In [2]:
```python
import pandas
```

In [10]:
```python
dataset = pandas.read_csv('Mall_Customers.csv')
```

In [11]:
```python
dataset.head()
```

Out[11]:

	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 [12]:
```python
new_dataset=dataset.iloc[:,:-1]
new_dataset.head()
```

Out[12]:

	CustomerID	Gender	Age	Annual Income (k$)
0	1	Male	19	15
1	2	Male	21	15
2	3	Female	20	16
3	4	Female	23	16
4	5	Female	31	17

In [14]:
```python
new_dataset.shape
```

Out[14]:
```
(200, 4)
```

In [15]:
```python
dataset.tail()
```

Out[15]:

	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

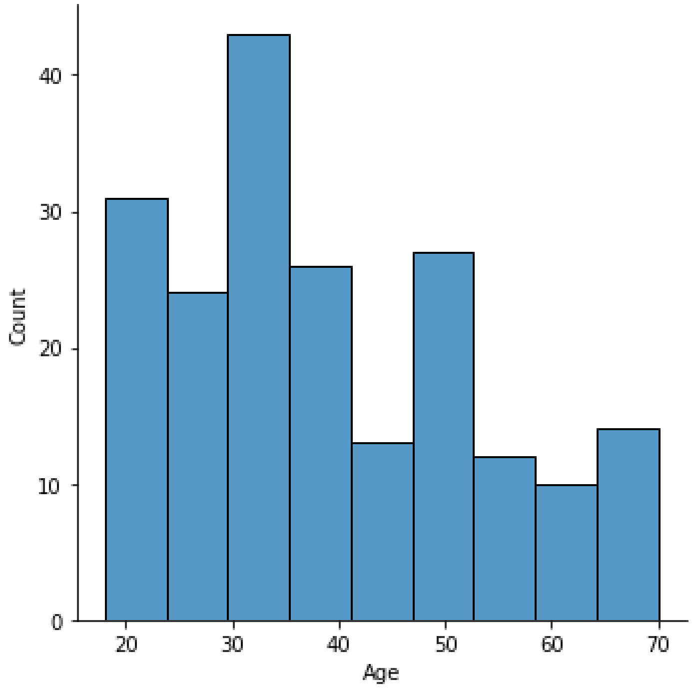
In [16]:
```python
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [17]:
```python
import seaborn as sns
```

In [18]:
```python
import matplotlib.pyplot as plt
```

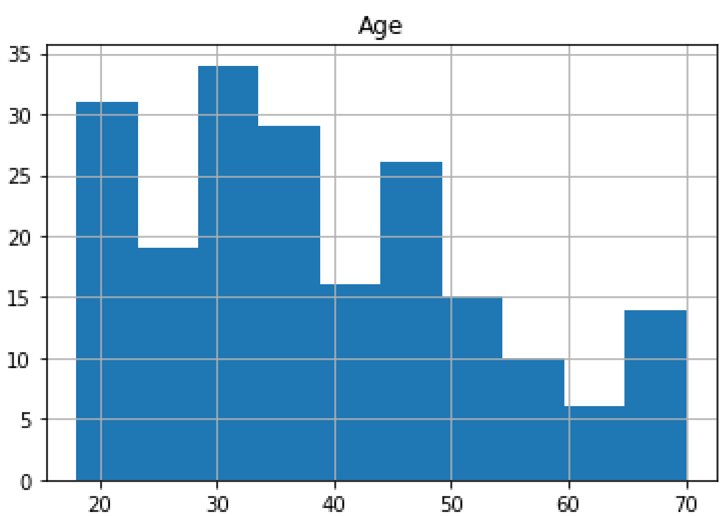
