Assignment -4
Mall Customers

Assignment Date	24 September 2022
Student Name	Paul Jabez Talakayala
Student Roll Number	195002079
Maximum Marks	2 Marks

Load the dataset into the tool.

```
[29] import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
```

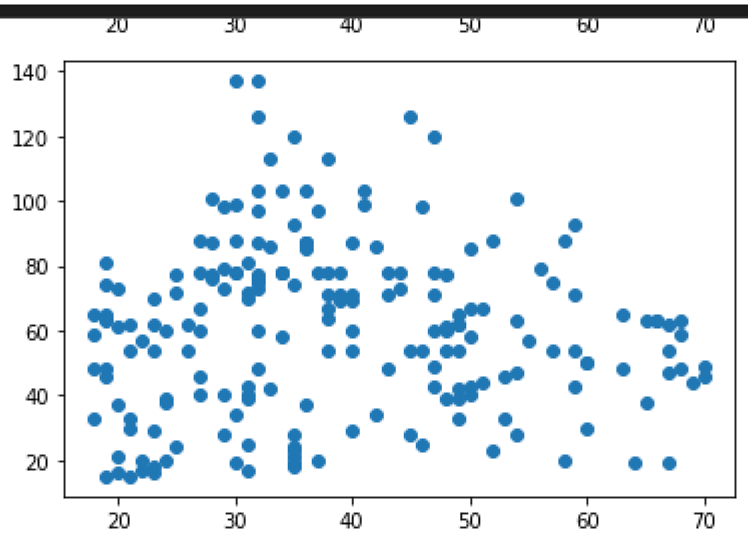
```
[30] df=pd.read_csv("Mall_Customers.csv")
     df.head()
```

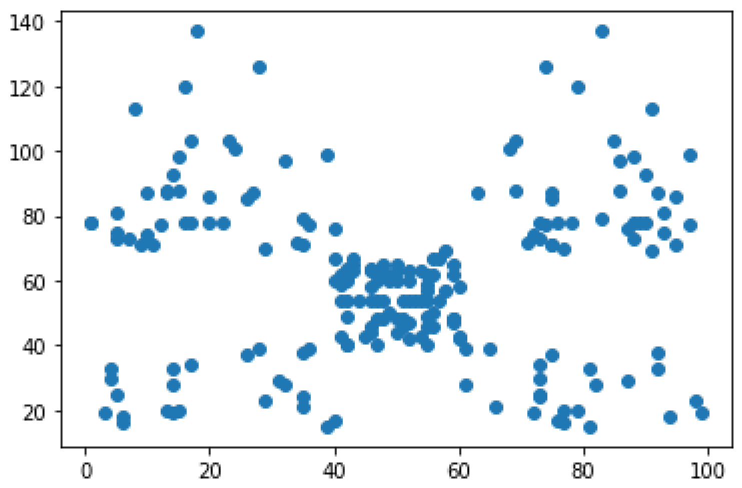
	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

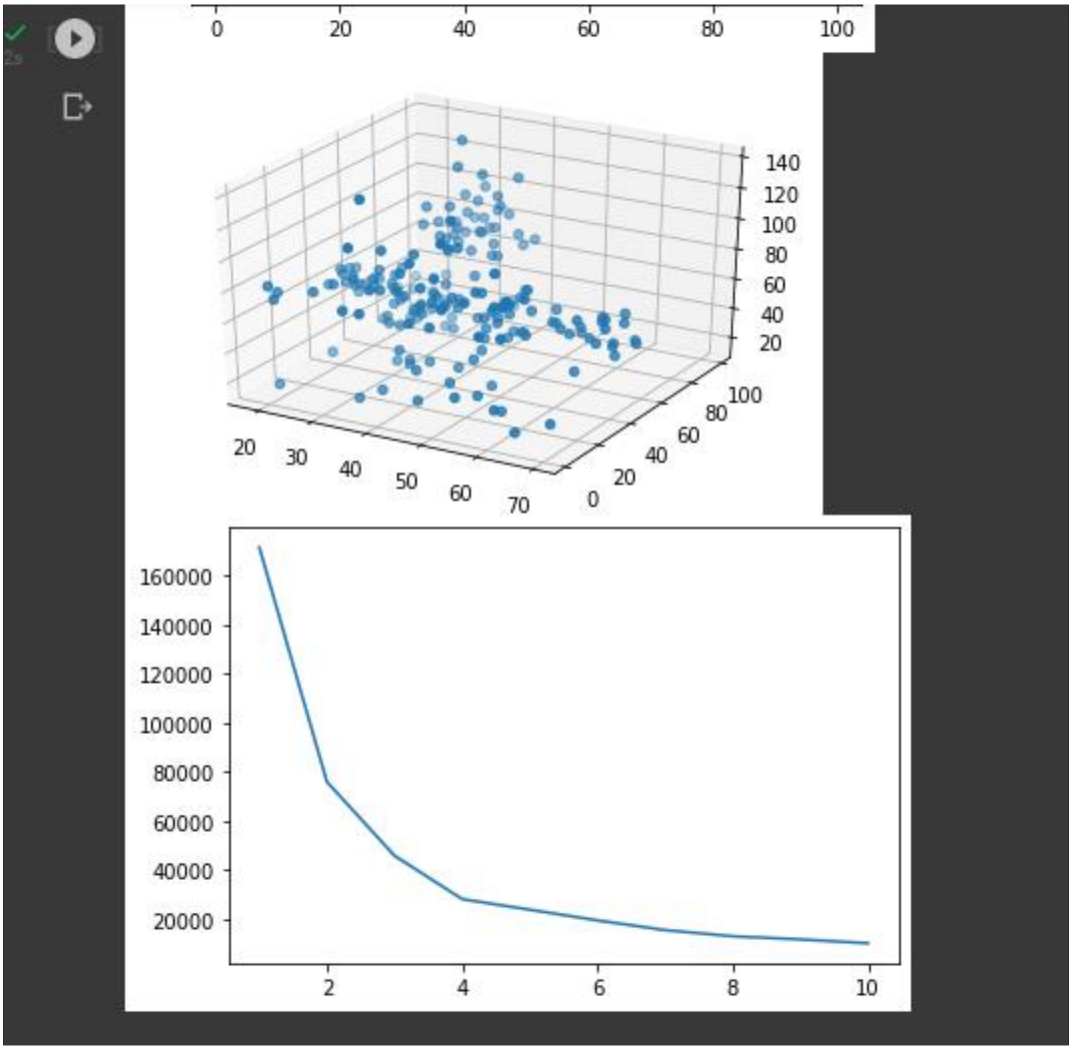
1.Perform Below Visualizations. Univariate Analysis Bivariate Analysis Multivariate Analysis

```
#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 20 40 60 80 100
140 120 100 80 60 40 20
20 30 40 50 60 70
0 20 40 60 80 100
160000 140000 120000 100000 80000 60000 40000 20000
2 4 6 8 10

Perform descriptive statistics on the dataset.

```
[32] 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   Genre                   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
```

```
[33] df.isnull().sum()
```

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

```
[34] df.columns
```

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

```
[35] df.shape
```

```
(200, 5)
```

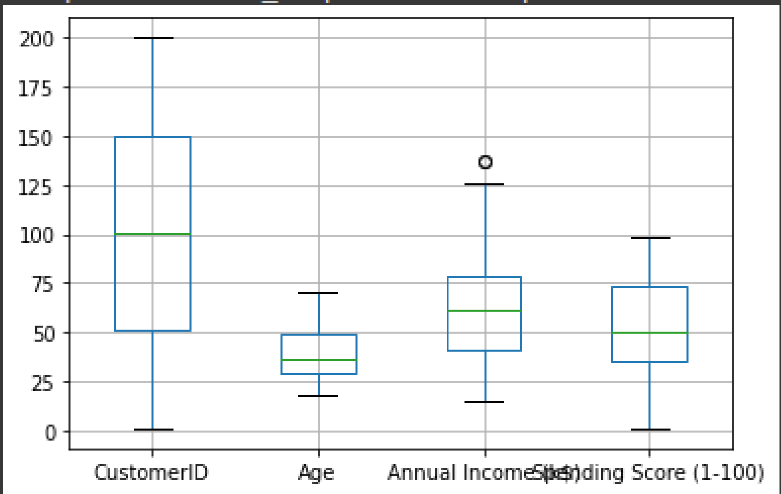
```
[36] df.dtypes
```

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

Find the outliers and replace them outliers

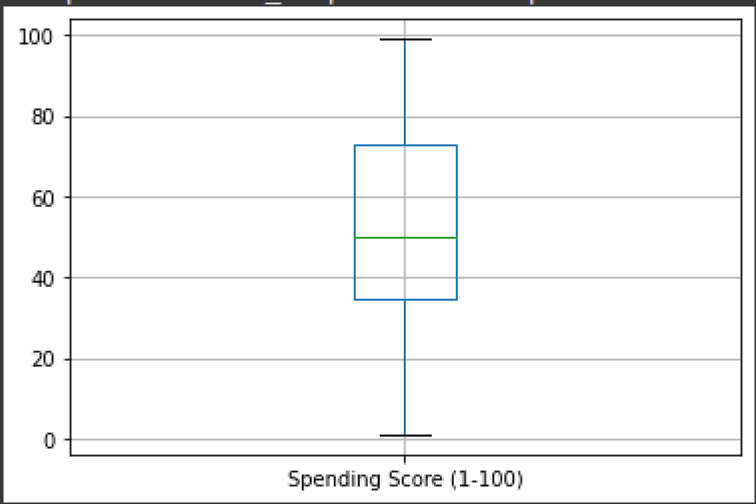
[37]
```
df.boxplot()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fe64dc80510>



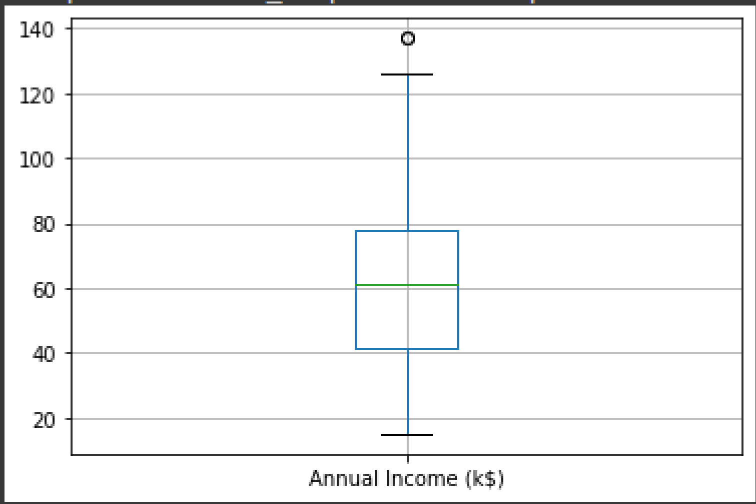
[38]
```
df.boxplot(column='Spending Score (1-100)')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fe64d53b990>



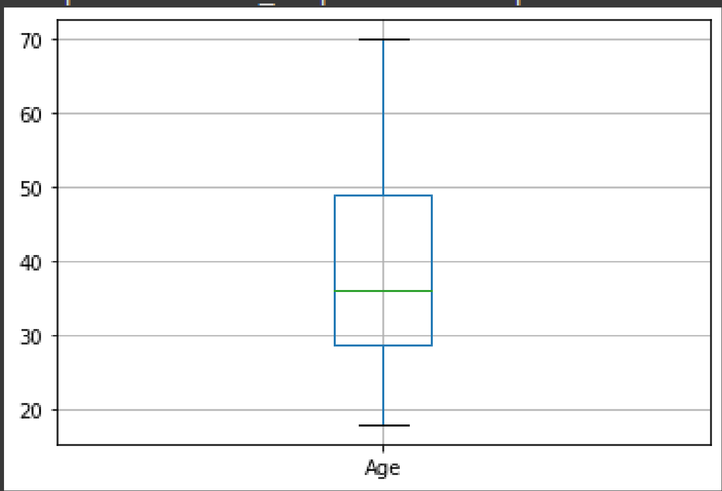
```
[39] df.boxplot(column='Annual Income (k$)')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe64d4b4f50>
```




```
[40] df.boxplot(column='Age')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe64d42cad0>
```



Check for Categorical columns and perform encoding.

```
[41] df.select_dtypes(include='object').columns
```

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

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

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

Scaling the data

```
[43] 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]])
```

Perform any of the clustering algorithms

```
[44] 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], dtype=int32)
```

Add the cluster data with the primary dataset

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

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

```
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]])
```

Split the data into dependent and independent variables.

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

Split the data into dependent and independent variables.

```
[49] 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)
```

Build the Model

```
[50] 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()
```

Train the Model

```
[51] model.score(X_train,y_train)
```

```
0.95
```

Test the Model

```
[52] model.score(X_test,y_test)
```

```
0.95
```

Measure the performance using Evaluation Metrics.

```
[53] 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]])
```

```
[54] 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
```