CUSTOMER SEGMENTATION

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
from sklearn.cluster import KMeans
```

In [4]:
```
cus_data=pd.read_csv('C:/Users/ASUS/Desktop/CUSTOMER SEGMENTATION/Mall_Customers.csv')
```

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

Out[5]:

	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 [6]:
```
cus_data.shape
```

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

In [7]:
```
cus_data.isna().sum()
```

Out[7]:
```
CustomerID                0
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [8]:
```
cus_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Genre                   200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

In [11]:
```
cus_data.drop(['CustomerID','Genre'],axis=1,inplace=True)
```

In [14]:
```
cus_data.head(3)
```

Out[14]:

	Age	Annual Income (k$)	Spending Score (1-100)
0	19	15	39
1	21	15	81
2	20	16	6

In [15]:
```
X=cus_data.drop('Age',axis=1)
```

In [16]:
```
X.head(3)
```

Out[16]:

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6

Chossing Optimum Number of Clusters using WCSS

For each value of K, we are calculating WCSS (Within-Cluster Sum of Square). WCSS is the sum of squared distance between each point and the centroid in a cluster. When we plot the WCSS with the K value, the plot looks like an Elbow. As the number of clusters increases, the WCSS value will start to decrease.

In [20]:
```
## finding wcss value for different number of clusters
WCSS=[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=0)
    kmeans.fit(X)

    WCSS.append(kmeans.inertia_)
```

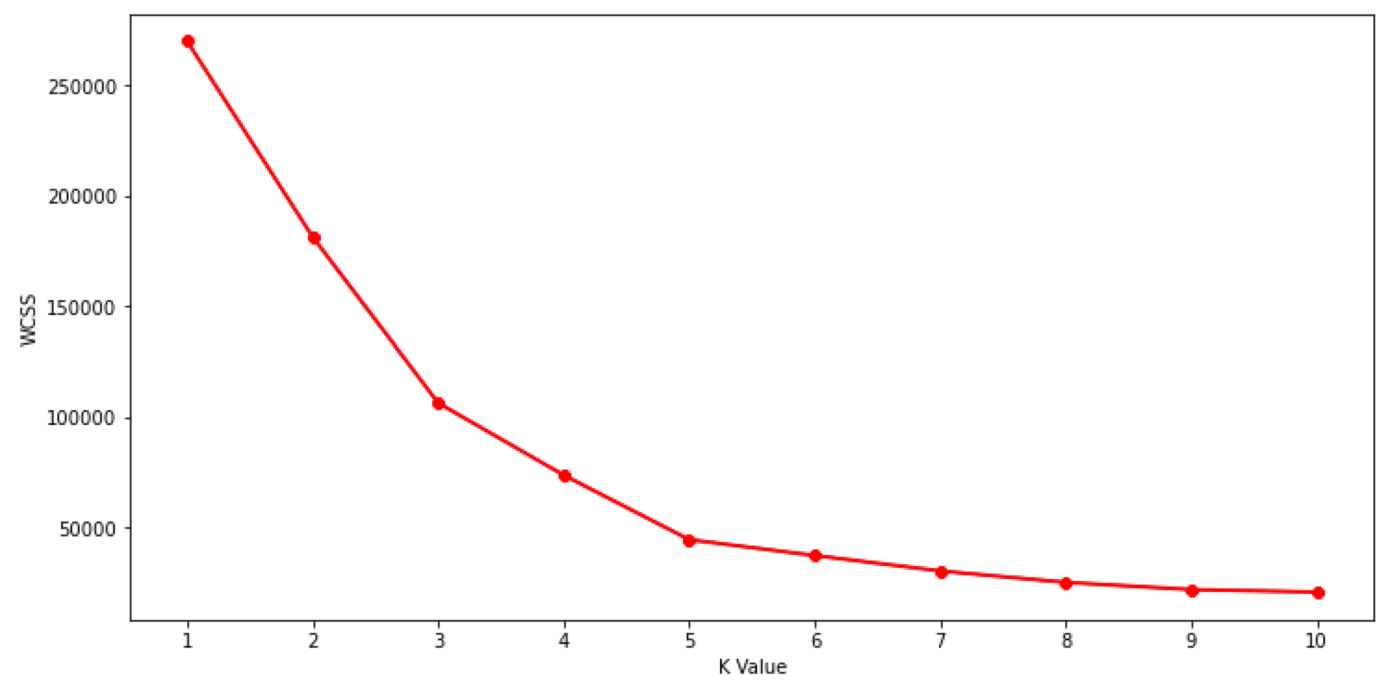
```
C:\Users\ASUS\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than av
ailable threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Inertia measures how well a dataset was clustered by K-Means. It is calculated by measuring the distance between each data point and its centroid, squaring this distance, and summing these squares across one cluster

In [21]:
```
WCSS
```

Out[21]:
```
[269981.28000000014,
 181363.59595959607,
 106348.37306211119,
 73679.78903948837,
 44448.45544793369,
 37265.86520484345,
 30259.657207285458,
 25095.703209997544,
 21830.04197804944,
 20736.67993892413]
```

In [22]:
```
# plot an elbow graph
#The elbow curve
plt.figure(figsize=(12,6))
plt.plot(range(1,11),WCSS)
plt.plot(range(1,11),WCSS, linewidth=2, color="red", marker ="8")
plt.xlabel("K Value")
plt.xticks(np.arange(1,11,1))
plt.ylabel("WCSS")
plt.show()
```



We find that there are 2 cut-off(elbow points[3,5]),so we take 5 cuz after that there is no sharp drop in graph

In [26]:
```
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=0)

# return a label for each data point based on their cluster
Y = kmeans.fit_predict(X)
print(Y)
```

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

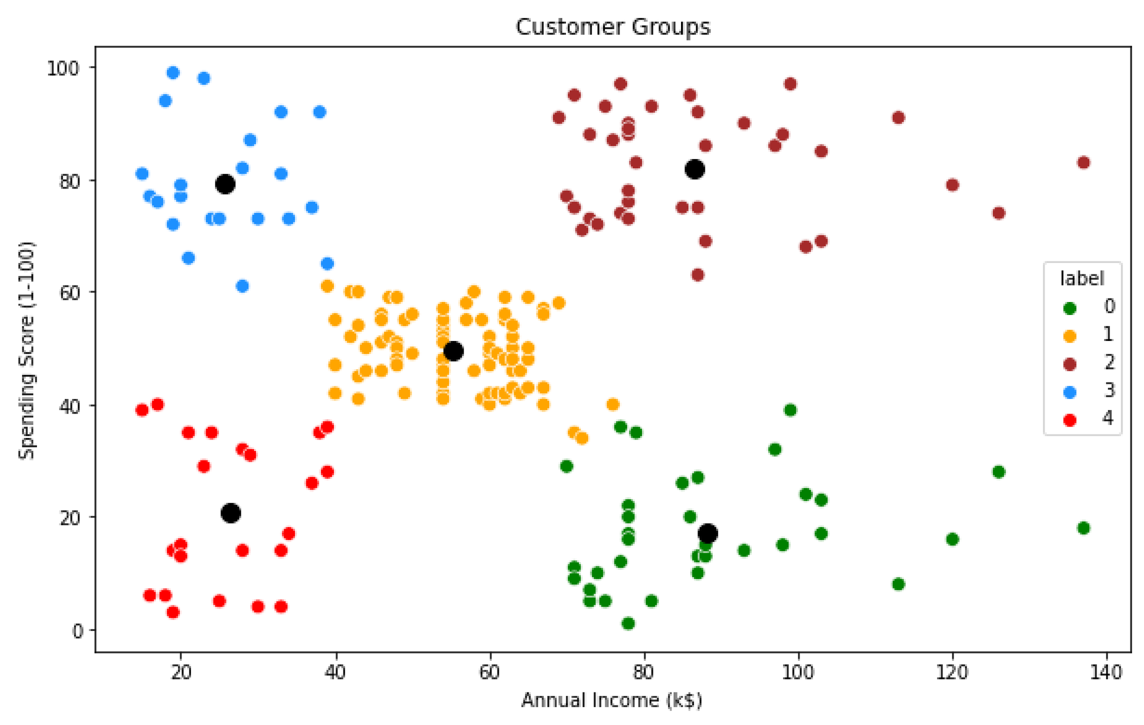
In [27]:
```
#adding the labels to a column named label
cus_data["label"] = Y
```

In [28]:
```
cus_data.head(3)
```

Out[28]:

	Age	Annual Income (k$)	Spending Score (1-100)	label
0	19	15	39	4
1	21	15	81	3
2	20	16	6	4

In [29]:
```
#Scatterplot of the clusters
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)',hue="label",
palette=['green','orange','brown','dodgerblue','red'], legend='full',data = cus_data ,s = 60 )
# plot the centroids
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=100, c='black', label='Centroids')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Customer Groups')
plt.show()
```



In [ ]: