# Assignment 4
# Python Programming

| | |
|---|---|
| Assignment Date | 25/10/2022 |
| Student Name | Kokila.V |
| Student Roll Number | 730419205023 |
| Maximum Marks | 2 Marks |

**Question 1:**

**1 . Importing Required Package**

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

**Question 2:**

**2 . Loading the Datase**

```
db = pd.read_csv('/content/Mall_Customers.csv')
db
```

```
     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
..          ...     ...  ...                 ...                     ...
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 x 5 columns]
```

**Question 3:**

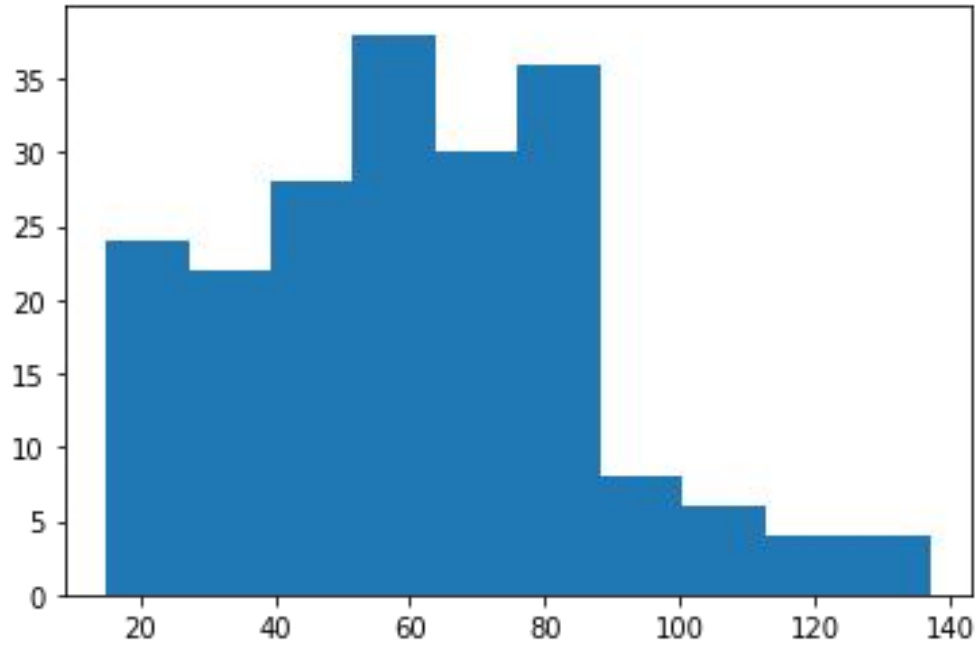
**3. Visualizations**

**3.1 UniVariate Analysis**

**3.1.1 Solution**

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

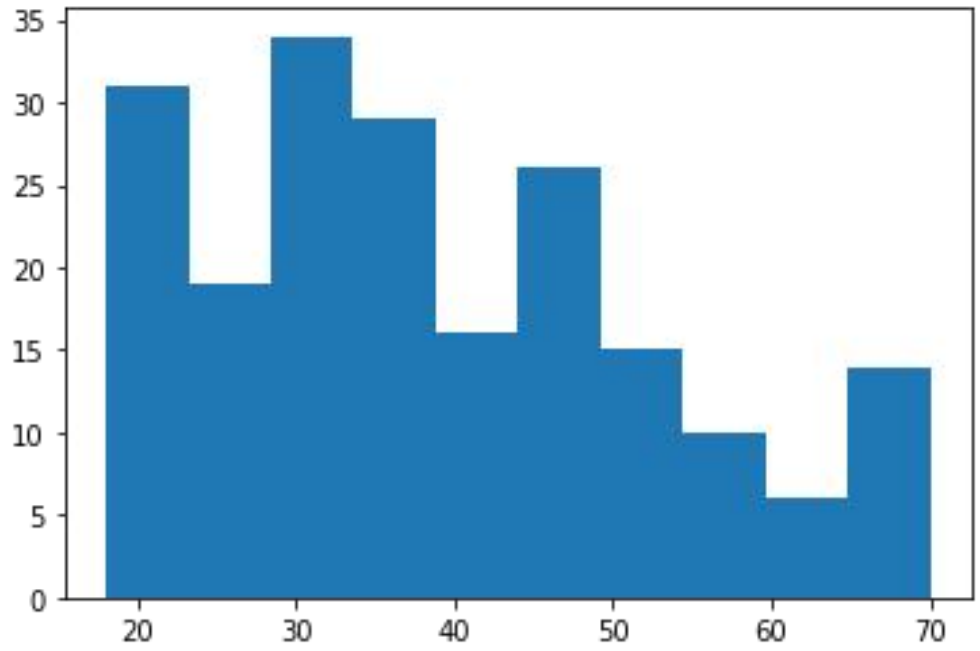
```
(array([24., 22., 28., 38., 30., 36.,  8.,  6.,  4.,  4.]),
 array([ 15. ,  27.2,  39.4,  51.6,  63.8,  76. ,  88.2, 100.4, 112.6,
        124.8, 137. ]),
 <a list of 10 Patch objects>)
```




### 3.1.2 Solution

```
plt.hist(db['Age'])
```

```
(array([31., 19., 34., 29., 16., 26., 15., 10.,  6., 14.]),
 array([18. , 23.2, 28.4, 33.6, 38.8, 44. , 49.2, 54.4, 59.6, 64.8, 70. ]),
 <a list of 10 Patch objects>)
```

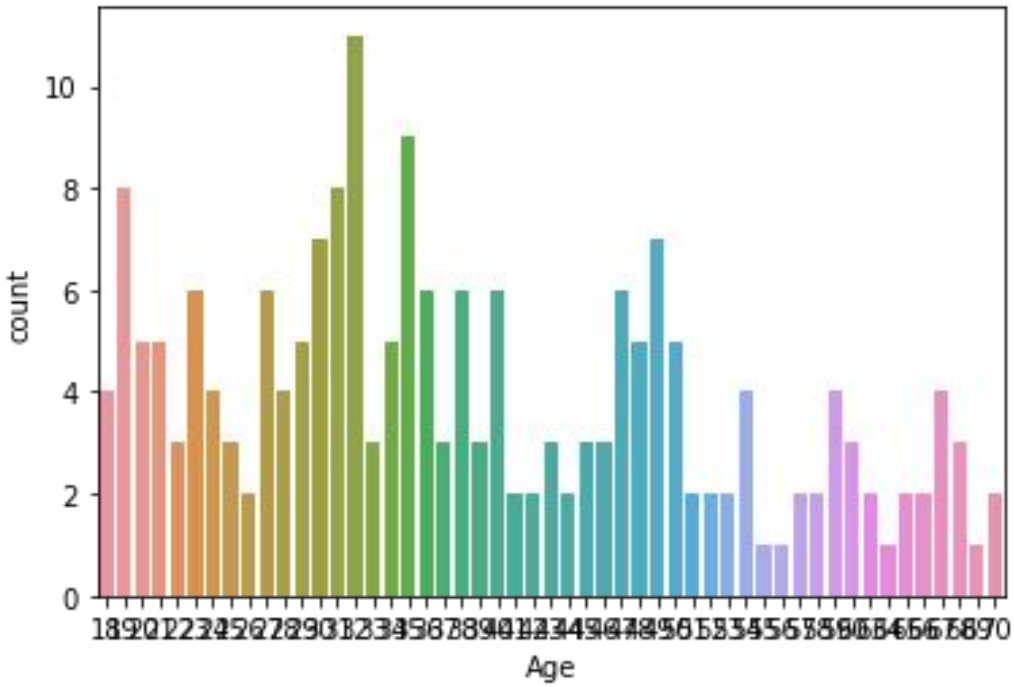



### 3.1.3 Solution

```
sbn.countplot(db['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4b0afd0090>
```

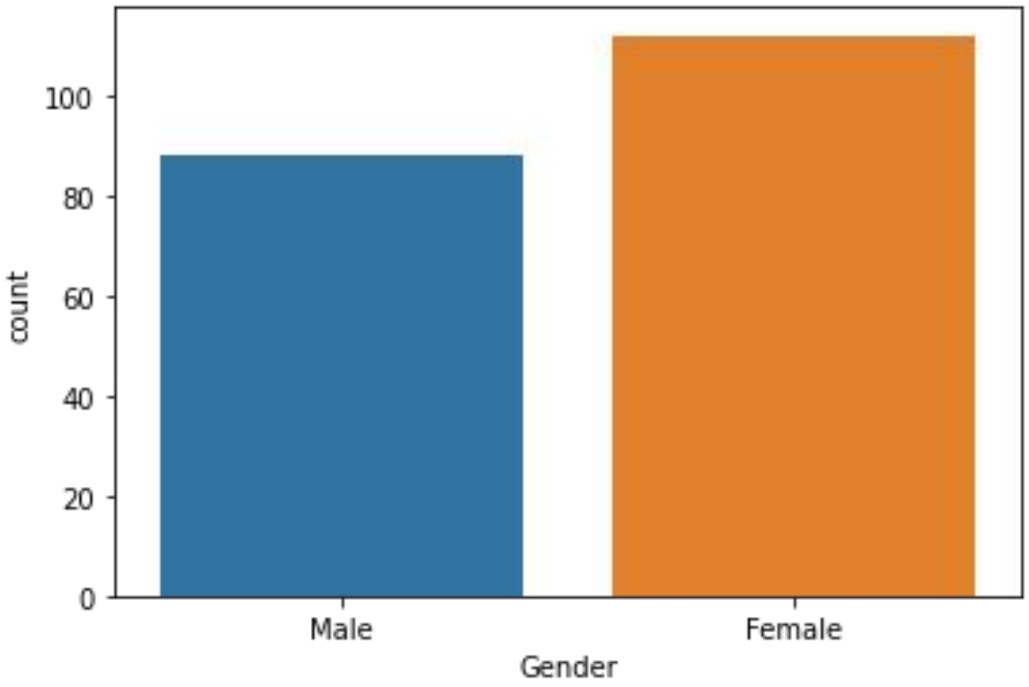



### 3.1.4 Solution

```
sbn.countplot(db['Gender'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

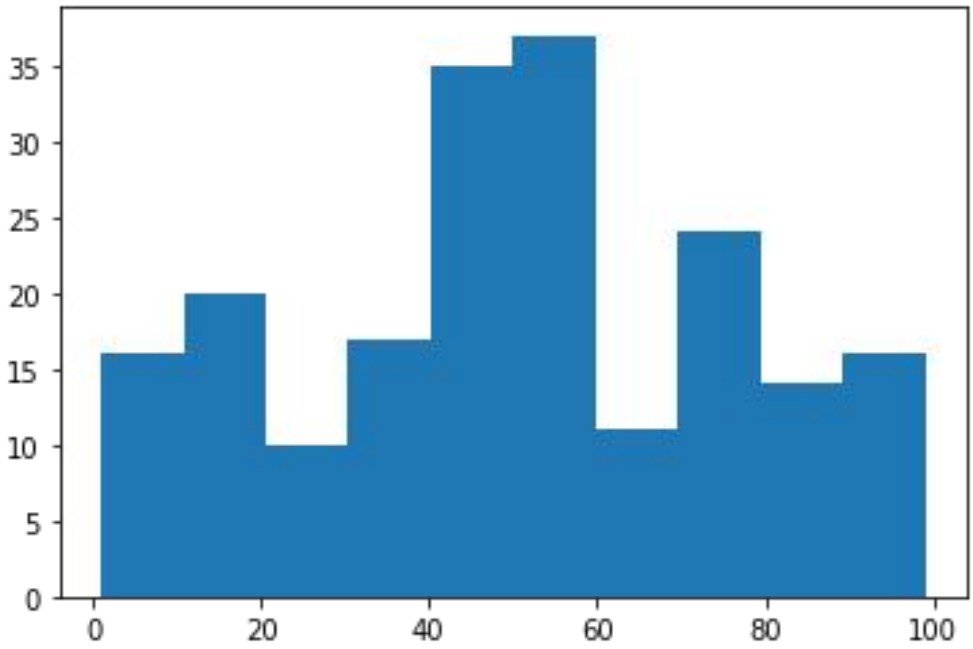
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4b0ae45790>
```




### 3.1.5 Solution

```
plt.hist(db['Spending Score (1-100)'])
```

```
(array([16., 20., 10., 17., 35., 37., 11., 24., 14., 16.]),
 array([ 1. , 10.8, 20.6, 30.4, 40.2, 50. , 59.8, 69.6, 79.4, 89.2, 99. ]),
 <a list of 10 Patch objects>)
```

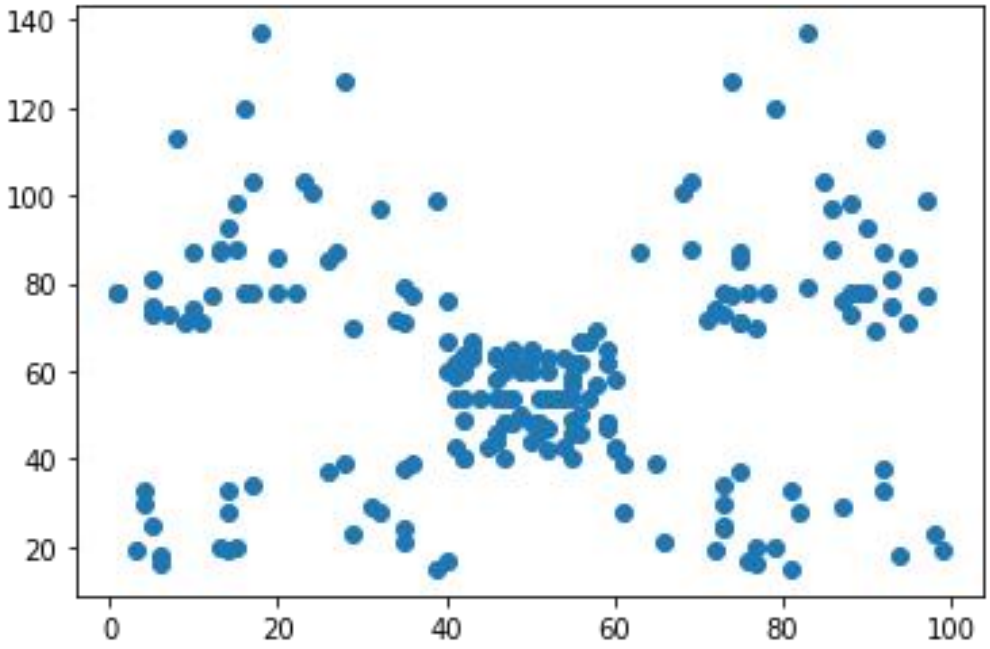



### 3.2 Bi-Variate Analysis

### 3.2.1 Solution

```
plt.scatter(db['Spending Score (1-100)'],db['Annual Income (k$)'])
```

```
<matplotlib.collections.PathCollection at 0x7f4b0acfadd0>
```




### 3.2.2 Solution

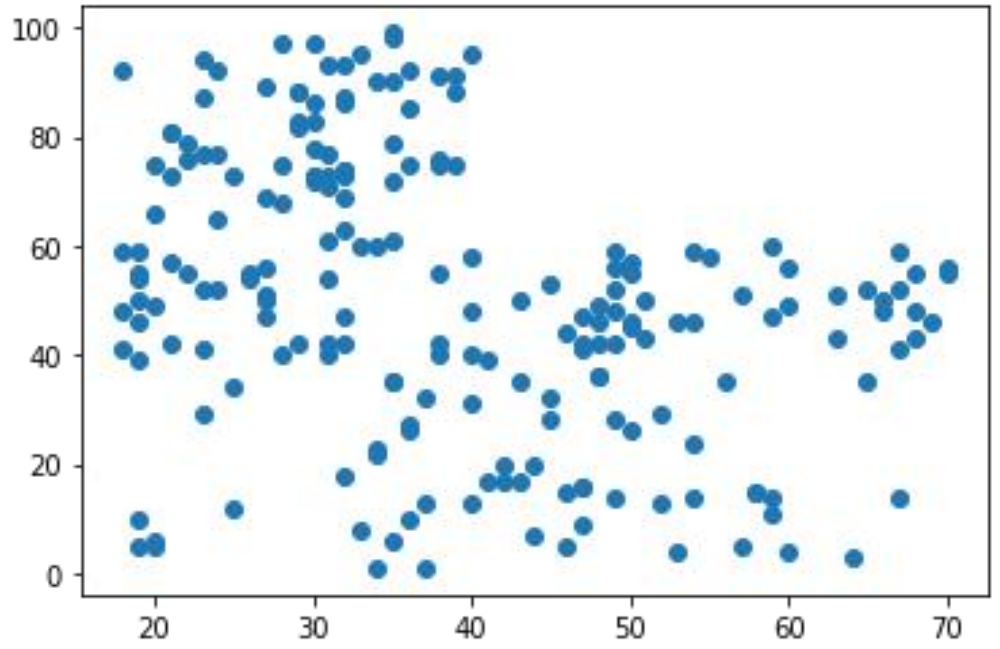
```
plt.scatter(db['Gender'],db['Annual Income (k$)'])
```

```
<matplotlib.collections.PathCollection at 0x7f4b0acd3410>
```


### 3.2.3 Solution

```
plt.scatter(db['Age'],db['Spending Score (1-100)'])
```

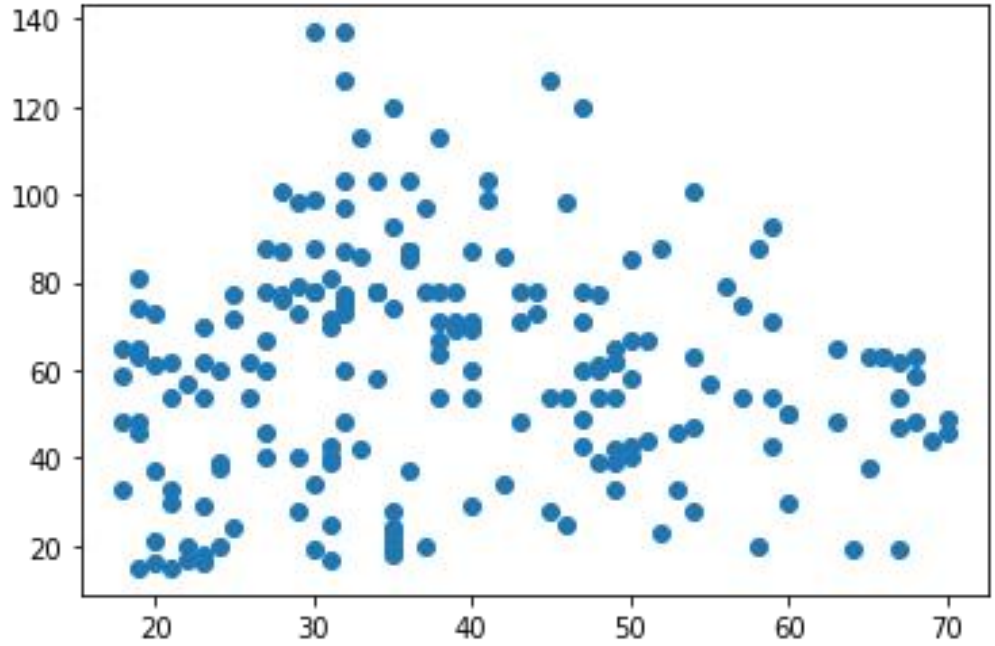
```
<matplotlib.collections.PathCollection at 0x7f4b0aca6550>
```




### 3.2.4 Solution

```
plt.scatter(db['Age'],db['Annual Income (k$)'])
```

```
<matplotlib.collections.PathCollection at 0x7f4b0c009790>
```




### 3.2.5 Solution

```
sbn.heatmap(db.corr(), annot = True)
```

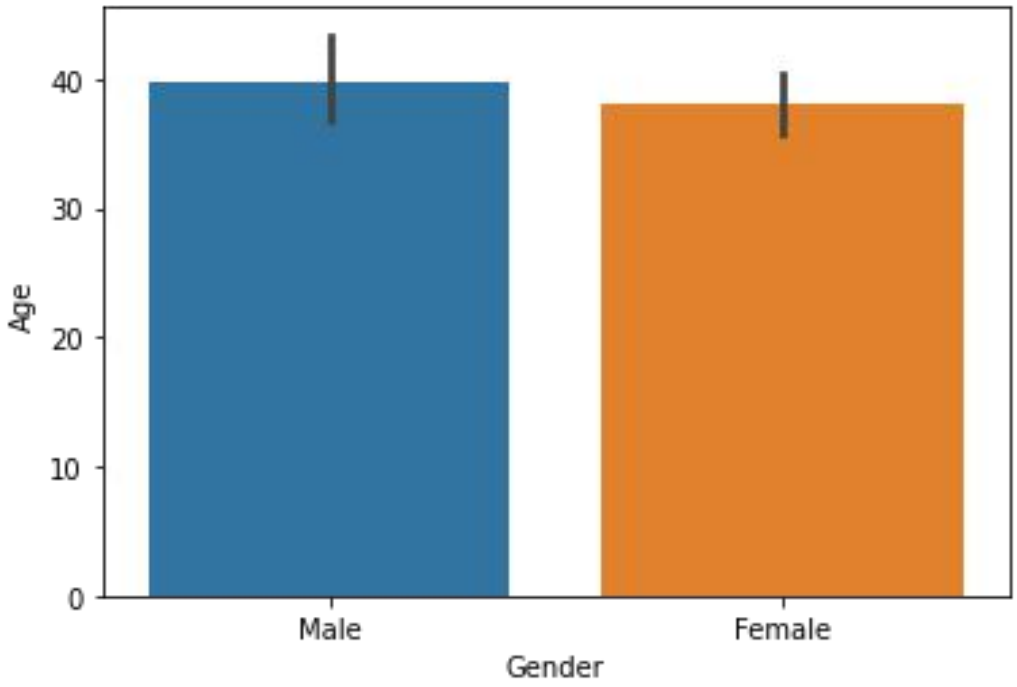
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4b0b527450>
```


### 3.2.6 Solution

```
sbn.barplot(db['Gender'], db['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y. 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.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4b0830c450>
```

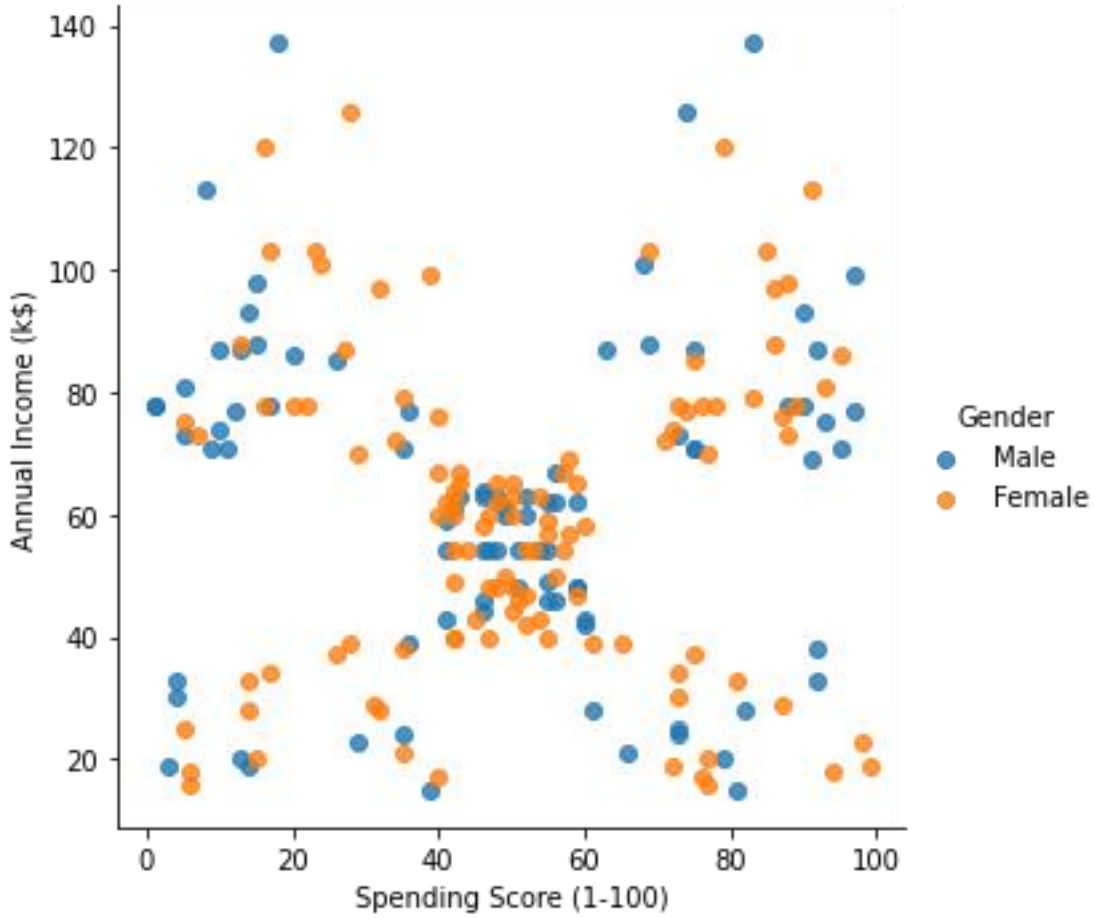



### 3.3 Multi-Variate Analysis

#### 3.3.1 Solution

```
sbn.lmplot("Spending Score (1-100)","Annual Income (k$)", db, hue="Gender",
fit_reg=False);
```

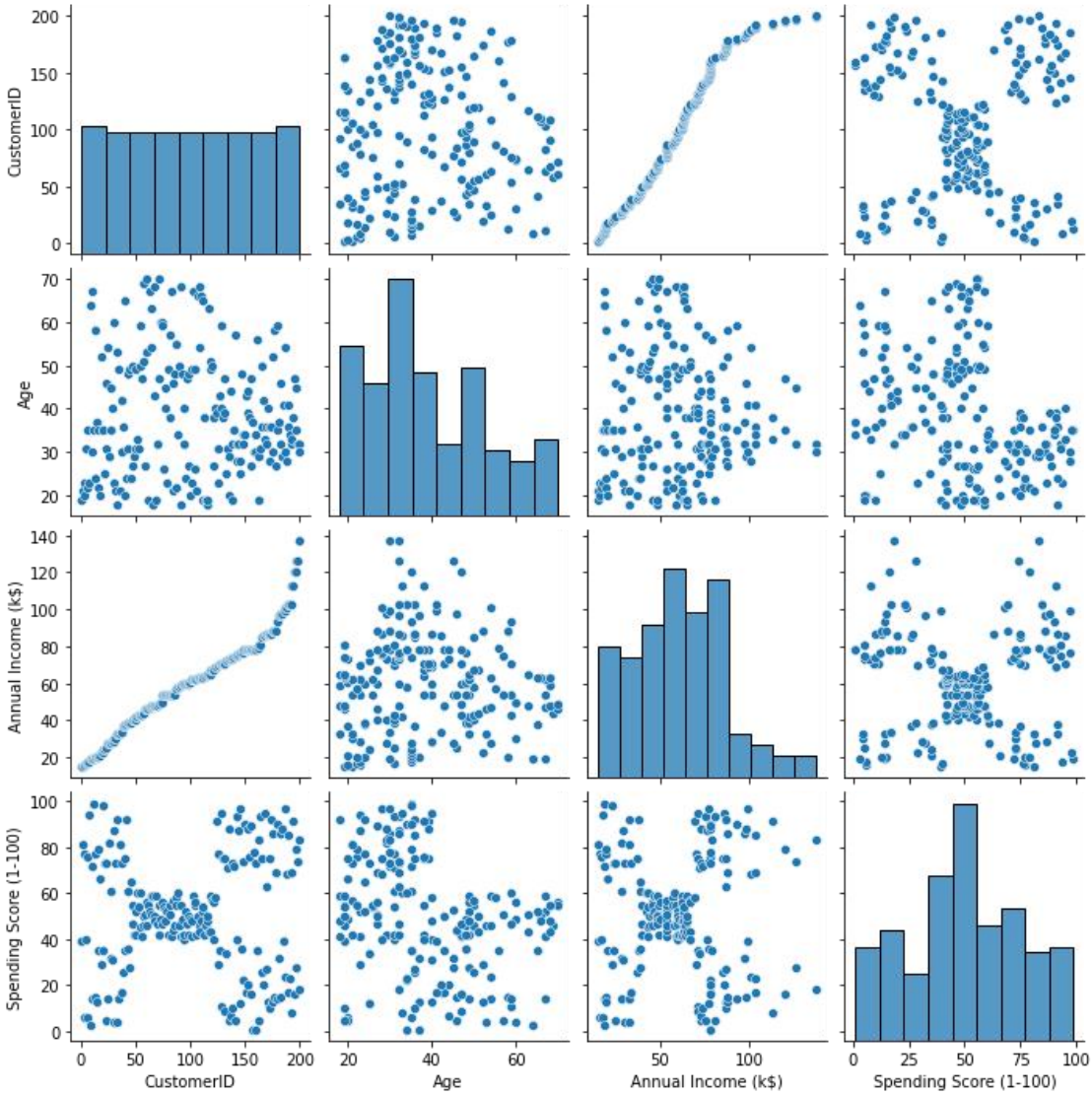
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y, data. From
version 0.12, the only valid positional argument will be `data`, and passing
other arguments without an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
```




### 3.3.2 Solution

```
sbn.pairplot(db)
```

```
<seaborn.axisgrid.PairGrid at 0x7f4b081f76d0>
```




**Question 4:**

**4. Perform descriptive statistics on the dataset**

**4.1 Solution**

```
db.describe()
```

| | 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 |

### 4.2 Solution

```
db.dtypes
```

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

### 4.3 Solution

```
db.var()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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.
  """Entry point for launching an IPython kernel.
```

```
CustomerID              3350.000000
Age                      195.133166
Annual Income (k$)       689.835578
Spending Score (1-100)   666.854271
dtype: float64
```

### 4.4 Solution

```
db.skew()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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.
  """Entry point for launching an IPython kernel.
```

```
CustomerID              0.000000
Age                     0.485569
Annual Income (k$)      0.321843
Spending Score (1-100)  -0.047220
dtype: float64
```

### 4.5 Solution

```
db.corr()
```

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

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

### 4.6 Solution

```
db.std()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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.
  """Entry point for launching an IPython kernel.
```

```
CustomerID              57.879185
Age                     13.969007
Annual Income (k$)      26.264721
Spending Score (1-100)  25.823522
dtype: float64
```

### Question 5:

### 5. Check for Missing values and deal with them

### 5.1 Solution

```
db.isna().sum()
```

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

### 5.2 Solution

```
db.isna().sum().sum()
```

```
0
```

### 5.3 Solution

```
db.duplicated().sum()
```

```
0
```

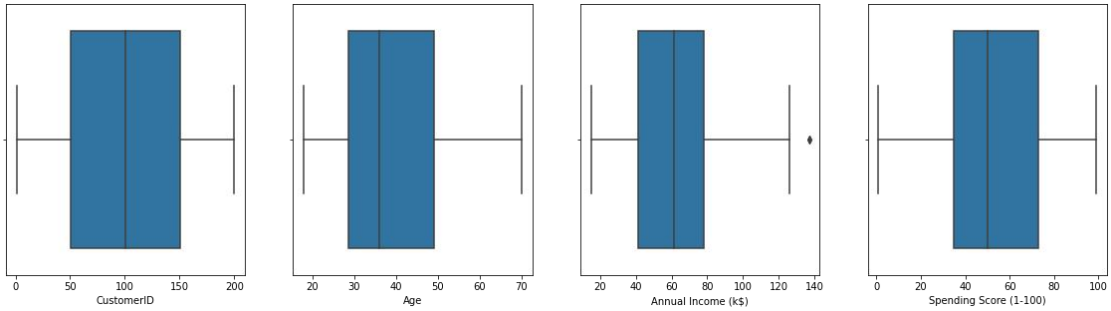
### Question 6:

### 6 . Find the outliers and replace them outliers

### 6.1 Solutiion

```
ig,ax=plt.subplots(figsize=(25,5))
plt.subplot(1, 5, 2)
sbn.boxplot(x=db['Age'])
plt.subplot(1, 5, 3)
sbn.boxplot(x=db['Annual Income (k$)'])
plt.subplot(1, 5, 4)
sbn.boxplot(x=db['Spending Score (1-100)'])
plt.subplot(1, 5, 1)
sbn.boxplot(x=db['CustomerID'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4b061aa610>
```




## 6.2 Solution

```
q=db.quantile(q = [0.25, 0.75])
q
```

```
      CustomerID    Age  Annual Income (k$)  Spending Score (1-100)
0.25       50.75  28.75                41.5                   34.75
0.75      150.25  49.00                78.0                   73.00
```

## 6.3 Solution

```
q.loc[0.75]
```

```
CustomerID              150.25
Age                      49.00
Annual Income (k$)         78.00
Spending Score (1-100)     73.00
Name: 0.75, dtype: float64
```

## 6.4 Solution

```
q.loc[0.25]
```

```
CustomerID              50.75
Age                    28.75
Annual Income (k$)       41.50
Spending Score (1-100)    34.75
Name: 0.25, dtype: float64
```

## 6.5 Solution

```
IQR=q.iloc[1]-q.iloc[0]
IQR
```

```
CustomerID               99.50
Age                     20.25
Annual Income (k$)       36.50
Spending Score (1-100)    38.25
dtype: float64
```

### 6.6 Solution

```
upper=q.iloc[1] + (1.5 *IQR)
upper
```

```
CustomerID              299.500
Age                      79.375
Annual Income (k$)       132.750
Spending Score (1-100)    130.375
dtype: float64
```

### 6.7 Solution

```
lower=q.iloc[0] - (1.5* IQR)
lower
```

```
CustomerID             -98.500
Age                     -1.625
Annual Income (k$)      -13.250
Spending Score (1-100)   -22.625
dtype: float64
```

### 6.8 Solution

```
db.mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.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.
  """Entry point for launching an IPython kernel.
```

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

### 6.9 Solution

```
db['Annual Income (k$)'].max()
```

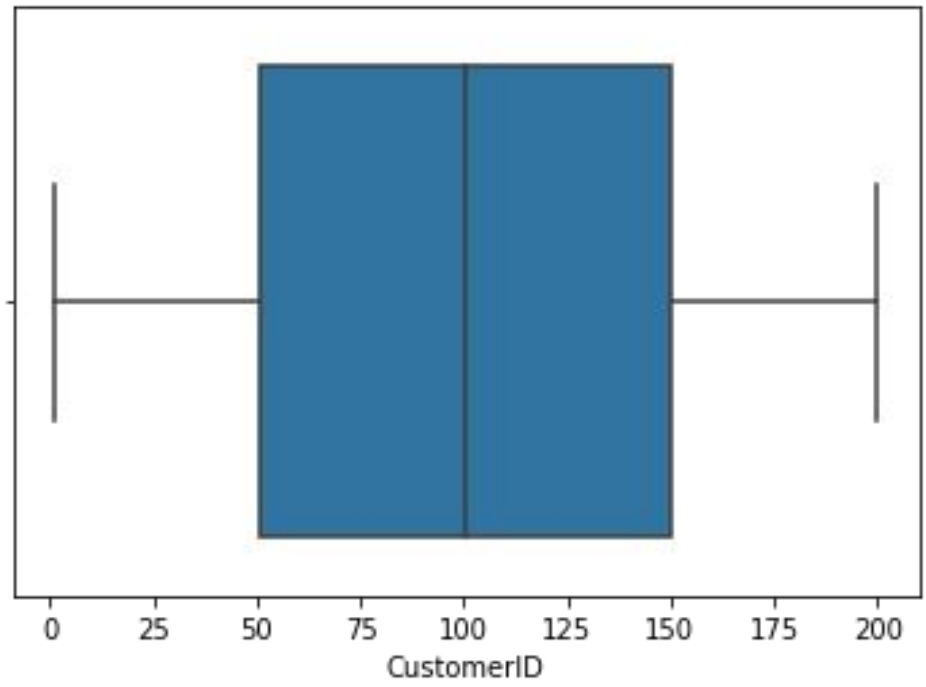
```
137
```

### 6.10 Solution

```
sbn.boxplot(db['CustomerID'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f94da48bfd0>
```

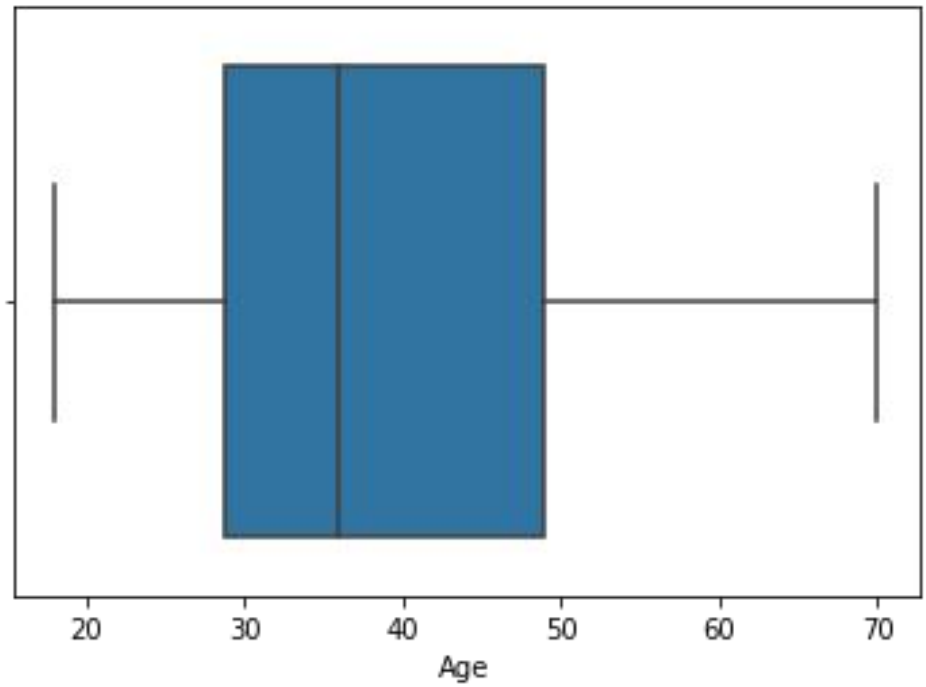



### 6.11 Solution

```
sbn.boxplot(db['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4b06126f10>
```

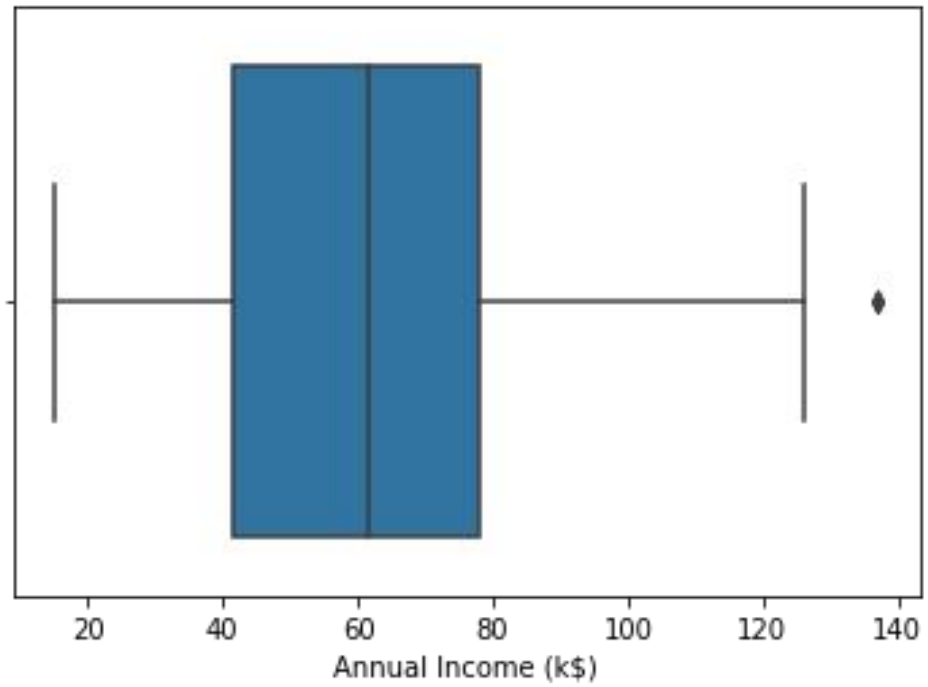



### 6.12 Solution

```
sbn.boxplot(db['Annual Income (k$)'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f94da411d10>
```

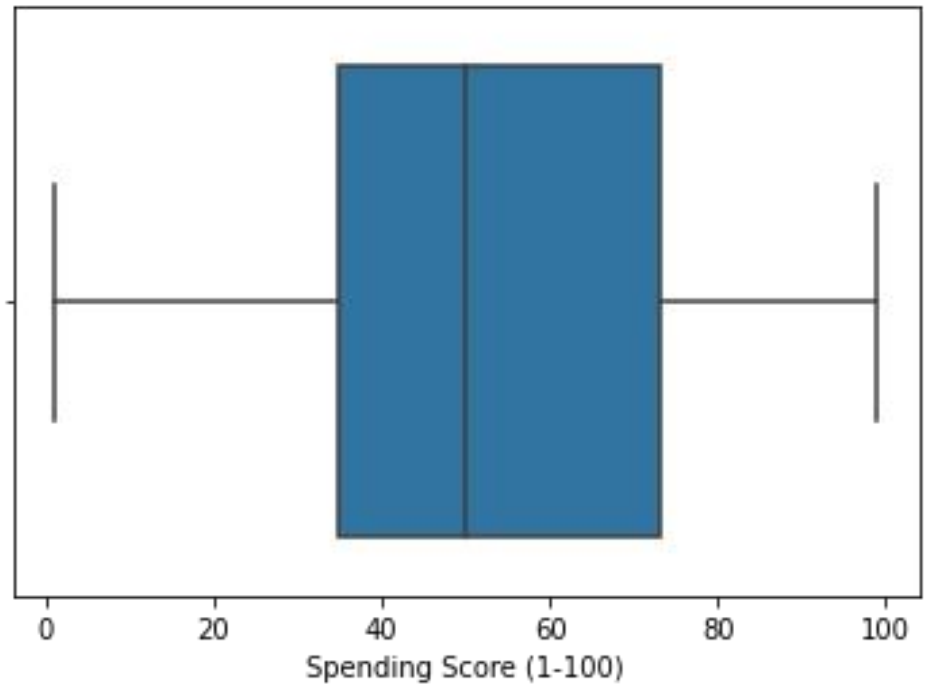



### 6.13 Solution

```
sbn.boxplot(db['Spending Score (1-100)'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4b060a9c10>
```




**Question 7**

**7. Check for Categorical columns and perform encoding**

**7.1 Solution**

```
db.select_dtypes(include='object').columns
```

```
Index(['Gender'], dtype='object')
```

**7.2 Solution**

```
db['Gender'].unique()
```

```
array(['Male', 'Female'], dtype=object)
```

**7.3 Solution**

```
db['Gender'].replace({'Male':1,'Female':0},inplace=True)
db
```

```
     CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0             1       1   19                  15                      39
1             2       1   21                  15                      81
2             3       0   20                  16                       6
3             4       0   23                  16                      77
4             5       0   31                  17                      40
..          ...     ...  ...                 ...                     ...
195         196       0   35                 120                      79
196         197       0   45                 126                      28
197         198       1   32                 126                      74
```

```
198         199       1   32                137                      18
199         200       1   30                137                      83

[200 rows x 5 columns]
```

### 7.4 Solution

```
db.head()
```

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

## Question 8

## 8 . Scaling the data

### 8.1 Solution

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler().fit_transform(db)

ss
```

```
array([[-1.7234121 ,  1.12815215, -1.42456879, -1.73899919, -0.43480148],
       [-1.70609137,  1.12815215, -1.28103541, -1.73899919,  1.19570407],
       [-1.68877065, -0.88640526, -1.3528021 , -1.70082976, -1.71591298],
       [-1.67144992, -0.88640526, -1.13750203, -1.70082976,  1.04041783],
       [-1.6541292 , -0.88640526, -0.56336851, -1.66266033, -0.39597992],
       [-1.63680847, -0.88640526, -1.20926872, -1.66266033,  1.00159627],
       [-1.61948775, -0.88640526, -0.27630176, -1.62449091, -1.71591298],
       [-1.60216702, -0.88640526, -1.13750203, -1.62449091,  1.70038436],
       [-1.5848463 ,  1.12815215,  1.80493225, -1.58632148, -1.83237767],
       [-1.56752558, -0.88640526, -0.6351352 , -1.58632148,  0.84631002],
       [-1.55020485,  1.12815215,  2.02023231, -1.58632148, -1.4053405 ],
       [-1.53288413, -0.88640526, -0.27630176, -1.58632148,  1.89449216],
       [-1.5155634 , -0.88640526,  1.37433211, -1.54815205, -1.36651894],
       [-1.49824268, -0.88640526, -1.06573534, -1.54815205,  1.04041783],
       [-1.48092195,  1.12815215, -0.13276838, -1.54815205, -1.44416206],
       [-1.46360123,  1.12815215, -1.20926872, -1.54815205,  1.11806095],
       [-1.4462805 , -0.88640526, -0.27630176, -1.50998262, -0.59008772],
       [-1.42895978,  1.12815215, -1.3528021 , -1.50998262,  0.61338066],
       [-1.41163905,  1.12815215,  0.94373197, -1.43364376, -0.82301709],
       [-1.39431833, -0.88640526, -0.27630176, -1.43364376,  1.8556706 ],
       [-1.3769976 ,  1.12815215, -0.27630176, -1.39547433, -0.59008772],
       [-1.35967688,  1.12815215, -0.99396865, -1.39547433,  0.88513158],
       [-1.34235616, -0.88640526,  0.51313183, -1.3573049 , -1.75473454],
       [-1.32503543,  1.12815215, -0.56336851, -1.3573049 ,  0.88513158],
       [-1.30771471, -0.88640526,  1.08726535, -1.24279661, -1.4053405 ],
```

```
[-1.29039398,  1.12815215, -0.70690189, -1.24279661,  1.23452563],
[-1.27307326, -0.88640526,  0.44136514, -1.24279661, -0.7065524 ],
[-1.25575253,  1.12815215, -0.27630176, -1.24279661,  0.41927286],
[-1.23843181, -0.88640526,  0.08253169, -1.20462718, -0.74537397],
[-1.22111108, -0.88640526, -1.13750203, -1.20462718,  1.42863343],
[-1.20379036,  1.12815215,  1.51786549, -1.16645776, -1.7935561 ],
[-1.18646963, -0.88640526, -1.28103541, -1.16645776,  0.88513158],
[-1.16914891,  1.12815215,  1.01549866, -1.05194947, -1.7935561 ],
[-1.15182818,  1.12815215, -1.49633548, -1.05194947,  1.62274124],
[-1.13450746, -0.88640526,  0.7284319 , -1.05194947, -1.4053405 ],
[-1.11718674, -0.88640526, -1.28103541, -1.05194947,  1.19570407],
[-1.09986601, -0.88640526,  0.22606507, -1.01378004, -1.28887582],
[-1.08254529, -0.88640526, -0.6351352 , -1.01378004,  0.88513158],
[-1.06522456, -0.88640526, -0.20453507, -0.89927175, -0.93948177],
[-1.04790384, -0.88640526, -1.3528021 , -0.89927175,  0.96277471],
[-1.03058311, -0.88640526,  1.87669894, -0.86110232, -0.59008772],
[-1.01326239,  1.12815215, -1.06573534, -0.86110232,  1.62274124],
[-0.99594166,  1.12815215,  0.65666521, -0.82293289, -0.55126616],
[-0.97862094, -0.88640526, -0.56336851, -0.82293289,  0.41927286],
[-0.96130021, -0.88640526,  0.7284319 , -0.82293289, -0.86183865],
[-0.94397949, -0.88640526, -1.06573534, -0.82293289,  0.5745591 ],
[-0.92665877, -0.88640526,  0.80019859, -0.78476346,  0.18634349],
[-0.90933804, -0.88640526, -0.85043527, -0.78476346, -0.12422899],
[-0.89201732, -0.88640526, -0.70690189, -0.78476346, -0.3183368 ],
[-0.87469659, -0.88640526, -0.56336851, -0.78476346, -0.3183368 ],
[-0.85737587, -0.88640526,  0.7284319 , -0.70842461,  0.06987881],
[-0.84005514,  1.12815215, -0.41983513, -0.70842461,  0.38045129],
[-0.82273442, -0.88640526, -0.56336851, -0.67025518,  0.14752193],
[-0.80541369,  1.12815215,  1.4460988 , -0.67025518,  0.38045129],
[-0.78809297, -0.88640526,  0.80019859, -0.67025518, -0.20187212],
[-0.77077224,  1.12815215,  0.58489852, -0.67025518, -0.35715836],
[-0.75345152, -0.88640526,  0.87196528, -0.63208575, -0.00776431],
[-0.73613079,  1.12815215,  2.16376569, -0.63208575, -0.16305055],
[-0.71881007, -0.88640526, -0.85043527, -0.55574689,  0.03105725],
[-0.70148935,  1.12815215,  1.01549866, -0.55574689, -0.16305055],
[-0.68416862,  1.12815215,  2.23553238, -0.55574689,  0.22516505],
[-0.6668479 ,  1.12815215, -1.42456879, -0.55574689,  0.18634349],
[-0.64952717, -0.88640526,  2.02023231, -0.51757746,  0.06987881],
[-0.63220645, -0.88640526,  1.08726535, -0.51757746,  0.34162973],
[-0.61488572,  1.12815215,  1.73316556, -0.47940803,  0.03105725],
[-0.597565  ,  1.12815215, -1.49633548, -0.47940803,  0.34162973],
[-0.58024427, -0.88640526,  0.29783176, -0.47940803, -0.00776431],
[-0.56292355, -0.88640526,  2.091999  , -0.47940803, -0.08540743],
[-0.54560282,  1.12815215, -1.42456879, -0.47940803,  0.34162973],
[-0.5282821 , -0.88640526, -0.49160182, -0.47940803, -0.12422899],
[-0.51096138,  1.12815215,  2.23553238, -0.4412386 ,  0.18634349],
[-0.49364065, -0.88640526,  0.58489852, -0.4412386 , -0.3183368 ],
[-0.47631993, -0.88640526,  1.51786549, -0.40306917, -0.04658587],
[-0.4589992 , -0.88640526,  1.51786549, -0.40306917,  0.22516505],
[-0.44167848,  1.12815215,  1.4460988 , -0.25039146, -0.12422899],
```

```
[-0.42435775,  1.12815215, -0.92220196, -0.25039146,  0.14752193],
[-0.40703703, -0.88640526,  0.44136514, -0.25039146,  0.10870037],
[-0.3897163 ,  1.12815215,  0.08253169, -0.25039146, -0.08540743],
[-0.37239558, -0.88640526, -1.13750203, -0.25039146,  0.06987881],
[-0.35507485, -0.88640526,  0.7284319 , -0.25039146, -0.3183368 ],
[-0.33775413,  1.12815215,  1.30256542, -0.25039146,  0.03105725],
[-0.3204334 ,  1.12815215, -0.06100169, -0.25039146,  0.18634349],
[-0.30311268,  1.12815215,  2.02023231, -0.25039146, -0.35715836],
[-0.28579196, -0.88640526,  0.51313183, -0.25039146, -0.24069368],
[-0.26847123, -0.88640526, -1.28103541, -0.25039146,  0.26398661],
[-0.25115051,  1.12815215,  0.65666521, -0.25039146, -0.16305055],
[-0.23382978, -0.88640526,  1.15903204, -0.13588317,  0.30280817],
[-0.21650906, -0.88640526, -1.20926872, -0.13588317,  0.18634349],
[-0.19918833, -0.88640526, -0.34806844, -0.09771374,  0.38045129],
[-0.18186761, -0.88640526,  0.80019859, -0.09771374, -0.16305055],
[-0.16454688, -0.88640526,  2.091999  , -0.05954431,  0.18634349],
[-0.14722616,  1.12815215, -1.49633548, -0.05954431, -0.35715836],
[-0.12990543,  1.12815215,  0.65666521, -0.02137488, -0.04658587],
[-0.11258471, -0.88640526,  0.08253169, -0.02137488, -0.39597992],
[-0.09526399, -0.88640526, -0.49160182, -0.02137488, -0.3183368 ],
[-0.07794326,  1.12815215, -1.06573534, -0.02137488,  0.06987881],
[-0.06062254, -0.88640526,  0.58489852, -0.02137488, -0.12422899],
[-0.04330181, -0.88640526, -0.85043527, -0.02137488, -0.00776431],
[-0.02598109,  1.12815215,  0.65666521,  0.01679455, -0.3183368 ],
[-0.00866036,  1.12815215, -1.3528021 ,  0.01679455, -0.04658587],
[ 0.00866036, -0.88640526, -1.13750203,  0.05496398, -0.35715836],
[ 0.02598109, -0.88640526,  0.7284319 ,  0.05496398, -0.08540743],
[ 0.04330181,  1.12815215,  2.02023231,  0.05496398,  0.34162973],
[ 0.06062254,  1.12815215, -0.92220196,  0.05496398,  0.18634349],
[ 0.07794326,  1.12815215,  0.7284319 ,  0.05496398,  0.22516505],
[ 0.09526399, -0.88640526, -1.28103541,  0.05496398, -0.3183368 ],
[ 0.11258471, -0.88640526,  1.94846562,  0.09313341, -0.00776431],
[ 0.12990543,  1.12815215,  1.08726535,  0.09313341, -0.16305055],
[ 0.14722616,  1.12815215,  2.091999  ,  0.09313341, -0.27951524],
[ 0.16454688,  1.12815215,  1.94846562,  0.09313341, -0.08540743],
[ 0.18186761,  1.12815215,  1.87669894,  0.09313341,  0.06987881],
[ 0.19918833, -0.88640526, -1.42456879,  0.09313341,  0.14752193],
[ 0.21650906, -0.88640526, -0.06100169,  0.13130284, -0.3183368 ],
[ 0.23382978,  1.12815215, -1.42456879,  0.13130284, -0.16305055],
[ 0.25115051, -0.88640526, -1.49633548,  0.16947227, -0.08540743],
[ 0.26847123, -0.88640526, -1.42456879,  0.16947227, -0.00776431],
[ 0.28579196, -0.88640526,  1.73316556,  0.16947227, -0.27951524],
[ 0.30311268, -0.88640526,  0.7284319 ,  0.16947227,  0.34162973],
[ 0.3204334 , -0.88640526,  0.87196528,  0.24581112, -0.27951524],
[ 0.33775413, -0.88640526,  0.80019859,  0.24581112,  0.26398661],
[ 0.35507485,  1.12815215, -0.85043527,  0.24581112,  0.22516505],
[ 0.37239558, -0.88640526, -0.06100169,  0.24581112, -0.39597992],
[ 0.3897163 , -0.88640526,  0.08253169,  0.32214998,  0.30280817],
[ 0.40703703,  1.12815215,  0.010765  ,  0.32214998,  1.58391968],
[ 0.42435775, -0.88640526, -1.13750203,  0.36031941, -0.82301709],
```

```
[ 0.44167848, -0.88640526, -0.56336851,  0.36031941,  1.04041783],
[ 0.4589992 ,  1.12815215,  0.29783176,  0.39848884, -0.59008772],
[ 0.47631993,  1.12815215,  0.08253169,  0.39848884,  1.73920592],
[ 0.49364065,  1.12815215,  1.4460988 ,  0.39848884, -1.52180518],
[ 0.51096138,  1.12815215, -0.06100169,  0.39848884,  0.96277471],
[ 0.5282821 ,  1.12815215,  0.58489852,  0.39848884, -1.5994483 ],
[ 0.54560282,  1.12815215,  0.010765  ,  0.39848884,  0.96277471],
[ 0.56292355, -0.88640526, -0.99396865,  0.43665827, -0.62890928],
[ 0.58024427, -0.88640526, -0.56336851,  0.43665827,  0.80748846],
[ 0.597565  ,  1.12815215, -1.3528021 ,  0.4748277 , -1.75473454],
[ 0.61488572, -0.88640526, -0.70690189,  0.4748277 ,  1.46745499],
[ 0.63220645, -0.88640526,  0.36959845,  0.4748277 , -1.67709142],
[ 0.64952717,  1.12815215, -0.49160182,  0.4748277 ,  0.88513158],
[ 0.6668479 ,  1.12815215, -1.42456879,  0.51299713, -1.56062674],
[ 0.68416862, -0.88640526, -0.27630176,  0.51299713,  0.84631002],
[ 0.70148935, -0.88640526,  1.30256542,  0.55116656, -1.75473454],
[ 0.71881007,  1.12815215, -0.49160182,  0.55116656,  1.6615628 ],
[ 0.73613079, -0.88640526, -0.77866858,  0.58933599, -0.39597992],
[ 0.75345152, -0.88640526, -0.49160182,  0.58933599,  1.42863343],
[ 0.77077224,  1.12815215, -0.99396865,  0.62750542, -1.48298362],
[ 0.78809297,  1.12815215, -0.77866858,  0.62750542,  1.81684904],
[ 0.80541369,  1.12815215,  0.65666521,  0.62750542, -0.55126616],
[ 0.82273442, -0.88640526, -0.49160182,  0.62750542,  0.92395314],
[ 0.84005514, -0.88640526, -0.34806844,  0.66567484, -1.09476801],
[ 0.85737587,  1.12815215, -0.34806844,  0.66567484,  1.54509812],
[ 0.87469659,  1.12815215,  0.29783176,  0.66567484, -1.28887582],
[ 0.89201732,  1.12815215,  0.010765  ,  0.66567484,  1.46745499],
[ 0.90933804, -0.88640526,  0.36959845,  0.66567484, -1.17241113],
[ 0.92665877, -0.88640526, -0.06100169,  0.66567484,  1.00159627],
[ 0.94397949, -0.88640526,  0.58489852,  0.66567484, -1.32769738],
[ 0.96130021, -0.88640526, -0.85043527,  0.66567484,  1.50627656],
[ 0.97862094,  1.12815215, -0.13276838,  0.66567484, -1.91002079],
[ 0.99594166, -0.88640526, -0.6351352 ,  0.66567484,  1.07923939],
[ 1.01326239,  1.12815215, -0.34806844,  0.66567484, -1.91002079],
[ 1.03058311, -0.88640526, -0.6351352 ,  0.66567484,  0.88513158],
[ 1.04790384, -0.88640526,  1.23079873,  0.70384427, -0.59008772],
[ 1.06522456, -0.88640526, -0.70690189,  0.70384427,  1.27334719],
[ 1.08254529,  1.12815215, -1.42456879,  0.78018313, -1.75473454],
[ 1.09986601, -0.88640526, -0.56336851,  0.78018313,  1.6615628 ],
[ 1.11718674,  1.12815215,  0.80019859,  0.93286085, -0.93948177],
[ 1.13450746, -0.88640526, -0.20453507,  0.93286085,  0.96277471],
[ 1.15182818,  1.12815215,  0.22606507,  0.97103028, -1.17241113],
[ 1.16914891, -0.88640526, -0.41983513,  0.97103028,  1.73920592],
[ 1.18646963, -0.88640526, -0.20453507,  1.00919971, -0.90066021],
[ 1.20379036,  1.12815215, -0.49160182,  1.00919971,  0.49691598],
[ 1.22111108,  1.12815215,  0.08253169,  1.00919971, -1.44416206],
[ 1.23843181,  1.12815215, -0.77866858,  1.00919971,  0.96277471],
[ 1.25575253,  1.12815215, -0.20453507,  1.00919971, -1.56062674],
[ 1.27307326,  1.12815215, -0.20453507,  1.00919971,  1.62274124],
[ 1.29039398, -0.88640526,  0.94373197,  1.04736914, -1.44416206],
```

```
       [ 1.30771471, -0.88640526, -0.6351352 ,  1.04736914,  1.38981187],
       [ 1.32503543,  1.12815215,  1.37433211,  1.04736914, -1.36651894],
       [ 1.34235616,  1.12815215, -0.85043527,  1.04736914,  0.72984534],
       [ 1.35967688,  1.12815215,  1.4460988 ,  1.23821628, -1.4053405 ],
       [ 1.3769976 ,  1.12815215, -0.27630176,  1.23821628,  1.54509812],
       [ 1.39431833, -0.88640526, -0.13276838,  1.390894  , -0.7065524 ],
       [ 1.41163905, -0.88640526, -0.49160182,  1.390894  ,  1.38981187],
       [ 1.42895978,  1.12815215,  0.51313183,  1.42906343, -1.36651894],
       [ 1.4462805 , -0.88640526, -0.70690189,  1.42906343,  1.46745499],
       [ 1.46360123, -0.88640526,  0.15429838,  1.46723286, -0.43480148],
       [ 1.48092195,  1.12815215, -0.6351352 ,  1.46723286,  1.81684904],
       [ 1.49824268, -0.88640526,  1.08726535,  1.54357172, -1.01712489],
       [ 1.5155634 ,  1.12815215, -0.77866858,  1.54357172,  0.69102378],
       [ 1.53288413, -0.88640526,  0.15429838,  1.61991057, -1.28887582],
       [ 1.55020485, -0.88640526, -0.20453507,  1.61991057,  1.35099031],
       [ 1.56752558, -0.88640526, -0.34806844,  1.61991057, -1.05594645],
       [ 1.5848463 , -0.88640526, -0.49160182,  1.61991057,  0.72984534],
       [ 1.60216702,  1.12815215, -0.41983513,  2.00160487, -1.63826986],
       [ 1.61948775, -0.88640526, -0.06100169,  2.00160487,  1.58391968],
       [ 1.63680847, -0.88640526,  0.58489852,  2.26879087, -1.32769738],
       [ 1.6541292 , -0.88640526, -0.27630176,  2.26879087,  1.11806095],
       [ 1.67144992, -0.88640526,  0.44136514,  2.49780745, -0.86183865],
       [ 1.68877065,  1.12815215, -0.49160182,  2.49780745,  0.92395314],
       [ 1.70609137,  1.12815215, -0.49160182,  2.91767117, -1.25005425],
       [ 1.7234121 ,  1.12815215, -0.6351352 ,  2.91767117,  1.27334719]])
```

**Question 9:**

**9 . Perform any of the clustering algorithms**

**9.1 Solution**

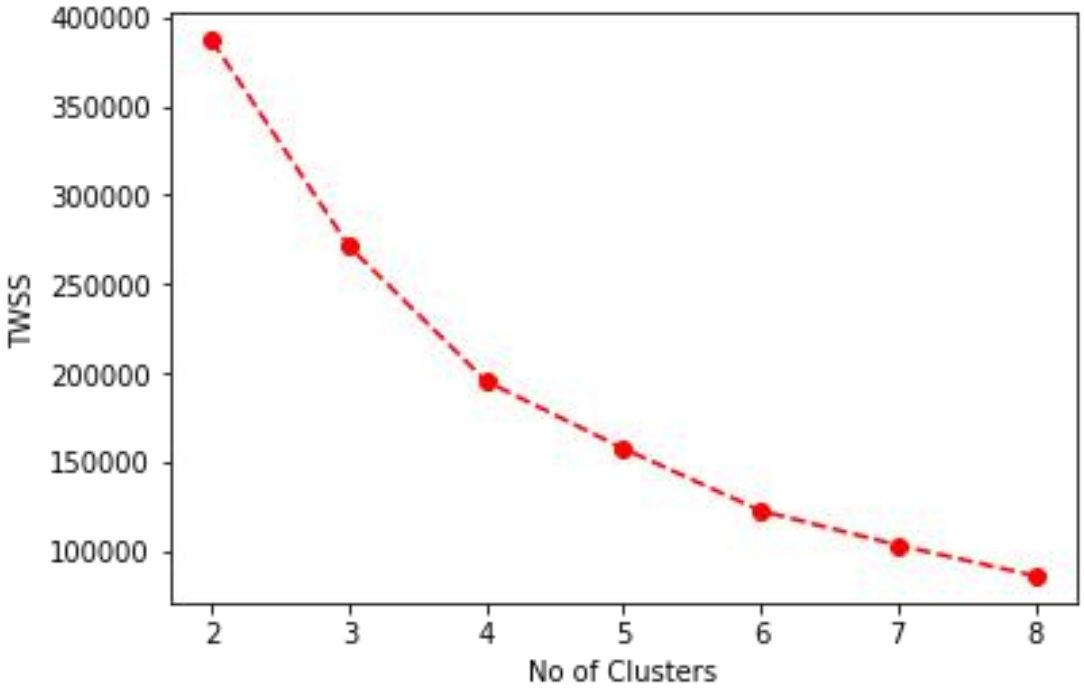
```
from sklearn.cluster import KMeans

TWSS = []
k = list(range(2,9))
for i in k:
  kmeans=KMeans(n_clusters=i, init='k-means++')
  kmeans.fit(db)
  TWSS.append(kmeans.inertia_)
TWSS
```

```
[387065.71377137717,
 271384.508782868,
 195401.19855991466,
 157620.9714797914,
 122625.19813553878,
 103244.10476410593,
 86053.67444777445]
```

**9.2 Solution**

```
plt.plot(k,TWSS, 'ro--')
plt.xlabel('No of Clusters')
plt.ylabel('TWSS')
```

```
Text(0, 0.5, 'TWSS')
```




### 9.3 Solution

```
model = KMeans(n_clusters = 4)
model.fit(db)
```

```
KMeans(n_clusters=4)
```

### 9.4 Solution

```
mb = pd.Series(model.labels_)
db['Cluster'] = mb
db
```

```
     CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)  \
0             1       1   19                  15                      39
1             2       1   21                  15                      81
2             3       0   20                  16                       6
3             4       0   23                  16                      77
4             5       0   31                  17                      40
..          ...     ...  ...                 ...                     ...
195         196       0   35                 120                      79
196         197       0   45                 126                      28
197         198       1   32                 126                      74
198         199       1   32                 137                      18
```

```
199         200      1   30                137                     83
```

```
     Cluster
0          3
1          3
2          3
3          3
4          3
..       ...
195        2
196        0
197        2
198        0
199        2

[200 rows x 6 columns]
```

### 9.5 Solution

```
mb=pd.Series(model.labels_)
db.head(3)
```

```
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)  \
0           1       1   19                  15                      39
1           2       1   21                  15                      81
2           3       0   20                  16                       6

   Cluster
0        3
1        3
2        3
```

## Question 10

### 10 . Add the cluster data with the primary dataset

### 10.1 Solution

```
db['Cluster']=kmeans.labels_
db.head()
```

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

   Cluster
0        6
1        5
```

```
2       6
3       5
4       6
```

### 10.2 Solution

```
db.tail()
```

```
     CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)  \
195         196       0   35                 120                      79
196         197       0   45                 126                      28
197         198       1   32                 126                      74
198         199       1   32                 137                      18
199         200       1   30                 137                      83

     Cluster
195        3
196        1
197        3
198        1
199        3
```

## Question 11

### 11 . Split the data into dependent and independent variables

### 11.1 Solution

```
X=db.drop('Cluster',axis=1)
Y=db['Cluster']
y=db['Cluster']
y
```

```
0      6
1      5
2      6
3      5
4      6
      ..
195    3
196    1
197    3
198    1
199    3
Name: Cluster, Length: 200, dtype: int32
```

### 11.2 Solution

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state
=42)
```

```
print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)
```

```
Number transactions X_train dataset:  (160, 5)
Number transactions y_train dataset:  (160,)
Number transactions X_test dataset:  (40, 5)
Number transactions y_test dataset:  (40,)
```

## Question 12

## 12 . Split the data into training and testing

### 12.1 Solution

```
X_train
```

| | CustomerID | Gender | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|---|
| 79 | 80 | 0 | 49 | 54 | 42 |
| 197 | 198 | 1 | 32 | 126 | 74 |
| 38 | 39 | 0 | 36 | 37 | 26 |
| 24 | 25 | 0 | 54 | 28 | 14 |
| 122 | 123 | 0 | 40 | 69 | 58 |
| .. | ... | ... | ... | ... | ... |
| 106 | 107 | 0 | 66 | 63 | 50 |
| 14 | 15 | 1 | 37 | 20 | 13 |
| 92 | 93 | 1 | 48 | 60 | 49 |
| 179 | 180 | 1 | 35 | 93 | 90 |
| 102 | 103 | 1 | 67 | 62 | 59 |

```
[160 rows x 5 columns]
```

### 12.2 Solution

```
X_test
```

| | CustomerID | Gender | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|---|
| 95 | 96 | 1 | 24 | 60 | 52 |
| 15 | 16 | 1 | 22 | 20 | 79 |
| 30 | 31 | 1 | 60 | 30 | 4 |
| 158 | 159 | 1 | 34 | 78 | 1 |
| 128 | 129 | 1 | 59 | 71 | 11 |
| 115 | 116 | 0 | 19 | 65 | 50 |
| 69 | 70 | 0 | 32 | 48 | 47 |
| 170 | 171 | 1 | 40 | 87 | 13 |
| 174 | 175 | 0 | 52 | 88 | 13 |
| 45 | 46 | 0 | 24 | 39 | 65 |
| 66 | 67 | 0 | 43 | 48 | 50 |
| 182 | 183 | 1 | 46 | 98 | 15 |
| 165 | 166 | 0 | 36 | 85 | 75 |
| 78 | 79 | 0 | 23 | 54 | 52 |

```
186        187       0   54                101                    24
177        178       1   27                 88                    69
56          57       0   51                 44                    50
152        153       0   44                 78                    20
82          83       1   67                 54                    41
68          69       1   19                 48                    59
124        125       0   23                 70                    29
16          17       0   35                 21                    35
148        149       0   34                 78                    22
93          94       0   40                 60                    40
65          66       1   18                 48                    59
60          61       1   70                 46                    56
84          85       0   21                 54                    57
67          68       0   68                 48                    48
125        126       0   31                 70                    77
132        133       0   25                 72                    34
9           10       0   30                 19                    72
18          19       1   52                 23                    29
55          56       1   47                 43                    41
75          76       1   26                 54                    54
150        151       1   43                 78                    17
104        105       1   49                 62                    56
135        136       0   29                 73                    88
137        138       1   32                 73                    73
164        165       1   50                 85                    26
76          77       0   45                 54                    53
```

### 12.3 Solution

```
y_train
```

```
79     6
197    2
38     4
24     4
122    7
      ..
106    7
14     4
92     7
179    2
102    7
Name: Cluster, Length: 160, dtype: int32
```

### 12.4 Solution

```
y_test
```

```
95     4
15     5
30     6
```

```
158    2
128    2
115    4
69     0
170    1
174    1
45     5
66     0
182    1
165    3
78     4
186    1
177    3
56     0
152    2
82     0
68     0
124    2
16     6
148    2
93     4
65     0
60     0
84     4
67     0
125    7
132    2
9      5
18     6
55     0
75     0
150    2
104    4
135    7
137    7
164    1
76     0
Name: Cluster, dtype: int32
```

### Question 13

### 13 . Build the Model

### 13. Solution

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818:
ConvergenceWarning: lbfgs failed to converge (status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,

LogisticRegression()
```

### Question 14

### 14. Train the Model

### 14. Solution

```
model.score(X_train,y_train)
```

```
0.85
```

### Question 15

### 15 . Test the Model

### 15. Solution

```
model.score(X_test,y_test)
```

```
0.725
```

### Question 16

### 16. Measure the performance using Evaluation Metrics

### 16. Solution

```
from sklearn.metrics import confusion_matrix,classification_report
y_pred=model.predict(X_test)
confusion_matrix(y_test,y_pred)
```

```
array([[8, 0, 0, 0, 3, 0, 0, 0],
       [0, 2, 3, 0, 0, 0, 0, 0],
       [0, 3, 4, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 1],
       [0, 0, 0, 1, 5, 0, 0, 0],
       [0, 0, 0, 0, 0, 3, 0, 0],
       [0, 0, 0, 0, 0, 0, 3, 0],
       [0, 0, 0, 0, 0, 0, 0, 3]])
```