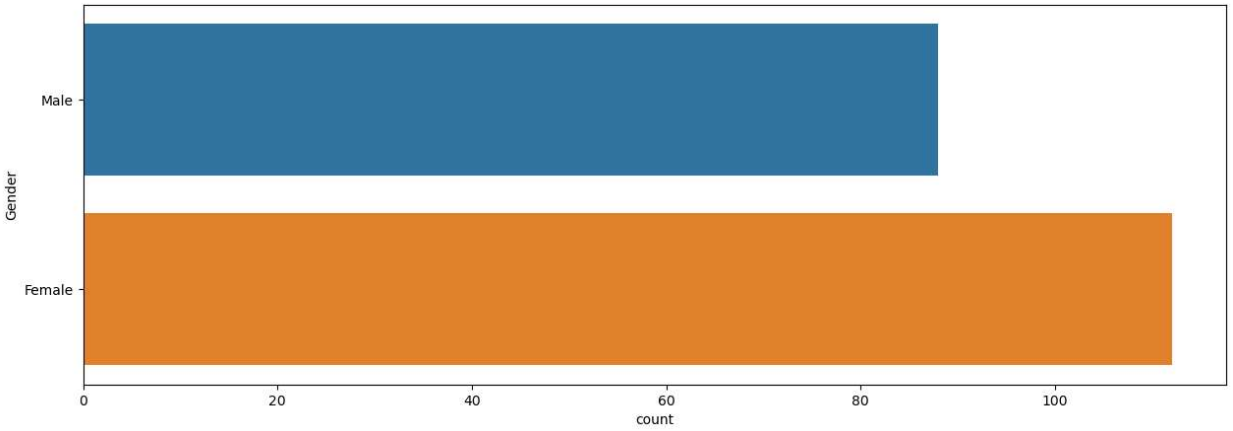
```
D:\Users\shrey\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
D:\Users\shrey\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
D:\Users\shrey\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



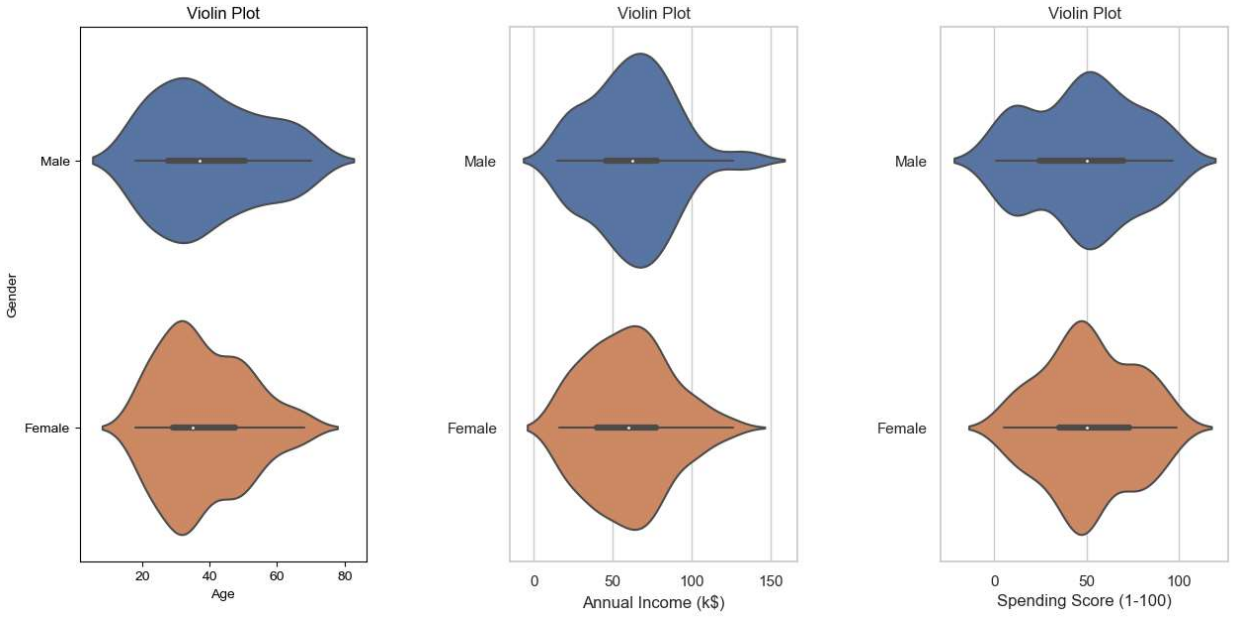




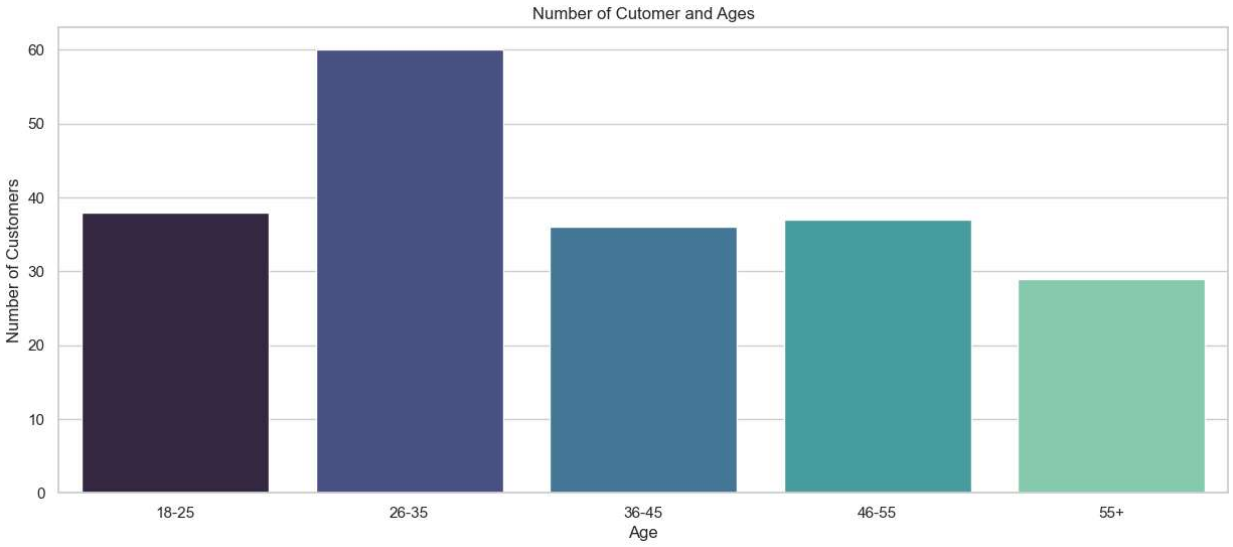
In [13]:
```
#comparison between male and female
plt.figure(figsize=(15,5))
