Assignment No.:1, Name: Tejaswini Anil Rathod, Roll No.:28, Div:B

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.read_csv("Iris.csv")
```

In [3]:
```
df
```

Out[3]:

	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	NaN	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 [4]:
```
df.isnull()
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	True	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 [5]:
```
df.isnull().any()
```

Out[5]:
```
Id               False
SepalLengthCm    False
SepalWidthCm     False
PetalLengthCm     True
PetalWidthCm     False
Species          False
dtype: bool
```

In [6]:
```
df.dtypes
```

Out[6]:
```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [7]:
```
df["Species"].unique()
```

Out[7]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=o
bject)
```

In [9]:
```
df["Species"]=df["Species"].replace({'Iris-setosa':1,'Iris-versicolo
```

In [10]:
```
df.dtypes
```

Out[10]:
```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species            int64
dtype: object
```

In [ ]:
```

```

Assignment No.:2, Name: Tejaswini Anil Rathod, Roll No.:28, Div:B

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv("Academic_performace.csv")
```

In [3]:
```
df
```

Out[3]:

	Sno	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Sei
0	1	M	KW	KuwaIT	lowerlevel	G-04	A	IT	
1	2	M	KW	KuwaIT	lowerlevel	G-04	A	IT	
2	3	M	KW	KuwaIT	lowerlevel	G-04	A	IT	
3	4	M	KW	KuwaIT	lowerlevel	G-04	A	IT	
4	5	M	KW	KuwaIT	lowerlevel	G-04	A	IT	
...	...	...	...	...	...	...	...	...	
475	476	F	Jordan	Jordan	MiddleSchool	G-08	A	Chemistry	
476	477	F	Jordan	Jordan	MiddleSchool	G-08	A	Geology	
477	478	F	Jordan	Jordan	MiddleSchool	G-08	A	Geology	
478	479	F	Jordan	Jordan	MiddleSchool	G-08	A	History	
479	480	F	Jordan	Jordan	MiddleSchool	G-08	A	History	

480 rows × 18 columns

In [4]:
```
df.head()
```

Out[4]:

	Sno	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	R
0	1	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	
1	2	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	
2	3	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	
3	4	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	
4	5	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	

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

Out[5]:

	Sno	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Sei
475	476	F	Jordan	Jordan	MiddleSchool	G-08	A	Chemistry	
476	477	F	Jordan	Jordan	MiddleSchool	G-08	A	Geology	
477	478	F	Jordan	Jordan	MiddleSchool	G-08	A	Geology	
478	479	F	Jordan	Jordan	MiddleSchool	G-08	A	History	
479	480	F	Jordan	Jordan	MiddleSchool	G-08	A	History	

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

Out[6]:

	Sno	raisedhands	VisITedResources	AnnouncementsView	Discussion
count	480.000000	480.000000	480.000000	480.000000	478.000000
mean	240.500000	46.775000	54.797917	38.462500	43.278243
std	138.708327	30.779223	33.080007	30.095579	27.646238
min	1.000000	0.000000	0.000000	0.000000	1.000000
25%	120.750000	15.750000	20.000000	14.000000	20.000000
50%	240.500000	50.000000	65.000000	33.000000	39.000000
75%	360.250000	75.000000	84.000000	58.000000	70.000000
max	480.000000	100.000000	99.000000	350.000000	99.000000

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 18 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Sno                       480 non-null    int64
 1   gender                    480 non-null    object
 2   NationalITy               480 non-null    object
 3   PlaceofBirth              480 non-null    object
 4   StageID                   480 non-null    object
 5   GradeID                   480 non-null    object
 6   SectionID                 480 non-null    object
 7   Topic                     480 non-null    object
 8   Semester                  480 non-null    object
 9   Relation                  480 non-null    object
 10  raisedhands               480 non-null    int64
 11  VisITedResources          480 non-null    int64
 12  AnnouncementsView         480 non-null    int64
 13  Discussion                478 non-null    float64
 14  ParentAnsweringSurvey     480 non-null    object
 15  ParentschoolSatisfaction  480 non-null    object
 16  StudentAbsenceDays        480 non-null    object
 17  Class                     480 non-null    object
dtypes: float64(1), int64(4), object(13)
memory usage: 67.6+ KB
```

In [8]:
```
df.shape
```

Out[8]:
```
(480, 18)
```

In [9]:
```
df.isnull().any().any()
```

Out[9]:
```
True
```

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
Sno                       0
gender                    0
NationalITy               0
PlaceofBirth              0
StageID                   0
GradeID                   0
SectionID                 0
Topic                     0
Semester                  0
Relation                  0
raisedhands               0
VisITedResources          0
AnnouncementsView         0
Discussion                2
ParentAnsweringSurvey     0
ParentschoolSatisfaction  0
StudentAbsenceDays        0
Class                     0
dtype: int64
```

In [11]:
```
avg_val = df["Discussion"].astype("float").mean()
avg_val
```

Out[11]:
```
43.27824267782427
```

In [12]:
```
df["Discussion"].replace(np.NaN, avg_val, inplace=True)
```

In [13]:
```
df.isnull().sum()
```

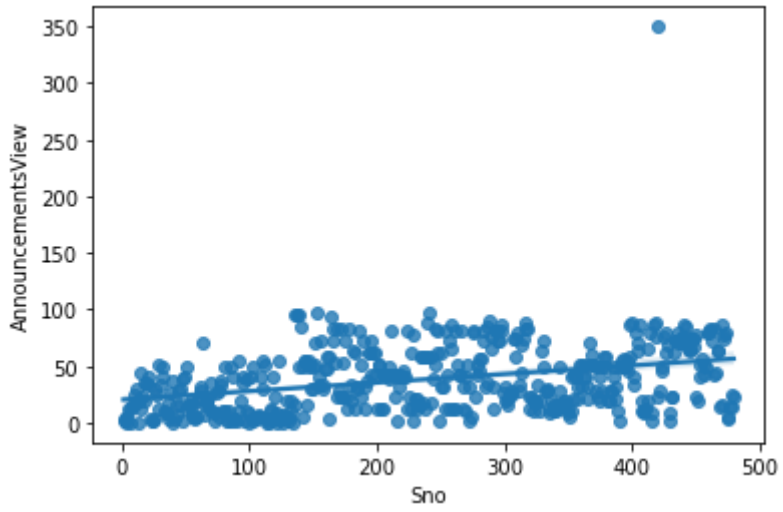
Out[13]:
```
Sno                       0
gender                    0
NationalITy               0
PlaceofBirth              0
StageID                   0
GradeID                   0
SectionID                 0
Topic                     0
Semester                  0
Relation                  0
raisedhands               0
VisITedResources          0
AnnouncementsView         0
Discussion                0
ParentAnsweringSurvey     0
ParentschoolSatisfaction  0
StudentAbsenceDays        0
Class                     0
dtype: int64
```

Step-II

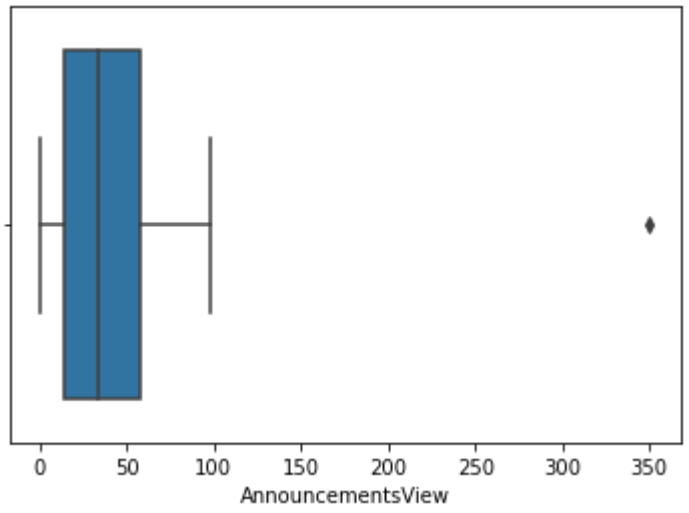
Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.

In [14]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
```

In [15]:
```
sns.regplot(x='Sno', y='AnnouncementsView', data=df)
plt.show()
```



In [16]:
```
sns.boxplot(x=df['AnnouncementsView'])
plt.show()
```



In [17]:
```
z = np.abs(stats.zscore(df['AnnouncementsView']))
print(z)
```

```
0      1.212821
1      1.179559
2      1.279345
3      1.113034
4      0.880199
         ...
475    1.113034
476    0.813675
477    0.447792
478    0.813675
479    0.514316
Name: AnnouncementsView, Length: 480, dtype: float64
```

In [18]:
```
threshold = 3
print(np.where(z > 3))
```

```
(array([419]),)
```

In [19]:
```
z[419]
```

Out[19]:
```
10.3624031636167
```

Step-III

Apply data transformations on at least one of the variables

In [20]:
```
df1 = pd.DataFrame({ 'Income': [15000, 1800, 120000, 10000],
'Age': [25, 18, 42, 51],
'Department': ['HR','Legal','Marketing','Management']})
```

In [21]:
```
df1
```

Out[21]:

	Income	Age	Department
0	15000	25	HR
1	1800	18	Legal
2	120000	42	Marketing
3	10000	51	Management

In [23]:
```
df1_scaled = df1.copy()
col_names = ['Income', 'Age']
features = df1_scaled[col_names]
```

In [24]:
```
features
```

Out[24]:

	Income	Age
0	15000	25
1	1800	18
2	120000	42
3	10000	51

In [25]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df1_scaled[col_names] = scaler.fit_transform(features.values)
```

In [26]:
```
print(df1_scaled[col_names])
```