In [20]:
```python
sns.displot(dataset.Age)
```

Out[20]:
```
<seaborn.axisgrid.FacetGrid at 0x27f0a3409d0>
```



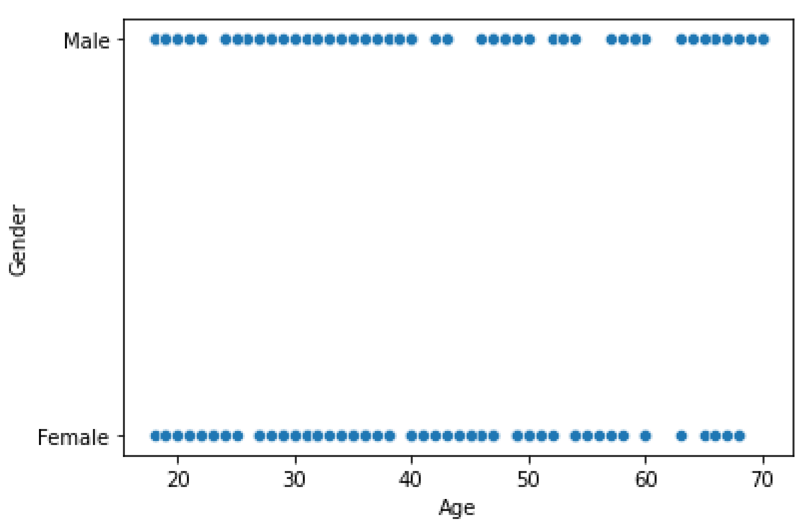
In [22]:
```
dataset.hist('Age')
```

Out[22]:
```
array([[<AxesSubplot:title={'center':'Age'}>]], dtype=object)
```



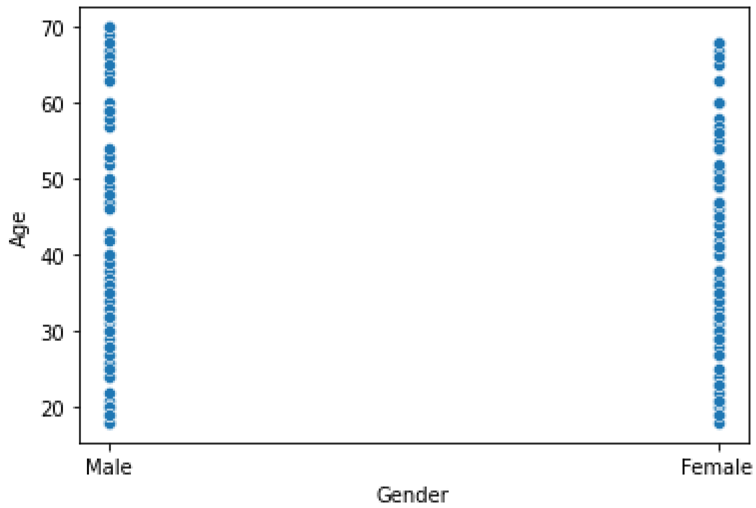
In [23]:
```
sns.scatterplot(x=dataset.Age,y=dataset.Gender)
```

Out[23]:
```
<AxesSubplot:xlabel='Age', ylabel='Gender'>
```



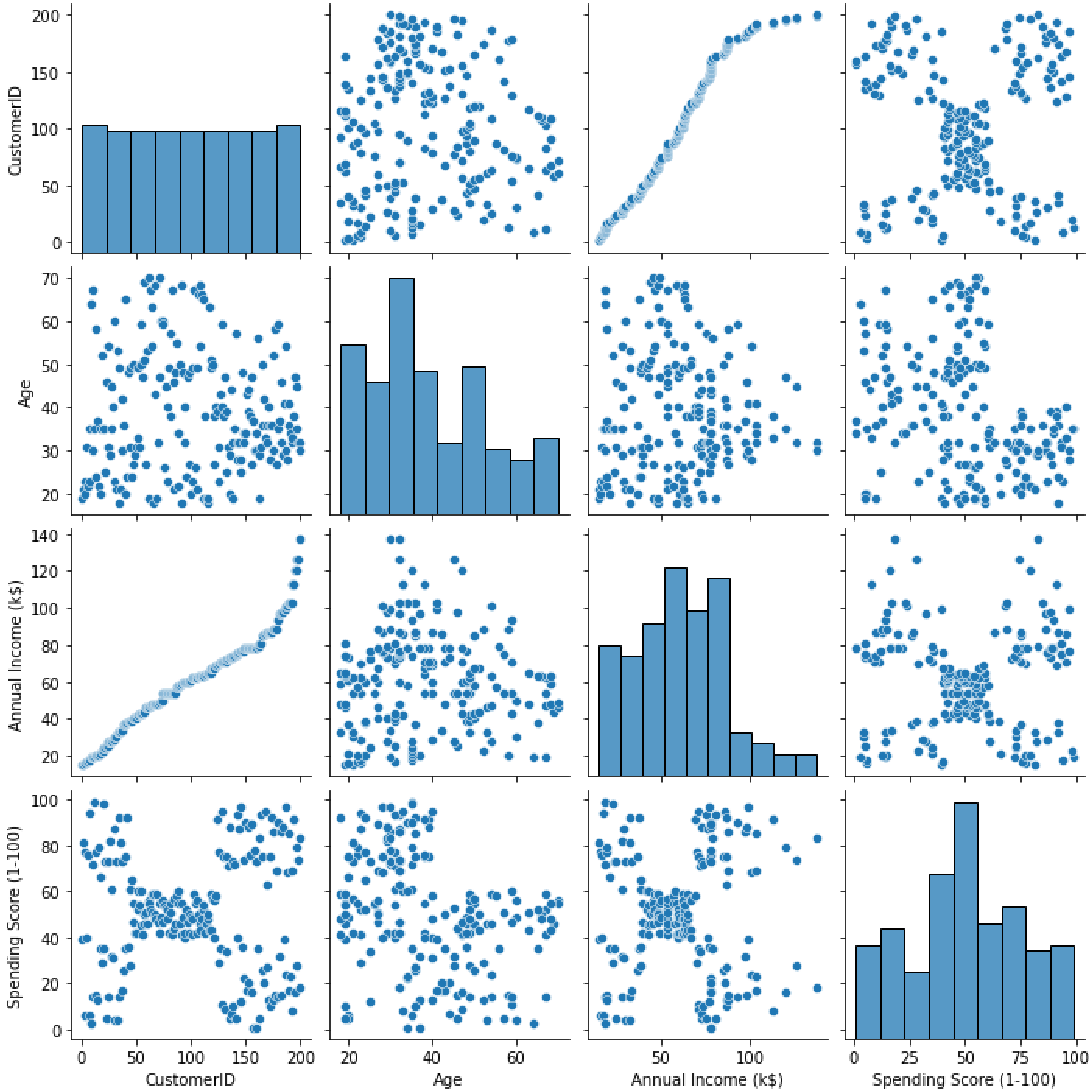
In [24]:
```
sns.scatterplot(y=dataset.Age,x=dataset.Gender)
```

Out[24]: <AxesSubplot:xlabel='Gender', ylabel='Age'>



In [25]:
```
sns.pairplot(dataset)
```

Out[25]: <seaborn.axisgrid.PairGrid at 0x27f0dd185e0>



In [26]:
```
dataset.describe()
```

Out[26]:

	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 deals withthem

In [27]:
```
dataset.isna()
```

Out[27]:

	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

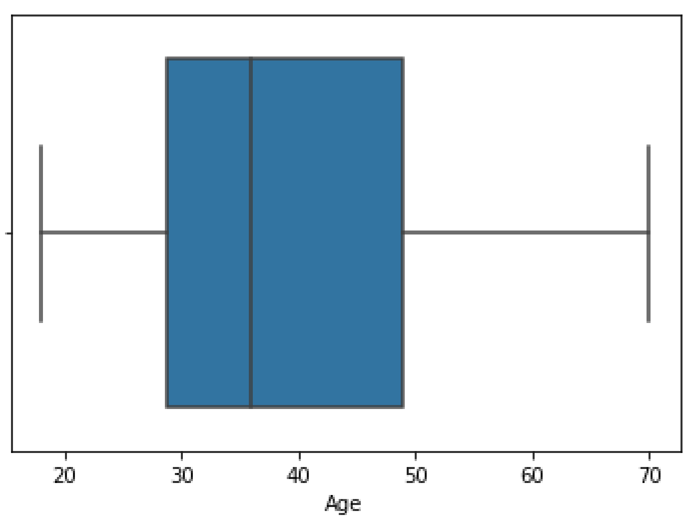
In [28]:
```
dataset.isnull().sum()
```

Out[28]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

find and replace the outliers

In [29]:
```
sns.boxplot(x=dataset['Age'])
```

Out[29]:
```
<AxesSubplot:xlabel='Age'>
```



check for categorial columns and performs encoding

In [31]:
```
x="Male"
y="Female"
dataset['Gender'].replace({'M':y,'F':x})
dataset
```

Out[31]:

	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 [32]:
```
dataset.tail()
```

Out[32]:

	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 [33]:
```
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
scale=StandardScaler()
```

In [35]:
```
x=dataset[['Age']]
scaledataset=scale.fit_transform(x)
print(scaledataset)
```