sns.countplot(y='Gender',data = df)
plt.show()
```



In [14]:
```
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()
```

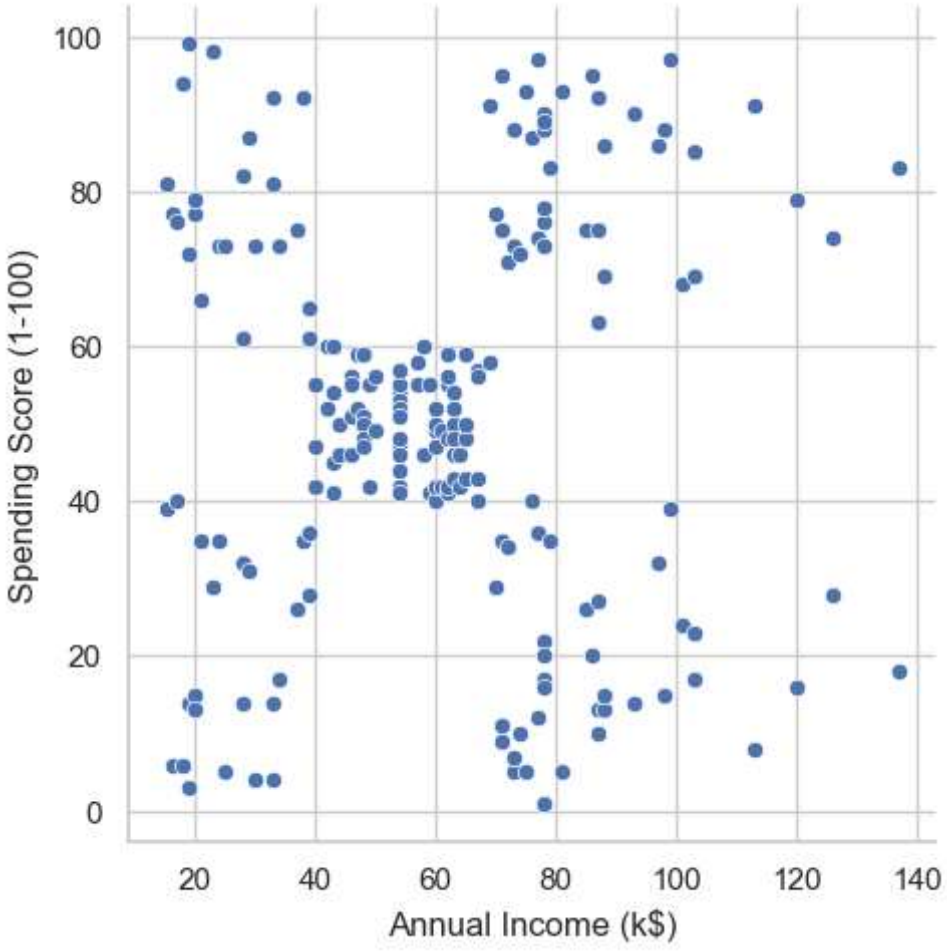


In [15]:
```
age_18_25 = df.Age[(df.Age >= 18) & (df.Age <= 25)]
age_26_35 = df.Age[(df.Age >= 26) & (df.Age <= 35)]
age_36_45 = df.Age[(df.Age >= 36) & (df.Age <= 45)]
age_46_55 = df.Age[(df.Age >= 46) & (df.Age <= 55)]
age_55above = df.Age[(df.Age >= 56)]