```
     Income       Age
0  0.111675  0.212121
1  0.000000  0.000000
2  1.000000  0.727273
3  0.069374  1.000000
```

In [ ]:
```

```

Assignment:3 , Name: Rathod Tejaswini Anil, Roll No.:28, Div:B

In [1]:
```
import numpy as np
import pandas as pd
import statistics as st
```

In [2]:
```
df = pd.read_csv("Mall_Customers.csv")
```

In [3]:
```
df
```

Out[3]:

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

1.Mean

In [5]:
```
df.mean() # mean of all columns
```

Out[5]:
```
CustomerID                  100.50
Age                          38.85
Annual Income (k$)           60.56
Spending Score (1-100)       50.20
dtype: float64
```

In [6]:
```
df.loc[:,'Age'].mean() # mean of specific column
```

Out[6]:
```
38.85
```

In [7]:
```
df.mean(axis=1)[0:4] # mean row wise
```

Out[7]:
```
0    18.50
1    29.75
2    11.25
3    30.00
dtype: float64
```

2.Median

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

Out[8]:
```
CustomerID                  100.5
Age                          36.0
Annual Income (k$)           61.5
Spending Score (1-100)       50.0
dtype: float64
```

In [9]:
```
df.loc[:,'Age'].median()
```

Out[9]:
```
36.0
```

In [10]:
```
df.median(axis=1)[0:4]
```

Out[10]:
```
0    17.0
1    18.0
2    11.0
3    19.5
dtype: float64
```

3.Mode

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

Out[11]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Female	32.0	54.0	42.0
1	2	NaN	NaN	78.0	NaN
2	3	NaN	NaN	NaN	NaN
3	4	NaN	NaN	NaN	NaN
4	5	NaN	NaN	NaN	NaN
...	...	...	...	...	...
195	196	NaN	NaN	NaN	NaN
196	197	NaN	NaN	NaN	NaN
197	198	NaN	NaN	NaN	NaN
198	199	NaN	NaN	NaN	NaN
199	200	NaN	NaN	NaN	NaN

200 rows × 5 columns

In [12]:
```
df.loc[:,'Age'].mode()
```

Out[12]:
```
0    32
dtype: int64
```

4.Minimum

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

Out[15]:
```
CustomerID                    1
Genre                    Female
Age                          18
Annual Income (k$)           15
Spending Score (1-100)        1
dtype: object
```

In [16]:
```
df.loc[:,'Age'].min(skipna = False)
```

Out[16]:
```
18
```

5.Maximum

In [17]:
```
df.max()
```

Out[17]:
```
CustomerID                  200
Genre                      Male
Age                          70
Annual Income (k$)          137
Spending Score (1-100)       99
dtype: object
```

In [18]:
```
df.loc[:,'Age'].max(skipna = False)
```

Out[18]:
```
70
```

6.Standard Deviation

In [19]:
```
df.std()
```

Out[19]:
```
CustomerID                57.879185
Age                       13.969007
Annual Income (k$)        26.264721
Spending Score (1-100)    25.823522
dtype: float64
```

In [20]:
```
df.loc[:,'Age'].std()
```

Out[20]:
```
13.969007331558883
```

In [21]:
```
df.std(axis=1)[0:4]
```

Out[21]:
```
0    15.695010
1    35.074920
2     8.057088
3    32.300671
dtype: float64
```

In [22]:
```
df.groupby(['Genre'])['Age'].mean()
```

Out[22]:
```
Genre
Female    38.098214
Male      39.806818
Name: Age, dtype: float64
```

In [23]:
```
df_u=df.rename(columns= {'Annual Income (k$)':'Income'}, inplace= Fal
```

In [24]:
```
df_u
```

Out[24]:

	CustomerID	Genre	Age	Income	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 [25]:
```
df_u.groupby(['Genre']).Income.mean()
```

Out[25]:
```
Genre
Female    59.250000
Male      62.227273
Name: Income, dtype: float64
```

In [26]:
```
from sklearn import preprocessing
enc = preprocessing.OneHotEncoder()
enc_df = pd.DataFrame(enc.fit_transform(df[['Genre']]).toarray())
enc_df
```

Out[26]:

	0	1
0	0.0	1.0
1	0.0	1.0
2	1.0	0.0
3	1.0	0.0
4	1.0	0.0
...	...	...
195	1.0	0.0
196	1.0	0.0
197	0.0	1.0
198	0.0	1.0
199	0.0	1.0

200 rows × 2 columns

In [27]:
```
df_encode =df_u.join(enc_df)
df_encode
```

Out[27]:

	CustomerID	Genre	Age	Income	Spending Score (1-100)	0	1
0	1	Male	19	15	39	0.0	1.0
1	2	Male	21	15	81	0.0	1.0
2	3	Female	20	16	6	1.0	0.0
3	4	Female	23	16	77	1.0	0.0
4	5	Female	31	17	40	1.0	0.0
...	...	...	...	...	...	...	...
195	196	Female	35	120	79	1.0	0.0
196	197	Female	45	126	28	1.0	0.0
197	198	Male	32	126	74	0.0	1.0
198	199	Male	32	137	18	0.0	1.0
199	200	Male	30	137	83	0.0	1.0

200 rows × 7 columns

In [28]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [30]:
```
df_Mall_Customers = pd.read_csv("Mall_Customers.csv")
df_Mall_Customers.head()
```

Out[30]:

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

In [ ]:

In [ ]:

Assignment:4, Name: Rathod Tejaswini Anil, Roll No.:28, Div:B

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
Boston = pd.read_csv("Boston.csv")
Boston.head()
```

Out[2]:

	Unnamed: 0	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7

In [3]:
```
Boston.info()
Boston.describe()
```

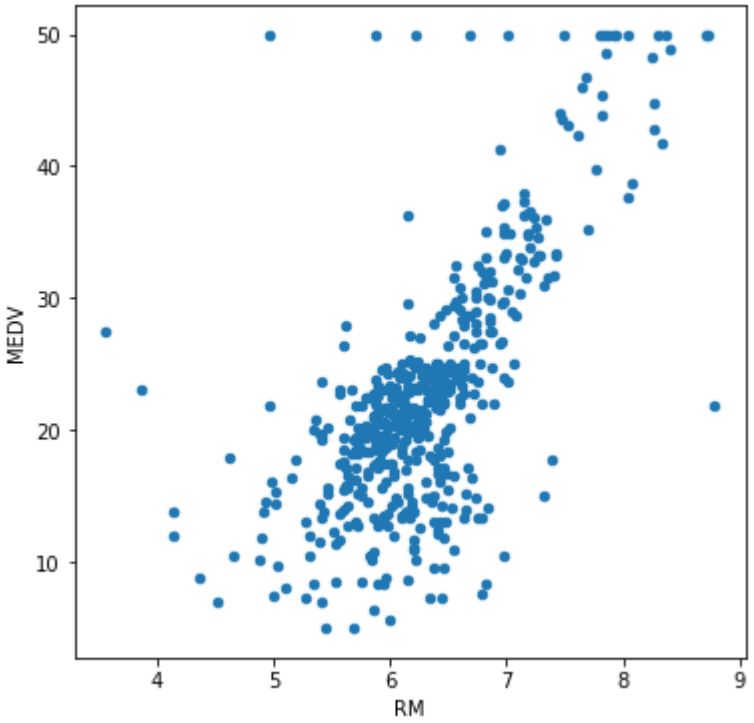
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  506 non-null    int64  
 1   CRIM        506 non-null    float64
 2   ZN          506 non-null    float64
 3   INDUS       506 non-null    float64
 4   CHAS        506 non-null    int64  
 5   NOX         506 non-null    float64
 6   RM          506 non-null    float64
 7   AGE         506 non-null    float64
 8   DIS         506 non-null    float64
 9   RAD         506 non-null    int64  
 10  TAX         506 non-null    int64  
 11  PTRATIO     506 non-null    float64
 12  BLACK       506 non-null    float64
 13  LSTAT       506 non-null    float64
 14  MEDV        506 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 59.4 KB
```

Out[3]:

	Unnamed: 0	CRIM	ZN	INDUS	CHAS	NOX	RM
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	253.500000	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634
std	146.213884	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617
min	1.000000	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000
25%	127.250000	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500
50%	253.500000	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500
75%	379.750000	3.677082	12.500000	18.100000	0.000000	0.624000	6.623500
max	506.000000	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000

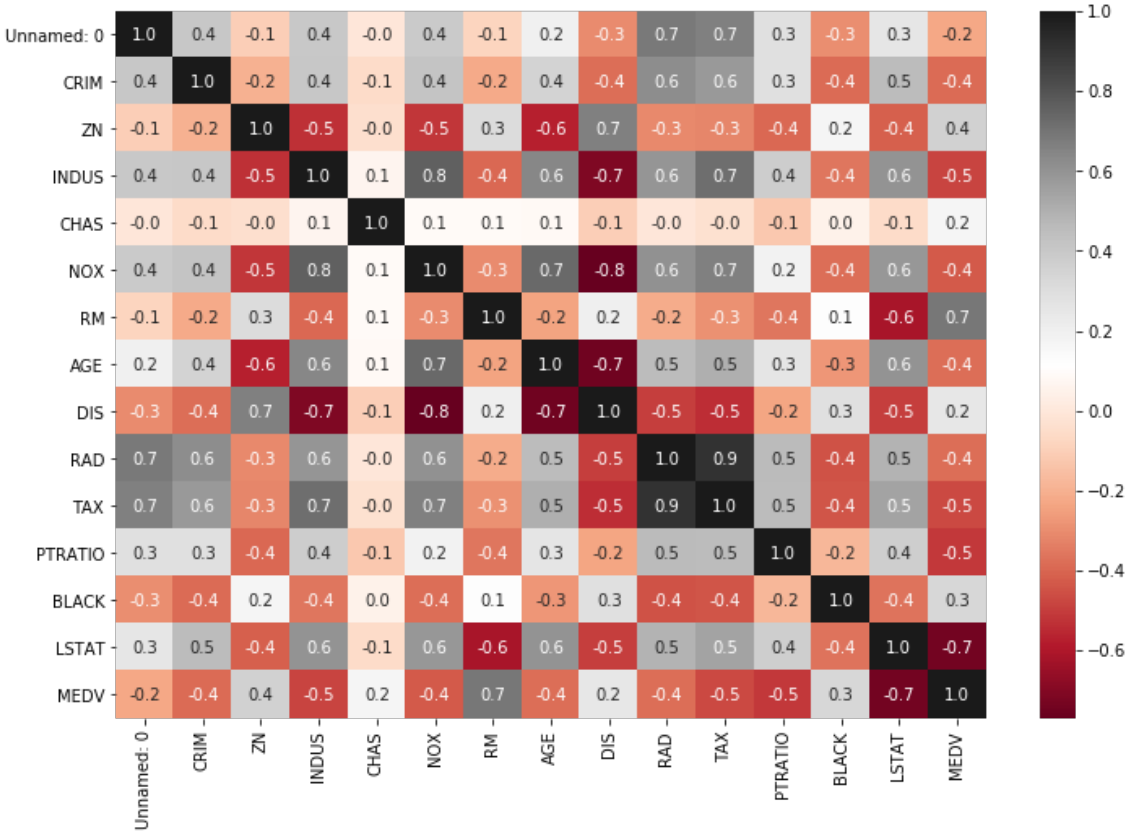
In [4]:
```
Boston.plot.scatter('RM', 'MEDV', figsize=(6, 6))
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7f22d5b49550>



In [5]:
```
plt.subplots(figsize=(12,8))
sns.heatmap(Boston.corr(), cmap = 'RdGy', annot = True, fmt = '.1f')
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f22d52f1750>



In [18]:
```
X = Boston[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS',
Y = Boston['MEDV']
```

In [13]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [19]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0
```

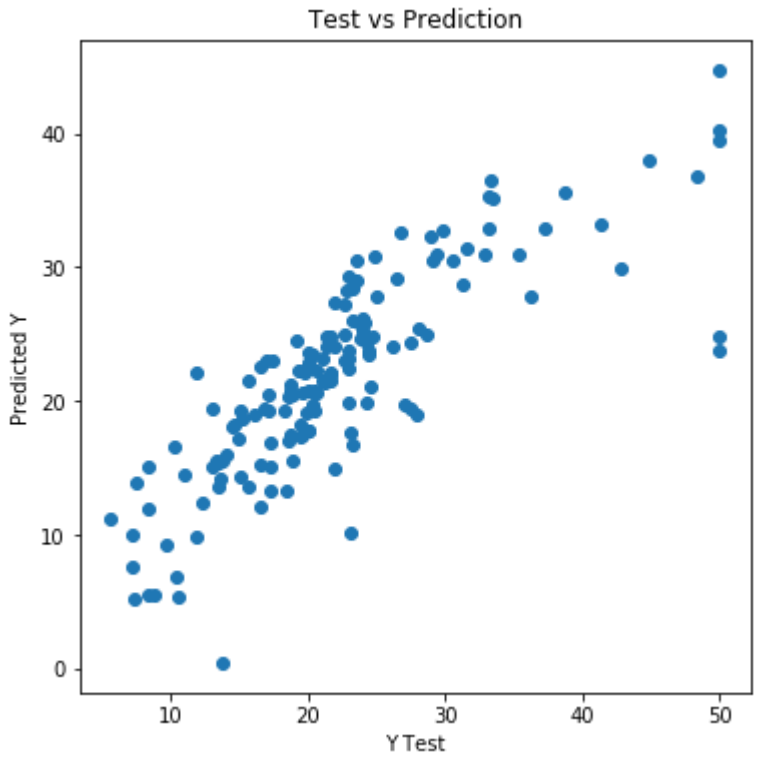
In [20]:
```
print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
print(f'Test Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')
```

```
Train Dataset Size - X: (354, 13), Y: (354,)
Test Dataset Size - X: (152, 13), Y: (152,)
```

