Customer Segmentation Analysis

```
## import required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm

from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
## loading the dataset

df=pd.read_csv('/content/drive/MyDrive/Mall_Customers.csv')
df.head()
```

```
   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
```

```
df.shape
```

```
(200, 5)
```

```
df.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
```

```
df.isnull().any()
```

```
CustomerID                False
Gender                    False
```

```
Age                     False
Annual Income (k$)      False
Spending Score (1-100)  False
dtype: bool
```

```
df.isnull().any()
```

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

```
df.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
```

Univariate Analysis

```
df['Age'].mean()
```

```
38.85
```

```
df['Age'].median()
```

```
36.0
```

```
df['Age'].std()
```

```
13.96900733155888
```

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

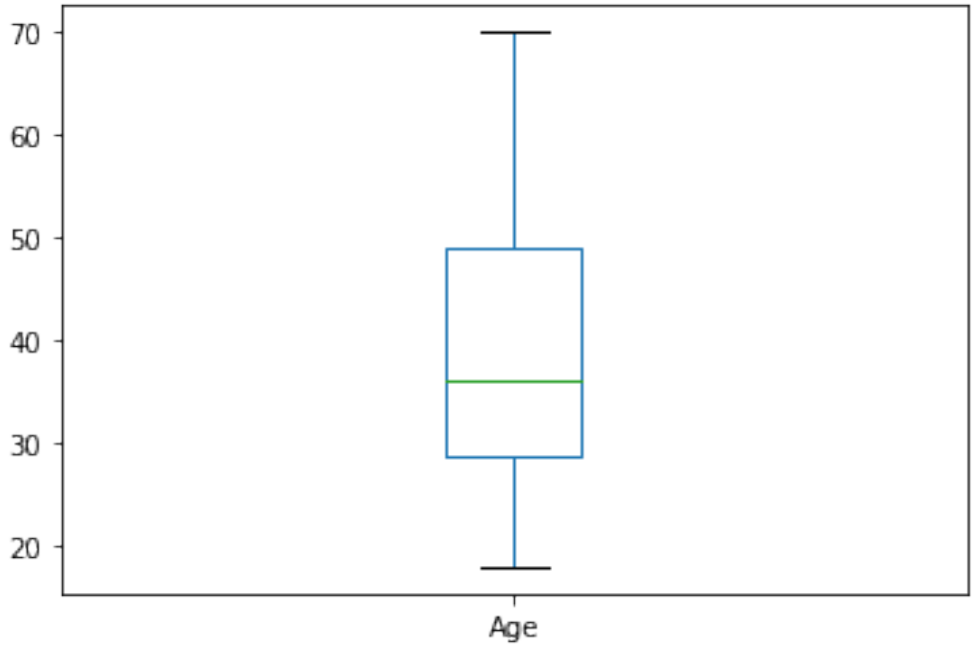
```
54     12
78     12
```

```
48      6
71      6
63      6
       ..
58      2
59      2
16      2
64      2
137     2
Name: Annual Income (k$), Length: 64, dtype: int64
```

Visualization

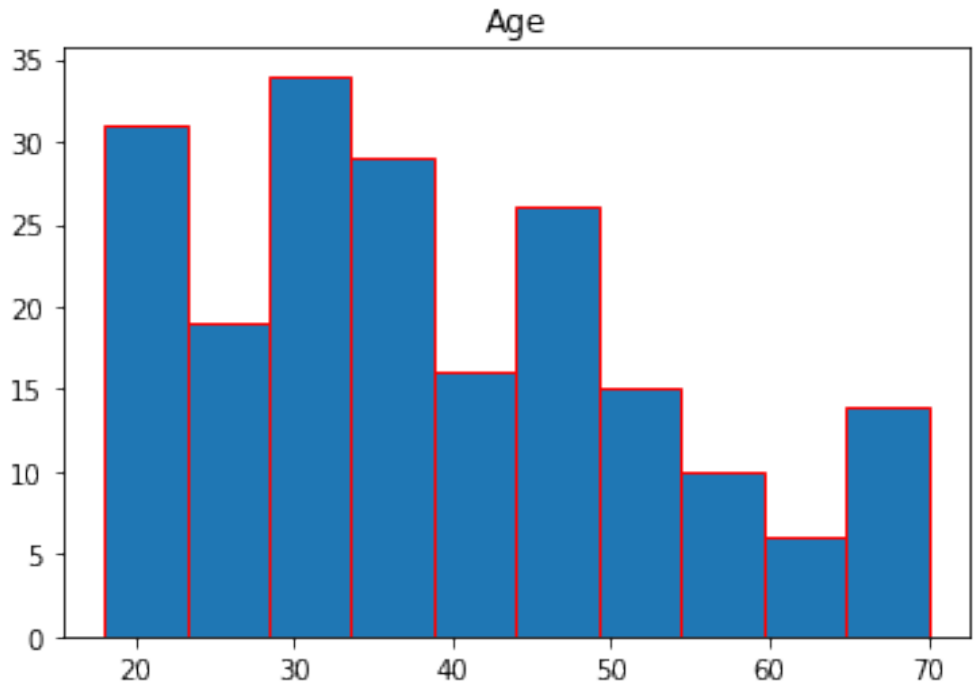
```
df.boxplot(column=['Age'], grid=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe6894a2150>
```



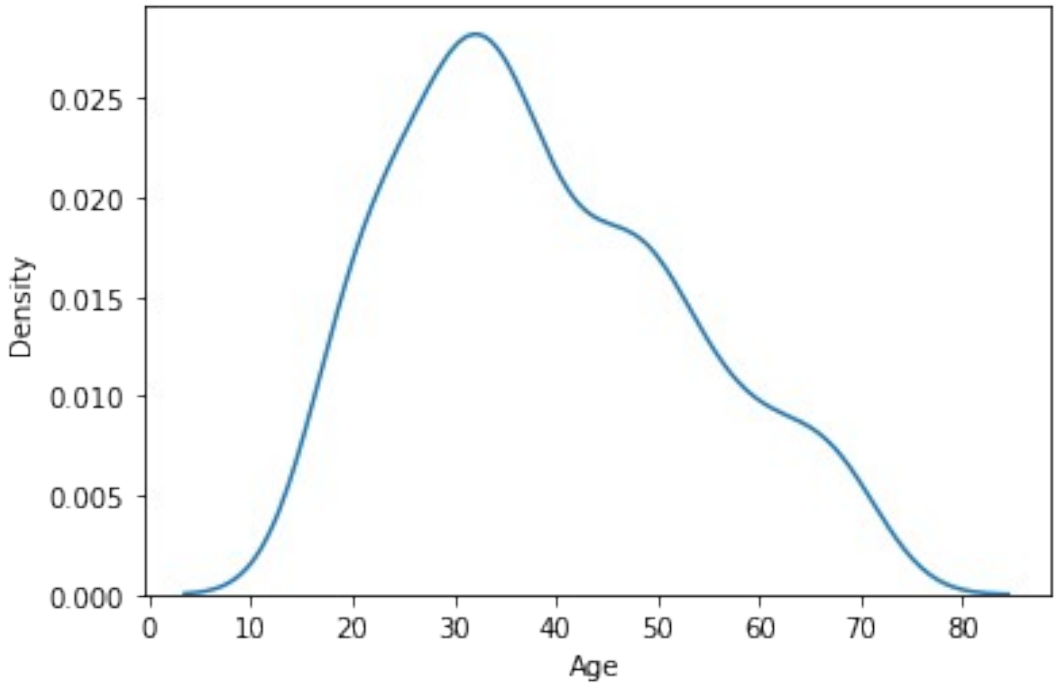

```
df.hist(column='Age', grid=False, edgecolor='Red')
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at
0x7fe688f2a690>]],
      dtype=object)
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe688ebba50>
```

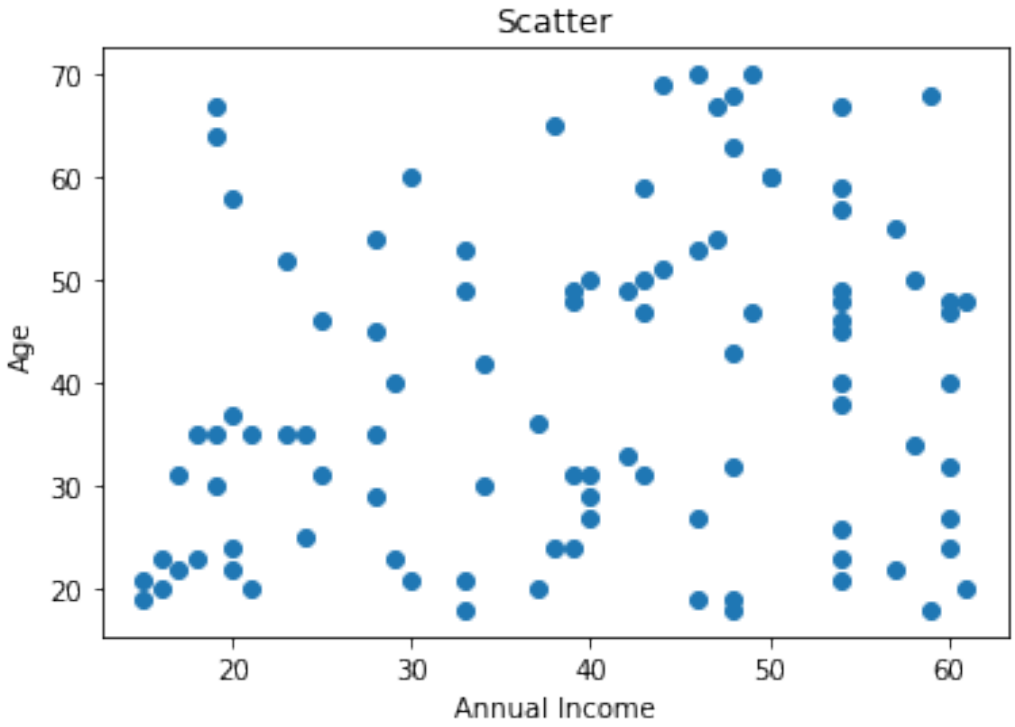


Bi - Variate Analysis

1. Scatterplots

```
plt.scatter(x=df["Annual Income (k$)"].head(100), y=df.Age.head(100))
plt.title('Scatter')
plt.xlabel('Annual Income')
plt.ylabel('Age')
```

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



1. Correlation Coefficients

```
df.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
```

```
y = df['Annual Income (k$)']
x = df['Spending Score (1-100)']
x = sm.add_constant(x)
model = sm.OLS(y,x).fit()
model.summary()
```

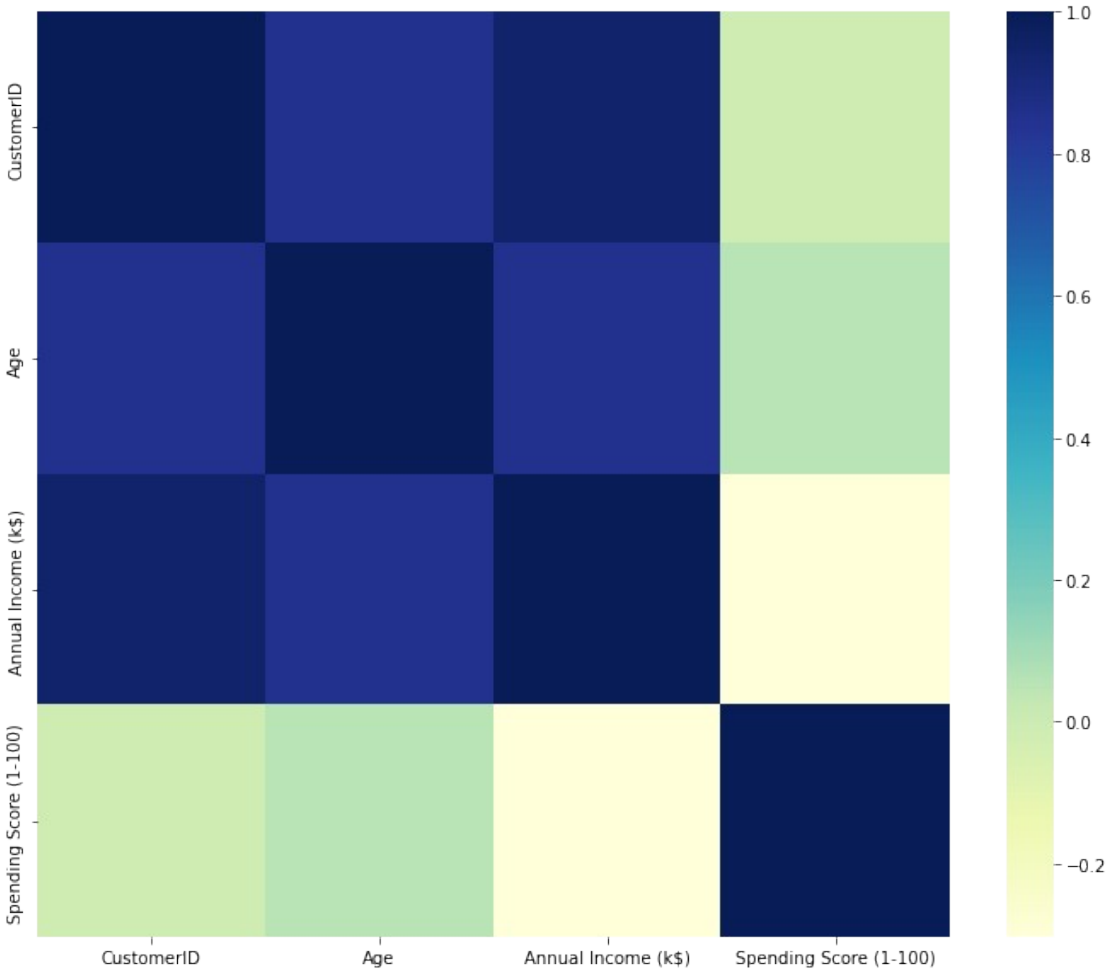
```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/
tsatools.py:142: FutureWarning: In a future version of pandas all
arguments of concat except for the argument 'objs' will be keyword-
only
  x = pd.concat(x[::order], 1)

<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results

==============================================================================
========
Dep. Variable:     Annual Income (k$)   R-squared:
0.000
Model:                             OLS   Adj. R-squared:
-0.005
Method:                  Least Squares   F-statistic:
0.01942
Date:                 Sat, 29 Oct 2022   Prob (F-statistic):
0.889
Time:                         09:59:22   Log-Likelihood:
-936.92
No. Observations:                  200   AIC:
1878.
Df Residuals:                      198   BIC:
1884.
Df Model:                            1

Covariance Type:            nonrobust

==============================================================================
====================
                               coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
--------------------
const                       60.0544      4.078     14.726      0.000
52.012      68.097
Spending Score (1-100)      0.0101      0.072      0.139      0.889
-0.132       0.153
==============================================================================
========
Omnibus:                        3.510   Durbin-Watson:
0.005
Prob(Omnibus):                  0.173   Jarque-Bera (JB):
3.531
Skew:                           0.319   Prob(JB):
0.171
Kurtosis:                       2.875   Cond. No.
124.
```

```
==============================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
"""
```

Multi - Variate Analysis

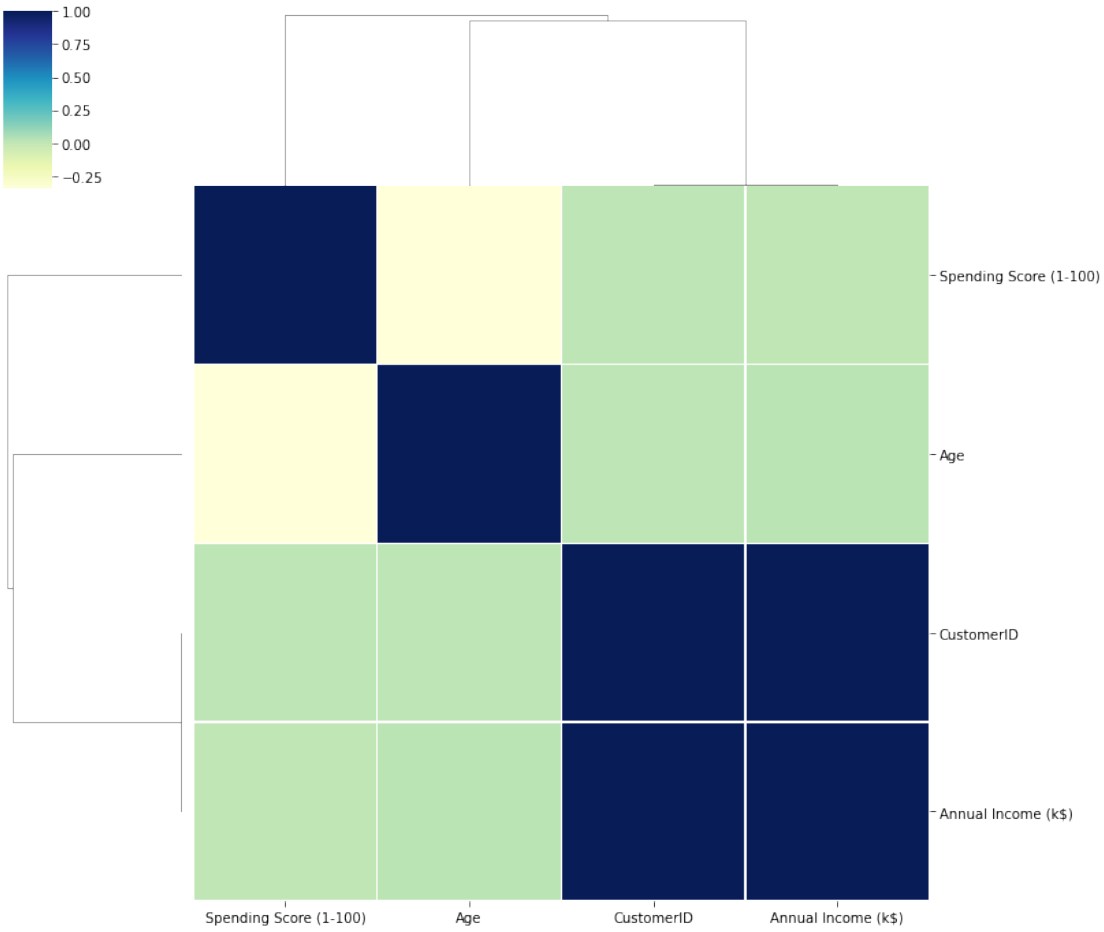
```
f = plt.subplots(figsize=(12,10))
sns.heatmap(df.head().corr(), cmap="YlGnBu")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe688c92e90>
```




```
corrmat = df.corr(method='spearman')
cg = sns.clustermap(corrmat, cmap="YlGnBu", linewidths=0.1);
plt.setp(cg.ax_heatmap.yaxis.get_majorticklabels(), rotation=0)
cg
```

```
<seaborn.matrix.ClusterGrid at 0x7fe688e5cc90>
```



1. Perform descriptive statistics on the dataset.

```
df.shape
```

```
(200, 5)
```

```
df.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
```

```
df.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
```

```
df.head()
```

```
   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
```

```
df.tail()
```

```
     CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-
100)
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
```

```
df["Annual Income (k$)"].mean()
```

```
60.56
```

```
df["Annual Income (k$)"].median()
```

```
61.5
```

```
df["Annual Income (k$)"].mode()
```

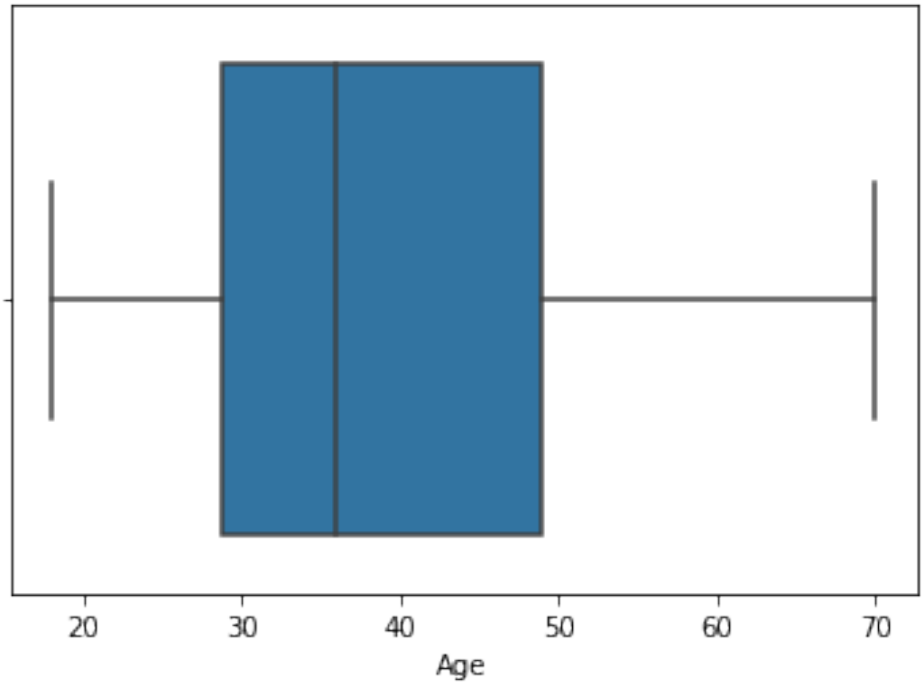
```
0    54
1    78
dtype: int64
```

```
df["Annual Income (k$)"].var()
```

```
689.8355778894472
```

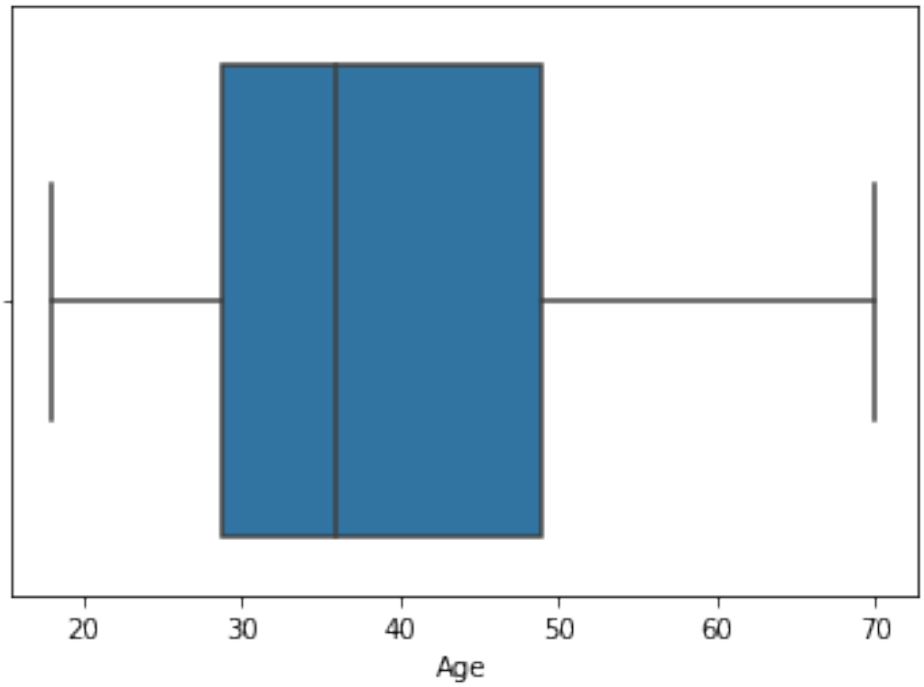
```
sns.boxplot(df["Age"])
import warnings
warnings.filterwarnings('ignore')
```

```
/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
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe6860d6850>
```



1. Handle the Missing values.

```
print(df.isnull())
```

```
     CustomerID  Gender    Age  Annual Income (k$)  Spending Score (1-
100)
0         False   False  False               False
False
1         False   False  False               False
False
2         False   False  False               False
False
3         False   False  False               False
False
4         False   False  False               False
False
..          ...     ...    ...                 ...
...
195       False   False  False               False
False
196       False   False  False               False
False
197       False   False  False               False
False
198       False   False  False               False
False
199       False   False  False               False
False

[200 rows x 5 columns]
```

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

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

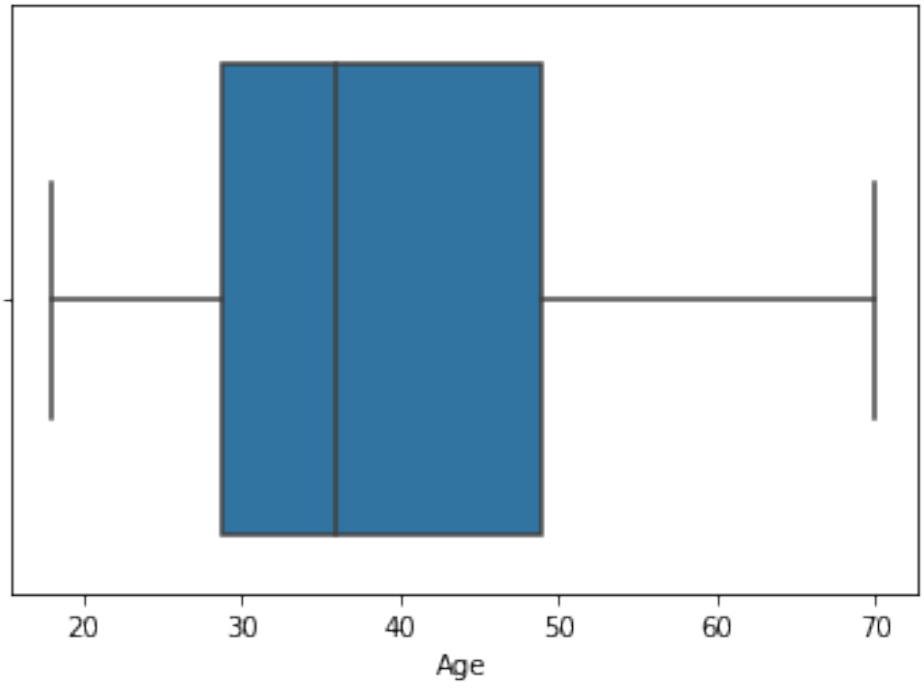
```
df.isna().any()
```

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

1. Find the outliers and replace the outliers

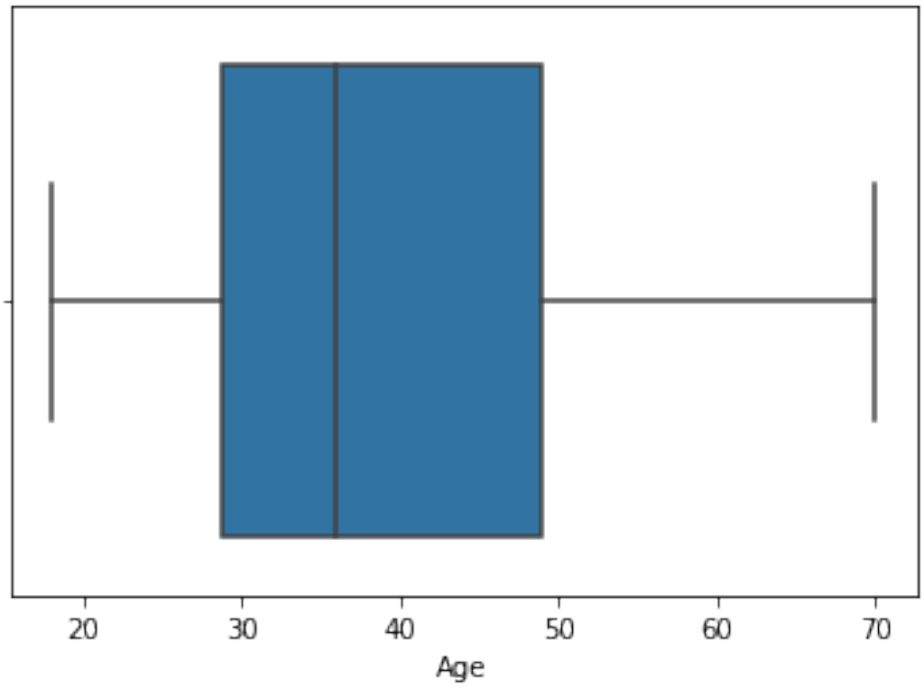
```
x = sns.boxplot(x=df["Age"])
x
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe686257850>
```




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

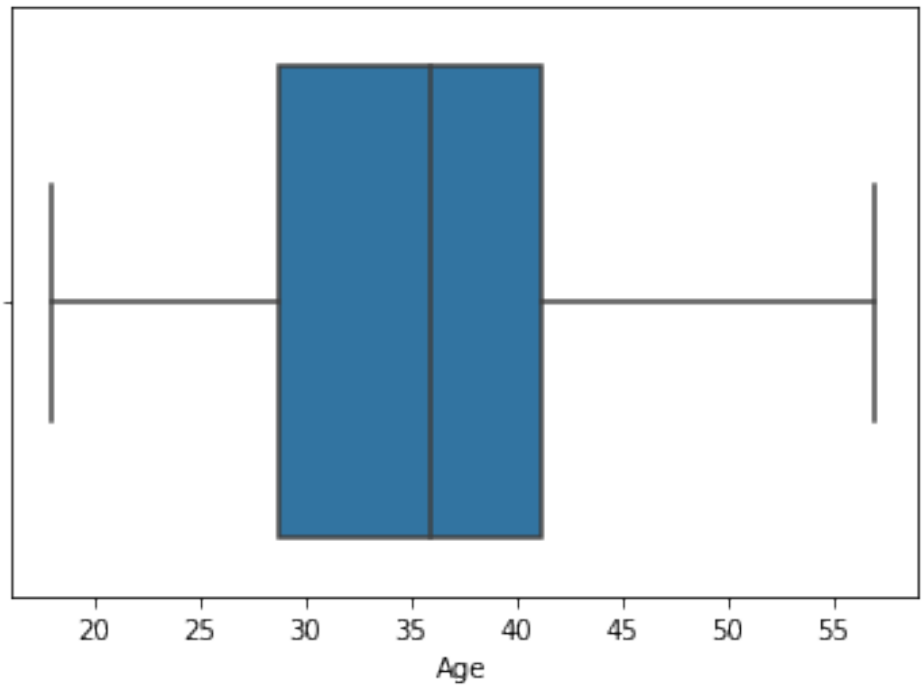
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe6861013d0>
```




```
df['Age']=np.where(df['Age']>57,39, df['Age'])
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe68604bc10>
```



7.Check for Categorical columns and perform encoding

```
pd.Categorical(df["Annual Income (k$)"])
```

```
[15, 15, 16, 16, 17, ..., 120, 126, 126, 137, 137]
Length: 200
Categories (64, int64): [15, 16, 17, 18, ..., 113, 120, 126, 137]
```

```
# One Hot Encoding
```

```
pd.get_dummies(df["Annual Income (k$)"]).head(10)
```

```
   15   16   17   18   19   20   21   23   24   25   ...  93   97   98
99   \
0    1    0    0    0    0    0    0    0    0    0  ...    0    0
0    0
1    1    0    0    0    0    0    0    0    0    0  ...    0    0
0    0
2    0    1    0    0    0    0    0    0    0    0  ...    0    0
0    0
3    0    1    0    0    0    0    0    0    0    0  ...    0    0
0    0
4    0    0    1    0    0    0    0    0    0    0  ...    0    0
0    0
5    0    0    1    0    0    0    0    0    0    0  ...    0    0
0    0
6    0    0    0    1    0    0    0    0    0    0  ...    0    0
0    0
7    0    0    0    1    0    0    0    0    0    0  ...    0    0
0    0
8    0    0    0    0    1    0    0    0    0    0  ...    0    0
0    0
9    0    0    0    0    1    0    0    0    0    0  ...    0    0
0    0

   101  103  113  120  126  137
0    0    0    0    0    0    0
1    0    0    0    0    0    0
2    0    0    0    0    0    0
3    0    0    0    0    0    0
4    0    0    0    0    0    0
5    0    0    0    0    0    0
6    0    0    0    0    0    0
7    0    0    0    0    0    0
8    0    0    0    0    0    0
9    0    0    0    0    0    0

[10 rows x 64 columns]
```

```
pd.get_dummies(df).head(10)
```

```
   CustomerID  Age  Annual Income (k$)  Spending Score (1-100)
Gender_Female  \
0           1   19                  15                      39
0
1           2   21                  15                      81
0
2           3   20                  16                       6
1
3           4   23                  16                      77
1
4           5   31                  17                      40
1
5           6   22                  17                      76
1
6           7   35                  18                       6
1
7           8   23                  18                      94
1
8           9   39                  19                       3
0
9          10   30                  19                      72
1

   Gender_Male
0            1
1            1
2            0
3            0
4            0
5            0
6            0
7            0
8            1
9            0
```

1. Scaling the data

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

label = LabelEncoder()
label = label.fit_transform(df['Gender'])
df["Gender"] = label
df['Gender'].value_counts()

X = df.drop("Age",axis=1)
Y = df['Age']
object1 = StandardScaler()
scale = object1.fit_transform(X)
scale
```

```
array([[-1.7234121 ,  1.12815215, -1.73899919, -0.43480148],
       [-1.70609137,  1.12815215, -1.73899919,  1.19570407],
       [-1.68877065, -0.88640526, -1.70082976, -1.71591298],
       [-1.67144992, -0.88640526, -1.70082976,  1.04041783],
       [-1.6541292 , -0.88640526, -1.66266033, -0.39597992],
       [-1.63680847, -0.88640526, -1.66266033,  1.00159627],
       [-1.61948775, -0.88640526, -1.62449091, -1.71591298],
       [-1.60216702, -0.88640526, -1.62449091,  1.70038436],
       [-1.5848463 ,  1.12815215, -1.58632148, -1.83237767],
       [-1.56752558, -0.88640526, -1.58632148,  0.84631002],
       [-1.55020485,  1.12815215, -1.58632148, -1.4053405 ],
       [-1.53288413, -0.88640526, -1.58632148,  1.89449216],
       [-1.5155634 , -0.88640526, -1.54815205, -1.36651894],
       [-1.49824268, -0.88640526, -1.54815205,  1.04041783],
       [-1.48092195,  1.12815215, -1.54815205, -1.44416206],
       [-1.46360123,  1.12815215, -1.54815205,  1.11806095],
       [-1.4462805 , -0.88640526, -1.50998262, -0.59008772],
       [-1.42895978,  1.12815215, -1.50998262,  0.61338066],
       [-1.41163905,  1.12815215, -1.43364376, -0.82301709],
       [-1.39431833, -0.88640526, -1.43364376,  1.8556706 ],
       [-1.3769976 ,  1.12815215, -1.39547433, -0.59008772],
       [-1.35967688,  1.12815215, -1.39547433,  0.88513158],
       [-1.34235616, -0.88640526, -1.3573049 , -1.75473454],
       [-1.32503543,  1.12815215, -1.3573049 ,  0.88513158],
       [-1.30771471, -0.88640526, -1.24279661, -1.4053405 ],
       [-1.29039398,  1.12815215, -1.24279661,  1.23452563],
       [-1.27307326, -0.88640526, -1.24279661, -0.7065524 ],
       [-1.25575253,  1.12815215, -1.24279661,  0.41927286],
       [-1.23843181, -0.88640526, -1.20462718, -0.74537397],
       [-1.22111108, -0.88640526, -1.20462718,  1.42863343],
       [-1.20379036,  1.12815215, -1.16645776, -1.7935561 ],
       [-1.18646963, -0.88640526, -1.16645776,  0.88513158],
       [-1.16914891,  1.12815215, -1.05194947, -1.7935561 ],
       [-1.15182818,  1.12815215, -1.05194947,  1.62274124],
       [-1.13450746, -0.88640526, -1.05194947, -1.4053405 ],
       [-1.11718674, -0.88640526, -1.05194947,  1.19570407],
       [-1.09986601, -0.88640526, -1.01378004, -1.28887582],
       [-1.08254529, -0.88640526, -1.01378004,  0.88513158],
       [-1.06522456, -0.88640526, -0.89927175, -0.93948177],
       [-1.04790384, -0.88640526, -0.89927175,  0.96277471],
       [-1.03058311, -0.88640526, -0.86110232, -0.59008772],
       [-1.01326239,  1.12815215, -0.86110232,  1.62274124],
       [-0.99594166,  1.12815215, -0.82293289, -0.55126616],
       [-0.97862094, -0.88640526, -0.82293289,  0.41927286],
       [-0.96130021, -0.88640526, -0.82293289, -0.86183865],
       [-0.94397949, -0.88640526, -0.82293289,  0.5745591 ],
       [-0.92665877, -0.88640526, -0.78476346,  0.18634349],
       [-0.90933804, -0.88640526, -0.78476346, -0.12422899],
       [-0.89201732, -0.88640526, -0.78476346, -0.3183368 ],
       [-0.87469659, -0.88640526, -0.78476346, -0.3183368 ],
```

```
[-0.85737587, -0.88640526, -0.70842461,  0.06987881],
[-0.84005514,  1.12815215, -0.70842461,  0.38045129],
[-0.82273442, -0.88640526, -0.67025518,  0.14752193],
[-0.80541369,  1.12815215, -0.67025518,  0.38045129],
[-0.78809297, -0.88640526, -0.67025518, -0.20187212],
[-0.77077224,  1.12815215, -0.67025518, -0.35715836],
[-0.75345152, -0.88640526, -0.63208575, -0.00776431],
[-0.73613079,  1.12815215, -0.63208575, -0.16305055],
[-0.71881007, -0.88640526, -0.55574689,  0.03105725],
[-0.70148935,  1.12815215, -0.55574689, -0.16305055],
[-0.68416862,  1.12815215, -0.55574689,  0.22516505],
[-0.6668479 ,  1.12815215, -0.55574689,  0.18634349],
[-0.64952717, -0.88640526, -0.51757746,  0.06987881],
[-0.63220645, -0.88640526, -0.51757746,  0.34162973],
[-0.61488572,  1.12815215, -0.47940803,  0.03105725],
[-0.597565  ,  1.12815215, -0.47940803,  0.34162973],
[-0.58024427, -0.88640526, -0.47940803, -0.00776431],
[-0.56292355, -0.88640526, -0.47940803, -0.08540743],
[-0.54560282,  1.12815215, -0.47940803,  0.34162973],
[-0.5282821 , -0.88640526, -0.47940803, -0.12422899],
[-0.51096138,  1.12815215, -0.4412386 ,  0.18634349],
[-0.49364065, -0.88640526, -0.4412386 , -0.3183368 ],
[-0.47631993, -0.88640526, -0.40306917, -0.04658587],
[-0.4589992 , -0.88640526, -0.40306917,  0.22516505],
[-0.44167848,  1.12815215, -0.25039146, -0.12422899],
[-0.42435775,  1.12815215, -0.25039146,  0.14752193],
[-0.40703703, -0.88640526, -0.25039146,  0.10870037],
[-0.3897163 ,  1.12815215, -0.25039146, -0.08540743],
[-0.37239558, -0.88640526, -0.25039146,  0.06987881],
[-0.35507485, -0.88640526, -0.25039146, -0.3183368 ],
[-0.33775413,  1.12815215, -0.25039146,  0.03105725],
[-0.3204334 ,  1.12815215, -0.25039146,  0.18634349],
[-0.30311268,  1.12815215, -0.25039146, -0.35715836],
[-0.28579196, -0.88640526, -0.25039146, -0.24069368],
[-0.26847123, -0.88640526, -0.25039146,  0.26398661],
[-0.25115051,  1.12815215, -0.25039146, -0.16305055],
[-0.23382978, -0.88640526, -0.13588317,  0.30280817],
[-0.21650906, -0.88640526, -0.13588317,  0.18634349],
[-0.19918833, -0.88640526, -0.09771374,  0.38045129],
[-0.18186761, -0.88640526, -0.09771374, -0.16305055],
[-0.16454688, -0.88640526, -0.05954431,  0.18634349],
[-0.14722616,  1.12815215, -0.05954431, -0.35715836],
[-0.12990543,  1.12815215, -0.02137488, -0.04658587],
[-0.11258471, -0.88640526, -0.02137488, -0.39597992],
[-0.09526399, -0.88640526, -0.02137488, -0.3183368 ],
[-0.07794326,  1.12815215, -0.02137488,  0.06987881],
[-0.06062254, -0.88640526, -0.02137488, -0.12422899],
[-0.04330181, -0.88640526, -0.02137488, -0.00776431],
[-0.02598109,  1.12815215,  0.01679455, -0.3183368 ],
[-0.00866036,  1.12815215,  0.01679455, -0.04658587],
```

```
[ 0.00866036, -0.88640526,  0.05496398, -0.35715836],
[ 0.02598109, -0.88640526,  0.05496398, -0.08540743],
[ 0.04330181,  1.12815215,  0.05496398,  0.34162973],
[ 0.06062254,  1.12815215,  0.05496398,  0.18634349],
[ 0.07794326,  1.12815215,  0.05496398,  0.22516505],
[ 0.09526399, -0.88640526,  0.05496398, -0.3183368 ],
[ 0.11258471, -0.88640526,  0.09313341, -0.00776431],
[ 0.12990543,  1.12815215,  0.09313341, -0.16305055],
[ 0.14722616,  1.12815215,  0.09313341, -0.27951524],
[ 0.16454688,  1.12815215,  0.09313341, -0.08540743],
[ 0.18186761,  1.12815215,  0.09313341,  0.06987881],
[ 0.19918833, -0.88640526,  0.09313341,  0.14752193],
[ 0.21650906, -0.88640526,  0.13130284, -0.3183368 ],
[ 0.23382978,  1.12815215,  0.13130284, -0.16305055],
[ 0.25115051, -0.88640526,  0.16947227, -0.08540743],
[ 0.26847123, -0.88640526,  0.16947227, -0.00776431],
[ 0.28579196, -0.88640526,  0.16947227, -0.27951524],
[ 0.30311268, -0.88640526,  0.16947227,  0.34162973],
[ 0.3204334 , -0.88640526,  0.24581112, -0.27951524],
[ 0.33775413, -0.88640526,  0.24581112,  0.26398661],
[ 0.35507485,  1.12815215,  0.24581112,  0.22516505],
[ 0.37239558, -0.88640526,  0.24581112, -0.39597992],
[ 0.3897163 , -0.88640526,  0.32214998,  0.30280817],
[ 0.40703703,  1.12815215,  0.32214998,  1.58391968],
[ 0.42435775, -0.88640526,  0.36031941, -0.82301709],
[ 0.44167848, -0.88640526,  0.36031941,  1.04041783],
[ 0.4589992 ,  1.12815215,  0.39848884, -0.59008772],
[ 0.47631993,  1.12815215,  0.39848884,  1.73920592],
[ 0.49364065,  1.12815215,  0.39848884, -1.52180518],
[ 0.51096138,  1.12815215,  0.39848884,  0.96277471],
[ 0.5282821 ,  1.12815215,  0.39848884, -1.5994483 ],
[ 0.54560282,  1.12815215,  0.39848884,  0.96277471],
[ 0.56292355, -0.88640526,  0.43665827, -0.62890928],
[ 0.58024427, -0.88640526,  0.43665827,  0.80748846],
[ 0.597565  ,  1.12815215,  0.4748277 , -1.75473454],
[ 0.61488572, -0.88640526,  0.4748277 ,  1.46745499],
[ 0.63220645, -0.88640526,  0.4748277 , -1.67709142],
[ 0.64952717,  1.12815215,  0.4748277 ,  0.88513158],
[ 0.6668479 ,  1.12815215,  0.51299713, -1.56062674],
[ 0.68416862, -0.88640526,  0.51299713,  0.84631002],
[ 0.70148935, -0.88640526,  0.55116656, -1.75473454],
[ 0.71881007,  1.12815215,  0.55116656,  1.6615628 ],
[ 0.73613079, -0.88640526,  0.58933599, -0.39597992],
[ 0.75345152, -0.88640526,  0.58933599,  1.42863343],
[ 0.77077224,  1.12815215,  0.62750542, -1.48298362],
[ 0.78809297,  1.12815215,  0.62750542,  1.81684904],
[ 0.80541369,  1.12815215,  0.62750542, -0.55126616],
[ 0.82273442, -0.88640526,  0.62750542,  0.92395314],
[ 0.84005514, -0.88640526,  0.66567484, -1.09476801],
[ 0.85737587,  1.12815215,  0.66567484,  1.54509812],
```

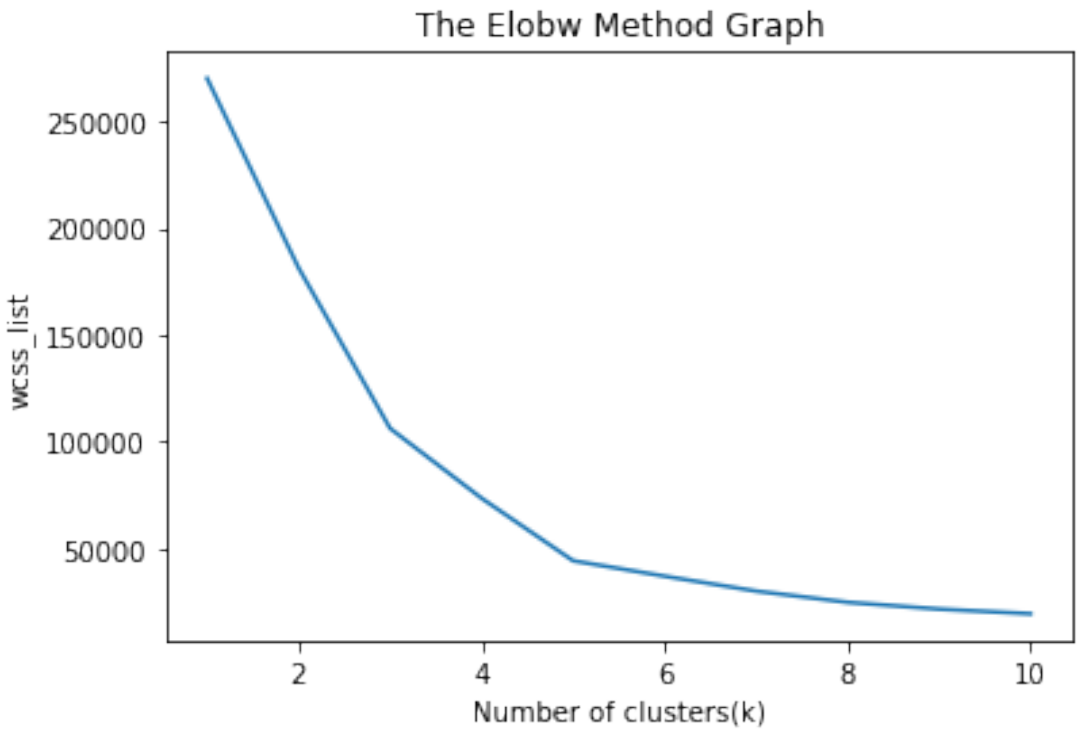
```
[ 0.87469659,  1.12815215,  0.66567484, -1.28887582],
[ 0.89201732,  1.12815215,  0.66567484,  1.46745499],
[ 0.90933804, -0.88640526,  0.66567484, -1.17241113],
[ 0.92665877, -0.88640526,  0.66567484,  1.00159627],
[ 0.94397949, -0.88640526,  0.66567484, -1.32769738],
[ 0.96130021, -0.88640526,  0.66567484,  1.50627656],
[ 0.97862094,  1.12815215,  0.66567484, -1.91002079],
[ 0.99594166, -0.88640526,  0.66567484,  1.07923939],
[ 1.01326239,  1.12815215,  0.66567484, -1.91002079],
[ 1.03058311, -0.88640526,  0.66567484,  0.88513158],
[ 1.04790384, -0.88640526,  0.70384427, -0.59008772],
[ 1.06522456, -0.88640526,  0.70384427,  1.27334719],
[ 1.08254529,  1.12815215,  0.78018313, -1.75473454],
[ 1.09986601, -0.88640526,  0.78018313,  1.6615628 ],
[ 1.11718674,  1.12815215,  0.93286085, -0.93948177],
[ 1.13450746, -0.88640526,  0.93286085,  0.96277471],
[ 1.15182818,  1.12815215,  0.97103028, -1.17241113],
[ 1.16914891, -0.88640526,  0.97103028,  1.73920592],
[ 1.18646963, -0.88640526,  1.00919971, -0.90066021],
[ 1.20379036,  1.12815215,  1.00919971,  0.49691598],
[ 1.22111108,  1.12815215,  1.00919971, -1.44416206],
[ 1.23843181,  1.12815215,  1.00919971,  0.96277471],
[ 1.25575253,  1.12815215,  1.00919971, -1.56062674],
[ 1.27307326,  1.12815215,  1.00919971,  1.62274124],
[ 1.29039398, -0.88640526,  1.04736914, -1.44416206],
[ 1.30771471, -0.88640526,  1.04736914,  1.38981187],
[ 1.32503543,  1.12815215,  1.04736914, -1.36651894],
[ 1.34235616,  1.12815215,  1.04736914,  0.72984534],
[ 1.35967688,  1.12815215,  1.23821628, -1.4053405 ],
[ 1.3769976 ,  1.12815215,  1.23821628,  1.54509812],
[ 1.39431833, -0.88640526,  1.390894  , -0.7065524 ],
[ 1.41163905, -0.88640526,  1.390894  ,  1.38981187],
[ 1.42895978,  1.12815215,  1.42906343, -1.36651894],
[ 1.4462805 , -0.88640526,  1.42906343,  1.46745499],
[ 1.46360123, -0.88640526,  1.46723286, -0.43480148],
[ 1.48092195,  1.12815215,  1.46723286,  1.81684904],
[ 1.49824268, -0.88640526,  1.54357172, -1.01712489],
[ 1.5155634 ,  1.12815215,  1.54357172,  0.69102378],
[ 1.53288413, -0.88640526,  1.61991057, -1.28887582],
[ 1.55020485, -0.88640526,  1.61991057,  1.35099031],
[ 1.56752558, -0.88640526,  1.61991057, -1.05594645],
[ 1.5848463 , -0.88640526,  1.61991057,  0.72984534],
[ 1.60216702,  1.12815215,  2.00160487, -1.63826986],
[ 1.61948775, -0.88640526,  2.00160487,  1.58391968],
[ 1.63680847, -0.88640526,  2.26879087, -1.32769738],
[ 1.6541292 , -0.88640526,  2.26879087,  1.11806095],
[ 1.67144992, -0.88640526,  2.49780745, -0.86183865],
[ 1.68877065,  1.12815215,  2.49780745,  0.92395314],
[ 1.70609137,  1.12815215,  2.91767117, -1.25005425],
[ 1.7234121 ,  1.12815215,  2.91767117,  1.27334719]])
```

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

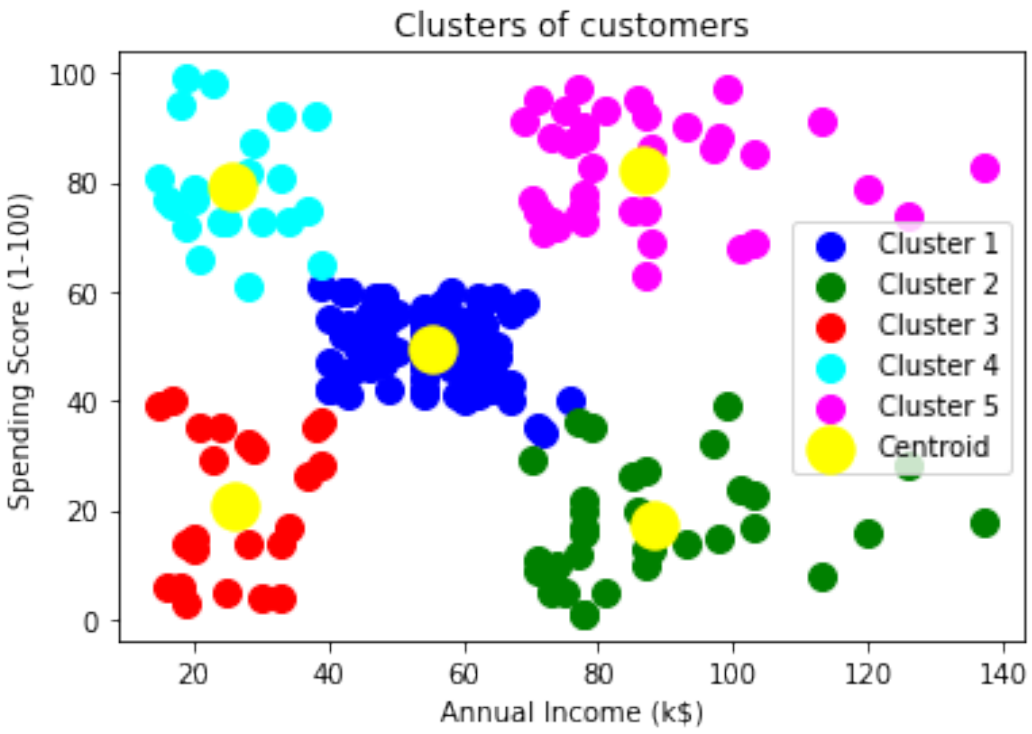
```
     CustomerID    Gender  Annual Income (k$)  Spending Score (1-100)
0     -1.723412  1.128152           -1.738999               -0.434801
1     -1.706091  1.128152           -1.738999                1.195704
2     -1.688771 -0.886405           -1.700830               -1.715913
3     -1.671450 -0.886405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

[200 rows x 4 columns]
```

1. Perform any of the clustering algorithms

```
from sklearn.cluster import KMeans
x = df.iloc[:, [3, 4]].values
list= []

for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)

    kmeans.fit(x)
    list.append(kmeans.inertia_)
plt.plot(range(1, 11), list)
plt.title('The Elobw Method Graph')
plt.xlabel('Number of clusters(k)')
plt.ylabel('wcss_list')
plt.show()
```




```
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)
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()
```



1. Add the cluster data with the primary dataset

```
df['Cluster']=kmeans.labels_
df.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        2
1        3
2        2
3        3
4        2

df.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        4
196        1
197        4
198        1
199        4
```

1. Split the data into dependent and independent variables.

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

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

```
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,rando
m_state=42)
```

```
X_train.shape
```

```
(160, 5)
```

```
y_train.shape
```

```
(160,)
```

1. Split the data into training and testing

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   39                  63
50
14            15       1   37                  20
13
92            93       1   48                  60
49
179          180       1   35                  93
90
102          103       1   39                  62
59

[160 rows x 5 columns]
```

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   39                  30
4
158          159       1   34                  78
1
128          129       1   39                  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 39 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 39 46
56
84 85 0 21 54
57
67 68 0 39 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
```

y_train

```
79     0
197    4
38     2
24     2
122    0
      ..
106    0
14     2
92     0
179    4
102    0
Name: Cluster, Length: 160, dtype: int32
```

y_test

```
95     0
15     3
30     2
158    1
128    1
115    0
69     0
170    1
174    1
45     3
66     0
182    1
165    4
78     0
186    1
177    4
56     0
```

```
152    1
82     0
68     0
124    1
16     2
148    1
93     0
65     0
60     0
84     0
67     0
125    4
132    0
9      3
18     2
55     0
75     0
150    1
104    0
135    4
137    4
164    1
76     0
Name: Cluster, dtype: int32
```

1. Build the Model

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

```
LogisticRegression()
```

```
LogisticRegression()
```

1. Train the Model

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

```
0.98125
```

1. Test the Model

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

```
0.925
```

1. Measure the performance using Evaluation Metrics.

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

```
array([[15,  1,  0,  0,  2],
       [ 0, 11,  0,  0,  0],
       [ 0,  0,  3,  0,  0],
       [ 0,  0,  0,  3,  0],
       [ 0,  0,  0,  0,  5]])
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.83      0.91        18
           1       0.92      1.00      0.96        11
           2       1.00      1.00      1.00         3
           3       1.00      1.00      1.00         3
           4       0.71      1.00      0.83         5

    accuracy                           0.93        40
   macro avg       0.93      0.97      0.94        40
weighted avg       0.94      0.93      0.93        40
```