agex = ["18-25","26-35","36-45","46-55","55+"]
agey = [len(age_18_25.values),len(age_26_35.values),len(age_36_45.values),len(age_46_5

plt.figure(figsize=(15,6))
sns.barplot(x=agex,y=agey, palette ="mako")
plt.title("Number of Cutomer and Ages")
plt.xlabel("Age")
plt.ylabel("Number of Customers")
plt.show()
```

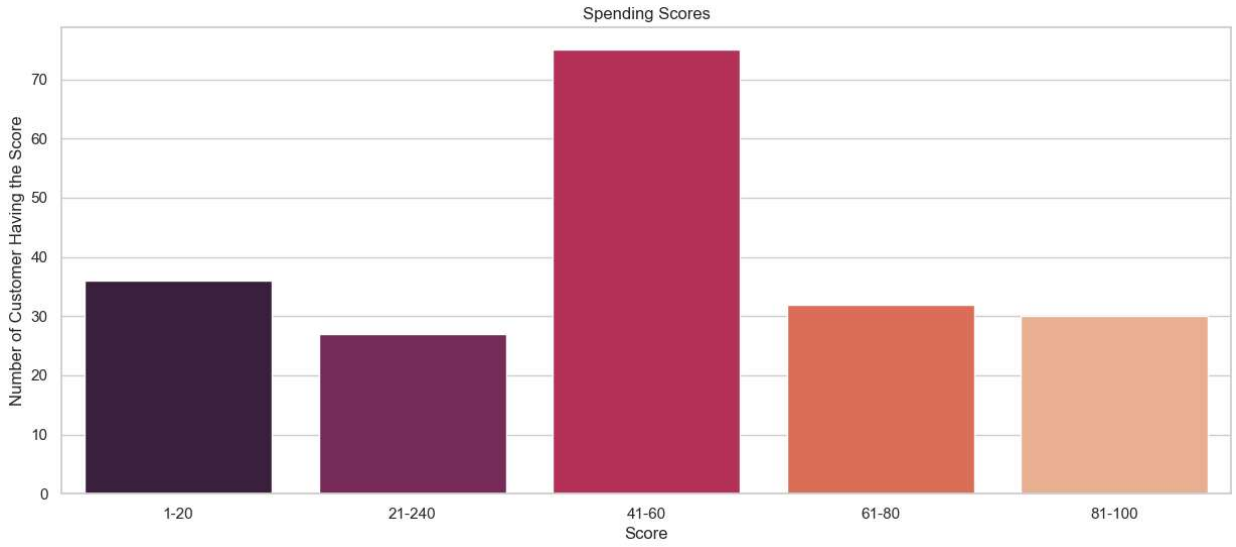


In [16]:
```
sns.relplot(x="Annual Income (k$)",y ="Spending Score (1-100)",data = df)
```

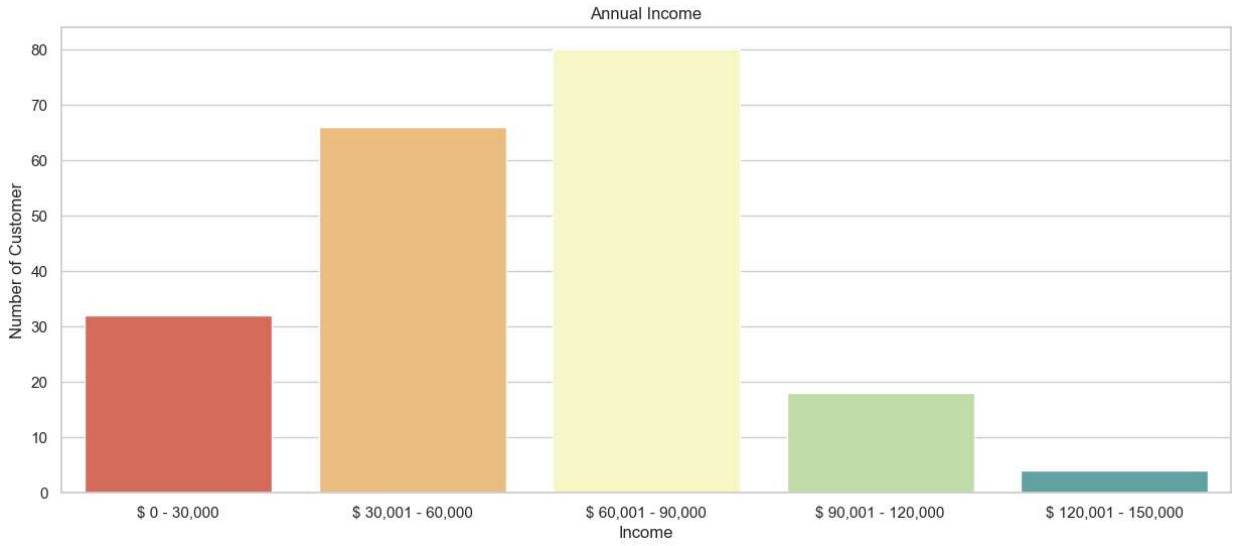
Out[16]:
```
<seaborn.axisgrid.FacetGrid at 0x1f464590820>
```



In [17]:
```
ss_1_20 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >=1) & (df["Spend
ss_21_40 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >=21) & (df["Spe
ss_41_60 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >=41) & (df["Spe
ss_61_80 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >=61) & (df["Spe
ss_81_100 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 81) & (df["S

ssx = ["1-20","21-240","41-60","61-80","81-100"]
ssy = [len(ss_1_20.values),len(ss_21_40.values),len(ss_41_60.values),len(ss_61_80.valu

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



In [18]:
```
ai0_30 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 0) & (df["Annual Income
ai31_60 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >=31) & (df["Annual Incom
ai61_90 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >=61) & (df["Annual Incom
ai91_120 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >=91) & (df["Annual Incc
ai121_150 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 121) & (df["Annual I

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

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

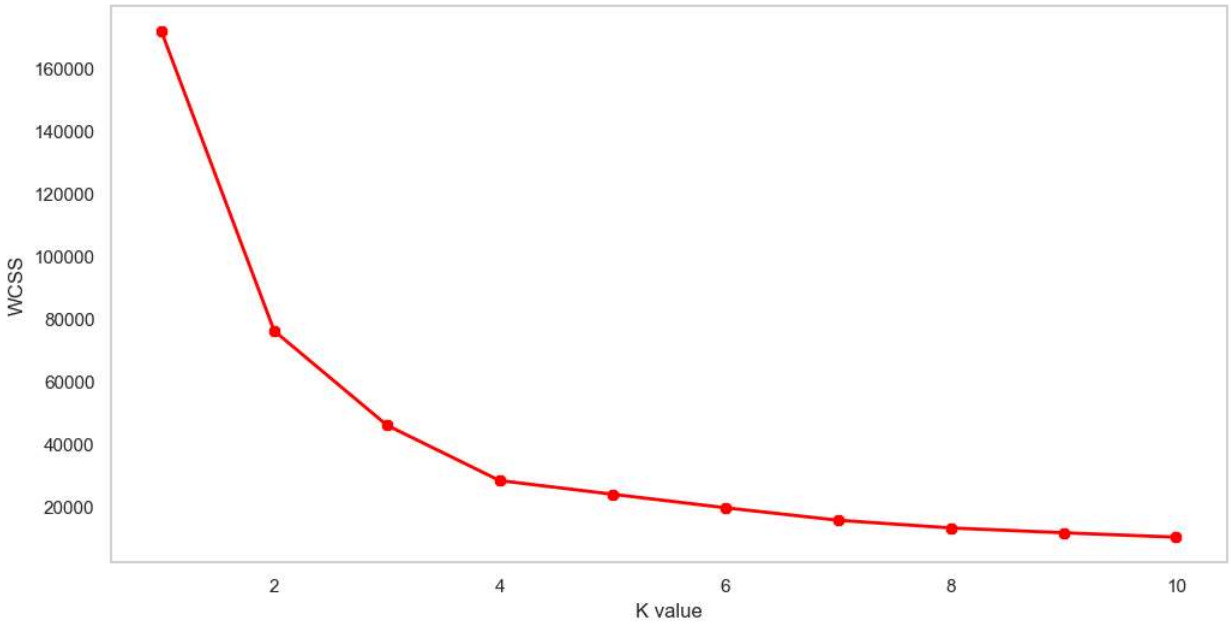


In [19]:
```
from sklearn.cluster import KMeans
```

In [20]:
```
#clustering process
#finding optimal k value
X1=df.loc[:,["Age","Spending Score (1-100)"]].values
```

```
wcss =[]
for k in range(1,11):
    kmeans = KMeans(n_clusters=k, init="k-means++")
    kmeans.fit(X1)
    wcss.append(kmeans.inertia_)
plt.figure(figsize =(12,6))
plt.grid()
plt.plot(range(1,11),wcss,linewidth=2,color="red",marker="8")
plt.xlabel("K value")
plt.ylabel("WCSS")
plt.show()
```

```
D:\Users\shrey\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
```



In [21]:
```
#Training the K-Means Clustering Model
kmeans = KMeans(n_clusters =4)
label = kmeans.fit_predict(X1)

print(label)
```

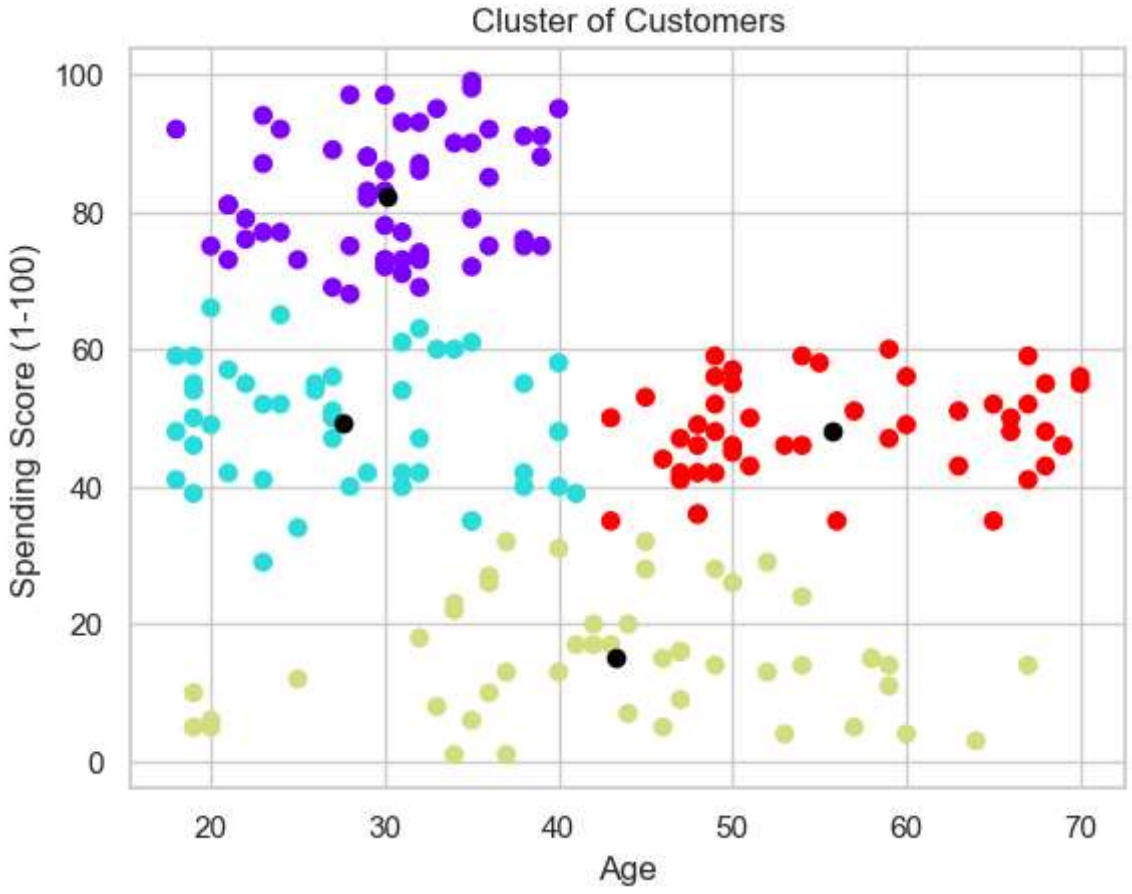
```
[1 0 2 0 1 0 2 0 2 0 2 0 2 0 2 0 1 1 2 0 1 0 2 0 2 0 2 1 2 0 2 0 2 0 2 0 2
 0 2 0 3 0 3 1 2 1 3 1 1 1 3 1 1 3 3 3 3 3 1 3 3 1 3 3 3 1 3 3 1 1 3 3 3 3
 3 1 3 1 1 3 3 1 3 3 1 3 3 1 1 3 3 1 3 1 1 1 3 1 3 1 1 3 3 1 3 1 3 3 3 3 3
 1 1 1 1 1 3 3 3 3 1 1 1 0 1 0 3 0 2 0 2 0 1 0 2 0 2 0 2 0 2 0 1 0 2 0 3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

In [22]:
```
#Checking the centers of out clusters (Also known as Centroids)
print(kmeans.cluster_centers_)
```

```
[[30.1754386  82.35087719]
 [27.61702128 49.14893617]
 [43.29166667 15.02083333]
 [55.70833333 48.22916667]]
```

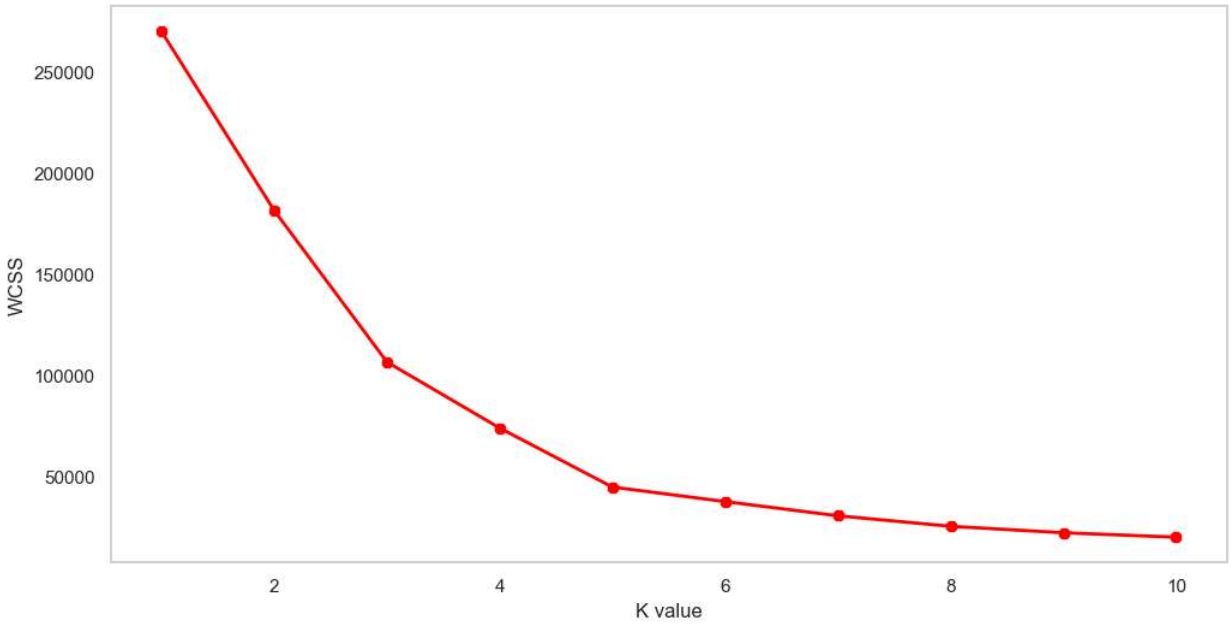
In [23]:
```
plt.scatter(X1[:,0], X1[:,1], c = kmeans.labels_, cmap = "rainbow")
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], color = "black
```

```
plt.title("Cluster of Customers")
plt.xlabel("Age")
plt.ylabel("Spending Score (1-100)")
plt.show()
```



In [24]:
```
X2=df.loc[:,["Annual Income (k$)","Spending Score (1-100)"]].values

from sklearn.cluster import KMeans

wcss =[]
for k in range(1,11):
    kmeans = KMeans(n_clusters=k, init="k-means++")
    kmeans.fit(X2)
    wcss.append(kmeans.inertia_)
plt.figure(figsize =(12,6))
plt.grid()
plt.plot(range(1,11),wcss,linewidth=2,color="red",marker="8")
plt.xlabel("K value")
plt.ylabel("WCSS")
plt.show()
```

```
D:\Users\shrey\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
```



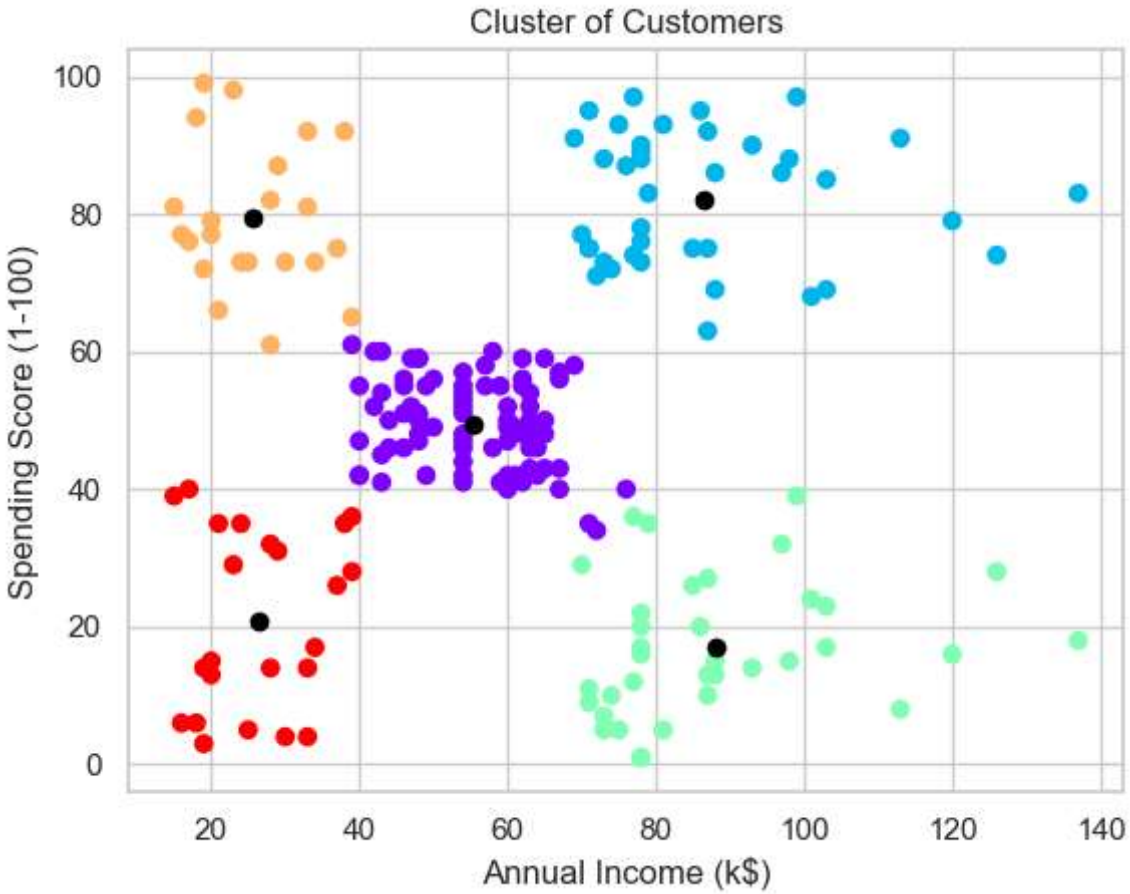
In [25]:
```
kmeans = KMeans(n_clusters = 5)
label = kmeans.fit_predict(X2)

print(label)
```

```
[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 0 4 3 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 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]
```

In [26]:
```
print(kmeans.cluster_centers_)
```

```
[[55.2962963  49.51851852]
 [86.53846154 82.12820513]
 [88.2        17.11428571]
 [25.72727273 79.36363636]
 [26.30434783 20.91304348]]
```

In [27]:
```
plt.scatter(X2[:,0], X2[:,1], c = kmeans.labels_, cmap = "rainbow")
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], color = "black
plt.title("Cluster of Customers")
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.show()
```


Cluster of Customers
Spending Score (1-100)
Annual Income (k$)
0
20
40
60
80
100
20
40
60
80
100
120
140

Business Insights

The result of the analysis shows that the retail store customers can be group into 5 clusters or segments for targeted marketing.

Cluster 1 (purple): These are average income earners with average spending scores. They are cautious with their spending at the store.

Cluster 2 (blue): The customers in this group are high income earners and with high spending scores. They bring in profit. Discounts and other offers targeted at this group will increase their spending score and maximize profit.

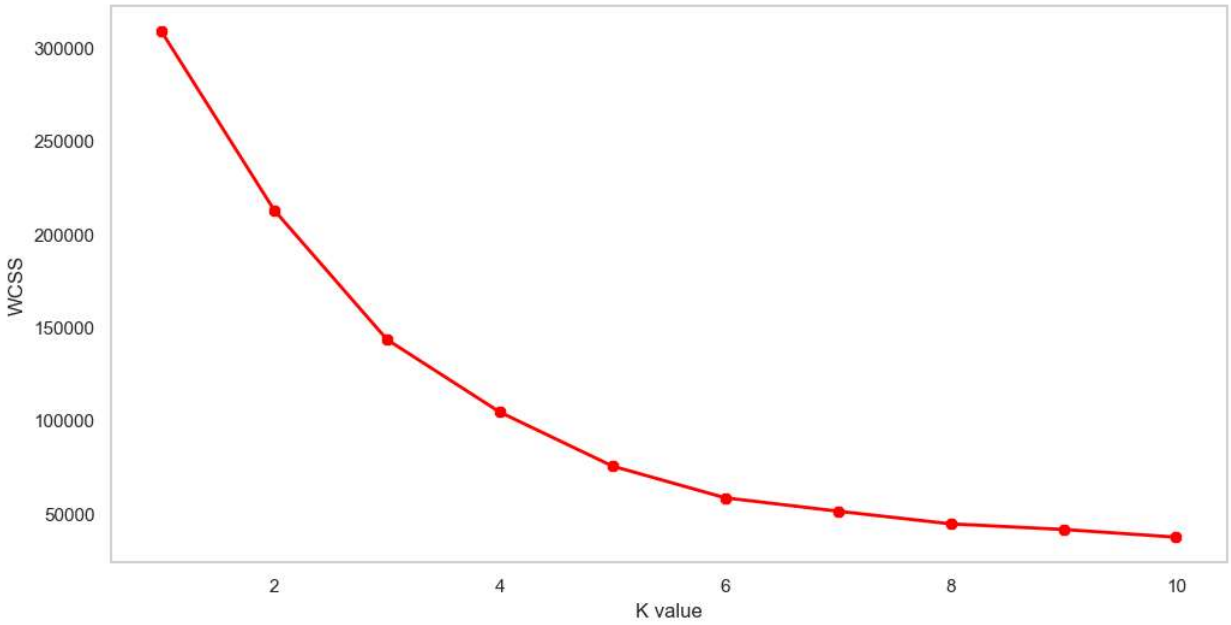
Cluster 3 (green): This group of customers have a higher income but they do not spend more at the store. One of the assumption could be that they are not satisfied with the services rendered at the store. They are another ideal group to be targeted by the marketing team because they have the potential to bring in increased profit for the store.

Cluster 4 (red): Low income earners with low spending score. I can assume that this is so because people with low income will tend to purchase less item at the store.

Cluster 5 (orange): These are low income earning customers with high spending scores. I can assume that why this group of customers spend more at the retail store despite earning less is because they enjoy and are satisfied with the services rendered at the retail store.

In [28]:
```
X3 = df.iloc[:,1:]

wcss =[]
for k in range(1,11):
    kmeans = KMeans(n_clusters=k, init="k-means++")
    kmeans.fit(X3)
    wcss.append(kmeans.inertia_)
plt.figure(figsize =(12,6))
plt.grid()
plt.plot(range(1,11),wcss,linewidth=2, color="red",marker="8")
plt.xlabel("K value")
plt.ylabel("WCSS")
plt.show()
```

```
D:\Users\shrey\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
```



In [29]:
```
kmeans = KMeans(n_clusters = 5)
label = kmeans.fit_predict(X3)

print(label)
```

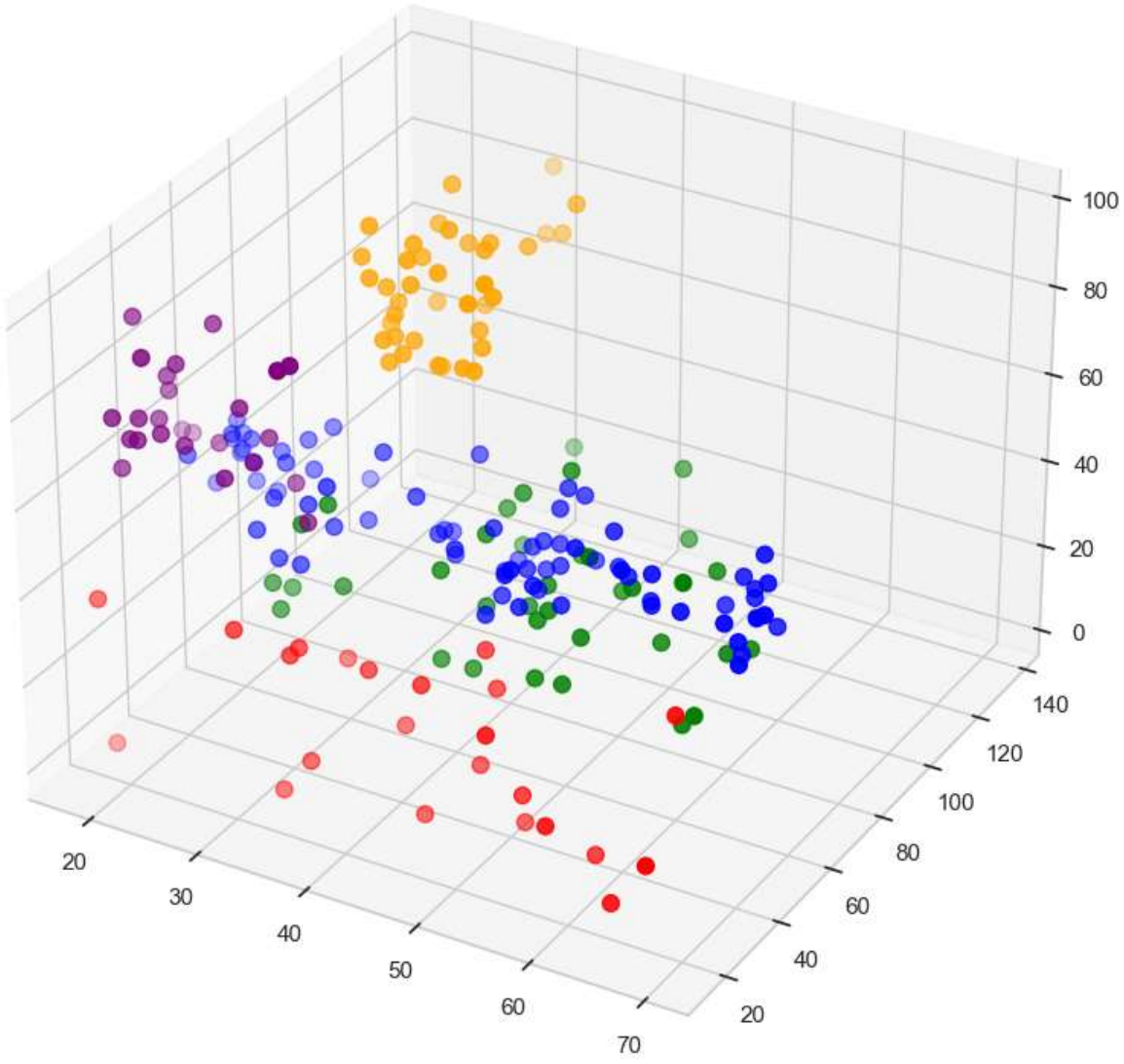
```
[4 0 4 0 4 0 4 0 4 0 4 0 4 0 4 0 4 0 4 0 4 0 4 0 4 0 4 0 4 0 4 0 4 0 4
 0 4 0 4 0 4 0 4 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 2 3 2 1 2 3 2 3 2 3 2 3 2 3 2 3 2 1 2 3 2 3 2
 3 2 3 2 3 2 3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

In [30]:
```
print(kmeans.cluster_centers_)
```

```
[[25.52173913 26.30434783 78.56521739]
 [43.08860759 55.29113924 49.56962025]
 [32.69230769 86.53846154 82.12820513]
 [40.66666667 87.75       17.58333333]
 [45.2173913  26.30434783 20.91304348]]
```

In [31]:
```
#Visualizing all the clusters
clusters = kmeans.fit_predict(X3)
df["label"] = clusters

from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize = (20,10))
ax = fig.add_subplot(111, projection = '3d')
ax.scatter(df.Age[df.label == 0],df["Annual Income (k$)"][df.label == 0],df["Spending
ax.scatter(df.Age[df.label == 1],df["Annual Income (k$)"][df.label == 1],df["Spending
ax.scatter(df.Age[df.label == 2],df["Annual Income (k$)"][df.label == 2],df["Spending
ax.scatter(df.Age[df.label == 3],df["Annual Income (k$)"][df.label == 3],df["Spending
ax.scatter(df.Age[df.label == 4],df["Annual Income (k$)"][df.label == 4],df["Spending
```

Out[31]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x1f464bc7d00>
```


100
80
60
40
20
0
140
120
100
80
60
40
20
20
30
40
50
60
70