In [21]:
```
lm = LinearRegression()
lm.fit(X_train,Y_train)
predictions = lm.predict(X_test)
```

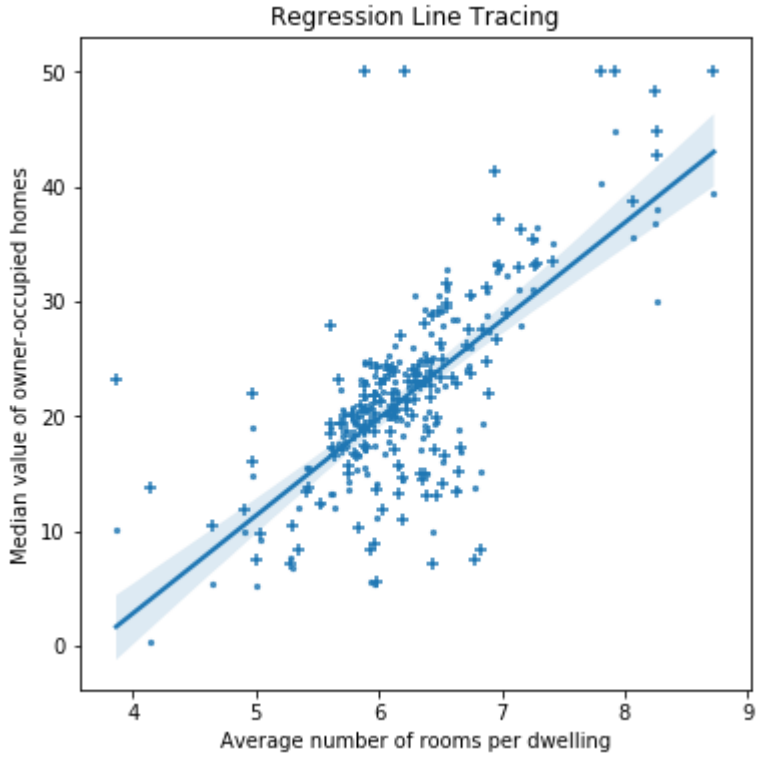
In [22]:
```
plt.figure(figsize=(6, 6))
plt.scatter(Y_test, predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
plt.title('Test vs Prediction')
```

Out[22]:
```
Text(0.5, 1.0, 'Test vs Prediction')
```



In [23]:
```
plt.figure(figsize=(6, 6))
sns.regplot(x = X_test['RM'], y = predictions, scatter_kws={'s':5})
plt.scatter(X_test['RM'], Y_test, marker = '+')
plt.xlabel('Average number of rooms per dwelling')
plt.ylabel('Median value of owner-occupied homes')
plt.title('Regression Line Tracing')
```

Out[23]:
```
Text(0.5, 1.0, 'Regression Line Tracing')
```



In [26]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_test, pre
print('Mean Square Error:', metrics.mean_squared_error(Y_test, predic
print('Root Mean Square Error:', np.sqrt(metrics.mean_squared_error(Y
```

```
Mean Absolute Error: 3.6099040603818233
Mean Square Error: 27.195965766883337
Root Mean Square Error: 5.214975145375416
```

In [27]:
```
coefficients = pd.DataFrame(lm.coef_.round(2), X.columns)
coefficients.columns = ['coefficients']
coefficients
```

Out[27]:

	coefficients
CRIM	-0.12
ZN	0.04
INDUS	0.01
CHAS	2.51
NOX	-16.23
RM	3.86
AGE	-0.01
DIS	-1.50
RAD	0.24
TAX	-0.01
PTRATIO	-1.02
BLACK	0.01
LSTAT	-0.49

In [ ]:

Assignment No.:5, Name: Rathod Tejaswini Anil, Div: B, Roll No.:28

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
df = pd.read_csv('Social_Network_ads.csv')
df.head()
```

Out[1]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [2]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [3]:
```
df.describe()
```

Out[3]:

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

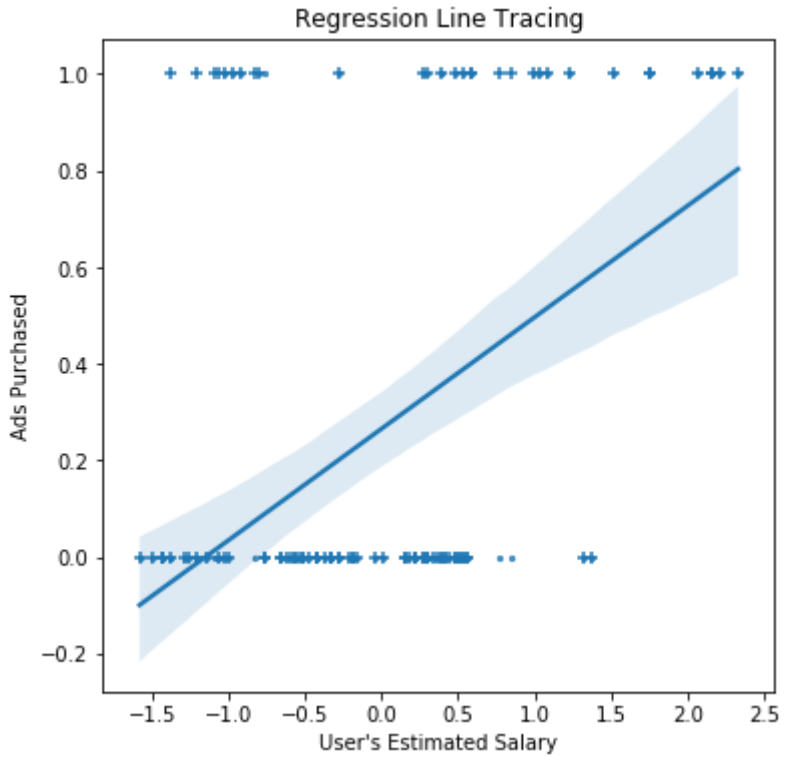
In [4]:
```
X = df[['Age', 'EstimatedSalary']]
Y = df['Purchased']
```

In [5]:
```
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 [6]:
```
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[6]: Text(0.5, 1.0, 'Regression Line Tracing')

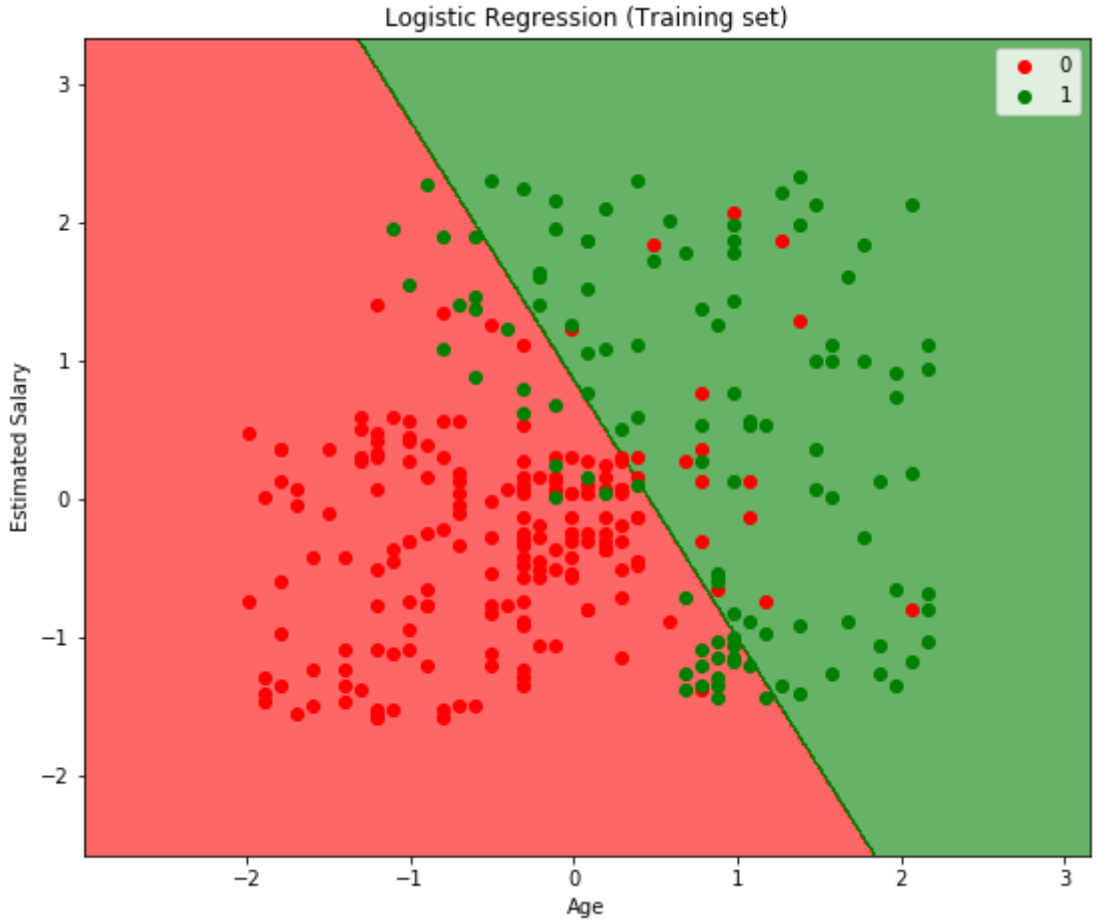


Confusion Matrix

In [7]:
```
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)
```

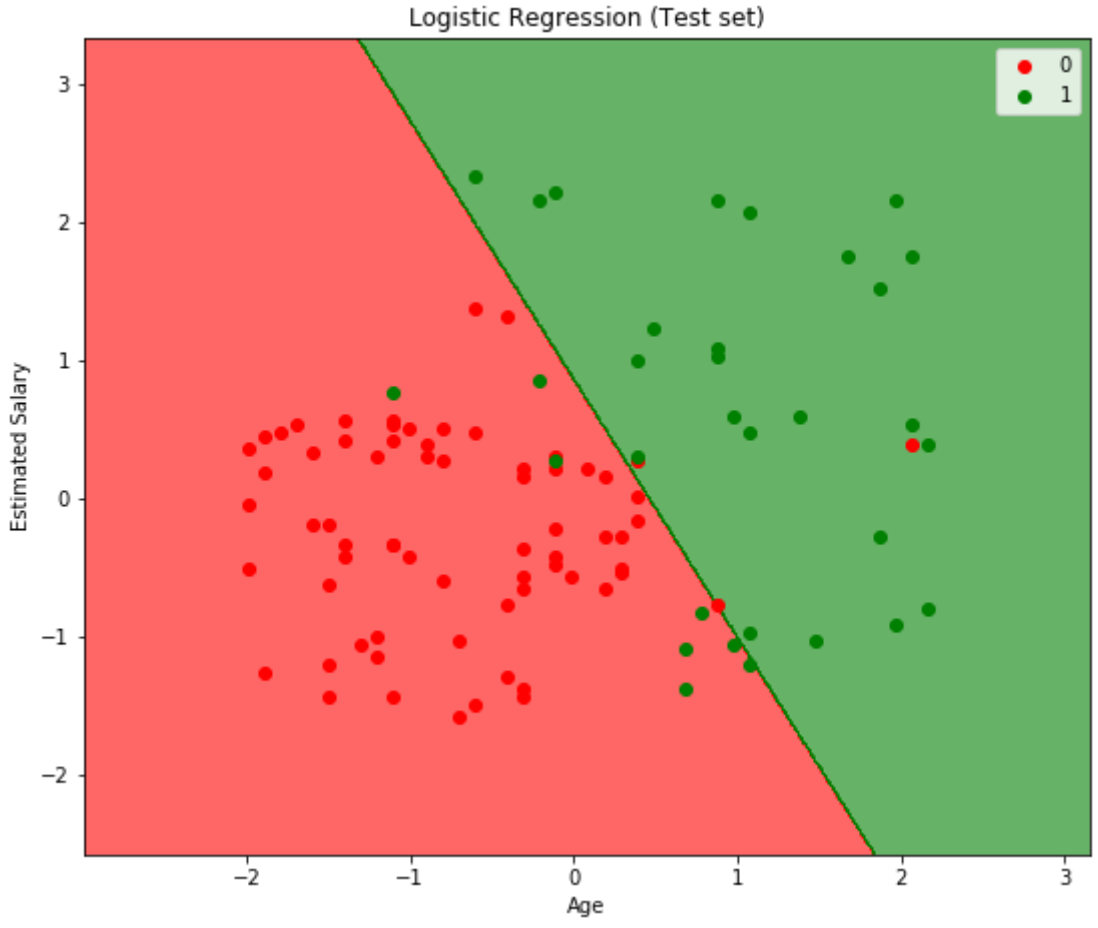
```
Confusion matrix :

| 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 [8]:
```
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 in enumerate(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 [9]:
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, Y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
                     np.arange(start = X_set[:, 1].min() - 1, stop = )
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 in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
        c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, whic
h should be avoided as value-mapping will have precedence in case i
ts length matches with 'x' & 'y'.  Please use a 2-D array with a si
ngle row if you really want to specify the same RGB or RGBA value f
or all points.
'c' argument looks like a single numeric RGB or RGBA sequence, whic
h should be avoided as value-mapping will have precedence in case i
ts length matches with 'x' & 'y'.  Please use a 2-D array with a si
ngle row if you really want to specify the same RGB or RGBA value f
or all points.
```



In [ ]:

Assignment No.:6 Name: Rathod Tejaswini Anil, Div:B, Roll No.: 28

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
df = pd.read_csv('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	1.4	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.info()
```

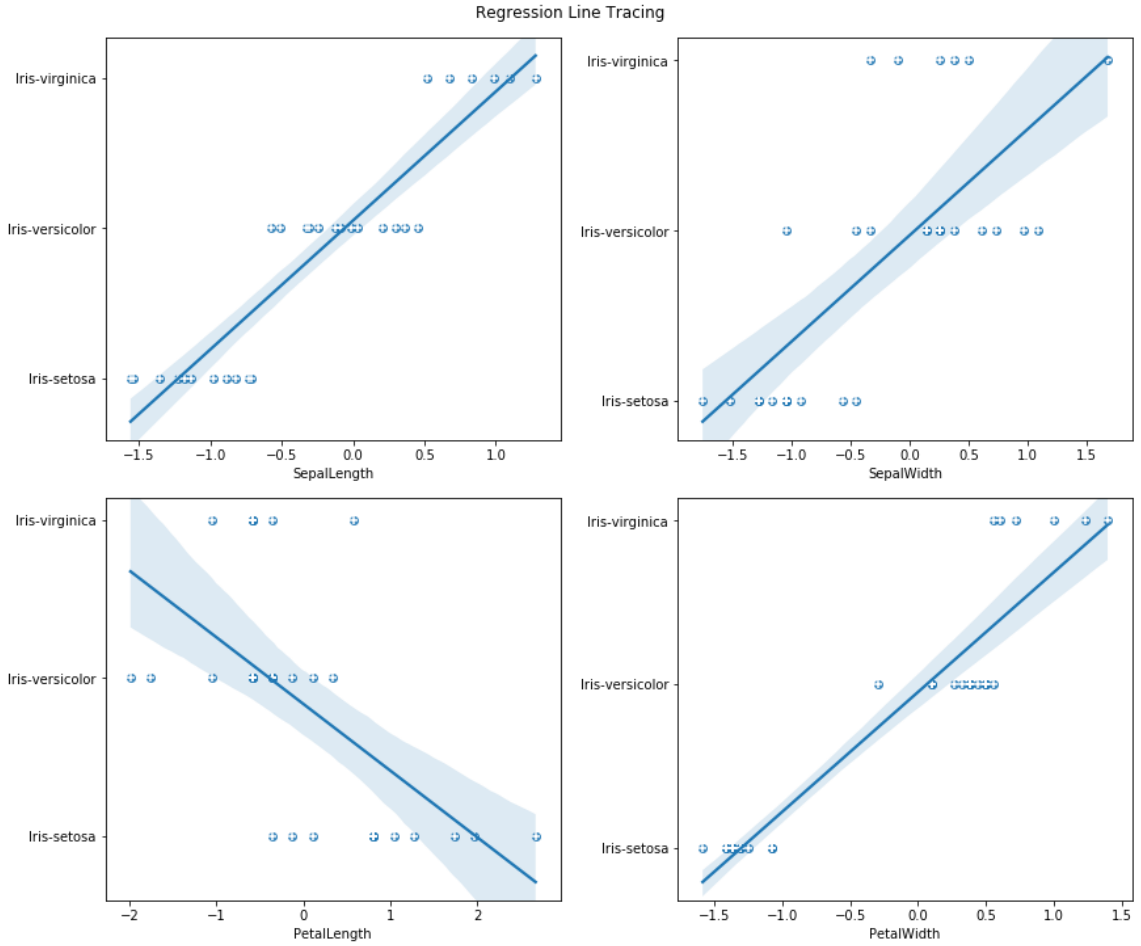
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   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 [7]:
```
X = df.iloc[:, :4].values
Y = df['Species'].values
```

In [9]:
```
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: (120, 4), Y: (120,)
Test Dataset Size - X: (30, 4), Y: (30,)
```

In [10]:
```
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
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 = '+',
    axs[x, y].set_xlabel(df.columns[i + 1][:-2])
```



Confusion matrix

In [12]:
```
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)
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 [ ]:

Assignment No.:7(A), Name: Tejaswini Anil Rathod, Roll No.: 28, Div:B

In [1]:
```
#Download the required packages
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]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/student/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /home/student/nltk_dat
a...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/student/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
```

Out[1]:
```
True
```

In [3]:
```
#Initialize the text
#Sentence Tokenization
text= "Tokenization is the first step in text analytics.The process o
from nltk.tokenize import sent_tokenize
tokenized_text= sent_tokenize(text)
print(tokenized_text)
```

```
['Tokenization is the first step in text analytics.The process of b
reaking down a text paragraph into smallerchunks such as words or s
entences is called Tokenization.']
```

In [4]:
```
#Word Tokenization
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analy
tics.The', 'process', 'of', 'breaking', 'down', 'a', 'text', 'parag
raph', 'into', 'smallerchunks', 'such', 'as', 'words', 'or', 'sente
nces', 'is', 'called', 'Tokenization', '.']
```

In [5]:
```
# print stop words of English
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

```
{'very', 'does', 'most', "shan't", 'they', 'not', 'their', 'doesn',
'and', 'are', 'had', 'to', "couldn't", "mustn't", "shouldn't", 'whe
re', "won't", 'out', 'it', "isn't", 'its', 'which', 'other', 'or',
'i', 'again', "didn't", 'as', 'of', 'few', 'aren', 'y', 'themselves
', 'some', 'yourselves', 'our', "doesn't", 'did', 'off', 'up', 'do
', 're', 'during', 'hers', 'after', 'o', 'below', 'a', "it's", 've
', 'hasn', 'those', 'theirs', 'wasn', 'who', 'under', 'is', 'but',
'don', 't', "you've", 'am', 'being', 'because', 'can', 'your', 'dow
n', 's', 'why', 'this', 'such', 'ain', 'all', 'his', 'these', 'me',
'him', 'on', 'further', 'here', 'will', 'm', 'by', "mightn't", 'whi
le', 'he', 'ourselves', 'between', 'how', 'needn', 'before', 'them
', "she's", 'too', "weren't", 'shan', 'has', 'mightn', 'her', 'now
', 'couldn', 'at', 'for', 'wouldn', 'with', 'no', 'whom', "should've
", 'ours', 'have', 'both', 'won', 'there', "you'll", 'then', 'own
', 'that', 'when', "wouldn't", 'weren', 'been', 'isn', 'nor', 'she
', "you're", "don't", 'my', 'only', 'more', "that'll", 'in', 'if',
'was', 'ma', "hadn't", 'what', "hasn't", 'd', 'we', 'having', 'over
', 'herself', 'doing', 'yours', "aren't", "you'd", 'itself', 'from
', 'any', 'an', 'same', 'should', 'haven', 'the', 'than', 'once', '
against', 'above', "wasn't", 'through', 'shouldn', 'about', 'mustn
', "haven't", 'so', 'myself', 'into', 'yourself', "needn't", 'hadn
', 'each', 'you', 'didn', 'were', 'just', 'll', 'until', 'himself',
'be'}
```

In [7]:
```
#Removing Punctuations and Stop Word
text= "How to remove stop words with NLTK library in Python?"
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)
```

```
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with
', 'nltk', 'library', 'in', 'python', '?']
Filterd Sentence: ['remove', 'stop', 'words', 'nltk', 'library', 'p
ython', '?']
```

In [8]:
```
#Perform Stemming
from nltk.stem import PorterStemmer
e_words= ["wait", "waiting", "waited", "waits"]
ps =PorterStemmer()
for w in e_words:
    rootWord=ps.stem(w)
    print(rootWord)
```

```
wait
wait
wait
wait
```

In [9]:
```
#Perform Lemmatization
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.lemmatize
```

```
Lemma for studies is study
Lemma for studying is studying
Lemma for cries is cry
Lemma for cry is cry
```

In [ ]:
```
#Apply POS Tagging to text
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]))
```

Assignment No.:7(B), Name: Tejaswini Anil Rathod, Roll No.: 28, Div:B

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 [8]:
```
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 [10]:
```
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 [15]:
```
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[15]:
```
{'Planet': 0.6931471805599453,
 'Jupiter': 0.6931471805599453,
 'Sun': 0.6931471805599453,
 'Mars': 0.6931471805599453,
 'planet': 0.6931471805599453,
 'the': 0.0,
 'from': 0.6931471805599453,
 'fourth': 0.6931471805599453,
 'largest': 0.6931471805599453,
 'is': 0.0}
```

In [16]:
```
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[16]:

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

In [ ]:

Assignment No.:8, Name: Tejaswini Anil Rathod, Roll No.: 28, Div:B

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data = pd.read_csv('https://raw.githubusercontent.com/dphi-official/D
data.head()
```

Out[2]:

	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]:
```
data.shape
```

Out[3]:
```
(891, 12)
```

In [4]:
```
data.describe()
```

Out[4]:

	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 [5]:
```
data.describe(include = 'object')
```

Out[5]:

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Attalah, Mr. Sleiman	male	CA. 2343	G6	S
freq	1	577	7	4	644

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
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 [7]:
```
data['Age'] = data['Age'].fillna(np.mean(data['Age']))
```

In [8]:
```
data['Cabin'] = data['Cabin'].fillna(data['Cabin'].mode()[0])
```

In [9]:
```
data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0]
```

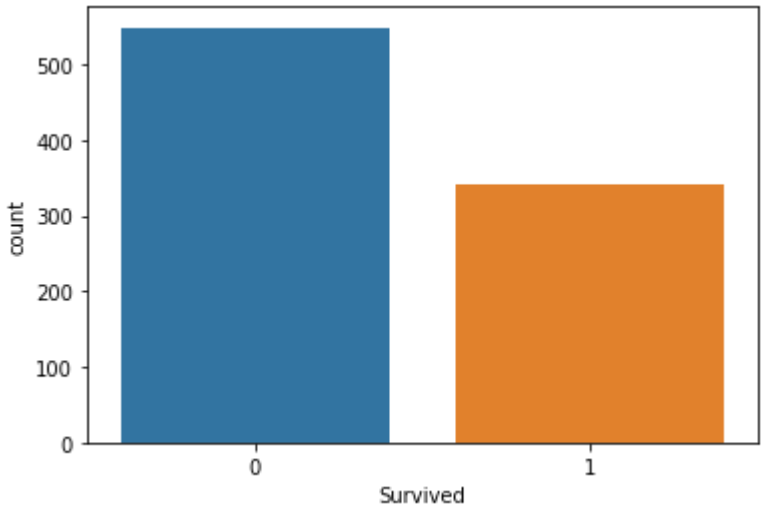
In [15]:
```
data.isnull().sum()
```

Out[15]:
```
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
```

CountPlot

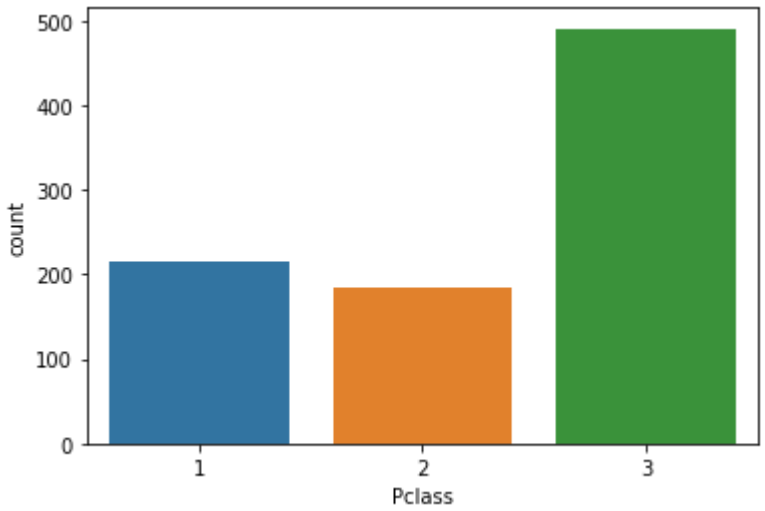
In [11]:
```
sns.countplot(x='Survived',data=data)
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f884b5775d0>
```



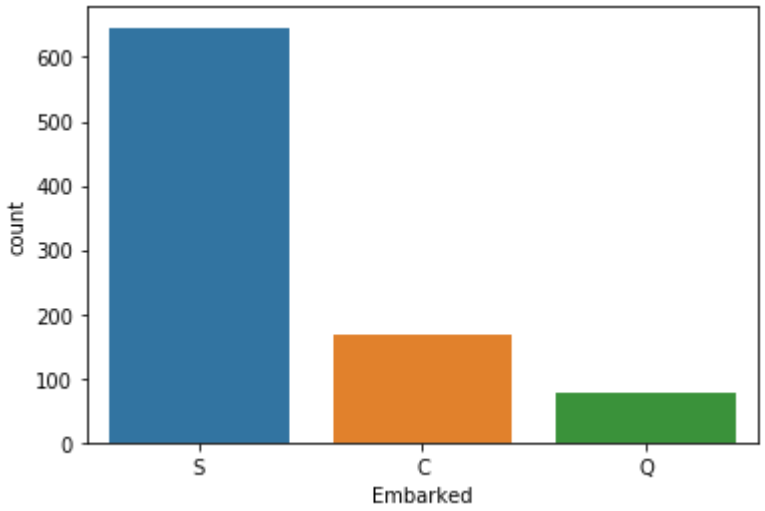
In [12]:
```
sns.countplot(x='Pclass',data=data)
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f884ad22510>
```



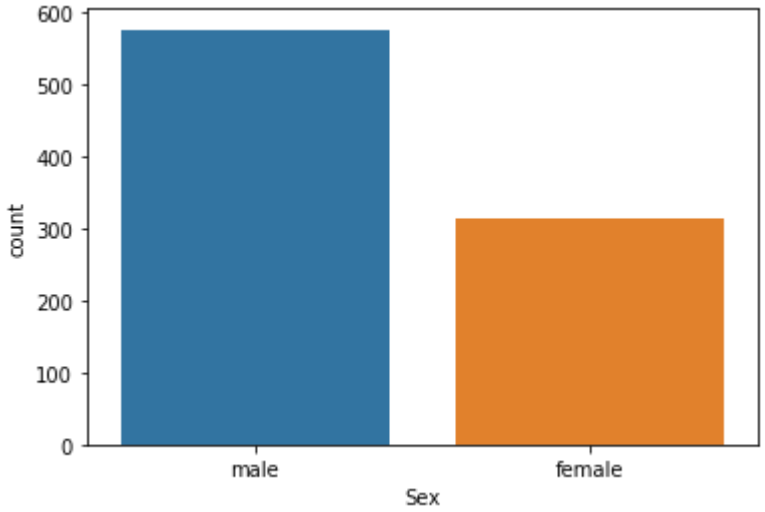
In [13]:
```
sns.countplot(x='Embarked',data=data)
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f884ac264d0>
```



In [14]:
```
sns.countplot(x='Sex',data=data)
```

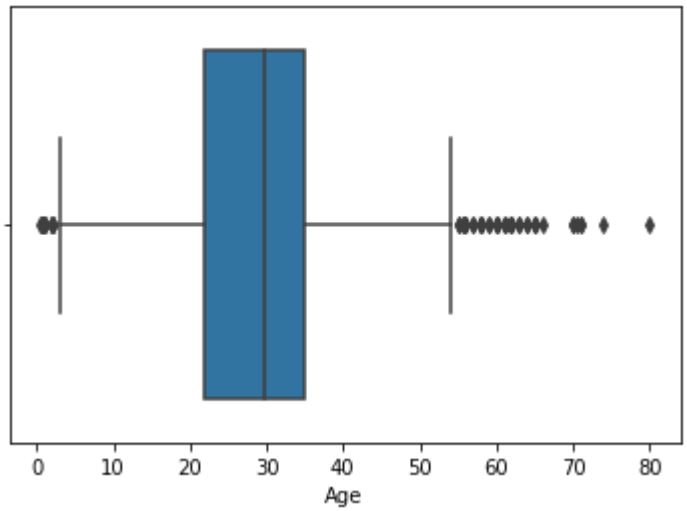
Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f884ac13e90>
```



BoxPlot

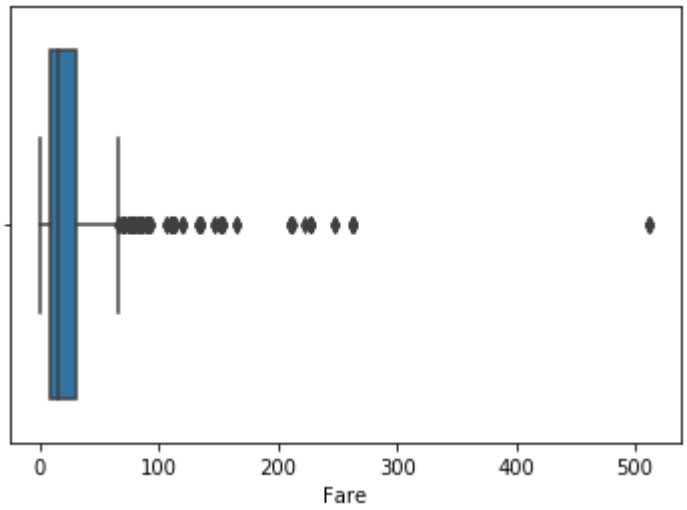
In [16]:
```
sns.boxplot(data['Age'])
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f884ab6c2d0>
```



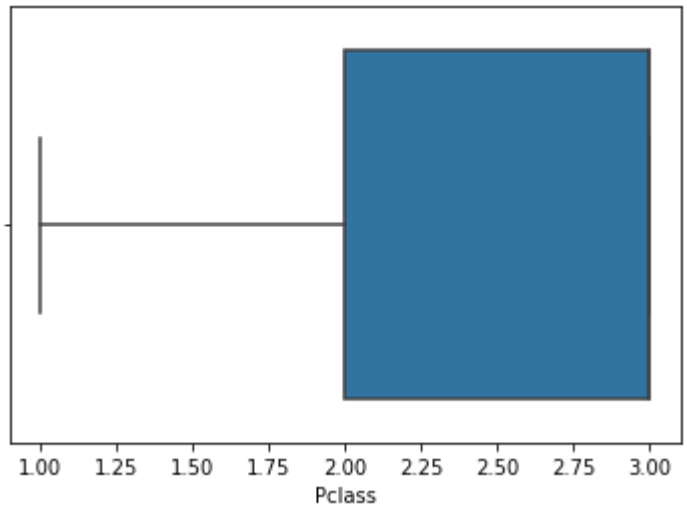
In [17]:
```
sns.boxplot(data['Fare'])
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f884ab5fc50>
```



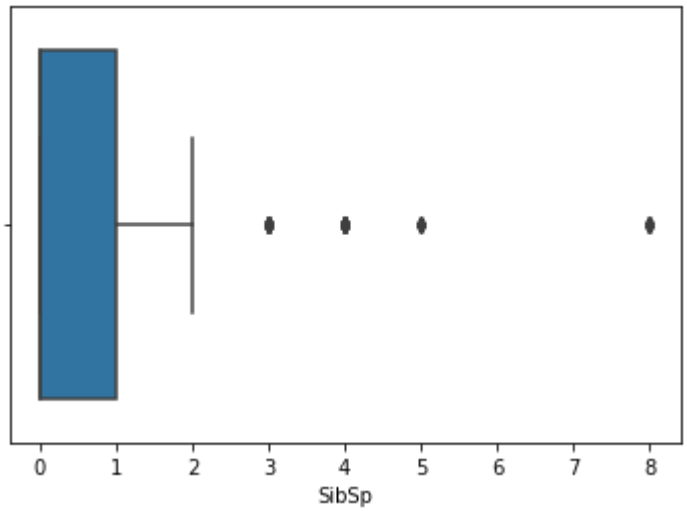
In [18]:
```
sns.boxplot(data['Pclass'])
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f884aad2350>
```



In [21]:
```
sns.boxplot(data['SibSp'])
```

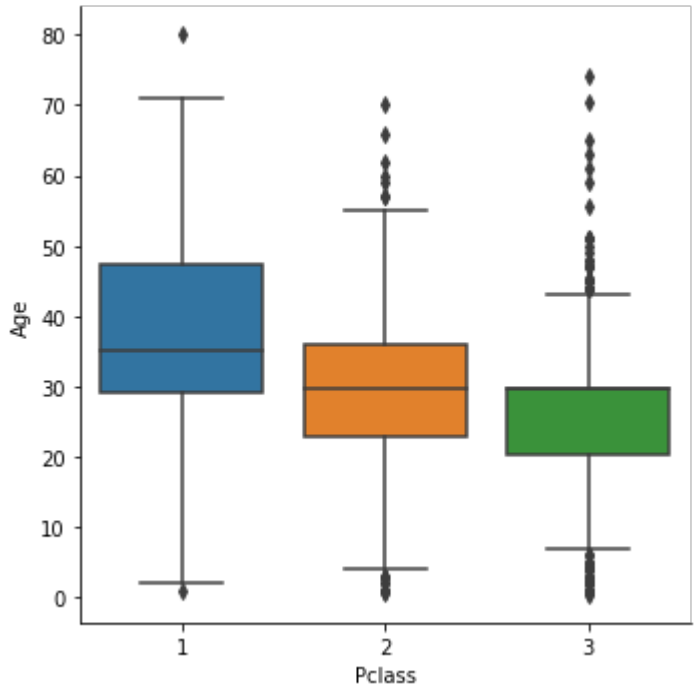
Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f884a9ade90>
```



CatPlot

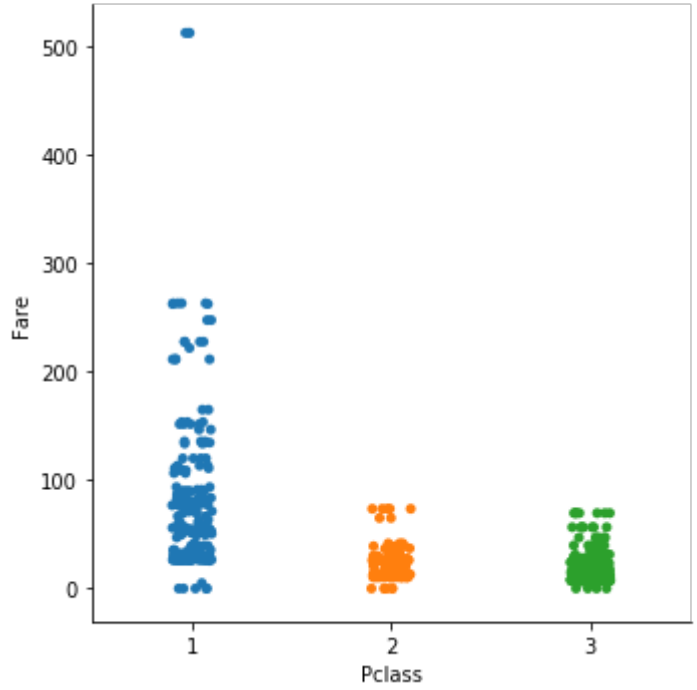
In [22]:
```
sns.catplot(x= 'Pclass', y = 'Age', data=data, kind = 'box')
```

Out[22]:
```
<seaborn.axisgrid.FacetGrid at 0x7f884a971910>
```



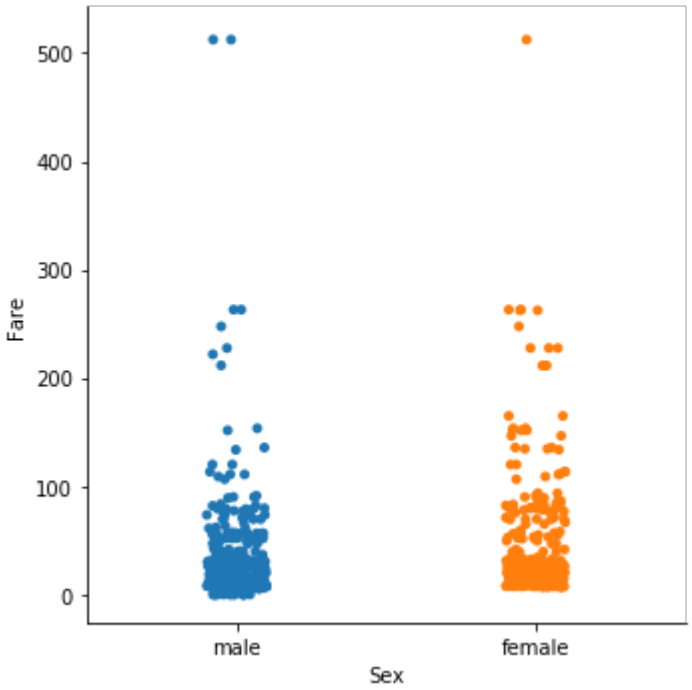
In [23]:
```
sns.catplot(x= 'Pclass', y = 'Fare', data=data, kind = 'strip')
```

Out[23]:
```
<seaborn.axisgrid.FacetGrid at 0x7f884a9bb790>
```



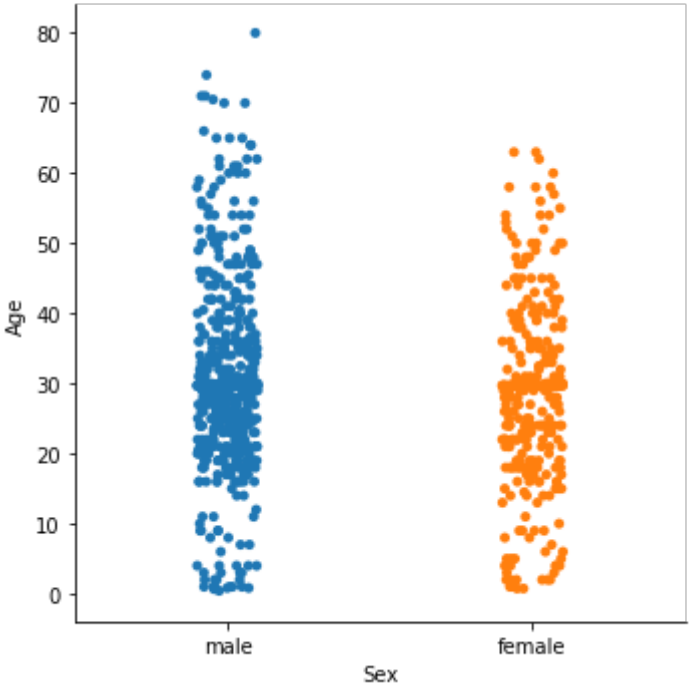
In [24]:
```
sns.catplot(x= 'Sex', y = 'Fare', data=data, kind = 'strip')
```

Out[24]:
```
<seaborn.axisgrid.FacetGrid at 0x7f884aa19190>
```



In [25]:
```
sns.catplot(x= 'Sex', y = 'Age', data=data, kind = 'strip')
```

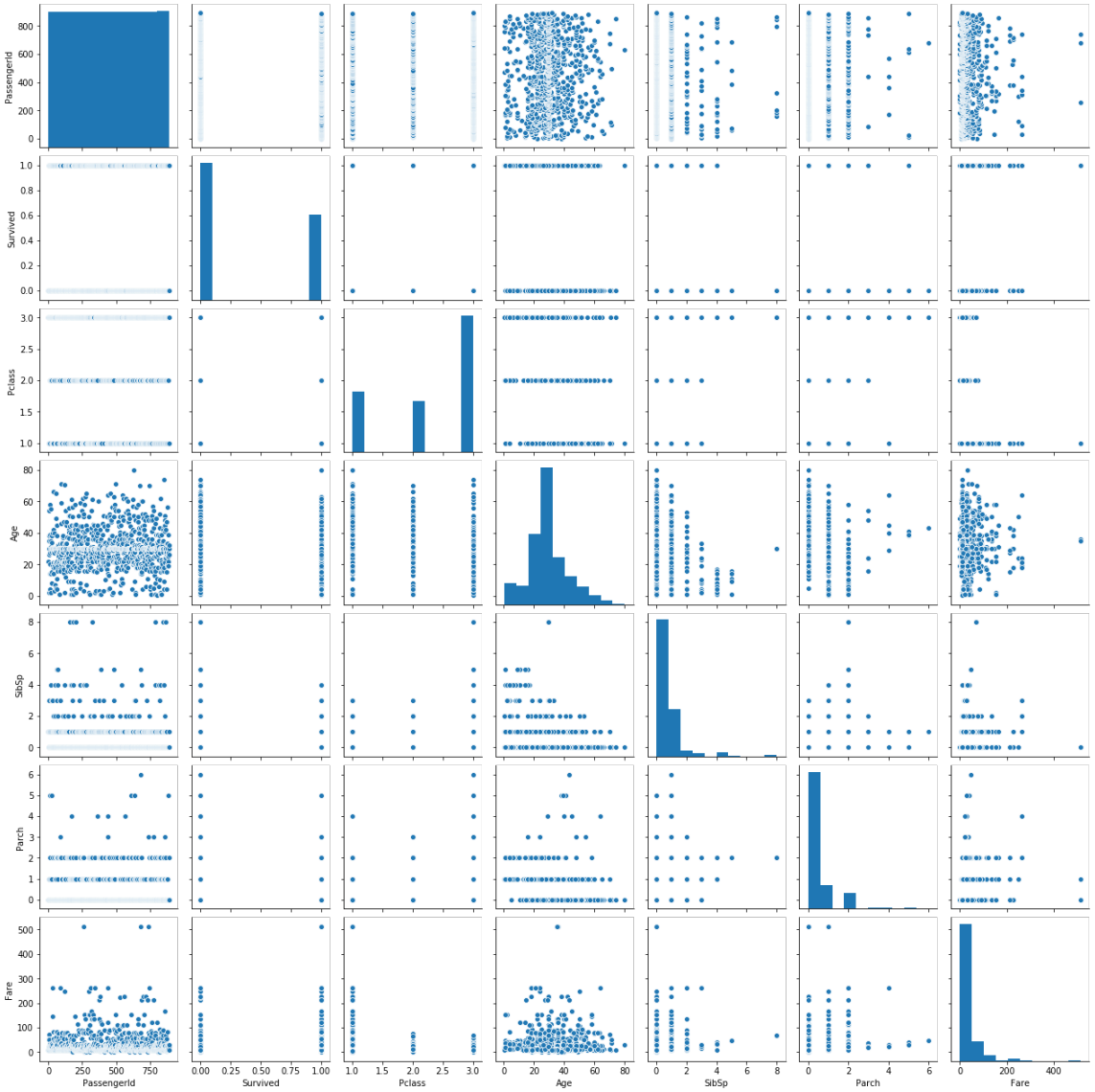
Out[25]:
```
<seaborn.axisgrid.FacetGrid at 0x7f884a87cd10>
```



PairPlot

In [26]:
```
sns.pairplot(data)
```

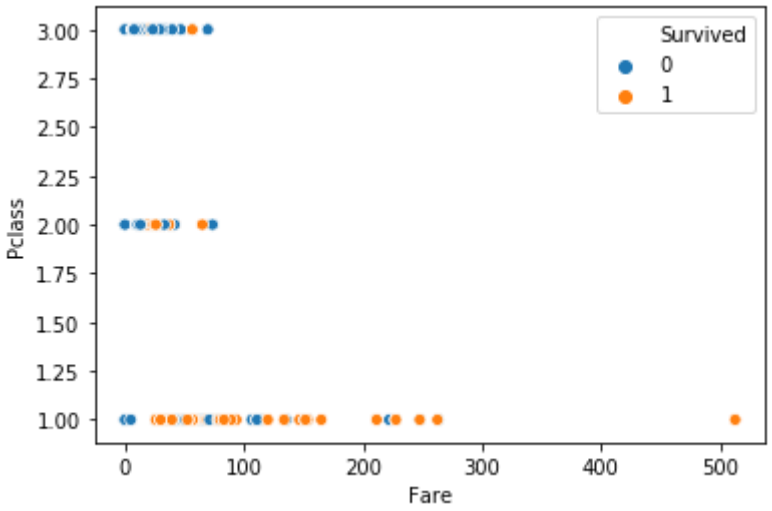
Out[26]:
```
<seaborn.axisgrid.PairGrid at 0x7f884a7aeed0>
```



ScatterPlot

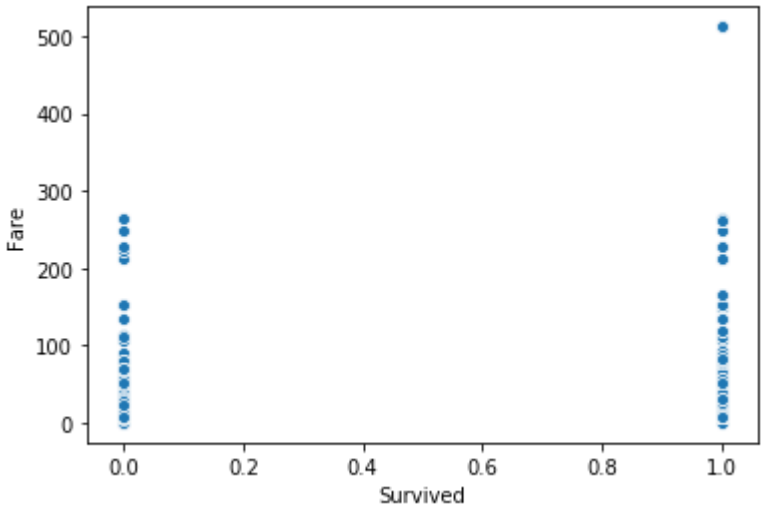
In [27]:
```
sns.scatterplot(x = 'Fare', y = 'Pclass', hue = 'Survived', data = da
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8848df1890>
```



In [28]:
```
sns.scatterplot(x = 'Survived', y = 'Fare', data = data)
```

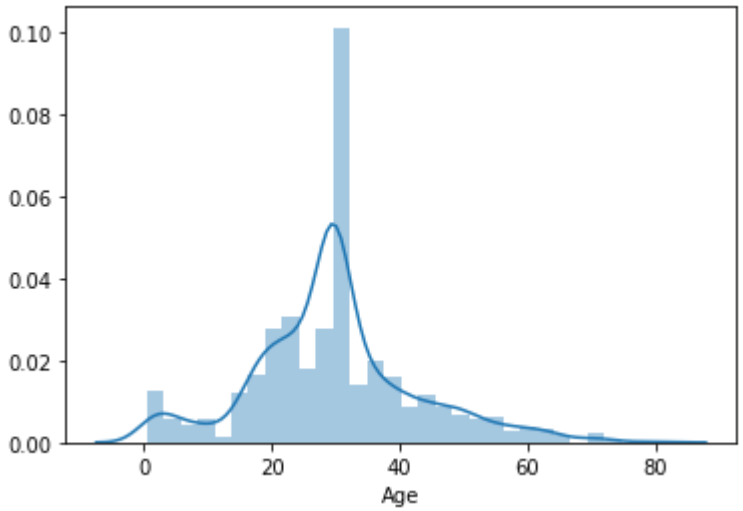
Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8848c7f850>
```



DistPoint

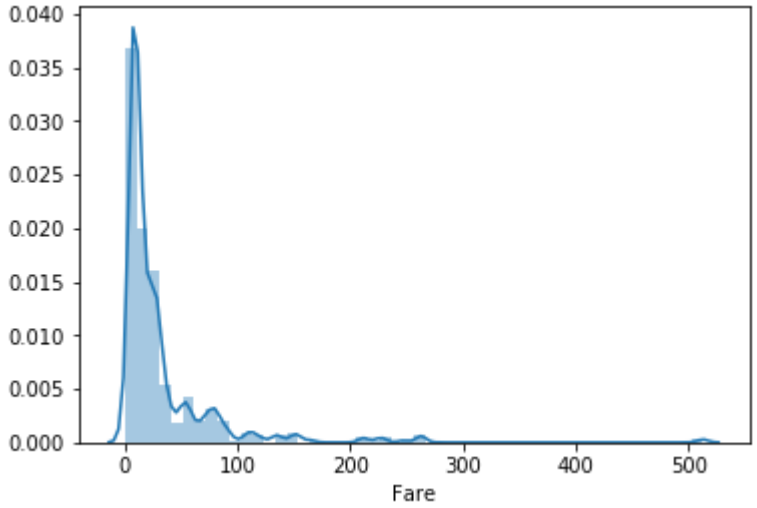
In [29]:
```
sns.distplot(data['Age'])
```

Out[29]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8846dd4790>
```



In [30]:
```
sns.distplot(data['Fare'])
```

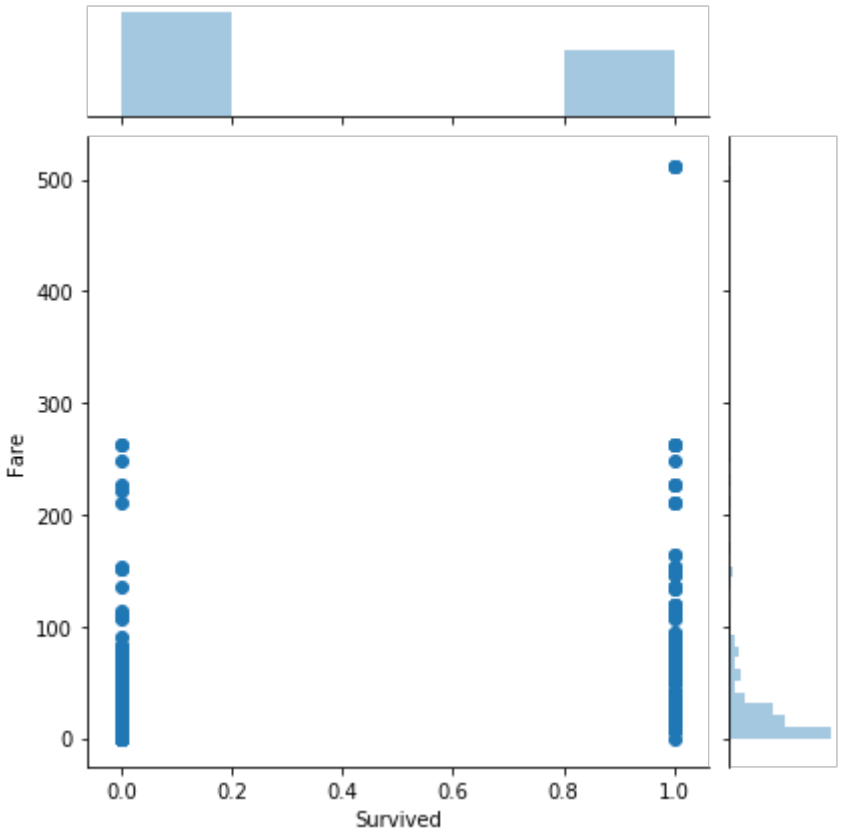
Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8846d251d0>
```



JoinPlot

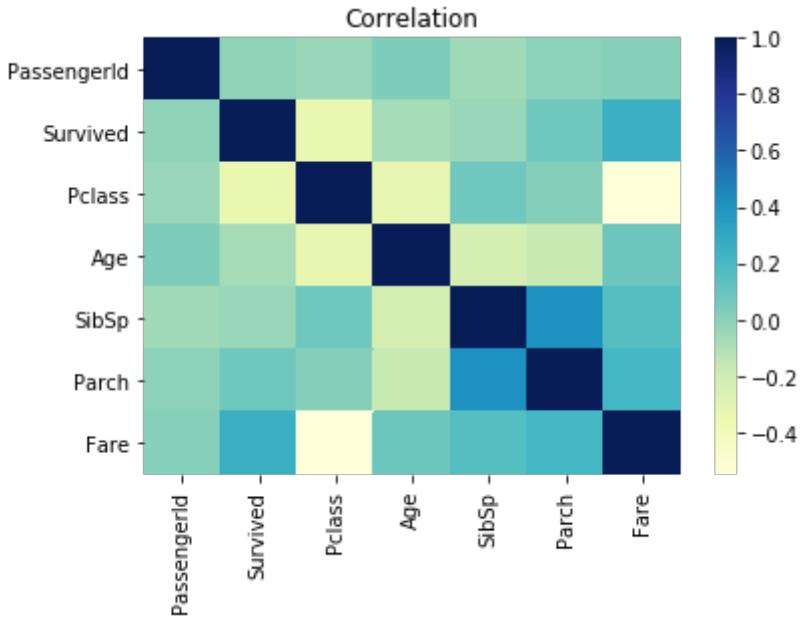
In [31]:
```
sns.jointplot(x = "Survived", y = "Fare", kind = "scatter", data = da
```

Out[31]:
```
<seaborn.axisgrid.JointGrid at 0x7f8846c25ed0>
```



In [32]:
```
tc = data.corr()
sns.heatmap(tc, cmap="YlGnBu")
plt.title('Correlation')
```

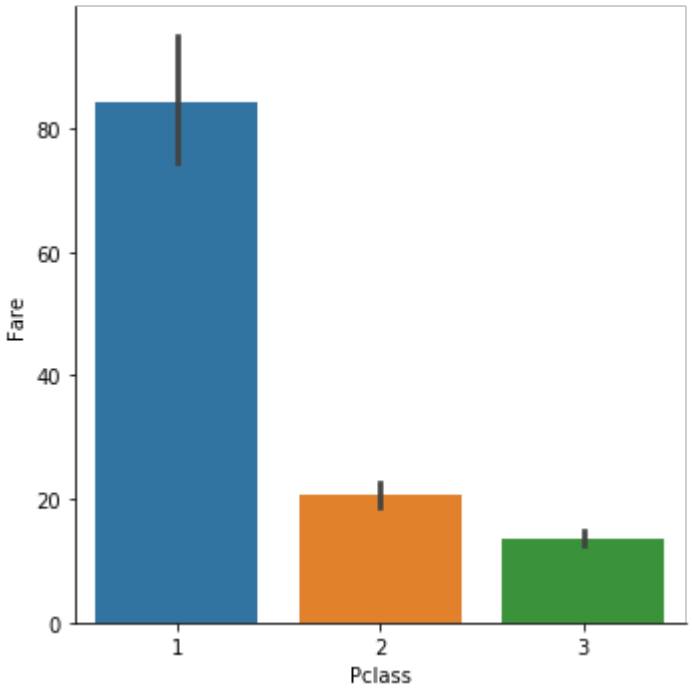
Out[32]:
```
Text(0.5, 1, 'Correlation')
```



Price of Ticket for each passenger is distributed

In [33]:
```
sns.catplot(x='Pclass', y='Fare', data=data, kind='bar')
```

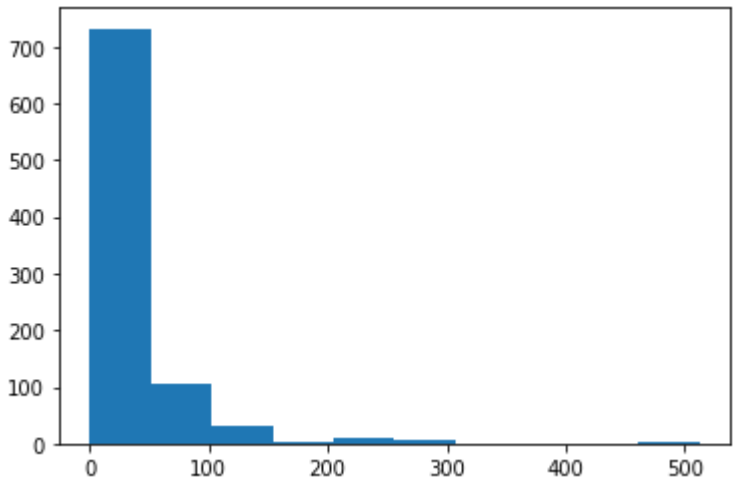
Out[33]:
```
<seaborn.axisgrid.FacetGrid at 0x7f8846b656d0>
```



In [34]:
```
import matplotlib.pyplot as plt
```

In [35]:
```
plt.hist(data['Fare'])
```

Out[35]:
```
(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 ]),
 <a list of 10 Patch objects>)
```



In [ ]:
```
```

Assignment No.:9, Name: Tejaswini Anil Rathod, Roll No.:28, Div:B

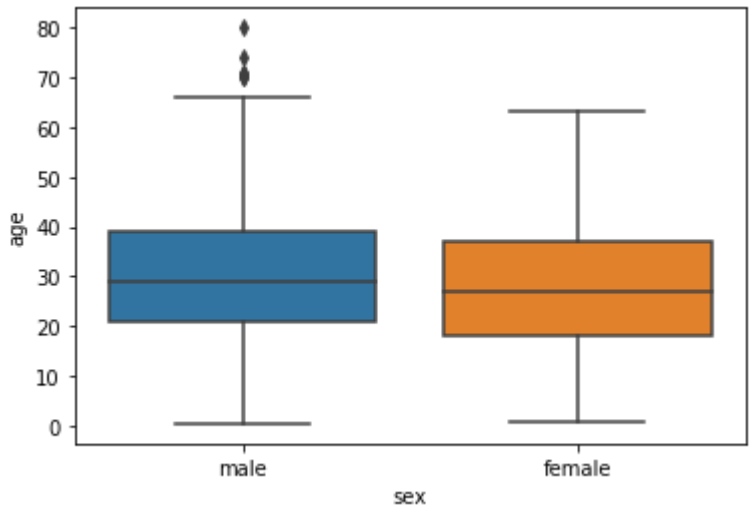
In [2]:
```
import seaborn as sns
dataset = sns.load_dataset('titanic')
dataset.head()
```

Out[2]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_mal
0	0	3	male	22.0	1	0	7.2500	S	Third	man	Tru
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fals
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fals
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fals
4	0	3	male	35.0	0	0	8.0500	S	Third	man	Tru

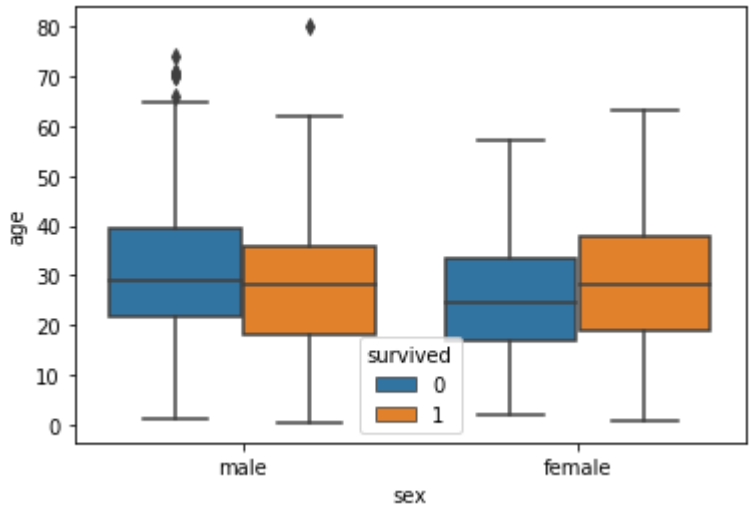
In [4]:
```
sns.boxplot(x='sex',y='age',data=dataset)
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7a2d5145d0>
```



In [5]:
```
sns.boxplot(x='sex',y='age',data=dataset,hue='survived')
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7a2d463590>
```



In [ ]:
```

```

Assignment No.:10, Name: Tejaswini Anil Rathod, Roll No.: 28, Div:B

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('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	NaN	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.isnull().sum()
```

Out[2]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    1
PetalWidthCm     0
Species          0
dtype: int64
```

In [3]:
```
df['PetalLengthCm']=df['PetalLengthCm'].fillna(np.mean(df['PetalLengt
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [5]:
```
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 [6]:
```
np.unique(df["Species"])
```

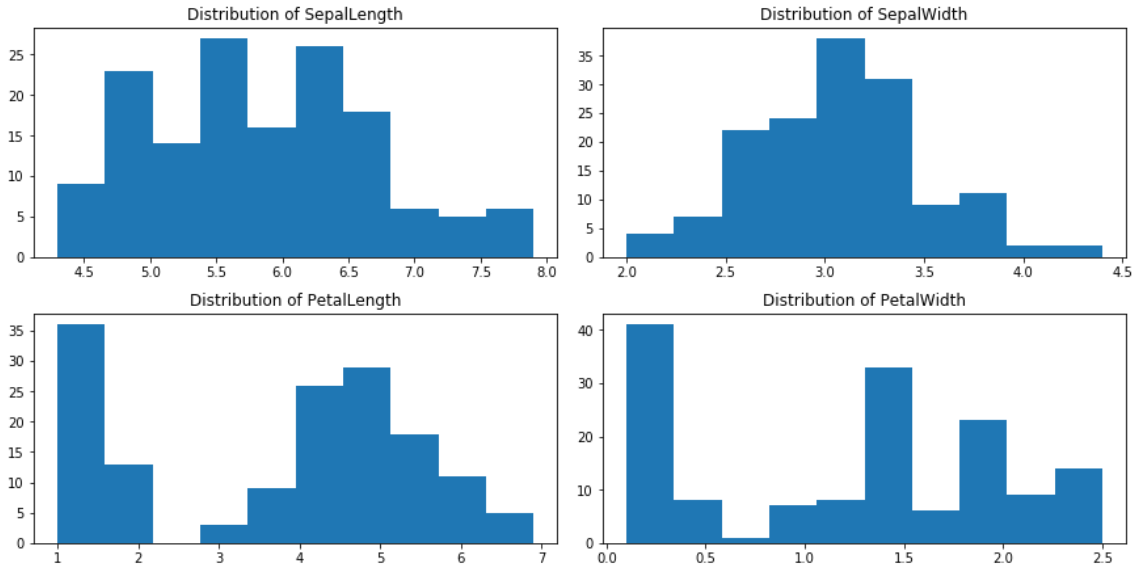
Out[6]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=o
bject)
```

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

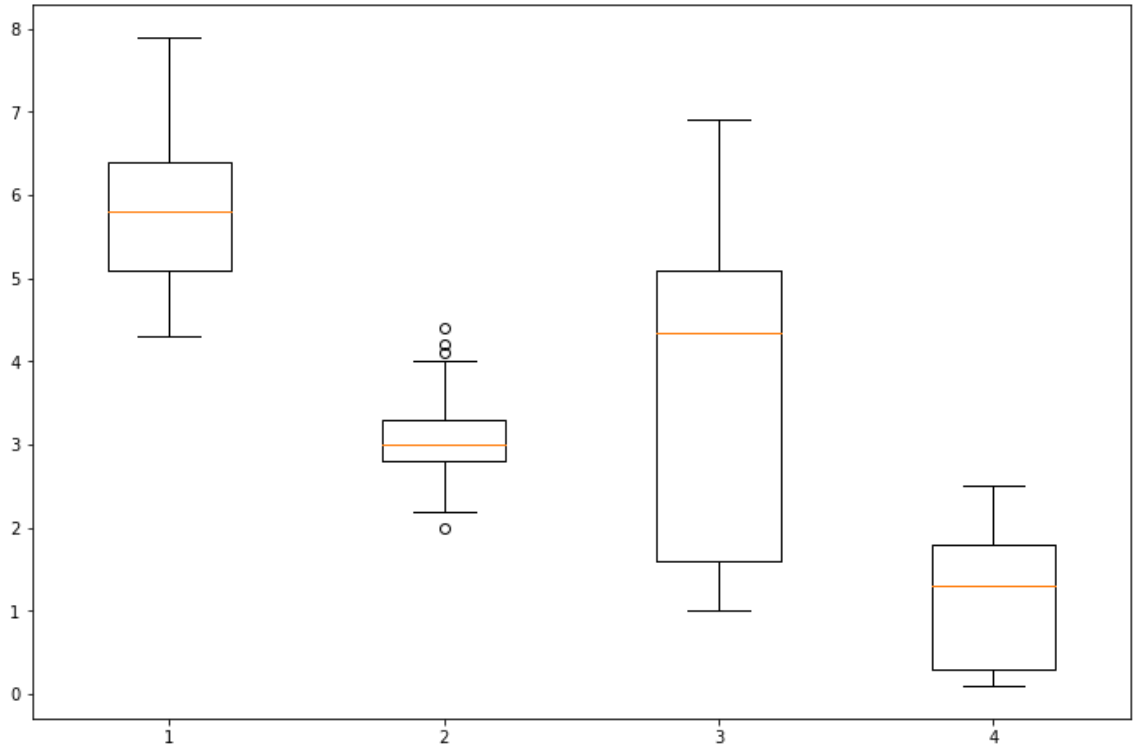
Out[7]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.775168	1.198667
std	43.445368	0.828066	0.433594	1.752808	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 [8]:
```
fig, axes = plt.subplots(2, 2, figsize=(12, 6), constrained_layout =
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 [ ]: