Name: Om Suryawanshi

Div: B

Roll No.: 55

Practical No:1

In [7]:

```
import pandas as pd
df=pd.read_csv("Desktop/Iris.csv")
df
```

Out[7]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [8]:

```
df.head(5)
```

Out[8]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [9]:

```
df.tail(5)
```

Out[9]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [11]:

```
df.describe()
```

Out[11]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [12]:

```
df.shape
```

Out[12]:

```
(150, 6)
```

In [15]:

```
import pandas as pd
import numpy as np
df = pd.DataFrame({'num_legs': [2, 4, 8, 0],
                   'num_wings': [2, 0, 0, 0],
                   'num_specimen_seen': [10, 2, 1, 8]},
                  index=['falcon', 'dog', 'spider', 'fish'])
df
```

Out[15]:

	num_legs	num_wings	num_specimen_seen
falcon	2	2	10
dog	4	0	2
spider	8	0	1
fish	0	0	8

In [16]:

```
import pandas as pd
df=pd.read_csv("Desktop/Iris.csv")
df.isnull()
```

Out[16]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
145	False	False	False	False	False	False
146	False	False	False	False	False	False
147	False	False	False	False	False	False
148	False	False	False	False	False	False
149	False	False	False	False	False	False

150 rows × 6 columns

In [18]:

```
df.isna
```

Out[18]:

```
<bound method DataFrame.isna of       Id  SepalLengthCm  SepalWidthCm  Pet
alLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]>
```

In [ ]:

```

```

Practical No:2

In [2]:

```
import pandas as pd
df=pd.read_csv("Desktop/StudentsPerformance.csv")
df
```

Out[2]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95
996	male	group C	high school	free/reduced	none	62	55	55
997	female	group C	high school	free/reduced	completed	59	71	65
998	female	group D	some college	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64 
 6   reading score                1000 non-null   int64 
 7   writing score                1000 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

In [4]:

```
df.describe()
```

Out[4]:

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

In [5]:

```
df.isnull()
```

Out[5]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
995	False	False	False	False	False	False	False	False
996	False	False	False	False	False	False	False	False
997	False	False	False	False	False	False	False	False
998	False	False	False	False	False	False	False	False
999	False	False	False	False	False	False	False	False

1000 rows × 8 columns

In [10]:

```
df.notnull()
```

Out[10]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	True	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True	True
...	...	...	...	...	...	...	...	...
995	True	True	True	True	True	True	True	True
996	True	True	True	True	True	True	True	True
997	True	True	True	True	True	True	True	True
998	True	True	True	True	True	True	True	True
999	True	True	True	True	True	True	True	True

1000 rows × 8 columns

In [14]:

```
series = pd.notnull(df["math score"])
df [series]
```

Out[11]:

	gender	race/ethnicity	parental le el of educ tion	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bach lor's c gree	standard	none	72	72	74
1	female	group C	some cc llege	standard	completed	69	90	88
2	female	group B	ma ter's d gree	standard	none	90	95	93
3	male	group A	assoc ate's d gree	free/reduced	none	47	57	44
4	male	group C	some cc llege	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	female	group E	ma ter's d gree	standard	completed	88	99	95
996	male	group C	high s hool	free/reduced	none	62	55	55
997	female	group C	high s hool	free/reduced	completed	59	71	65
998	female	group D	some cc llege	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

In [15]:

```
m_v=df['math score'] .mean()
df['math score'].fillna(value=m_v, inplace=True)
df
```

Out[12]:

	gender	race/ethnicity	parental le el of educ tion	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bach lor's c gree	standard	none	72	72	74
1	female	group C	some cc llege	standard	completed	69	90	88
2	female	group B	ma ter's d gree	standard	none	90	95	93
3	male	group A	assoc ate's d gree	free/reduced	none	47	57	44
4	male	group C	some cc llege	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	female	group E	ma ter's d gree	standard	completed	88	99	95
996	male	group C	high s hool	free/reduced	none	62	55	55
997	female	group C	high s hool	free/reduced	completed	59	71	65
998	female	group D	some cc llege	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

In [15]:

```
new_data = df.dropna(axis = 0, how ='any')

new_data
```

Out[15]:

	gender	race/ethnicity	parental le el of educ tion	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bach lor's c gree	standard	none	72	72	74
1	female	group C	some cc llege	standard	completed	69	90	88
2	female	group B	ma ter's d gree	standard	none	90	95	93
3	male	group A	assoc ate's d gree	free/reduced	none	47	57	44
4	male	group C	some cc llege	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	female	group E	ma ter's d gree	standard	completed	88	99	95
996	male	group C	high s hool	free/reduced	none	62	55	55
997	female	group C	high s hool	free/reduced	completed	59	71	65
998	female	group D	some cc llege	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Practical No:3

In [3]:
```
import pandas as pd
df=pd.read_csv('/home/student/Downloads/archive/iris.csv')
print(df.shape)
```

```
(150, 5)
```

In [ ]:

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   sepal_length   150 non-null    float64
 1   sepal_width    150 non-null    float64
 2   petal_length   150 non-null    float64
 3   petal_width    150 non-null    float64
 4   species        150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [6]:
```
df.mean()
```

Out[6]:
```
sepal_length    5.843333
sepal_width     3.054000
petal_length    3.758667
petal_width     1.198667
dtype: float64
```

In [7]:
```
df.mode()
```

Out[7]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.0	3.0	1.5	0.2	setosa
1	NaN	NaN	NaN	NaN	versicolor
2	NaN	NaN	NaN	NaN	virginica

In [8]:
```
df.median()
```

Out[8]:
```
sepal_length    5.80
sepal_width     3.00
petal_length    4.35
petal_width     1.30
dtype: float64
```

In [9]:
```
print(df.loc[:,'sepal_length'].mean())
```

```
5.843333333333335
```

In [10]:
```
df.std()
```

Out[10]:
```
sepal_length    0.828066
sepal_width     0.433594
petal_length    1.764420
petal_width     0.763161
dtype: float64
```

In [11]:
```
df.var()
```

Out[11]:
```
sepal_length    0.685694
sepal_width     0.188004
petal_length    3.113179
petal_width     0.582414
dtype: float64
```

In [18]:
```
df.std(axis=1)[0:5]
```

Out[18]:
```
0    2.179449
1    2.036950
2    1.997498
3    1.912241
4    2.156386
dtype: float64
```

In [13]:
```
from scipy.stats import iqr
iqr(df['sepal_length'])
```

Out[13]:
```
1.3000000000000007
```

In [14]:
```
df.skew()
```

Out[14]:
```
sepal_length    0.314911
sepal_width     0.334053
petal_length   -0.274464
petal_width    -0.104997
dtype: float64
```

In [15]:
```
df.describe()
```

Out[15]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [16]:
```
df.describe(include='all')
```

Out[16]:

	sepal_length	sepal_width	petal_length	petal_width	species
count	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	virginica
freq	NaN	NaN	NaN	NaN	50
mean	5.843333	3.054000	3.758667	1.198667	NaN
std	0.828066	0.433594	1.764420	0.763161	NaN
min	4.300000	2.000000	1.000000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.350000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.400000	6.900000	2.500000	NaN

In [ ]:
```

```

Practical No:4

In [1]:
```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


# Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Importing Data
from sklearn.datasets import load_boston
boston = load_boston()


# In[2]:


boston.data.shape


# In[3]:


boston.feature_names


# In[4]:


data = pd.DataFrame(boston.data)
data.columns = boston.feature_names

data.head(10)


# In[5]:


# Adding 'Price' (target) column to the data
boston.target.shape


# In[6]:


data['Price'] = boston.target
data.head()


# In[7]:


data.describe()


# In[8]:
```

```
data.info()
```

```
# In[9]:
```

```
# Input Data
x = boston.data

# Output Data
y = boston.target
```

```
# splitting data to training and testing dataset.

#from sklearn.cross_validation import train_test_split
#the submodule cross_validation is renamed and deprecated to model_s
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2
                                                random_state = 0

print("xtrain shape : ", xtrain.shape)
print("xtest shape  : ", xtest.shape)
print("ytrain shape : ", ytrain.shape)
print("ytest shape  : ", ytest.shape)
```

```
# In[10]:
```

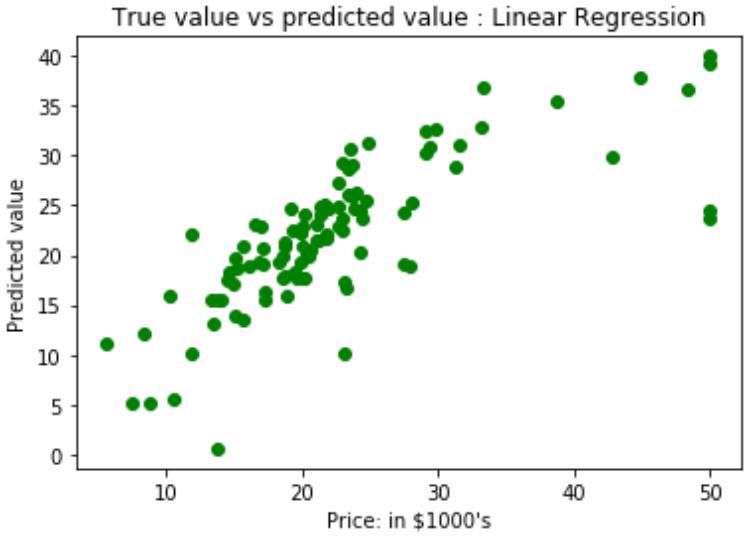
```
# Fitting Multi Linear regression model to training model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(xtrain, ytrain)

# predicting the test set results
y_pred = regressor.predict(xtest)
```

```
# In[11]:
```

```
# Plotting Scatter graph to show the prediction
# results - 'ytrue' value vs 'y_pred' value
plt.scatter(ytest, y_pred, c = 'green')
plt.xlabel("Price: in $1000's")
plt.ylabel("Predicted value")
plt.title("True value vs predicted value : Linear Regression")
plt.show()
```

```
# In[ ]:
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  Price    506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
xtrain shape :  (404, 13)
xtest shape  :  (102, 13)
ytrain shape :  (404,)
ytest shape  :  (102,)
```



In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Importing Data
from sklearn.datasets import load_boston
boston = load_boston()


# In[2]:


boston.data.shape
```

Out[3]:
```
(506, 13)
```

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Importing Data
from sklearn.datasets import load_boston
boston = load_boston()


# In[2]:
boston.feature names


# In[4]:


#data = pd.DataFrame(boston.data)
#data.columns = boston.feature_names

#data.head(10)
```

Out[4]:
```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'R
AD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

In [7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Importing Data
from sklearn.datasets import load_boston
boston = load_boston()


data = pd.DataFrame(boston.data)
data.columns = boston.feature_names

data.head(10)
```

Out[7]:

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
00.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
10.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
20.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
30.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
40.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90
50.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12
60.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60
70.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.90
80.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0	15.2	386.63
90.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0	15.2	386.71

In [8]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Importing Data
from sklearn.datasets import load_boston
boston = load_boston()


data = pd.DataFrame(boston.data)
data.columns = boston.feature_names

data.tail(20)
```

Out[8]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
4865.69175	0.0	18.10	0.0	0.583	6.114	79.8	3.5459	24.0	666.0	20.2	392.68
4874.83567	0.0	18.10	0.0	0.583	5.905	53.2	3.1523	24.0	666.0	20.2	388.22
4880.15086	0.0	27.74	0.0	0.609	5.454	92.7	1.8209	4.0	711.0	20.1	395.09
4890.18337	0.0	27.74	0.0	0.609	5.414	98.3	1.7554	4.0	711.0	20.1	344.05
4900.20746	0.0	27.74	0.0	0.609	5.093	98.0	1.8226	4.0	711.0	20.1	318.43
4910.10574	0.0	27.74	0.0	0.609	5.983	98.8	1.8681	4.0	711.0	20.1	390.11
4920.11132	0.0	27.74	0.0	0.609	5.983	83.5	2.1099	4.0	711.0	20.1	396.90
4930.17331	0.0	9.69	0.0	0.585	5.707	54.0	2.3817	6.0	391.0	19.2	396.90
4940.27957	0.0	9.69	0.0	0.585	5.926	42.6	2.3817	6.0	391.0	19.2	396.90
4950.17899	0.0	9.69	0.0	0.585	5.670	28.8	2.7986	6.0	391.0	19.2	393.29
4960.28960	0.0	9.69	0.0	0.585	5.390	72.9	2.7986	6.0	391.0	19.2	396.90
4970.26838	0.0	9.69	0.0	0.585	5.794	70.6	2.8927	6.0	391.0	19.2	396.90
4980.23912	0.0	9.69	0.0	0.585	6.019	65.3	2.4091	6.0	391.0	19.2	396.90
4990.17783	0.0	9.69	0.0	0.585	5.569	73.5	2.3999	6.0	391.0	19.2	395.77
5000.22438	0.0	9.69	0.0	0.585	6.027	79.7	2.4982	6.0	391.0	19.2	396.90
5010.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99
5020.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90
5030.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90
5040.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45
5050.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90

In [9]:
```
boston.target.shape
```

Out[9]: (506,)

In [13]:
```
data['Price'] = boston.target
data.head()
```

Out[13]:

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
00.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
10.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
20.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
30.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
40.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90

In [14]:
```
data.describe()
```

Out[14]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000

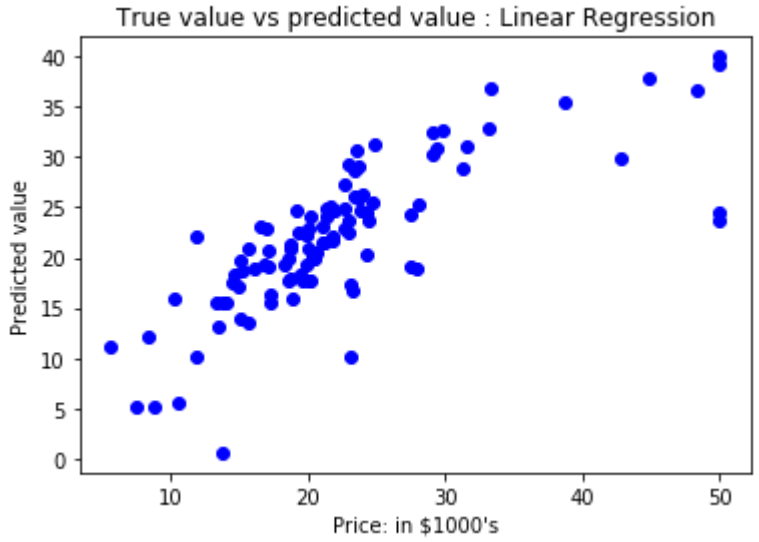
In [15]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 15 columns):
 #   Column   Non-Null Count  Dtype  
---  -------  --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  price    506 non-null    float64
 14  Price    506 non-null    float64
dtypes: float64(15)
memory usage: 59.4 KB
```

In [20]:
```
x=boston.data
y=boston.target
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2

print("xtrain shape : ", xtrain.shape)
print("xtest shape  : ", xtest.shape)
print("ytrain shape : ", ytrain.shape)
print("ytest shape  : ", ytest.shape)
```

```
xtrain shape :  (404, 13)
xtest shape  :  (102, 13)
ytrain shape :  (404,)
ytest shape  :  (102,)
```

In [26]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(xtrain, ytrain)

y_pred = regressor.predict(xtest)


plt.scatter(ytest, y_pred, c = 'blue')
plt.xlabel("Price: in $1000's")
plt.ylabel("Predicted value")
plt.title("True value vs predicted value : Linear Regression")
plt.show()
```



In [27]:
```
data['Sell'] = boston.target
data.head()
```

Out[27]:

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
00.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
10.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
20.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
30.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
40.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90

In [ ]:

In [29]:
```
del data['Sell']
```

In [31]:
```
data.head()
```

Out[31]:

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
00.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
10.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
20.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
30.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
40.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90

In [32]:
```
del data['price']
data.head()
```

Out[32]:

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
00.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
10.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
20.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
30.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
40.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90

In [ ]:

Practical No:5

In [1]:
```
import pandas as pd
df=pd.read_csv("/home/student/Downloads/archive/IRIS.csv")
df
```

Out[1]:

sepal_length	sepal_width	petal_length	petal_width	species
05.1	3.5	1.4	0.2	Iris-setosa
14.9	3.0	1.4	0.2	Iris-setosa
24.7	3.2	1.3	0.2	Iris-setosa
34.6	3.1	1.5	0.2	Iris-setosa
45.0	3.6	1.4	0.2	Iris-setosa
.....	...	...	...	...
1456.7	3.0	5.2	2.3	Iris-virginica
1466.3	2.5	5.0	1.9	Iris-virginica
1476.5	3.0	5.2	2.0	Iris-virginica
1486.2	3.4	5.4	2.3	Iris-virginica
1495.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [2]:
```
df.shape
```

Out[2]: (150, 5)

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
df = pd.read_csv('/home/student/Downloads/archive(1)/Social_Network_
df.head()
```

Out[3]:

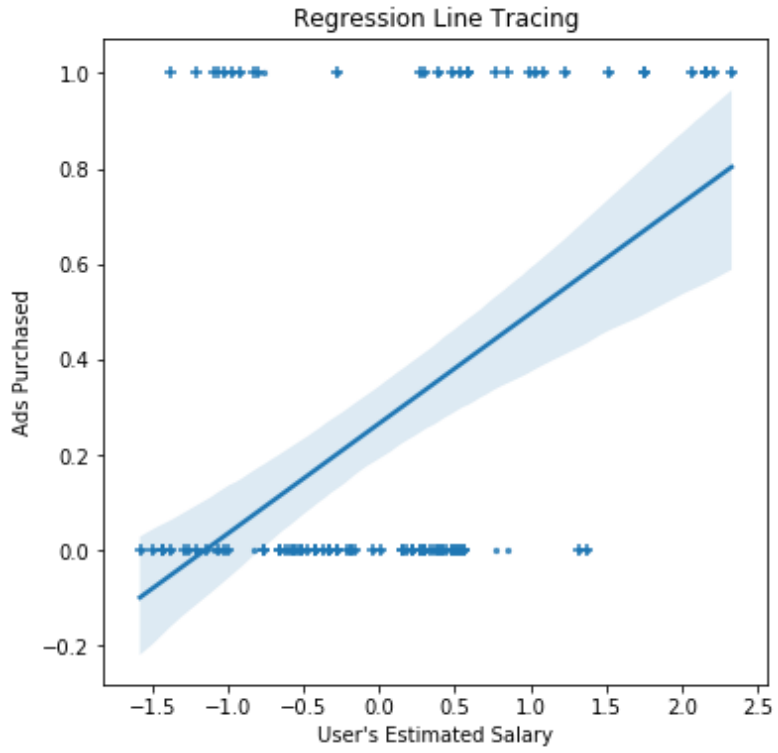
Age	EstimatedSalary	Purchased
019	19000	0
135	20000	0
226	43000	0
327	57000	0
419	76000	0

In [4]:
```
X = df[['Age', 'EstimatedSalary']]
Y = df['Purchased']
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}'
print(f'Test Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')
```

```
Train Dataset Size - X: (300, 2), Y: (300,)
Test Dataset Size - X: (100, 2), Y: (100,)
```

In [5]:
```
from sklearn.linear_model import LogisticRegression
lm = LogisticRegression(random_state = 0, solver='lbfgs' )
lm.fit(X_train, Y_train)
predictions = lm.predict(X_test)
plt.figure(figsize=(6, 6))
sns.regplot(x = X_test[:, 1], y = predictions, scatter_kws={'s':5})
plt.scatter(X_test[:, 1], Y_test, marker = '+')
plt.xlabel("User's Estimated Salary")
plt.ylabel('Ads Purchased')
plt.title('Regression Line Tracing')
```

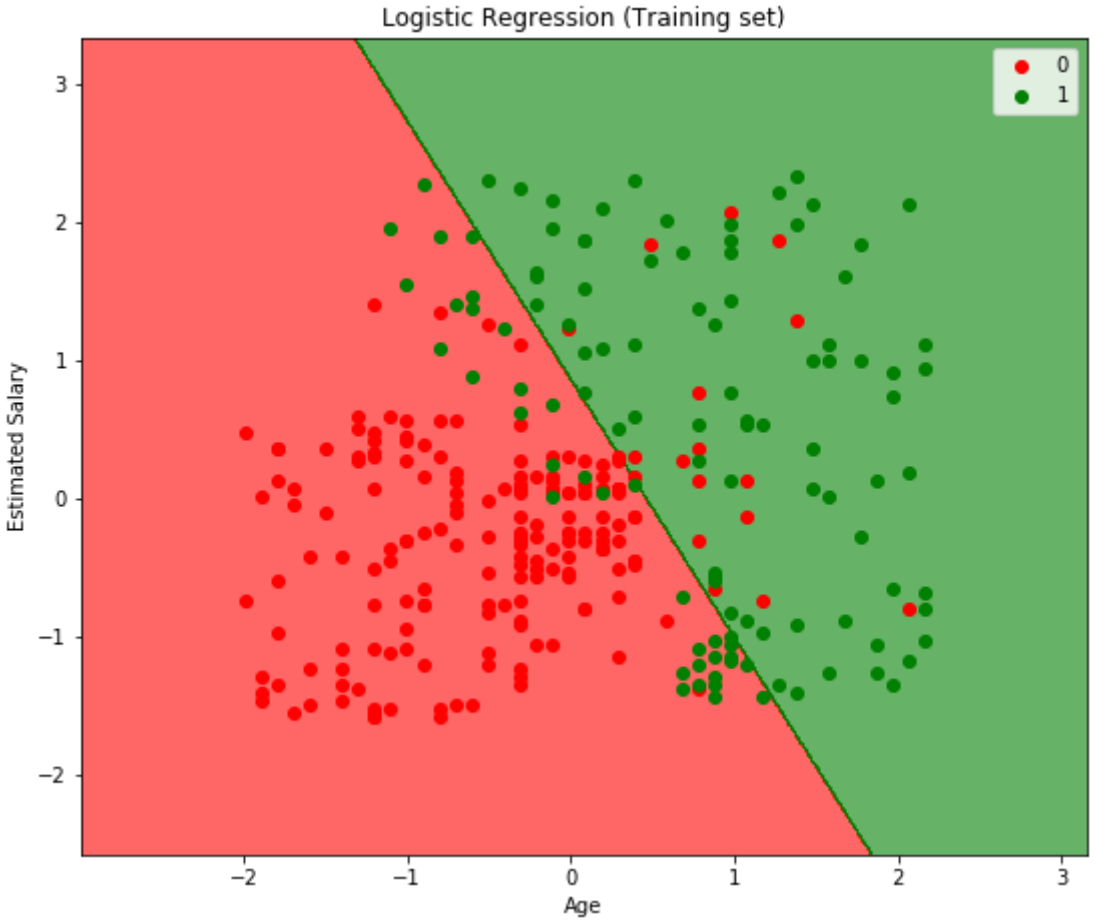
Out[5]: Text(0.5, 1.0, 'Regression Line Tracing')



In [6]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
cm = confusion_matrix(Y_test, predictions)
print(f'''Confusion matrix :\n
| Positive Prediction\t| Negative Prediction
----------------+-------------------------+-----------------------
Positive Class | True Positive (TP) {cm[0, 0]}\t| False Negative (FN
----------------+-------------------------+-----------------------
Negative Class | False Positive (FP) {cm[1, 0]}\t| True Negative (TN
cr = classification_report(Y_test, predictions)
print('Classification report : \n', cr)
```

```
Confusion matrix :

| Positive Prediction    | Negative Prediction
----------------+-------------------------+-----------------------
Positive Class | True Positive (TP) 65   | False Negative (FN) 3
----------------+-------------------------+-----------------------
Negative Class | False Positive (FP) 8   | True Negative (TN) 24

Classification report :
               precision    recall  f1-score   support

           0       0.89      0.96      0.92        68
           1       0.89      0.75      0.81        32

    accuracy                           0.89       100
   macro avg       0.89      0.85      0.87       100
weighted avg       0.89      0.89      0.89       100
```

In [7]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, Y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() +
plt.figure(figsize=(9, 7.5))
plt.contourf(X1, X2, lm.predict(np.array([X1.ravel(), X2.ravel()]).T
alpha = 0.6, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j inenumerate (np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
color = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



In [ ]:

Practical No:6

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
df = pd.read_csv('Desktop/Iris.csv')
df.head()
```

