Group A : 01

Name	Abhishek Santosh Bankar
Roll No	305A012
Div	TE-1

In [77]:
```
import pandas as pd
import numpy as np
```

In [78]:
```
iris = pd.read_csv('Iris.csv')
```

In [79]:
```
iris.head()
```

Out[79]:

	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 [80]:
```
# col_names = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Species']
```

In [81]:
```
# iris = pd.read_csv('Iris.csv', names = col_names)
```

In [82]:
```
iris.head()
```

Out[82]:

	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 [83]:
```
iris.tail()
```

Out[83]:

	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 [84]:
```
iris.index
```

Out[84]:
```
RangeIndex(start=0, stop=150, step=1)
```

In [85]:
```
iris.columns
```

Out[85]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [86]:
```
iris.shape
```

Out[86]:
```
(150, 6)
```

In [87]:
```
iris.dtypes
```

Out[87]:
```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [88]:
```
iris.columns.values
```

Out[88]:
```
array(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
       'PetalWidthCm', 'Species'], dtype=object)
```

In [89]:
```
iris.describe(include='all')
```

Out[89]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	150.000000	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	NaN	Iris-setosa
freq	NaN	NaN	NaN	NaN	NaN	50
mean	75.500000	5.843333	3.054000	3.758667	1.198667	NaN
std	43.445368	0.828066	0.433594	1.764420	0.763161	NaN
min	1.000000	4.300000	2.000000	1.000000	0.100000	NaN
25%	38.250000	5.100000	2.800000	1.600000	0.300000	NaN
50%	75.500000	5.800000	3.000000	4.350000	1.300000	NaN
75%	112.750000	6.400000	3.300000	5.100000	1.800000	NaN
max	150.000000	7.900000	4.400000	6.900000	2.500000	NaN

In [91]:
```
iris['SepalLengthCm']
```

Out[91]:
```
0      5.1
1      4.9
2      4.7
3      4.6
4      5.0
      ...
145    6.7
146    6.3
147    6.5
148    6.2
149    5.9
Name: SepalLengthCm, Length: 150, dtype: float64
```

In [92]:
```
iris.sort_index(axis=1, ascending = False)
```

Out[92]:

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

150 rows × 6 columns

In [93]:
```
iris.sort_values(by = 'SepalLengthCm', ascending = False)
```

Out[93]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
131	132	7.9	3.8	6.4	2.0	Iris-virginica
135	136	7.7	3.0	6.1	2.3	Iris-virginica
122	123	7.7	2.8	6.7	2.0	Iris-virginica
117	118	7.7	3.8	6.7	2.2	Iris-virginica
118	119	7.7	2.6	6.9	2.3	Iris-virginica
...	...	...	...	...	...	...
41	42	4.5	2.3	1.3	0.3	Iris-setosa
42	43	4.4	3.2	1.3	0.2	Iris-setosa

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
38	39	4.4	3.0	1.3	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
13	14	4.3	3.0	1.1	0.1	Iris-setosa

150 rows × 6 columns

In [94]:
```
iris.iloc[5]
```

Out[94]:
```
Id                        6
SepalLengthCm           5.4
SepalWidthCm            3.9
PetalLengthCm           1.7
PetalWidthCm            0.4
Species         Iris-setosa
Name: 5, dtype: object
```

In [95]:
```
iris[0:3]
```

Out[95]:

	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

In [96]:
```
iris.loc[:, ['Species', 'PetalLengthCm']]
```

Out[96]:

	Species	PetalLengthCm
0	Iris-setosa	1.4
1	Iris-setosa	1.4
2	Iris-setosa	1.3
3	Iris-setosa	1.5
4	Iris-setosa	1.4
...	...	...
145	Iris-virginica	5.2
146	Iris-virginica	5.0
147	Iris-virginica	5.2
148	Iris-virginica	5.4
149	Iris-virginica	5.1

150 rows × 2 columns

In [97]:
```
iris.iloc[:4, :]
```

Out[97]:

	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

In [98]:
```
iris.iloc[:, :3]
```

Out[98]:

	Id	SepalLengthCm	SepalWidthCm
0	1	5.1	3.5
1	2	4.9	3.0
2	3	4.7	3.2
3	4	4.6	3.1
4	5	5.0	3.6
...	...	...	...
145	146	6.7	3.0
146	147	6.3	2.5
147	148	6.5	3.0
148	149	6.2	3.4
149	150	5.9	3.0

150 rows × 3 columns

In [99]:
```
iris.iloc[:5, :2]
```

Out[99]:

	Id	SepalLengthCm
0	1	5.1
1	2	4.9
2	3	4.7
3	4	4.6
4	5	5.0

In [100…
```
iris.isnull()
```

Out[100…

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species

	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 [101…

```
iris.isna()
```

Out[101…

	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 [102…

```
iris.isnull().any()
```

Out[102…

```
Id               False
SepalLengthCm    False
SepalWidthCm     False
PetalLengthCm    False
PetalWidthCm     False
```

```
Species          False
dtype: bool
```

In [103…

```
iris.isnull().sum()
```

Out[103…

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

In [104…

```
iris.isnull().sum(axis = 1)
```

Out[104…

```
0      0
1      0
2      0
3      0
4      0
      ..
145    0
146    0
147    0
148    0
149    0
Length: 150, dtype: int64
```

In [105…

```
iris.Species.isnull().sum()
```

Out[105…

```
0
```

In [107…

```
iris.groupby(['Species'])['SepalLengthCm'].apply(lambda x : x.isnull().sum())
```

Out[107…

```
Species
Iris-setosa        0
Iris-versicolor    0
Iris-virginica     0
Name: SepalLengthCm, dtype: int64
```

In [108…

```
iris.dtypes
```

Out[108…

```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [ ]:

Data normalization

In [67]:
```
from sklearn import preprocessing
```

In [111…
```
df = pd.DataFrame(iris)
```

In [112…
```
df.head()
```

Out[112…

	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 [122…
```
x = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']].values.astyp
```

In [123…
```
min_max_scaler = preprocessing.MinMaxScaler()
```

In [124…
```
x_scaled = min_max_scaler.fit_transform(x)
```

In [125…
```
df_normalized = pd.DataFrame(x_scaled)
```

In [127…
```
df_normalized.head()
```

Out[127…

	0	1	2	3
0	0.222222	0.625000	0.067797	0.041667
1	0.166667	0.416667	0.067797	0.041667
2	0.111111	0.500000	0.050847	0.041667
3	0.083333	0.458333	0.084746	0.041667
4	0.194444	0.666667	0.067797	0.041667

In [ ]:
```

```

Categorical Values - Label Encoding!

In [129…
```
df['Species'].unique()
```

Out[129…
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [130…
```
label_encoder = preprocessing.LabelEncoder()
```

In [131…
```
df['Species'] = label_encoder.fit_transform(df['Species'])
```

In [133…
```
df['Species'].unique()
```

Out[133…
```
array([0, 1, 2])
```

In [ ]:

Sklearn for One-hot-encoding

In [134…
```
df['Species'].unique()
```

Out[134…
```
array([0, 1, 2])
```

In [135…
```
features_df = df.drop(columns = ['Species'])
```

In [139…
```
enc = preprocessing.OneHotEncoder()
```

In [148…
```
y = enc.fit_transform(df[['Species']]).toarray()
```

In [149…
```
enc_df = pd.DataFrame(y)
```

In [153…
```
enc_df.head()
```

Out[153…

	0	1	2
0	1.0	0.0	0.0
1	1.0	0.0	0.0
2	1.0	0.0	0.0
3	1.0	0.0	0.0
4	1.0	0.0	0.0

In [154…
```
df_encode = features_df.join(enc_df)
```

In [155…
```
df_encode.head()
```

Out[155…

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	0	1	2
0	1	5.1	3.5	1.4	0.2	1.0	0.0	0.0
1	2	4.9	3.0	1.4	0.2	1.0	0.0	0.0
2	3	4.7	3.2	1.3	0.2	1.0	0.0	0.0
3	4	4.6	3.1	1.5	0.2	1.0	0.0	0.0
4	5	5.0	3.6	1.4	0.2	1.0	0.0	0.0

In [156…

```
df_encode.rename(columns = {0: 'Iris-setosa', 1: 'Iris-versicolor', 2: 'Iris-virginica'
```

In [157…

```
df_encode.head()
```

Out[157…

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Iris-setosa	Iris-versicolor	Iris-virginica
0	1	5.1	3.5	1.4	0.2	1.0	0.0	0.0
1	2	4.9	3.0	1.4	0.2	1.0	0.0	0.0
2	3	4.7	3.2	1.3	0.2	1.0	0.0	0.0
3	4	4.6	3.1	1.5	0.2	1.0	0.0	0.0
4	5	5.0	3.6	1.4	0.2	1.0	0.0	0.0

In [ ]:

Dummy Variable Encoding

In [159…

```
df['Species'].unique()
```

Out[159…

```
array([0, 1, 2])
```

In [160…

```
one_hot_df = pd.get_dummies(df, prefix='Species', columns=['Species'], drop_first=False
```

In [162…

```
one_hot_df.head()
```

Out[162…

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species_0	Species_1	Species_2
0	1	5.1	3.5	1.4	0.2	1	0	0
1	2	4.9	3.0	1.4	0.2	1	0	0
2	3	4.7	3.2	1.3	0.2	1	0	0
3	4	4.6	3.1	1.5	0.2	1	0	0
4	5	5.0	3.6	1.4	0.2	1	0	0

Group A : 02

Name	Abhishek Santosh Bankar
Roll No	305A012
Div	TE-1

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

```
df= pd.read_csv('studentData.csv')
```

```
df.head()
```

Email Address	Name	Email	Roll no	
nbare19@gmail.com	Sejal Zambare	sejal.zambare19@gmail.com	TECOC359	72
thorat19@gmail.com	Rushikesh Vilas Thorat	rushikesh.thorat19@gmail.com	TECOC347	72
takke19@gmail.com	Atharv Sontakke	atharv123sontakke@gmail.com	TECOC340	72
erekar19@gmail.com	Amisha Sunil Sherekar	amisha.sherekar19@gmail.com	TECOC328	72
dekar19@gmail.com	Saurabh Raju Sawardekar	saurabh.sawardekar19@gmail.com	TECOC326	72

```
df.isnull()
```

	Timestamp	Email Address	Name	Email	Roll no	PRN No.	Mobile No.	First year: Sem 1	First year: Sem 2	S
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	

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

```
Timestamp               0
Email Address           0
Name                    0
Email                   0
Roll no                 0
PRN No.                 0
Mobile No.              0
First year:   Sem 1     0
First year:   Sem 2     0
Second year:   Sem 1    0
Second year:   Sem 2    0
dtype: int64
```

```
df.describe()
```