```
[[-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]:
```python
from sklearn import datasets
```

In [38]:
```python
import warnings
warnings.filterwarnings("ignore")
```

In [39]:
```python
dataset=datasets.load_iris()
dir(datasets)
```

Out[39]:
```
['__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',
 'data',
 'descr',
 '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 [40]:
```
print(dataset)
```

Loading [MathJax]/extensions/Safe.js

```
{'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': None, 'target_na
mes': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'), 'DESCR': '.. _iris_dat
aset:\n\nIris plants dataset\n--------------------\n\n**Data Set Characteristics:**\n\n
:Number of Instances: 150 (50 in each of three classes)\n    :Number of Attributes: 4 nu
meric, predictive attributes and the class\n    :Attribute Information:\n        - sepal
length in cm\n        - sepal width in cm\n        - petal length in cm\n        - petal
width in cm\n        - class:\n                - Iris-Setosa\n                - Iris-Ver
sicolour\n                - Iris-Virginica\n                \n    :Summary Statistics:\n
\n    ============== ==== ==== ======= ===== ====================\n                    M
in  Max   Mean    SD   Class Correlation\n    ============== ==== ==== ======= ===== ===
=================\n    sepal length:   4.3  7.9   5.84   0.83    0.7826\n    sepal widt
h:    2.0  4.4   3.05   0.43   -0.4194\n    petal length:   1.0  6.9   3.76   1.76    0.
9490  (high!)\n    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)\n    ======
======== ==== ==== ======= ===== ====================\n\n    :Missing Attribute Values:
None\n    :Class Distribution: 33.3% for each of 3 classes.\n    :Creator: R.A. Fisher\n
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThe fa
mous Iris database, first used by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s
paper. Note that it\'s the same as in R, but not as in the UCI\nMachine Learning Reposit
ory, which has two wrong data points.\n\nThis is perhaps the best known database to be f
ound in the\npattern recognition literature.  Fisher\'s paper is a classic in the field
and\nis referenced frequently to this day.  (See Duda & Hart, for example.)  The\ndata s
