Load the dataset into the tool.

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

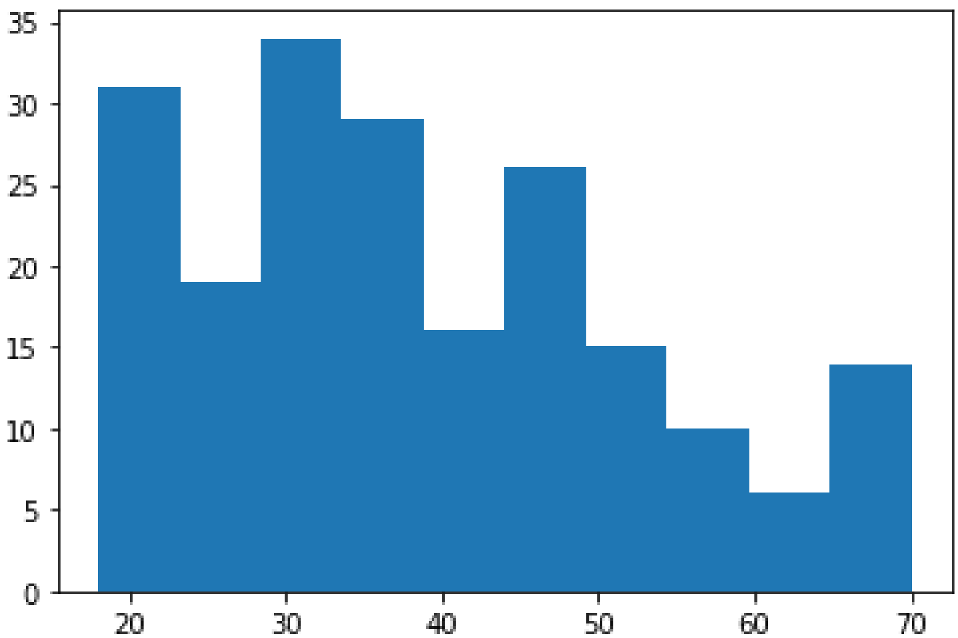
```
df=pd.read_csv("Mall_Customers.csv")
df.head()
```

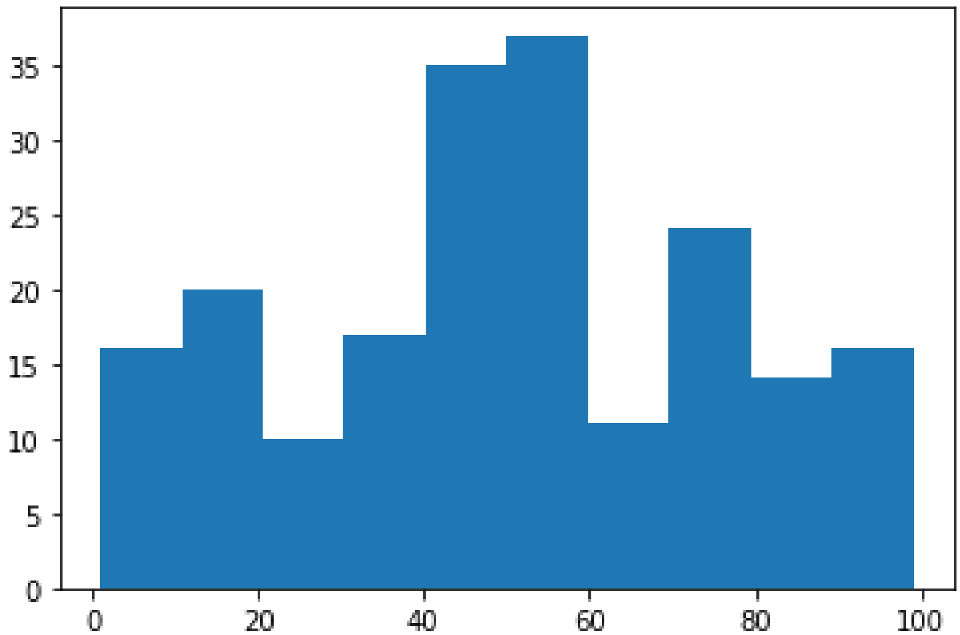
```
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4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

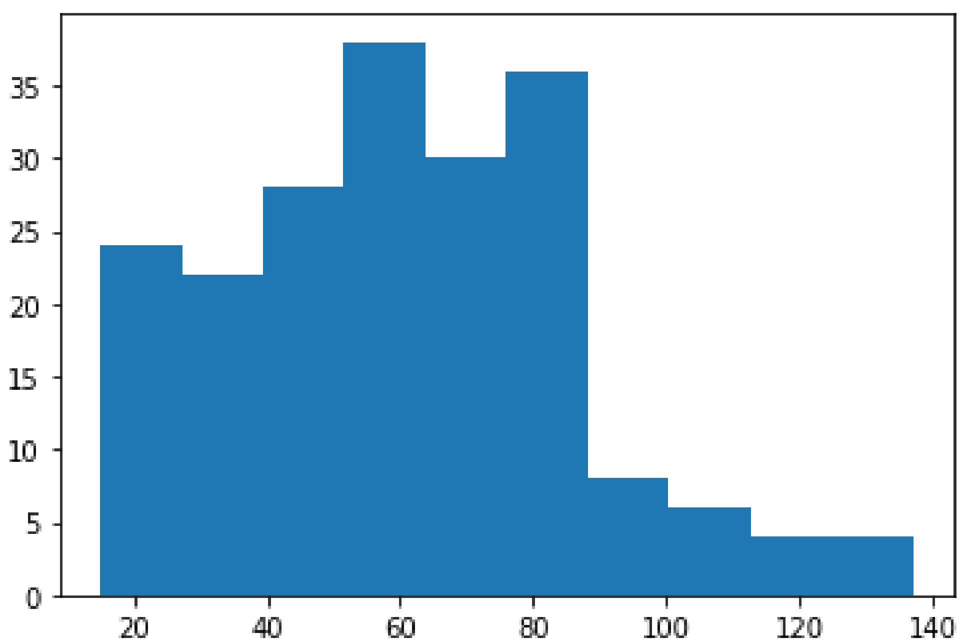
1. Perform Below Visualizations. Univariate Analysis Bivariate Analysis Multivariate Analysis

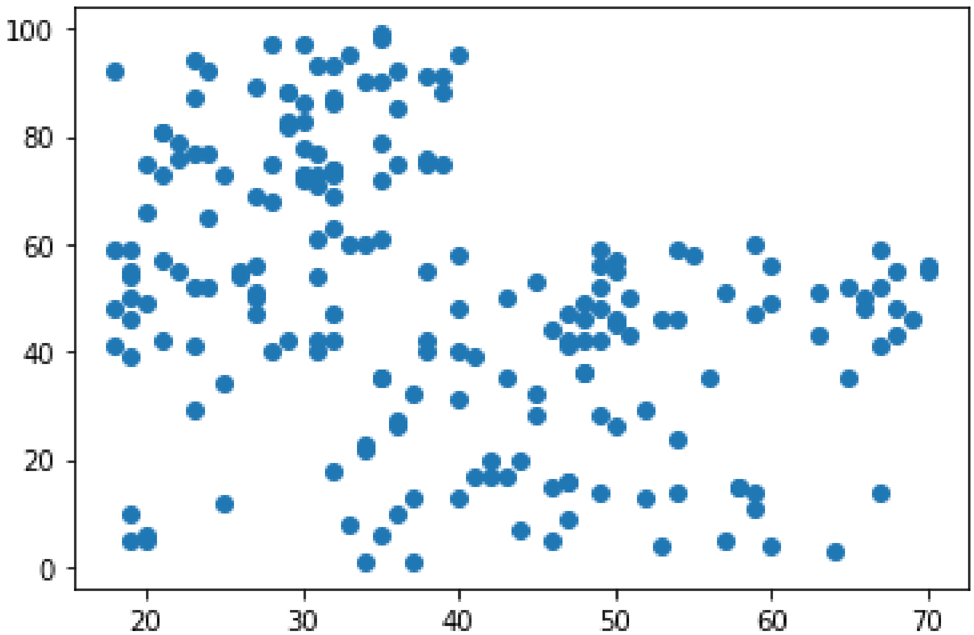
```
#Univariate Analysis
#Age
plt.hist(df['Age'])
plt.show()

#Spending Score
plt.hist(df['Spending Score (1-100)'])
plt.show()

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

#Bi-Variate Analysis
#Age vs Spending Score
plt.scatter(df['Age'],df['Spending Score (1-100)'])
plt.show()

#Age vs Annual Income
plt.scatter(df['Age'],df['Annual Income (k$)'])
plt.show()

#Spending Score vs Annual Income
plt.scatter(df['Spending Score (1-100)'],df['Annual Income (k$)'])
plt.show()

#Multi-Variate Analysis
#Age vs Spending Score vs Annual Income
from mpl_toolkits.mplot3d import Axes3D
fig=plt.figure()
```

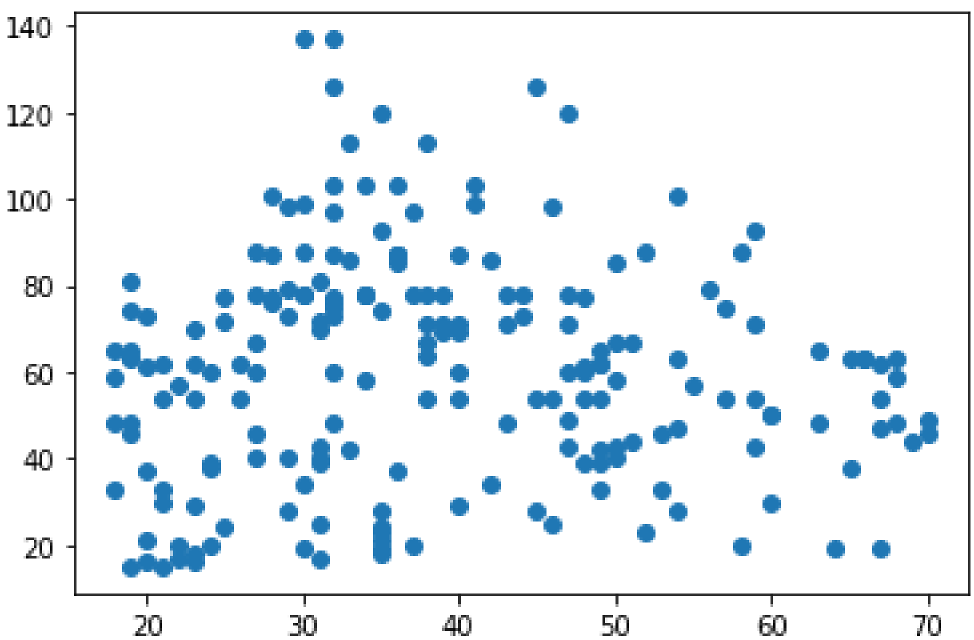
```
ax=fig.add_subplot(111,projection='3d')
ax.scatter(df['Age'],df['Spending Score (1-100)'],df['Annual Income
(k$)'])
plt.show()

#K-Means Clustering
#Age vs Spending Score
X=df.iloc[:,[2,4]].values
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.show()
```

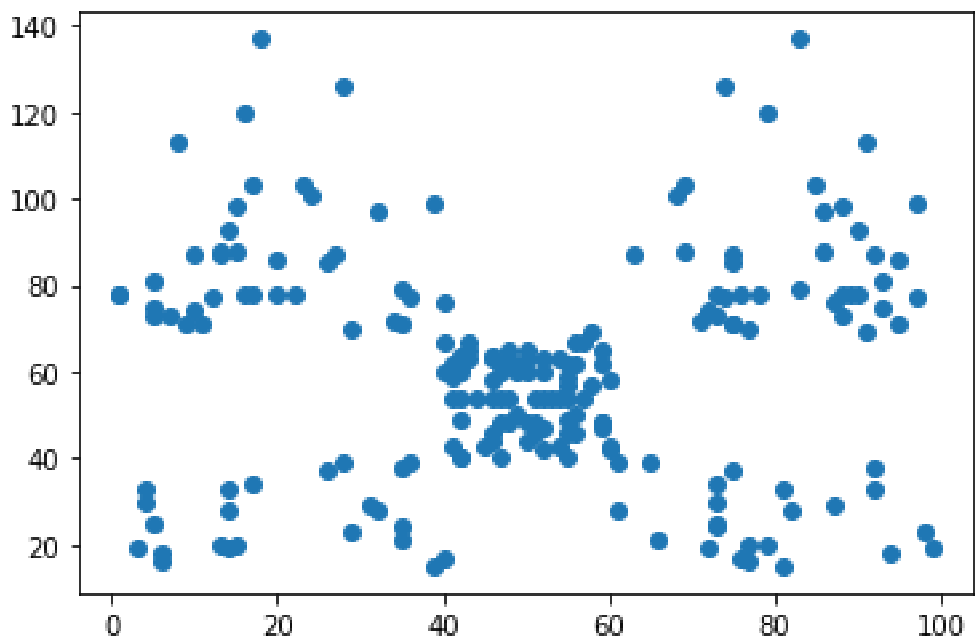


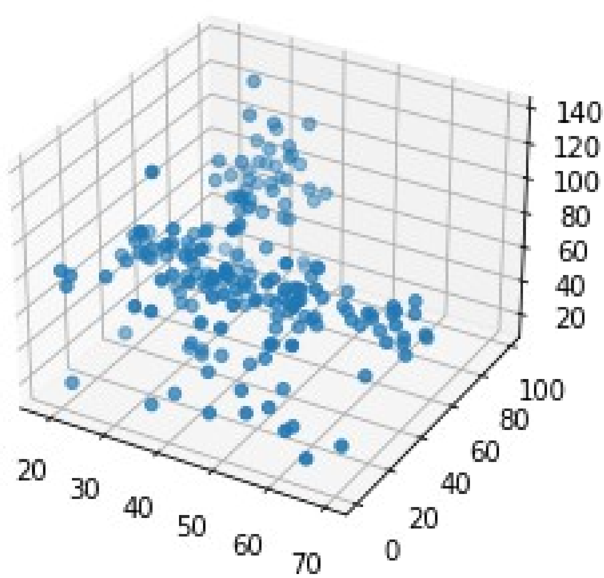

0
5
10
15
20
25
30
35
0
20
40
60
80
100

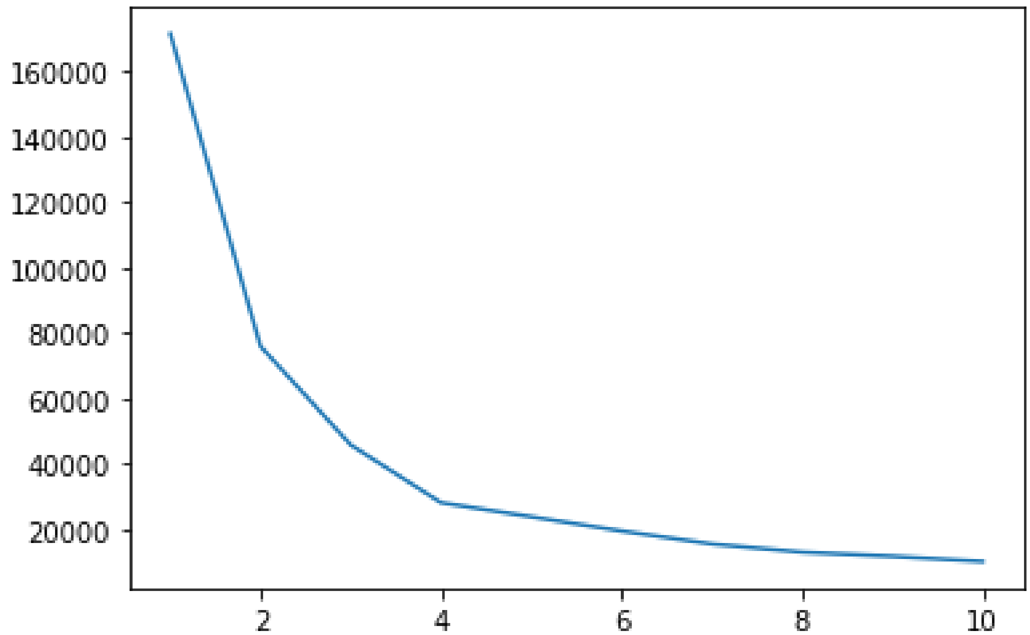

0
5
10
15
20
25
30
35
20
40
60
80
100
120
140


0
20
40
60
80
100
20
30
40
50
60
70


20
40
60
80
100
120
140
20
30
40
50
60
70


140
120
100
80
60
40
20
0
20
40
60
80
100


140
120
100
80
60
40
20
100
80
60
40
20
0
20
30
40
50
60
70



1. Perform descriptive statistics on the dataset.

```
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().sum()

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

df.columns

Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

```
df.shape
```

```
(200, 5)
```

```
df.dtypes
```

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

1. Check for Missing values and deal with them.

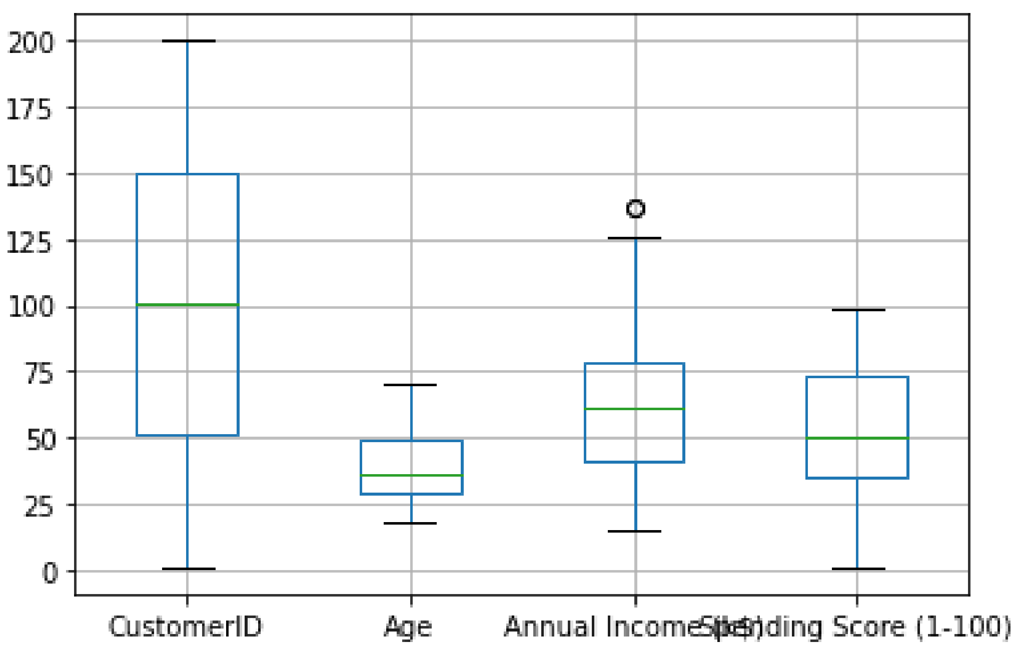
```
df.isnull().sum()
```

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

1. Find the outliers and replace them outliers

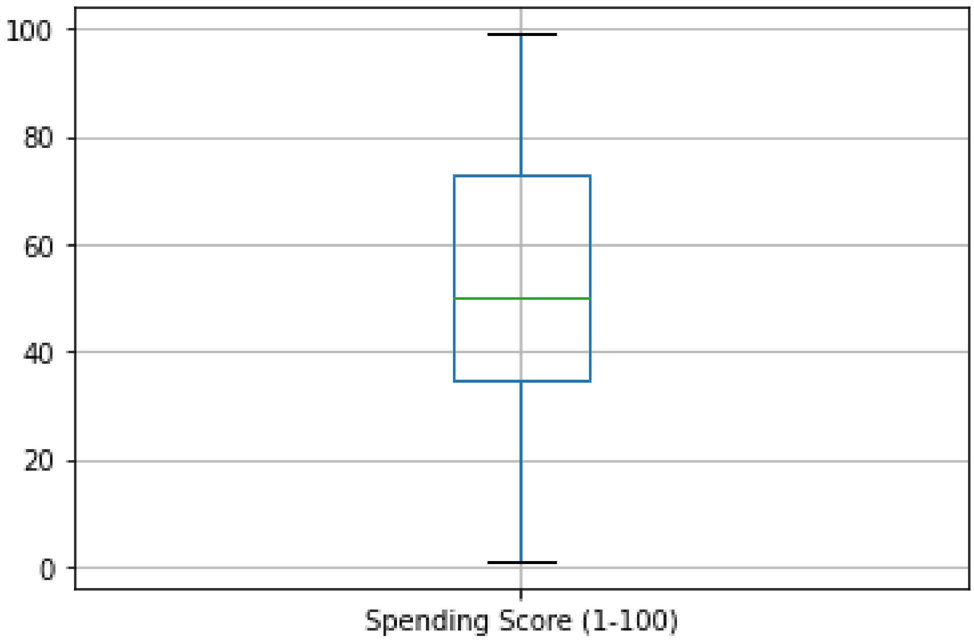
```
df.boxplot()
```

```
<AxesSubplot: >
```



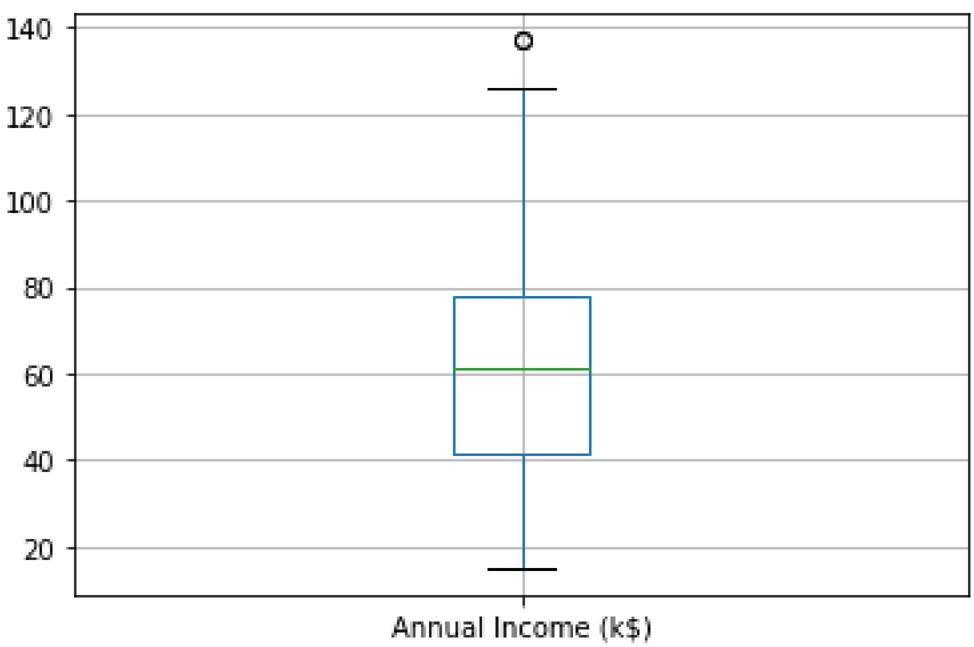

```
df.boxplot(column='Spending Score (1-100)')
```

```
<AxesSubplot: >
```



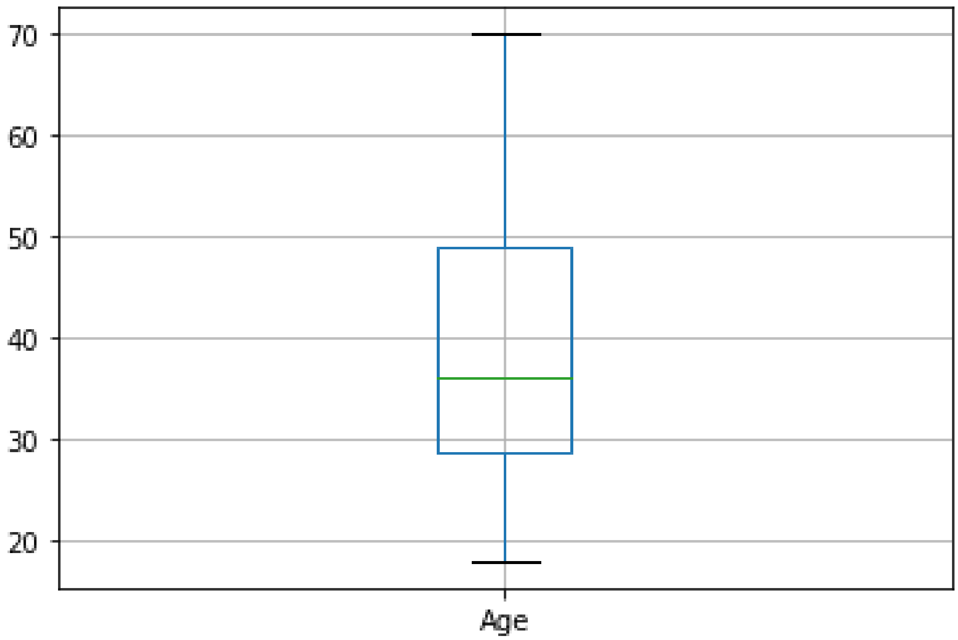

```
df.boxplot(column='Annual Income (k$)')
```