	Mobile No.	First year: Sem 1	First year: Sem 2	Second year: Sem 1	Second year: Sem 2
count	6.400000e+01	64.000000	64.000000	64.000000	64.000000
mean	8.623097e+09	8.834219	9.095469	9.292031	9.377187
std	9.132070e+08	11.187839	11.171986	0.528523	0.495185
min	7.028870e+09	0.000000	0.000000	6.900000	7.200000
25%	7.766559e+09	7.237500	7.655000	9.050000	9.140000
50%	8.805720e+09	8.260000	8.400000	9.445000	9.450000
75%	9.335094e+09	8.802500	9.115000	9.645000	9.725000

```
plt.hist(df['Second year:   Sem 1'])
plt.show()
```

```
lower_bound=.69
upper_bound=.991
res=df['Second year:   Sem 1'].quantile([lower_bound,upper_bound])
res
```

```
0.690    9.59940
0.991    9.88165
Name: Second year:   Sem 1, dtype: float64
```

```
trueindex=(res.loc[lower_bound]<df['Second year:   Sem 1'].values) & (df['Second year:   S
trueindex
```

```
array([ True, False, False, False, False, False,  True, False, False,
       False, False,  True,  True, False, False, False,  True,  True,
       False, False,  True, False,  True,  True, False, False, False,
       False, False, False, False,  True, False, False, False,  True,
        True,  True, False,  True, False, False, False, False,  True,
       False,  True, False, False, False,  True, False, False, False,
       False,  True, False, False,  True, False, False, False, False,
       False])
```

```
falseindex=~trueindex
falseindex
```

```
array([False,  True,  True,  True,  True,  True, False,  True,  True,
        True,  True, False, False,  True,  True,  True, False, False,
        True,  True, False,  True, False, False,  True,  True,  True,
        True,  True,  True,  True, False,  True,  True,  True, False,
       False, False,  True, False,  True,  True,  True,  True, False,
        True, False,  True,  True,  True, False,  True,  True,  True,
        True, False,  True,  True, False,  True,  True,  True,  True,
        True])
```

```
df['Second year:   Sem 1'][trueindex]
```

```
0     9.80
6     9.68
11    9.77
12    9.77
16    9.68
17    9.86
20    9.64
22    9.82
23    9.86
31    9.86
35    9.64
36    9.75
37    9.66
39    9.86
44    9.70
46    9.67
50    9.61
```

```
55    9.73
58    9.63
Name: Second year:   Sem 1, dtype: float64
```

```
mid=np.median(df['Second year:   Sem 1'][trueindex])
mid
```

```
9.73
```

✓ 0s completed at 10:10 PM

Group A : 03

Name	Abhishek Santosh Bankar
Roll No	305A012
Div	TE-1

Assignment No. 3- Descriptive Statistics - Measures of Central Tendency and Variability

```
from google.colab import files
uploaded = files.upload()
```

Choose files Iris.csv

- **Iris.csv**(text/csv) - 5107 bytes, last modified: 26/04/2022 - 100% done

```
Saving Iris.csv to Iris.csv
```

```
import pandas as pd
import numpy as np
df = pd.read_csv('Iris.csv')
df
```

	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

```
df.shape
```

```
(150, 6)
```

describe() function

```
df.describe()
```

	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

```
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
```

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

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

Mean

```
np.mean(df)
```

```
/usr/local/lib/python3.7/dist-packages/numpy/core/fromnumeric.py:3438: FutureW
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
Id               75.500000
SepalLengthCm     5.843333
SepalWidthCm      3.054000
PetalLengthCm     3.758667
PetalWidthCm      1.198667
dtype: float64
```

Mode

```
from scipy import stats
stats.mode(df)
```

```
ModeResult(mode=array([[1, 5.0, 3.0, 1.5, 0.2, 'Iris-setosa']], dtype=object),
```

Median

```
np.median(df['SepalLengthCm'])
```

```
5.8
```

```
np.median(df['PetalLengthCm'])
```

```
4.35
```

```
np.median(df['PetalWidthCm'])
```

```
1.3
```

```
np.median(df['SepalWidthCm'])
```

```
3.0
```

Variance

```
np.var(df)
```

```
/usr/local/lib/python3.7/dist-packages/numpy/core/fromnumeric.py:3721: FutureW
  return var(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
Id               1874.916667
SepalLengthCm       0.681122
SepalWidthCm        0.186751
PetalLengthCm       3.092425
PetalWidthCm        0.578532
dtype: float64
```

Standard Deviation

```
np.std(df)
```

```
/usr/local/lib/python3.7/dist-packages/numpy/core/fromnumeric.py:3579: FutureW
  return std(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
Id               43.300308
SepalLengthCm     0.825301
SepalWidthCm      0.432147
PetalLengthCm     1.758529
PetalWidthCm      0.760613
dtype: float64
```

Group A : 04

Name	Abhishek Santosh Bankar
Roll No	305A012
Div	TE-1

In [ ]:
```
# DATA ANALYSIS I
```

In [ ]:
```
# Linear Regression Model on Boston Housing Dataset
```

In [24]:
```
# import dependencies
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.model_selection import train_test_split
```

In [25]:
```
# load the Boston Housing Dataset from sklearn.datasets and print it
from sklearn.datasets import load_boston
boston = load_boston()
print(boston)
```

```
{'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
        4.9800e+00],
       [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
        9.1400e+00],
       [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
        4.0300e+00],
       ...,
       [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        5.6400e+00],
       [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
        6.4800e+00],
       [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        7.8800e+00]]), 'target': array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1,
16.5, 18.9, 15. ,
       18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
       15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
       13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
       21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
       35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
       19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
       20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
       23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
       33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
       21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
       20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
       23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 13.4,
       15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3, 19.4,
       17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. , 22.7,
       25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6, 29.4,
       23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 50. ,
       32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3, 30.3,
       34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5, 24.4,
       20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 23. ,
       26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5, 24.3,
       31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. , 20.1,
       22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8, 29.6,
       42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 31. ,
       36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2, 32.4,
       32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2, 22. ,
```

```
       20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6, 27.1,
       20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4, 28.2,
       22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8, 23.1,
       21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3, 22.6,
       19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. , 18.7,
       32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9, 24.1,
       18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9, 20.8,
       16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. , 13.8,
       13.8, 15. , 13.9, 13.3, 13.1, 10.2, 10.4, 10.9, 11.3, 12.3,  8.8,
        7.2, 10.5,  7.4, 10.2, 11.5, 15.1, 23.2,  9.7, 13.8, 12.7, 13.1,
       12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. , 11.9,
       27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5, 10.4,
        8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9, 11. ,
        9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4, 12.8,
       10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. , 13.4,
       15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4, 17.7,
       19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6, 23.2,
       29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. , 21.8,
       20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8, 24.5,
       23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 11.9]), 'feature_name
s': array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7'), 'DESCR': ".. _boston_dataset:\n\nB
oston house prices dataset\n----------------------------\n\n**Data Set Characteristics:**
\n\n    :Number of Instances: 506 \n\n    :Number of Attributes: 13 numeric/categorical
predictive. Median Value (attribute 14) is usually the target.\n\n    :Attribute Informa
tion (in order):\n        - CRIM     per capita crime rate by town\n        - ZN       p
roportion of residential land zoned for lots over 25,000 sq.ft.\n        - INDUS    prop
ortion of non-retail business acres per town\n        - CHAS     Charles River dummy var
iable (= 1 if tract bounds river; 0 otherwise)\n        - NOX      nitric oxides concent
ration (parts per 10 million)\n        - RM       average number of rooms per dwelling\n
- AGE      proportion of owner-occupied units built prior to 1940\n        - DIS      we
ighted distances to five Boston employment centres\n        - RAD      index of accessib
ility to radial highways\n        - TAX      full-value property-tax rate per $10,000\n
- PTRATIO  pupil-teacher ratio by town\n        - B        1000(Bk - 0.63)^2 where Bk is
the proportion of black people by town\n        - LSTAT    % lower status of the populat
ion\n        - MEDV     Median value of owner-occupied homes in $1000's\n\n    :Missing
Attribute Values: None\n\n    :Creator: Harrison, D. and Rubinfeld, D.L.\n\nThis is a co
py of UCI ML housing dataset.\nhttps://archive.ics.uci.edu/ml/machine-learning-database
s/housing/\n\n\nThis dataset was taken from the StatLib library which is maintained at C
arnegie Mellon University.\n\nThe Boston house-price data of Harrison, D. and Rubinfeld,
D.L. 'Hedonic\nprices and the demand for clean air', J. Environ. Economics & Managemen
t,\nvol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics\n...',
Wiley, 1980.   N.B. Various transformations are used in the table on\npages 244-261 of t
he latter.\n\nThe Boston house-price data has been used in many machine learning papers
that address regression\nproblems.   \n     \n.. topic:: References\n\n   - Belsley, Kuh
& Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinear
ity', Wiley, 1980. 244-261.\n   - Quinlan,R. (1993). Combining Instance-Based and Model-
Based Learning. In Proceedings on the Tenth International Conference of Machine Learnin
g, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.\n", 'filename':
'C:\\Users\\Lenovo\\anaconda3\\lib\\site-packages\\sklearn\\datasets\\data\\boston_house
_prices.csv'}
```

In [ ]:

In [26]:

```
# Transform dataset into a dataframe
# data = the data we want or the independent variables also known as the x values
# feature_names = column names of the data
# target = the target variable or the price of the houses or the dependent variable als
```

```
df_x = pd.DataFrame(boston.data, columns = boston.feature_names)
df_y = pd.DataFrame(boston.target)
```

In [27]:

```
# get statistics from the dataset
df_x.describe()
```

Out[27]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS
count	506.000000	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	3.795043
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500

In [28]:

```
# Initialize Linear Regression Model
reg = linear_model.LinearRegression()
```

In [29]:

```
# split the data into 67% training and 33% testing data
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size = 0.33, rando
```

In [30]:

```
# train the model with the training data
reg.fit(x_train, y_train)
```

Out[30]:

```
LinearRegression()
```

In [31]:

```
# print the coefficients/weights for each feature/column of our model
print(reg.coef_)
```

```
[[-1.28749718e-01  3.78232228e-02  5.82109233e-02  3.23866812e+00
  -1.61698120e+01  3.90205116e+00 -1.28507825e-02 -1.42222430e+00
   2.34853915e-01 -8.21331947e-03 -9.28722459e-01  1.17695921e-02
  -5.47566338e-01]]
```

In [32]:

```
# print the predictions on the test data
y_pred = reg.predict(x_test)
print(y_pred)
```

```
[[28.53469469]
 [36.6187006 ]
 [15.63751079]
 [25.5014496 ]
```

```
[18.7096734 ]
[23.16471591]
[17.31011035]
[14.07736367]
[23.01064388]
[20.54223482]
[24.91632351]
[18.41098052]
[-6.52079687]
[21.83372604]
[19.14903064]
[26.0587322 ]
[20.30232625]
[ 5.74943567]
[40.33137811]
[17.45791446]
[27.47486665]
[30.2170757 ]
[10.80555625]
[23.87721728]
[17.99492211]
[16.02608791]
[23.268288  ]
[14.36825207]
[22.38116971]
[19.3092068 ]
[22.17284576]
[25.05925441]
[25.13780726]
[18.46730198]
[16.60405712]
[17.46564046]
[30.71367733]
[20.05106788]
[23.9897768 ]
[24.94322408]
[13.97945355]
[31.64706967]
[42.48057206]
[17.70042814]
[26.92507869]
[17.15897719]
[13.68918087]
[26.14924245]
[20.2782306 ]
[29.99003492]
[21.21260347]
[34.03649185]
[15.41837553]
[25.95781061]
[39.13897274]
[22.96118424]
[18.80310558]
[33.07865362]
[24.74384155]
[12.83640958]
[22.41963398]
[30.64804979]
[31.59567111]
[16.34088197]
```

```
[20.9504304 ]
[16.70145875]
[20.23215646]
[26.1437865 ]
[31.12160889]
[11.89762768]
[20.45432404]
[27.48356359]
[10.89034224]
[16.77707214]
[24.02593714]
[ 5.44691807]
[21.35152331]
[41.27267175]
[18.13447647]
[ 9.8012101 ]
[21.24024342]
[13.02644969]
[21.80198374]
[ 9.48201752]
[22.99183857]
[31.90465631]
[18.95594718]
[25.48515032]
[29.49687019]
[20.07282539]
[25.5616062 ]
[ 5.59584382]
[20.18410904]
[15.08773299]
[14.34562117]
[20.85155407]
[24.80149389]
[-0.19785401]
[13.57649004]
[15.64401679]
[22.03765773]
[24.70314482]
[10.86409112]
[19.60231067]
[23.73429161]
[12.08082177]
[18.40997903]
[25.4366158 ]
[20.76506636]
[24.68588237]
[ 7.4995836 ]
[18.93015665]
[21.70801764]
[27.14350579]
[31.93765208]
[15.19483586]
[34.01357428]
[12.85763091]
[21.06646184]
[28.58470042]
[15.77437534]
[24.77512495]
[ 3.64655689]
[23.91169589]
```

```
[25.82292925]
[23.03339677]
[25.35158335]
[33.05655447]
[20.65930467]
[38.18917361]
[14.04714297]
[25.26034469]
[17.6138723 ]
[20.60883766]
[ 9.8525544 ]
[21.06756951]
[22.20145587]
[32.2920276 ]
[31.57638342]
[15.29265938]
[16.7100235 ]
[29.10550932]
[25.17762329]
[16.88159225]
[ 6.32621877]
[26.70210263]
[23.3525851 ]
[17.24168182]
[13.22815696]
[39.49907507]
[16.53528575]
[18.14635902]
[25.06620426]
[23.70640231]
[22.20167772]
[21.22272327]
[16.89825921]
[23.15518273]
[28.69699805]
[ 6.65526482]
[23.98399958]
[17.21004545]
[21.0574427 ]
[25.01734597]
[27.65461859]
[20.70205823]
[40.38214871]]
```

In [33]:

```
# print the actual value
print(y_test)
```

```
        0
173  23.6
274  32.4
491  13.6
72   22.8
452  16.1
..    ...
110  21.7
321  23.1
265  22.8
29   21.0
262  48.8
```

```
[167 rows x 1 columns]
```

In [34]:

```
# Check Model performance accuracy using Mean Squared Error (MSE)
# using numpy
print( np.mean( (y_pred - y_test)**2 ))
```

```
0    20.724023
dtype: float64
```

In [35]:

```
# Check Model performance accuracy using Mean Squared Error (MSE)
# using sklearn.metrics
from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test, y_pred))
```

```
20.72402343733974
```

In [ ]:

In [38]:

```
from sklearn.metrics import r2_score
y_train_prediction = reg.predict(x_train)
rmse = (np.sqrt(mean_squared_error(y_train, y_train_prediction)))
r2 = r2_score(y_train, y_train_prediction)
```

In [39]:

```
print("Model performance for training set: ")
print(rmse)
print(r2)
```

```
Model performance for training set:
4.794269062151269
0.7392344370995025
```

In [40]:

```
y_test_prediction = reg.predict(x_test)
rmse1 = (np.sqrt(mean_squared_error(y_test, y_test_prediction)))
r21 = r2_score(y_test, y_test_prediction)
```

In [41]:

```
print("Model performance for training set: ")
print(rmse1)
print(r21)
```

```
Model performance for training set:
4.55236459846306
0.7261570836552478
```

Group A : 05

Name	Abhishek Santosh Bankar
Roll No	305A012
Div	TE-1

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn import metrics
```

```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
```

```
LogisticRegression(random_state=0)
```

```
y_pred = classifier.predict(X_test)
```

```
cm = confusion_matrix(y_test, y_pred)
```

```
confusion_matrix=cm
FP = confusion_matrix.sum(axis=0) - np.diag(confusion_matrix)
FN = confusion_matrix.sum(axis=1) - np.diag(confusion_matrix)
TP = np.diag(confusion_matrix)
TN = confusion_matrix.sum() - (FP + FN + TP)
```

```
ACC = (TP+TN)/(TP+FP+FN+TN)
```

```
ERR = 1- ACC
```

```
PRC = TP/(TP+FP)
```

```
RCL = TP/(TP+FN)
```

```
print('True Positives (TP): ', TP[0])
print('False Positives (FP): ', FP[0])
print('True Negatives (TN): ', TN[0])
```

```
print('True Negatives (TN): ', TN[0])
print('False Negatives (FN): ', FN[0])
print('Accuracy: ', ACC[0])
print('Error Rate: ', ERR[0])
print('Precision: ', PRC[0])
print('Recall: ', RCL[0])
```

```
True Positives (TP):  57
False Positives (FP):  5
True Negatives (TN):  17
False Negatives (FN):  1
Accuracy:  0.925
Error Rate:  0.07499999999999996
Precision:  0.9193548387096774
Recall:  0.9827586206896551
```

✓ 0s completed at 8:42 PM

Group A : 06

Name	Abhishek Santosh Bankar
Roll No	305A012
Div	TE-1

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn import datasets
```

```
dataset = datasets.load_iris()
X = dataset.data[:, :2]
y = dataset.target
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
classifier = BernoulliNB()
classifier.fit(X_train, y_train)
```

```
BernoulliNB()
```

```
y_pred = classifier.predict(X_test)
```

```
cm = confusion_matrix(y_test, y_pred)
```

```
confusion_matrix=cm
FP = confusion_matrix.sum(axis=0) - np.diag(confusion_matrix)
FN = confusion_matrix.sum(axis=1) - np.diag(confusion_matrix)
TP = np.diag(confusion_matrix)
TN = confusion_matrix.sum() - (FP + FN + TP)
```

```
ACC = (TP+TN)/(TP+FP+FN+TN)
```

```
ERR = 1- ACC
```

```
PRC = TP/(TP+FP)
```

```
RCL = TP/(TP+FN)
```

```
print('True Positives (TP): ', TP[0])
print('False Positives (FP): ', FP[0])
```

```
print('False Positives (FP): ', FP[0])
print('True Negatives (TN): ', TN[0])
print('False Negatives (FN): ', FN[0])
print('Accuracy: ', ACC[0])
print('Error Rate: ', ERR[0])
print('Precision: ', PRC[0])
print('Recall: ', RCL[0])
```

```
True Positives (TP):  9
False Positives (FP):  0
True Negatives (TN):  19
False Negatives (FN):  2
Accuracy:  0.9333333333333333
Error Rate:  0.06666666666666665
Precision:  1.0
Recall:  0.8181818181818182
```

✓ 0s completed at 8:49 PM

Group A : 07

Name	Abhishek Santosh Bankar
Roll No	305A012
Div	TE-1

```
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
from nltk.tokenize import word_tokenize
import docx
import pandas as pd
import math
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
word_Doc=docx.Document('sample_data.docx')
```

```
extractedText = word_Doc.paragraphs[0].text
```

```
tokens = word_tokenize("The quick brown fox jumps over the lazy dog")
nltk.download('stopwords')
print(tokens)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
```

```
tag = nltk.pos_tag(tokens)
print(tag)
```

```
[('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox', 'NN'), ('jumps', 'VBZ'), (
```

```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
tokens = [w for w in tokens if not w in stop_words]
print(tokens)
```

```
['The', 'quick', 'brown', 'fox', 'jumps', 'lazy', 'dog']
```

```
from nltk.stem.porter import PorterStemmer
porter = PorterStemmer()
stems = []
for t in tokens:
    stems.append(porter.stem(t))
print(stems)
```

```
['the', 'quick', 'brown', 'fox', 'jump', 'lazi', 'dog']
```

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

for w in tokens:
  print(lemmatizer.lemmatize(w))
```

```
The
quick
brown
fox
jump
lazy
dog
```

```
first_sentence = word_Doc.paragraphs[1].text
second_sentence = word_Doc.paragraphs[2].text
```

```
first_sentence = first_sentence.split(" ")
second_sentence = second_sentence.split(" ")#join them to remove common duplicate words
total= set(first_sentence).union(set(second_sentence))
print(total)
```

```
{'job', 'machine', 'key', 'best', 'Data', 'the', 'of', '21st', 'is', 'science', 'for
```

```
wordDictA = dict.fromkeys(total, 0)
wordDictB = dict.fromkeys(total, 0)
for word in first_sentence:
    wordDictA[word]+=1

for word in second_sentence:
    wordDictB[word]+=1

print(pd.DataFrame([wordDictA, wordDictB]))
```

```
   job  machine  key  best  Data  the  of  21st  is  science  for  Science  \
0    1        0    0     1     1    2   1     1   1        0    0        1
1    0        1    1     0     0    1   0     0   1        1    1        0

   century  learning  data
0        1         0     0
1        0         1     1
```

```
def computeTF(wordDict, doc):
    tfDict = {}
    corpusCount = len(doc)
    for word, count in wordDict.items():
        tfDict[word] = count/float(corpusCount)
    return(tfDict)

#running our sentences through the tf function:
```

```
tfFirst = computeTF(wordDictA, first_sentence)
tfSecond = computeTF(wordDictB, second_sentence)

#Converting to dataframe for visualization
tf = pd.DataFrame([tfFirst, tfSecond])

print(tf)
```

```
   job  machine    key  best  Data    the   of  21st     is  science    for  \
0  0.1    0.000  0.000   0.1   0.1  0.200  0.1   0.1  0.100    0.000  0.000
1  0.0    0.125  0.125   0.0   0.0  0.125  0.0   0.0  0.125    0.125  0.125

   Science  century  learning   data
0      0.1      0.1     0.000  0.000
1      0.0      0.0     0.125  0.125
```

```
def computeIDF(docList):
    idfDict = {}
    N = len(docList)

    idfDict = dict.fromkeys(docList[0].keys(), 0)
    for word, val in idfDict.items():
        idfDict[word] = math.log10(N / (float(val) + 1))

    return(idfDict)
#inputing our sentences in the log file
idfs = computeIDF([wordDictA, wordDictB])
print(idfs)
```

```
56639812, 'Data': 0.3010299956639812, 'the': 0.3010299956639812, 'of': 0.301029995663
```

0s completed at 9:41 PM

Group A : 08

Name	Abhishek Santosh Bankar
Roll No	305A012
Div	TE-1

```
import pandas as pd
import numpy as np
import seaborn as sns
```

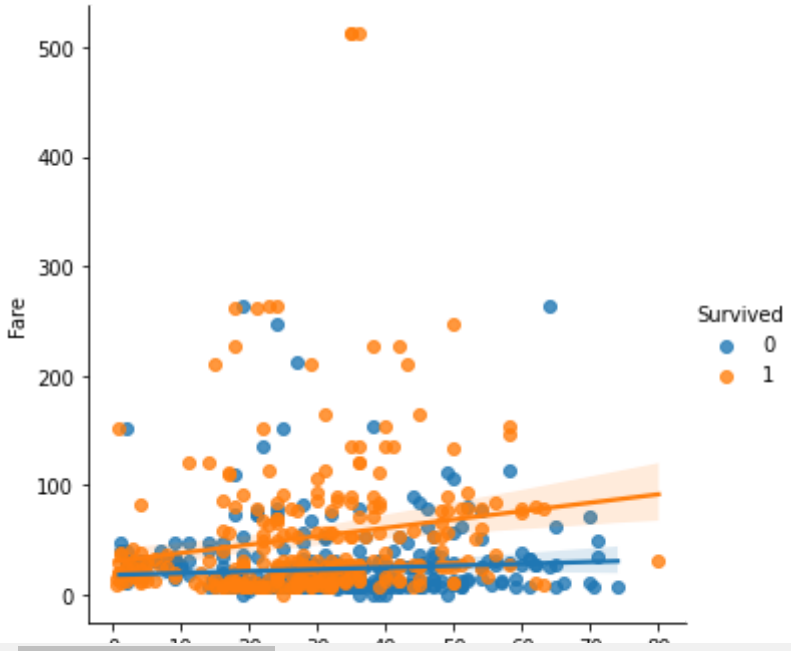
```
df=pd.read_csv('https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.
```

```
df.shape
```

```
(891, 12)
```

```
sns.lmplot('Age','Fare',df,hue='Survived')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
  FutureWarning
<seaborn.axisgrid.FacetGrid at 0x7f8a2ae7ea90>
```

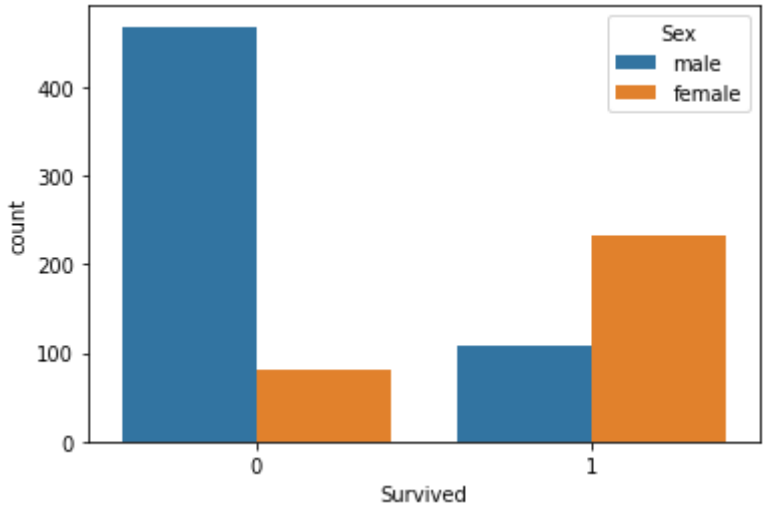



```
sns.scatterplot(data=df,x='Age',y='Fare',hue='Sex')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2ae11050>
```

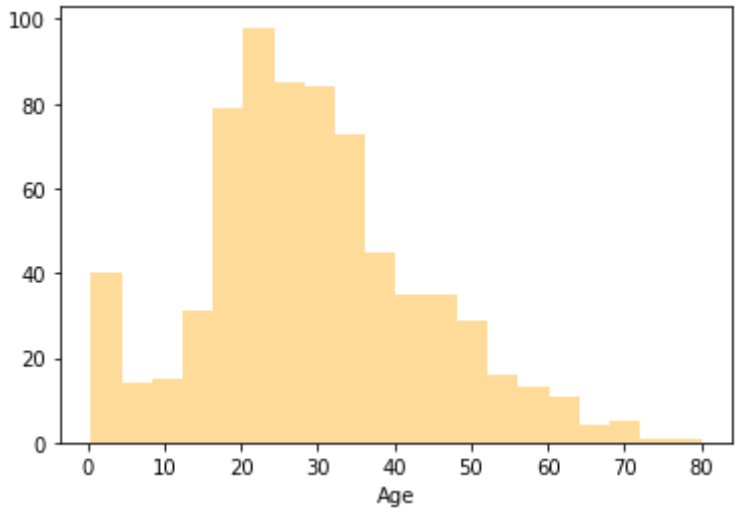
```
sns.countplot(data=df,x='Survived',hue='Sex')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2af0ea10>
```



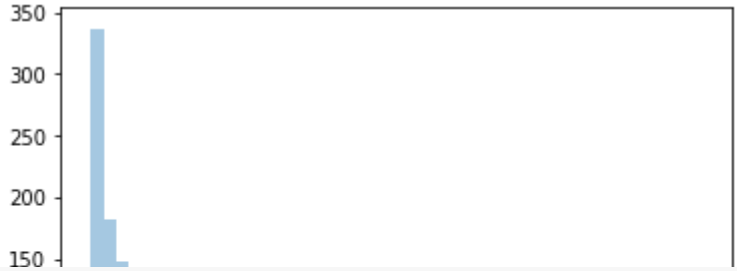

```
sns.distplot(df['Age'],kde=False,color='orange')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2acf4b90>
```



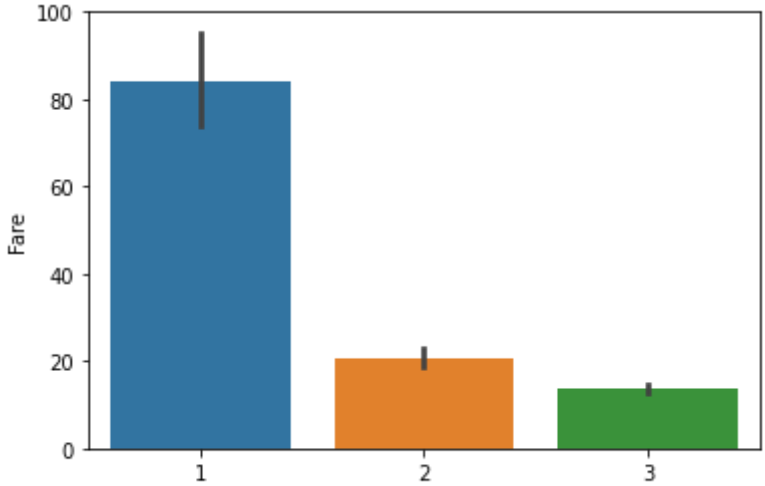

```
#dist plot is used
sns.distplot(df['Fare'],kde=False)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2ac9c910>
```



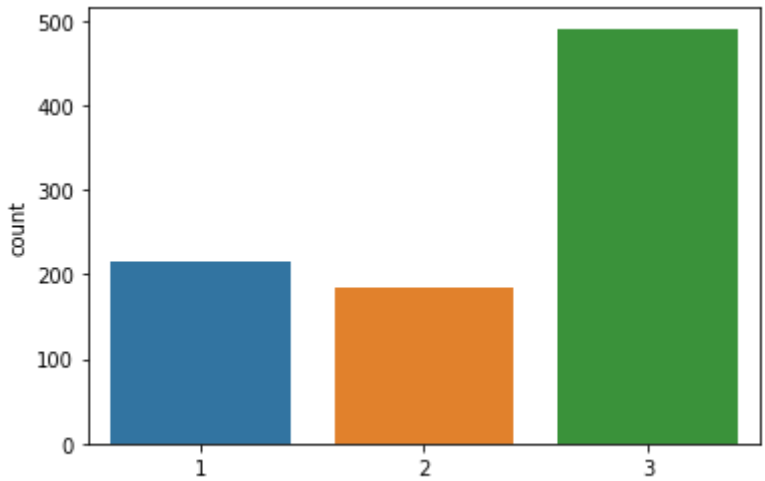

```
#which class has highest fare
sns.barplot(data=df,y='Fare',x='Pclass')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2ac1dc50>
```



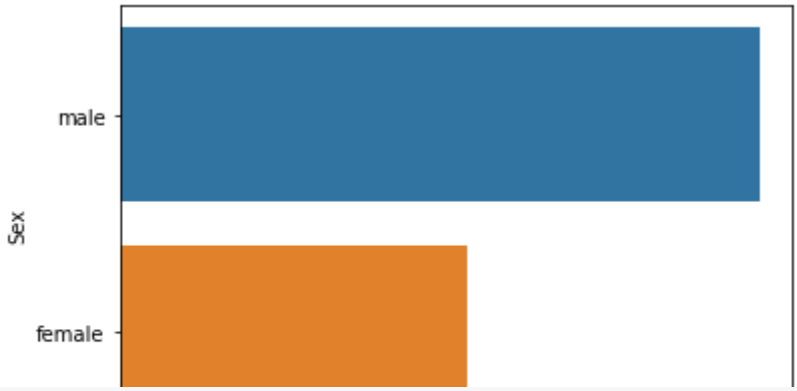

```
sns.countplot(data=df,x='Pclass')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2ab03550>
```




```
sns.countplot(data=df, y='Sex')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2aad6dd0>
```

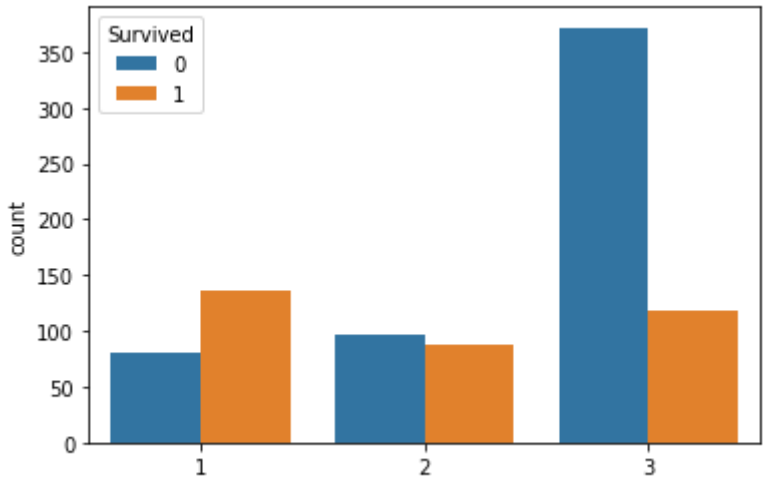



```
#To find
df.Pclass.value_counts()
```

```
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

```
#from which classs people does't survived
sns.countplot(x='Pclass',data=df,hue='Survived')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2aaa52d0>
```

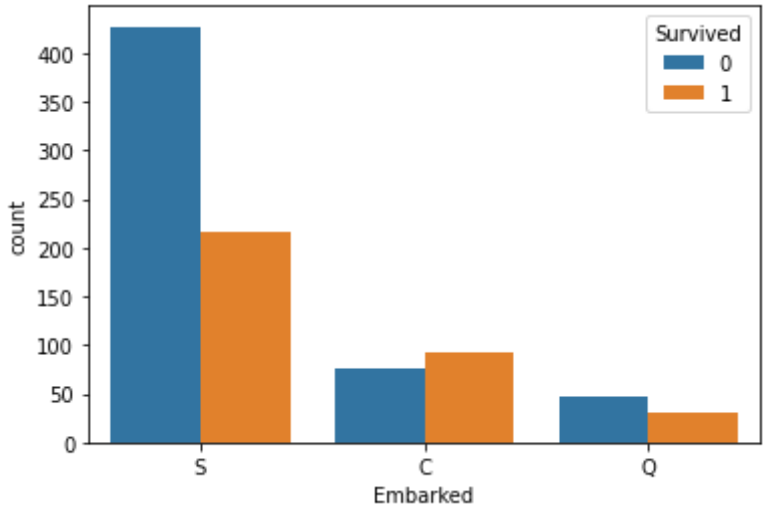



```
#From which  station most of passangers boarded
sns.countplot(data=df,x='Embarked')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2a9b8f90>
```

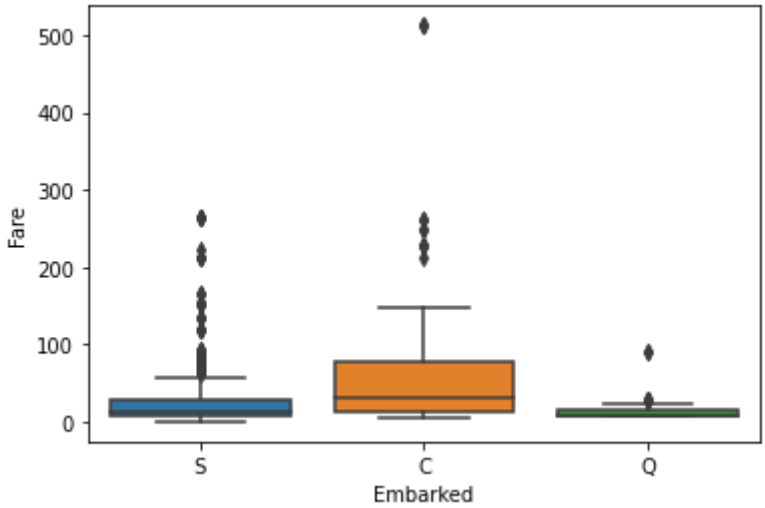
```
#from which station most of people not survived
sns.countplot(data=df,x='Embarked',hue='Survived')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2a9a0890>
```




```
#for which station fare was more
sns.boxplot(data=df,x="Embarked",y='Fare')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2a91d110>
```

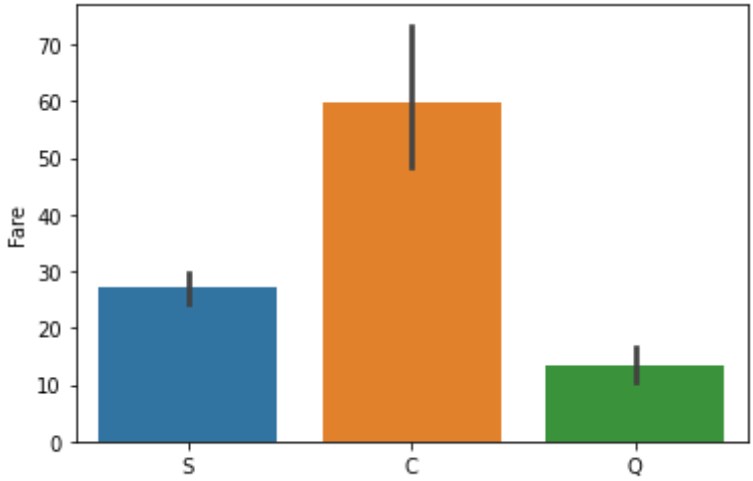



```
sns.scatterplot(data=df,x='Embarked',y='Fare')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2a8a8510>
```

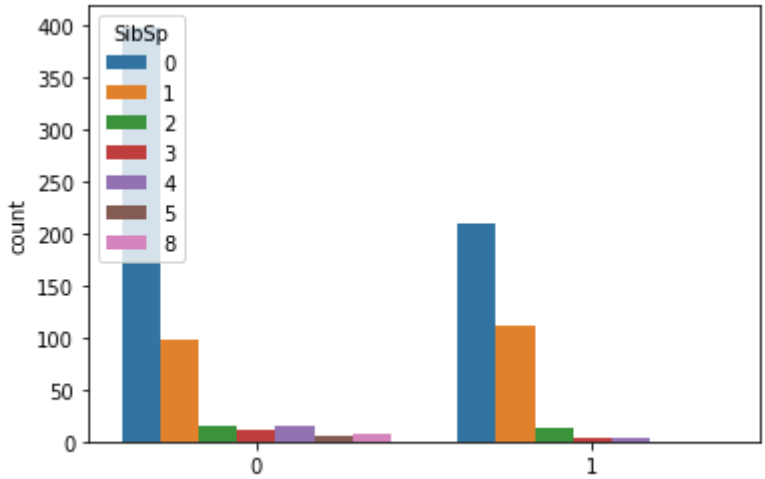
```
sns.barplot(data=df,x='Embarked',y='Fare')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2a81af50>
```



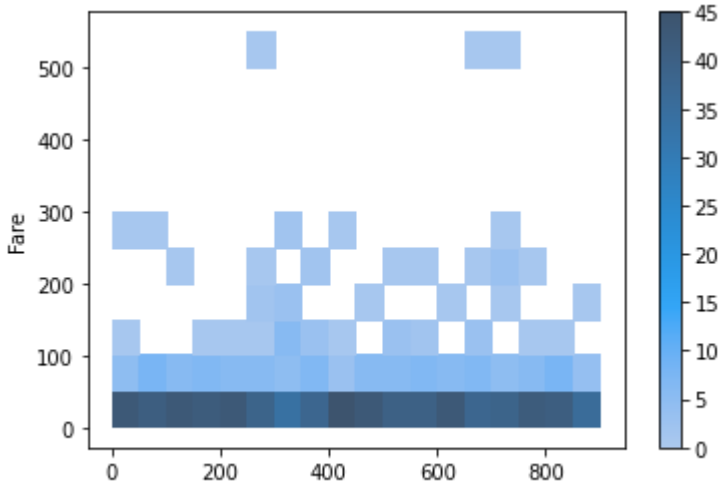

```
sns.countplot(data=df,x='Survived',hue='SibSp')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2a785f50>
```




```
#How the fare for each passenger is distributed by plotting a histogram
#dark color represents count of values is more
sns.histplot(data=df,x='PassengerId',y='Fare',binwidth=50,cbar=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a2a727b10>
```



✓ 0s completed at 9:55 PM

Group A : 09

Name	Abhishek Santosh Bankar
Roll No	305A012
Div	TE-1

```
import pandas as pd
import seaborn as sns
```

```
df=pd.read_csv('https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.
```

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	P(
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	ST 3

```
df.shape
```

```
(891, 12)
```

```
df.Age.unique()
```

```
array([22.  , 38.  , 26.  , 35.  ,   nan, 54.  ,  2.  , 27.  , 14.  ,
        4.  , 58.  , 20.  , 39.  , 55.  , 31.  , 34.  , 15.  , 28.  ,
        8.  , 19.  , 40.  , 66.  , 42.  , 21.  , 18.  ,  3.  ,  7.  ,
       49.  , 29.  , 65.  , 28.5 ,  5.  , 11.  , 45.  , 17.  , 32.  ,
       16.  , 25.  ,  0.83, 30.  , 33.  , 23.  , 24.  , 46.  , 59.  ,
       71.  , 37.  , 47.  , 14.5 , 70.5 , 32.5 , 12.  ,  9.  , 36.5 ,
       51.  , 55.5 , 40.5 , 44.  ,  1.  , 61.  , 56.  , 50.  , 36.  ,
       45.5 , 20.5 , 62.  , 41.  , 52.  , 63.  , 23.5 ,  0.92, 43.  ,
       60.  , 10.  , 64.  , 13.  , 48.  ,  0.75, 53.  , 57.  , 80.  ,
       70.  , 24.5 ,  6.  ,  0.67, 30.5 ,  0.42, 34.5 , 74.  ])
```

```
df.Survived.unique()
```

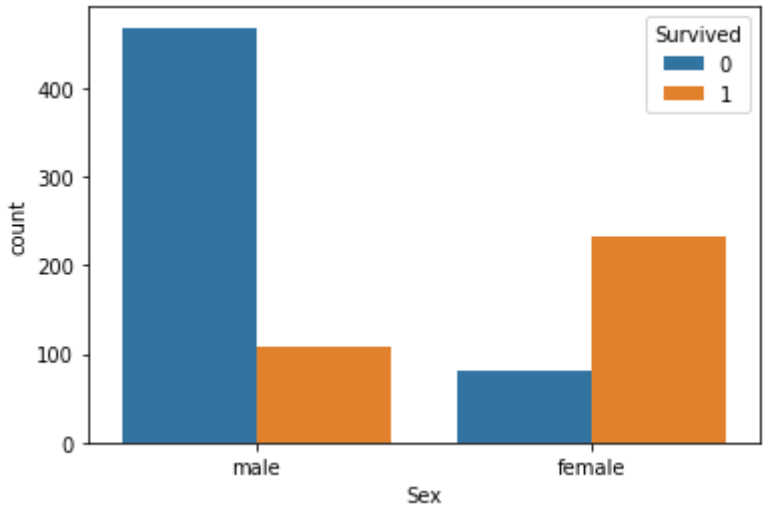
```
array([0, 1])
```

```
df.Sex.value_counts()
```

```
male      577
female    314
Name: Sex, dtype: int64
```

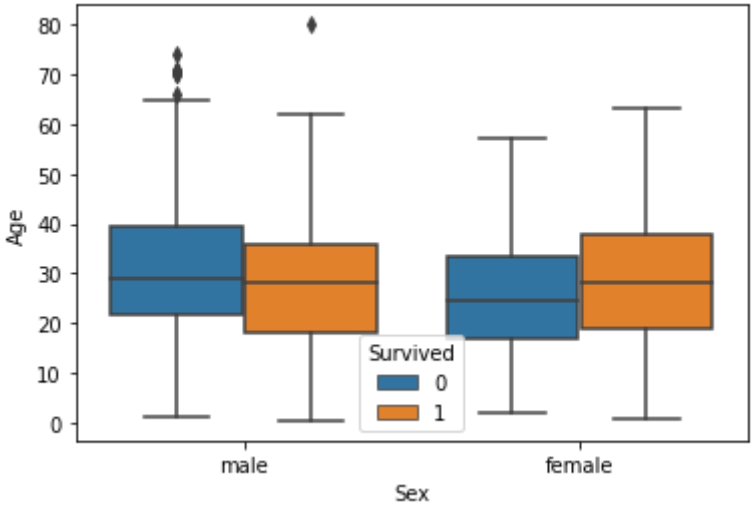
```
sns.countplot(data=df,x='Sex',hue='Survived')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f61058ed790>
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f61057ec110>
```



We have plotted the graph gender vs Age and setted hue as Survived. From above boxplot we can infer that

1. Maximum age in data is 40.
2. Female count of survival is greater the=an male count of survival.
3. Least age in the data is 18.
4. Survived female age group is between 18 to 40.
5. More older men died than younger men.

Group A : 10

Name	Abhishek Santosh Bankar
Roll No	305A012
Div	TE-1

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

```
df=pd.read_csv("https://gist.githubusercontent.com/netj/8836201/raw/6f9306ad21398ea43cba4f
```

```
df.head()
```

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

```
df.tail()
```

	sepal.length	sepal.width	petal.length	petal.width	variety
145	6.7	3.0	5.2	2.3	Virginica
146	6.3	2.5	5.0	1.9	Virginica
147	6.5	3.0	5.2	2.0	Virginica
148	6.2	3.4	5.4	2.3	Virginica
149	5.9	3.0	5.1	1.8	Virginica

```
df.shape
```

```
(150, 5)
```

```
df.columns
```

```
Index(['sepal.length', 'sepal.width', 'petal.length', 'petal.width',
       'variety'],
      dtype='object')
```

```
df["sepal.length"].unique()
```

```
array([5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.4, 4.8, 4.3, 5.8, 5.7, 5.2, 5.5,
       4.5, 5.3, 7. , 6.4, 6.9, 6.5, 6.3, 6.6, 5.9, 6. , 6.1, 5.6, 6.7,
       6.2, 6.8, 7.1, 7.6, 7.3, 7.2, 7.7, 7.4, 7.9])
```

```
df["sepal.width"].unique()
```

```
array([3.5, 3. , 3.2, 3.1, 3.6, 3.9, 3.4, 2.9, 3.7, 4. , 4.4, 3.8, 3.3,
       4.1, 4.2, 2.3, 2.8, 2.4, 2.7, 2. , 2.2, 2.5, 2.6])
```

```
df["petal.length"].unique()
```

```
array([1.4, 1.3, 1.5, 1.7, 1.6, 1.1, 1.2, 1. , 1.9, 4.7, 4.5, 4.9, 4. ,
       4.6, 3.3, 3.9, 3.5, 4.2, 3.6, 4.4, 4.1, 4.8, 4.3, 5. , 3.8, 3.7,
       5.1, 3. , 6. , 5.9, 5.6, 5.8, 6.6, 6.3, 6.1, 5.3, 5.5, 6.7, 6.9,
       5.7, 6.4, 5.4, 5.2])
```

```
df["petal.width"].unique()
```

```
array([0.2, 0.4, 0.3, 0.1, 0.5, 0.6, 1.4, 1.5, 1.3, 1.6, 1. , 1.1, 1.8,
       1.2, 1.7, 2.5, 1.9, 2.1, 2.2, 2. , 2.4, 2.3])
```

```
df["sepal.length"].value_counts().sum()
```

```
150
```

```
df["sepal.length"].value_counts()
```

```
5.0    10
5.1     9
6.3     9
5.7     8
6.7     8
5.8     7
5.5     7
6.4     7
4.9     6
5.4     6
6.1     6
6.0     6
5.6     6
4.8     5
6.5     5
6.2     4
7.7     4
6.9     4
4.6     4
5.2     4
5.9     3
4.4     3
7.2     3
6.8     3
6.6     2
4.7     2
7.6     1
7.4     1
7.3     1
7.0     1
7.1     1
5.3     1
4.3     1
```

```
4.5     1
7.9     1
Name: sepal.length, dtype: int64
```

```
df["sepal.width"].value_counts()
```

```
3.0    26
2.8    14
3.2    13
3.4    12
3.1    11
2.9    10
2.7     9
2.5     8
3.5     6
3.3     6
3.8     6
2.6     5
2.3     4
3.6     4
3.7     3
2.4     3
2.2     3
3.9     2
4.4     1
4.0     1
4.1     1
4.2     1
2.0     1
Name: sepal.width, dtype: int64
```

```
df["petal.length"].value_counts()
```

```
1.4    13
1.5    13
5.1     8
4.5     8
1.6     7
1.3     7
5.6     6
4.7     5
4.9     5
4.0     5
4.2     4
5.0     4
4.4     4
4.8     4
1.7     4
3.9     3
4.6     3
5.7     3
4.1     3
5.5     3
6.1     3
5.8     3
3.3     2
5.4     2
6.7     2
```

```
5.3     2
5.9     2
6.0     2
1.2     2
4.3     2
1.9     2
3.5     2
5.2     2
3.0     1
1.1     1
3.7     1
3.8     1
6.6     1
6.3     1
1.0     1
6.9     1
3.6     1
6.4     1
Name: petal.length, dtype: int64
```

```
df["petal.width"].value_counts()
```

```
0.2    29
1.3    13
1.8    12
1.5    12
1.4     8
2.3     8
1.0     7
0.4     7
0.3     7
2.1     6
2.0     6
0.1     5
1.2     5
1.9     5
1.6     4
2.5     3
2.2     3
2.4     3
1.1     3
1.7     2
0.6     1
0.5     1
Name: petal.width, dtype: int64
```

```
df["sepal.length"].dtype
```

```
dtype('float64')
```

```
df["sepal.width"].dtype
```

```
dtype('float64')
```

```
df["petal.length"].dtype
```

```
dtype('float64')
```

```
df["petal.width"].dtype
```

```
dtype('float64')
```

PART 1

```
#LIST DOWN THE FEATURES AND THEIR TYPES
```

```
df.columns
```

```
Index(['sepal.length', 'sepal.width', 'petal.length', 'petal.width',
       'variety'],
      dtype='object')
```

```
df["sepal.length"].dtype
```

```
dtype('float64')
```

```
df["sepal.width"].dtype
```

```
dtype('float64')
```

```
df["petal.length"].dtype
```

```
dtype('float64')
```

```
df["petal.width"].dtype
```

```
dtype('float64')
```

```
sns.countplot(data=df,x="variety")
plt.title("Species Count")
```

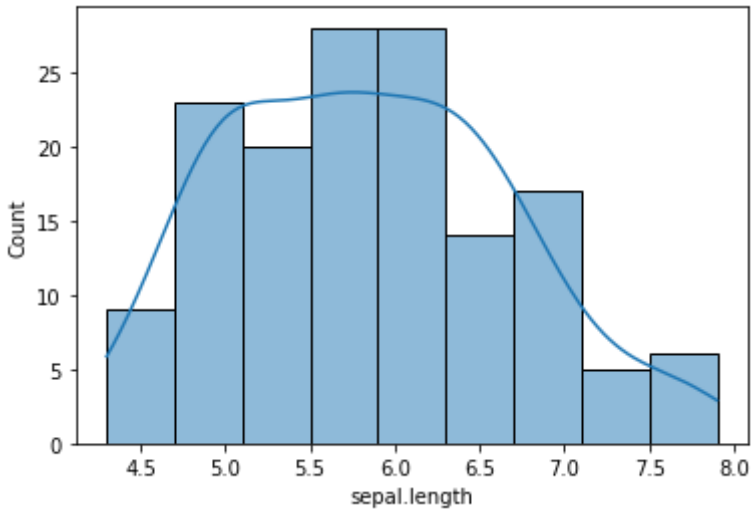
```
Text(0.5, 1.0, 'Species Count')
```

PART 2

```
#CREATE HISTOGRAM FOR EACH FEATURE IN DATASET TO ILLUSTRATE FEATRURE DISTRIBUTION
```

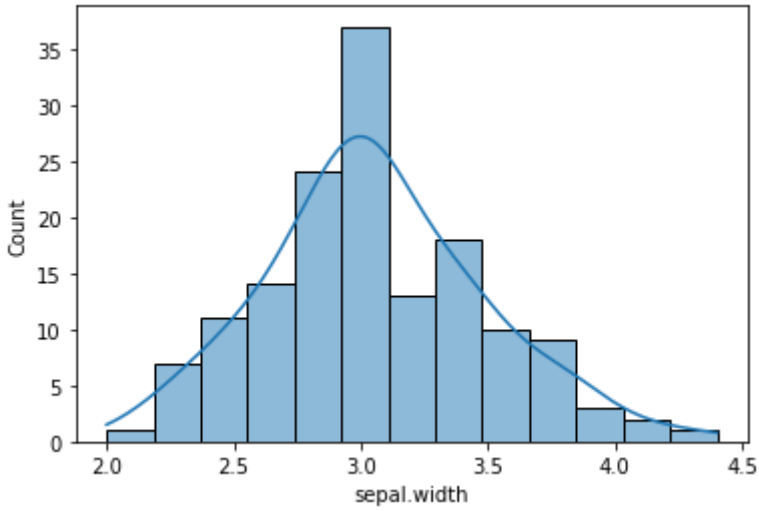
```
sns.histplot(data=df,x="sepal.length",kde=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb1a501310>
```



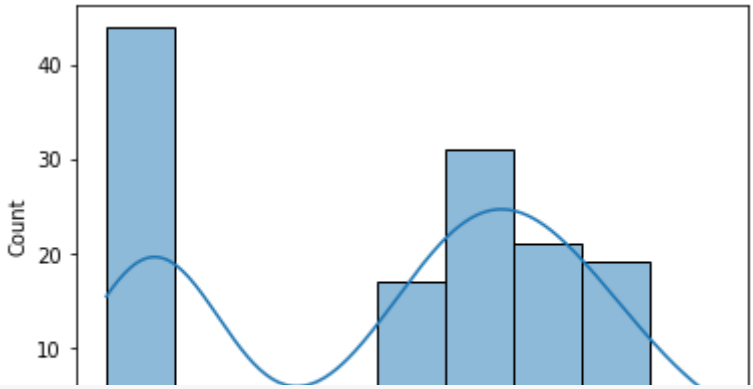

```
sns.histplot(data=df,x="sepal.width",kde=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb1a544690>
```



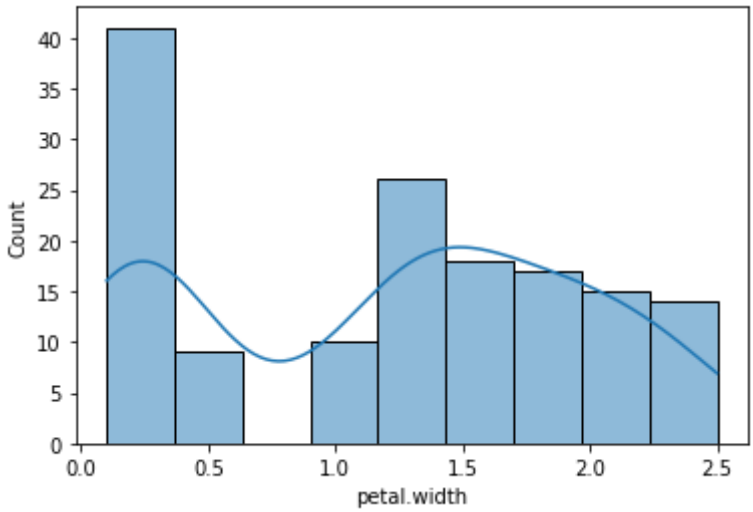

```
sns.histplot(data=df,x="petal.length",kde=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb1a3a7410>
```




```
sns.histplot(data=df,x="petal.width",kde=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb1a3d0a90>
```

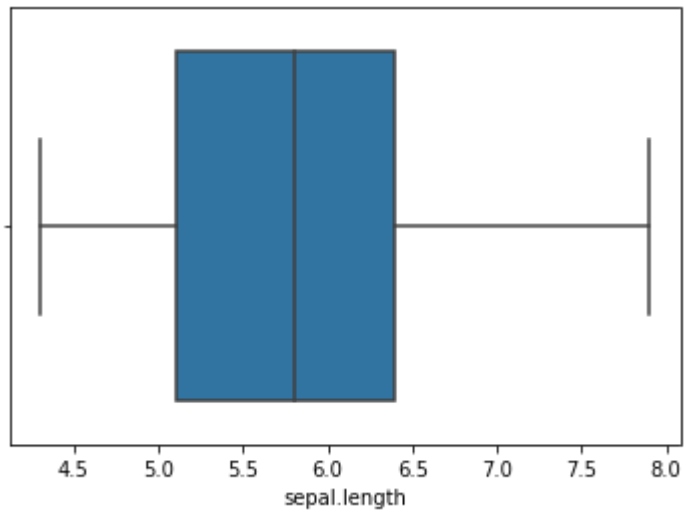


PART 3

```
#CREATE A BOX PLOT FOR EACH FEATURE IN THE DATASET
```

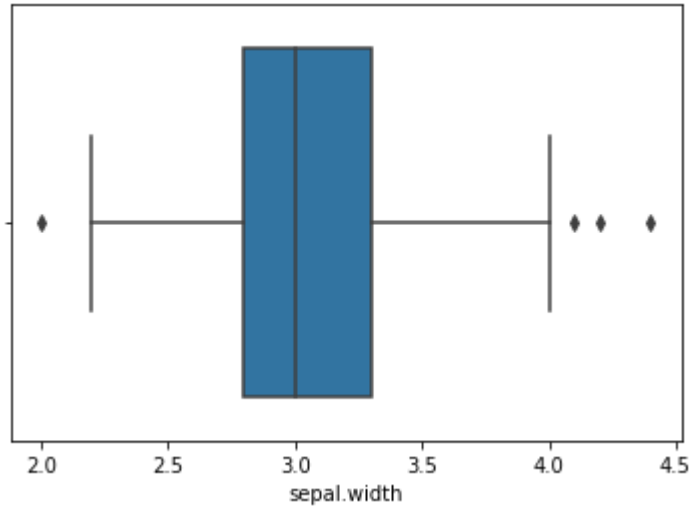
```
sns.boxplot(data=df,x="sepal.length")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb1a2b2e90>
```



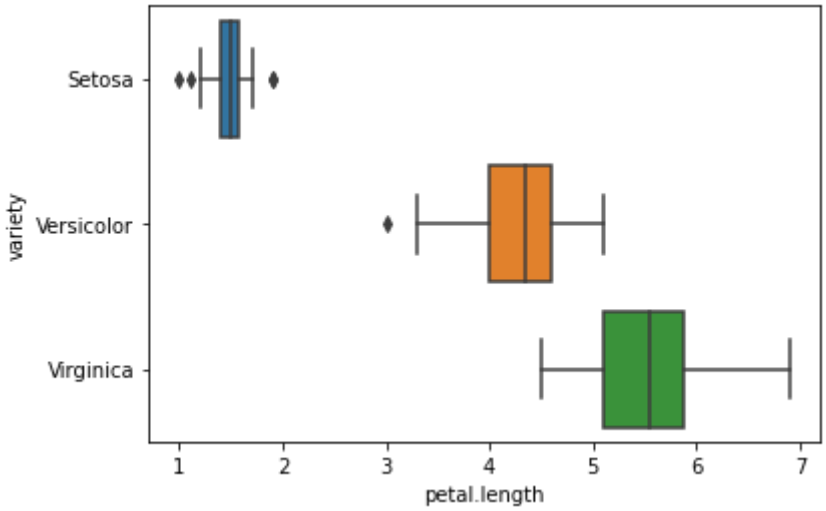
```
sns.boxplot(data=df,x="sepal.width")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb1a7b7050>
```



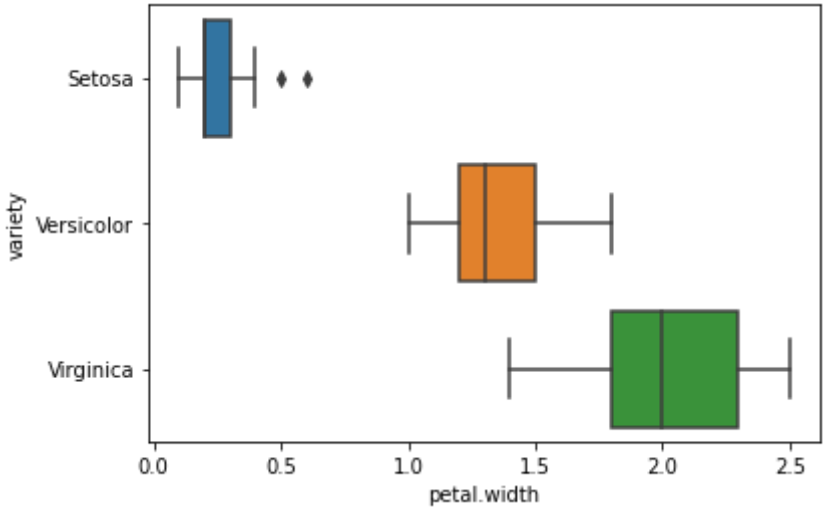

```
sns.boxplot(data=df,x="petal.length",y="variety")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb1a6e6910>
```




```
sns.boxplot(data=df,x="petal.width",y="variety")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb1a162ad0>
```

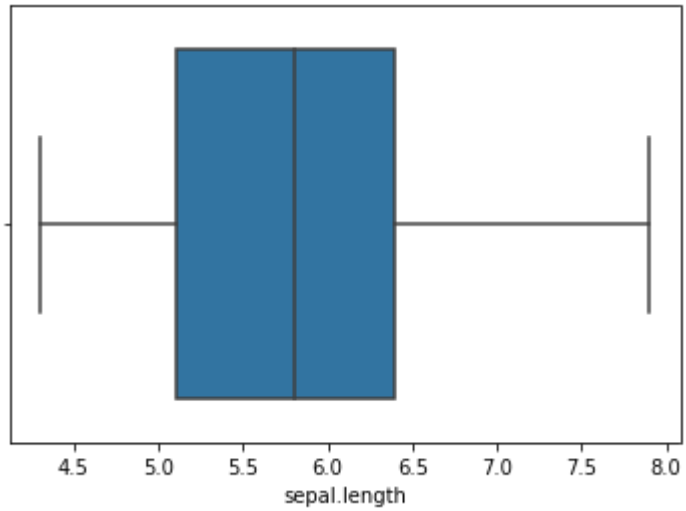


PART 4

```
#COMPARE DISTRIBUTIONS AND OUTLIERS
```

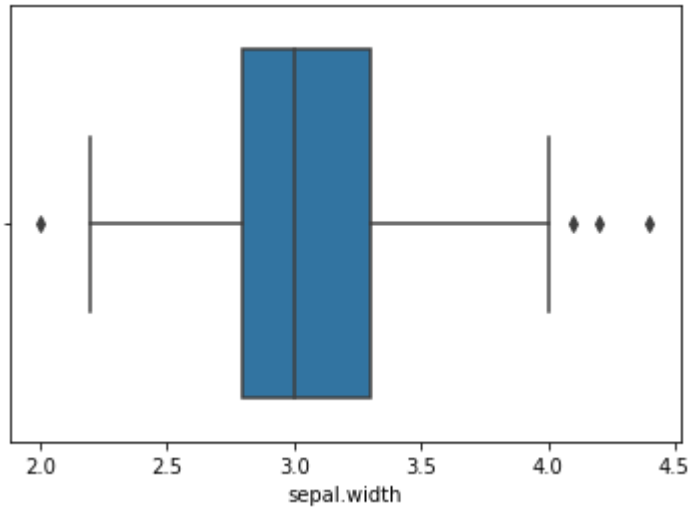
```
sns.boxplot(data=df,x="sepal.length")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb1a10f690>
```



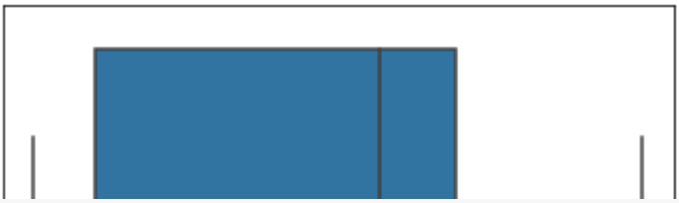

```
sns.boxplot(data=df,x="sepal.width")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb1a0744d0>
```



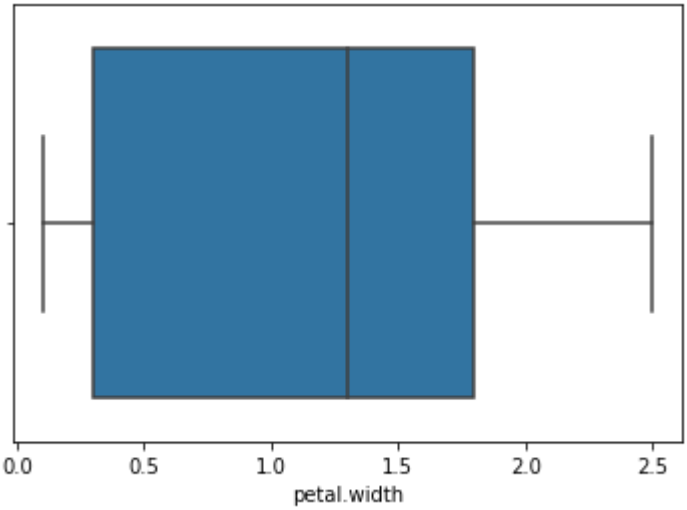

```
sns.boxplot(data=df,x="petal.length")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb19fe0e50>
```



```
sns.boxplot(data=df,x="petal.width")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb19fcf450>
```



✓ 0s completed at 10:03 PM

Group B : 01

Name	Abhishek Santosh Bankar
Roll No	305A012
Div	TE-1

Program :

```
package org.myorg;

import java.io.IOException;

import java.util.*;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.*;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class WordCount

{

	public static class Map extends Mapper<LongWritable, Text, Text, IntWritable>

	{
```

```
        private final static IntWritable one = new
        IntWritable(1); private Text word = new Text();
        public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException
        {
            String line = value.toString();
            StringTokenizer tokenizer = new
            StringTokenizer(line); while
            (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                context.write(word, one);
            }
        }
}

public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable>
{
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException
    {
        int sum = 0;
        for (IntWritable val : values)
        {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```

```
public static void main(String[] args) throws Exception
{
        Configuration conf = new Configuration(); Job job =
       new Job(conf, "wordcount");
       job.setOutputKeyClass(Text.class);
       job.setOutputValueClass(IntWritable.class);
       job.setMapperClass(Map.class);
       job.setReducerClass(Reduce.class);
       job.setInputFormatClass(TextInputFormat.class);
       job.setOutputFormatClass(TextOutputFormat.class);
       FileInputFormat.addInputPath(job, new Path(args[0]));
       FileOutputFormat.setOutputPath(job, new Path(args[1]));
       job.waitForCompletion(true);
}
```

Output :

Compile WordCount.java and create a jar:

```
$ bin/hadoop com.sun.tools.javac.Main WordCount.java
$ jar cf wc.jar WordCount*.class
```

Sample text-files as input:

```
$ bin/hadoop fs -ls /user/wordcount/input/
/user/wordcount/input/file01
/user/wordcount/input/file02
```

```
$ bin/hadoop fs -cat /user/wordcount/input/file01

Hello World Bye World


$ bin/hadoop fs -cat /user/wordcount/input/file02

Hello Hadoop Goodbye Hadoop
```

Output:

```
$ bin/hadoop jar wc.jar WordCount /user/wordcount/input /user/wordcount/output
```

Output:

```
$ bin/hadoop fs -cat /user/wordcount/output/part-r-00000

Bye 1

Goodbye 1

Hadoop 2

Hello 2

World 2
```

Group B : 02

Name	Abhishek Santosh Bankar
Roll No	305A012
Div	TE-1

SalesCountry.java

```
package SalesCountry;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
public class SalesCountryDriver
{
	public static void main(String[] args)
	{
		JobClient my_client = new JobClient();
		// Create a configuration object for the job
		JobConf job_conf = new JobConf(SalesCountryDriver.class);
		// Set a name of the Job
		job_conf.setJobName("SalePerCountry");
		// Specify data type of output key and value
		job_conf.setOutputKeyClass(Text.class);
		job_conf.setOutputValueClass(IntWritable.class);
		// Specify names of Mapper and Reducer Class
```

```
                job_conf.setMapperClass(SalesCountry.SalesMapper.class);
                job_conf.setReducerClass(SalesCountry.SalesCountryReducer.class);
                // Specify formats of the data type of Input and output
                job_conf.setInputFormat(TextInputFormat.class);
                job_conf.setOutputFormat(TextOutputFormat.class);
                // Set input and output directories using command line arguments,
                //arg[0] = name of input directory on HDFS, and arg[1] = name of output
                //directory to be created to store the output file.
                FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
                FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));
                my_client.setConf(job_conf);
                try
                 {
                        // Run the job
                        JobClient.runJob(job_conf);
                } catch (Exception e) { e.printStackTrace(); }
        }
}
```

SalesCountryReducer.java

```
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, r<Text,IntWritable>
output, Reporter reporter) throws IOException

	{

		Text key = t_key; int frequencyForCountry = 0;

		while (values.hasNext())

		{

			// replace type of value with the actual type of our value

			IntWritable value = (IntWritable) values.next(); frequencyForCountry
		+= value.get();

		}

		output.collect(key, new IntWritable(frequencyForCountry));

	}

}
```

SalesMapper.java

```
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[7]), one);
        }
}
```

Group B : 03

Name	Abhishek Santosh Bankar
Roll No	305A012
Div	TE-1

```
import java.io.IOException;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat;

import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient; import

org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.mapred.KeyValueTextInputFormat;

import org.apache.hadoop.mapred.MapReduceBase; import

org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;
```

```
import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;


public class Weather extends Configured implements Tool

{

	final long DEFAULT_SPLIT_SIZE = 128 * 1024 * 1024;

	public static class MapClass extends MapReduceBase
implements Mapper<LongWritable, Text, Text, Text>

	{

		private Text word = new Text();

		private Text values = new Text();

		public void map(LongWritable key, Text value, OutputCollector<Text,
	Text> output, Reporter reporter) throws IOException

		{

			String line = value.toString();

			StringTokenizer itr = new StringTokenizer(line);

			int counter = 0;

			String key_out = null;

			String value_str = null;

			boolean skip = false;

			loop:while (itr.hasMoreTokens() && counter<13)

			{

				String str = itr.nextToken();

				switch (counter)

				 {

					case 0:
```

```
key_out = str;
if(str.contains("STN"))
{
    //Ignoring rows where station id is all 9
    skip = true;
    break loop;
}
else
{ break;}
case 2:
int hour
=Integer.valueOf(str.substring(str.lastIndexOf("_")+1,
str.length()));
str = str.substring(4,str.lastIndexOf("_")-2);
if(hour>4 && hour<=10)
{ str = str.concat("_section1"); }
else if(hour>10 && hour<=16) { str
= str.concat("_section2"); } else
if(hour>16 && hour<=22) { str =
str.concat("_section3"); } else{ str
= str.concat("_section4"); }
key_out = key_out.concat("_").concat(str);
break;
case 3:
if(str.equals("9999.9"))
{
    skip = true;
    break loop;
```

```
                    }
                    Else
                    { value_str = str.concat(" "); break; }
                    case 4: if(str.equals("9999.9"))

                    {
                        skip = true;
                        break loop;
                    }
                    else{ value_str = value_str.concat(str).concat(" "); break;
                    } case 12:
                     if(str.equals("999.9"))
                    {
                        skip = true; break loop;
                    }
                    else{ value_str = value_str.concat(str).concat(" "); break;
                    } default: break;
                } counter++;
            }
            if(!skip)
            {
                word.set(key_out);
                values.set(value_str);
                output.collect(word, values);
            }
        }
    }
```

```
public static class MapClassForJob2 extends MapReduceBase implements Mapper<Text,
Text, Text, Text>

 {

        private Text key_text = new Text();

        private Text value_text = new Text();

        public void map(Text key, Text value, OutputCollector<Text, Text> output,
Reporter reporter) throws IOException

         {

                String str = key.toString();

                String station = str.substring(str.lastIndexOf("_")+1,

                str.length()); str = str.substring(0,str.lastIndexOf("_"));

                key_text.set(str);

                StringTokenizer itr = new StringTokenizer(value.toString());

                String str_out = station.concat("<");

                while (itr.hasMoreTokens())

                {

                         String nextToken = itr.nextToken(" ");

                        str_out = str_out.concat(nextToken);

                        str_out = ((itr.hasMoreTokens()) ? str_out.concat(",") :
                str_out.concat(">"));

                }

                value_text.set(str_out); output.collect(key_text,value_text);

        }

}

public static class Reduce extends MapReduceBase implements Reducer<Text, Text,
Text, Text> {

        private Text value_out_text = new Text();

        public void reduce(Text key, Iterator<Text> values, OutputCollector<Text, Text>
output, Reporter reporter) throws IOException
```

```
{
        double sum_temp = 0;
        double sum_dew = 0;
        double sum_wind = 0;
        int count = 0;
        while (values.hasNext())
        {
                String str = values.next().toString();
                StringTokenizer itr = new StringTokenizer(str);
                int count_vector = 0;
                while (itr.hasMoreTokens())
                {
                        String nextToken = itr.nextToken(" ");
                        if(count_vector==0)
                        {
                                sum_temp += Double.valueOf(nextToken);
                        }
                        if(count_vector==1)
                        {
                                sum_dew += Double.valueOf(nextToken);
                        }
                        if(count_vector==2)
                        {
                                sum_wind += Double.valueOf(nextToken);
                        }
                        count_vector++;
                } count++;
```

```
                }

                double avg_tmp = sum_temp / count;

                double avg_dew = sum_dew / count;

                double avg_wind = sum_wind / count;

                System.out.println(key.toString()+" count is "+count+" sum of temp is
        "+sum_temp+" sum of dew is "+sum_dew+" sum of wind is "+sum_wind+"\n");

                String value_out =
        String.valueOf(avg_tmp).concat("").concat(String.valueOf(avg_dew)).concat("
        ").concat(String.valueOf(avg_wind));

                 value_out_text.set(value_out);

                output.collect(key, value_out_text);

        }

}

public static class ReduceForJob2 extends MapReduceBase implements Reducer<Text, Text,
Text, Text>

 {

        private Text value_out_text = new Text();

        public void reduce(Text key, Iterator<Text> values, OutputCollector<Text, Text>
output, Reporter reporter) throws IOException

        {

                 String value_out = "";

                while (values.hasNext())

                {

                        value_out = value_out.concat(values.next().toString()).concat(" ");

                }

                 value_out_text.set(value_out);

                output.collect(key, value_out_text);

        }

}
```

```
static int printUsage()
{
	System.out.println("weather [-m <maps>] [-r <reduces>] <job_1 input> <job_1 output> <job_2 output>");
	ToolRunner.printGenericCommandUsage(System.out);
	return -1;
}
public int run(String[] args) throws Exception
{
	Configuration config = getConf();
	JobConf conf = new JobConf(config, Weather.class);
	conf.setJobName("Weather Job1");
	conf.setOutputKeyClass(Text.class);
	conf.setOutputValueClass(Text.class);
	conf.setMapOutputKeyClass(Text.class);
	conf.setMapOutputValueClass(Text.class);
	conf.setMapperClass(MapClass.class);
	conf.setReducerClass(Reduce.class);
	List<String> other_args = new ArrayList<String>();
	for(int i=0; i < args.length; ++i)
	{
		try
		{
			if ("-m".equals(args[i]))
			{
				conf.setNumMapTasks(Integer.parseInt(args[++i]));
			}
```

```
                else if ("-r".equals(args[i]))
                {
                        conf.setNumReduceTasks(Integer.parseInt(args[++i]));
                }
                else
                {
                        other_args.add(args[i]);
                }
        }
        catch (NumberFormatException except)
        {
                System.out.println("ERROR: Integer expected instead of " + args[i]);
                return printUsage();
        }
        catch (ArrayIndexOutOfBoundsException except)
        {
                 System.out.println("ERROR: Required parameter missing from "
        + args[i-1]);
                return printUsage();
        }
}
FileInputFormat.setInputPaths(conf, other_args.get(0));
FileOutputFormat.setOutputPath(conf, new Path(other_args.get(1)));
JobClient.runJob(conf);
JobConf conf2 = new JobConf(config, Weather.class);
conf2.setJobName("Weather Job 2");
conf2.setOutputKeyClass(Text.class);
conf2.setOutputValueClass(Text.class);
```

```
        conf2.setInputFormat(KeyValueTextInputFormat.class);
        conf2.setMapOutputKeyClass(Text.class);
        conf2.setMapOutputValueClass(Text.class);
        conf2.setMapperClass(MapClassForJob2.class);
        conf2.setReducerClass(ReduceForJob2.class);
        FileInputFormat.setInputPaths(conf2, new Path(other_args.get(1)));
        FileOutputFormat.setOutputPath(conf2, new Path(other_args.get(2)));
        JobClient.runJob(conf2);
        return 0;
}
public static void main(String[] args) throws Exception
{
         int res = ToolRunner.run(new Configuration(), new Weather(), args);
        System.exit(res);
}
}
```