et contains 3 classes of 50 instances each, where each class refers to a\ntype of iris p
lant.  One class is linearly separable from the other 2; the\nlatter are NOT linearly se
parable from each other.\n\n.. topic:: References\n\n   - Fisher, R.A. "The use of multi
ple measurements in taxonomic problems"\n     Annual Eugenics, 7, Part II, 179-188 (193
6); also in "Contributions to\n     Mathematical Statistics" (John Wiley, NY, 1950).\n
- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.\n     (Q32
7.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.\n   - Dasarathy, B.V. (198
0) "Nosing Around the Neighborhood: A New System\n     Structure and Classification Rule
for Recognition in Partially Exposed\n     Environments".  IEEE Transactions on Pattern
Analysis and Machine\n     Intelligence, Vol. PAMI-2, No. 1, 67-71.\n   - Gates, G.W. (1
972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions\n     on Information Theor
y, May 1972, 431-433.\n   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AU
TOCLASS II\n     conceptual clustering system finds 3 classes in the data.\n   - Many, m
any more ...', 'feature_names': ['sepal length (cm)', 'sepal width (cm)', 'petal length
```

(cm)', 'petal width (cm)'], 'filename': 'iris.csv', 'data_module': 'sklearn.datasets.dat
a'}

In [41]:
```
dir(dataset)
```

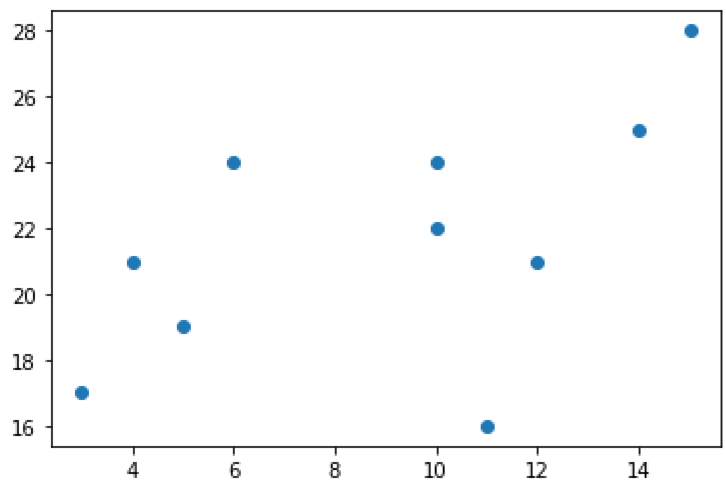
Out[41]:
```
['DESCR',
 'data',
 'data_module',
 'feature_names',
 'filename',
 'frame',
 'target',
 'target_names']
```

In [43]:
```
dataset.feature_names
```

Out[43]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [44]:
```
import matplotlib.pyplot as plt
```

In [45]:
```
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()
```

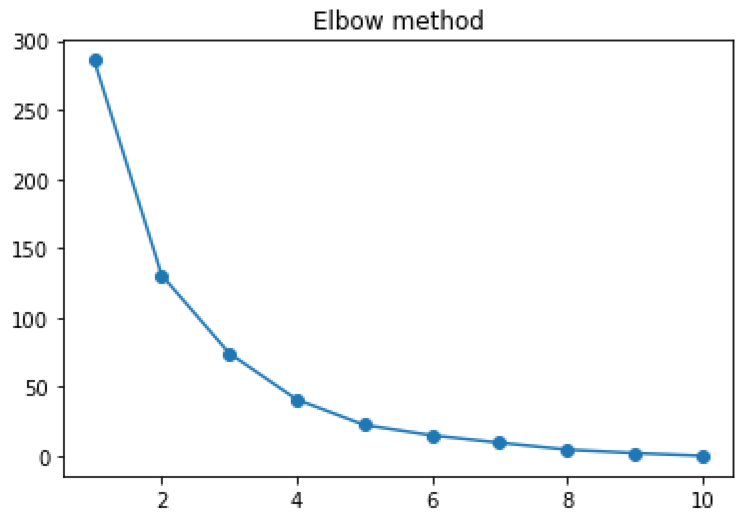


In [46]:
```
from sklearn.cluster import KMeans
```

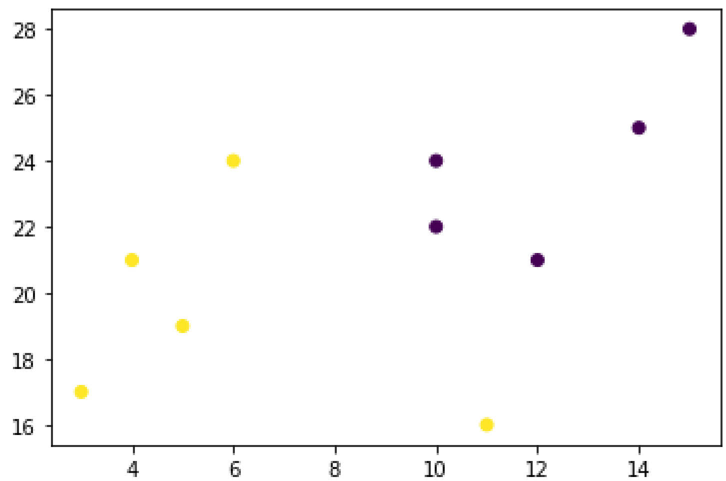
In [47]:
```
data=list(zip(x,y))
```

In [56]:
```
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()
```

Loading [MathJax]/extensions/Safe.js



In [58]:
```
kmeans=KMeans(n_clusters=2)
kmeans.fit(data)
plt.scatter(x,y,c=kmeans.labels_)
plt.show()
```



In [59]:
```
print(data)
```

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

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: