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

load the dataset

In [2]:

```
df=pd.read_csv("D:\\Users\ELCOT\Mall_Customers.csv")
```

In [3]:

```
df
```

Out[3]:

	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 × 5 columns

In [4]:

```
df.head()
```

Out[4]:

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

	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

In [5]:

```
df.tail()
```

Out[5]:

	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

Univariate Analysis

In [6]:

```
sns.displot(df.Age)
```

Out[6]:

In [7]:

```
df.hist('Age')
```

Out[7]:

```
array([[]], dtype=object)
```

Bivariate Analysis

In [8]:

```
sns.scatterplot(x=df. Age,y=df.Gender)
```

Out[8]:

In [9]:

```
sns.scatterplot(y=df.Age,x=df.Gender)
```

Out[9]:

Multivariate Analysis

In [10]:

```
sns.pairplot(df)
```

Out[10]:

Perform descriptive statistic on dataset

In [11]:

```
df.describe()
```

Out[11]:

	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

Check the missing values and deal with them

In [12]:

```
df.isna()
```

Out[12]:

	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 × 5 columns

In [13]:

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

Out[13]:

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

Find and replace the outliers

In [14]:

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

Out[14]:

check for categorical columns and perform encoding

In [20]:

```
x="Male"
y="Female"
df['Gender'].replace({'M':y,'F':x})
df
```

Out[20]:

	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 × 5 columns

In [24]:

```
df.tail()
```

Out[24]:

	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

Scaling the data

In [28]:

```
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
scale=StandardScaler()
```

In [34]:

```
X=df[['Age']]
scaledX=scale.fit_transform(X)
print(scaledX)
```

```
[[-1.42456879]
 [-1.28103541]
 [-1.3528021 ]
 [-1.13750203]
 [-0.56336851]
 [-1.20926872]
 [-0.27630176]
 [-1.13750203]
 [ 1.80493225]
 [-0.6351352 ]
 [ 2.02023231]
 [-0.27630176]
 [ 1.37433211]
 [-1.06573534]
 [-0.13276838]
 [-1.20926872]
 [-0.27630176]
 [-1.3528021 ]
 [ 0.94373197]
 [-0.27630176]
 [-0.27630176]
 [-0.99396865]
```

```
[ 0.51313183]
[-0.56336851]
[ 1.08726535]
[-0.70690189]
[ 0.44136514]
[-0.27630176]
[ 0.08253169]
[-1.13750203]
[ 1.51786549]
[-1.28103541]
[ 1.01549866]
[-1.49633548]
[ 0.7284319 ]
[-1.28103541]
[ 0.22606507]
[-0.6351352 ]
[-0.20453507]
[-1.3528021 ]
[ 1.87669894]
[-1.06573534]
[ 0.65666521]
[-0.56336851]
[ 0.7284319 ]
[-1.06573534]
[ 0.80019859]
[-0.85043527]
[-0.70690189]
[-0.56336851]
[ 0.7284319 ]
[-0.41983513]
[-0.56336851]
[ 1.4460988 ]
[ 0.80019859]
[ 0.58489852]
[ 0.87196528]
[ 2.16376569]
[-0.85043527]
[ 1.01549866]
[ 2.23553238]
[-1.42456879]
[ 2.02023231]
[ 1.08726535]
[ 1.73316556]
[-1.49633548]
[ 0.29783176]
[ 2.091999  ]
[-1.42456879]
[-0.49160182]
[ 2.23553238]
[ 0.58489852]
[ 1.51786549]
[ 1.51786549]
[ 1.4460988 ]
[-0.92220196]
[ 0.44136514]
[ 0.08253169]
[-1.13750203]
```

```
[ 0.7284319 ]
[ 1.30256542]
[-0.06100169]
[ 2.02023231]
[ 0.51313183]
[-1.28103541]
[ 0.65666521]
[ 1.15903204]
[-1.20926872]
[-0.34806844]
[ 0.80019859]
[ 2.091999  ]
[-1.49633548]
[ 0.65666521]
[ 0.08253169]
[-0.49160182]
[-1.06573534]
[ 0.58489852]
[-0.85043527]
[ 0.65666521]
[-1.3528021 ]
[-1.13750203]
[ 0.7284319 ]
[ 2.02023231]
[-0.92220196]
[ 0.7284319 ]
[-1.28103541]
[ 1.94846562]
[ 1.08726535]
[ 2.091999  ]
[ 1.94846562]
[ 1.87669894]
[-1.42456879]
[-0.06100169]
[-1.42456879]
[-1.49633548]
[-1.42456879]
[ 1.73316556]
[ 0.7284319 ]
[ 0.87196528]
[ 0.80019859]
[-0.85043527]
[-0.06100169]
[ 0.08253169]
[ 0.010765  ]
[-1.13750203]
[-0.56336851]
[ 0.29783176]
[ 0.08253169]
[ 1.4460988 ]
[-0.06100169]
[ 0.58489852]
[ 0.010765  ]
[-0.99396865]
[-0.56336851]
[-1.3528021 ]
[-0.70690189]
```

```
[ 0.36959845]
[-0.49160182]
[-1.42456879]
[-0.27630176]
[ 1.30256542]
[-0.49160182]
[-0.77866858]
[-0.49160182]
[-0.99396865]
[-0.77866858]
[ 0.65666521]
[-0.49160182]
[-0.34806844]
[-0.34806844]
[ 0.29783176]
[ 0.010765  ]
[ 0.36959845]
[-0.06100169]
[ 0.58489852]
[-0.85043527]
[-0.13276838]
[-0.6351352 ]
[-0.34806844]
[-0.6351352 ]
[ 1.23079873]
[-0.70690189]
[-1.42456879]
[-0.56336851]
[ 0.80019859]
[-0.20453507]
[ 0.22606507]
[-0.41983513]
[-0.20453507]
[-0.49160182]
[ 0.08253169]
[-0.77866858]
[-0.20453507]
[-0.20453507]
[ 0.94373197]
[-0.6351352 ]
[ 1.37433211]
[-0.85043527]
[ 1.4460988 ]
[-0.27630176]
[-0.13276838]
[-0.49160182]
[ 0.51313183]
[-0.70690189]
[ 0.15429838]
[-0.6351352 ]
[ 1.08726535]
[-0.77866858]
[ 0.15429838]
[-0.20453507]
[-0.34806844]
[-0.49160182]
[-0.41983513]
```

```
 [-0.06100169]
 [ 0.58489852]
 [-0.27630176]
 [ 0.44136514]
 [-0.49160182]
 [-0.49160182]
 [-0.6351352 ]]
```

Clustering algorithms

In [36]:

```
from sklearn import datasets

import warnings
warnings.filterwarnings("ignore")
```

In [37]:

```
df=datasets.load_iris()
dir(datasets)
```

Out[37]:

```
['__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 '_base',
 '_california_housing',
 '_covtype',
 '_kddcup99',
 '_lfw',
 '_olivetti_faces',
 '_openml',
 '_rcv1',
 '_samples_generator',
 '_species_distributions',
 '_svmlight_format_fast',
 '_svmlight_format_io',
 '_twenty_newsgroups',
 'clear_data_home',
 'dump_svmlight_file',
 'fetch_20newsgroups',
 'fetch_20newsgroups_vectorized',
 'fetch_california_housing',
 'fetch_covtype',
 'fetch_kddcup99',
 'fetch_lfw_pairs',
 'fetch_lfw_people',
 'fetch_olivetti_faces',
 'fetch_openml',
 'fetch_rcv1',
```

```
 'fetch_species_distributions',
 'get_data_home',
 'load_boston',
 'load_breast_cancer',
 'load_diabetes',
 'load_digits',
 'load_files',
 'load_iris',
 'load_linnerud',
 'load_sample_image',
 'load_sample_images',
 'load_svmlight_file',
 'load_svmlight_files',
 'load_wine',
 'make_biclusters',
 'make_blobs',
 'make_checkerboard',
 'make_circles',
 'make_classification',
 'make_friedman1',
 'make_friedman2',
 'make_friedman3',
 'make_gaussian_quantiles',
 'make_hastie_10_2',
 'make_low_rank_matrix',
 'make_moons',
 'make_multilabel_classification',
 'make_regression',
 'make_s_curve',
 'make_sparse_coded_signal',
 'make_sparse_spd_matrix',
 'make_sparse_uncorrelated',
 'make_spd_matrix',
 'make_swiss_roll']
```

In [38]:

```
print(df)
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
```

```
[5.4, 3.4, 1.7, 0.2],
[5.1, 3.7, 1.5, 0.4],
[4.6, 3.6, 1. , 0.2],
[5.1, 3.3, 1.7, 0.5],
[4.8, 3.4, 1.9, 0.2],
[5. , 3. , 1.6, 0.2],
[5. , 3.4, 1.6, 0.4],
[5.2, 3.5, 1.5, 0.2],
[5.2, 3.4, 1.4, 0.2],
[4.7, 3.2, 1.6, 0.2],
[4.8, 3.1, 1.6, 0.2],
[5.4, 3.4, 1.5, 0.4],
[5.2, 4.1, 1.5, 0.1],
[5.5, 4.2, 1.4, 0.2],
[4.9, 3.1, 1.5, 0.2],
[5. , 3.2, 1.2, 0.2],
[5.5, 3.5, 1.3, 0.2],
[4.9, 3.6, 1.4, 0.1],
[4.4, 3. , 1.3, 0.2],
[5.1, 3.4, 1.5, 0.2],
[5. , 3.5, 1.3, 0.3],
[4.5, 2.3, 1.3, 0.3],
[4.4, 3.2, 1.3, 0.2],
[5. , 3.5, 1.6, 0.6],
[5.1, 3.8, 1.9, 0.4],
[4.8, 3. , 1.4, 0.3],
[5.1, 3.8, 1.6, 0.2],
[4.6, 3.2, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[5. , 3.3, 1.4, 0.2],
[7. , 3.2, 4.7, 1.4],
[6.4, 3.2, 4.5, 1.5],
[6.9, 3.1, 4.9, 1.5],
[5.5, 2.3, 4. , 1.3],
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
```

```
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
[6.3, 2.7, 4.9, 1.8],
[6.7, 3.3, 5.7, 2.1],
[7.2, 3.2, 6. , 1.8],
[6.2, 2.8, 4.8, 1.8],
[6.1, 3. , 4.9, 1.8],
[6.4, 2.8, 5.6, 2.1],
[7.2, 3. , 5.8, 1.6],
[7.4, 2.8, 6.1, 1.9],
[7.9, 3.8, 6.4, 2. ],
[6.4, 2.8, 5.6, 2.2],
[6.3, 2.8, 5.1, 1.5],
```

```
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]]), 'target': 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, 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, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]), 'frame': Non
e, 'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='
```

In [39]:

```
dir(df)
```

Out[39]:

```
['DESCR',
 'data',
 'feature_names',
 'filename',
 'frame',
 'target',
 'target_names']
```

In [41]:

```
df.feature_names
```

Out[41]:

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [47]:

```
import matplotlib.pyplot as plt
```

In [50]:

```
x=[4,5,10,3,11,14,6,10,12,15]
y=[21,19,24,17,16,25,24,22,21,28]
plt.scatter(x,y)
plt.show()
```

In [51]:

```
from sklearn.cluster import KMeans
```

In [62]:

```
data=list(zip(x,y))
```

```
inertias=[]
for i in range(1,11):
    kmeans=KMeans(n_clusters=i)
    kmeans.fit(data)
    inertias. append(kmeans.inertia_)
plt.plot(range(1,11),inertias,marker='o')
plt.title("Elbow method")
plt.show()
```

In [63]:

```
kmeans=KMeans(n_clusters=2)
kmeans.fit(data)
plt.scatter(x,y,c=kmeans.labels_)
plt.show()
```

In [64]:

```
print(data)
```

```
[(4, 21), (5, 19), (10, 24), (3, 17), (11, 16), (14, 25), (6, 24), (10, 22)
, (12, 21), (15, 28)]
```