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [3]:

```
df.describe()
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

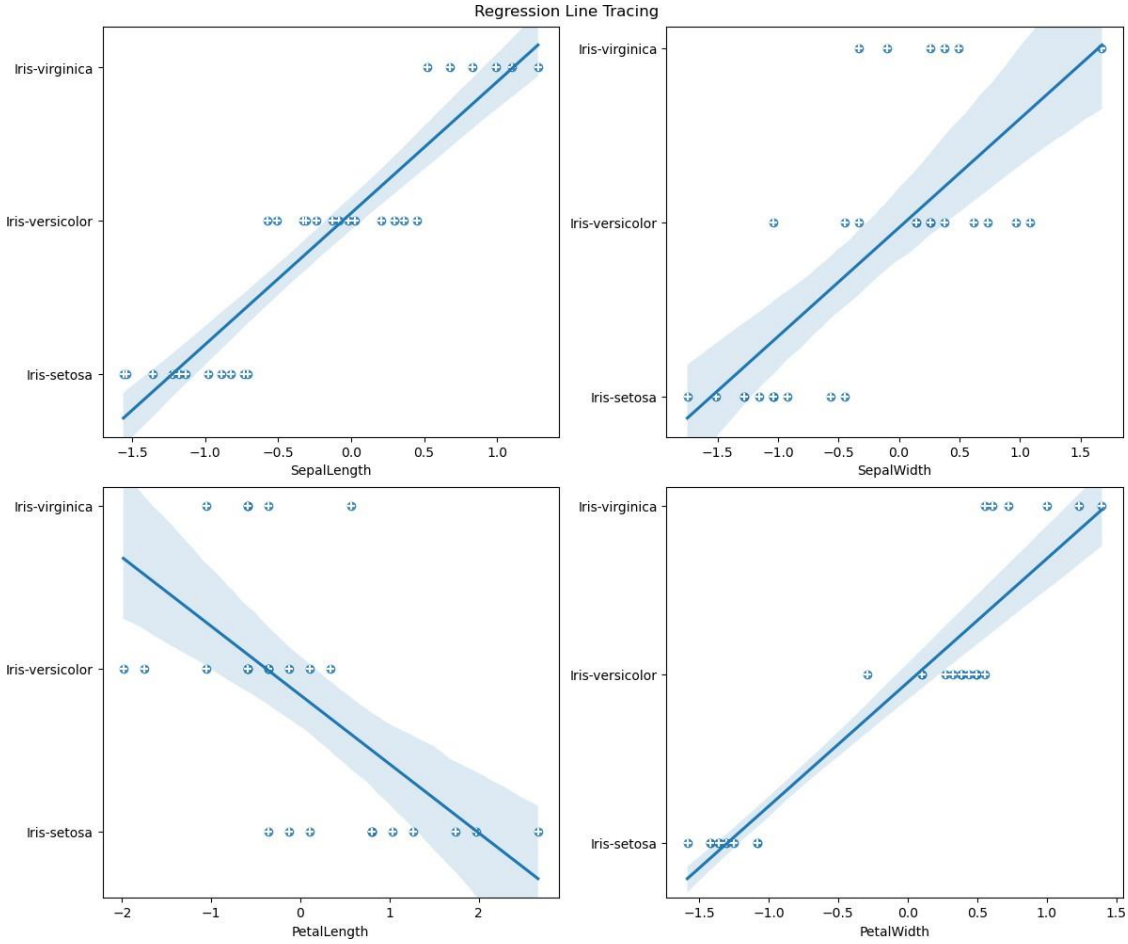
In [2]:

```
X = df.iloc[:, :4].values
Y = df['Species'].values
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
print(f'Test Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')
```

```
Train Dataset Size - X: (120, 4), Y: (120,)
Test Dataset Size - X: (30, 4), Y: (30,)
```

In [3]:

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, Y_train)
predictions = classifier.predict(X_test)
mapper = {'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2}
predictions_ = [mapper[i] for i in predictions]
fig, axs = plt.subplots(2, 2, figsize = (12, 10), constrained_layout = True)
fig.suptitle('Regression Line Tracing')
for i in range(4):
 x, y = i // 2, i % 2
 sns.regplot(x = X_test[:, i], y = predictions_, ax=axs[x, y])
 axs[x, y].scatter(X_test[:, i][::-1], Y_test[::-1], marker = '+', color="white")
 axs[x, y].set_xlabel(df.columns[i + 1][:-2])
```



In [4]:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
cm = confusion_matrix(Y_test, predictions)
print(f'''Confusion matrix :\n
| Positive Prediction\t| Negative Prediction
---------------+------------------------+----------------------
Positive Class | True Positive (TP) {cm[0, 0]}\t| False Negative (FN) {cm[0, 1]}
---------------+------------------------+----------------------
Negative Class | False Positive (FP) {cm[1, 0]}\t| True Negative (TN) {cm[1, 1]}\n''')
cm = classification_report(Y_test, predictions)
print('Classification report : \n', cm)
```

```
Confusion matrix :

| Positive Prediction   | Negative Prediction
---------------+------------------------+----------------------
Positive Class | True Positive (TP) 11  | False Negative (FN) 0
---------------+------------------------+----------------------
Negative Class | False Positive (FP) 0  | True Negative (TN) 13

Classification report :
                  precision    recall  f1-score   support

     Iris-setosa       1.00      1.00      1.00        11
 Iris-versicolor       1.00      1.00      1.00        13
  Iris-virginica       1.00      1.00      1.00         6

        accuracy                           1.00        30
       macro avg       1.00      1.00      1.00        30
    weighted avg       1.00      1.00      1.00        30
```

In [ ]:

Practical No:7

In [16]:
```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to /home/student/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/student/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /home/student/nltk_dat
a..
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/student/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[16]: True

In [15]:
```
text= "Tokenization is the first step in text analytics.The process
from nltk.tokenize import sent_tokenize
tokenized_text= sent_tokenize(text)
print(tokenized_text)

print ('-'*80)

from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)

print ('-'*80)

from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)

print ('-'*80)

word_tokens= word_tokenize(text.lower())
filtered_sentence = []
for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)
print("Tokenized Sentence:",word_tokens)
print("Filterd Sentence:",filtered_sentence)

print ('-'*80)

from nltk.stem import PorterStemmer
e_words= ["wait", "waiting", "waited", "waits"]
ps =PorterStemmer()
for w in e_words:
    rootWord=ps.stem(w)
    print(rootWord)

print ('-'*80)

from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)
for w in tokenization:
    print("Lemma for {} is {}".format(w, wordnet_lemmatizer.lemmatiz

print ('-'*80)

from nltk.tokenize import word_tokenize
data="The pink sweater fit her perfectly"
words=word_tokenize(data)
for word in words:
    print(nltk.pos_tag([word]))
```

```
['Tokenization is the first step in text analytics.The process of b
reaking down a text paragraph into smaller chunks such as words or
sentences is called Tokenization.']
--------------------------------------------------------------------
-------------
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analy
```

```
tics.The', 'process', 'of', 'breaking', 'down', 'a', 'text', 'parag
raph', 'into', 'smaller', 'chunks', 'such', 'as', 'words', 'or', 's
entences', 'is', 'called', 'Tokenization', '.']
------------------------------------------------------------------------
-------------
{'then', 'yourselves', 'him', 'that', 'until', 'as', 'here', 'not',
'where', 'my', 'both', 'about', 'so', 'each', 'aren', 'am', 'she',
'does', 'have', 'should', 'your', "you've", 'during', 'out', 'do',
'just', 'through', "isn't", 'these', 'won', 'its', 'myself', 'under
', "needn't", 'weren', 'a', 're', 'same', "hadn't", "you'll", 'how
', 't', 'on', 'some', 'can', 'ma', 'them', 'shouldn', 'further', 't
hemselves', "should've", 'such', "it's", 'which', 'now', "shouldn'
t", 'between', 'too', 'other', 'll', 'than', "didn't", 'there', 'no
', "you'd", 'by', 'those', 'above', 'all', "hasn't", "won't", 'your
self', "doesn't", 'doesn', "you're", 'don', "she's", 'yours', 'own
', 'an', 'most', 'at', 'with', 'are', 've', 'was', 'this', "weren'
t", 'needn', 'ourselves', 'ain', 'if', 'only', "couldn't", 'they',
'his', 'again', 'before', 'into', 'having', 's', 'had', 'her', 'it
', 'what', 'below', 'isn', 'wasn', 'we', 'who', 'the', 'mustn', 'di
d', 'itself', 'to', 'haven', 'while', 'been', 'o', 'and', 'nor', 't
heir', "mustn't", 'more', "wouldn't", 'shan', "mightn't", 'couldn',
'once', 'y', 'hasn', 'has', 'he', 'didn', "wasn't", 'be', 'against
', 'is', 'because', 'doing', 'ours', 'but', 'hers', "don't", 'will
', 'hadn', 'you', 'for', 'of', 'when', 'any', 'why', 'himself', 'me
', "aren't", "haven't", 'herself', 'from', 'over', 'our', 'off', 'm
', 'wouldn', "that'll", 'in', 'being', 'after', 'were', 'or', 'migh
tn', 'down', "shan't", 'up', 'very', 'theirs', 'i', 'd', 'few', 'wh
om'}
------------------------------------------------------------------------
-------------
Tokenized Sentence: ['tokenization', 'is', 'the', 'first', 'step',
'in', 'text', 'analytics.the', 'process', 'of', 'breaking', 'down',
'a', 'text', 'paragraph', 'into', 'smaller', 'chunks', 'such', 'as
', 'words', 'or', 'sentences', 'is', 'called', 'tokenization', '.']
Filterd Sentence: ['tokenization', 'first', 'step', 'text', 'analyt
ics.the', 'process', 'breaking', 'text', 'paragraph', 'smaller', 'c
hunks', 'words', 'sentences', 'called', 'tokenization', '.']
------------------------------------------------------------------------
-------------
wait
wait
wait
wait
------------------------------------------------------------------------
-------------
Lemma for studies is study
Lemma for studying is studying
Lemma for cries is cry
Lemma for cry is cry
------------------------------------------------------------------------
-------------
[('The', 'DT')]
[('pink', 'NN')]
[('sweater', 'NN')]
[('fit', 'NN')]
[('her', 'PRP$')]
```

In [ ]:

In [1]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [2]:
```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
bagOfWordsA = documentA.split(' ')
bagOfWordsA
```

Out[2]:
```
['Jupiter', 'is', 'the', 'largest', 'Planet']
```

In [3]:
```
bagOfWordsB = documentB.split(' ')
bagOfWordsB
```

Out[3]:
```
['Mars', 'is', 'the', 'fourth', 'planet', 'from', 'the', 'Sun']
```

In [4]:
```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
uniqueWords
```

Out[4]:
```
{'Jupiter',
 'Mars',
 'Planet',
 'Sun',
 'fourth',
 'from',
 'is',
 'largest',
 'planet',
 'the'}
```

In [5]:
```
numOfWordsA = dict.fromkeys(uniqueWords, 0)
```

In [6]:
```
numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsA:
    numOfWordsA[word] += 1
    numOfWordsB = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsB:
    numOfWordsB[word] += 1
```

In [7]:
```
def computeTF(wordDict, bagOfWords):
    tfDict = {}
    bagOfWordsCount = len(bagOfWords)
    for word, count in wordDict.items():
        tfDict[word] = count / float(bagOfWordsCount)
    return tfDict
tfA = computeTF(numOfWordsA, bagOfWordsA)
tfB = computeTF(numOfWordsB, bagOfWordsB)
```

In [10]:
```
def computeIDF(documents):
    import math
    N = len(documents)
    idfDict = dict.fromkeys(documents[0].keys(), 0)
    for document in documents:
        for word, val in document.items():
            if val > 0:
                idfDict[word] += 1
    for word, val in idfDict.items():
        idfDict[word] = math.log(N / float(val))
    return idfDict
idfs = computeIDF([numOfWordsA, numOfWordsB])
idfs
```

Out[10]:
```
{'from': 0.6931471805599453,
 'Mars': 0.6931471805599453,
 'is': 0.0,
 'Sun': 0.6931471805599453,
 'Planet': 0.6931471805599453,
 'the': 0.0,
 'fourth': 0.6931471805599453,
 'largest': 0.6931471805599453,
 'planet': 0.6931471805599453,
 'Jupiter': 0.6931471805599453}
```

In [11]:
```
def computeTFIDF(tfBagOfWords, idfs):
    tfidf = {}
    for word, val in tfBagOfWords.items():
        tfidf[word] = val * idfs[word]
    return tfidf
tfidfA = computeTFIDF(tfA, idfs)
tfidfB = computeTFIDF(tfB, idfs)
df = pd.DataFrame([tfidfA, tfidfB])
df
```

Out[11]:

	from	Mars	is	Sun	Planet	the	fourth	largest	planet	Jupiter
0	0.000000	0.000000	0.0	0.000000	0.138629	0.0	0.000000	0.138629	0.000000	0.138629
1	0.086643	0.086643	0.0	0.086643	0.000000	0.0	0.086643	0.000000	0.086643	0.000000

In [ ]:

Practical No:8

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
data = pd.read_csv('https://raw.githubusercontent.com/dphi-official/Datasets/master/tita
data.head()
```

Out[1]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

In [2]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [3]:

```
data.describe()
```

Out[3]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [4]:

```
data.isnull().sum()
```

Out[4]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [5]:

```
data['Age'] = data['Age'].fillna(np.mean(data['Age']))
data['Cabin'] = data['Cabin'].fillna(data['Cabin'].mode()[0])
data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])
data.isnull().sum()
```

Out[5]:

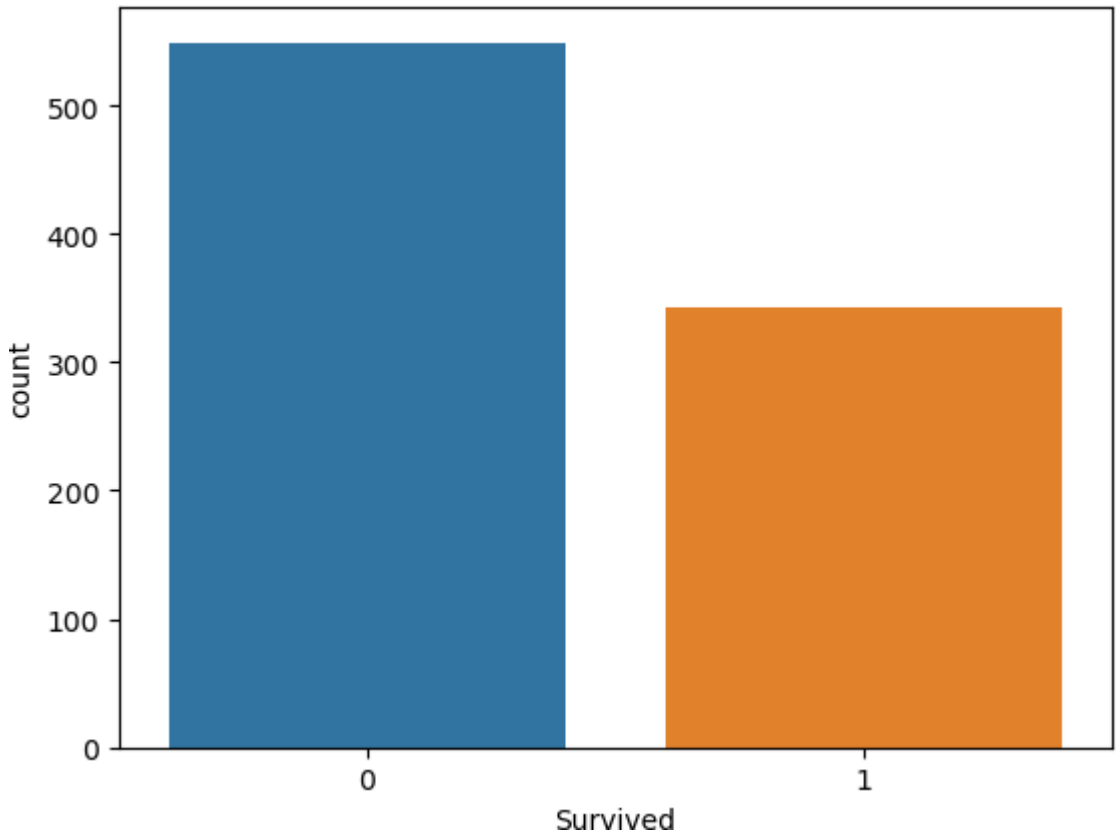
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

In [6]:

```
sns.countplot(x='Survived',data=data)
```

Out[6]:

```
<Axes: xlabel='Survived', ylabel='count'>
```

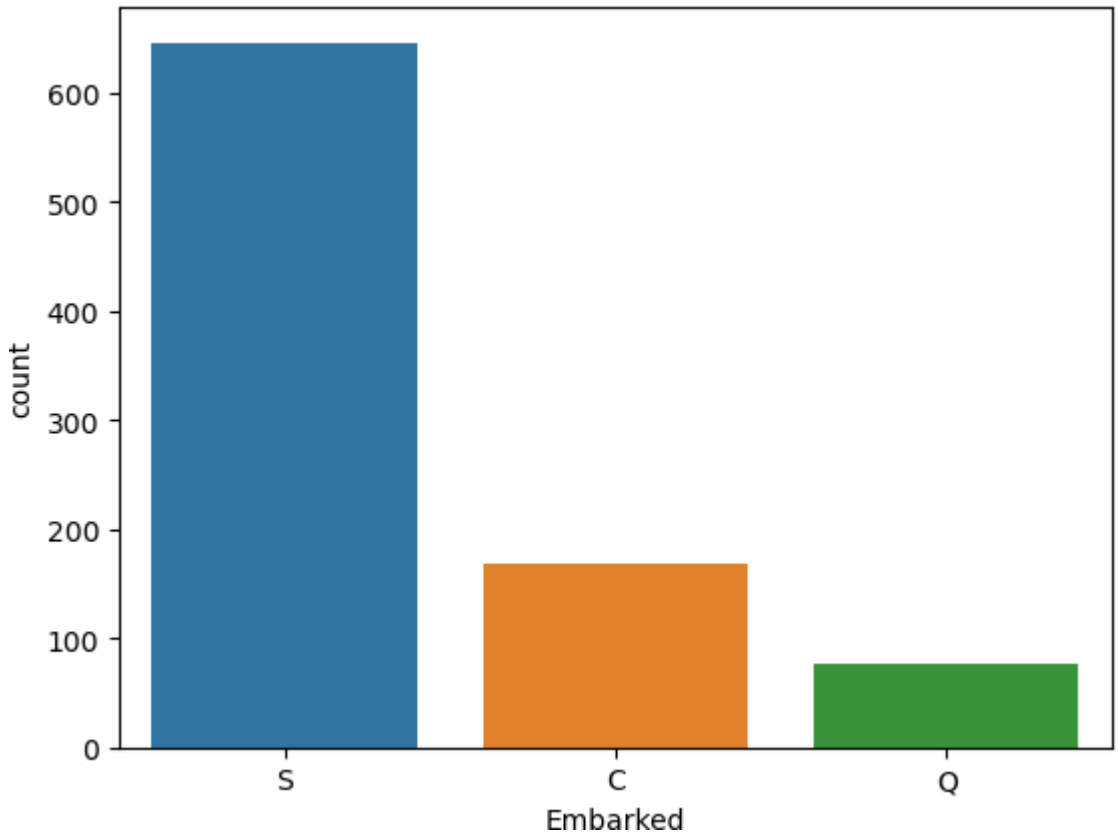


In [5]:

```
sns.countplot(x='Embarked',data=data)
```

Out[7]:

```
<Axes: xlabel='Embarked', ylabel='count'>
```

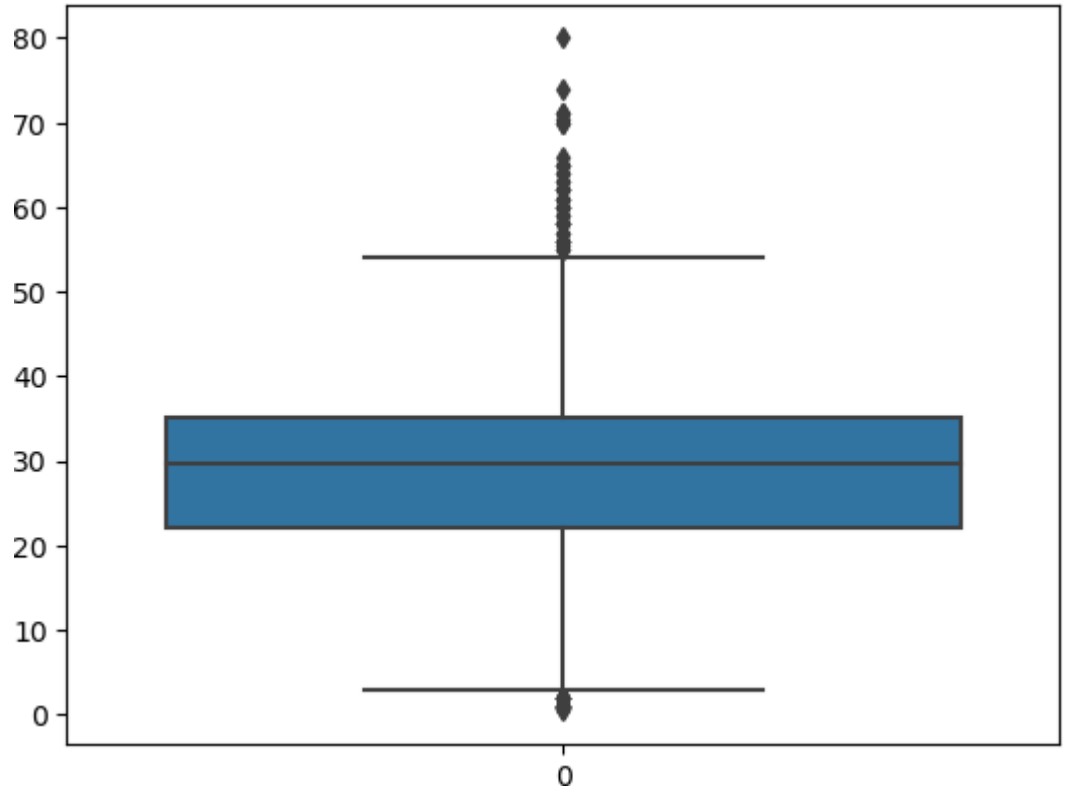


In [6]:

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

Out[8]:

<Axes: >

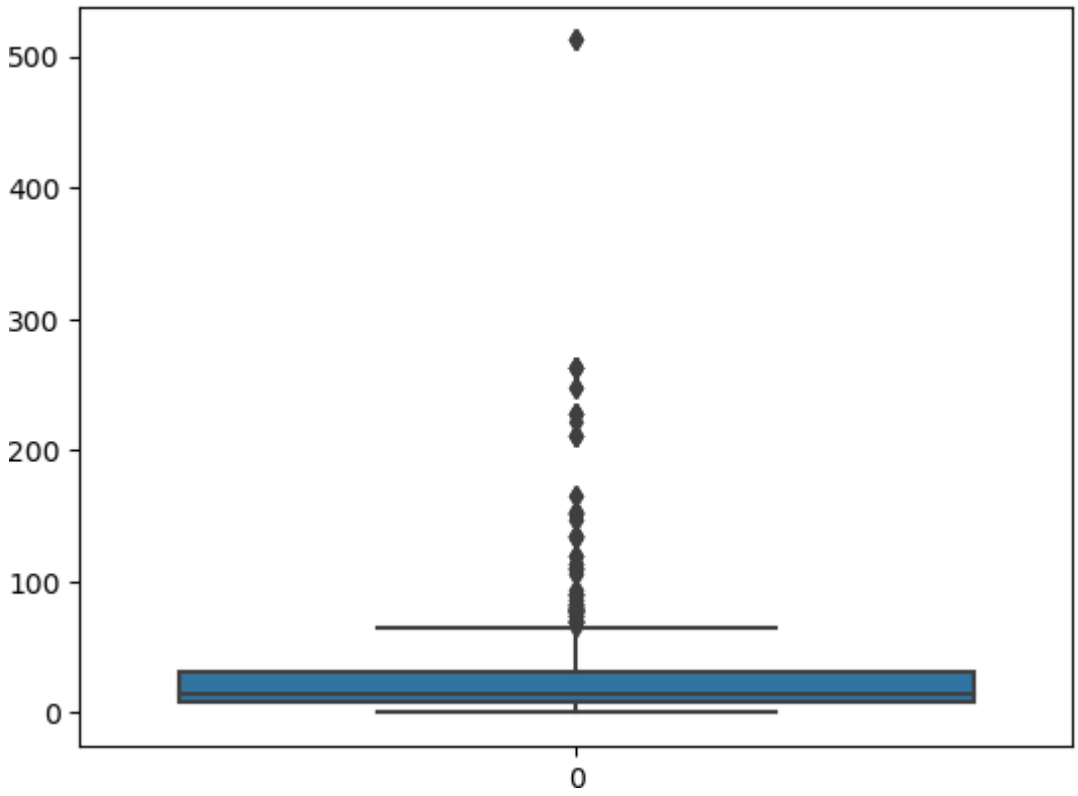


In [7]:

```
sns.boxplot(data['Fare'])
```

Out[9]:

<Axes: >

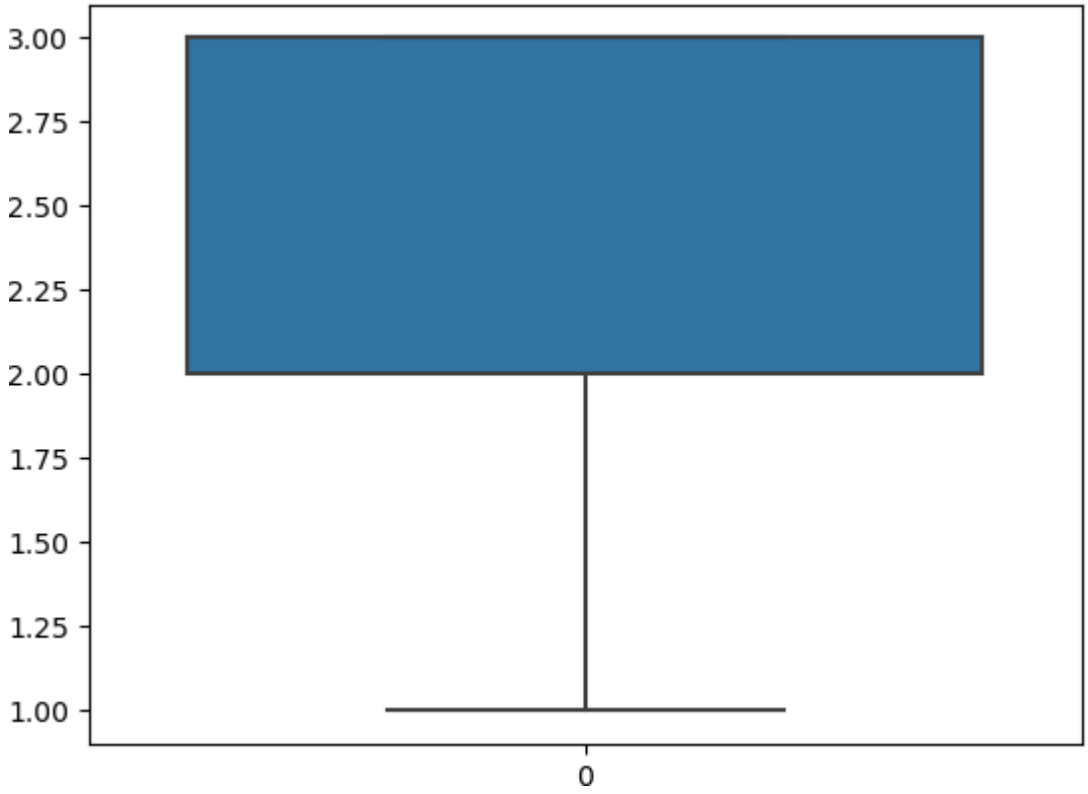


In [18]:

```
sns.boxplot(data['Pclass'])
```

Out[11]:

<Axes: >

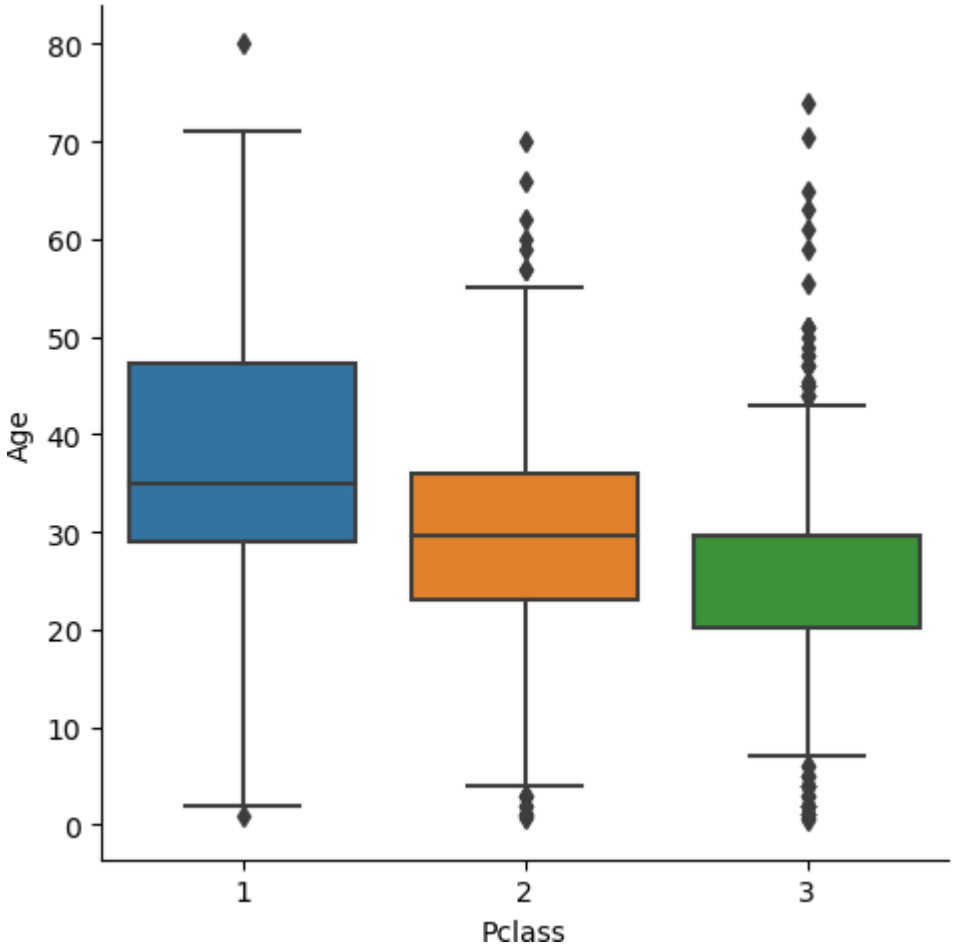


In [19]:

```
sns.catplot(x= 'Pclass', y = 'Age', data=data, kind = 'box')
```

Out[12]:

```
<seaborn.axisgrid.FacetGrid at 0x215991962f0>
```

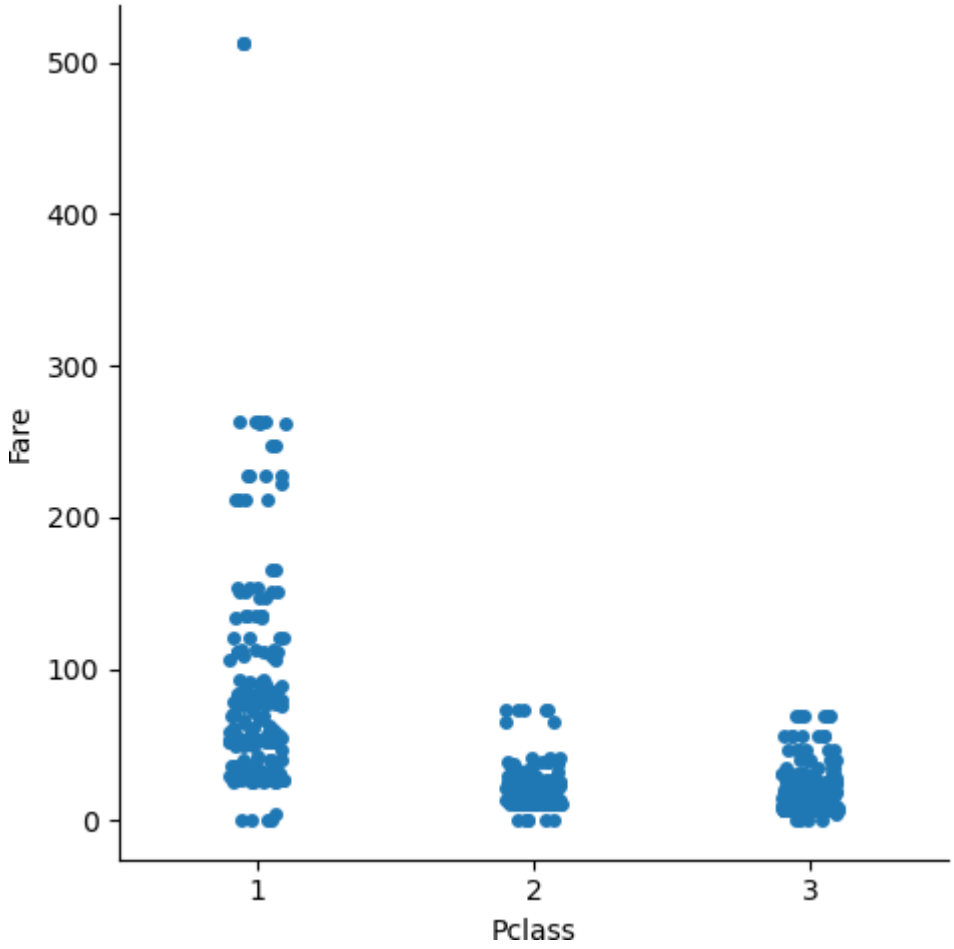


In [110

```
sns.catplot(x= 'Pclass', y = 'Fare', data=data, kind = 'strip')
```

Out[13]:

```
<seaborn.axisgrid.FacetGrid at 0x215992af550>
```

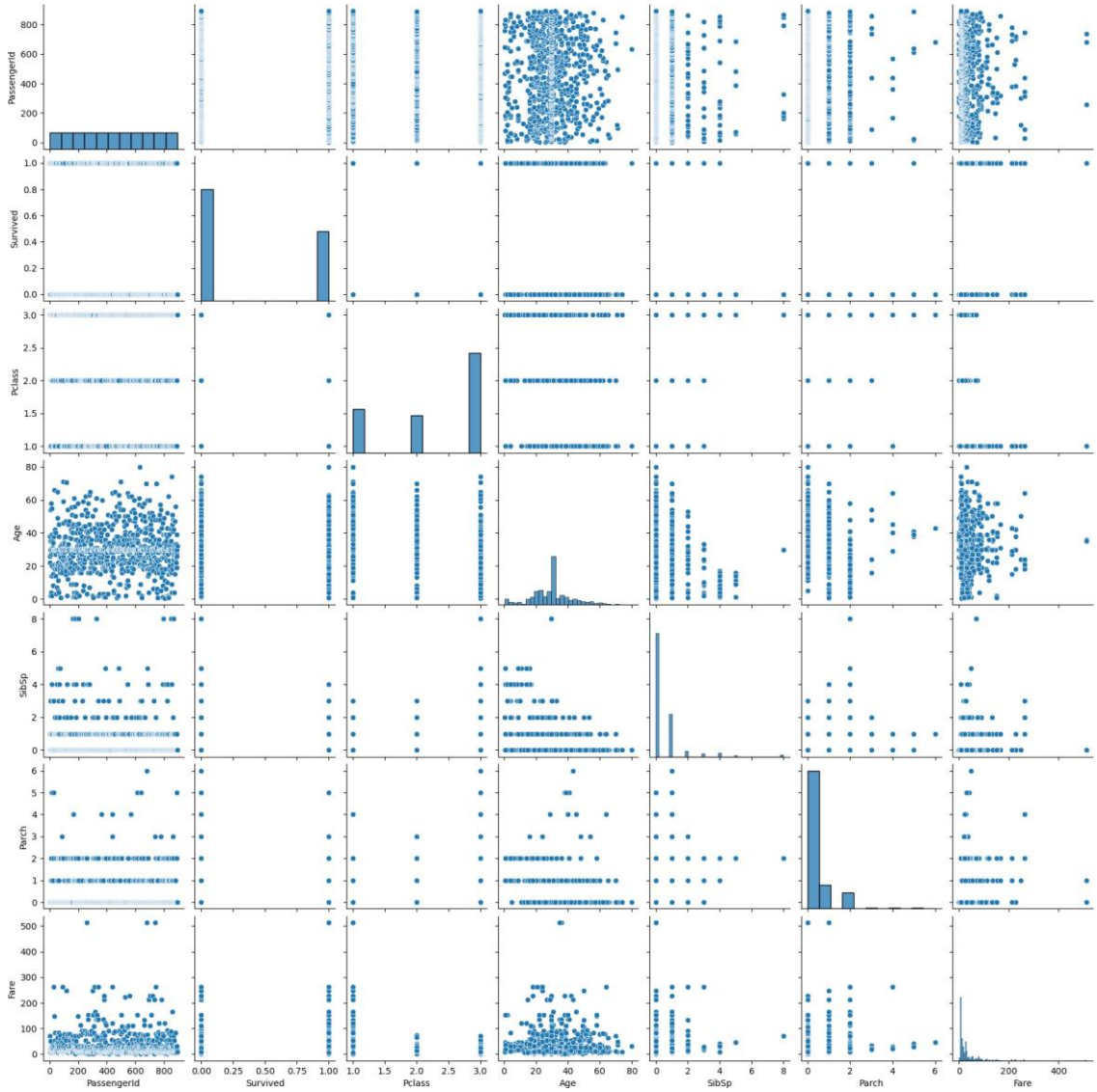


In [111

```
sns.pairplot(data)
```

Out[14]:

```
<seaborn.axisgrid.PairGrid at 0x215992ffaf0>
```



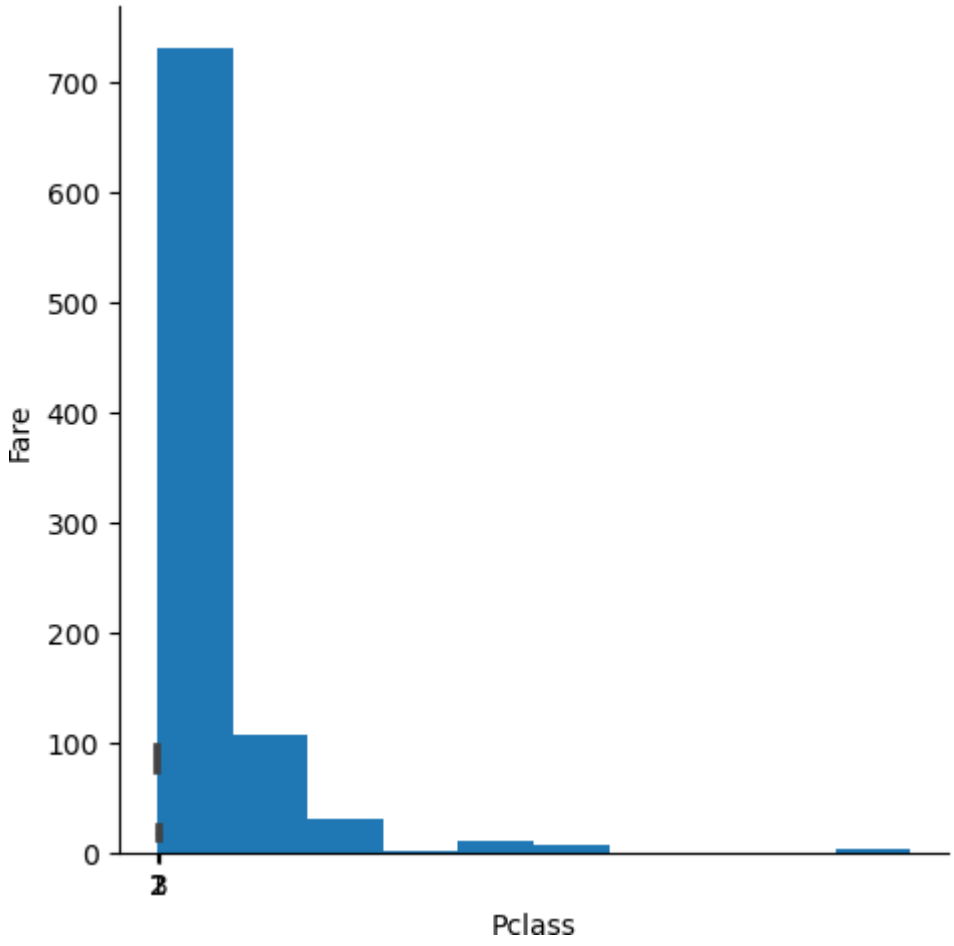
In [112

```
print("Price of Ticket for each passenger is distributed")
sns.catplot(x='Pclass', y='Fare', data=data, kind='bar')
import matplotlib.pyplot as plt
plt.hist(data['Fare'])
```

```
Price of Ticket for each passenger is distributed
```

Out[16]:

```
(array([732., 106.,  31.,   2.,  11.,   6.,   0.,   0.,   0.,   3.]),
 array([  0.     ,  51.23292, 102.46584, 153.69876, 204.93168, 256.1646 ,
        307.39752, 358.63044, 409.86336, 461.09628, 512.3292 ]),
 <BarContainer object of 10 artists>)
```



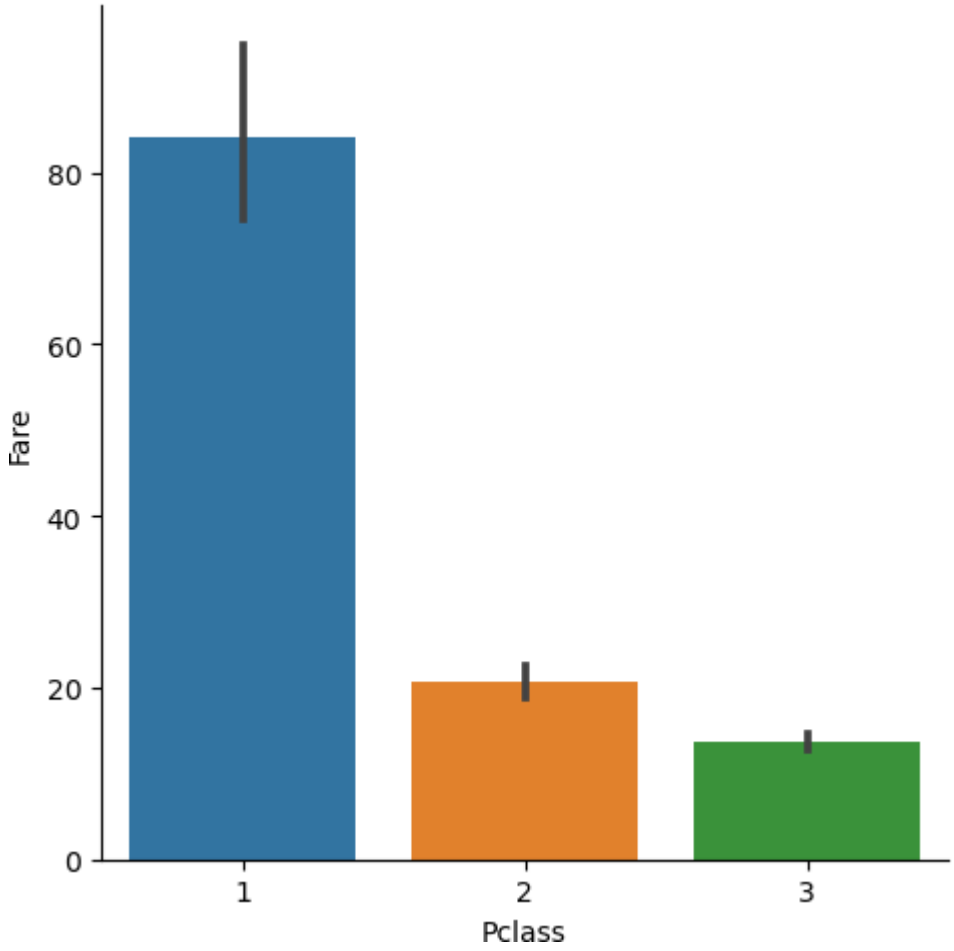
In [113

```
print("Price of Ticket for each passenger is distributed")
sns.catplot(x='Pclass', y='Fare', data=data, kind='bar')
```

```
Price of Ticket for each passenger is distributed
```

Out[17]:

```
<seaborn.axisgrid.FacetGrid at 0x2159d578d60>
```



In [ ]:

Practical No:9

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
data = pd.read_csv('https://raw.githubusercontent.com/dphi-official/Datasets/master/tita
data.head()
```

Out[1]:

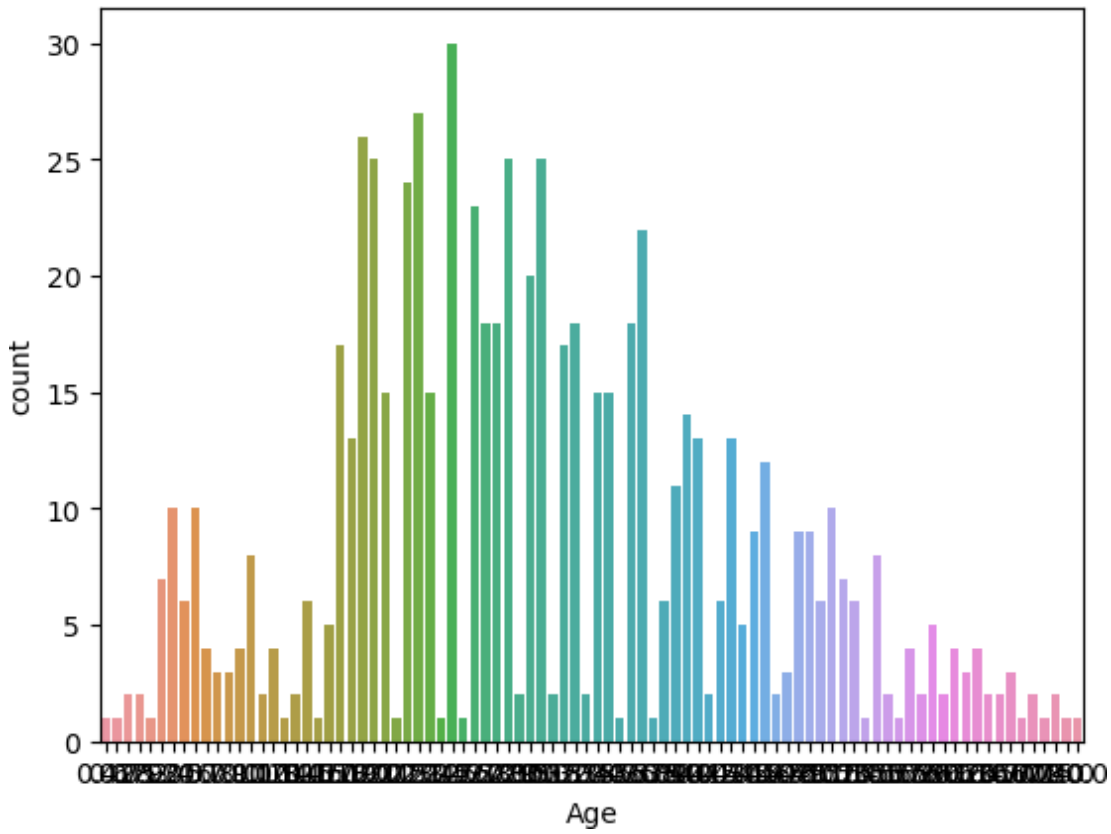
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

In [3]:

```
sns.countplot(x='Age',data=data)
```

Out[3]:

```
<Axes: xlabel='Age', ylabel='count'>
```

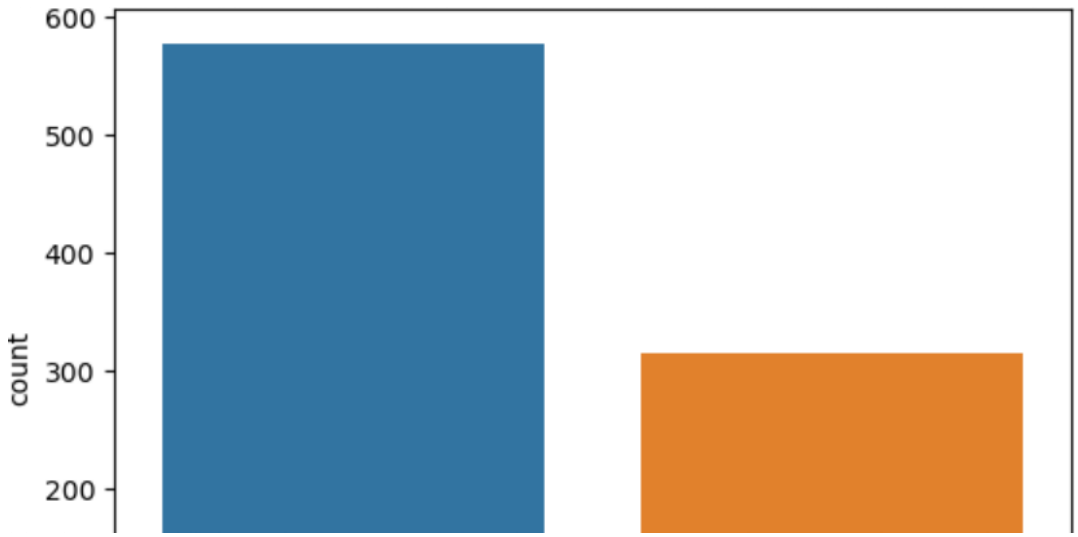


In [4]:

```
sns.countplot(x='Sex',data=data)
```

Out[4]:

```
<Axes: xlabel='Sex', ylabel='count'>
```

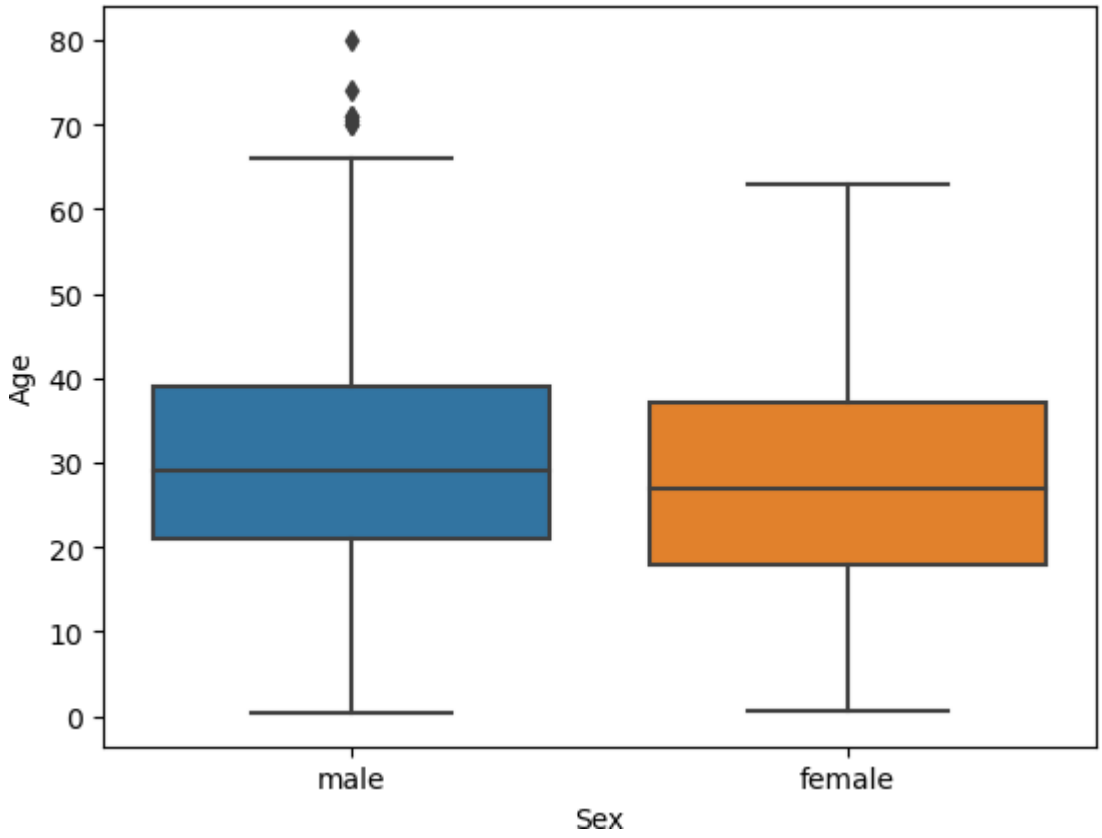


In [6]:

```
sns.boxplot(x='Sex',y='Age',data=data)
```

Out[6]:

```
<Axes: xlabel='Sex', ylabel='Age'>
```

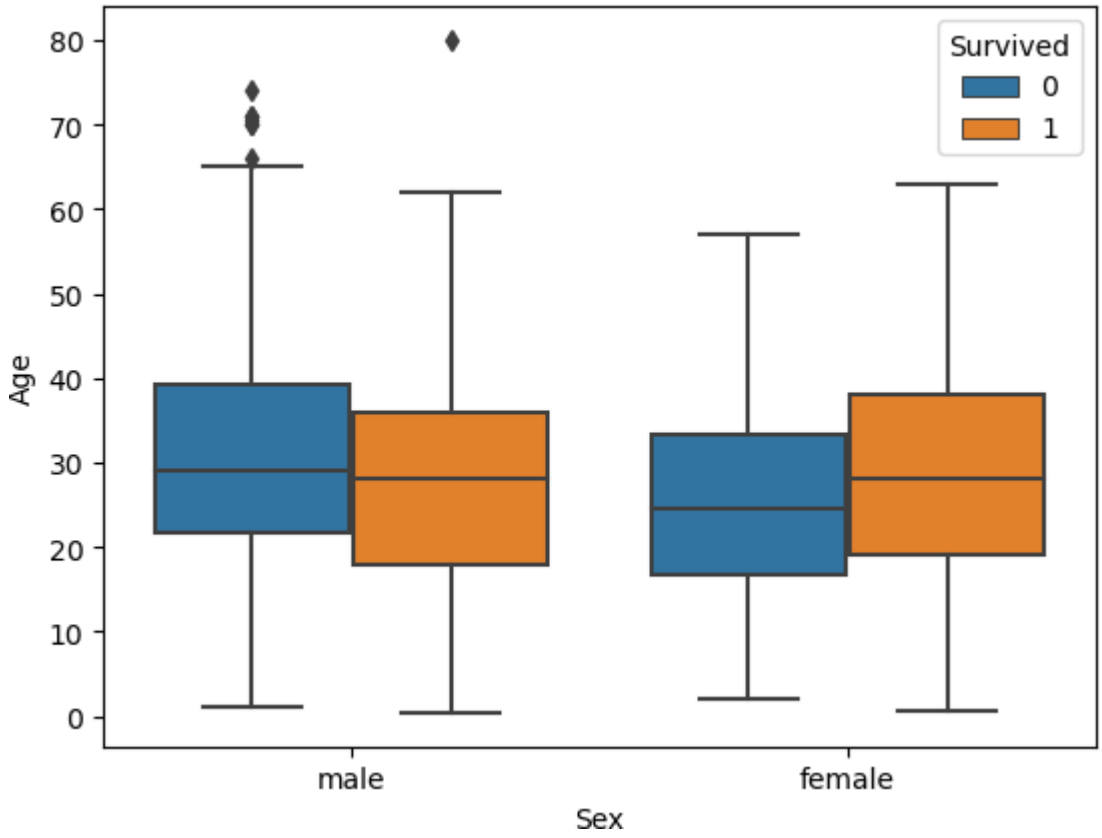


In [11]:

```
sns.boxplot(x='Sex',y='Age',data=data,hue='Survived')
```

Out[11]:

```
<Axes: xlabel='Sex', ylabel='Age'>
```



In [ ]:

```

```

Practical No:10

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('Desktop/Iris.csv')
df.head()
```

Out[1]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [2]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [3]:

```
df.describe()
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [7]:

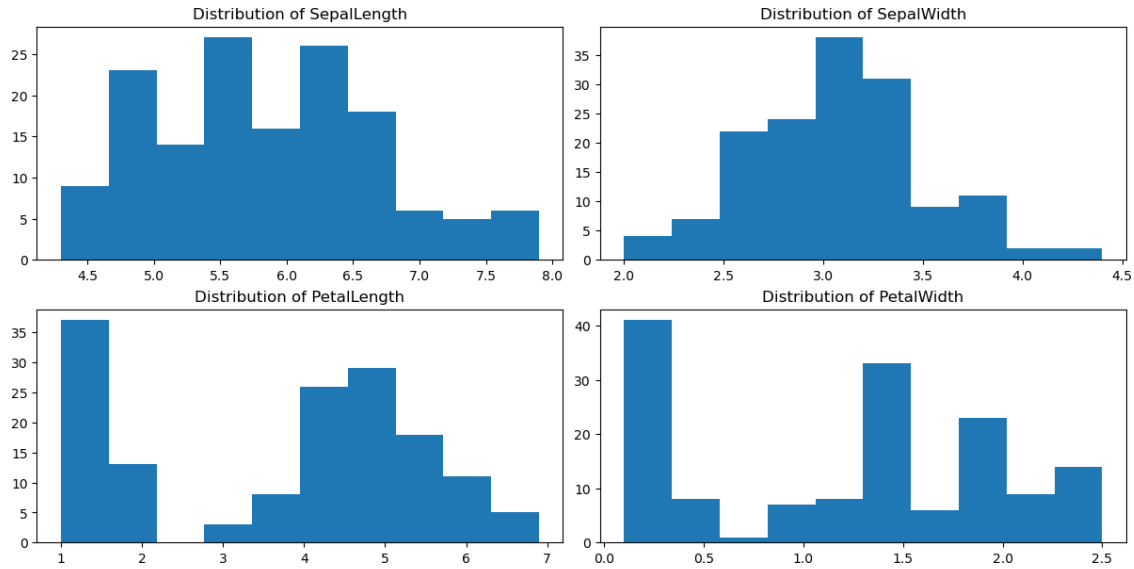
```
df.isnull().sum()
```

Out[7]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

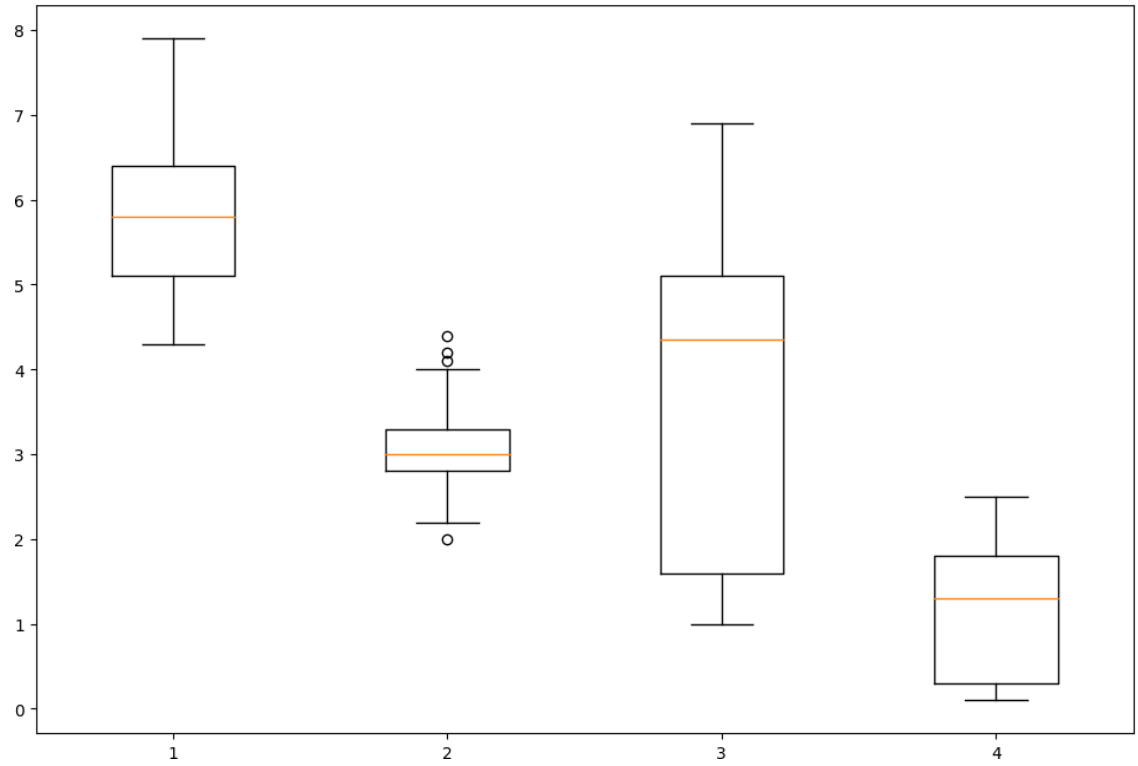
In [8]:

```
fig, axes = plt.subplots(2, 2, figsize=(12, 6), constrained_layout = True)
for i in range(4):
 x, y = i // 2, i % 2
 axes[x, y].hist(df[df.columns[i + 1]])
 axes[x, y].set_title(f"Distribution of {df.columns[i + 1][:-2]}")
```



In [9]:

```
data_to_plot = [df[x] for x in df.columns[1:-1]]
fig, axes = plt.subplots(1, figsize=(12,8))
bp = axes.boxplot(data_to_plot)
```

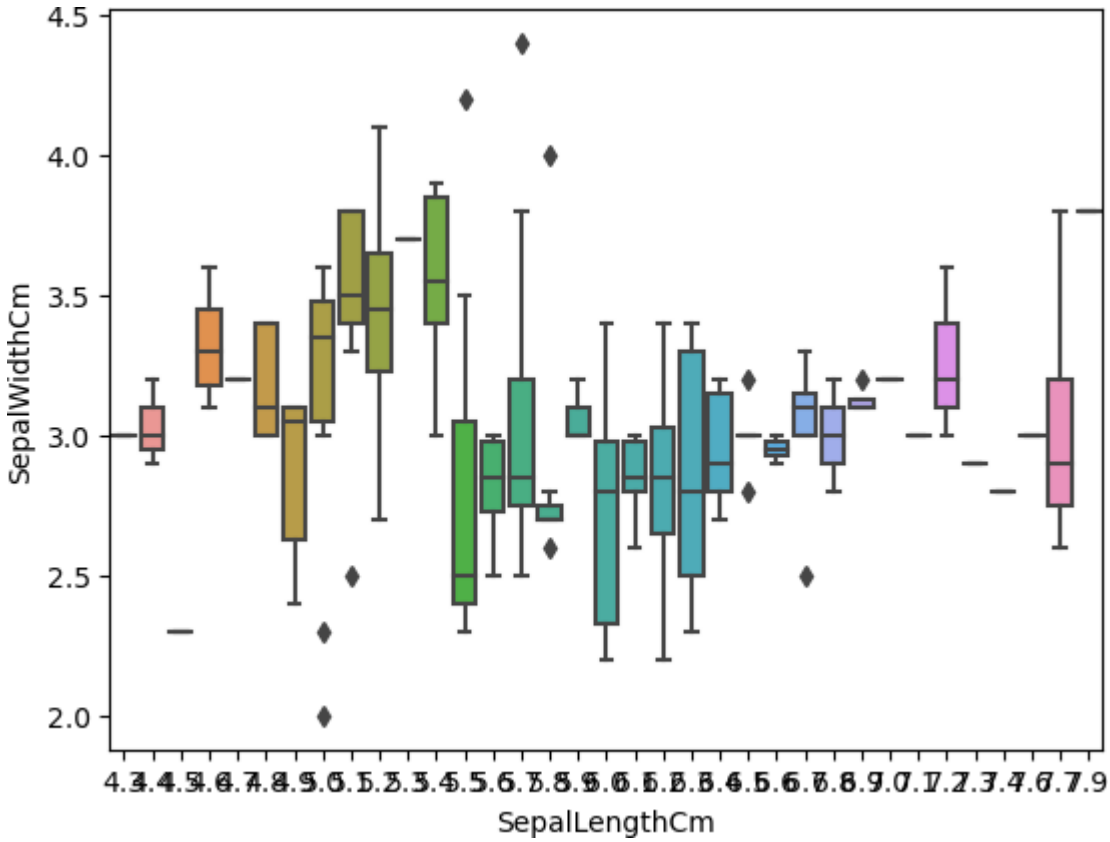


In [11]:

```
print("identify the outliers")
sns.boxplot(x='SepalLengthCm',y='SepalWidthCm',data=df)
```

Out[11]:

```
<Axes: xlabel='SepalLengthCm', ylabel='SepalWidthCm'>
```



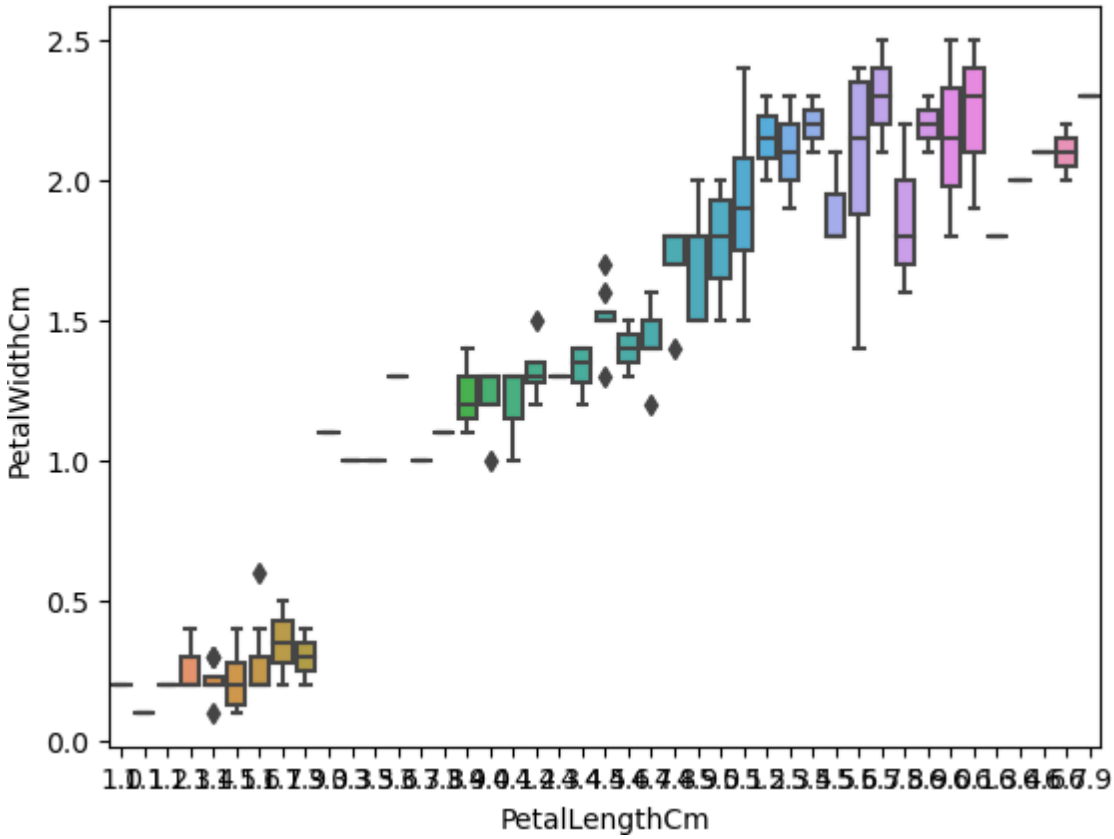
In [13]:

```
print("identify the outliers")
sns.boxplot(x='PetalLengthCm',y='PetalWidthCm',data=df)
```

```
identify the outliers
```

Out[13]:

```
<Axes: xlabel='PetalLengthCm', ylabel='PetalWidthCm'>
```



In [ ]:

Practical Number: 11

Title: Write a code in JAVA for a simple Wordcount application that counts the number of occurrences of each word in a given input set using the Hadoop MapReduce framework on local-standalone set-up.

Java Code for word count:

```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.*;
import org.apache.hadoop.mapreduce.lib.output.*;
import org.apache.hadoop.util.*;
public class WordCount extends Configured implements Tool
{
    public static void main(String args[]) throws Exception
    {
        int res = ToolRunner.run(new WordCount(), args);
        System.exit(res);
    }
    public int run(String[] args) throws Exception
    {
        Path inputPath = new Path(args[0]);
        Path outputPath = new Path(args[1]);
        Configuration conf = getConf();
        Job job = new Job(conf, this.getClass().toString());
        job.setJarByClass(WordCount.class);
        FileInputFormat.setInputPaths(job, inputPath);
        FileOutputFormat.setOutputPath(job, outputPath);
        job.setJobName("WordCount");

        job.setMapperClass(Map.class);
        job.setCombinerClass(Reduce.class);
        job.setReducerClass(Reduce.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        return job.waitForCompletion(true) ? 0 : 1;
    }
```

```
40      public static class Map extends Mapper<LongWritable, Text, Text, IntWritable>
41      {
42           private final static IntWritable one = new IntWritable(1);
43           private Text word = new Text();
44           public void map(LongWritable key, Text value, Mapper.Context
45          context) throws IOException, InterruptedException
46          {
47               String line = value.toString();
48               StringTokenizer tokenizer = new StringTokenizer(line);
49               while (tokenizer.hasMoreTokens())
50               {
51               word.set(tokenizer.nextToken());
52               context.write(word, one);
53               }
54          }
55      }
56      public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable>
57      {
58          public void reduce(Text key, Iterable<IntWritable> values, Context
59          context) throws IOException, InterruptedException
60          {
61               int sum = 0;
62               for(IntWritable value : values)
63               {
64               sum += value.get();
65               }
66               context.write(key, new IntWritable(sum));
67           }
68      }
69  }
70
```

Input File:

Pune

Mumbai

Nashik

Pune

Nashik

Kolapur

Practical Number: 12

Title: Design a distributed application using MapReduce which processes a log file of a system.

Java Code to process logfile

Mapper Class:

```
package SalesCountry;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class SalesMapper extends MapReduceBase implements Mapper<LongWritable,
Text, Text, IntWritable>
{
    private final static IntWritable one = new IntWritable(1);
    public void map(LongWritable key, Text value, OutputCollector<Text,
    IntWritable> output, Reporter reporter) throws IOException {
        String valueString = value.toString();
        String[] SingleCountryData = valueString.split("-");
        output.collect(new Text(SingleCountryData[0]), one);
    }
}
```

Reducer Class:

```
package SalesCountry;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class SalesCountryReducer extends MapReduceBase implements Reducer<Text,
IntWritable, Text, IntWritable> {
    public void reduce(Text t_key, Iterator<IntWritable> values,
    OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException
    {
        Text key = t_key;
        int frequencyForCountry = 0;
        while (values.hasNext())
        {
        IntWritable value = (IntWritable) values.next();
        frequencyForCountry += value.get();
        }
        output.collect(key, new IntWritable(frequencyForCountry));
    }
}
```

Driver Class:

```
1  package SalesCountry;
2  import org.apache.hadoop.fs.Path;
3  import org.apache.hadoop.io.*;
4  import org.apache.hadoop.mapred.*;
5
6  public class SalesCountryDriver
7  {
8      public static void main(String[] args) {
9          JobClient my_client = new JobClient();
10         JobConf job_conf = new JobConf(SalesCountryDriver.class);
11         job_conf.setJobName("SalePerCountry");
12         job_conf.setOutputKeyClass(Text.class);
13         job_conf.setOutputValueClass(IntWritable.class);
14         job_conf.setMapperClass(SalesCountry.SalesMapper.class);
15         job_conf.setReducerClass(SalesCountry.SalesCountryReducer.class);
16         job_conf.setInputFormat(TextInputFormat.class);
17         job_conf.setOutputFormat(TextOutputFormat.class);
18         //arg[0] = name of input directory on HDFS, and arg[1] = name of
19         output directory to be created to store the output file.
20         FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
21         FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));
22         my_client.setConf(job_conf);
23         try {
24         JobClient.runJob(job_conf);
25         } catch (Exception e) {
26         e.printStackTrace();
27         }
28     }
29 }
30
31
32
```

Input File:

Pune

Mumbai

Nashik

Pune

Nashik

Kolapur

Practical Number: 13

Title: Write a simple program in SCALA using Apache Spark framework.

Code:

Sample Code to print Statement

```
object ExampleString {

    def main(args: Array[String]) {
        //declare and assign string variable "text"
        val text : String = "You are reading SCALA programming language.";
        //print the value of string variable "text"
        println("Value of text is: " + text);
    }
}
```

Output:

```
Result
CPU Time: 4.73 sec(s), Memory: 140148 kilobyte(s)

Value of text is: You are reading SCALA programming language.

warning: 1 deprecation (since 2.13.0); re-run with -deprecation for details
```

Scala program to find a number is positive, negative or positive.

```
object ExCheckNumber {
    def main(args: Array[String]) {
        /**declare a variable*/
        var number= (-100);
        if(number==0){
            println("number is zero");
        }
        else if(number>0){
            println("number is positive");
        }
        else{
            println("number is negative");
        }
    }
}
```

Output:

```
Result
CPU Time: 6.77 sec(s), Memory: 143140 kilobyte(s)

number is negative

warning: 1 deprecation (since 2.13.0); re-run with -deprecation for details
```

Scala program to print your name

```
1
2  object ExPrintName {
3      def main(args: Array[String]) {
4          println("My name is Mike!")
5      }
6  }
```

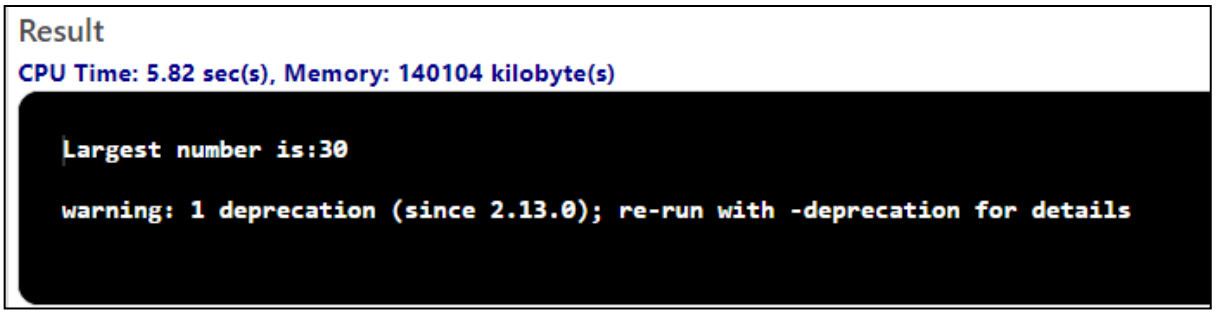
Output:

```
Result
CPU Time: 8.29 sec(s), Memory: 153080 kilobyte(s)

My name is Mike!

warning: 1 deprecation (since 2.13.0); re-run with -deprecation for details
```

Scala Program to find largest number among two numbers.

```
1  object ExFindLargest {
2      def main(args: Array[String]) {
3          var number1=20;
4          var number2=30;
5          var x = 10;
6          if( number1>number2){
7              println("Largest number is:" + number1);
8          }
9          else{
10             println("Largest number is:" + number2);
11         }
12     }
13 }
14
15
```

Output:


Result
CPU Time: 5.82 sec(s), Memory: 140104 kilobyte(s)
Largest number is:30
warning: 1 deprecation (since 2.13.0); re-run with -deprecation for details