

Assignment No 1 : Data wrangling 1

1. Import Required Libraries

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

2. Locate open source data from web

Dataset link : <https://www.kaggle.com/c/titanic/data> (<https://www.kaggle.com/c/titanic/data>)

3. Load the dataset into Pandas Dataframe

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

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

```
Out[3]:
```

	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

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W/C 6807	23.4500
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000
890	891	0	3	Dooley, Mr.	male	30.0	0	0	373450	7.2500

4. Data Preprocessing

```
In [4]: # check dimensions of dataframe
df.shape
```

```
Out[4]: (891, 12)
```

```
In [5]: # check the columns present
df.columns
```

```
Out[5]: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
   'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
   dtype='object')
```

```
In [6]: # check total size (rows * columns)
df.size
```

```
Out[6]: 10692
```

```
In [7]: # check datatypes of each column
df.dtypes
```

```
Out[7]:
```

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
```

```
In [8]: #check for missing values
df.isnull().sum()
```

```
Out[8]: 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 [9]: # check initial statistics
df.describe()
```

	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 [10]: # Print information about the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64
```

5. Data Formatting and Data Normalization

```
In [11]: # Handling missing values
# 1. fillna() : method which we can use replace null(NaN) values with some other value.
# or we can perform forward fill(ffill), backward fill(bfill).

# 2. interpolate() : performs Linear interpolation(by default but you can change it)
# and replaces the null with the result.

# 3. dropna() : It will drop all the rows that have NaN values. but using this
```

```
In [12]: df["Cabin"] = df["Cabin"].replace(to_replace=np.nan, value="unknown")
df
```

```
Out[12]:
```

	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
...
886	887	0	2	Montville, Rev. Juozas	male	27.0	0	0	211536	13.0000
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W/C- 6607	23.4500
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000
890	891	0	3	Dooley, Mr. Dennis	male	32.0	0	0	370376	7.7500

```
In [13]: df["Age"] = df["Age"].interpolate()
df
```

	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
...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	22.5	1	2	W/C- 6607	23.4500
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000

```
In [16]: df.isnull().sum()
```

```
Out[16]: PassengerId      0  
Survived        0  
Pclass          0  
Name            0  
Sex             0  
Age             0  
SibSp           0  
Parch           0  
Ticket          0  
Fare            0  
Cabin           0  
Embarked        0  
dtype: int64
```

```
In [17]: # Changing Datatypes  
df.dtypes
```

```
Out[17]: PassengerId      int64  
Survived        int64  
Pclass          int64  
Name            object  
Sex             object  
Age             float64  
SibSp           int64  
Parch           int64  
Ticket          object  
Fare            float64  
Cabin           object  
Embarked        object  
dtype: object
```

```
In [18]: df["Age"] = df["Age"].astype('int64')
```

```
In [19]: df.dtypes
```

```
Out[19]: PassengerId      int64  
Survived        int64  
Pclass          int64  
Name            object  
Sex             object  
Age             int64  
SibSp           int64  
Parch           int64  
Ticket          object  
Fare            float64  
Cabin           object  
Embarked        object  
dtype: object
```

6. Turning categorical variables into quantitative

```
In [20]: quantitative_data = pd.get_dummies(df.Embarked,prefix = 'Embarked')
```

```
In [21]: quantitative_data
```

Out[21]:	Embarked_C	Embarked_Q	Embarked_S
0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	0	0	1
...
886	0	0	1
887	0	0	1
888	0	0	1
889	1	0	0
890	0	1	0

891 rows × 3 columns

```
In [22]: df = df.join(quantitative_data)
df
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500
1	2	1	Cummings, Mrs. John Bradley (Florence Briggs Th...)	female	38	1	0	PC 17599	71.2833
2	3	1	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000
4	5	0	Allen, Mr. William Henry	male	35	0	0	373450	8.0500

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
886	887	0	2	Montvila, Rev. Juozas	male	27	0	0	211536	13.0000
887	888	1	1	Graham, Miss. Margaret Edith	female	19	0	0	112053	30.0000
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	22	1	2	W/J.C. 6607	23.4500
889	890	1	1	Behr, Mr. Karl Howell	male	26	0	0	111369	30.0000
***	***	0	0	Dooley, ***	male	38	0	0	396852	7.2500

```
In [23]: df.drop(['Embarked'],axis = 1,inplace = True)
df
```

Out[23]:	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
	0	1	0	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500
	1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38	1	0	PC 17599	71.2833
	2	3	1	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250
	3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000
	4	5	0	Allen, Mr. William Henry	male	35	0	0	373450	8.0500
	***	***	***	Montvila, Rev. Juozas	male	27	0	0	211536	13.0000
	886	887	0	Graham, Miss. Margaret Edith	female	19	0	0	112053	30.0000
	887	888	1							

Assignment No 2 : Data Wrangling 2

1. Import Required Libraries

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

2. Load the dataset into dataframe

