CUSTOMER SEGMENTATION USING DATA SCIENCE

INTRODUCTION:

Customer segmentation is a crucial practice in data science that involves dividing a customer base into groups that share similar characteristics such as demographics, behavior, or preferences. This process enables businesses to understand their customers better, tailor marketing strategies, and enhance overall customer experience. By leveraging techniques such as clustering algorithms, machine learning, and data mining, businesses can gain insights into customer behavior patterns, preferences, and needs, leading to more effective targeting and personalized offerings.

Given Dataset:

	A	B	C	D	E	F
1	CustomerID	Genre	Age	Annual Income (	Spending Score (1-100)	
2	1	Male	19	15	39	
3	2	Male	21	15	81	
4	3	Female	20	16	6	
5	4	Female	23	16	77	
6	5	Female	31	17	40	
7	6	Female	22	17	76	
8	7	Female	35	18	6	
9	8	Female	23	18	94	
10	9	Male	64	19	3	
11	10	Female	30	19	72	
12	11	Male	67	19	14	
13	12	Female	35	19	99	
14	13	Female	58	20	15	
15	14	Female	24	20	77	
16	15	Male	37	20	13	
17	16	Male	22	20	79	
18	17	Female	35	21	35	
19	18	Male	20	21	66	

Overview of the Process:

The process of customer segmentation using data science generally involves several key steps:

1. **Data Collection:** Gather relevant data about customers, including demographic information, purchase history, interactions, and other relevant variables.

2. **Data Preprocessing:** Clean the data, handle missing values, and standardize or normalize the data to ensure consistency and accuracy.

3. **Feature Selection:** Identify the most relevant features or attributes that are crucial for segmenting customers effectively.

4. **Exploratory Data Analysis (EDA):** Conduct in-depth exploratory analysis to understand the distribution, correlations, and patterns within the data, using techniques such as data visualization and statistical analysis.

5. **Segmentation Technique Selection:** Choose appropriate data science techniques such as clustering algorithms (e.g., k-means, hierarchical clustering), classification models, or collaborative filtering, depending on the nature of the data and the specific business goals.

6. **Segmentation Model Building:** Apply the selected techniques to group customers into distinct segments based on similarities in their behaviors, preferences, or characteristics.

7. **Validation and Evaluation:** Assess the quality and effectiveness of the segmentation by using metrics such as silhouette scores, within-cluster sum of squares, or other domain-specific evaluation criteria.

8. **Interpretation of Segments:** Analyze and interpret the characteristics of each customer segment to gain actionable insights that can inform marketing strategies and business decisions.

9. **Implementation of Marketing Strategies:** Tailor marketing campaigns, product recommendations, and customer interactions based on the unique needs and preferences of each identified customer segment.

10. **Monitoring and Refinement:** Continuously monitor the performance of the segmentation strategy and refine the approach based on feedback, new data, and changes in customer behavior over time.

This iterative process enables businesses to understand their customers more deeply and create targeted marketing approaches that can lead to improved customer satisfaction and higher profitability.

PROCEDURE:

Features Explaination:

In a customer segmentation using data science, feature engineering plays a crucial role in preparing the data for effective analysis and segmentation. Here are some key aspects of feature engineering in this context:

1. **Feature Extraction:** Extract meaningful features from the raw data that can help in distinguishing different customer segments. This may involve deriving new variables from existing ones or transforming data into a more suitable format for analysis.

2. **Normalization and Scaling:** Normalize and scale the features to ensure that all variables are on a comparable scale, which is essential for certain algorithms like k-means clustering that are sensitive to the scale of the data.

3. **Encoding Categorical Variables:** Convert categorical variables into a format that can be easily interpreted by machine learning algorithms, such as one-hot encoding or label encoding, enabling the algorithms to process the data effectively.

4. **Handling Missing Data:** Implement strategies to handle missing data points, either through imputation techniques or by removing the data carefully, ensuring that the absence of certain data points does not skew the segmentation results.

5. **Creation of Derived Features:** Create new features based on domain knowledge or hypotheses that might be relevant in distinguishing customer segments. These derived features can be combinations of existing variables or interactions between different attributes.

6. **Dimensionality Reduction:** Employ dimensionality reduction techniques such as principal component analysis (PCA) or t-distributed stochastic neighbor embedding (t-SNE) to reduce the complexity of the dataset while retaining the most crucial information for segmentation.

7. **Temporal Feature Engineering:** Incorporate time-based features, seasonal patterns, or trends that can provide valuable insights into customer behavior changes over different time periods.

By carefully engineering features, data scientists can ensure that the data is optimized for segmentation, allowing for more accurate and meaningful identification of customer segments that can guide targeted marketing strategies and personalized customer experiences.

Applying Clustering Algorithms:

In customer segmentation, various clustering algorithms can be applied to group customers into distinct segments based on their similarities. Some commonly used clustering algorithms for customer segmentation include:

1. **K-means Clustering:** A popular and efficient algorithm that partitions the data into K clusters, minimizing the sum of squared distances between data points and the centroid of the clusters.

2. **Hierarchical Clustering:** Builds a hierarchy of clusters by either recursively merging or splitting clusters based on the distance between data points, creating a dendrogram that visually represents the clusters.

K-Means Clustering:

Import Libraries:

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns

import matplotlib.pyplot as plt
%matplotlib inline

from IPython.display import Image
```

Loading Data:

In [2]:

```
df = pd.read_csv('/kaggle/input/mall-customers/Mall_Customers.csv')
df.head()
```

Out[2]:

	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

Creating the Clusters:

In [3]:

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

In [4]:

```
wcss = []
```

In [5]:

```
for i in range(1,11):
    km = KMeans(n_clusters=i,init = 'k-means++',max_iter=300,n_init=10,ran
dom_state=0)
    km.fit(X)
    wcss.append(km.inertia_)
```

Applying K-means to mall dataset:

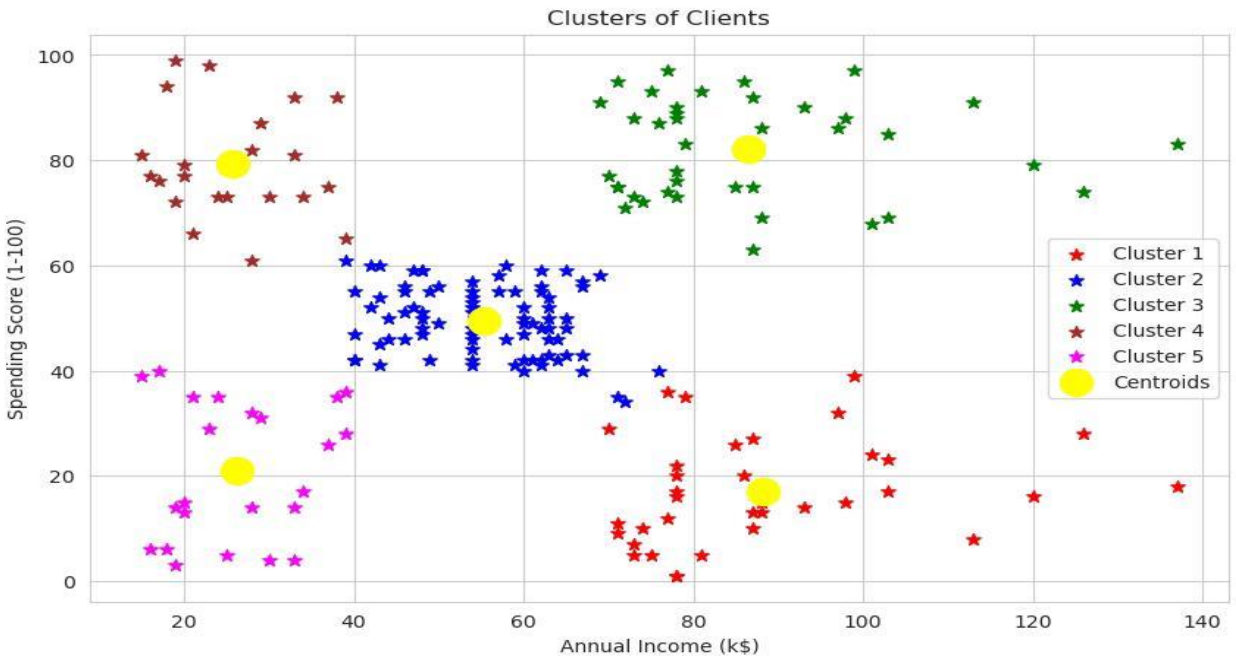
In [6]:

