**Assignment -4**
Python Programming

| Assignment Date | 25 October 2022 |
|---|---|
| Student Name | Suraj R |
| Student Roll Number | 621319106093 |
| Maximum Marks | 2 Marks |

**Question-1:**

AFTER DOWNLOADING THE DATASET ,IMPORT NECESSARY LIBRARIES

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

LOAD THE DATASET

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

**Solution:**



```
data.tail()
```

**Solution:**

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   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
```
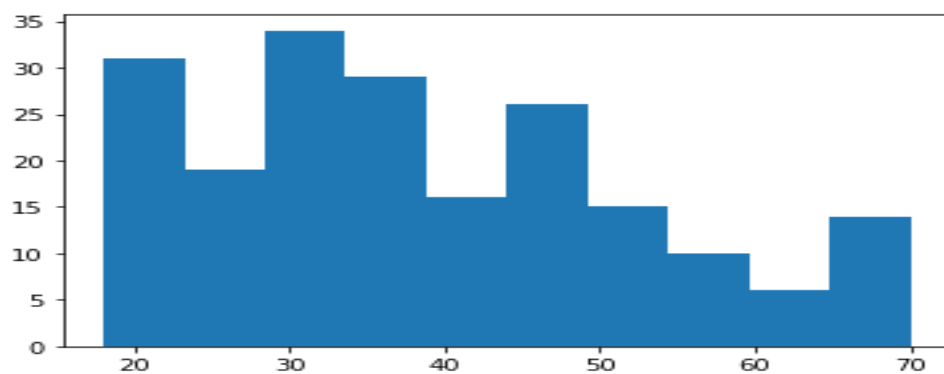
```
data.shape
```

**Solution:**
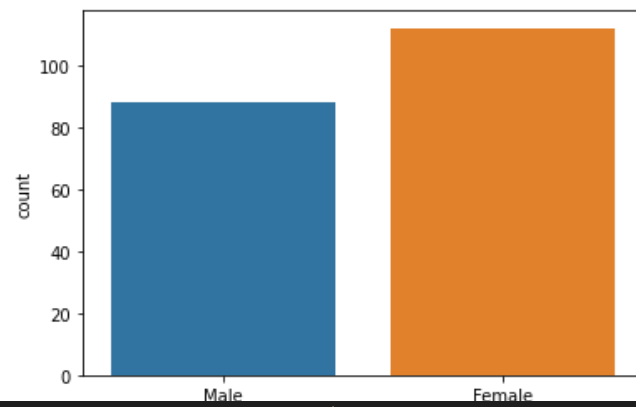
```
(200, 5)
```

**Question-2:**

```
#Univariate Analysis
plt.hist(data['Age'])
```

**Solution:**
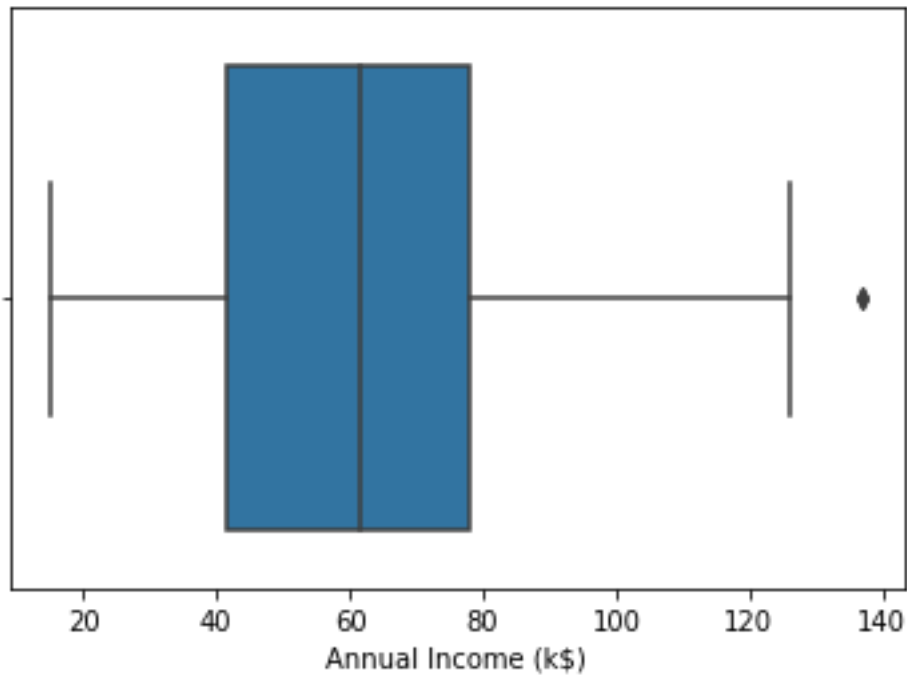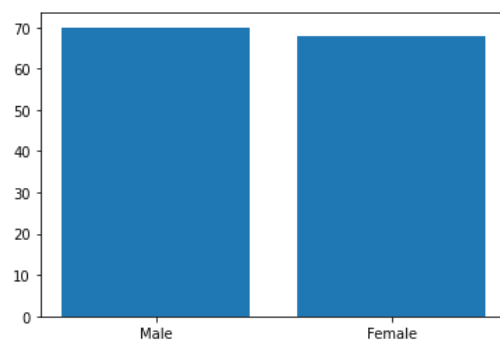
```
sns.countplot(data.Gender)
```



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

**Solution:**



```
#Bi- Variate Analysis
plt.bar(data['Gender'],data['Age'])
```

```
plt.scatter(data['Age'],data['CustomerID'])
```

**Solution:**
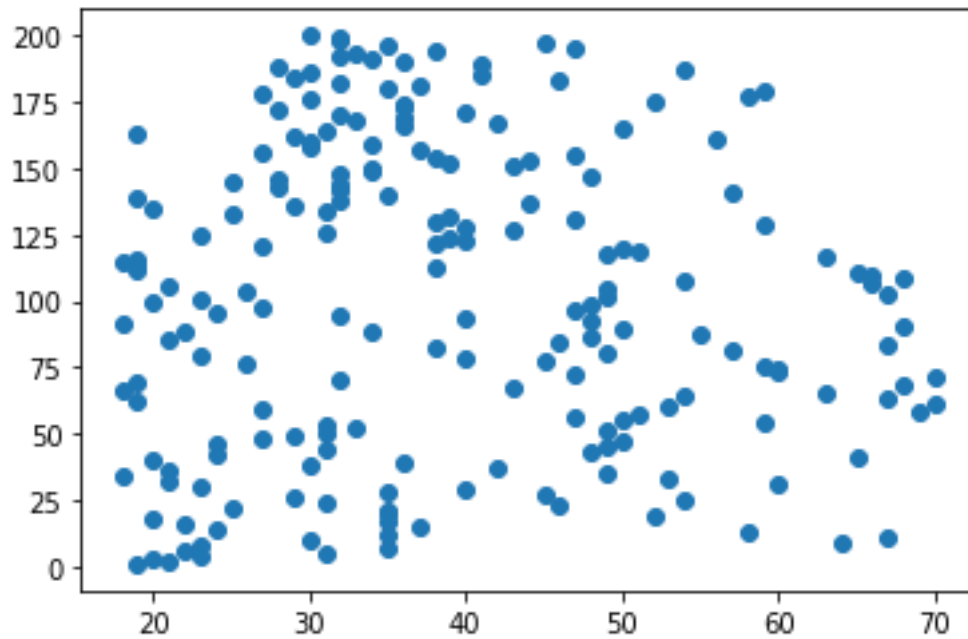