```
In [2]: df = pd.read_csv('StudentPerformance.csv')
```

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

```
Out[3]:
```

	math_score	reading_score	writing_score	placement_score	club_join_date	placement_offer_
0	71.0	78	65.0	99.0	2018.0	
1	63.0	86	71.0	100.0	NaN	
2	71.0	92	64.0	76.0	2021.0	
3	77.0	95	68.0	84.0	2021.0	
4	63.0	81	70.0	76.0	2021.0	
5	NaN	92	NaN	89.0	2020.0	
6	66.0	75	67.0	93.0	2019.0	
7	50.0	100	68.0	200.0	2015.0	
8	60.0	90	61.0	87.0	2021.0	
9	64.0	89	72.0	80.0	2019.0	
10	63.0	81	62.0	95.0	2021.0	
11	66.0	95	77.0	90.0	2020.0	
12	60.0	88	73.0	91.0	2020.0	
13	72.0	80	69.0	98.0	2021.0	
14	60.0	76	NaN	82.0	2020.0	
15	69.0	83	69.0	NaN	2019.0	
16	72.0	80	20.0	75.0	2021.0	
17	90.0	95	66.0	85.0	2002.0	
18	70.0	89	73.0	90.0	2020.0	

	math_score	reading_score	writing_score	placement_score	club_join_date	placement_offer_
19	66.0	91	77.0	100.0	2020.0	
20	61.0	75	74.0	76.0	NaN	
21	NaN	77	79.0	82.0	2018.0	
22	65.0	95	60.0	100.0	2021.0	
23	63.0	86	78.0	82.0	2019.0	
24	78.0	95	66.0	NaN	2019.0	
25	71.0	84	72.0	89.0	2018.0	
26	78.0	78	73.0	87.0	2018.0	
27	67.0	79	67.0	89.0	2018.0	
28	NaN	NaN	NaN	NaN	NaN	NaN

Check the dimensions of dataframe

```
In [4]: df.shape
```

```
Out[4]: (30, 6)
```

Check the number of null values in each column

```
In [5]: df.isnull().sum()
```

```
Out[5]: math_score      2
reading_score     0
writing_score     3
placement_score   2
club_join_date    2
placement_offer_count 1
dtype: int64
```

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

	math_score	reading_score	writing_score	placement_score	club_join_date	placement_offer_
0	False	False	False	False	False	False
1	False	False	False	False	False	True
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
5	True	False	True	False	False	False
6	False	False	False	False	False	False
7	False	False	False	False	False	False

	math_score	reading_score	writing_score	placement_score	club_join_date	placement_offer_
8	False	False	False	False	False	False
9	False	False	False	False	False	False
10	False	False	False	False	False	False
11	False	False	False	False	False	False
12	False	False	False	False	False	False
13	False	False	False	False	False	False
14	False	False	True	False	False	False
15	False	False	False	True	False	False
16	False	False	False	False	False	False
17	False	False	False	False	False	False
18	False	False	False	False	False	False
19	False	False	False	False	False	False
20	False	False	False	False	False	True
21	True	False	False	False	False	False
22	False	False	False	False	False	False
23	False	False	False	False	False	False
24	False	False	False	True	False	False
25	False	False	False	False	False	False
26	False	False	False	False	False	False
27	False	False	False	False	False	False
28	False	False	True	False	False	False

Handling missing/null values

```
In [7]: df['math_score']=df['math_score'].interpolate()
df
```

	math_score	reading_score	writing_score	placement_score	club_join_date	placement_offer_
0	71.0	78	65.0	99.0	2018.0	
1	63.0	86	71.0	100.0		NaN
2	71.0	92	64.0	76.0	2021.0	
3	77.0	95	68.0	84.0	2021.0	
4	63.0	81	70.0	76.0	2021.0	
5	64.5	92	NaN	89.0	2020.0	
6	66.0	75	67.0	93.0	2019.0	
7	50.0	100	68.0	200.0	2015.0	

	math_score	reading_score	writing_score	placement_score	club_join_date	placement_offer_
5	64.5	92	70.0	89.0	2020.0	
6	66.0	75	67.0	93.0	2019.0	
7	50.0	100	68.0	200.0	2015.0	
8	60.0	90	61.0	87.0	2021.0	
9	64.0	89	72.0	80.0	2019.0	
10	63.0	81	62.0	95.0	2021.0	
11	66.0	95	77.0	90.0	2020.0	
12	60.0	88	73.0	91.0	2020.0	
13	72.0	80	69.0	98.0	2021.0	
14	60.0	76	69.0	82.0	2020.0	
15	69.0	83	69.0	NaN	2019.0	
16	72.0	80	20.0	75.0	2021.0	
17	90.0	95	66.0	85.0	2002.0	
18	70.0	89	73.0	90.0	2020.0	
19	66.0	91	77.0	100.0	2020.0	
20	61.0	75	74.0	76.0	NaN	
21	63.0	77	79.0	82.0	2018.0	
22	65.0	95	60.0	100.0	2021.0	
23	63.0	86	78.0	82.0	2019.0	
24	78.0	95	66.0	NaN	2019.0	
25	71.0	84	72.0	89.0	2018.0	
26	78.0	78	73.0	87.0	2018.0	
27	67.0	79	67.0	89.0	2018.0	
28	75.0	79	67.0	97.0	2020.0	

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