<AxesSubplot: >




```
df.boxplot(column='Age')
```

<AxesSubplot: >



1. Check for Categorical columns and perform encoding.

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

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

```
#Perform encoding for gender column

df=pd.get_dummies(df,columns=['Gender'])
df.head()
```

```
   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  

   Gender_Male  
0            1  
1            1  
2            0  
3            0  
4            0  
```

1. Scaling the data

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
df_scaled=scaler.fit_transform(df)
df_scaled
```

```
array([[-1.7234121 , -1.42456879, -1.73899919, -0.43480148, -
1.12815215,
         1.12815215],
       [-1.70609137, -1.28103541, -1.73899919,  1.19570407, -
1.12815215,
         1.12815215],
       [-1.68877065, -1.3528021 , -1.70082976, -1.71591298,
0.88640526,
        -0.88640526],
       ...,
       [ 1.68877065, -0.49160182,  2.49780745,  0.92395314, -
1.12815215,
         1.12815215],
       [ 1.70609137, -0.49160182,  2.91767117, -1.25005425, -
1.12815215,
         1.12815215],
       [ 1.7234121 , -0.6351352 ,  2.91767117,  1.27334719, -
1.12815215,
         1.12815215]])
```

1. Perform any of the clustering algorithms

```
from sklearn.cluster import KMeans
kmeans=KMeans(n_clusters=5,random_state=42)
kmeans.fit(df_scaled)
```

```
KMeans(n_clusters=5, random_state=42)
```

```
kmeans.labels_
```

```
array([4, 4, 2, 2, 2, 2, 2, 2, 3, 2, 3, 2, 2, 2, 3, 4, 2, 4, 3, 2, 4,
4,
       2, 4, 2, 4, 2, 4, 2, 2, 3, 2, 3, 4, 2, 2, 2, 2, 2, 2, 2, 4, 3,
2,
       2, 2, 2, 2, 2, 2, 2, 4, 2, 3, 2, 3, 2, 3, 2, 3, 3, 4, 2, 2, 3,
4,
       2, 2, 4, 2, 3, 2, 2, 2, 3, 4, 2, 3, 2, 2, 3, 4, 3, 2, 2, 3, 2,
2,
       2, 2, 2, 4, 3, 2, 2, 4, 2, 1, 3, 4, 1, 2, 3, 4, 3, 1, 2, 3, 3,
3,
       3, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 3, 0, 0,
0,
       1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
1,
       1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
1,
```

```
       0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
0,
       0, 0])
```

```
kmeans.cluster_centers_
```

```
array([[ 1.00571233, -0.25605987,  0.9485201 , -0.09735253, -
1.12815215,
         1.12815215],
       [ 0.88269077, -0.19763442,  0.8088102 ,  0.11840576,
0.88640526,
        -0.88640526],
       [-0.85997398,  0.07057058, -0.79430582, -0.00647026,
0.88640526,
        -0.88640526],
       [-0.53075649,  1.33075947, -0.48486081, -0.42786906, -
1.12815215,
         1.12815215],
       [-0.88871813, -1.01105596, -0.84837918,  0.47658087, -
1.12815215,
         1.12815215]])
```

1. Add the cluster data with the primary dataset

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

```
   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

   Gender_Male  Cluster
0            1        4
1            1        4
2            0        2
3            0        2
4            0        2
```

1. Split the data into dependent and independent variables.

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

1. Split the data into dependent and independent variables.

```
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)
```

1. Build the Model

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

```
c:\Users\harin\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\linear_model\_logistic.py:444: 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
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

1. Train the Model

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

```
0.95
```

1. Test the Model

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

```
0.95
```

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([[ 7,  0,  0,  0,  0],
       [ 0, 10,  0,  0,  0],
       [ 0,  0, 11,  0,  0],
       [ 1,  0,  0,  6,  0],
       [ 1,  0,  0,  0,  4]], dtype=int64)
```

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

```
              precision    recall  f1-score   support

           0       0.78      1.00      0.88         7
```

1	1.00	1.00	1.00	10
2	1.00	1.00	1.00	11
3	1.00	0.86	0.92	7
4	1.00	0.80	0.89	5
accuracy			0.95	40
macro avg	0.96	0.93	0.94	40
weighted avg	0.96	0.95	0.95	40