```
km = KMeans(n_clusters=5,init = 'k-means++',max_iter=300,n_init=10,random_
state=0) # setting default values for max_iter and n_init
y_means = km.fit_predict(X)
```

Visualizing the Clusters:

In [7]:

```
#Plotting scatter plot of clusters along with their highlighted clusters.
sns.set_style('whitegrid')
plt.figure(figsize=(10,6))
plt.scatter(X[y_means==0,0],X[y_means==0,1],s=50,c='red',label='Cluster 1'
,marker='*') #X[y_means==0,0] for x-coordinates for cluter1,X[y_means==0,1]
 for y-coordinates ,s for size of datapoint
plt.scatter(X[y_means==1,0],X[y_means==1,1],s=50,c='blue',label='Cluster 2
',marker='*')
plt.scatter(X[y_means==2,0],X[y_means==2,1],s=50,c='green',label='Cluster
3',marker='*')
plt.scatter(X[y_means==3,0],X[y_means==3,1],s=50,c='brown',label='Cluster
4',marker='*')
plt.scatter(X[y_means==4,0],X[y_means==4,1],s=50,c='magenta',label='Cluste
r 5',marker='*')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],s=250,c='yel
low',label='Centroids') #Centroids are highlighted with bigger size

plt.title('Clusters of Clients')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend();
```



- **Cluster 1 -** In this cluster, clients earn High income but don't spend much money.We could call these clients as "Careful"

- **Cluster 2 -** In this cluster, clients earn average income and have average spending score.Let's call them "Standard”

- **Cluster 3 -** In this cluster, clients earn High income and have high spending score.This is main potential customer base for mall's marketing campaigns and it would be very insightful for the mall to understand what kind of product are brought by these set of clients.So we would call this cluster as " Target"

- **Cluster 4 -** In this cluster, clients earn low income but high spending score.Let's call these clients "Careless"

- **Cluster 5** - In this cluster, clients earn low income and have low spending score.Lets' name them "Sensible"

Hierarchial Clustering:

In[1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will
list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
kaggle/input/mall-customers/Mall_Customers.csv
```

Import Libraries:

In [2]:

```
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering
import matplotlib.pyplot as plt
```

In [3]:

```
#Reading Data
train_path = "/kaggle/input/mall-customers/Mall_Customers.csv"
train_data = pd.read_csv(train_path)
```

In [4]:

```
#Checking the data
train_data.head()
```

Out[4]:

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

In [5]:

```
#Choosing potential features for clustering such as  Annual Income and Spen
ding Score
train_data = train_data.drop(columns=["CustomerID","Genre","Age"],axis=1)
```

In [6]:

```
#Checking the data after dropping columns
train_data.head()
```

Out[6]:

	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 [7]:

```
#Standardazing the dataset
sc = StandardScaler()
train_data = sc.fit_transform(train_data)
train_data
```

Out[7]:

```
array([[-1.73899919, -0.43480148],
       [-1.73899919,  1.19570407],
       [-1.70082976, -1.71591298],
       [-1.70082976,  1.04041783],
       [-1.66266033, -0.39597992],
       [-1.66266033,  1.00159627],
       [-1.62449091, -1.71591298],
       [-1.62449091,  1.70038436],
       [-1.58632148, -1.83237767],
       [-1.58632148,  0.84631002],
       [-1.58632148, -1.4053405 ],
       [-1.58632148,  1.89449216],
       [-1.54815205, -1.36651894],
       [-1.54815205,  1.04041783],
       [-1.54815205, -1.44416206],
       [-1.54815205,  1.11806095],
       [-1.50998262, -0.59008772],
       [-1.50998262,  0.61338066],
       [-1.43364376, -0.82301709],
       [-1.43364376,  1.8556706 ],
       [-1.39547433, -0.59008772],
       [-1.39547433,  0.88513158],
```

```
       [-1.3573049 , -1.75473454],
       [-1.3573049 ,  0.88513158],
       [-1.24279661, -1.4053405 ],
       [-1.24279661,  1.23452563],
       [-1.24279661, -0.7065524 ],
       [-1.24279661,  0.41927286],
       [-1.20462718, -0.74537397],
       [-1.20462718,  1.42863343],
       [-1.16645776, -1.7935561 ],
       [-1.16645776,  0.88513158],
       [-1.05194947, -1.7935561 ],
       [-1.05194947,  1.62274124],
       [-1.05194947, -1.4053405 ],
       [-1.05194947,  1.19570407],
       [-1.01378004, -1.28887582],
       [-1.01378004,  0.88513158],
       [-0.89927175, -0.93948177],
       [-0.89927175,  0.96277471],
       [-0.86110232, -0.59008772],
       [-0.86110232,  1.62274124],
       [-0.82293289, -0.55126616],
       [-0.82293289,  0.41927286],
       [-0.82293289, -0.86183865],
       [-0.82293289,  0.5745591 ],
       [-0.78476346,  0.18634349],
       [-0.78476346, -0.12422899],
       [-0.78476346, -0.3183368 ],
       [-0.78476346, -0.3183368 ],
       [-0.70842461,  0.06987881],
       [-0.70842461,  0.38045129],
       [-0.67025518,  0.14752193],
       [-0.67025518,  0.38045129],
       [-0.67025518, -0.20187212],
       [-0.67025518, -0.35715836],
       [-0.63208575, -0.00776431],
       [-0.63208575, -0.16305055],
       [-0.55574689,  0.03105725],
       [-0.55574689, -0.16305055],
       [-0.55574689,  0.22516505],
       [-0.55574689,  0.18634349],
       [-0.51757746,  0.06987881],
       [-0.51757746,  0.34162973],
       [-0.47940803,  0.03105725],
       [-0.47940803,  0.34162973],
       [-0.47940803, -0.00776431],
       [-0.47940803, -0.08540743],
```

```
       [-0.47940803,  0.34162973],
       [-0.47940803, -0.12422899],
       [-0.4412386 ,  0.18634349],
       [-0.4412386 , -0.3183368 ],
       [-0.40306917, -0.04658587],
       [-0.40306917,  0.22516505],
       [-0.25039146, -0.12422899],
       [-0.25039146,  0.14752193],
       [-0.25039146,  0.10870037],
       [-0.25039146, -0.08540743],
       [-0.25039146,  0.06987881],
       [-0.25039146, -0.3183368 ],
       [-0.25039146,  0.03105725],
       [-0.25039146,  0.18634349],
       [-0.25039146, -0.35715836],
       [-0.25039146, -0.24069368],
       [-0.25039146,  0.26398661],
       [-0.25039146, -0.16305055],
       [-0.13588317,  0.30280817],
       [-0.13588317,  0.18634349],
       [-0.09771374,  0.38045129],
       [-0.09771374, -0.16305055],
       [-0.05954431,  0.18634349],
       [-0.05954431, -0.35715836],
       [-0.02137488, -0.04658587],
       [-0.02137488, -0.39597992],
       [-0.02137488, -0.3183368 ],
       [-0.02137488,  0.06987881],
       [-0.02137488, -0.12422899],
       [-0.02137488, -0.00776431],
       [ 0.01679455, -0.3183368 ],
       [ 0.01679455, -0.04658587],
       [ 0.05496398, -0.35715836],
       [ 0.05496398, -0.08540743],
       [ 0.05496398,  0.34162973],
       [ 0.05496398,  0.18634349],
       [ 0.05496398,  0.22516505],
       [ 0.05496398, -0.3183368 ],
       [ 0.09313341, -0.00776431],
       [ 0.09313341, -0.16305055],
       [ 0.09313341, -0.27951524],
       [ 0.09313341, -0.08540743],
       [ 0.09313341,  0.06987881],
       [ 0.09313341,  0.14752193],
       [ 0.13130284, -0.3183368 ],
       [ 0.13130284, -0.16305055],
```

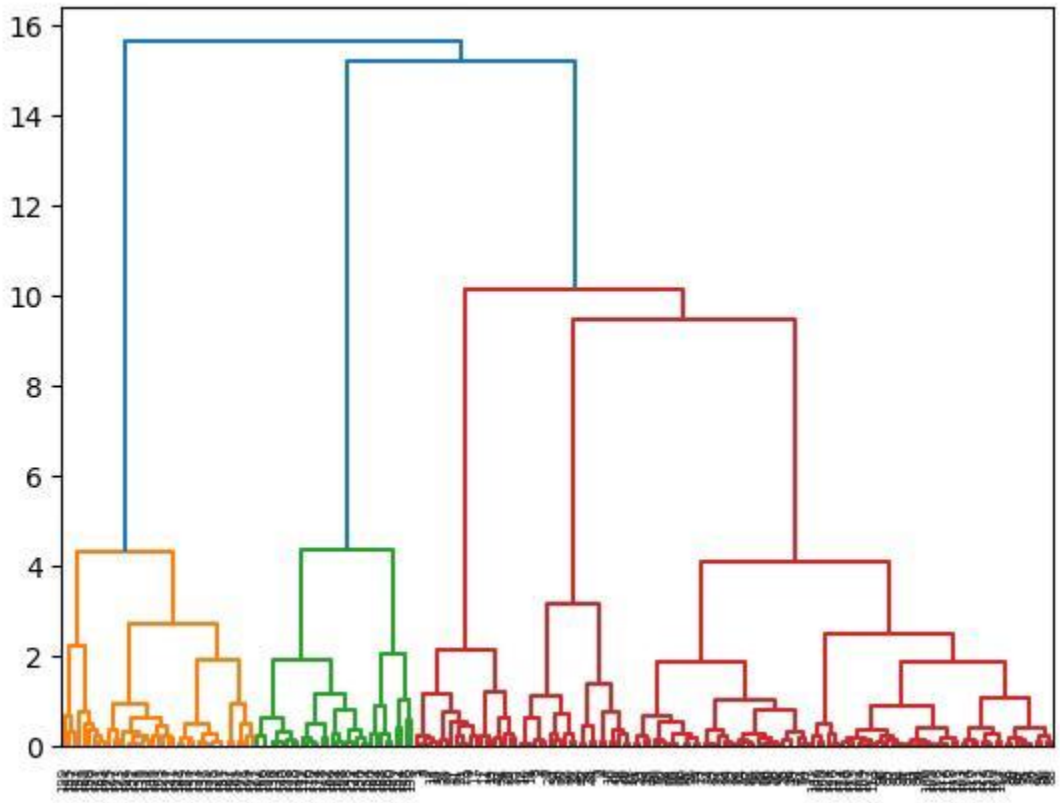
```
       [ 0.16947227, -0.08540743],
       [ 0.16947227, -0.00776431],
       [ 0.16947227, -0.27951524],
       [ 0.16947227,  0.34162973],
       [ 0.24581112, -0.27951524],
       [ 0.24581112,  0.26398661],
       [ 0.24581112,  0.22516505],
       [ 0.24581112, -0.39597992],
       [ 0.32214998,  0.30280817],
       [ 0.32214998,  1.58391968],
       [ 0.36031941, -0.82301709],
       [ 0.36031941,  1.04041783],
       [ 0.39848884, -0.59008772],
       [ 0.39848884,  1.73920592],
       [ 0.39848884, -1.52180518],
       [ 0.39848884,  0.96277471],
       [ 0.39848884, -1.5994483 ],
       [ 0.39848884,  0.96277471],
       [ 0.43665827, -0.62890928],
       [ 0.43665827,  0.80748846],
       [ 0.4748277 , -1.75473454],
       [ 0.4748277 ,  1.46745499],
       [ 0.4748277 , -1.67709142],
       [ 0.4748277 ,  0.88513158],
       [ 0.51299713, -1.56062674],
       [ 0.51299713,  0.84631002],
       [ 0.55116656, -1.75473454],
       [ 0.55116656,  1.6615628 ],
       [ 0.58933599, -0.39597992],
       [ 0.58933599,  1.42863343],
       [ 0.62750542, -1.48298362],
       [ 0.62750542,  1.81684904],
       [ 0.62750542, -0.55126616],
       [ 0.62750542,  0.92395314],
       [ 0.66567484, -1.09476801],
       [ 0.66567484,  1.54509812],
       [ 0.66567484, -1.28887582],
       [ 0.66567484,  1.46745499],
       [ 0.66567484, -1.17241113],
       [ 0.66567484,  1.00159627],
       [ 0.66567484, -1.32769738],
       [ 0.66567484,  1.50627656],
       [ 0.66567484, -1.91002079],
       [ 0.66567484,  1.07923939],
       [ 0.66567484, -1.91002079],
       [ 0.66567484,  0.88513158],
```

```
       [ 0.70384427, -0.59008772],
       [ 0.70384427,  1.27334719],
       [ 0.78018313, -1.75473454],
       [ 0.78018313,  1.6615628 ],
       [ 0.93286085, -0.93948177],
       [ 0.93286085,  0.96277471],
       [ 0.97103028, -1.17241113],
       [ 0.97103028,  1.73920592],
       [ 1.00919971, -0.90066021],
       [ 1.00919971,  0.49691598],
       [ 1.00919971, -1.44416206],
       [ 1.00919971,  0.96277471],
       [ 1.00919971, -1.56062674],
       [ 1.00919971,  1.62274124],
       [ 1.04736914, -1.44416206],
       [ 1.04736914,  1.38981187],
       [ 1.04736914, -1.36651894],
       [ 1.04736914,  0.72984534],
       [ 1.23821628, -1.4053405 ],
       [ 1.23821628,  1.54509812],
       [ 1.390894  , -0.7065524 ],
       [ 1.390894  ,  1.38981187],
       [ 1.42906343, -1.36651894],
       [ 1.42906343,  1.46745499],
       [ 1.46723286, -0.43480148],
       [ 1.46723286,  1.81684904],
       [ 1.54357172, -1.01712489],
       [ 1.54357172,  0.69102378],
       [ 1.61991057, -1.28887582],
       [ 1.61991057,  1.35099031],
       [ 1.61991057, -1.05594645],
       [ 1.61991057,  0.72984534],
       [ 2.00160487, -1.63826986],
       [ 2.00160487,  1.58391968],
       [ 2.26879087, -1.32769738],
       [ 2.26879087,  1.11806095],
       [ 2.49780745, -0.86183865],
       [ 2.49780745,  0.92395314],
       [ 2.91767117, -1.25005425],
       [ 2.91767117,  1.27334719]])
```

In [8]:

```
#Dendogram
linkage_data = linkage(train_data, method='ward', metric='euclidean')
dendrogram(linkage_data)
```

```
plt.show()
```



In [9]:

```
#Hierarchial Clustering Model
hierarchical_cluster = AgglomerativeClustering(n_clusters=2, affinity='euc
lidean', linkage='ward')
labels = hierarchical_cluster.fit_predict(train_data)
```

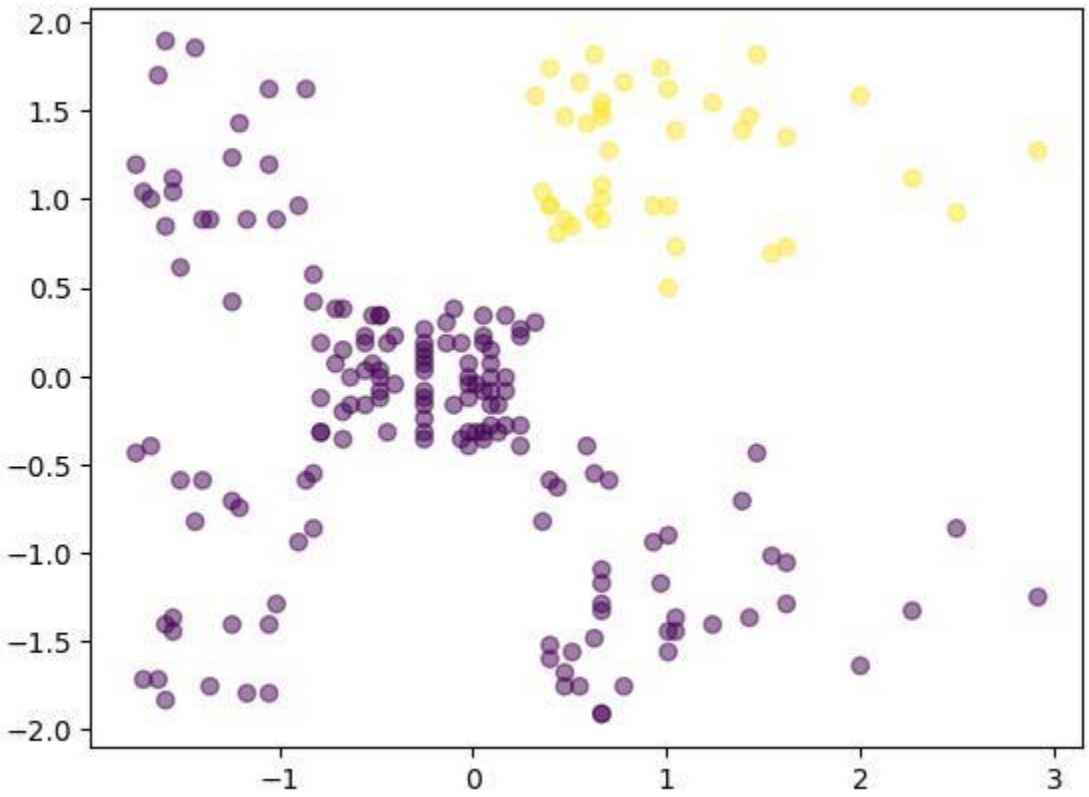
Visualization of the Clustering:

In [10]:

```
#Visualizing the clusters of Hierarchial Clustering
xs = train_data[:,0]
ys = train_data[:,1]
plt.scatter(xs,ys,c=labels,alpha=0.5)
```

Out[10]:

```
<matplotlib.collections.PathCollection at 0x799566dd0990>
```



Customer Segmentation:

- This Python 3 environment comes with many helpful analytics libraries installed
- It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
- For example, here's several helpful packages to load.

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will
list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/mall-customers/Mall_Customers.csv
```

In [2]:

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

In [3]:

```
df=pd.read_csv('/kaggle/input/mall-customers/Mall_Customers.csv')

df.rename(columns={'Genre':'Gender'},inplace=True)
df.head()
```

Out[3]:

	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 [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.isnull().sum()
```

Out[5]:

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

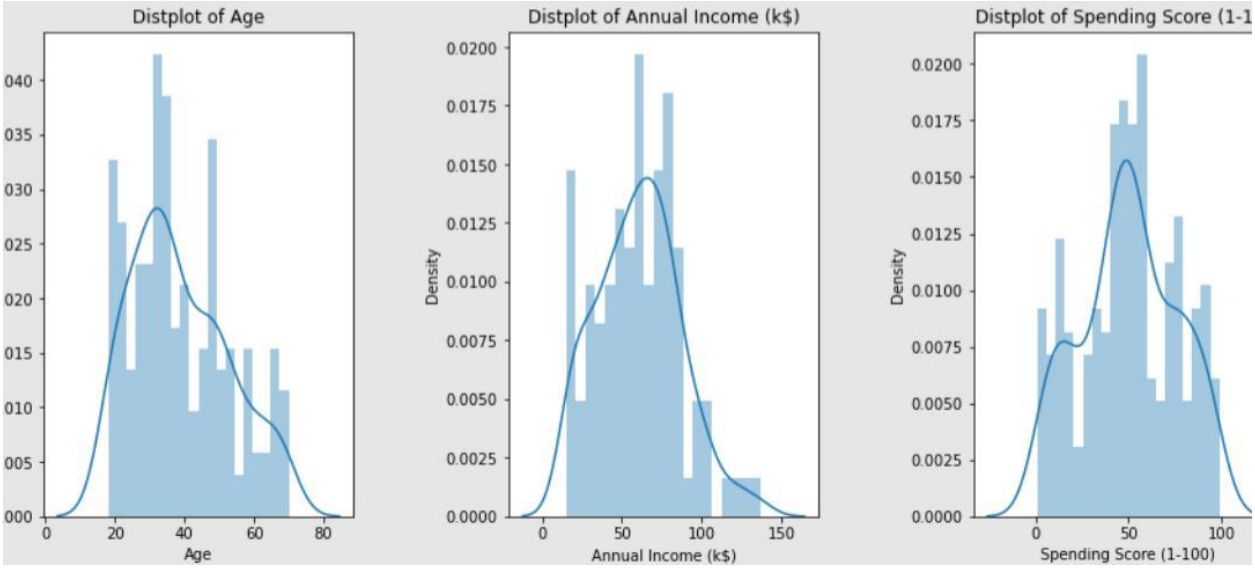
In [6]:

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

In [7]:

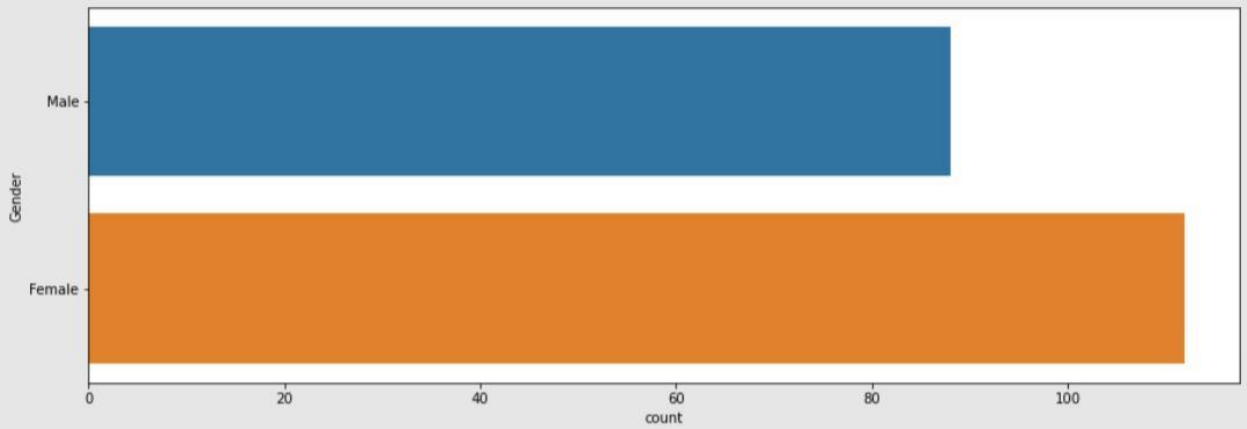
```
plt.figure(1,figsize=(15,6))
n = 0
for x in ['Age','Annual Income (k$)','Spending Score (1-100)']:
    n +=1
    plt.subplot(1,3,n)
    plt.subplots_adjust(hspace=0.5,wspace=0.5)
    sns.distplot(df[x],bins=20)
    plt.title('Distplot of {}'.format(x))
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: Futu
reWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: Futu
reWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: Futu
reWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
```



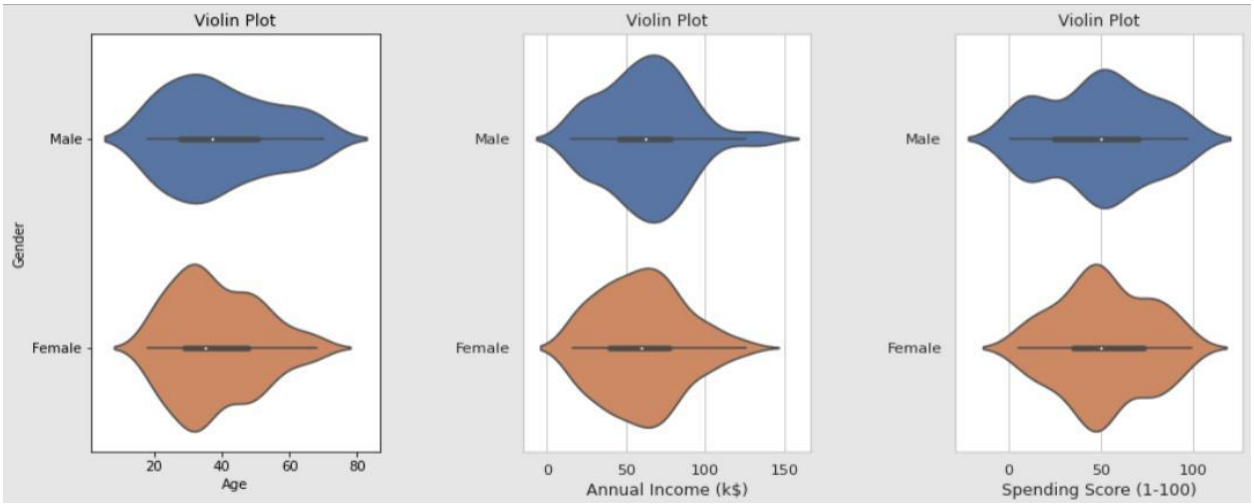
In [8]:

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



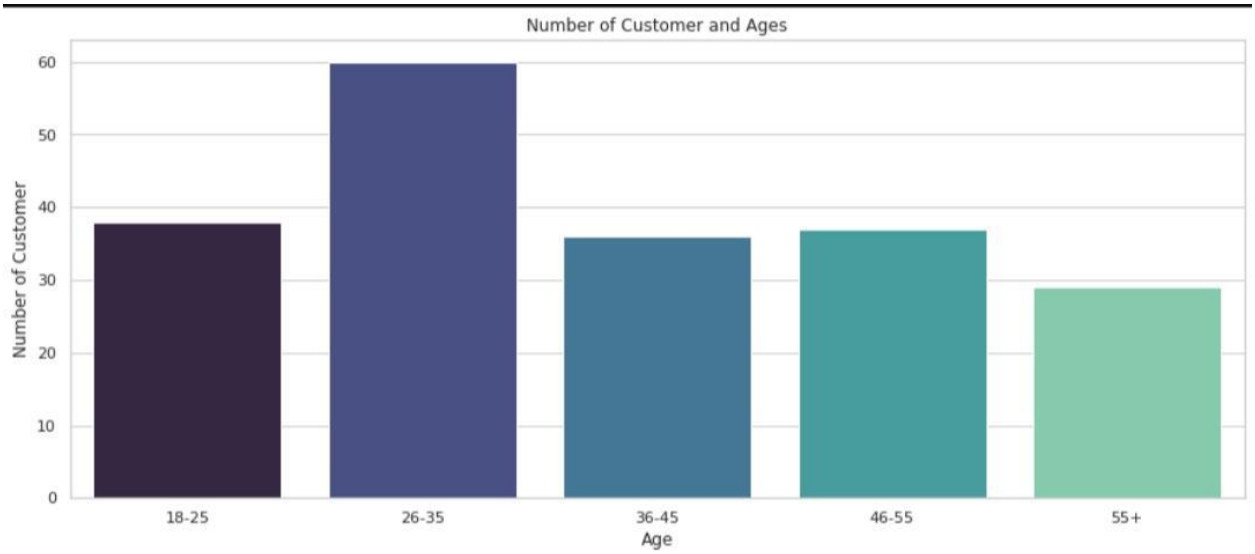
In [9]:

```
plt.figure(1,figsize=(15,6))
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 [10]:

```
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_55_above = df.Age[(df.Age >= 56)]

age_x =["18-25","26-35","36-45","46-55","55+"]
age_y = [len(age_18_25.values),len(age_26_35.values),len(age_36_45),len(ag
e_46_55),len(age_55_above)]

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

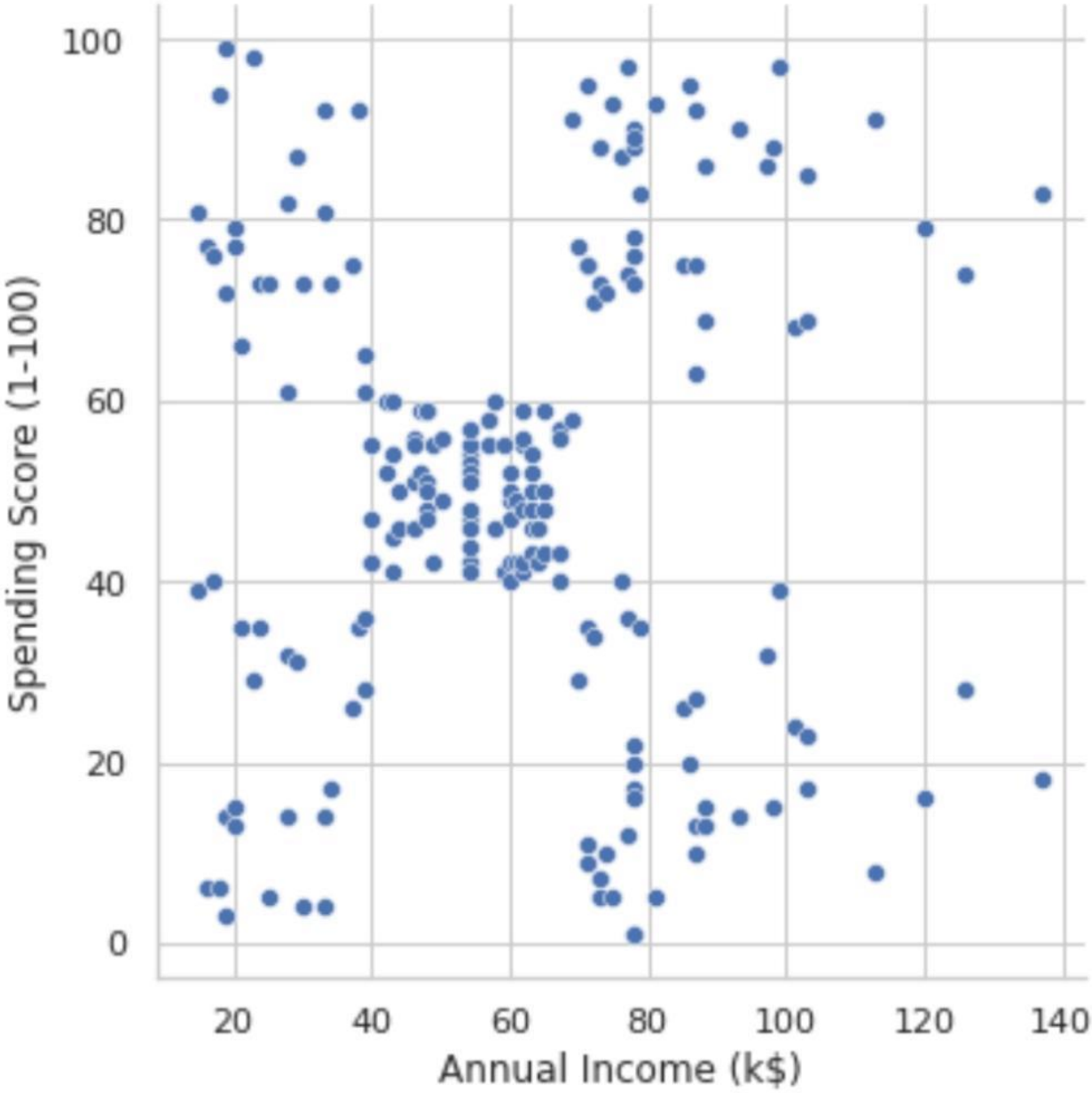


In [11]:

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

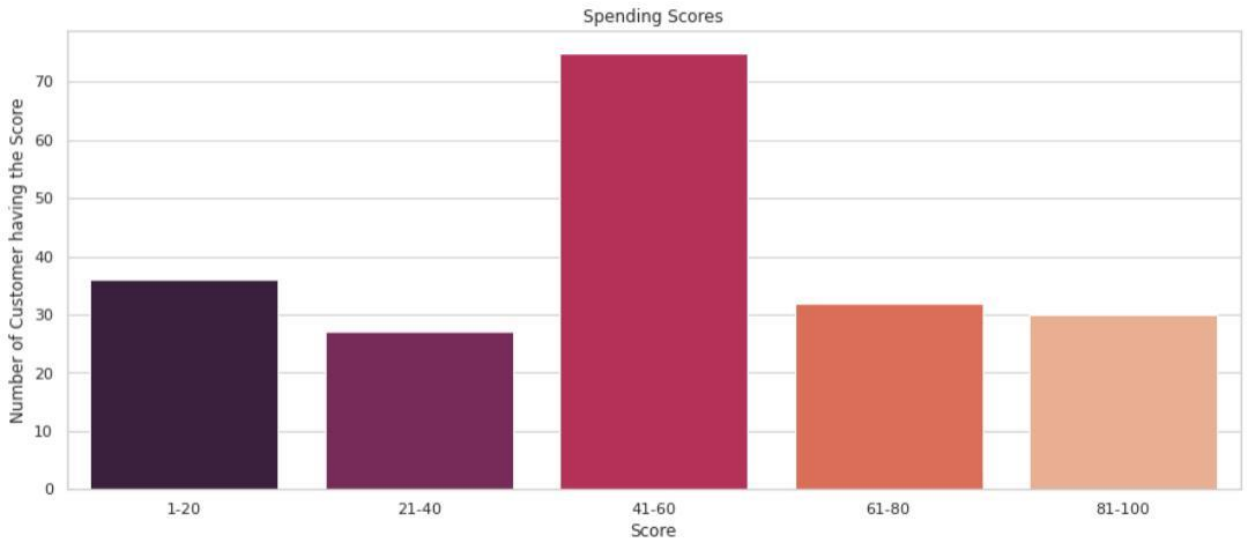
Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x7fcef0536fd0>
```



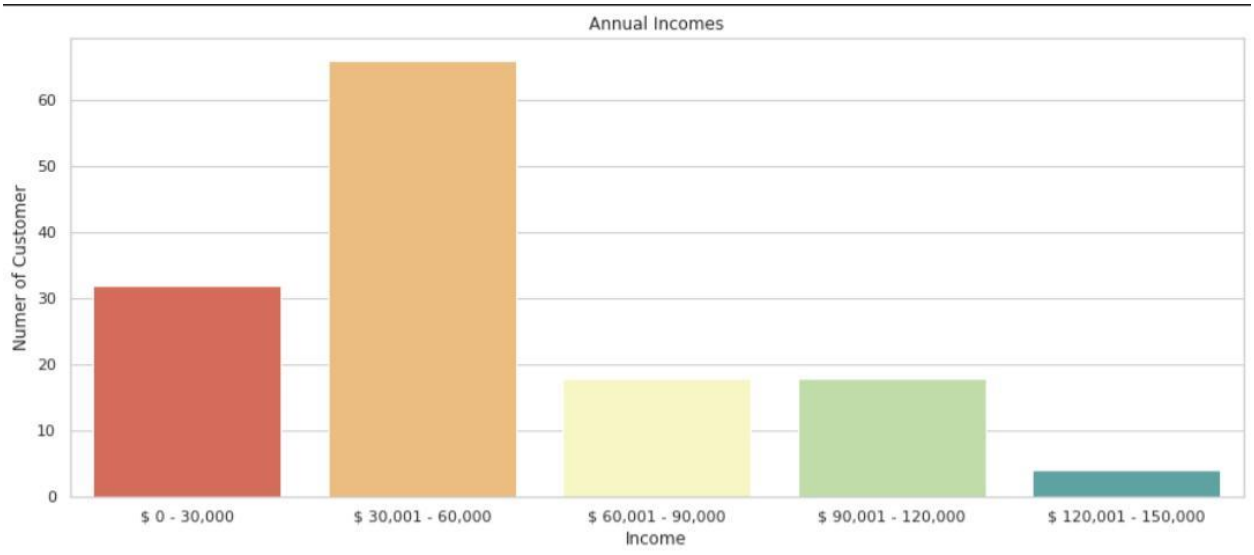
In [12]:

```
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)"] >= 2
1) & (df["Spending Score (1-100)"] <= 40)]
ss_41_60 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 4
1) & (df["Spending Score (1-100)"] <= 60)]
ss_61_80 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >= 6
1) & (df["Spending Score (1-100)"] <= 80)]

ss_81_100 = df["Spending Score (1-100)"][(df["Spending Score (1-100)"] >=
81) & (df["Spending Score (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()
```



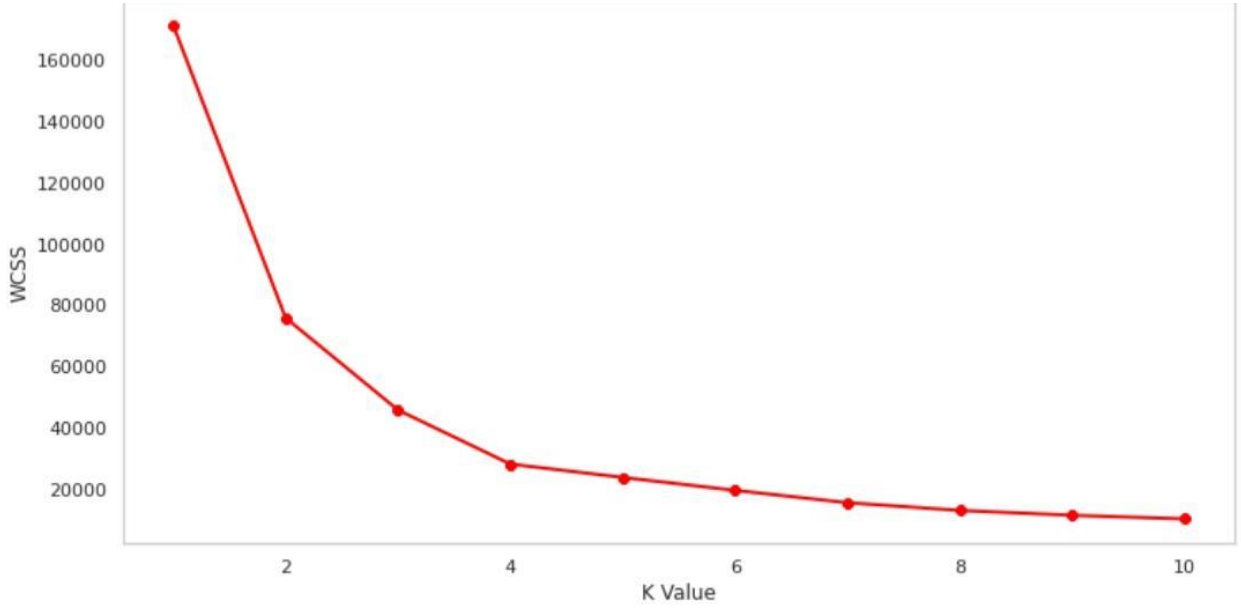
In[13]:

```
ai_0_30 = df["Annual Income (k$)"][(df["Annual Income (k$)"] >= 0) & (df["
Annual Income (k$)"] <= 30)]
ai_31_60= df["Annual Income (k$)"][(df["Annual Income (k$)"] >=31)& (df["A
nnual Income (k$)"] <=60)]
ai_61_90= df["Annual Income (k$)"][(df["Annual Income (k$)"] >=61)& (df["A
nnual Income (k$)"] <=90)]
ai_61_90=df["Annual Income (k$)"][(df["Annual Income (k$)"] >=91)& (df["An
nual Income (k$)"] <=120)]
ai_121_150 = df["Annual Income (k$)"][(df["Annual Income (k$)"]>=121) & (d
f["Annual Income (k$)"] <=150)]

aix = ["$ 0 - 30,000","$ 30,001 - 60,000","$ 60,001 - 90,000","$ 90,001 -
120,000","$ 120,001 - 150,000"]
aiy = [len(ai_0_30.values),len(ai_31_60.values),len(ai_61_90.values),len(a
i_61_90.values),len(ai_121_150.values)]

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



In [14]:

```
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()
```



In [15]:

```
kmeans = KMeans(n_clusters=4)

label = kmeans.fit_predict(X1)

print(label)
```

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

In [16]:

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

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

In [17]:

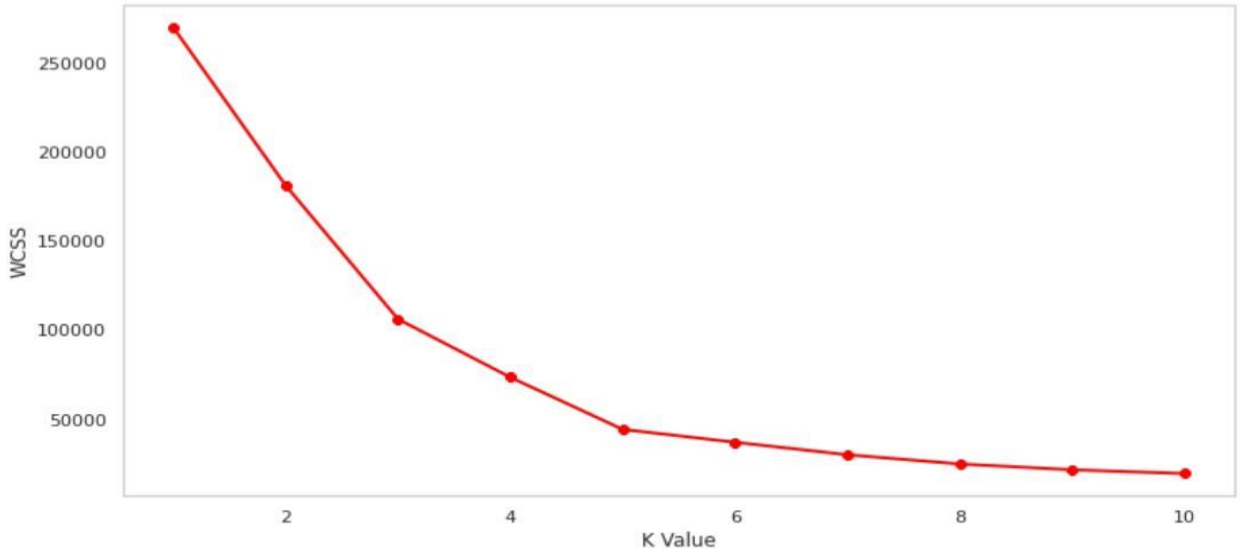
```
plt.scatter(X1[:,0],X1[:,1],c=kmeans.labels_,cmap='rainbow')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],colo
r='black')
plt.title('Clusters of Customers')
plt.xlabel('Age')
plt.ylabel('Spending Score(1-100)')
plt.show
```

Out[17]:

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

In [18]:

```
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()
```



In [19]:

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

In [20]:

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

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

In [21]:

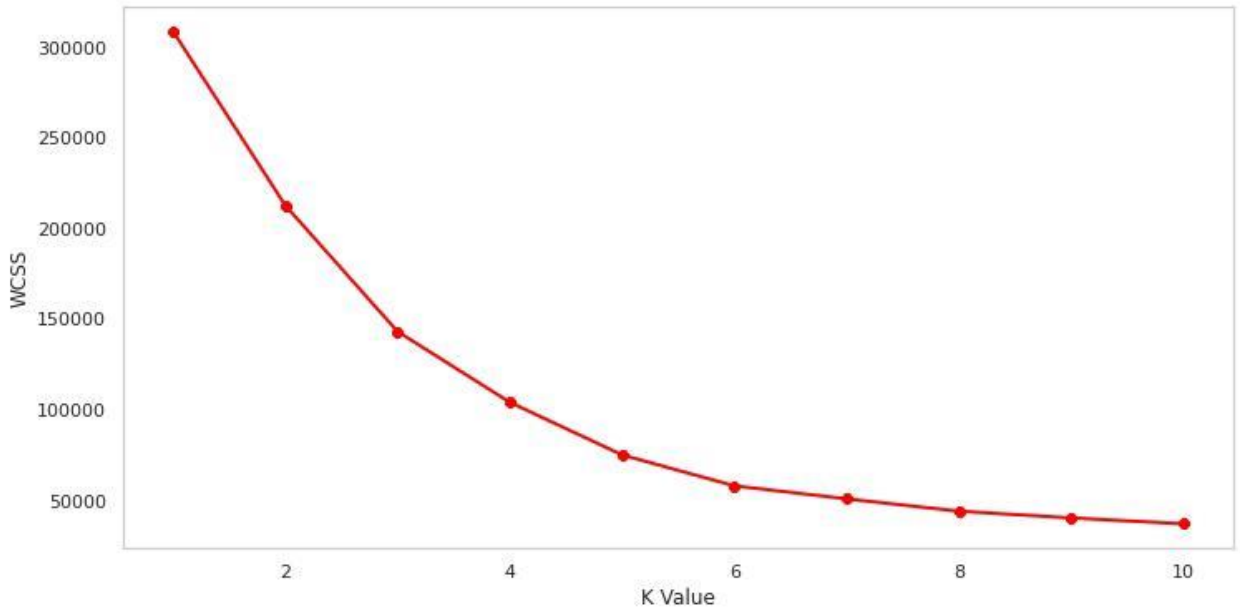
```
plt.scatter(X2[:,0],X1[:,1],c=kmeans.labels_,cmap='rainbow')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],colo
r='black')
plt.title('Clusters of Customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score(1-100)')
plt.show
```

Out[21]:

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

In [22]:

```
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()
```



In [23]:

```
kmeans = KMeans(n_clusters=5)

label = kmeans.fit_predict(X3)

print(label)
```

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

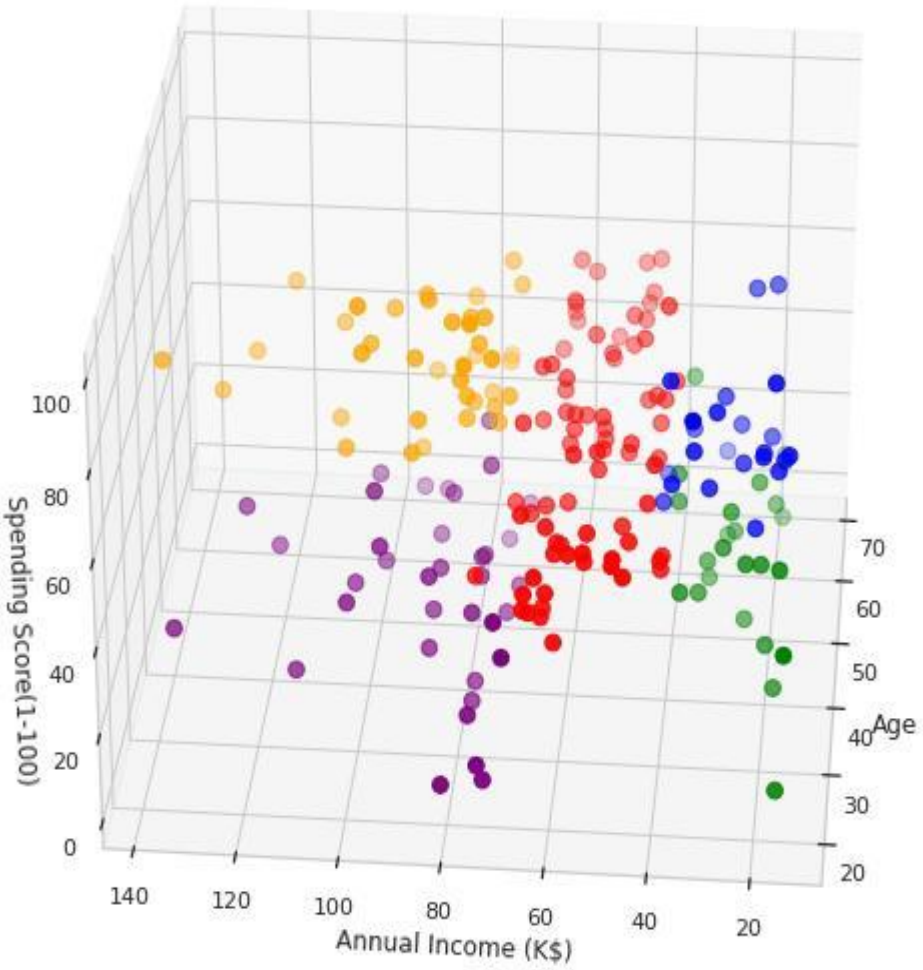
In [24]:

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

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

In [25]:

```
cluster = kmeans.fit_predict(X3)
df["label"] = cluster

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()
```


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

CONCLUSION:

Building a customer segmentation model by leveraging feature enginee ring, applying clustering algorithms, utilizing visualization, and interpret ing the results is a multifaceted process that yields valuable insights for businesses. Here's a conclusion to this approach:

The process of building a customer segmentation model through featur e engineering, clustering algorithms, visualization, and interpretation re presents a powerful strategy for businesses to better understand and e ngage with their customer base. This multifaceted approach not only en hances marketing strategies but also paves the way for improved custo mer experiences, product development, and business decision-making. Let's summarize the key takeaways from this process:

Feature Engineering:

- Feature engineering allows for the creation of meaningful cust omer characteristics that can significantly enhance the quality of s egmentation. By deriving new features or transforming existing on es, businesses gain a more nuanced view of their customers' beha viors and preferences.

Applying Clustering Algorithms:

- Clustering algorithms, such as K-Means, hierarchical clustering, or DBSCAN, enable the grouping of customers into distinct segme nts based on shared attributes. This process identifies hidden patt erns and similarities, which are essential for personalized marketi ng and tailored strategies.

Visualization:

- Visualization techniques, including scatter plots, heatmaps, and dendrograms, bring the customer segmentation results to life. Visual representations help stakeholders grasp the characteristics and relationships among customer segments, making the insights more accessible and actionable.

Interpretation:

- The interpretation of segmented customer data is the bridge between analysis and action. It's crucial to understand what each segment represents, their unique traits, and how these insights can be applied to marketing, product development, and customer service.

Incorporating these steps into the customer segmentation process not only enhances the precision of targeting specific customer groups but also empowers businesses to adapt to changing customer behaviors and preferences. This approach fosters continuous improvement and innovation, ensuring that customer segmentation remains a dynamic and valuable tool for any data-driven organization.

By understanding your customers on a granular level through feature engineering, clustering, visualization, and interpretation, businesses can forge more meaningful connections, deliver personalized experiences, and ultimately drive success in a competitive marketplace.