

ASSIGNMENT-4

Assignment date	
Student name	NALLAPANENI DEEPAK
Student roll number	111519104084
Maximum marks	2m

1. Download the dataset: Dataset

ANS: Data set has been downloaded.

2. Load the dataset into the tool.

ANS: `import pandas as pd`
`import numpy as np`

```
df=pd.read_csv('Mall_Customers.csv')  
df.head()
```

output:

```
In [8]: import pandas as pd  
import numpy as np
```

```
In [9]: df=pd.read_csv('Mall_Customers.csv')  
df.head()
```

```
Out[9]:
```

	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

3. Perform Below Visualizations.

- Univariate Analysis
- Bi- Variate Analysis
- Multi-Variate Analysis

ANS:

.Univariate Analysis

In [10]:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

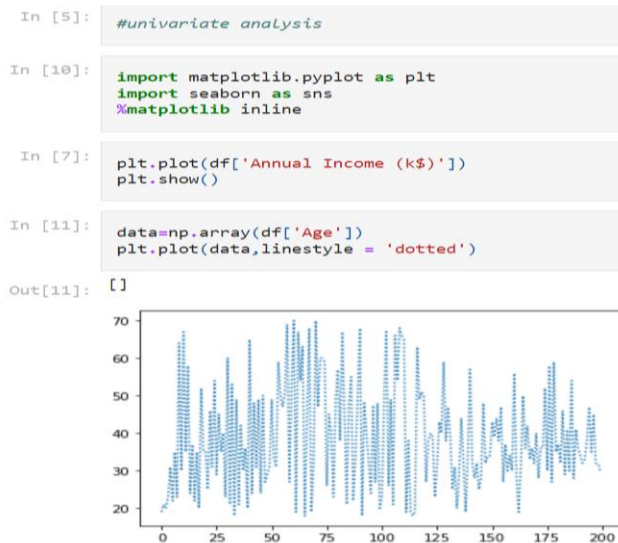
In [7]:

```
plt.plot(df['Annual Income (k$)'])
plt.show()
```

In [11]:

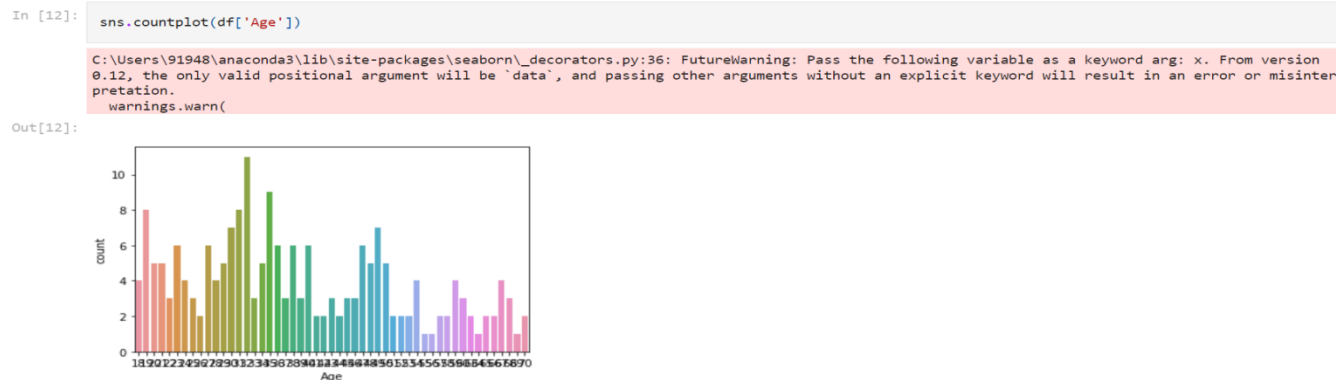
```
data=np.array(df['Age'])
plt.plot(data,linestyle= 'dotted')
```

Output:



```
sns.countplot(df['Age'])
```

Output:

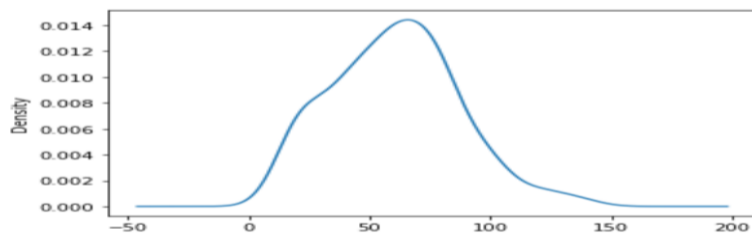


```
df['Annual Income (k$)'].plot(kind='density')
```

Output:

```
In [13]: df['Annual Income (k$)'].plot(kind='density')
```

Out[13]:



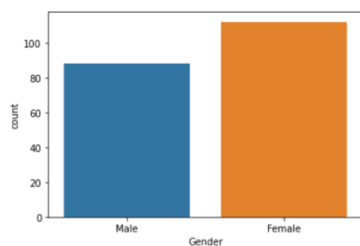
```
sns.countplot(df['Gender'])
```

Output:

```
In [14]: sns.countplot(df['Gender'])
```

C:\Users\91948\anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
warnings.warn(

Out[14]:



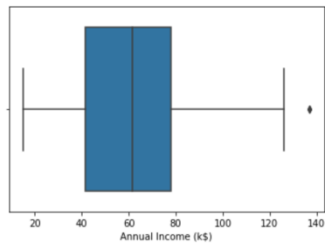
```
sns.boxplot(df['Annual Income (k$)'])
```

Output:

```
In [15]: sns.boxplot(df['Annual Income (k$)'])
```

```
C:\Users\91948\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be 'data', and passing other arguments without an explicit keyword will result in an error or misinterpretation.
warnings.warn(
```

```
Out[15]:
```

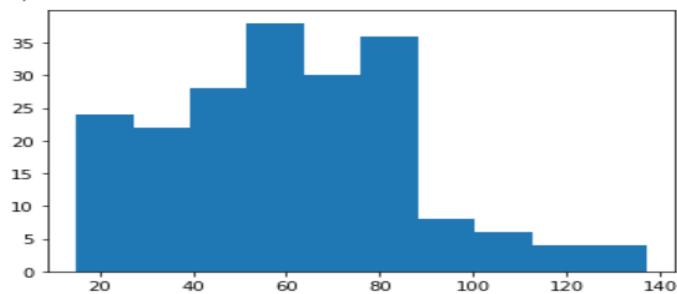


```
plt.hist(df['Annual Income (k$)'])
```

Output:

```
In [16]: plt.hist(df['Annual Income (k$)'])
```

```
Out[16]: (array([24., 22., 28., 38., 30., 36., 8., 6., 4., 4.]),
array([ 15., 27.2, 39.4, 51.6, 63.8, 76., 88.2, 100.4, 112.6,
124.8, 137. ]),
)
```



Bi-variate Analysis:

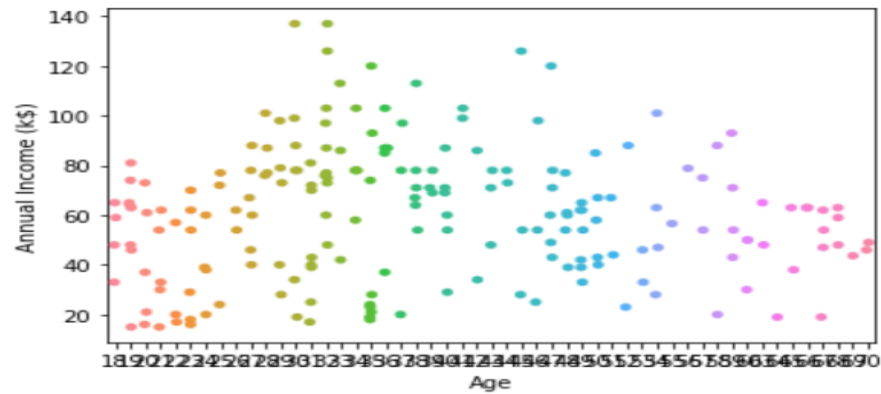
```
sns.stripplot(x=df['Age'],y=df['Annual Income (k$)'])
```

Output:

```
In [17]: #Bivariate Analysis
```

```
In [18]: sns.stripplot(x=df['Age'],y=df['Annual Income (k$)'])
```

Out[18]:

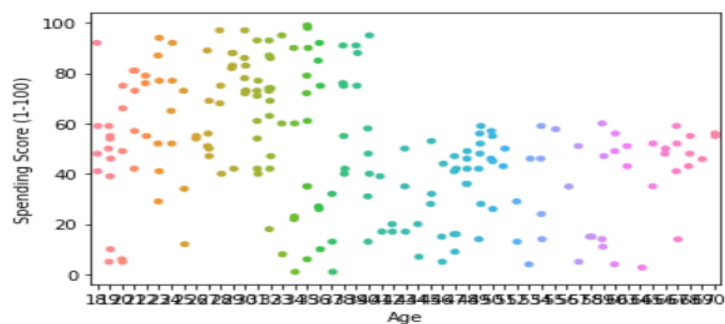


```
sns.stripplot(x=df['Age'],y=df['Spending Score (1-100)'])
```

Output:

```
In [19]: sns.stripplot(x=df['Age'],y=df['Spending Score (1-100)'])
```

Out[19]:

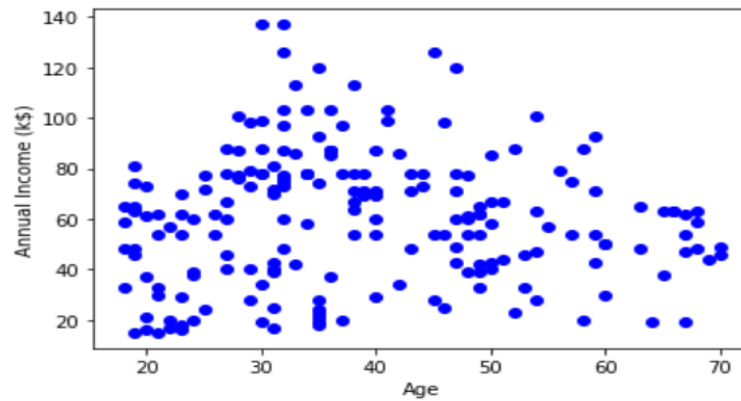


```
plt.scatter(df['Age'],df['Annual Income (k$)'],color='blue')  
plt.xlabel("Age")  
plt.ylabel("Annual Income (k$)")
```

Output:

```
In [20]: plt.scatter(df['Age'],df['Annual Income (k$)'],color='blue')
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
```

```
Out[20]: Text(0, 0.5, 'Annual Income (k$)')
```

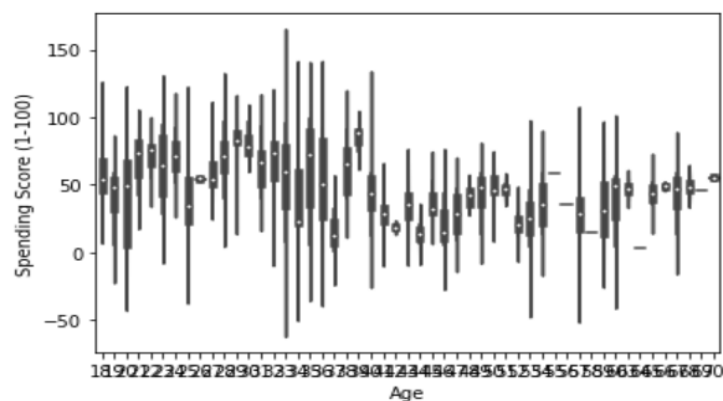


```
sns.violinplot(x='Age',y='Spending Score (1-100)',data=df)
```

Output:

```
In [21]: sns.violinplot(x = 'Age', y = 'Spending Score (1-100)', data = df)
```

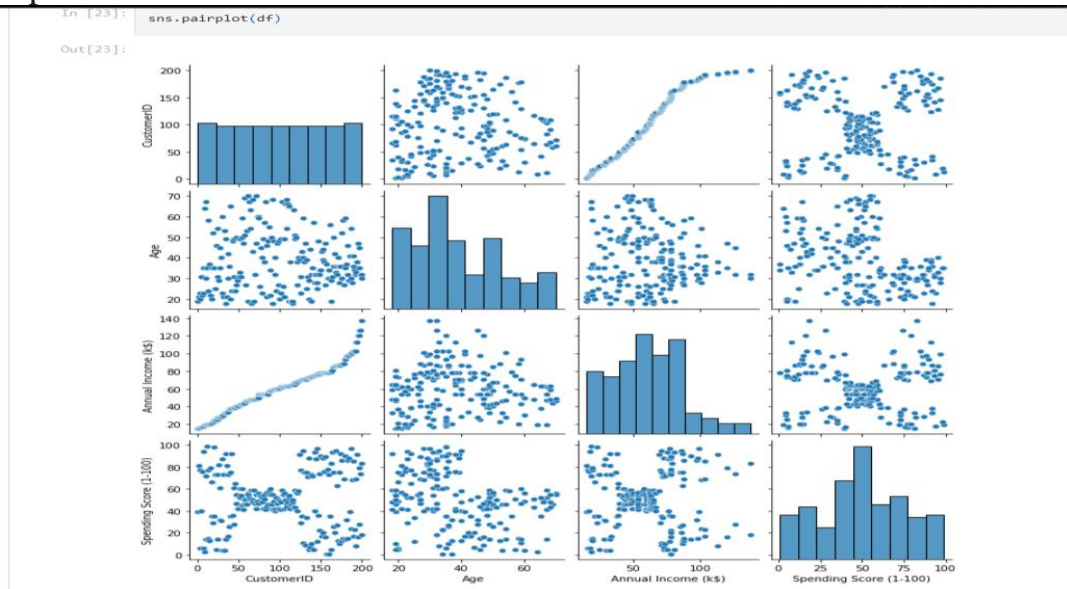
```
Out[21]:
```



Multi-variate Analysis:

```
sns.pairplot(df)
```

Output:



3. Perform descriptive statistics on the dataset.

ANS:

```
sns.heatmap(df.corr(),annot=True)
```

Output:

```
In [24]: #Descriptive Statistics
```

```
In [25]: sns.heatmap(df.corr(),annot=True)
```

Out[25]:



```
df.shape
```

Output:

```
(200, 5)
```

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

Output:

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

```
df.info()
```

Output:

```
RangeIndex: 200 entries, 0 to 199
```

```
Data columns (total 5 columns):
```

#	Column	Non-Null Count	Dtype
0	CustomerID	200 non-null	int64
1	Gender	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 [26]: df.shape
Out[26]: (200, 5)

In [31]: df.isnull().sum()
Out[31]: CustomerID      0
Gender          0
Age            0
Annual Income (k$)  0
Spending Score (1-100)  0
dtype: int64

In [32]: df.info()
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
#   Column              Non-Null Count  Dtype
---  ---
0   CustomerID          200 non-null   int64
1   Gender              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
```

```
df.describe()
```

Output:


```
In [33]: df.describe()
```

```
Out[33]:
```

	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

```
df.mean()
```

Output:

```
CustomerID      100.50
Age              38.85
Annual Income (k$)  60.56
Spending Score (1-100)  50.20
dtype: float64
```

```
df.median()
```

Output:

```
CustomerID      100.5
Age              36.0
Annual Income (k$)  61.5
Spending Score (1-100)  50.0
dtype: float64
```

```
In [34]: df.mean()
```

C:\Users\91948\AppData\Local\Temp\ipykernel_6316\3698961737.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.

```
Out[34]: CustomerID      100.50
Age              38.85
Annual Income (k$)  60.56
Spending Score (1-100)  50.20
dtype: float64
```

```
In [35]: df.median()
```

C:\Users\91948\AppData\Local\Temp\ipykernel_6316\530051474.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError. Select only valid columns before calling the reduction.

```
Out[35]: CustomerID      100.5
Age              36.0
Annual Income (k$)  61.5
Spending Score (1-100)  50.0
dtype: float64
```

```
df.mode()
```

Output:

```
In [36]: df.mode()
```

```
Out[36]:
```

	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Female	32.0	54.0	42.0
1	2	NaN	NaN	78.0	NaN
2	3	NaN	NaN	NaN	NaN
3	4	NaN	NaN	NaN	NaN
4	5	NaN	NaN	NaN	NaN
...
195	196	NaN	NaN	NaN	NaN
196	197	NaN	NaN	NaN	NaN
197	198	NaN	NaN	NaN	NaN
198	199	NaN	NaN	NaN	NaN
199	200	NaN	NaN	NaN	NaN

200 rows × 5 columns

```
df['Gender'].value_counts()
```

Output:

```
Female    112
Male       88
Name: Gender, dtype: int64
```

```
In [37]: df['Gender'].value_counts()
```

```
Out[37]: Female    112
Male       88
Name: Gender, dtype: int64
```

5. Check for Missing values and deal with them.

ANS:

```
df.isna().sum()
```

Output:

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

```
In [38]: #check fo Missing values
```

```
In [39]: df.isna().sum()
```

```
Out[39]: CustomerID      0
Gender      0
Age         0
Annual Income (k$)      0
Spending Score (1-100)  0
dtype: int64
```

6. Find the outliers and replace them outliers

ANS:

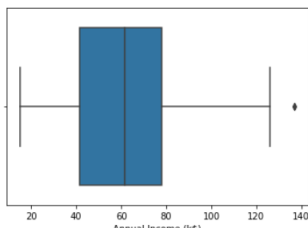
```
sns.boxplot(df['Annual Income (k$)'])
```

Output:

```
In [43]: sns.boxplot(df['Annual Income (k$)'])
```

C:\Users\91948\anaconda3\lib\site-packages\seaborn\decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be 'data', and passing other arguments without an explicit keyword will result in an error or misinterpretation.
warnings.warn(

```
Out[43]:
```



```
Q1 =df['Annual Income (k$)'].quantile(0.25)
```

```
Q3 =df['Annual Income (k$)'].quantile(0.75)
```

```
IQR = Q3 - Q1
```

```
whisker_width= 1.5
```

```
lower_whisker= Q1 -(whisker_width*IQR)
```

```
upper_whisker= Q3 +(whisker_width*IQR)
```

```
df['Annual Income (k$)']=np.where(df['Annual Income (k$)']>upper_whisker,upper_whisker,np.where(df['Annual Income (k$)']<lower_whisker,lower_whisker,df['Annual Income (k$)']))
```

```
sns.boxplot(df['Annual Income (k$)'])
```

Output:

```

In [44]: Q1 = df['Annual Income (k$)'].quantile(0.25)
          Q3 = df['Annual Income (k$)'].quantile(0.75)
          IQR = Q3 - Q1
          whisker_width = 1.5
          lower_whisker = Q1 - (whisker_width*IQR)
          upper_whisker = Q3 + (whisker_width*IQR)
          df['Annual Income (k$)'] = np.where(df['Annual Income (k$)'] > upper_whisker, upper_whisker, np.where(df['Annual Income (k$)'] < lower_whisker, lower_whisker, df['Annual Income (k$)']))

In [45]: sns.boxplot(df['Annual Income (k$)'])

C:\Users\91948\anaconda3\lib\site-packages\seaborn\decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be 'data', and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(

Out[45]:

```



7. Check for Categorical columns and perform encoding.

ANS:

```

numeric_data=df.select_dtypes(include=[np.number])
categorical_data=df.select_dtypes(exclude=[np.number])
print("Number of numerical variables: ", numeric_data.shape[1])
print("Number of categorical variables: ", categorical_data.shape[1])

Number of numerical variables:  4
Number of categorical variables:  1

```

```

print("Number of categorical variables: ", categorical_data.shape[1])
Categorical_variables=list(categorical_data.columns)
Categorical_variables

```

Output:

Number of categorical variables: 1

```
['Gender']
```

```
df['Gender'].value_counts()
```

Output:

```

Female    112
Male       88
Name: Gender, dtype: int64

```

```

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
label=le.fit_transform(df['Gender'])
df["Gender"] = label

```

```
df['Gender'].value_counts()
```

Output:

```

0    112
1     88

```

Name: Gender, dtype: int64

```
In [46]: #Encoding Categorical Values

In [47]: numeric_data = df.select_dtypes(include=[np.number])
categorical_data = df.select_dtypes(exclude=[np.number])
print("Number of numerical variables: ", numeric_data.shape[1])
print("Number of categorical variables: ", categorical_data.shape[1])

Number of numerical variables: 4
Number of categorical variables: 1

In [48]: print("Number of categorical variables: ", categorical_data.shape[1])
Categorical_variables = list(categorical_data.columns)
Categorical_variables

Number of categorical variables: 1
Out[48]: ['Gender']

In [49]: df['Gender'].value_counts()

Out[49]: Female    112
Male         88
Name: Gender, dtype: int64

In [50]: from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
label = le.fit_transform(df['Gender'])
df['Gender'] = label

In [51]: df['Gender'].value_counts()

Out[51]: 0    112
1     88
Name: Gender, dtype: int64
```

8. Scaling the data

ANS:

```
X =df.drop("Age",axis=1)
Y =df['Age']
```

In [55]:

```
fromsklearn.preprocessingimportStandardScaler
object=StandardScaler()
scale=object.fit_transform(X)
print(scale)
```

Output:

```
[[-1.7234121  1.12815215 -1.74542941 -0.43480148]
 [-1.70609137  1.12815215 -1.74542941  1.19570407]
 [-1.68877065 -0.88640526 -1.70708307 -1.71591298]
 [-1.67144992 -0.88640526 -1.70708307  1.04041783]
 [-1.6541292  -0.88640526 -1.66873673 -0.39597992]
 [-1.63680847 -0.88640526 -1.66873673  1.00159627]
 [-1.61948775 -0.88640526 -1.6303904  -1.71591298]
 [-1.60216702 -0.88640526 -1.6303904  1.70038436]
 [-1.5848463   1.12815215 -1.59204406 -1.83237767]
 [-1.56752558 -0.88640526 -1.59204406  0.84631002]
 [-1.55020485  1.12815215 -1.59204406 -1.4053405 ]
 [-1.53288413 -0.88640526 -1.59204406  1.89449216]
 [-1.5155634  -0.88640526 -1.55369772 -1.36651894]
 [-1.49824268 -0.88640526 -1.55369772  1.04041783]
 [-1.48092195  1.12815215 -1.55369772 -1.44416206]
 [-1.46360123  1.12815215 -1.55369772  1.11806095]]
```

[-1.4462805 -0.88640526 -1.51535138 -0.59008772]
[-1.42895978 1.12815215 -1.51535138 0.61338066]
[-1.41163905 1.12815215 -1.43865871 -0.82301709]
[-1.39431833 -0.88640526 -1.43865871 1.8556706]
[-1.3769976 1.12815215 -1.40031237 -0.59008772]
[-1.35967688 1.12815215 -1.40031237 0.88513158]
[-1.34235616 -0.88640526 -1.36196603 -1.75473454]
[-1.32503543 1.12815215 -1.36196603 0.88513158]
[-1.30771471 -0.88640526 -1.24692702 -1.4053405]
[-1.29039398 1.12815215 -1.24692702 1.23452563]
[-1.27307326 -0.88640526 -1.24692702 -0.7065524]
[-1.25575253 1.12815215 -1.24692702 0.41927286]
[-1.23843181 -0.88640526 -1.20858069 -0.74537397]
[-1.22111108 -0.88640526 -1.20858069 1.42863343]
[-1.20379036 1.12815215 -1.17023435 -1.7935561]
[-1.18646963 -0.88640526 -1.17023435 0.88513158]
[-1.16914891 1.12815215 -1.05519534 -1.7935561]
[-1.15182818 1.12815215 -1.05519534 1.62274124]
[-1.13450746 -0.88640526 -1.05519534 -1.4053405]
[-1.11718674 -0.88640526 -1.05519534 1.19570407]
[-1.09986601 -0.88640526 -1.016849 -1.28887582]
[-1.08254529 -0.88640526 -1.016849 0.88513158]
[-1.06522456 -0.88640526 -0.90180999 -0.93948177]
[-1.04790384 -0.88640526 -0.90180999 0.96277471]
[-1.03058311 -0.88640526 -0.86346365 -0.59008772]
[-1.01326239 1.12815215 -0.86346365 1.62274124]
[-0.99594166 1.12815215 -0.82511731 -0.55126616]
[-0.97862094 -0.88640526 -0.82511731 0.41927286]
[-0.96130021 -0.88640526 -0.82511731 -0.86183865]
[-0.94397949 -0.88640526 -0.82511731 0.5745591]
[-0.92665877 -0.88640526 -0.78677098 0.18634349]
[-0.90933804 -0.88640526 -0.78677098 -0.12422899]
[-0.89201732 -0.88640526 -0.78677098 -0.3183368]
[-0.87469659 -0.88640526 -0.78677098 -0.3183368]
[-0.85737587 -0.88640526 -0.7100783 0.06987881]
[-0.84005514 1.12815215 -0.7100783 0.38045129]
[-0.82273442 -0.88640526 -0.67173196 0.14752193]
[-0.80541369 1.12815215 -0.67173196 0.38045129]
[-0.78809297 -0.88640526 -0.67173196 -0.20187212]
[-0.77077224 1.12815215 -0.67173196 -0.35715836]
[-0.75345152 -0.88640526 -0.63338563 -0.00776431]
[-0.73613079 1.12815215 -0.63338563 -0.16305055]
[-0.71881007 -0.88640526 -0.55669295 0.03105725]
[-0.70148935 1.12815215 -0.55669295 -0.16305055]
[-0.68416862 1.12815215 -0.55669295 0.22516505]
[-0.6668479 1.12815215 -0.55669295 0.18634349]
[-0.64952717 -0.88640526 -0.51834661 0.06987881]
[-0.63220645 -0.88640526 -0.51834661 0.34162973]
[-0.61488572 1.12815215 -0.48000028 0.03105725]
[-0.597565 1.12815215 -0.48000028 0.34162973]
[-0.58024427 -0.88640526 -0.48000028 -0.00776431]
[-0.56292355 -0.88640526 -0.48000028 -0.08540743]
[-0.54560282 1.12815215 -0.48000028 0.34162973]
[-0.5282821 -0.88640526 -0.48000028 -0.12422899]
[-0.51096138 1.12815215 -0.44165394 0.18634349]
[-0.49364065 -0.88640526 -0.44165394 -0.3183368]
[-0.47631993 -0.88640526 -0.4033076 -0.04658587]

[-0.4589992 -0.88640526 -0.4033076 0.22516505]
[-0.44167848 1.12815215 -0.24992225 -0.12422899]
[-0.42435775 1.12815215 -0.24992225 0.14752193]
[-0.40703703 -0.88640526 -0.24992225 0.10870037]
[-0.3897163 1.12815215 -0.24992225 -0.08540743]
[-0.37239558 -0.88640526 -0.24992225 0.06987881]
[-0.35507485 -0.88640526 -0.24992225 -0.3183368]
[-0.33775413 1.12815215 -0.24992225 0.03105725]
[-0.3204334 1.12815215 -0.24992225 0.18634349]
[-0.30311268 1.12815215 -0.24992225 -0.35715836]
[-0.28579196 -0.88640526 -0.24992225 -0.24069368]
[-0.26847123 -0.88640526 -0.24992225 0.26398661]
[-0.25115051 1.12815215 -0.24992225 -0.16305055]
[-0.23382978 -0.88640526 -0.13488324 0.30280817]
[-0.21650906 -0.88640526 -0.13488324 0.18634349]
[-0.19918833 -0.88640526 -0.0965369 0.38045129]
[-0.18186761 -0.88640526 -0.0965369 -0.16305055]
[-0.16454688 -0.88640526 -0.05819057 0.18634349]
[-0.14722616 1.12815215 -0.05819057 -0.35715836]
[-0.12990543 1.12815215 -0.01984423 -0.04658587]
[-0.11258471 -0.88640526 -0.01984423 -0.39597992]
[-0.09526399 -0.88640526 -0.01984423 -0.3183368]
[-0.07794326 1.12815215 -0.01984423 0.06987881]
[-0.06062254 -0.88640526 -0.01984423 -0.12422899]
[-0.04330181 -0.88640526 -0.01984423 -0.00776431]
[-0.02598109 1.12815215 0.01850211 -0.3183368]
[-0.00866036 1.12815215 0.01850211 -0.04658587]
[0.00866036 -0.88640526 0.05684845 -0.35715836]
[0.02598109 -0.88640526 0.05684845 -0.08540743]
[0.04330181 1.12815215 0.05684845 0.34162973]
[0.06062254 1.12815215 0.05684845 0.18634349]
[0.07794326 1.12815215 0.05684845 0.22516505]
[0.09526399 -0.88640526 0.05684845 -0.3183368]
[0.11258471 -0.88640526 0.09519478 -0.00776431]
[0.12990543 1.12815215 0.09519478 -0.16305055]
[0.14722616 1.12815215 0.09519478 -0.27951524]
[0.16454688 1.12815215 0.09519478 -0.08540743]
[0.18186761 1.12815215 0.09519478 0.06987881]
[0.19918833 -0.88640526 0.09519478 0.14752193]
[0.21650906 -0.88640526 0.13354112 -0.3183368]
[0.23382978 1.12815215 0.13354112 -0.16305055]
[0.25115051 -0.88640526 0.17188746 -0.08540743]
[0.26847123 -0.88640526 0.17188746 -0.00776431]
[0.28579196 -0.88640526 0.17188746 -0.27951524]
[0.30311268 -0.88640526 0.17188746 0.34162973]
[0.3204334 -0.88640526 0.24858013 -0.27951524]
[0.33775413 -0.88640526 0.24858013 0.26398661]
[0.35507485 1.12815215 0.24858013 0.22516505]
[0.37239558 -0.88640526 0.24858013 -0.39597992]
[0.3897163 -0.88640526 0.32527281 0.30280817]
[0.40703703 1.12815215 0.32527281 1.58391968]
[0.42435775 -0.88640526 0.36361914 -0.82301709]
[0.44167848 -0.88640526 0.36361914 1.04041783]
[0.4589992 1.12815215 0.40196548 -0.59008772]
[0.47631993 1.12815215 0.40196548 1.73920592]
[0.49364065 1.12815215 0.40196548 -1.52180518]
[0.51096138 1.12815215 0.40196548 0.96277471]

[0.5282821	1.12815215	0.40196548	-1.5994483]
[0.54560282	1.12815215	0.40196548	0.96277471]
[0.56292355	-0.88640526	0.44031182	-0.62890928]
[0.58024427	-0.88640526	0.44031182	0.80748846]
[0.597565	1.12815215	0.47865816	-1.75473454]
[0.61488572	-0.88640526	0.47865816	1.46745499]
[0.63220645	-0.88640526	0.47865816	-1.67709142]
[0.64952717	1.12815215	0.47865816	0.88513158]
[0.6668479	1.12815215	0.51700449	-1.56062674]
[0.68416862	-0.88640526	0.51700449	0.84631002]
[0.70148935	-0.88640526	0.55535083	-1.75473454]
[0.71881007	1.12815215	0.55535083	1.6615628]
[0.73613079	-0.88640526	0.59369717	-0.39597992]
[0.75345152	-0.88640526	0.59369717	1.42863343]
[0.77077224	1.12815215	0.6320435	-1.48298362]
[0.78809297	1.12815215	0.6320435	1.81684904]
[0.80541369	1.12815215	0.6320435	-0.55126616]
[0.82273442	-0.88640526	0.6320435	0.92395314]
[0.84005514	-0.88640526	0.67038984	-1.09476801]
[0.85737587	1.12815215	0.67038984	1.54509812]
[0.87469659	1.12815215	0.67038984	-1.28887582]
[0.89201732	1.12815215	0.67038984	1.46745499]
[0.90933804	-0.88640526	0.67038984	-1.17241113]
[0.92665877	-0.88640526	0.67038984	1.00159627]
[0.94397949	-0.88640526	0.67038984	-1.32769738]
[0.96130021	-0.88640526	0.67038984	1.50627656]
[0.97862094	1.12815215	0.67038984	-1.91002079]
[0.99594166	-0.88640526	0.67038984	1.07923939]
[1.01326239	1.12815215	0.67038984	-1.91002079]
[1.03058311	-0.88640526	0.67038984	0.88513158]
[1.04790384	-0.88640526	0.70873618	-0.59008772]
[1.06522456	-0.88640526	0.70873618	1.27334719]
[1.08254529	1.12815215	0.78542885	-1.75473454]
[1.09986601	-0.88640526	0.78542885	1.6615628]
[1.11718674	1.12815215	0.9388142	-0.93948177]
[1.13450746	-0.88640526	0.9388142	0.96277471]
[1.15182818	1.12815215	0.97716054	-1.17241113]
[1.16914891	-0.88640526	0.97716054	1.73920592]
[1.18646963	-0.88640526	1.01550688	-0.90066021]
[1.20379036	1.12815215	1.01550688	0.49691598]
[1.22111108	1.12815215	1.01550688	-1.44416206]
[1.23843181	1.12815215	1.01550688	0.96277471]
[1.25575253	1.12815215	1.01550688	-1.56062674]
[1.27307326	1.12815215	1.01550688	1.62274124]
[1.29039398	-0.88640526	1.05385321	-1.44416206]
[1.30771471	-0.88640526	1.05385321	1.38981187]
[1.32503543	1.12815215	1.05385321	-1.36651894]
[1.34235616	1.12815215	1.05385321	0.72984534]
[1.35967688	1.12815215	1.2455849	-1.4053405]
[1.3769976	1.12815215	1.2455849	1.54509812]
[1.39431833	-0.88640526	1.39897025	-0.7065524]
[1.41163905	-0.88640526	1.39897025	1.38981187]
[1.42895978	1.12815215	1.43731659	-1.36651894]
[1.4462805	-0.88640526	1.43731659	1.46745499]
[1.46360123	-0.88640526	1.47566292	-0.43480148]
[1.48092195	1.12815215	1.47566292	1.81684904]
[1.49824268	-0.88640526	1.5523556	-1.01712489]


```
[ 1.5155634  1.12815215  1.5523556  0.69102378]
[ 1.53288413 -0.88640526  1.62904827 -1.28887582]
[ 1.55020485 -0.88640526  1.62904827  1.35099031]
[ 1.56752558 -0.88640526  1.62904827 -1.05594645]
[ 1.5848463  -0.88640526  1.62904827  0.72984534]
[ 1.60216702  1.12815215  2.01251165 -1.63826986]
[ 1.61948775 -0.88640526  2.01251165  1.58391968]
[ 1.63680847 -0.88640526  2.28093601 -1.32769738]
[ 1.6541292  -0.88640526  2.28093601  1.11806095]
[ 1.67144992 -0.88640526  2.51101403 -0.86183865]
[ 1.68877065  1.12815215  2.51101403  0.92395314]
[ 1.70609137  1.12815215  2.76985181 -1.25005425]
[ 1.7234121  1.12815215  2.76985181  1.27334719]]
```

```
X_scaled=pd.DataFrame(scale,columns=X.columns)
X_scaled
```

Output:

```
In [56]: X_scaled = pd.DataFrame(scale, columns = X.columns)
X_scaled
```

```
Out[56]:
```

	CustomerID	Gender	Annual Income (k\$)	Spending Score (1-100)
0	-1.723412	1.128152	-1.745429	-0.434801
1	-1.706091	1.128152	-1.745429	1.195704
2	-1.688771	-0.886405	-1.707083	-1.715913
3	-1.671450	-0.886405	-1.707083	1.040418
4	-1.654129	-0.886405	-1.668737	-0.395980
...
195	1.654129	-0.886405	2.280936	1.118061
196	1.671450	-0.886405	2.511014	-0.861839
197	1.688771	1.128152	2.511014	0.923953
198	1.706091	1.128152	2.769852	-1.250054
199	1.723412	1.128152	2.769852	1.273347

200 rows × 4 columns

```
#train test split
fromsklearn.model_selectionimporttrain_test_split
# split the dataset
X_train, X_test, Y_train, Y_test=train_test_split(X_scaled, Y,
test_size=0.20, random_state=0)
```

In [58]:

```
X_train.shape
```

Output:

```
(160, 4)
```

```
X_test.shape
```

Output:

```
(40, 4)
```

```
Y_train.shape
```

Output:

```
(160,)
```

```
Y_test.shape
```

Output:

```
(40,)
```

9. Perform any of the clustering algorithms

ANS:

#Clustering Algorithm

```
x =df.iloc[:, [3, 4]].values
```

In [63]:

#finding optimal number of clusters using the elbow method

```
fromsklearn.clusterimportKMeans
```

```
wcss_list= [] #Initializing the list for the values of WCSS
```

In [64]:

#Using for loop for iterations from 1 to 10.

```
foriin range(1, 11):
```

```
    kmeans=KMeans(n_clusters=i, init='k-means++', random_state= 42)
```

```
    kmeans.fit(x)
```

```
    wcss_list.append(kmeans.inertia_)
```

```
plt.plot(range(1, 11), wcss_list)
```

```
plt.title('The Elbow Method Graph')
```

```
plt.xlabel('Number of clusters(k)')
```

```
plt.ylabel('wcss_list')
```

```
plt.show()
```

Output:



```

#training the K-means model on a dataset
kmeans=KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict=kmeans.fit_predict(x)

```

In [66]:

```

#visulaizing the clusters
plt.scatter(x[y_predict== 0, 0], x[y_predict== 0, 1], s = 100, c = 'blue',
label = 'Cluster 1') #for first cluster
plt.scatter(x[y_predict== 1, 0], x[y_predict== 1, 1], s = 100, c = 'green',
label = 'Cluster 2') #for second cluster
plt.scatter(x[y_predict== 2, 0], x[y_predict== 2, 1], s = 100, c = 'red',
label = 'Cluster 3') #for third cluster
plt.scatter(x[y_predict== 3, 0], x[y_predict== 3, 1], s = 100, c = 'cyan',
label = 'Cluster 4') #for fourth cluster
plt.scatter(x[y_predict== 4, 0], x[y_predict== 4, 1], s = 100, c =
'magenta', label = 'Cluster 5') #for fifth cluster
plt.scatter(kmeans.cluster_centers_[0], kmeans.cluster_centers_[0], s
= 300, c = 'yellow', label = 'Centroid')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()

```

Output:

```
In [65]: #training the K-means model on a dataset
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)
```

```
In [66]: #visualaizing the clusters
plt.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #for first cluster
plt.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #for second cluster
plt.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #for third cluster
plt.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #for fourth cluster
plt.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') #for fifth cluster
plt.scatter(kmeans.cluster_centers[:, 0], kmeans.cluster_centers[:, 1], s = 300, c = 'yellow', label = 'Centroid')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

