In [1]:

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

In [2]:

```
df = pd.read_csv("C:/Users/DEEPIKA/OneDrive/Desktop/Mall_Customers.csv")
df.head()
```

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

In [3]:

```
df.tail()
```

Out[3]:

	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 [4]:

```
df.describe()
```

Out[4]:

	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 [5]:

```
df.shape
```

Out[5]:

```
(200, 5)
```

In [6]:

```
df.dtypes
```

Out[6]:

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

In [7]:

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

Out[7]:

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

In [8]:

```
df.drop(["CustomerID"],axis=1,inplace=True)
```

In [9]:

```
df
```

Out[9]:

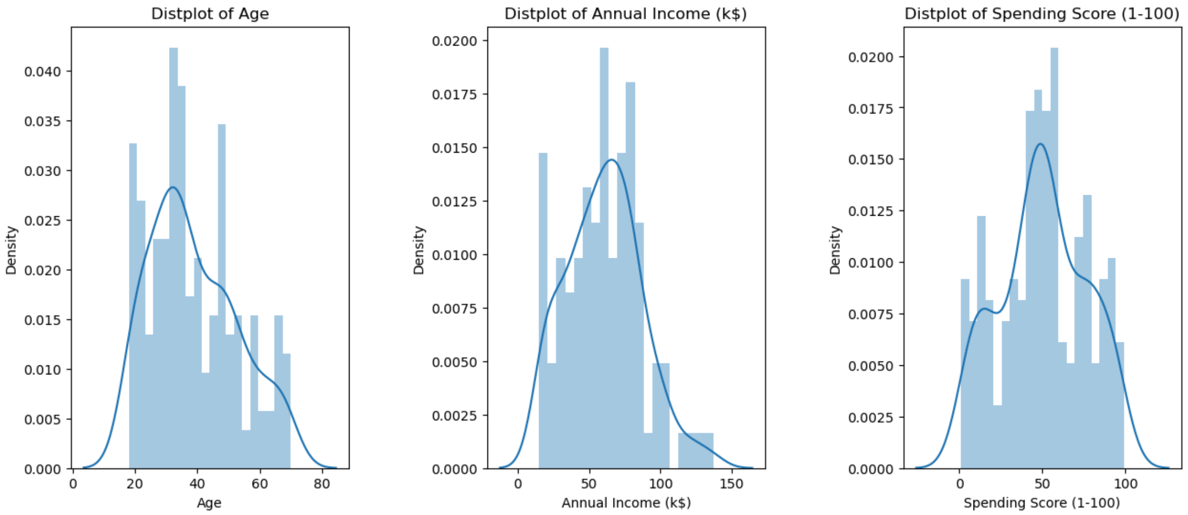
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40
...	...	...	...	...
195	Female	35	120	79
196	Female	45	126	28
197	Male	32	126	74
198	Male	32	137	18
199	Male	30	137	83

200 rows × 4 columns

In [10]:

```
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\DEEPIKA\AppData\Local\Temp\ipykernel_13840\3814829345.py:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[x],bins=20)
C:\Users\DEEPIKA\AppData\Local\Temp\ipykernel_13840\3814829345.py:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[x],bins=20)
C:\Users\DEEPIKA\AppData\Local\Temp\ipykernel_13840\3814829345.py:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[x],bins=20)
```

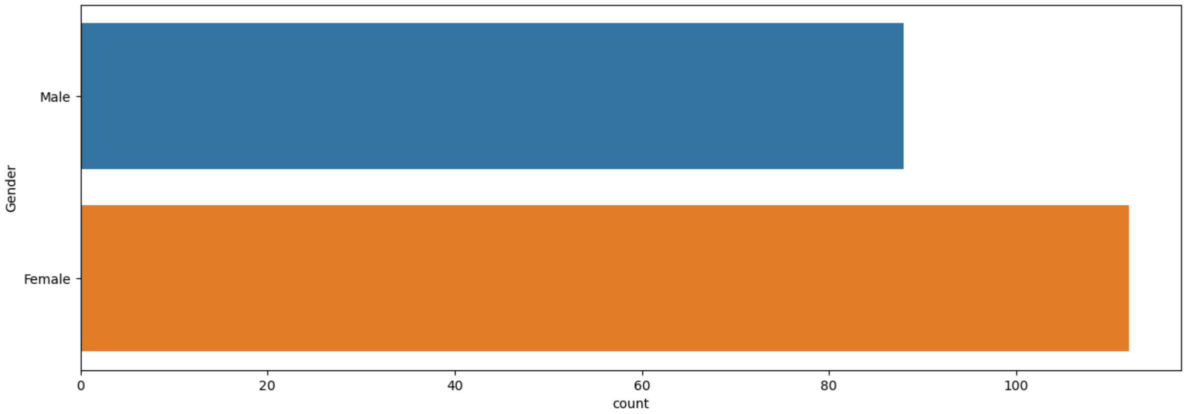


In [11]:

```
plt.figure(figsize=(15,5))
sns.countplot(y='Gender',data=df)
plt.show
```

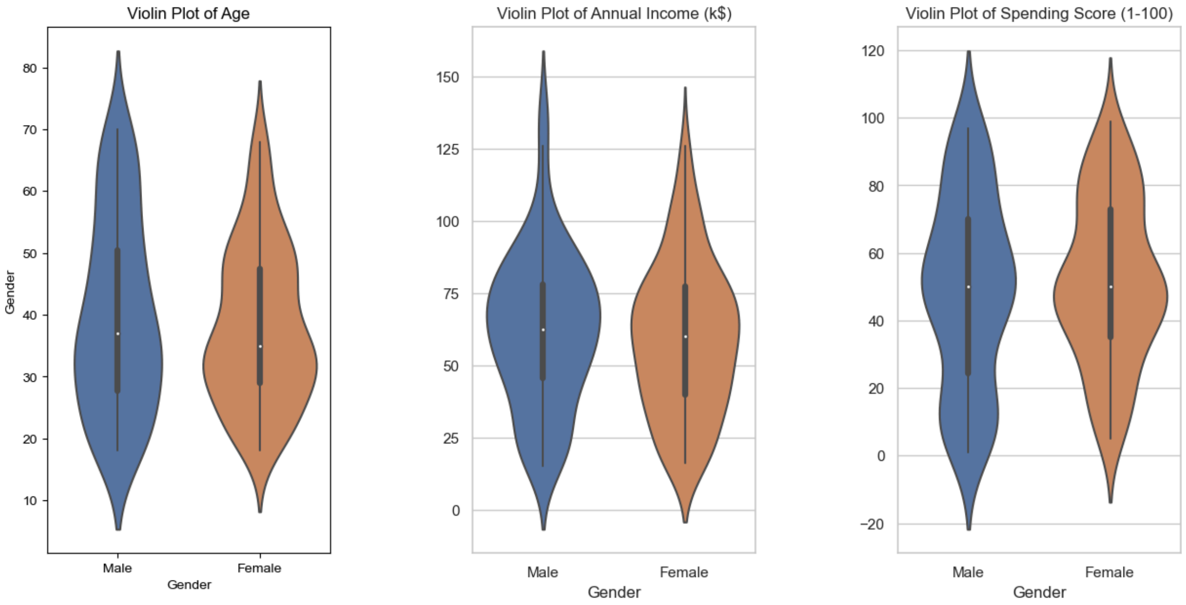
Out[11]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [12]:

```
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='Gender', y=cols, data=df)
    plt.ylabel('Gender' if n == 1 else '')
    plt.title('Violin Plot of {}'.format(cols))

plt.show()
```

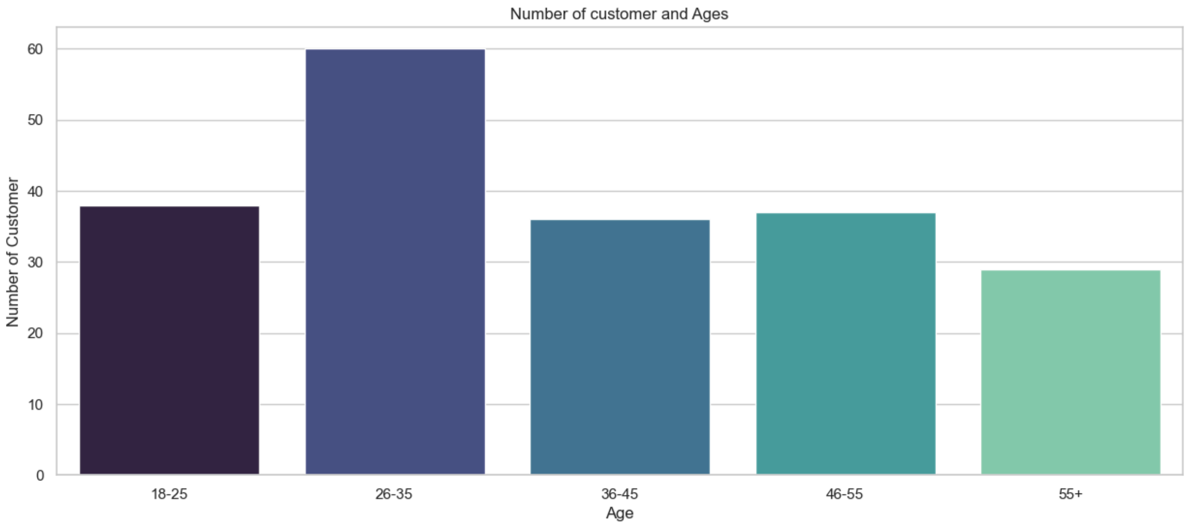

Violin Plot of Age

Violin Plot of Annual Income (k$)

Violin Plot of Spending Score (1-100)

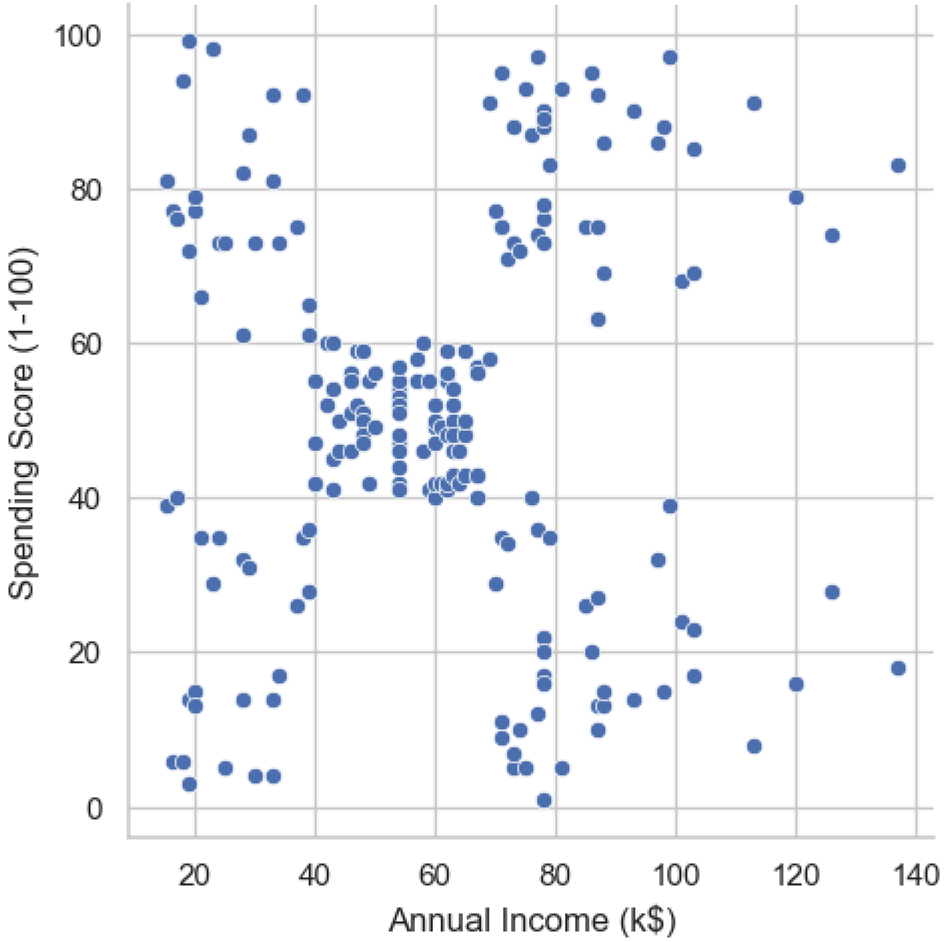
In [13]:

```
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),len(age_36_45),len(age_46_55),len(age_5

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()
```



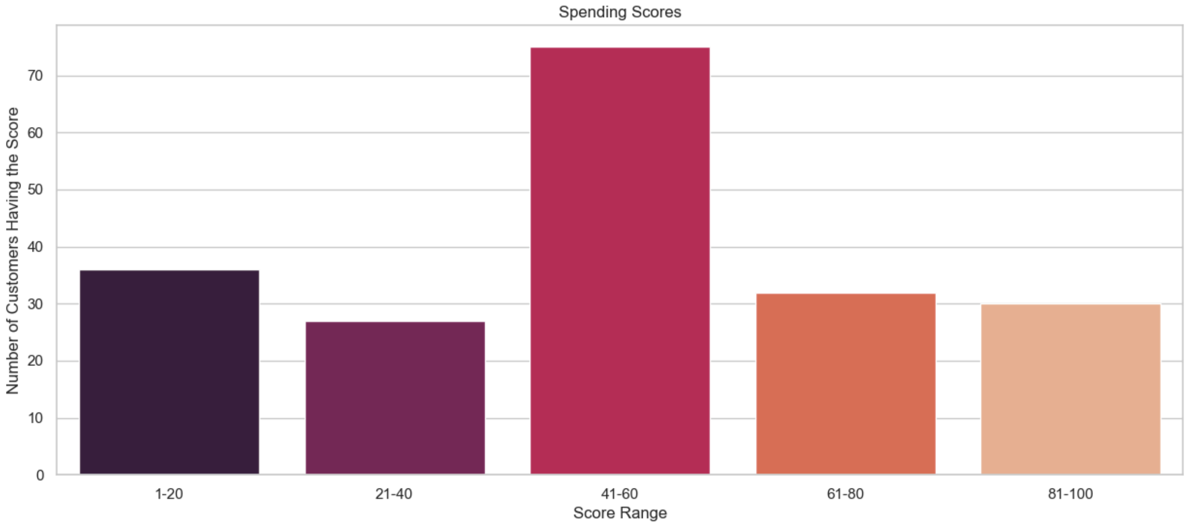
In [14]:
```
sns.relplot(x="Annual Income (k$)", y="Spending Score (1-100)", data=df)
```

Out[14]:
```
<seaborn.axisgrid.FacetGrid at 0x221a20d5490>
```



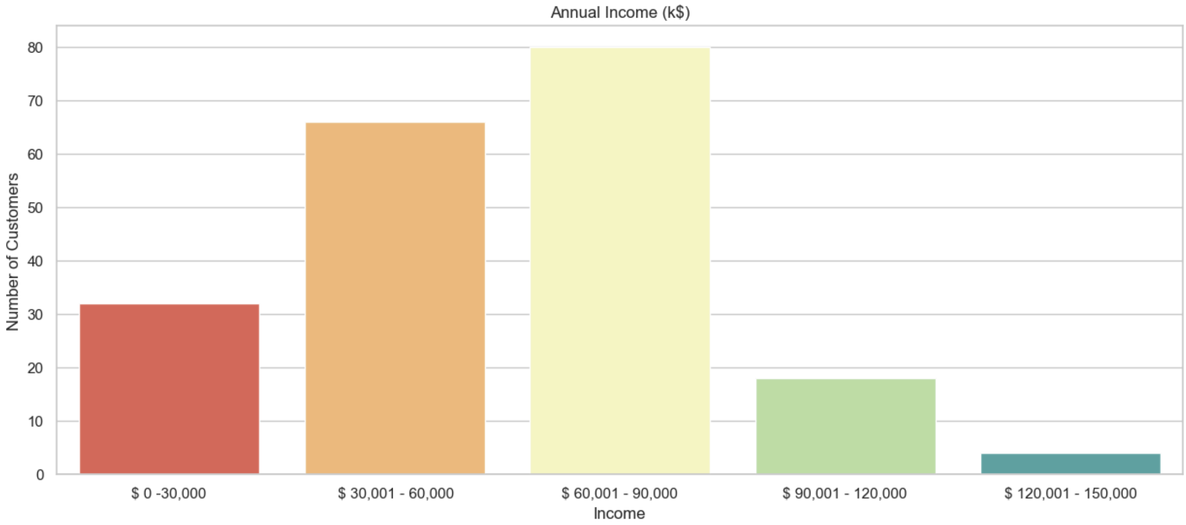
In [15]:
```
ss_1_20 = df['Spending Score (1-100)'][(df['Spending Score (1-100)'] >= 1) & (df['S
ss_21_40 = df['Spending Score (1-100)'][(df['Spending Score (1-100)'] >= 21) & (df[
ss_41_60 = df['Spending Score (1-100)'][(df['Spending Score (1-100)'] >= 41) & (df[
ss_61_80 = df['Spending Score (1-100)'][(df['Spending Score (1-100)'] >= 61) & (df[
ss_81_100 = df['Spending Score (1-100)'][(df['Spending Score (1-100)'] >= 81) & (df

# Fix the labels and the counts
ssx = ["1-20", "21-40", "41-60", "61-80", "81-100"]
ssy = [len(ss_1_20), len(ss_21_40), len(ss_41_60), len(ss_61_80), len(ss_81_100)]

# Plot
plt.figure(figsize=(15, 6))
```

```
sns.barplot(x=ssx, y=ssy, palette='rocket')
plt.title("Spending Scores")
plt.xlabel("Score Range")
plt.ylabel("Number of Customers Having the Score")
plt.show()
```



In [16]:

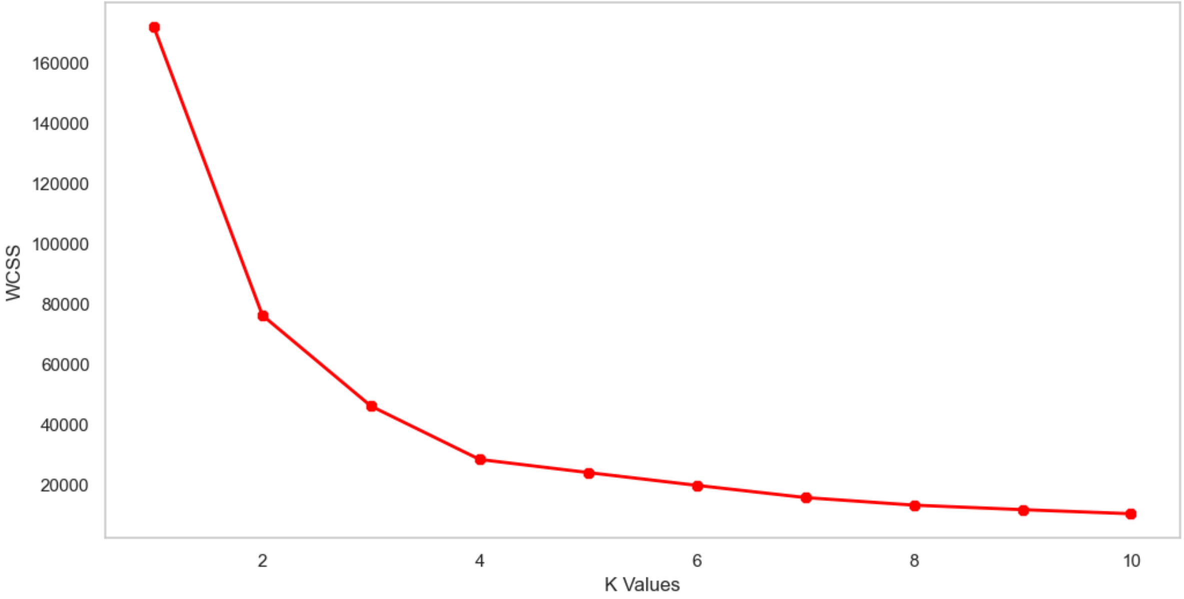
```
ai0_30 = df["Annual Income (k$)"] [(df["Annual Income (k$)"] >= 0) & (df["Annual Ir
ai0_60 = df["Annual Income (k$)"] [(df["Annual Income (k$)"] >= 31) & (df["Annual I
ai0_90 = df["Annual Income (k$)"] [(df["Annual Income (k$)"] >= 61) & (df["Annual I
ai0_120 = df["Annual Income (k$)"] [(df["Annual Income (k$)"] >= 91) & (df["Annual 
ai0_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(ai0_60.values),len(ai0_90.values),len(ai0_120.values)

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



In [17]:

```
X1 = df.loc[:, ["Age", "Spending Score (1-100)"]].values

from sklearn.cluster import KMeans
wcss = []  # Initialize an empty list
for k in range(1, 11):  # Fix the range syntax
    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 Values")
plt.ylabel("WCSS")
plt.show()
```

```
C:\Users\DEEPIKA\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment va
riable OMP_NUM_THREADS=1.
  warnings.warn(
```



In [18]:

```
kmeans = KMeans(n_clusters=5)

label = kmeans.fit_predict(X1)

print(label)
```

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

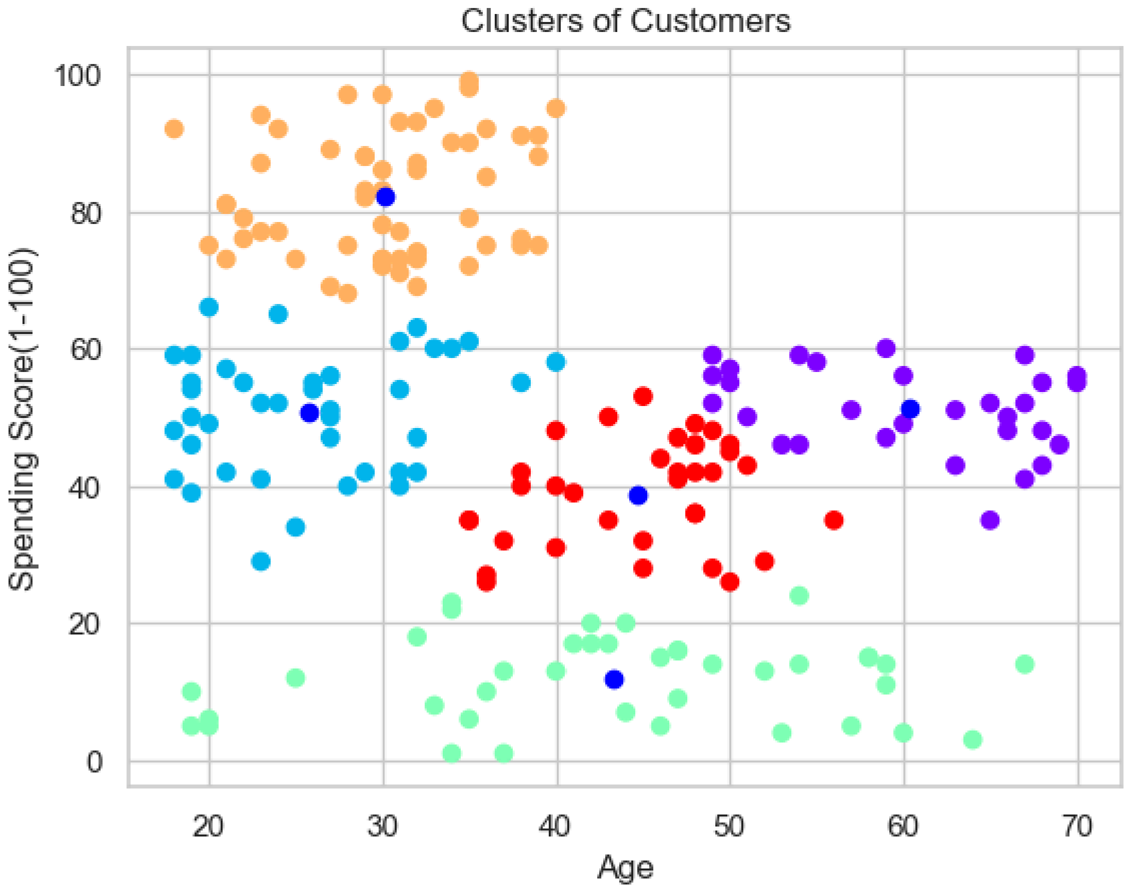
In [19]:

```
print(kmeans.cluster_centers_)
```

```
[[60.36666667 51.16666667]
 [25.775      50.775     ]
 [43.28205128 11.84615385]
 [30.1754386  82.35087719]
 [44.70588235 38.76470588]]
```

In [20]:

```
plt.scatter(X1[:,0], X1[:,1],c = kmeans.labels_,cmap='rainbow')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],color = 'blue
plt.title('Clusters of Customers')
plt.xlabel('Age')
plt.ylabel('Spending Score(1-100)')
plt.show()
```

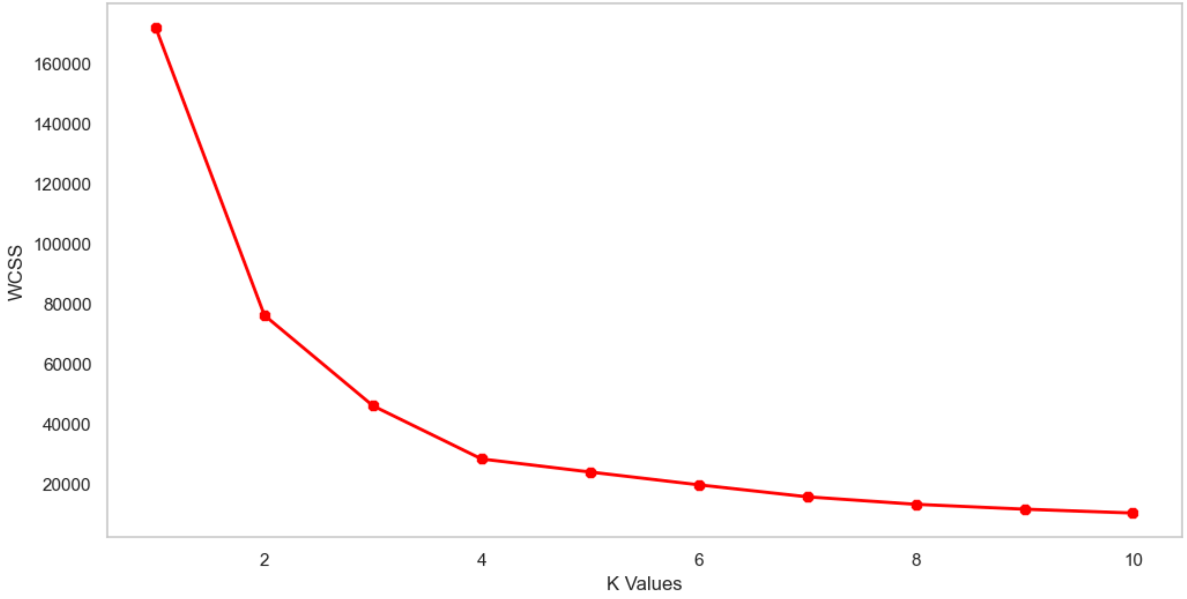


```
In [21]: print(kmeans.cluster_centers_)
```

```
[[60.36666667 51.16666667]
 [25.775      50.775     ]
 [43.28205128 11.84615385]
 [30.1754386  82.35087719]
 [44.70588235 38.76470588]]
```

```
In [22]: X2 = df.loc[:, ["Age", "Spending Score (1-100)"]].values

         from sklearn.cluster import KMeans
         wcss = []  # Initialize an empty list
         for k in range(1, 11):  # Fix the range syntax
             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 Values")
         plt.ylabel("WCSS")
         plt.show()
```

```
C:\Users\DEEPIKA\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment va
riable OMP_NUM_THREADS=1.
  warnings.warn(
```



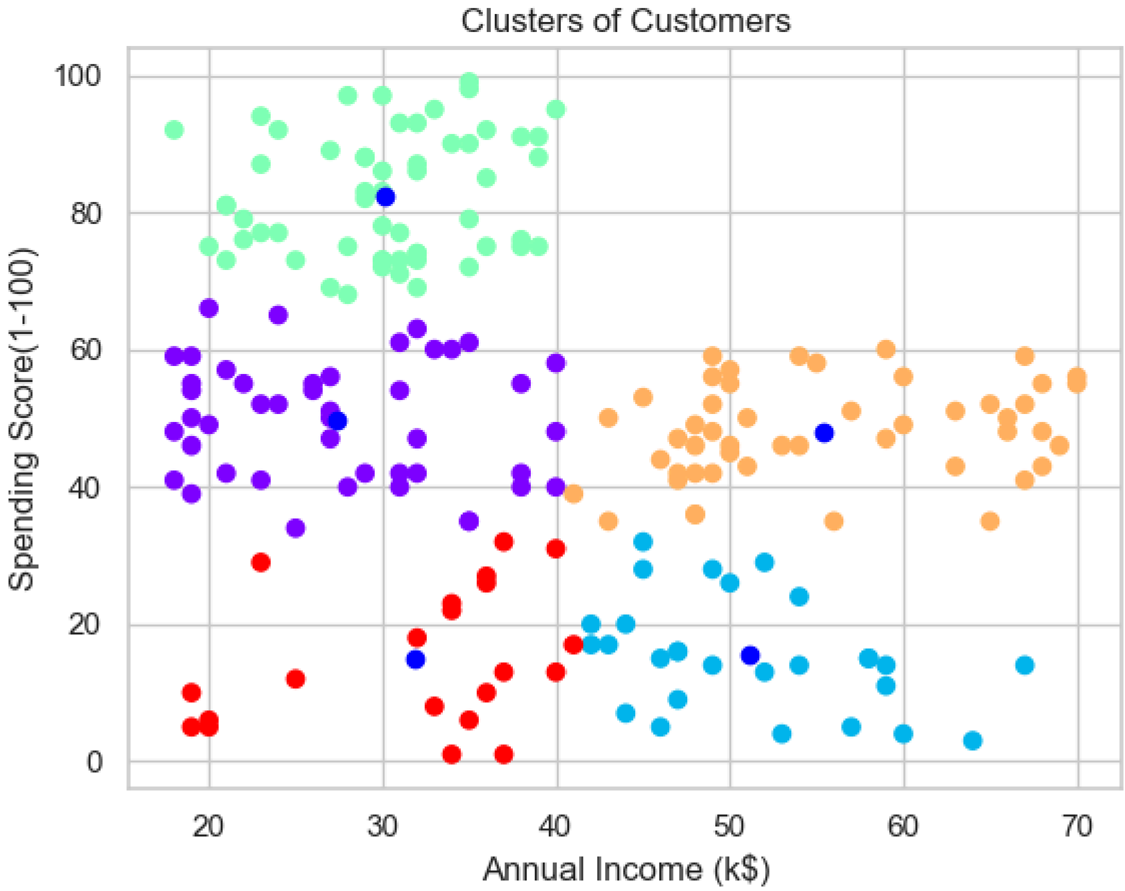
In [23]:

```
kmeans =KMeans(n_clusters=5)

label = kmeans.fit_predict(X2)

print(label)
```

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

In [24]:

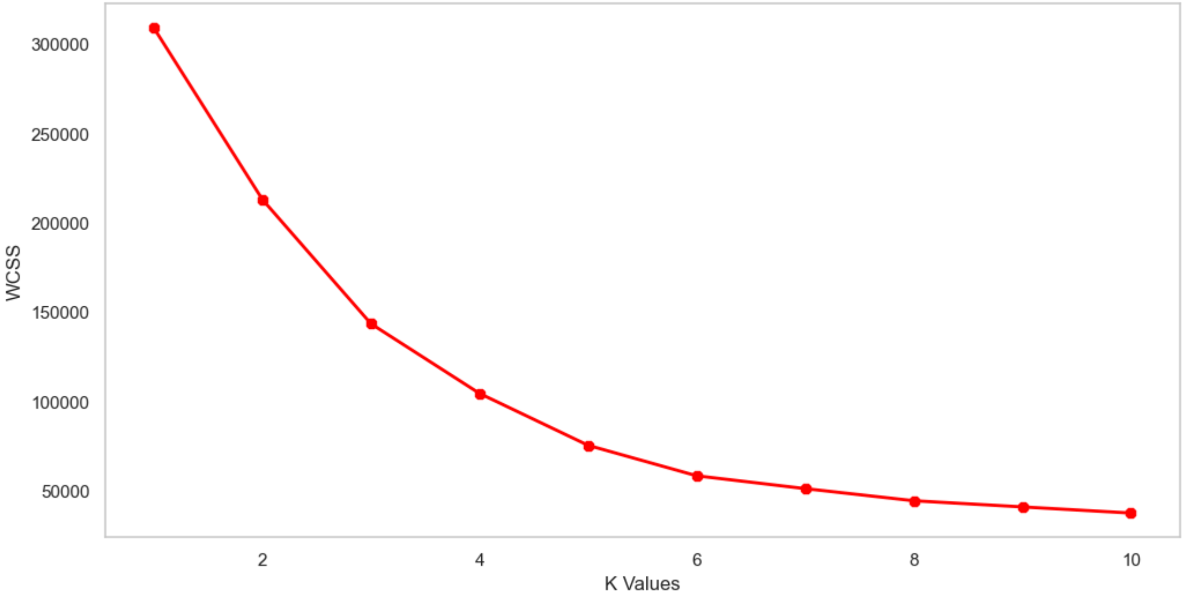
```
plt.scatter(X2[:,0], X1[:,1],c = kmeans.labels_,cmap='rainbow')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],color = 'blue
plt.title('Clusters of Customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score(1-100)')
plt.show()
```



In [25]:

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

from sklearn.cluster import KMeans
wcss = []  # Initialize an empty list
for k in range(1, 11):  # Fix the range syntax
    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 Values")
plt.ylabel("WCSS")
plt.show()
```

```
C:\Users\DEEPIKA\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment va
riable OMP_NUM_THREADS=1.
  warnings.warn(
```



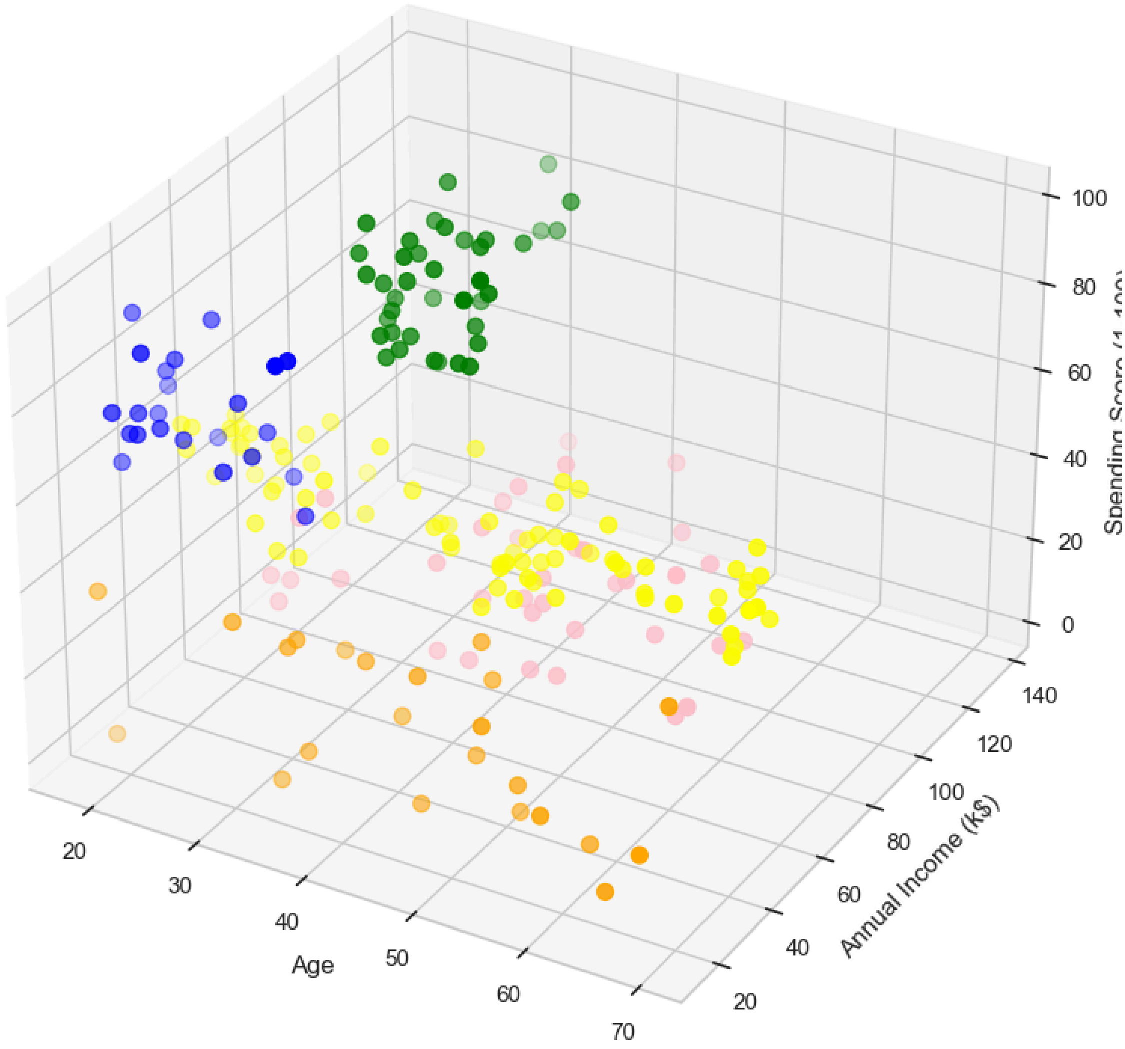
In [26]:
```
kmeans =KMeans(n_clusters=5)

label = kmeans.fit_predict(X3)

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

In [27]:
```
print(kmeans.cluster_centers_)
```

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

In [28]:
```
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

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

plt.show()
```



100
80
60
40
20
0
Spending Score (1-100)
140
120
100
80
60
40
20
Annual Income (k$)
20
30
40
50
60
70
Age

In [ ]: