St. Francis Institute of Technology, Mumbai-400 103

Department of Information Technology

A.Y. 2024-2025

Class: TE-ITA/B, Semester: VI

Subject: **Data Science Lab**

Experiment – 7: To implement Clustering.

1. **Aim:** To implement unsupervised learning with clustering concepts.
2. **Objectives:** After study of this experiment, the student will be able to
   - Understand clustering.
3. **Outcomes:** After study of this experiment, the student will be able to
   - Understand concepts of clustering in data science.
4. **Prerequisite:** Fundamentals of Python Programming and Database Management System.
5. **Requirements:** Python Installation, Personal Computer, Windows operating system, Internet Connection, Microsoft Word.
6. **Pre-Experiment Exercise:**

Brief Theory:

- Concept of clustering machine learning. (Naive Byes, ID3, KNN, Random Forest)

Laboratory Exercise

A. **Procedure:** (Mall_Customers Dataset)

Refer a separate DSL_EXPT7_Clustering.ipynb file for commands.

B. Paste Screenshots of above commands.

8. Post-Experiments Exercise

A. Extended Theory: (Soft Copy)

- Types of clustering.

B. Questions:

- Application of Clustering

C. Conclusion:

Write the significance of the topic studied in the experiment.

D. References:

1. https://machinelearningmastery.com/clustering-algorithms-with-python/

Name: Vishal Rajesh Mahajan AIDS EXP 07

Class: TE IT A Roll No: 56

A. Extended Theory:

Types of clustering..

Partitioning Clustering

In partitioning clustering, the dataset is divided into a fixed number of distinct non-overlapping clusters. Each data point belongs to exactly one cluster. The most widely used algorithm in this category is K-Means, where the objective is to minimize the distance between data points and the center of their assigned clusters. It is simple and efficient but requires the number of clusters to be specified in advance.

Hierarchical Clustering

Hierarchical clustering builds a tree-like structure of clusters known as a dendrogram. It can be done in two ways: agglomerative (bottom-up) and divisive (top-down). In agglomerative clustering, each data point starts in its own cluster, and clusters are merged step by step. In divisive clustering, all data points start in one cluster and are split recursively. This method does not require specifying the number of clusters beforehand.

Density-Based Clustering

Density-based clustering groups data points that are closely packed together while marking points in low-density regions as outliers. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a popular algorithm that can form clusters of arbitrary shape and is robust to noise. It performs well on datasets with irregular cluster shapes and varying densities.

Model-Based Clustering

Model-based clustering assumes that data is generated from a mixture of underlying probability distributions. It tries to estimate the parameters of these distributions to identify clusters. Gaussian Mixture Models (GMM) are commonly used in this type, where each cluster is represented by a Gaussian distribution. This method provides flexibility and probabilistic interpretation of clustering.

Grid-Based Clustering

Grid-based clustering involves dividing the data space into a finite number of cells that form a grid structure. Clustering is performed on these cells rather than the individual data points. This approach is efficient for large datasets and reduces computation time. STING (Statistical Information Grid) is one of the well-known grid-based clustering methods.

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

In [7]:

```
data=pd.read_excel("Mall_Customers.xlsx")
```

In [9]:

```
print("Number of customers we have data for-" , len(data))
```

```
Number of customers we have data for- 200
```

In [10]:

```
data.head()
```

Out[10]:

	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 [14]:

```
data.corr(numeric_only=True)
```

Out[14]:

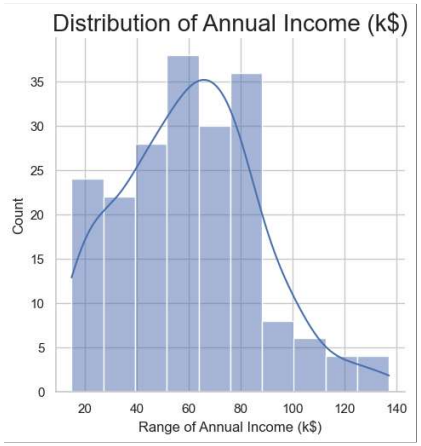
	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
CustomerID	1.000000	-0.026763	0.977548	0.013835
Age	-0.026763	1.000000	-0.012398	-0.327227
Annual Income (k$)	0.977548	-0.012398	1.000000	0.009903
Spending Score (1-100)	0.013835	-0.327227	0.009903	1.000000

In [15]:

```
#Distribution of Annnual Income

plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.displot(data['Annual Income (k$)'],kde="true")
plt.title('Distribution of Annual Income (k$)', fontsize = 20)
plt.xlabel('Range of Annual Income (k$)')
plt.ylabel('Count')
```

Out[15]:

```
Text(16.819444444444443, 0.5, 'Count')
<Figure size 1000x600 with 0 Axes>
```

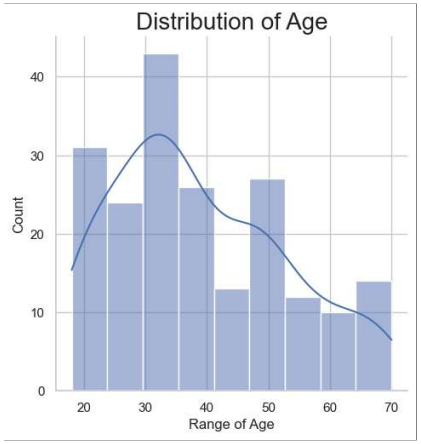


In [48]:

```
#Distribution of age

plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.displot(data['Age'],kde="true")
plt.title('Distribution of Age', fontsize = 20)
plt.xlabel('Range of Age')
plt.ylabel('Count')
```

Out[48]:

```
Text(16.819444444444443, 0.5, 'Count')
<Figure size 1000x600 with 0 Axes>
```

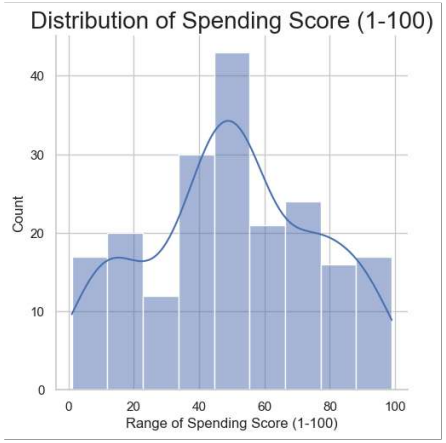


In [49]:

```
#Distribution of spending score

plt.figure(figsize=(10, 6))
sns.set(style = 'whitegrid')
sns.displot(data['Spending Score (1-100)'],kde="true")
plt.title('Distribution of Spending Score (1-100)', fontsize = 20)
plt.xlabel('Range of Spending Score (1-100)')
plt.ylabel('Count')
```

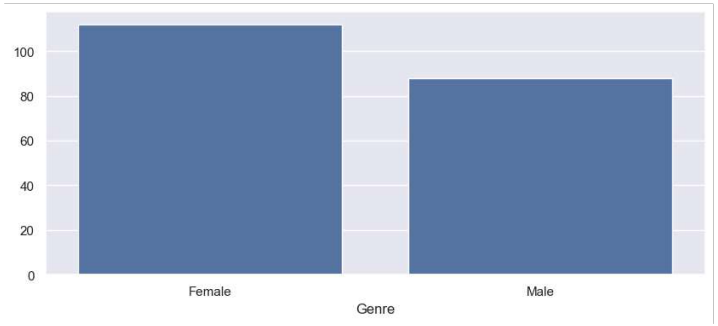
Out[49]:

```
Text(16.819444444444443, 0.5, 'Count')
<Figure size 1000x600 with 0 Axes>
```



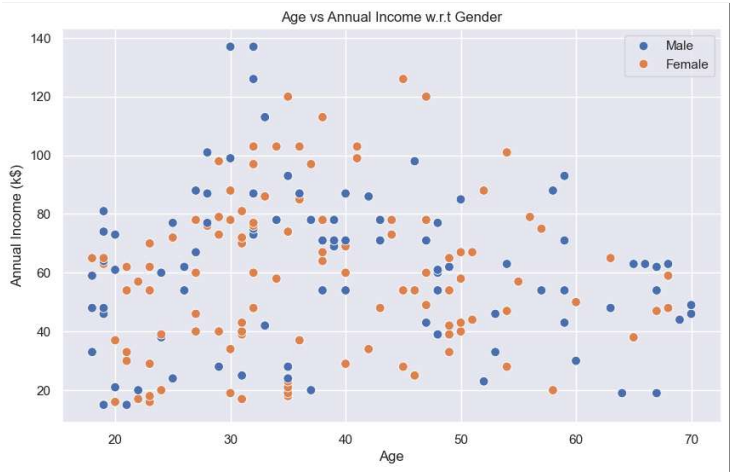
In [18]:

```
# Gender Analysis
genders = data.Genre.value_counts()
sns.set_style("darkgrid")
plt.figure(figsize=(10,4))
sns.barplot(x=genders.index, y=genders.values)
plt.show()
```



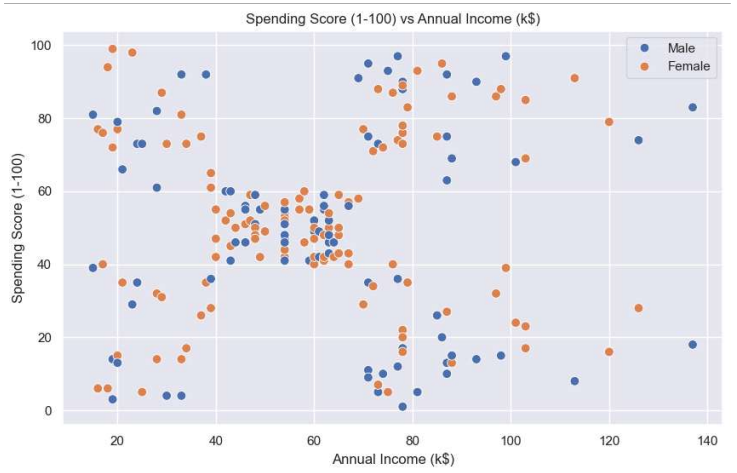
In [19]:

```
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Age' , y = 'Annual Income (k$)' , hue="Genre",data = data  ,s = 60 )
plt.xlabel('Age'), plt.ylabel('Annual Income (k$)')
plt.title('Age vs Annual Income w.r.t Gender')
plt.legend()
plt.show()
```



In [20]:

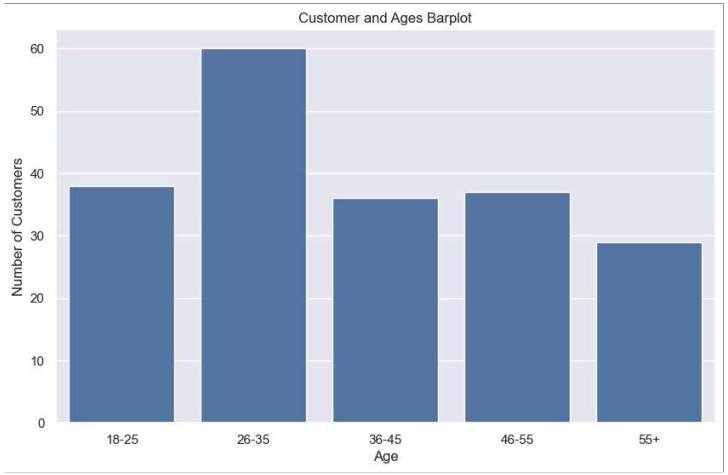
```
plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)'  , hue="Genre",data = data  ,s = 60
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.legend()
plt.show()
```



In [21]:

```
age18_25 = data.Age[(data.Age <= 25) & (data.Age >= 18)]
age26_35 = data.Age[(data.Age <= 35) & (data.Age >= 26)]
age36_45 = data.Age[(data.Age <= 45) & (data.Age >= 36)]
age46_55 = data.Age[(data.Age <= 55) & (data.Age >= 46)]
age55above = data.Age[data.Age >= 56]

x = ["18-25","26-35","36-45","46-55","55+"]
y = [len(age18_25.values),len(age26_35.values),len(age36_45.values),len(age46_55.values),len(age55above.va

plt.figure(figsize=(10,6))
sns.barplot(x=x, y=y)
plt.title("Customer and Ages Barplot")
plt.xlabel("Age")
```

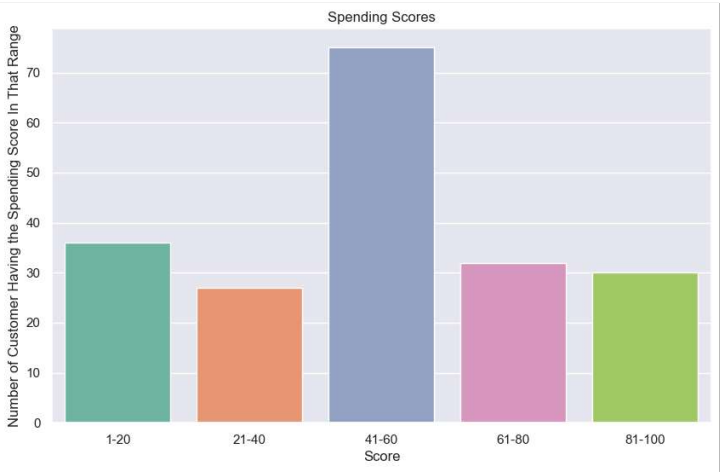
```
plt.ylabel("Number of Customers")
plt.show()
```



In [22]:

```
ss1_20 = data["Spending Score (1-100)"][(data["Spending Score (1-100)"] >= 1) & (data["Spending Score (1-1
ss21_40 = data["Spending Score (1-100)"][(data["Spending Score (1-100)"] >= 21) & (data["Spending Score (1
ss41_60 = data["Spending Score (1-100)"][(data["Spending Score (1-100)"] >= 41) & (data["Spending Score (1
ss61_80 = data["Spending Score (1-100)"][(data["Spending Score (1-100)"] >= 61) & (data["Spending Score (1
ss81_100 = data["Spending Score (1-100)"][(data["Spending Score (1-100)"] >= 81) & (data["Spending Score (

score_x = ["1-20", "21-40", "41-60", "61-80", "81-100"]
score_y = [len(ss1_20.values), len(ss21_40.values), len(ss41_60.values), len(ss61_80.values), len(ss81_100

plt.figure(figsize=(10,6))
sns.barplot(x=score_x, y=score_y,palette="Set2")
plt.title("Spending Scores")
plt.xlabel("Score")
plt.ylabel("Number of Customer Having the Spending Score In That Range")
plt.show()
```

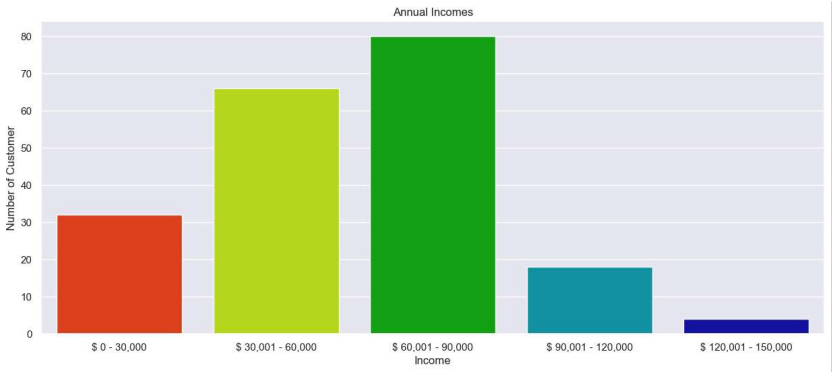
```
C:\Users\vism0\AppData\Local\Temp\ipykernel_27028\1653149555.py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=score_x, y=score_y,palette="Set2")
```



In [23]:

```
ai0_30 = data["Annual Income (k$)"][(data["Annual Income (k$)"] >= 0) & (data["Annual Income (k$)"] <= 30)
ai31_60 = data["Annual Income (k$)"][(data["Annual Income (k$)"] >= 31) & (data["Annual Income (k$)"] <= 6
ai61_90 = data["Annual Income (k$)"][(data["Annual Income (k$)"] >= 61) & (data["Annual Income (k$)"] <= 9
ai91_120 = data["Annual Income (k$)"][(data["Annual Income (k$)"] >= 91) & (data["Annual Income (k$)"] <=
ai121_150 = data["Annual Income (k$)"][(data["Annual Income (k$)"] >= 121) & (data["Annual Income (k$)"] <

income_x = ["$ 0 - 30,000", "$ 30,001 - 60,000", "$ 60,001 - 90,000", "$ 90,001 - 120,000", "$ 120,001 - 1
income_y = [len(ai0_30.values), len(ai31_60.values), len(ai61_90.values), len(ai91_120.values), len(ai121_

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

```
C:\Users\vism0\AppData\Local\Temp\ipykernel_27028\723480284.py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.
0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=income_x, y=income_y, palette="nipy_spectral_r")
```



In [24]:

```
data
```

Out[24]:

	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

In [25]:

```
#We take just the Annual Income and Spending score
df1=data[["CustomerID","Genre","Age","Annual Income (k$)","Spending Score (1-100)"]]

X=df1[["Annual Income (k$)","Spending Score (1-100)"]]
```

In [26]:

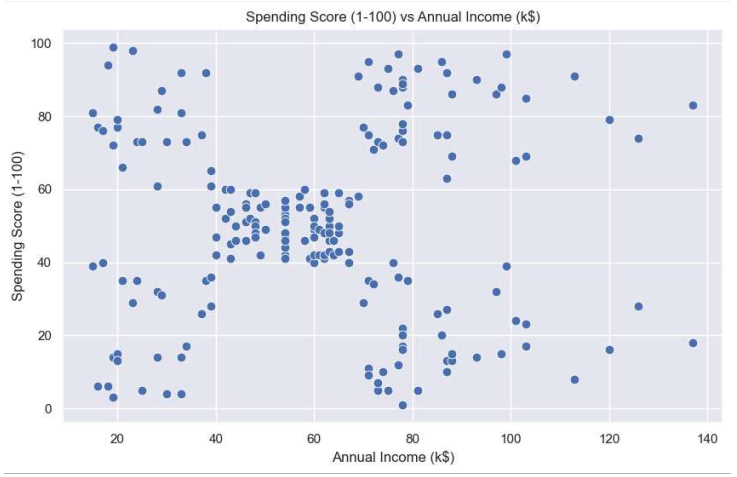
```
X.head()
```

Out[26]:

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40

In [27]:

```
#Scatterplot of the input data

plt.figure(figsize=(10,6))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)',  data = X  ,s = 60 )
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.show()
```



In [28]:

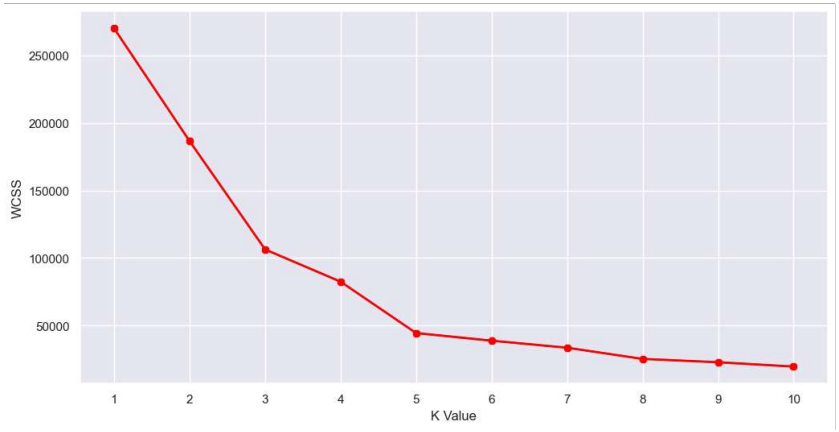
```
from sklearn.cluster import KMeans
```

In [ ]:

```
wcss=[]

for i in range(1,11):
    km=KMeans(n_clusters=i)
    km.fit(X)
    wcss.append(km.inertia_)
```

In [30]:

```
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()
```



In [31]:

```
km1=KMeans(n_clusters=5)
```

In [32]:

```
X
```

Out[32]:

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40
...	...	...
195	120	79
196	126	28
197	126	74
198	137	18
199	137	83

200 rows × 2 columns

In [33]:

```
#Fitting the input data

km1.fit(X)
```

Out[33]:

KMeans

KMeans(n_clusters=5)

In [34]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=5, n_init=10,random_state=None, tol=0.0001, verbose=0)
```

Out[34]:

KMeans

KMeans(algorithm='auto', n_clusters=5, n_init=10)

In [35]:

```
#predicting the labels of the input data

y=km1.predict(X)
```

In [36]:

```
#adding the labels to a column named label

df1["label"] = y
```

In [37]:

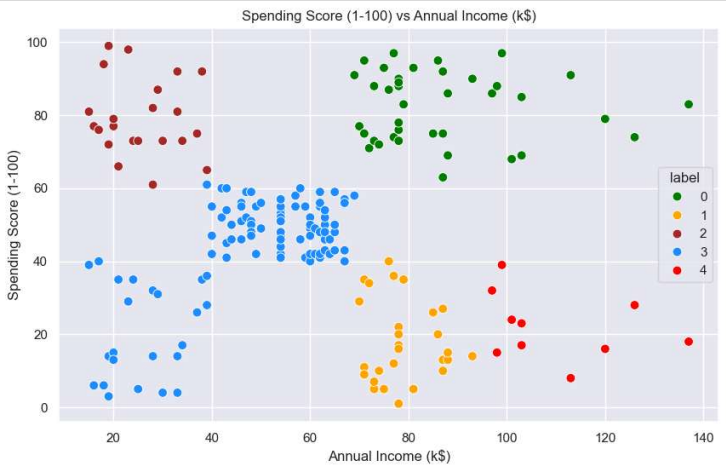
```
#The new dataframe with the clustering done

df1.head()
```

Out[37]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)	label
0	1	Male	19	15	39	3
1	2	Male	21	15	81	2
2	3	Female	20	16	6	3
3	4	Female	23	16	77	2
4	5	Female	31	17	40	3

In [38]:

```
#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 = df1  ,s = 60

plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.show()
```



In [39]:

```
cust1=df1[df1["label"]==1]
print('Number of customer in 1st group=', len(cust1))
print('They are -', cust1["CustomerID"].values)
print("----------------------------------------")
cust2=df1[df1["label"]==2]
print('Number of customer in 2nd group=', len(cust2))
print('They are -', cust2["CustomerID"].values)
print("----------------------------------------")
cust3=df1[df1["label"]==0]
print('Number of customer in 3rd group=', len(cust3))
print('They are -', cust3["CustomerID"].values)
print("----------------------------------------")
cust4=df1[df1["label"]==3]
print('Number of customer in 4th group=', len(cust4))
print('They are -', cust4["CustomerID"].values)
```

```
print("----------------------------------------")
cust5=df1[df1["label"]==4]
print('Number of customer in 5th group=', len(cust5))
print('They are -', cust5["CustomerID"].values)
print("----------------------------------------")
```

```
Number of customer in 1st group= 28
They are - [125 127 129 131 133 135 137 139 141 143 145 147 149 151 153 155 157 159
 161 163 165 167 169 171 173 175 177 179]
----------------------------------------
Number of customer in 2nd group= 22
They are - [ 2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 46]
----------------------------------------
Number of customer in 3rd group= 39
They are - [124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156 158
 160 162 164 166 168 170 172 174 176 178 180 182 184 186 188 190 192 194
 196 198 200]
----------------------------------------
Number of customer in 4th group= 101
They are - [  1   3   5   7   9  11  13  15  17  19  21  23  25  27  29  31  33  35
  37  39  41  43  44  45  47  48  49  50  51  52  53  54  55  56  57  58
  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76
  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94
  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112
 113 114 115 116 117 118 119 120 121 122 123]
----------------------------------------
Number of customer in 5th group= 10
They are - [181 183 185 187 189 191 193 195 197 199]
----------------------------------------
```

In [40]:

```
#Now we shall take 3 input features

df2=data[["CustomerID","Genre","Age","Annual Income (k$)","Spending Score (1-100)"]]
df2.head()
```

Out[40]:

	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 [41]:

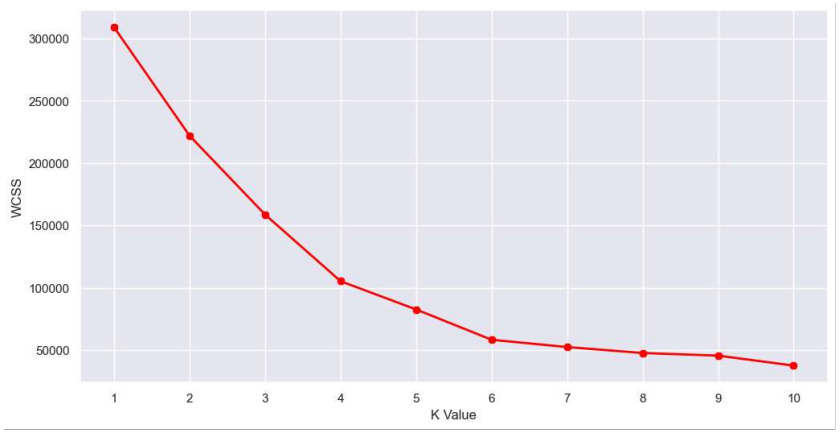
```
#Taking the features

X2=df2[["Age","Annual Income (k$)","Spending Score (1-100)"]]
```

In [42]:

```
#Now we calculate the Within CLuster Sum of Squared Errors (WSS) for different values of k.


wcss = []
for k in range(1,11):
    kmeans = KMeans(n_clusters=k, init="k-means++")
    kmeans.fit(X2)
    wcss.append(kmeans.inertia_)
```

In [43]:

```
plt.figure(figsize=(12,6))

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()
```



In [44]:

```
#We choose the k for which WSS starts to diminish

km2 = KMeans(n_clusters=5)
y2 = km.fit_predict(X2)
df2["label"] = y2
```

In [45]:

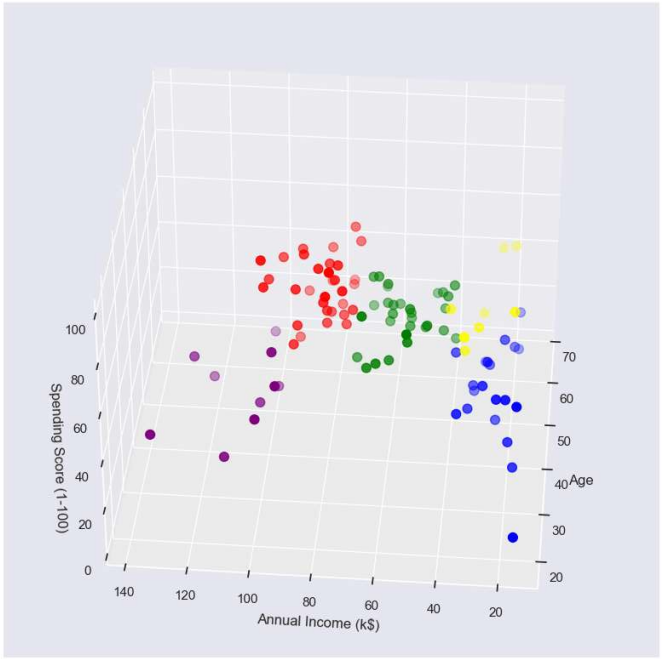
```
df2.head()
```

Out[45]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)	label
0	1	Male	19	15	39	7
1	2	Male	21	15	81	7
2	3	Female	20	16	6	2
3	4	Female	23	16	77	7
4	5	Female	31	17	40	2

In [46]:

```
#3D Plot as we did the clustering on the basis of 3 input features

fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df2.Age[df2.label == 0], df2["Annual Income (k$)"][df2.label == 0], df2["Spending Score (1-100)
ax.scatter(df2.Age[df2.label == 1], df2["Annual Income (k$)"][df2.label == 1], df2["Spending Score (1-100)
ax.scatter(df2.Age[df2.label == 2], df2["Annual Income (k$)"][df2.label == 2], df2["Spending Score (1-100)
ax.scatter(df2.Age[df2.label == 3], df2["Annual Income (k$)"][df2.label == 3], df2["Spending Score (1-100)
ax.scatter(df2.Age[df2.label == 4], df2["Annual Income (k$)"][df2.label == 4], df2["Spending Score (1-100)
ax.view_init(35, 185)
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel('Spending Score (1-100)')
plt.show()
```



In [47]:

```
cust1=df2[df2["label"]==1]
print('Number of customer in 1st group=', len(cust1))
print('They are -', cust1["CustomerID"].values)
print("----------------------------------------")
cust2=df2[df2["label"]==2]
print('Number of customer in 2nd group=', len(cust2))
print('They are -', cust2["CustomerID"].values)
print("----------------------------------------")
cust3=df2[df2["label"]==0]
print('Number of customer in 3rd group=', len(cust3))
print('They are -', cust3["CustomerID"].values)
print("----------------------------------------")
cust4=df2[df2["label"]==3]
print('Number of customer in 4th group=', len(cust4))
print('They are -', cust4["CustomerID"].values)
print("----------------------------------------")
cust5=df2[df2["label"]==4]
print('Number of customer in 5th group=', len(cust5))
print('They are -', cust5["CustomerID"].values)
print("----------------------------------------")
```

```
Number of customer in 1st group= 32
They are - [124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156 158
 160 162 164 166 168 170 172 174 176 178 180 182 184 186]
----------------------------------------
Number of customer in 2nd group= 20
They are - [ 3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 45]
----------------------------------------
Number of customer in 3rd group= 10
They are - [181 183 185 187 189 191 193 195 197 199]
----------------------------------------
Number of customer in 4th group= 30
They are - [ 43  47  51  55  56  57  60  67  72  77  78  80  82  84  86  90  93  94
  97  99 102 105 108 113 118 119 120 122 123 127]
----------------------------------------
Number of customer in 5th group= 8
They are - [ 8 12 20 26 30 34 36 42]
----------------------------------------
```