```
sns.barplot(x=data.Gender,y=data.Age)
```
**Solution:**
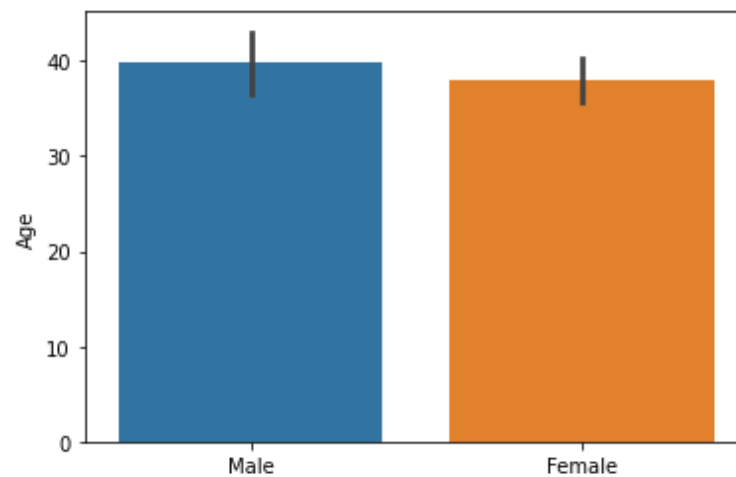


```
sns.boxplot(x=data.Gender,y=data.Age)
```
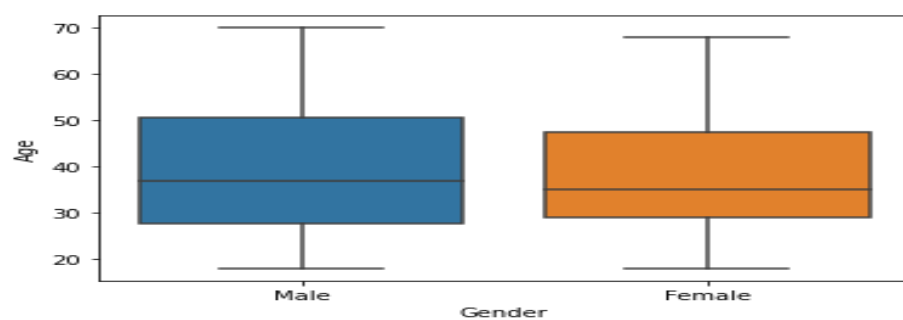
**Solution:**

```
sns.scatterplot(data.CustomerID,data.Age)
```

**Solution:**
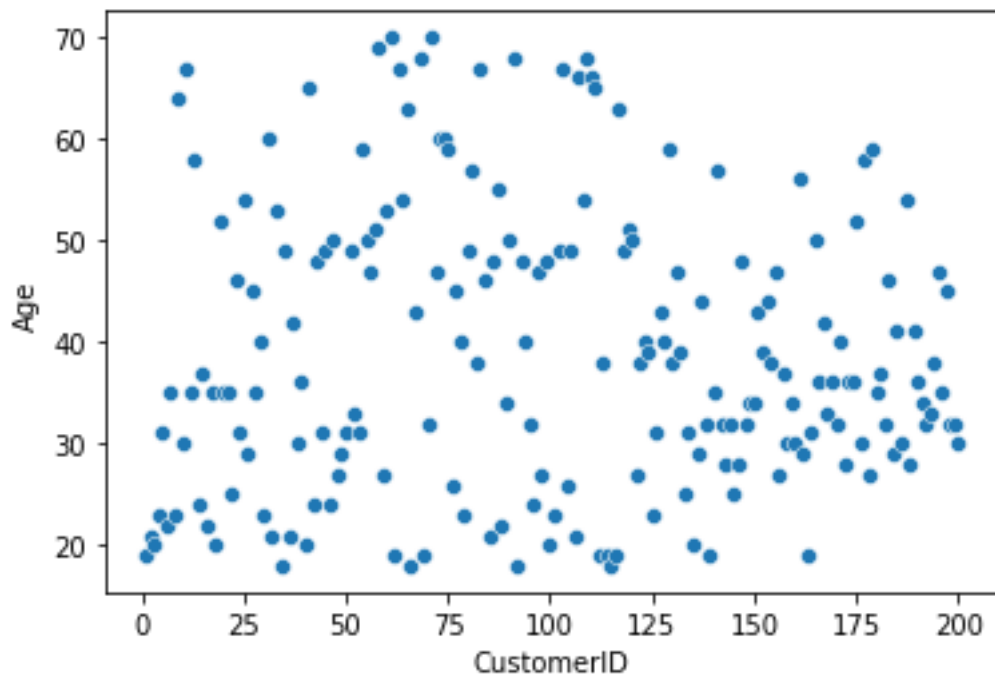


**Question-4:**

```python
#Multi-Variate Analysis
sns.set(font_scale=1.15)
plt.figure(figsize=(30,15))
sns.heatmap(data.corr(),cmap='RdYlGn_r',annot=True,).set_title('Multiva
riate analysis')
```

**Solution:**

```
sns.pairplot(data)
```

**Solution:**
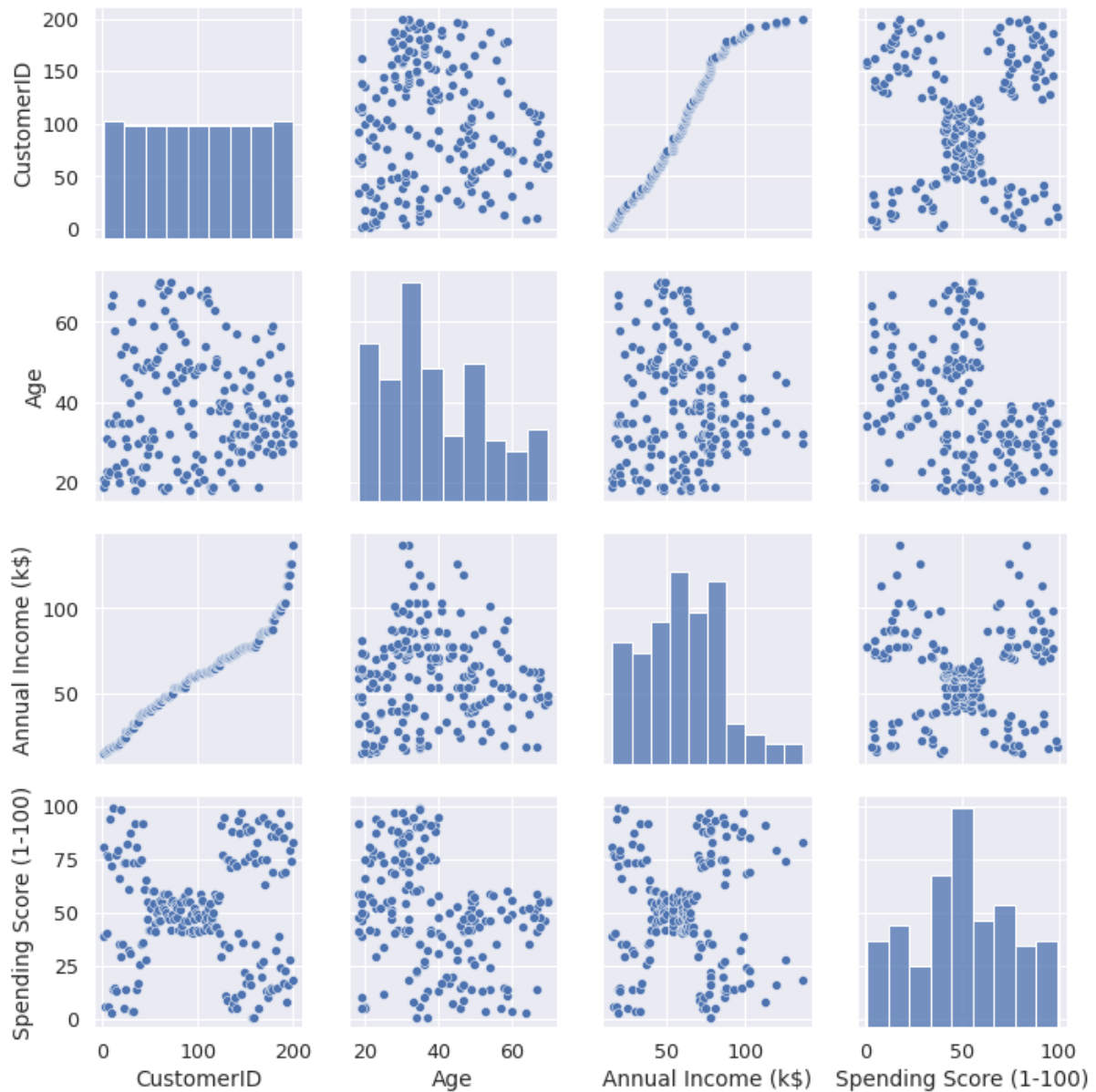


**Question-6:**

```
#Perform descriptive statistics on the dataset.
data.describe()
```

**Solution:**



| | 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 |

```
#mean
data['Age'].mean()
```

**Solution:**

```
38.85
```

```
#median
data.median()
```

**Solution:**

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

```
#mode
data.mode()
```

**Solution:**



```
#mode
data.mode()
```

| | 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

```
#std
data.std()
```

**Solution:**



```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will ra
CustomerID                57.879185
Age                       13.969007
Annual Income (k$)        26.264721
Spending Score (1-100)    25.823522
dtype: float64
```

```
#var
data.var()
```



```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will ra

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

```
#skewness
data.skew()
```
**Solution:**



```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will ra

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

```
print(sns.distplot(data['Age'])),print(sns.distplot(data['Age'],kde=True,))
```
**Solution:**
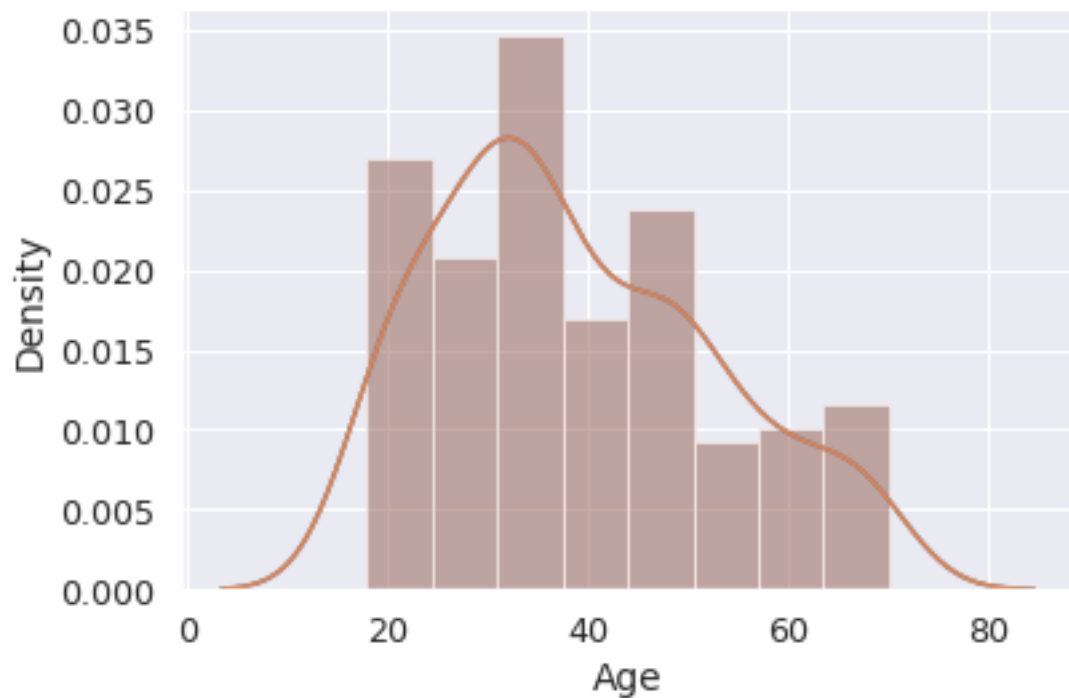


```
#Check for Missing values and deal with them.
data.isna().any().sum()
```

**Solution:**

```
0
```

```
# checking missing values we use isnull() function
data.isnull().any()
```

**Solution:**

```
[x]    CustomerID              False
       Gender                  False
       Age                     False
       Annual Income (k$)      False
       Spending Score (1-100)  False
       dtype: bool
```

```
#check sum of all null values
data.isnull().sum()
```

**Solution:**

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

**Question-9:**

```
#Find the outliers and replace them outliers
qnt=data.quantile(q=(0.09,1.00))
qnt
```

**Solution:**

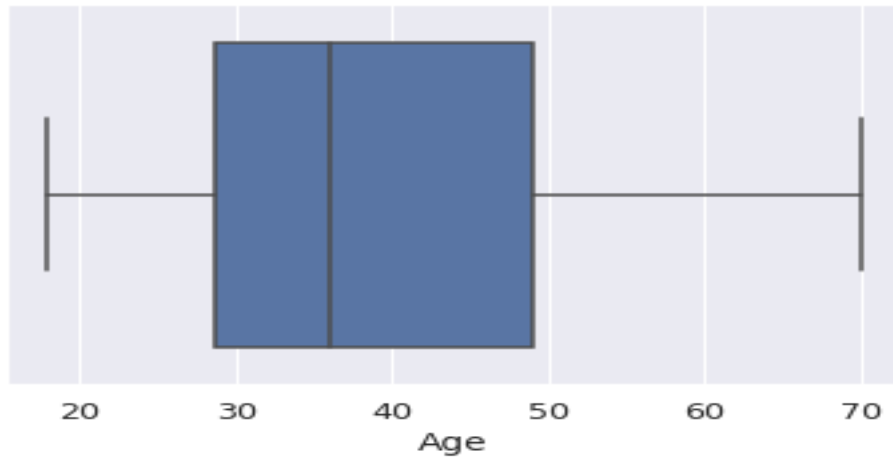| | CustomerID | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| 0.09 | 18.91 | 21.0 | 22.82 | 12.91 |
| 1.00 | 200.00 | 70.0 | 137.00 | 99.00 |

```
Q1 = data.Age.quantile(0.25)
Q3 = data.Age.quantile(0.175)
IQR = Q3 - Q1
lower_limit = Q1 - 1.5 * IQR
data.median(numeric_only=True)
```

**Solution:**

```
CustomerID              100.5
Age                     36.0
Annual Income (k$)      61.5
Spending Score (1-100)  50.0
dtype: float64
```
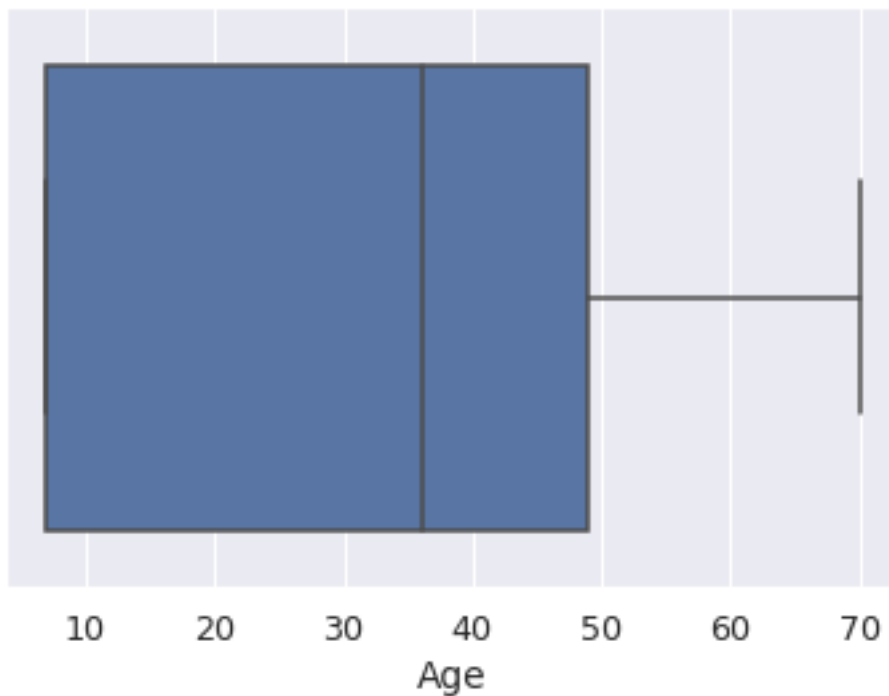
```
sns.boxplot(x=data.Age)
```

**Solution:**



```
data['Age'] = np.where(data['Age'] < lower_limit, 7, data['Age'])
sns.boxplot(x=data.Age,showfliers = False)
```

**Solution:**



```
#Check for Categorical columns and perform encoding.
data.head(2)
```

**Solution:**

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

**Solution:**



```
#Scaling the data
x=data
names=x.columns
names
```

**Solution:**

Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',

  'Spending Score (1-100)'],

  dtype='object')

```
from sklearn.preprocessing import scale
x=scale(x)
x
```

**Solution:**

```
array([[-1.7234121 ,  1.12815215, -1.04704552, -1.73899919, -
0.43480148], [-1.70609137,  1.12815215, -1.04704552, -1.73899919,
1.19570407], [-1.68877065, -0.88640526, -1.04704552, -1.70082976, -
1.71591298], [-1.67144992, -0.88640526, -1.04704552, -1.70082976,
1.04041783], [-1.6541292 , -0.88640526, -1.04704552, -1.66266033, -
0.39597992], [-1.63680847, -0.88640526, -1.04704552, -1.66266033,
1.00159627], [-1.61948775, -0.88640526,  0.21690251, -1.62449091, -
1.71591298], [-1.60216702, -0.88640526, -1.04704552, -1.62449091,
1.70038436], [-1.5848463 ,  1.12815215,  1.52599154, -1.58632148, -
1.83237767], [-1.56752558, -0.88640526, -1.04704552, -1.58632148,
0.84631002], [-1.55020485,  1.12815215,  1.66141454, -1.58632148, -
1.4053405 ], [-1.53288413, -0.88640526,  0.21690251, -1.58632148,
1.89449216], [-1.5155634 , -0.88640526,  1.25514553, -1.54815205, -
```

1.36651894], [-1.49824268, -0.88640526, -1.04704552, -1.54815205,
1.04041783], [-1.48092195, 1.12815215, 0.30718451, -1.54815205, -
1.44416206], [-1.46360123, 1.12815215, -1.04704552, -1.54815205,
1.11806095], [-1.4462805 , -0.88640526, 0.21690251, -1.50998262, -
0.59008772], [-1.42895978, 1.12815215, -1.04704552, -1.50998262,
0.61338066], [-1.41163905, 1.12815215, 0.98429952, -1.43364376, -
0.82301709], [-1.39431833, -0.88640526, 0.21690251, -1.43364376,
1.8556706 ], [-1.3769976 , 1.12815215, 0.21690251, -1.39547433, -
0.59008772], [-1.35967688, 1.12815215, -1.04704552, -1.39547433,
0.88513158], [-1.34235616, -0.88640526, 0.71345352, -1.3573049 , -
1.75473454], [-1.32503543, 1.12815215, -1.04704552, -1.3573049 ,
0.88513158], [-1.30771471, -0.88640526, 1.07458153, -1.24279661, -
1.4053405 ], [-1.29039398, 1.12815215, -1.04704552, -1.24279661,
1.23452563], [-1.27307326, -0.88640526, 0.66831252, -1.24279661, -
0.7065524 ], [-1.25575253, 1.12815215, 0.21690251, -1.24279661,
0.41927286], [-1.23843181, -0.88640526, 0.44260751, -1.20462718, -
0.74537397], [-1.22111108, -0.88640526, -1.04704552, -1.20462718,
1.42863343], [-1.20379036, 1.12815215, 1.34542753, -1.16645776, -
1.7935561 ], [-1.18646963, -0.88640526, -1.04704552, -1.16645776,
0.88513158], [-1.16914891, 1.12815215, 1.02944053, -1.05194947, -
1.7935561 ], [-1.15182818, 1.12815215, -1.04704552, -1.05194947,
1.62274124], [-1.13450746, -0.88640526, 0.84887652, -1.05194947, -
1.4053405 ], [-1.11718674, -0.88640526, -1.04704552, -1.05194947,
1.19570407], [-1.09986601, -0.88640526, 0.53288952, -1.01378004, -
1.28887582], [-1.08254529, -0.88640526, -1.04704552, -1.01378004,
0.88513158], [-1.06522456, -0.88640526, 0.26204351, -0.89927175, -
0.93948177], [-1.04790384, -0.88640526, -1.04704552, -0.89927175,
0.96277471], [-1.03058311, -0.88640526, 1.57113254, -0.86110232, -
0.59008772], [-1.01326239, 1.12815215, -1.04704552, -0.86110232,
1.62274124], [-0.99594166, 1.12815215, 0.80373552, -0.82293289, -
0.55126616], [-0.97862094, -0.88640526, -1.04704552, -0.82293289,
0.41927286], [-0.96130021, -0.88640526, 0.84887652, -0.82293289, -
0.86183865], [-0.94397949, -0.88640526, -1.04704552, -0.82293289,
0.5745591 ], [-0.92665877, -0.88640526, 0.89401752, -0.78476346,
0.18634349], [-0.90933804, -0.88640526, -1.04704552, -0.78476346, -
0.12422899], [-0.89201732, -0.88640526, -1.04704552, -0.78476346, -
0.3183368 ], [-0.87469659, -0.88640526, -1.04704552, -0.78476346, -
0.3183368 ], [-0.85737587, -0.88640526, 0.84887652, -0.70842461,
0.06987881], [-0.84005514, 1.12815215, -1.04704552, -0.70842461,
0.38045129], [-0.82273442, -0.88640526, -1.04704552, -0.67025518,
0.14752193], [-0.80541369, 1.12815215, 1.30028653, -0.67025518,
0.38045129], [-0.78809297, -0.88640526, 0.89401752, -0.67025518, -
0.20187212], [-0.77077224, 1.12815215, 0.75859452, -0.67025518, -
0.35715836], [-0.75345152, -0.88640526, 0.93915852, -0.63208575, -
0.00776431], [-0.73613079, 1.12815215, 1.75169654, -0.63208575, -
0.16305055], [-0.71881007, -0.88640526, -1.04704552, -0.55574689,
0.03105725], [-0.70148935, 1.12815215, 1.02944053, -0.55574689, -
0.16305055], [-0.68416862, 1.12815215, 1.79683754, -0.55574689,
0.22516505], [-0.6668479 , 1.12815215, -1.04704552, -0.55574689,
0.18634349], [-0.64952717, -0.88640526, 1.66141454, -0.51757746,
0.06987881], [-0.63220645, -0.88640526, 1.07458153, -0.51757746,
0.34162973], [-0.61488572, 1.12815215, 1.48085053, -0.47940803,
0.03105725], [-0.597565 , 1.12815215, -1.04704552, -0.47940803,
0.34162973], [-0.58024427, -0.88640526, 0.57803052, -0.47940803, -

0.00776431], [-0.56292355, -0.88640526, 1.70655554, -0.47940803, -
0.08540743], [-0.54560282, 1.12815215, -1.04704552, -0.47940803,
0.34162973], [-0.5282821 , -0.88640526, -1.04704552, -0.47940803, -
0.12422899], [-0.51096138, 1.12815215, 1.79683754, -0.4412386 ,
0.18634349], [-0.49364065, -0.88640526, 0.75859452, -0.4412386 , -
0.3183368 ], [-0.47631993, -0.88640526, 1.34542753, -0.40306917, -
0.04658587], [-0.4589992 , -0.88640526, 1.34542753, -0.40306917,
0.22516505], [-0.44167848, 1.12815215, 1.30028653, -0.25039146, -
0.12422899], [-0.42435775, 1.12815215, -1.04704552, -0.25039146,
0.14752193], [-0.40703703, -0.88640526, 0.66831252, -0.25039146,
0.10870037], [-0.3897163 , 1.12815215, 0.44260751, -0.25039146, -
0.08540743], [-0.37239558, -0.88640526, -1.04704552, -0.25039146,
0.06987881], [-0.35507485, -0.88640526, 0.84887652, -0.25039146, -
0.3183368 ], [-0.33775413, 1.12815215, 1.21000453, -0.25039146,
0.03105725], [-0.3204334 , 1.12815215, 0.35232551, -0.25039146,
0.18634349], [-0.30311268, 1.12815215, 1.66141454, -0.25039146, -
0.35715836], [-0.28579196, -0.88640526, 0.71345352, -0.25039146, -
0.24069368], [-0.26847123, -0.88640526, -1.04704552, -0.25039146,
0.26398661], [-0.25115051, 1.12815215, 0.80373552, -0.25039146, -
0.16305055], [-0.23382978, -0.88640526, 1.11972253, -0.13588317,
0.30280817], [-0.21650906, -0.88640526, -1.04704552, -0.13588317,
0.18634349], [-0.19918833, -0.88640526, -1.04704552, -0.09771374,
0.38045129], [-0.18186761, -0.88640526, 0.89401752, -0.09771374, -
0.16305055], [-0.16454688, -0.88640526, 1.70655554, -0.05954431,
0.18634349], [-0.14722616, 1.12815215, -1.04704552, -0.05954431, -
0.35715836], [-0.12990543, 1.12815215, 0.80373552, -0.02137488, -
0.04658587], [-0.11258471, -0.88640526, 0.44260751, -0.02137488, -
0.39597992], [-0.09526399, -0.88640526, -1.04704552, -0.02137488, -
0.3183368 ], [-0.07794326, 1.12815215, -1.04704552, -0.02137488,
0.06987881], [-0.06062254, -0.88640526, 0.75859452, -0.02137488, -
0.12422899], [-0.04330181, -0.88640526, -1.04704552, -0.02137488, -
0.00776431], [-0.02598109, 1.12815215, 0.80373552, 0.01679455, -
0.3183368 ], [-0.00866036, 1.12815215, -1.04704552, 0.01679455, -
0.04658587], [ 0.00866036, -0.88640526, -1.04704552, 0.05496398, -
0.35715836], [ 0.02598109, -0.88640526, 0.84887652, 0.05496398, -
0.08540743], [ 0.04330181, 1.12815215, 1.66141454, 0.05496398,
0.34162973], [ 0.06062254, 1.12815215, -1.04704552, 0.05496398,
0.18634349], [ 0.07794326, 1.12815215, 0.84887652, 0.05496398,
0.22516505], [ 0.09526399, -0.88640526, -1.04704552, 0.05496398, -
0.3183368 ], [ 0.11258471, -0.88640526, 1.61627354, 0.09313341, -
0.00776431], [ 0.12990543, 1.12815215, 1.07458153, 0.09313341, -
0.16305055], [ 0.14722616, 1.12815215, 1.70655554, 0.09313341, -
0.27951524], [ 0.16454688, 1.12815215, 1.61627354, 0.09313341, -
0.08540743], [ 0.18186761, 1.12815215, 1.57113254, 0.09313341,
0.06987881], [ 0.19918833, -0.88640526, -1.04704552, 0.09313341,
0.14752193], [ 0.21650906, -0.88640526, 0.35232551, 0.13130284, -
0.3183368 ], [ 0.23382978, 1.12815215, -1.04704552, 0.13130284, -
0.16305055], [ 0.25115051, -0.88640526, -1.04704552, 0.16947227, -
0.08540743], [ 0.26847123, -0.88640526, -1.04704552, 0.16947227, -
0.00776431], [ 0.28579196, -0.88640526, 1.48085053, 0.16947227, -
0.27951524], [ 0.30311268, -0.88640526, 0.84887652, 0.16947227,
0.34162973], [ 0.3204334 , -0.88640526, 0.93915852, 0.24581112, -
0.27951524], [ 0.33775413, -0.88640526, 0.89401752, 0.24581112,
0.26398661], [ 0.35507485, 1.12815215, -1.04704552, 0.24581112,

0.22516505], [ 0.37239558, -0.88640526, 0.35232551, 0.24581112, -
0.39597992], [ 0.3897163 , -0.88640526, 0.44260751, 0.32214998,
0.30280817], [ 0.40703703, 1.12815215, 0.39746651, 0.32214998,
1.58391968], [ 0.42435775, -0.88640526, -1.04704552, 0.36031941, -
0.82301709], [ 0.44167848, -0.88640526, -1.04704552, 0.36031941,
1.04041783], [ 0.4589992 , 1.12815215, 0.57803052, 0.39848884, -
0.59008772], [ 0.47631993, 1.12815215, 0.44260751, 0.39848884,
1.73920592], [ 0.49364065, 1.12815215, 1.30028653, 0.39848884, -
1.52180518], [ 0.51096138, 1.12815215, 0.35232551, 0.39848884,
0.96277471], [ 0.5282821 , 1.12815215, 0.75859452, 0.39848884, -
1.5994483 ], [ 0.54560282, 1.12815215, 0.39746651, 0.39848884,
0.96277471], [ 0.56292355, -0.88640526, -1.04704552, 0.43665827, -
0.62890928], [ 0.58024427, -0.88640526, -1.04704552, 0.43665827,
0.80748846], [ 0.597565 , 1.12815215, -1.04704552, 0.4748277 , -
1.75473454], [ 0.61488572, -0.88640526, -1.04704552, 0.4748277 ,
1.46745499], [ 0.63220645, -0.88640526, 0.62317152, 0.4748277 , -
1.67709142], [ 0.64952717, 1.12815215, -1.04704552, 0.4748277 ,
0.88513158], [ 0.6668479 , 1.12815215, -1.04704552, 0.51299713, -
1.56062674], [ 0.68416862, -0.88640526, 0.21690251, 0.51299713,
0.84631002], [ 0.70148935, -0.88640526, 1.21000453, 0.55116656, -
1.75473454], [ 0.71881007, 1.12815215, -1.04704552, 0.55116656,
1.6615628 ], [ 0.73613079, -0.88640526, -1.04704552, 0.58933599, -
0.39597992], [ 0.75345152, -0.88640526, -1.04704552, 0.58933599,
1.42863343], [ 0.77077224, 1.12815215, -1.04704552, 0.62750542, -
1.48298362], [ 0.78809297, 1.12815215, -1.04704552, 0.62750542,
1.81684904], [ 0.80541369, 1.12815215, 0.80373552, 0.62750542, -
0.55126616], [ 0.82273442, -0.88640526, -1.04704552, 0.62750542,
0.92395314], [ 0.84005514, -0.88640526, -1.04704552, 0.66567484, -
1.09476801], [ 0.85737587, 1.12815215, -1.04704552, 0.66567484,
1.54509812], [ 0.87469659, 1.12815215, 0.57803052, 0.66567484, -
1.28887582], [ 0.89201732, 1.12815215, 0.39746651, 0.66567484,
1.46745499], [ 0.90933804, -0.88640526, 0.62317152, 0.66567484, -
1.17241113], [ 0.92665877, -0.88640526, 0.35232551, 0.66567484,
1.00159627], [ 0.94397949, -0.88640526, 0.75859452, 0.66567484, -
1.32769738], [ 0.96130021, -0.88640526, -1.04704552, 0.66567484,
1.50627656], [ 0.97862094, 1.12815215, 0.30718451, 0.66567484, -
1.91002079], [ 0.99594166, -0.88640526, -1.04704552, 0.66567484,
1.07923939], [ 1.01326239, 1.12815215, -1.04704552, 0.66567484, -
1.91002079], [ 1.03058311, -0.88640526, -1.04704552, 0.66567484,
0.88513158], [ 1.04790384, -0.88640526, 1.16486353, 0.70384427, -
0.59008772], [ 1.06522456, -0.88640526, -1.04704552, 0.70384427,
1.27334719], [ 1.08254529, 1.12815215, -1.04704552, 0.78018313, -
1.75473454], [ 1.09986601, -0.88640526, -1.04704552, 0.78018313,
1.6615628 ], [ 1.11718674, 1.12815215, 0.89401752, 0.93286085, -
0.93948177], [ 1.13450746, -0.88640526, 0.26204351, 0.93286085,
0.96277471], [ 1.15182818, 1.12815215, 0.53288952, 0.97103028, -
1.17241113], [ 1.16914891, -0.88640526, -1.04704552, 0.97103028,
1.73920592], [ 1.18646963, -0.88640526, 0.26204351, 1.00919971, -
0.90066021], [ 1.20379036, 1.12815215, -1.04704552, 1.00919971,
0.49691598], [ 1.22111108, 1.12815215, 0.44260751, 1.00919971, -
1.44416206], [ 1.23843181, 1.12815215, -1.04704552, 1.00919971,
0.96277471], [ 1.25575253, 1.12815215, 0.26204351, 1.00919971, -
1.56062674], [ 1.27307326, 1.12815215, 0.26204351, 1.00919971,
1.62274124], [ 1.29039398, -0.88640526, 0.98429952, 1.04736914, -

```
1.44416206], [ 1.30771471, -0.88640526, -1.04704552, 1.04736914,
1.38981187], [ 1.32503543, 1.12815215, 1.25514553, 1.04736914, -
1.36651894], [ 1.34235616, 1.12815215, -1.04704552, 1.04736914,
0.72984534], [ 1.35967688, 1.12815215, 1.30028653, 1.23821628, -
1.4053405 ], [ 1.3769976 , 1.12815215, 0.21690251, 1.23821628,
1.54509812], [ 1.39431833, -0.88640526, 0.30718451, 1.390894 , -
0.7065524 ], [ 1.41163905, -0.88640526, -1.04704552, 1.390894 ,
1.38981187], [ 1.42895978, 1.12815215, 0.71345352, 1.42906343, -
1.36651894], [ 1.4462805 , -0.88640526, -1.04704552, 1.42906343,
1.46745499], [ 1.46360123, -0.88640526, 0.48774851, 1.46723286, -
0.43480148], [ 1.48092195, 1.12815215, -1.04704552, 1.46723286,
1.81684904], [ 1.49824268, -0.88640526, 1.07458153, 1.54357172, -
1.01712489], [ 1.5155634 , 1.12815215, -1.04704552, 1.54357172,
0.69102378], [ 1.53288413, -0.88640526, 0.48774851, 1.61991057, -
1.28887582], [ 1.55020485, -0.88640526, 0.26204351, 1.61991057,
1.35099031], [ 1.56752558, -0.88640526, -1.04704552, 1.61991057, -
1.05594645], [ 1.5848463 , -0.88640526, -1.04704552, 1.61991057,
0.72984534], [ 1.60216702, 1.12815215, -1.04704552, 2.00160487, -
1.63826986], [ 1.61948775, -0.88640526, 0.35232551, 2.00160487,
1.58391968], [ 1.63680847, -0.88640526, 0.75859452, 2.26879087, -
1.32769738], [ 1.6541292 , -0.88640526, 0.21690251, 2.26879087,
1.11806095], [ 1.67144992, -0.88640526, 0.66831252, 2.49780745, -
0.86183865], [ 1.68877065, 1.12815215, -1.04704552, 2.49780745,
0.92395314], [ 1.70609137, 1.12815215, -1.04704552, 2.91767117, -
1.25005425], [ 1.7234121 , 1.12815215, -1.04704552, 2.91767117,
1.27334719]])
```

```
x=pd.DataFrame(x,columns=names)
x
```
**Solution:**

| | CustomerID | Gender | Age | Annual Income (k$) | Spending Score (1-100) | |
|---|---|---|---|---|---|---|
| 0 | -1.723412 | 1.128152 | -1.047046 | -1.738999 | -0.434801 | |
| 1 | -1.706091 | 1.128152 | -1.047046 | -1.738999 | 1.195704 | |
| 2 | -1.688771 | -0.886405 | -1.047046 | -1.700830 | -1.715913 | |
| 3 | -1.671450 | -0.886405 | -1.047046 | -1.700830 | 1.040418 | |
| 4 | -1.654129 | -0.886405 | -1.047046 | -1.662660 | -0.395980 | |
| ... | ... | ... | ... | ... | ... | |
| 195 | 1.654129 | -0.886405 | 0.216903 | 2.268791 | 1.118061 | |
| 196 | 1.671450 | -0.886405 | 0.668313 | 2.497807 | -0.861839 | |
| 197 | 1.688771 | 1.128152 | -1.047046 | 2.497807 | 0.923953 | |
| 198 | 1.706091 | 1.128152 | -1.047046 | 2.917671 | -1.250054 | |
| 199 | 1.723412 | 1.128152 | -1.047046 | 2.917671 | 1.273347 | |

200 rows × 5 columns

```
#Perform any of the clustering algorithms
#Kmeans clustering
from sklearn.cluster import KMeans
PJAA=[]
k=list(range(2,9))


for i in k:
    kmeans=KMeans(n_clusters=i,init='k-means++')
```
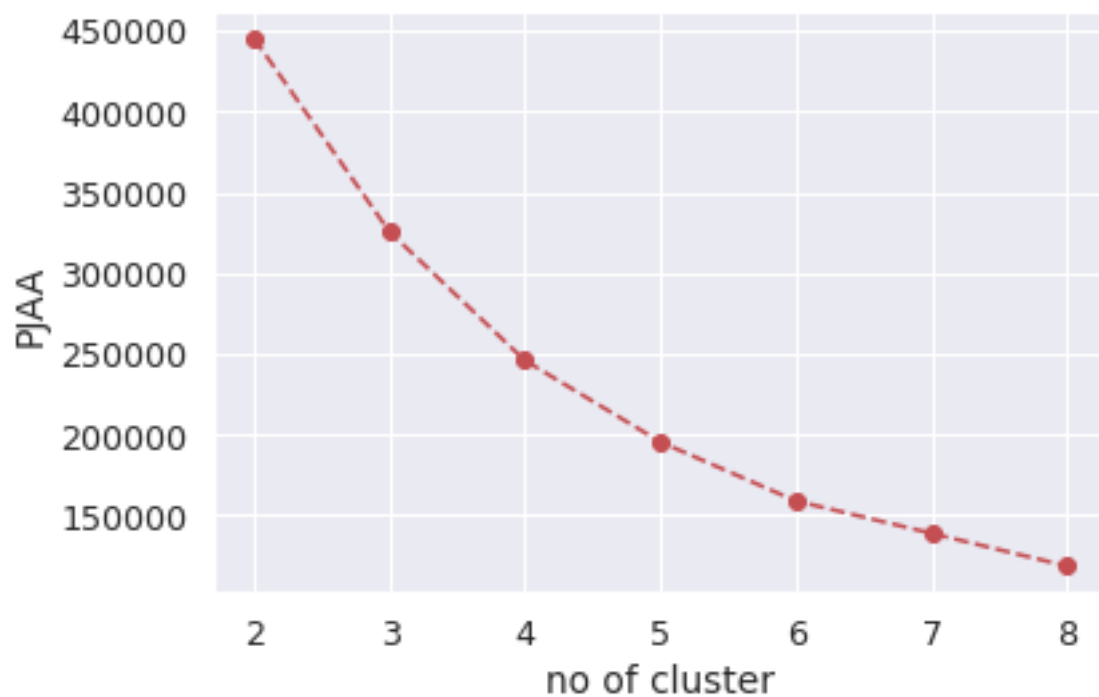
```
    kmeans.fit(data)
    PJAA.append(kmeans.inertia_)
PJAA
```

**Solution:**

```
[445698.6278627863, 325787.09074274875, 245928.9448954999,
195438.05301054876, 158881.73689847524, 138853.0978883425,
118523.53235349475]
```

```
plt.plot(k,PJAA,'ro--')
plt.xlabel('no of cluster')
plt.ylabel('PJAA')
```

**Solution:**



```
#Add the cluster data with the primary dataset
model.labels_
```

**Solution:**

```
array([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, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3,
2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3,
2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3,
2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2], dtype=int32)
```

```
data.to_csv('Kmeans_Mall_Customers.csv',encoding='utf-8')
#Split the data into dependent and independent variables.
X=data.iloc[:,:-1].values
y=data.iloc[:,-1].values
X
```
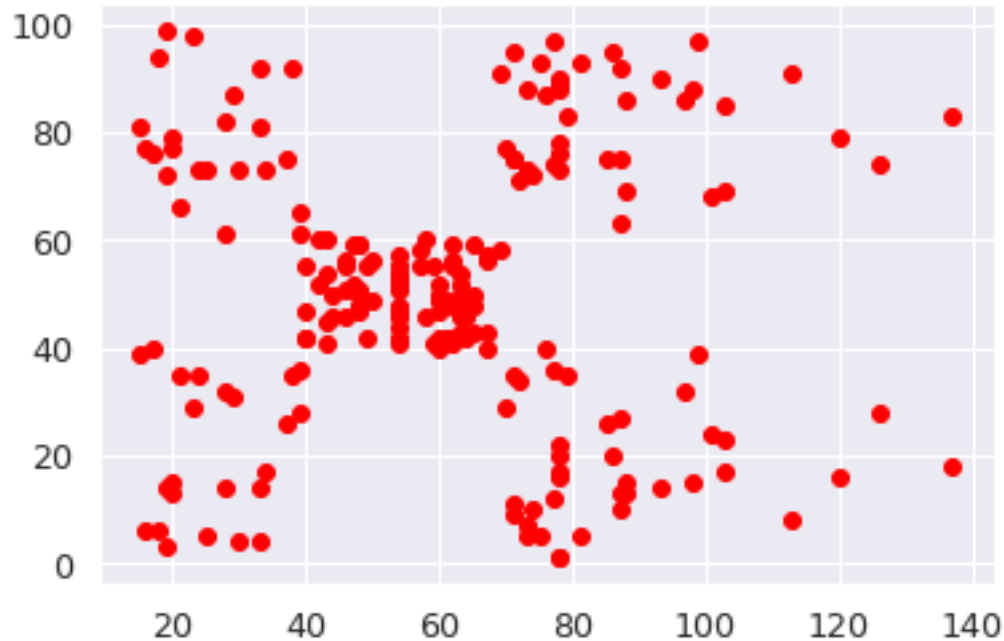
Solution:

```
array([[ 1, 1, 7, 15], [ 2, 1, 7, 15], [ 3, 0, 7, 16], [ 4, 0, 7, 16],
[ 5, 0, 7, 17], [ 6, 0, 7, 17], [ 7, 0, 35, 18], [ 8, 0, 7, 18], [ 9,
1, 64, 19], [ 10, 0, 7, 19], [ 11, 1, 67, 19], [ 12, 0, 35, 19], [ 13,
0, 58, 20], [ 14, 0, 7, 20], [ 15, 1, 37, 20], [ 16, 1, 7, 20], [ 17,
0, 35, 21], [ 18, 1, 7, 21], [ 19, 1, 52, 23], [ 20, 0, 35, 23], [ 21,
1, 35, 24], [ 22, 1, 7, 24], [ 23, 0, 46, 25], [ 24, 1, 7, 25], [ 25,
0, 54, 28], [ 26, 1, 7, 28], [ 27, 0, 45, 28], [ 28, 1, 35, 28], [ 29,
0, 40, 29], [ 30, 0, 7, 29], [ 31, 1, 60, 30], [ 32, 0, 7, 30], [ 33,
1, 53, 33], [ 34, 1, 7, 33], [ 35, 0, 49, 33], [ 36, 0, 7, 33], [ 37,
0, 42, 34], [ 38, 0, 7, 34], [ 39, 0, 36, 37], [ 40, 0, 7, 37], [ 41,
0, 65, 38], [ 42, 1, 7, 38], [ 43, 1, 48, 39], [ 44, 0, 7, 39], [ 45,
0, 49, 39], [ 46, 0, 7, 39], [ 47, 0, 50, 40], [ 48, 0, 7, 40], [ 49,
0, 7, 40], [ 50, 0, 7, 40], [ 51, 0, 49, 42], [ 52, 1, 7, 42], [ 53, 0,
7, 43], [ 54, 1, 59, 43], [ 55, 0, 50, 43], [ 56, 1, 47, 43], [ 57, 0,
51, 44], [ 58, 1, 69, 44], [ 59, 0, 7, 46], [ 60, 1, 53, 46], [ 61, 1,
70, 46], [ 62, 1, 7, 46], [ 63, 0, 67, 47], [ 64, 0, 54, 47], [ 65, 1,
63, 48], [ 66, 1, 7, 48], [ 67, 0, 43, 48], [ 68, 0, 68, 48], [ 69, 1,
7, 48], [ 70, 0, 7, 48], [ 71, 1, 70, 49], [ 72, 0, 47, 49], [ 73, 0,
60, 50], [ 74, 0, 60, 50], [ 75, 1, 59, 54], [ 76, 1, 7, 54], [ 77, 0,
45, 54], [ 78, 1, 40, 54], [ 79, 0, 7, 54], [ 80, 0, 49, 54], [ 81, 1,
57, 54], [ 82, 1, 38, 54], [ 83, 1, 67, 54], [ 84, 0, 46, 54], [ 85, 0,
7, 54], [ 86, 1, 48, 54], [ 87, 0, 55, 57], [ 88, 0, 7, 57], [ 89, 0,
7, 58], [ 90, 0, 50, 58], [ 91, 0, 68, 59], [ 92, 1, 7, 59], [ 93, 1,
48, 60], [ 94, 0, 40, 60], [ 95, 0, 7, 60], [ 96, 1, 7, 60], [ 97, 0,
47, 60], [ 98, 0, 7, 60], [ 99, 1, 48, 61], [100, 1, 7, 61], [101, 0,
7, 62], [102, 0, 49, 62], [103, 1, 67, 62], [104, 1, 7, 62], [105, 1,
49, 62], [106, 0, 7, 62], [107, 0, 66, 63], [108, 1, 54, 63], [109, 1,
68, 63], [110, 1, 66, 63], [111, 1, 65, 63], [112, 0, 7, 63], [113, 0,
38, 64], [114, 1, 7, 64], [115, 0, 7, 65], [116, 0, 7, 65], [117, 0,
63, 65], [118, 0, 49, 65], [119, 0, 51, 67], [120, 0, 50, 67], [121, 1,
7, 67], [122, 0, 38, 67], [123, 0, 40, 69], [124, 1, 39, 69], [125, 0,
7, 70], [126, 0, 7, 70], [127, 1, 43, 71], [128, 1, 40, 71], [129, 1,
59, 71], [130, 1, 38, 71], [131, 1, 47, 71], [132, 1, 39, 71], [133, 0,
7, 72], [134, 0, 7, 72], [135, 1, 7, 73], [136, 0, 7, 73], [137, 0, 44,
73], [138, 1, 7, 73], [139, 1, 7, 74], [140, 0, 35, 74], [141, 0, 57,
75], [142, 1, 7, 75], [143, 0, 7, 76], [144, 0, 7, 76], [145, 1, 7,
77], [146, 1, 7, 77], [147, 1, 48, 77], [148, 0, 7, 77], [149, 0, 7,
78], [150, 1, 7, 78], [151, 1, 43, 78], [152, 1, 39, 78], [153, 0, 44,
78], [154, 0, 38, 78], [155, 0, 47, 78], [156, 0, 7, 78], [157, 1, 37,
78], [158, 0, 7, 78], [159, 1, 7, 78], [160, 0, 7, 78], [161, 0, 56,
79], [162, 0, 7, 79], [163, 1, 7, 81], [164, 0, 7, 81], [165, 1, 50,
85], [166, 0, 36, 85], [167, 1, 42, 86], [168, 0, 7, 86], [169, 0, 36,
87], [170, 1, 7, 87], [171, 1, 40, 87], [172, 1, 7, 87], [173, 1, 36,
87], [174, 1, 36, 87], [175, 0, 52, 88], [176, 0, 7, 88], [177, 1, 58,
88], [178, 1, 7, 88], [179, 1, 59, 93], [180, 1, 35, 93], [181, 0, 37,
97], [182, 0, 7, 97], [183, 1, 46, 98], [184, 0, 7, 98], [185, 0, 41,
99], [186, 1, 7, 99], [187, 0, 54, 101], [188, 1, 7, 101], [189, 0, 41,
```

```
103], [190, 0, 36, 103], [191, 0, 7, 103], [192, 0, 7, 103], [193, 1,
7, 113], [194, 0, 38, 113], [195, 0, 47, 120], [196, 0, 35, 120], [197,
0, 45, 126], [198, 1, 7, 126], [199, 1, 7, 137], [200, 1, 7, 137]])
```

```
plt.scatter(data['Annual Income (k$)'],data['Spending Score (1-
100)'],color='red')
```
**Solution:**



```
#Split the data into training and testing
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)
print(X_train.shape,X_test.shape)
```
**Solution:**

```
(160, 4) (40, 4)
```

```
# Build the Model
my_dict=pd.read_csv("/content/Mall_Customers.csv")
df = pd.DataFrame(my_dict)
print(df)
```

**Solution:**

```
#  Build the Model
import csv
with open("/content/Mall_Customers.csv") as csv_file:
    csv_reader = csv.reader(csv_file)
    df = pd.DataFrame([csv_reader], index = None)
for val in list(df[1]):
    print(val)
```

**Solution:**

```
['1', 'Male', '19', '15', '39']
```

```
#Train the Model
model.fit(X_train,y_train)
```
**Solution:**

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

```
#Test the Model
pred2=model.predict(X_test)
#Measure the performance using Evaluation Metrics
from sklearn.metrics import accuracy_score
accuracy_score(y_test,pred2)
```
**Solution:**

```
0.0
```

```
#prediction on the test
pred=rf.predict(X_test)
# Accuracy of DT model
from sklearn.metrics import accuracy_score
accuracy_score(y_test,pred)
```
**Solution:**

```
0.025
```

```
#Test the model
pred=nb.predict(X_test)
pred
```
**Solution:**

```
array([90, 58, 77, 97, 92, 40, 10, 77, 54, 97, 40, 58, 28, 40, 97, 40,
       40, 92, 58, 92, 90, 97, 73, 32, 90, 77, 97, 77, 74, 77, 77, 90, 58, 77,
       73, 32, 92, 77, 51, 51])
```

```
#evalute model matrix
from sklearn.metrics import accuracy_score, confusion_matrix
accuracy_score(y_test,pred)
```

**Solution:**

```
0.025
```