In [ ]:

```
# Customer Segmentation using K-Means Algorithm
```

In [1]:

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

In [4]:

```
# Importing dataset(Mall_customers (1))
df=pd.read_csv(r"C:\Users\KAAVYA\Downloads\Mall_Customers (1).csv")
df.head()
```

Out[4]:

	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 [5]:

```
# Checking rows and columns of Dataset
df.shape
```

Out[5]:

```
(200, 5)
```

In [7]:

```
# Infomation regarding Dataset
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 [9]:

```
# Checking Data Types
df.dtypes
```

Out[9]:

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

In [10]:

```
# Cheacking nullvalues in Dataset
df.isnull().sum()
```

Out[10]:

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

In [14]:

```
# Deleting Customer ID column from Dataset
df.drop(["CustomerID"],axis=1,inplace=True)
df.head()
```

Out[14]:

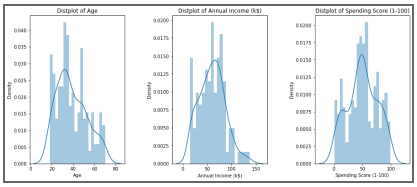
	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40

In [16]:

```
# Data visualization by plotting Distribution plot graph and analyzing Trends
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)
```

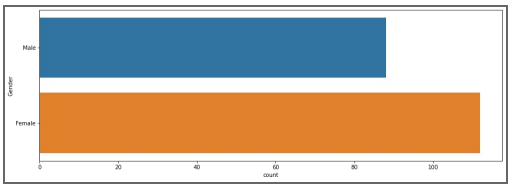
```
        plt.subplots_adjust(hspace= 0.5, wspace=0.5)
        sns.distplot(df[x] , bins=20)
        plt.title('Distplot of {}'.format(x))
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



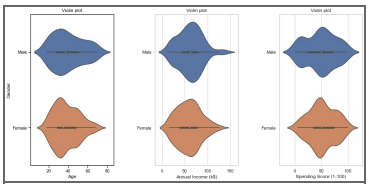
In [20]:

```
# Plotting "Countplot Graph" to show Comparsion between number of females and males
plt.figure(figsize=(15,5))
sns.countplot(y='Gender',data=df)
plt.show()
```



In [22]:

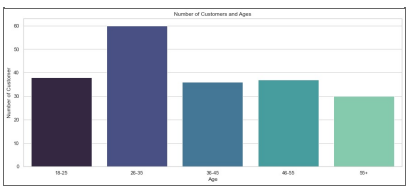
```
# Plotting "ViolinPlot" of Age, Annual Income (k$) and Spending Score (1-100) based on Gend
er Distribution
plt.figure(1,figsize=(15,7))
n=0
for cols in ['Age','Annual Income (k$)','Spending Score (1-100)']:
        n+=1
        plt.subplot(1,3,n)
        sns.set(style="whitegrid")
        plt.subplots_adjust(hspace= 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 [24]:

```
# Dividing the ages in different categories to get which range has the highest number of Cu
stomers by plotting Bar Graph
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>=55]

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_55.value
s),len(age_55above.values)]
plt.figure(figsize=(15,6))
sns.barplot(x=agex, y=agey, palette="mako")
plt.title("Number of Customers and Ages")
plt.xlabel("Age")
plt.ylabel("Number of Customer")
plt.show()
```

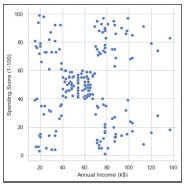


In [26]:

```
# Understanding relationship Between Annual Income (k$) and Spending Score (1-100)
sns.relplot(x='Annual Income (k$)', y='Spending Score (1-100)',data=df)
```

Out[26]:

```
<seaborn.axisgrid.FacetGrid at 0x166a28ad160>
```

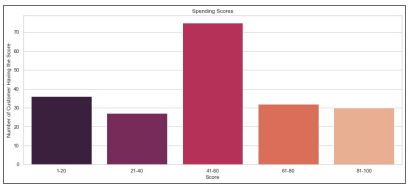


In [31]:

```
# Dividing the Spending Score (1-100) in different categories to get which range has the hi
ghest number of Customers
# By plotting Bar Graph
ss_1_20=df["Spending Score (1-100)"][(df["Spending Score (1-100)"]>=1) &(df["Spending Score
 (1-100)"]<=20)]
ss_21_40=df["Spending Score (1-100)"][(df["Spending Score (1-100)"]>=21) &(df["Spending Sco
re (1-100)"]<=40)]
ss_41_60=df["Spending Score (1-100)"][(df["Spending Score (1-100)"]>=41) &(df["Spending Sco
re (1-100)"]<=60)]
ss_61_80=df["Spending Score (1-100)"][(df["Spending Score (1-100)"]>=61) &(df["Spending Sco
re (1-100)"]<=80)]
ss_81_100=df["Spending Score (1-100)"][(df["Spending Score (1-100)"]>=81) &(df["Spending Sc
ore (1-100)"]<=100)]

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.values),len
(ss_81_100.values)]

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()
```



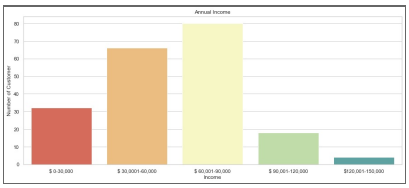
In [34]:

```
# Dividing the Annual Income (k$) in different categories to get which range has the highes
t number of Customers
# By plotting Bar Graph
ai0_30 = df["Annual Income (k$)"][(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","$ 30,0001-60,000","$ 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(ai
121_150.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()
```

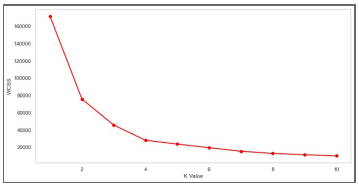


In [45]:

```
# Finding out the Optum number of clusters between Age and Spending Score (1-100)
# By importing Kmeans algorithm
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:\ProgramData\Anaconda3\lib\site-packa
ges\sklearn\cluster\_kmeans.py:881: Use
rWarning: KMeans is known to have a mem
ory leak on Windows with MKL, when ther
e are less chunks than available thread
s. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



In [46]:

```
# Creating optum number of clusters = 4
kmeans =KMeans(n_clusters=4)
label=kmeans.fit_predict(X1)
print(label)
```

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

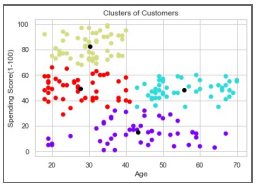
In [47]:

```
# Print the cluster_centers values
print(kmeans.cluster_centers_)
```

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

In [51]:

```
# Visulization the clusters by plotting Scatter plot
# Black dot represent cluster_centers values
# We see four different groups in graph
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()
```



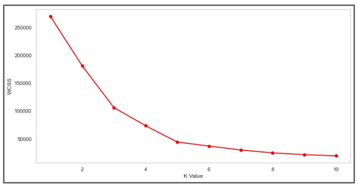
In [53]:

```
# Finding out the Optum number of clusters between Annual Income (k$) and Spending Score (1
-100)
# By importing Kmeans algorithm
X2=df.loc[:, ["Annual Income (k$)","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:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: 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 variable OMP_NUM_THREADS=1.
  warnings.warn(
```



In [54]:

```
# Creating optum number of clusters = 5
kmeans =KMeans(n_clusters=5)
label=kmeans.fit_predict(X2)
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 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 1 1 1 1 1 3 2 3 1 3 2 3
2 3 1 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 2 3]
```

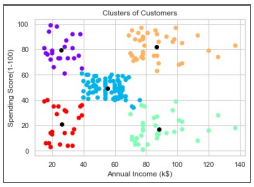
In [55]:

```
# Print the cluster_centers values
print(kmeans.cluster_centers_)
```

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

In [57]:

```
# Visulization the clusters by plotting Scatter plot
# Black dot represent cluster_centers values
# We see five different groups in graph
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('Annual Income (k$)')
plt.ylabel('Spending Score(1-100)')
plt.show()
```

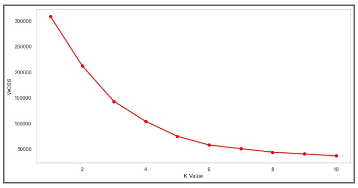


In [58]:

```
# Finding out the Optum number of clusters between Age, Annual Income (k$) and Spending Sco
re (1-100)
# By importing Kmeans algorithm
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:\ProgramData\Anaconda3\lib\site-packa
ges\sklearn\cluster\_kmeans.py:881: Use
rWarning: KMeans is known to have a mem
ory leak on Windows with MKL, when ther
e are less chunks than available thread
s. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



In [59]:

```
# Creating optum number of clusters = 5
kmeans =KMeans(n_clusters=5)
label=kmeans.fit_predict(X3)
print(label)
```

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

In [60]:

```
# Print the cluster_centers values
print(kmeans.cluster_centers_)
```

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

In [65]:

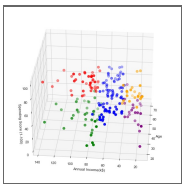
```
# Visulization the clusters by plotting Scatter plot in 3d
# Black dot represent cluster_centers values
# We see five different groups in graph
clusters = kmeans.fit_predict(X3)
df['label'] = clusters

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$)"][df.label ==0],df["Spending Score (
1-100)"][df.label==0],c='blue',s=60)
ax.scatter(df.Age[df.label ==1],df["Annual Income (k$)"][df.label ==1],df["Spending Score (
1-100)"][df.label==1],c='red',s=60)
```

```
ax.scatter(df.Age[df.label ==2],df["Annual Income (k$)"][df.label ==2],df["Spending Score (
1-100)"][df.label==2],c='green',s=60)
ax.scatter(df.Age[df.label ==3],df["Annual Income (k$)"][df.label ==3],df["Spending Score (
1-100)"][df.label==3],c='orange',s=60)
ax.scatter(df.Age[df.label ==4],df["Annual Income (k$)"][df.label ==4],df["Spending Score (
1-100)"][df.label==4],c='purple',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 [ ]:

```
# End project
```