In [68]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [69]:
```
#importing data set
df=pd.read_csv(r"Mall_Customers.csv")
#view data set
df.head()
```

Out[69]:

	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 [70]:
```
# check for data set shape
df.shape
```

Out[70]:
```
(200, 5)
```

In [71]:
```
#describing our data set
df.describe()
```

Out[71]:

	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 [72]:
```
# check for the data type of each of the variable
df.dtypes
```

Out[72]:
```
CustomerID                int64
Gender                   object
Age                       int64
Annual Income (k$)        int64
Spending Score (1-100)    int64
dtype: object
```

In [73]:

```
# check if any dataset has any null values
df.isnull().sum()
```

Out[73]:

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

In [74]:

```
# dropping customer id data
df.drop(["CustomerID"],axis=1,inplace=True)
```

In [75]:

```
#cheching data set
df.head()
```

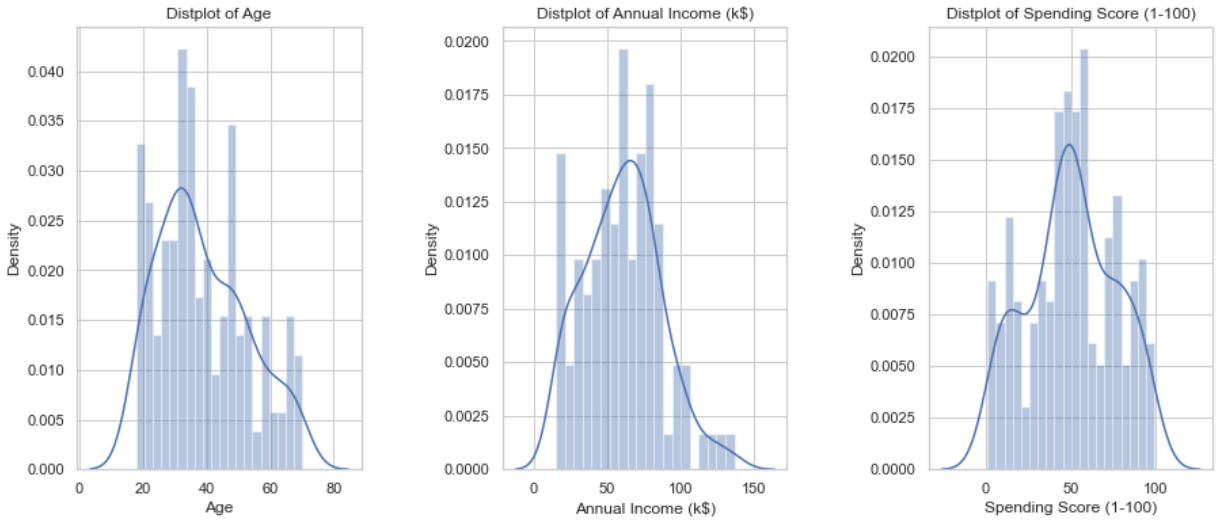
Out[75]:

	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

In [76]:

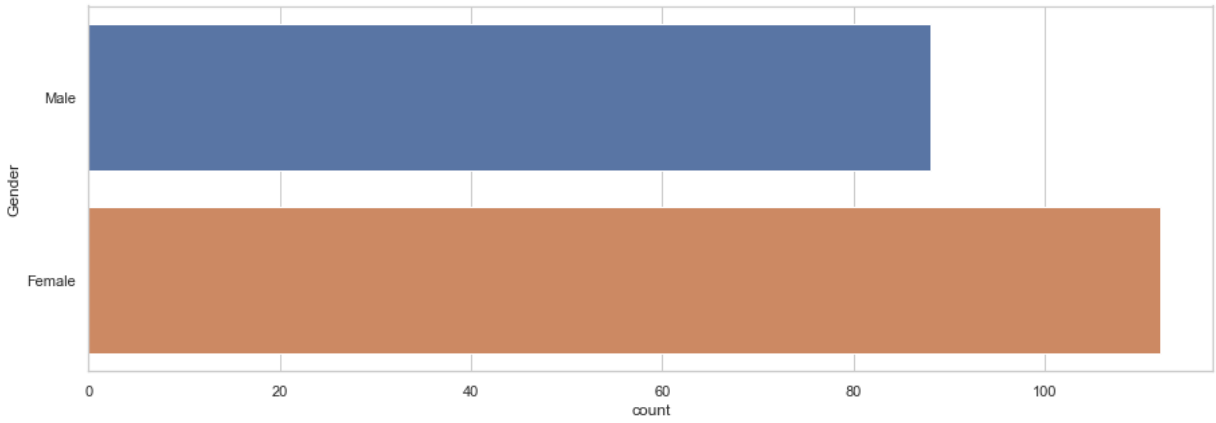
```
# analysis and visualization of data
# distrubution plot for age, annual income, and spending score
# here (1,3,n) represent no. of rows, columns, index we are incrementing
plt.figure(1, figsize=(15,6))
n=0
for x in ['Age' , 'Annual Income (k$)' , 'Spending Score (1-100)']:
    n += 1
    plt.subplot(1 , 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()
```

```
C:\Users\ipsit\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\ipsit\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\ipsit\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



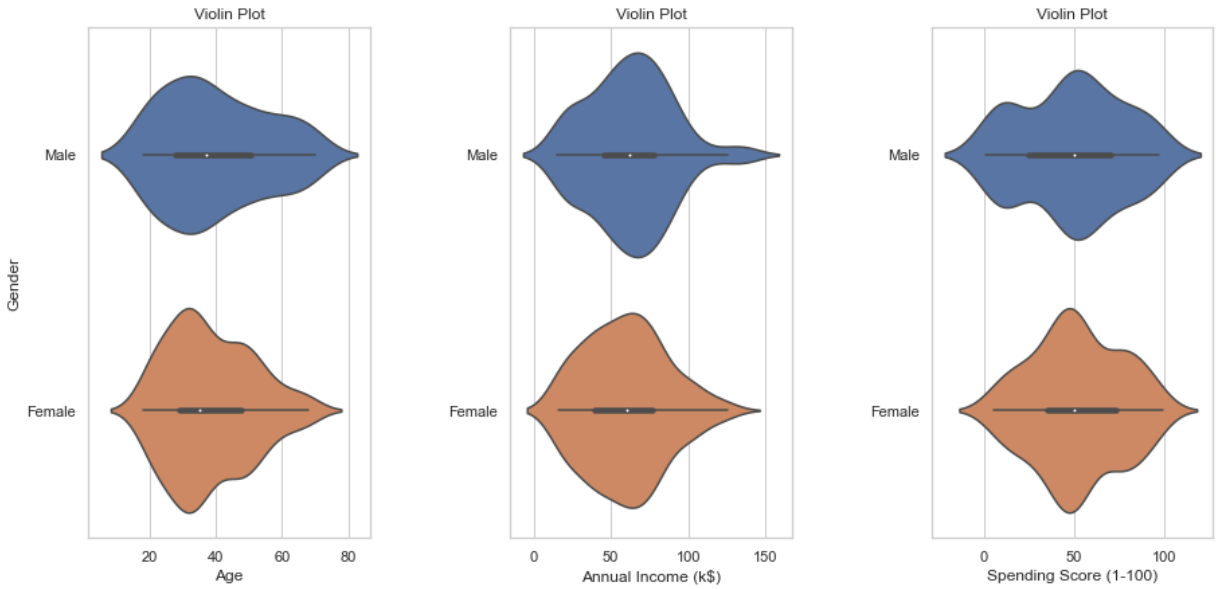
In [77]:

```
# compare no of male and female
plt.figure(figsize=(15,5))
sns.countplot(y='Gender',data=df)
plt.show()
```



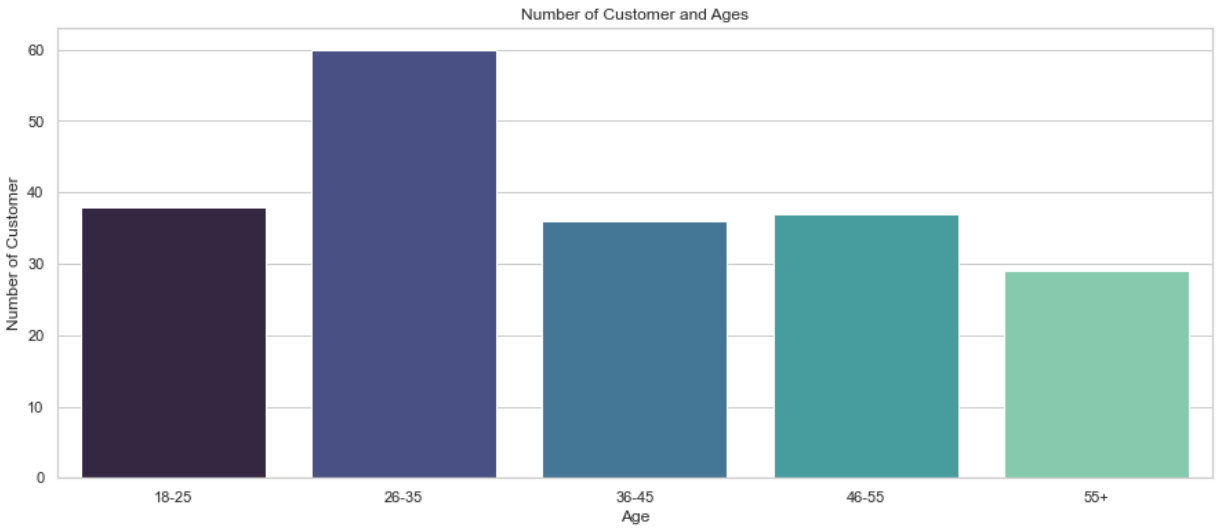
In [78]:

```
# representing violin representation for age, annual income and spending score
# x axis represent age, annual income and spending score and y axis represent the ge
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 [79]:

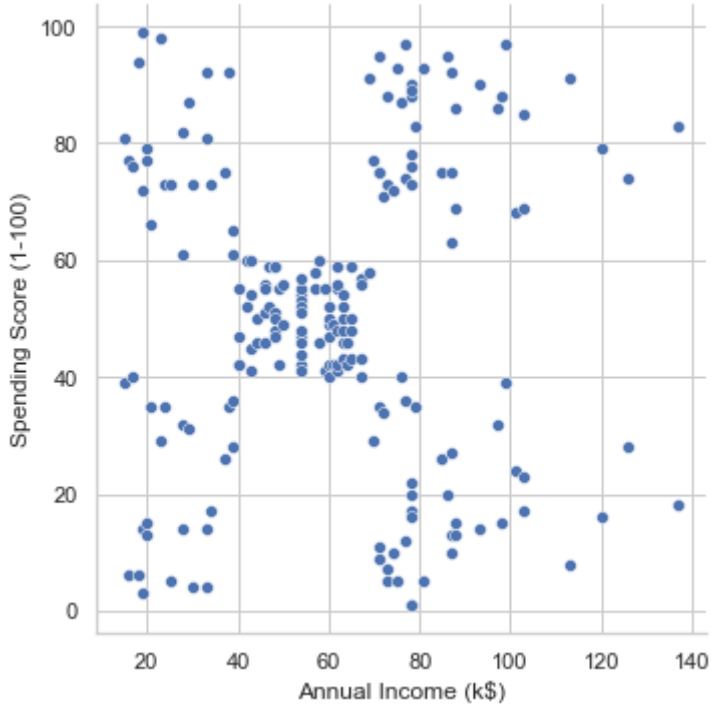
```
#for better visulization and analysis lets divide the age into different catagories
#to uderstand which renge of age has highest number of customers
#1.first we divided the ages
#2.for the graph x axis will have age renge and y axis will have the no of customer
#3.bar plot
age_18_25 = df.Age[(df.Age >= 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

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



In [80]:

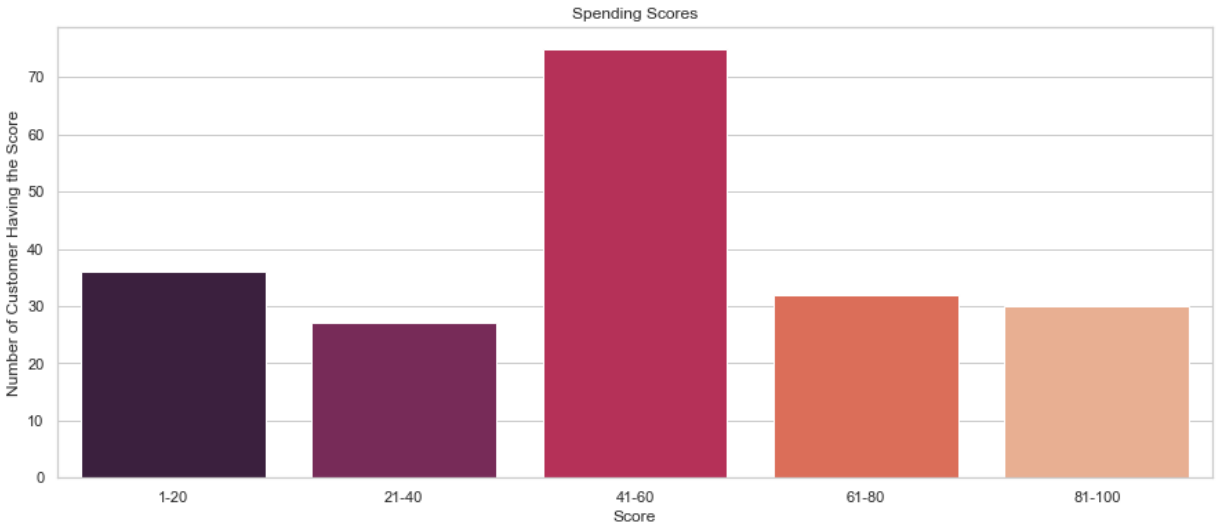
```
# relationship b/w annual income and spending score
sns.relplot(x="Annual Income (k$)", y="Spending Score (1-100)", data=df)
```

Out[80]: <seaborn.axisgrid.FacetGrid at 0x1fe8dd6fc10>



In [81]:

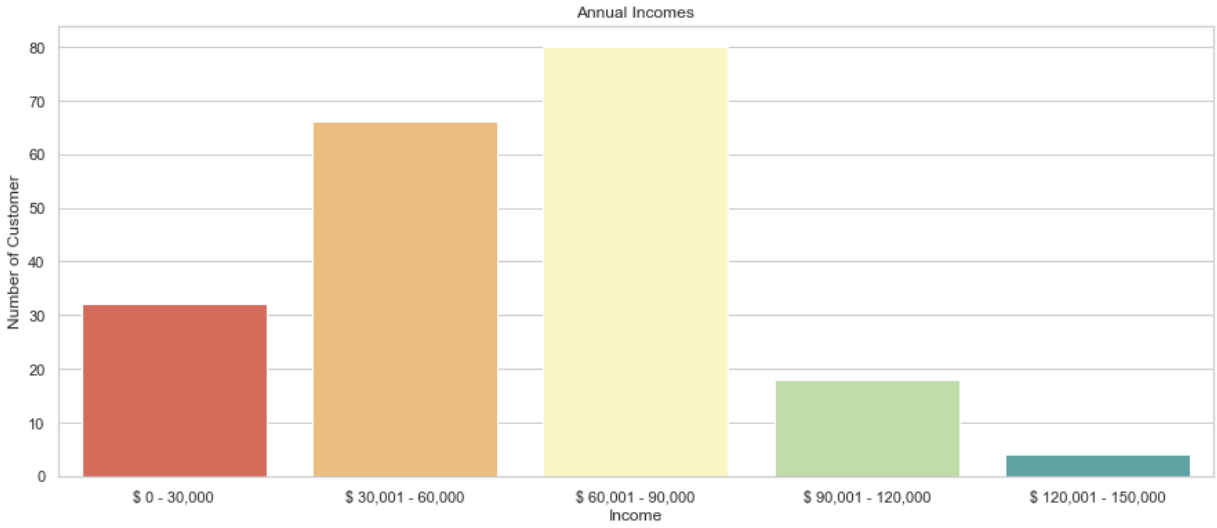
```
# 1.divide the spending score in catagories
# 2.x axis represent renge of spending score and y axis has no of customers present
# 3.represent the data
ss_1_20 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 1) & (df["Sp
ss_21_40 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 21) & (df["
ss_41_60 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 41) & (df["
ss_61_80 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 61) & (df["
ss_81_100 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 81) & (df[

ssx = ["1-20", "21-40", "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

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 [82]:

```
# 1.check the min and max value for the annual income
# 2.x axis represent renge of annual income and y axis has count of present in each
```

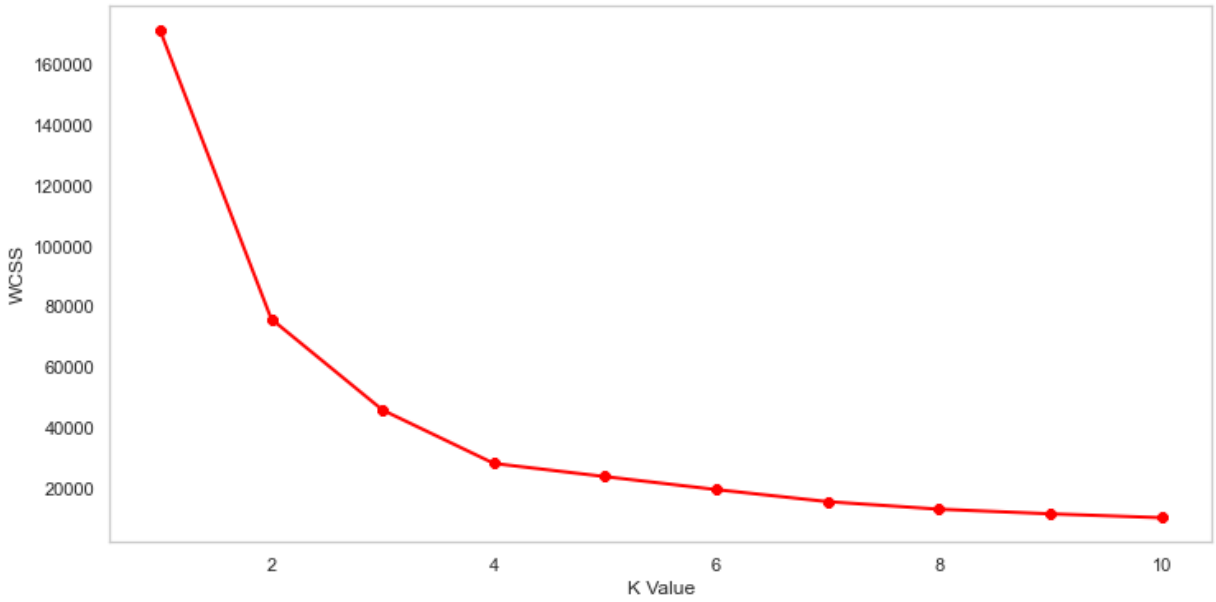
```
# 3.represent the data
ai0_30 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 0) & (df["Annual Inco
ai31_60 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 31) & (df["Annual In
ai61_90 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 61) & (df["Annual In
ai91_120 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 91) & (df["Annual I
ai121_150 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 121) & (df["Annual

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.va

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



In [83]:

```
#starting the clustering process
#1. Let's find the optimum numbers of clusters between age and spending score
X1=df.loc[:, ["Age","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(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()
```

```
C:\Users\ipsit\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less c
hunks than available threads. You can avoid it by setting the environment variable O
MP_NUM_THREADS=1.
  warnings.warn(
```



In [86]:
```
#Let's the optimum no of clusters of our data is 4 i.e (0,1,2,3)
kmeans = KMeans(n_clusters=4)

label = kmeans.fit_predict(X1)

print(label)
```

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

In [87]:
```
# check the centroids with x and y coordinates
print(kmeans.cluster_centers_)
```

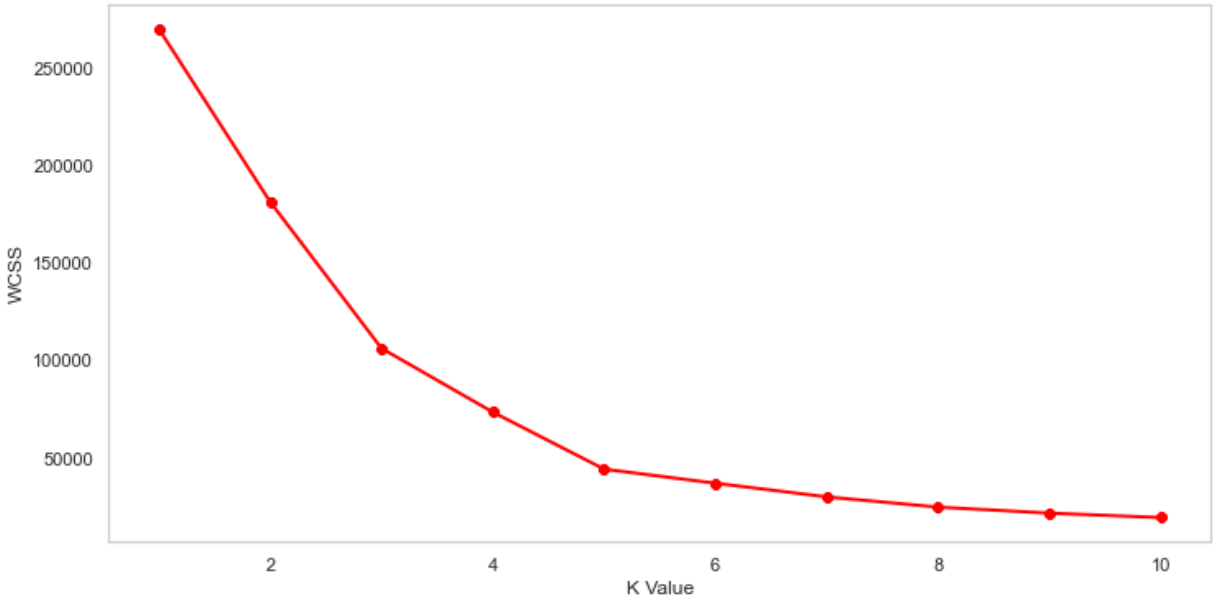
```
[[30.1754386  82.35087719]
 [55.70833333 48.22916667]
 [27.61702128 49.14893617]
 [43.29166667 15.02083333]]
```

In [90]:
```
# visualizing clusters of age and spending score on a graph(scattered plot)
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('Clusters of Customers')
plt.xlabel('Age')
plt.ylabel('Spending Score(1-100)')
plt.show()
```

In [92]:

```
#starting the clustering process
#1. Let's find the optimum numbers of clusters between annual income and spending sc
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()
```

```
C:\Users\ipsit\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less c
hunks than available threads. You can avoid it by setting the environment variable O
MP_NUM_THREADS=1.
  warnings.warn(
```



In [93]:

```
#Let's the optimum no of clusters of our data is 5 i.e (0,1,2,3,4)
kmeans = KMeans(n_clusters=5)
```

```
label = kmeans.fit_predict(X2)

print(label)
```

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

In [94]:

```
# check the centroids with x and y coordinates
print(kmeans.cluster_centers_)
```

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

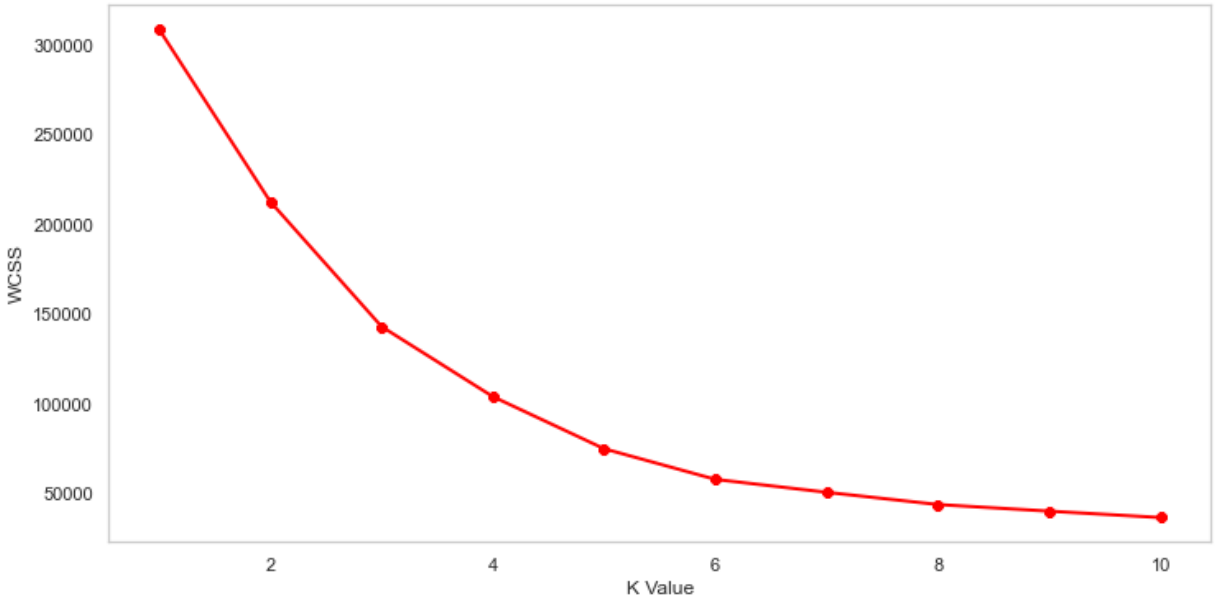
In [95]:

```
# visualizing clusters of annual income and spending score on a graph(scattered plot
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('Clusters of Customers')
plt.xlabel('Age')
plt.ylabel('Spending Score(1-100)')
plt.show()
```

In [96]:

```
# consider age, annual income and spending score all togather
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()
```

```
C:\Users\ipsit\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less c
hunks than available threads. You can avoid it by setting the environment variable O
MP_NUM_THREADS=1.
  warnings.warn(
```



In [97]:

```
#Let's the optimum no of clusters of our data is 5 i.e (0,1,2,3,4)
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 1 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 3 2 3 1 3 2 3 2 3 2 3 2 3 2 3 2 3 1 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 3 2 3]
```

In [98]:

```
# check the centroids with x, y and z coordinates
print(kmeans.cluster_centers_)
```

```
[[25.27272727 25.72727273 79.36363636]
 [42.9375     55.0875     49.7125    ]
 [40.66666667 87.75       17.58333333]
 [32.69230769 86.53846154 82.12820513]
 [45.2173913  26.30434783 20.91304348]]
```

In [100…

```
# visualizing clusters of age, annual income and spending score on a 3D graph(scatte
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$)"][df.label == 0], df["Spend
ax.scatter(df.Age[df.label == 1], df["Annual Income (k$)"][df.label == 1], df["Spend
ax.scatter(df.Age[df.label == 2], df["Annual Income (k$)"][df.label == 2], df["Spend
ax.scatter(df.Age[df.label == 3], df["Annual Income (k$)"][df.label == 3], df["Spend
ax.scatter(df.Age[df.label == 4], df["Annual Income (k$)"][df.label == 4], df["Spend
ax.view_init(30, 185)
```

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

plt.show()
```