```
Out[10]: math_score      0
reading_score     0
writing_score     0
placement_score    2
club_join_date    2
placement_offer_count 1
dtype: int64
```

```
In [11]: df['placement_score'] = df['placement_score'].fillna(method='bfill')
df
```

```
Out[11]:   math_score  reading_score  writing_score  placement_score  club_join_date  placement_offer_
0        71.0          78            65.0          99.0        2018.0
1        63.0          86            71.0          100.0         NaN
```

	math_score	reading_score	writing_score	placement_score	club_join_date	placement_offer_count
2	71.0	92	64.0	76.0	2021.0	0
3	77.0	95	68.0	84.0	2021.0	0
4	63.0	81	70.0	76.0	2021.0	0
5	64.5	92	70.0	89.0	2020.0	0
6	66.0	75	67.0	93.0	2019.0	0
7	50.0	100	68.0	200.0	2015.0	0
8	60.0	90	61.0	87.0	2021.0	0
9	64.0	89	72.0	80.0	2019.0	0
10	63.0	81	62.0	95.0	2021.0	0
11	66.0	95	77.0	90.0	2020.0	0
12	60.0	88	73.0	91.0	2020.0	0
13	72.0	80	69.0	98.0	2021.0	0
14	60.0	76	69.0	82.0	2020.0	0
15	69.0	83	69.0	75.0	2019.0	0
16	72.0	80	20.0	75.0	2021.0	0
17	90.0	95	66.0	85.0	2002.0	0
18	70.0	89	73.0	90.0	2020.0	0
19	66.0	91	77.0	100.0	2020.0	0
20	61.0	75	74.0	76.0	NaN	0
21	63.0	77	79.0	82.0	2018.0	0
22	65.0	95	60.0	100.0	2021.0	0
23	63.0	86	78.0	82.0	2019.0	0
24	78.0	95	66.0	89.0	2019.0	0
25	71.0	84	72.0	89.0	2018.0	0
26	78.0	78	73.0	87.0	2018.0	0
27	67.0	79	67.0	89.0	2018.0	0
28	75.0	79	67.0	97.0	2020.0	0

```
In [12]: df.isnull().sum()
```

```
Out[12]: math_score      0
reading_score     0
writing_score     0
placement_score   0
club_join_date    2
placement_offer_count 1
dtype: int64
```

```
In [13]: df['club_join_date'] = df['club_join_date'].replace(to_replace=np.nan,value='2
df
```

Out[13]:

	math_score	reading_score	writing_score	placement_score	club_join_date	placement_offer_
0	71.0	78	65.0	99.0	2018	
1	63.0	86	71.0	100.0	2019	
2	71.0	92	64.0	76.0	2021	
3	77.0	95	68.0	84.0	2021	
4	63.0	81	70.0	76.0	2021	
5	64.5	92	70.0	89.0	2020	
6	66.0	75	67.0	93.0	2019	
7	50.0	100	68.0	200.0	2015	
8	60.0	90	61.0	87.0	2021	
9	64.0	89	72.0	80.0	2019	
10	63.0	81	62.0	95.0	2021	
11	66.0	95	77.0	90.0	2020	
12	60.0	88	73.0	91.0	2020	
13	72.0	80	69.0	98.0	2021	
14	60.0	76	69.0	82.0	2020	
15	69.0	83	69.0	75.0	2019	
16	72.0	80	20.0	75.0	2021	
17	90.0	95	66.0	85.0	2002	
18	70.0	89	73.0	90.0	2020	
19	66.0	91	77.0	100.0	2020	
20	61.0	75	74.0	76.0	2019	
21	63.0	77	79.0	82.0	2018	
22	65.0	95	60.0	100.0	2021	
23	63.0	86	78.0	82.0	2019	
24	78.0	95	66.0	89.0	2019	
25	71.0	84	72.0	89.0	2018	
26	78.0	78	73.0	87.0	2018	
27	67.0	79	67.0	89.0	2018	
28	75.0	79	67.0	97.0	2020	
29	73.0	78	61.0	75.0	2020	

In [14]: df.isnull().sum()

Out[14]:

```
math_score          0  
reading_score      0
```

```
In [15]: df['placement_offer_count']=df['placement_offer_count'].fillna('1')  
df
```

```
Out[15]:   math_score  reading_score  writing_score  placement_score  club_join_date  placement_offer_  
0           71.0          78.0         65.0          99.0        2018  
1           63.0          86.0         71.0          100.0       2019  
2           71.0          92.0         64.0          76.0        2021  
3           77.0          95.0         68.0          84.0        2021  
4           63.0          81.0         70.0          76.0        2021  
5           64.5          92.0         70.0          89.0        2020  
6           66.0          75.0         67.0          93.0        2019  
7           50.0          100.0        68.0          200.0       2015  
8           60.0          90.0         61.0          87.0        2021  
9           64.0          89.0         72.0          80.0        2019  
10          63.0          81.0         62.0          95.0        2021  
11          66.0          95.0         77.0          90.0        2020  
12          60.0          88.0         73.0          91.0        2020  
13          72.0          80.0         69.0          98.0        2021  
14          60.0          76.0         69.0          82.0        2020  
15          69.0          83.0         69.0          75.0        2019  
16          72.0          80.0         20.0          75.0        2021  
17          90.0          95.0         66.0          85.0        2002  
18          70.0          89.0         73.0          90.0        2020  
19          66.0          91.0         77.0          100.0       2020  
20          61.0          75.0         74.0          76.0        2019  
21          63.0          77.0         79.0          82.0        2018  
22          65.0          95.0         60.0          100.0       2021  
23          63.0          86.0         78.0          82.0        2019  
24          78.0          95.0         66.0          89.0        2019  
25          71.0          84.0         72.0          89.0        2018  
26          78.0          78.0         73.0          87.0        2018  
27          67.0          79.0         67.0          89.0        2018  
28          75.0          79.0         67.0          97.0        2020  
29          73.0          78.0         61.0          75.0        2020
```

```
In [16]: df.isnull().sum()
```

```
Out[16]: math_score      0  
reading_score     0  
writing_score      0  
placement_score    0  
club_join_date     0  
placement_offer_count 0  
dtype: int64
```

Check the data types and convert into appropriate types

```
In [17]: df.dtypes
```

```
Out[17]: math_score      float64  
reading_score     int64  
writing_score      float64  
placement_score    float64  
club_join_date     object  
placement_offer_count  object  
dtype: object
```

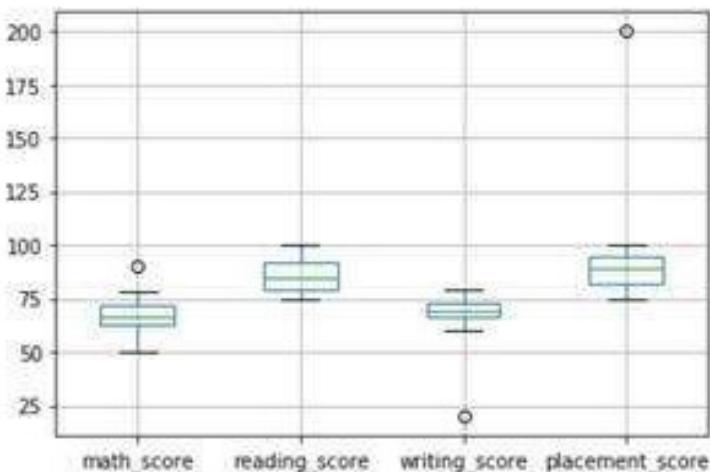
```
In [18]: df['math_score']=df['math_score'].astype('int64')  
df['reading_score']=df['reading_score'].astype('int64')  
df['writing_score']=df['writing_score'].astype('int64')  
df['placement_score']=df['placement_score'].astype('int64')  
df['club_join_date']=df['club_join_date'].astype('int64')  
df['placement_offer_count']=df['placement_offer_count'].astype('int64')  
df.dtypes
```

```
Out[18]: math_score      int64  
reading_score     int64  
writing_score      int64  
placement_score    int64  
club_join_date     int64  
placement_offer_count  int64  
dtype: object
```

Check for outliers

```
In [19]: columns = ['math_score', 'reading_score', 'writing_score', 'placement_score']
df.boxplot(columns)
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x5d9471ffc8>
```



```
In [20]: np.where(df['math_score'] < 60)
```

```
Out[20]: (array([7], dtype=int64),)
```

```
In [21]: np.where(df['math_score'] > 80)
```

```
Out[21]: (array([17], dtype=int64),)
```

```
In [22]: np.where(df['reading_score'] < 75)
```

```
Out[22]: (array([], dtype=int64),)
```

```
In [23]: np.where(df['reading_score'] > 95)
```

```
Out[23]: (array([7], dtype=int64),)
```

```
In [24]: np.where(df['writing_score'] < 60)
```

```
Out[24]: (array([16], dtype=int64),)
```

```
In [25]: np.where(df['writing_score'] > 80)
```

```
Out[25]: (array([], dtype=int64),)
```

```
In [26]: np.where(df['placement_score'] < 75)
```

```
Out[26]: (array([], dtype=int64),)
```

```
In [27]: np.where(df['placement_score'] > 100)
```

```
Out[27]: (array([7], dtype=int64),)
```

```
    math_score reading_score writing_score placement_score club_join_date placement_offer
In [37]: df[(df['placement_score'] < 75) & (df['placement_offer_count'] > 1)]
Out[37]:   math_score reading_score writing_score placement_score club_join_date placement_offer_co
In [38]: df[((df['placement_score'] > 75) & (df['placement_score'] <= 85)) & ((df['placement_offer_count'] > 2) | (df['placement_offer_count'] < 2))]
Out[38]:   math_score reading_score writing_score placement_score club_join_date placement_offer_
          17        90        95        66        85      2002
          21        63        77        79        82      2018
In [39]: df[((df['placement_score'] > 85) & (df['placement_offer_count'] < 3))]
Out[39]:   math_score reading_score writing_score placement_score club_join_date placement_offer_
          10        63        81        62        95      2021
          24        78        95        66        89      2019
```

Outlier Removal

```
In [40]: new_df1 = df[
    ((df.math_score >= 60) & (df.math_score <= 80)) &
    ((df.reading_score >= 75) & (df.reading_score <= 95)) &
    ((df.writing_score >= 60) & (df.writing_score <= 80)) &
    ((df.placement_score >= 75) & (df.placement_score <= 100)) &
    ((df.club_join_date >= 2018) & (df.club_join_date <= 2021))]
In [41]: new_df1
Out[41]:   math_score reading_score writing_score placement_score club_join_date placement_offer_
          0        71        78        65        99      2018
          1        63        86        71        100     2019
          2        71        92        64        76      2021
          3        77        95        68        84      2021
          4        63        81        70        76      2021
          5        64        92        70        89      2020
          6        66        75        67        93      2019
          8        60        90        61        87      2021
          9        64        89        72        80      2019
         10        63        81        62        95      2021
         11        66        95        77        90      2020
         12        60        88        73        91      2020
```

```
math_score  reading_score  writing_score  placement_score  club_join_date  placement_offer_
13          72              80             69                  98            2021
14          60              76             69                  82            2020
15          69              83             69                  75            2019
18          70              89             73                  90            2020
19          66              91             77                  100           2020
20          61              75             74                  76            2019
21          63              77             79                  82            2018
22          65              95             60                  100           2021
23          63              86             78                  82            2019
24          78              95             66                  89            2019
25          71              84             72                  89            2018
26          78              78             73                  87            2018
27          67              79             67                  89            2018
28          75              79             67                  97            2020
```

```
In [42]: new_df1[(new_df1['placement_score'] < 75) & (new_df1['placement_offer_count'] > 2)]
```

```
Out[42]:   math_score  reading_score  writing_score  placement_score  club_join_date  placement_offer_count
```

```
In [43]: new_df1[((new_df1['placement_score'] >= 75) & (new_df1['placement_score'] <= 85)) | ((new_df1['placement_offer_count'] > 2) & (new_df1['placement_offer_count'] <= 3))]
```

```
Out[43]:   math_score  reading_score  writing_score  placement_score  club_join_date  placement_offer_count
```

	math_score	reading_score	writing_score	placement_score	club_join_date	placement_offer_count
15	69	83	69	75	2019	
21	63	77	79	82	2018	

```
In [44]: new_df1[((new_df1['placement_score'] > 85) & (new_df1['placement_offer_count'] <= 3))]
```

```
Out[44]:   math_score  reading_score  writing_score  placement_score  club_join_date  placement_offer_count
```

	math_score	reading_score	writing_score	placement_score	club_join_date	placement_offer_count
10	63	81	62	95	2021	
24	78	95	66	89	2019	

```
In [45]: new_df2 = new_df1[
    ((new_df1.placement_score < 75) & (new_df1.placement_offer_count==1)) | 
    (((new_df1.placement_score >= 75) & (new_df1.placement_score <= 85)) & (new_df1.placement_offer_count==2)) |
    ((new_df1.placement_score > 85) & (new_df1.placement_offer_count==3))]
```

```
In [46]: new_df2
```

```
Out[46]:   math_score  reading_score  writing_score  placement_score  club_join_date  placement_offer_count
```

	math_score	reading_score	writing_score	placement_score	club_join_date	placement_offer_count
0	71	78	65	99	2018	
1	63	86	71	100	2019	

	math_score	reading_score	writing_score	placement_score	club_join_date	placement_offer
2	71	92	64	76	2021	
3	77	95	68	84	2021	
4	63	81	70	76	2021	
5	64	92	70	89	2020	
6	66	75	67	93	2019	
8	60	90	61	87	2021	
9	64	89	72	80	2019	
11	66	95	77	90	2020	
12	60	88	73	91	2020	
13	72	80	69	98	2021	
14	60	76	69	82	2020	
18	70	89	73	90	2020	
19	66	91	77	100	2020	
20	61	75	74	76	2019	
22	65	95	60	100	2021	
23	63	86	78	82	2019	
25	71	84	72	89	2018	
26	78	78	73	87	2018	
27	67	79	67	89	2018	
28	75	79	67	97	2020	

In [47]: new_df2.shape

Out[47]: (23, 6)

Check if the outliers are completely removed

In [48]: new_df2[(new_df2['math_score'] < 60) | (new_df2['math_score'] > 80)]

Out[48]: math_score reading_score writing_score placement_score club_join_date placement_offer_co

In [49]: new_df2[(new_df2['reading_score'] < 75) | (new_df2['reading_score'] > 95)]

Out[49]: math_score reading_score writing_score placement_score club_join_date placement_offer_co

In [50]: new_df2[(new_df2['writing_score'] < 60) | (new_df2['writing_score'] > 80)]

Out[50]: math_score reading_score writing_score placement_score club_join_date placement_offer_co

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

Import Required Libraries

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

Read csv into Dataframe

```
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	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

Data Preprocessing

```
In [4]: df.shape
```

```
Out[4]: (150, 6)
```

```
In [5]: df.isnull().sum()
```

```
Out[5]: Id          0  
SepalLengthCm    0  
SepalWidthCm     0  
PetalLengthCm    0  
PetalWidthCm     0  
Species          0  
dtype: int64
```

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

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

```
In [7]: 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 [8]: df.drop(columns = 'Id', inplace = True)  
df
```

```
Out[8]:   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species  
0           5.1         3.5          1.4          0.2  Iris-setosa  
1           4.9         3.0          1.4          0.2  Iris-setosa  
2           4.7         3.2          1.3          0.2  Iris-setosa  
3           4.6         3.1          1.5          0.2  Iris-setosa  
4           5.0         3.6          1.4          0.2  Iris-setosa  
..          ..          ..          ..          ..  ...  
145          6.7         3.0          5.2          2.3  Iris-virginica  
146          6.3         2.5          5.0          1.9  Iris-virginica  
147          6.5         3.0          5.2          2.0  Iris-virginica
```

```
SepalLengthCm      0.825301
SepalWidthCm       0.432147
PetalLengthCm     1.758529
PetalWidthCm      0.760613
```

Calculating minimum value

```
In [16]: np.min(df)
```

```
Out[16]: SepalLengthCm      4.3
          SepalWidthCm       2
          PetalLengthCm     1
          PetalWidthCm      0.1
          Species           Iris-setosa
          dtype: object
```

Calculating maximum value

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

```
Out[17]: SepalLengthCm      7.9
          SepalWidthCm       4.4
          PetalLengthCm     6.9
          PetalWidthCm      2.5
          Species           Iris-virginica
          dtype: object
```

Calculating 1st quantile (25th percentile)

```
In [18]: df.quantile(0.25)
```

```
Out[18]: SepalLengthCm      5.1
          SepalWidthCm       2.8
          PetalLengthCm     1.6
          PetalWidthCm      0.3
          Name: 0.25, dtype: float64
```

Calculating 2nd quantile (50th percentile)

```
In [19]: df.quantile(0.50)
```

```
Out[19]: SepalLengthCm      5.80
          SepalWidthCm       3.00
          PetalLengthCm     4.35
          PetalWidthCm      1.30
          Name: 0.5, dtype: float64
```

Calculating 3rd quantile (75th percentile)

```
In [20]: df.quantile(0.75)
```

```
Out[20]:
```

SepalLengthCm	6.4
SepalWidthCm	3.3
PetalLengthCm	5.1
PetalWidthCm	1.8

Assignment No 4 : Data Analytics 1

Reading Required Dataset

```
In [1]: from sklearn.datasets import load_boston  
boston = load_boston()
```

```
In [2]: 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, 2
2.9, 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., 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,
```

Importing Required Libraries

```
In [3]: import pandas as pd  
import numpy as np  
from sklearn import linear_model  
from sklearn.model_selection import train_test_split
```

Creating Dataframe

```
In [4]: df_x = pd.DataFrame(boston.data, columns=boston.feature_names)
```

```
In [5]: df_y = pd.DataFrame(boston.target)
```

```
In [6]: df_x
```

```
Out[6]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
0	0.00632	18.0	2.31	0.0	0.538	5.575	65.2	4.0900	1.0	296.0	15.3	396.90	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	
..	
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	

506 rows × 13 columns

```
In [8]: df_y
```

```
Out[8]:
```

	0
0	24.0
1	21.6
2	34.7
3	33.4
4	36.2
..	..
501	22.4

```
      0  
502  20.6  
503  23.9  
504  22.0  
505  11.9
```

```
In [9]: df_x.describe()
```

```
Out[9]:      CRIM       ZN      INDUS      CHAS      NOX      RM      AGE  
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000  506.  
mean   3.613524  11.363636  11.136779  0.069170  0.554695  6.284634  68.574901  3.  
std    8.601545  23.322453  6.860353  0.253994  0.115878  0.702617  28.148861  2.  
min    0.006320  0.000000  0.460000  0.000000  0.385000  3.561000  2.900000  1.  
25%   0.062045  0.000000  5.190000  0.000000  0.449000  5.885500  45.025000  2.  
50%   0.256510  0.000000  9.690000  0.000000  0.538000  6.208500  77.500000  3.  
75%   3.677083  12.500000  18.100000  0.000000  0.624000  6.623500  94.075000  5.  
max   88.976200  100.000000 27.740000  1.000000  0.871000  8.780000  100.000000  12.
```

```
In [10]: df_y.describe()
```

```
Out[10]:      0  
count  506.000000  
mean   22.532806  
std    9.197104  
min    5.000000  
25%   17.025000  
50%   21.200000  
75%   25.000000  
max   50.000000
```

```
In [11]: df_x.shape
```

```
Out[11]: (506, 13)
```

```
In [12]: df_x.columns
```

```
Out[12]: Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',  
'PTRATIO', 'B', 'LSTAT'],  
dtype='object')
```

Initializing the linear regression Model

```
In [13]: reg = linear_model.LinearRegression()
```

Splitting Data into 70% training and 30% testing

```
In [14]: x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.3,
```

Train the model

```
In [15]: reg.fit(x_train, y_train)
```

```
Out[15]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
In [16]: #print the coefficient/ weights for each feature/column
```

```
# $f(x) = mx + c = y$  where m is the coefficient
```

```
print(reg.coef_)
```

```
[[ -1.33470103e-01  3.58089136e-02  4.95226452e-02  3.11983512e+00  
  -1.54170609e+01  4.05719923e+00  -1.08208352e-02  -1.38599824e+00  
   2.42727340e-01  -8.70223437e-03  -9.10685208e-01   1.17941159e-02  
  -5.47113313e-01]]
```

Prediction

```
In [17]: #print prediction for model
```

```
y_pred = reg.predict(x_test)  
print(y_pred)
```

```
[[28.64896005]  
 [36.49501384]  
 [15.4111932 ]  
 [25.40321303]  
 [18.85527988]  
 [23.14668944]  
 [17.3921241 ]  
 [14.07859899]  
 [23.03692679]  
 [20.59943345]  
 [24.82286159]  
 [18.53057049]  
 [-6.86543527]  
 [21.80172334]  
 [19.22571177]  
 [26.19191985]  
 [20.27733882]  
 [ 5.61596432]  
 [40.44887974]  
 ... ....]
```

```
In [18]: print(y_test)
```

```
    0  
173  23.6  
274  32.4  
491  13.6  
72   22.8  
452  16.1  
..   ..  
441  17.1  
23   14.5  
225  50.0  
433  14.3  
447  12.6
```

```
In [21]: #check the model accuracy using mean squared error  
print(np.mean((y_pred-y_test)**2))
```

```
0    21.517444  
dtype: float64
```

```
In [29]: from sklearn.metrics import mean_squared_error  
print(mean_squared_error(y_test,y_pred))
```

```
21.517444231176995
```

Assignment No 5 : Data Analytics 2

Importing Required Libraries

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

Read csv into dataframe

```
In [2]: df = pd.read_csv('Social_Network_Ads.csv')
```

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

```
Out[3]:
```

	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
..
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

Data preprocessing

```
In [5]: df.shape
```

```
Out[5]: (400, 5)
```

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

```
Out[6]: Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'], dtype='object')
```

```
In [7]: df.dtypes
```

```
Out[7]: User ID           int64
Gender            object
Age              int64
EstimatedSalary  int64
Purchased        int64
dtype: object
```

```
In [8]: 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 [9]: df.describe()
```

```
Out[9]:
```

	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 [10]: x = df.iloc[:, [2, 3]].values
```

```
In [11]: x
```

```
Out[11]:
```

```
array([[ 19, 19000],  
       [ 35, 20000],  
       [ 26, 43000],  
       [ 27, 57000],  
       [ 19, 76000],  
       [ 27, 58000],  
       [ 27, 84000],  
       [ 32, 150000],  
       [ 25, 33000],  
       [ 35, 65000],  
       [ 26, 80000],  
       [ 26, 52000],  
       [ 20, 86000],  
       [ 32, 18000],
```

```
In [12]: y = df.iloc[:, 4].values
```

In [13]: y

Splitting Data into 75% training and 25% testing data

```
In [14]: from sklearn.model_selection import train_test_split  
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, ran
```

```
In [15]: x_train
```

Out[15]:

```
[ 29, 83000],  
[ 26, 80000],  
[ 49, 28000],  
[ 23, 20000],  
[ 32, 18000],  
[ 60, 42000],  
[ 19, 76000],  
[ 36, 99000],  
[ 19, 26000],  
[ 60, 83000],  
[ 24, 89000],  
[ 27, 58000],  
[ 40, 47000],  
[ 42, 70000],  
[ 32, 150000],  
[ 35, 77000],  
[ 22, 63000],  
[ 45, 22000],  
[ 27, 89000],  
[ 18, 82000],  
[ 42, 79000],  
[ 40, 60000],  
[ 53, 34000],  
[ 47, 107000],  
[ 58, 144000],  
[ 59, 83000],  
[ 24, 55000],  
[ 74, 25000]
```

In [20]: `x_test.shape`

Out[20]: (100, 2)

In [21]: `y_test`

Out[21]: `array([0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1,`
`dtype=int64)`

In [22]: `y_test.shape`

Out[22]: (100,)

Data Standardization

In [23]: `from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
x_train = sc_x.fit_transform(x_train)
x_test = sc_x.fit_transform(x_test)`

In [24]: `x_train`

Out[24]:

```
array([[-0.54748976,  0.5130727 ],
       [ 0.15442019, -0.61825566],
       [-0.10879604,  0.14615539],
       [-0.54748976,  0.26846116],
       [-0.10879604, -0.61825566],
       [-0.81070599, -1.53554892],
       [-0.45975102, -1.68843113],
       [-0.0210573 ,  2.25592989],
       [-1.60035469, -0.0678797 ],
       [ 0.94406888, -0.83229075],
       [-0.54748976, -0.6488321 ],
       [-0.72296725, -0.46537345],
       [ 0.06668145, -0.46537345],
       [ 0.24215893,  0.20730828],
       [-1.4248772 ,  0.48249625],
       [-0.37201227,  1.43036596],
       [ 0.06668145,  0.20730828],
       [-1.51261594,  0.45191981],
       [ 1.64597884,  1.8278597 ],
       [-0.10879604, -1.47439603],
       [-0.10879604, -0.70998498],
       [ 0.94406888,  2.25592989],
       [ 0.41763642, -0.58767922],
       [ 0.94406888,  1.06344865],
       [-1.16166097, -1.29093738],
       [ 1.11954637,  2.16420057],
       [-0.72296725,  0.5130727 ],
       [-0.63522851,  0.2990376 ],
       [ 0.06668145, -0.25133835],
       [-0.37201227,  0.48249625],
       [-1.33713846,  0.54364914],
       [ 0.06668145,  0.26846116],
       [ 1.82145632, -0.31249124],
       [ 0.06668145, -0.52652633],
       [-1.07392223, -0.37364412],
       [-1.60035469, -0.55710277],
       [-1.24939971,  0.32961404],
       [-0.19653479, -0.83229075],
       [-0.45975102, -1.10747873],
       [ 1.11954637, -1.04632585],
       [-0.81070599,  0.54364914],
       [ 0.41763642, -0.55710277],
       [-0.81070599,  0.42134337],
       [-0.10879604, -1.53554892],
       [ 0.59311391,  1.27748375],
       [-0.81070599, -0.37364412],
       [ 0.06668145,  0.2990376 ],
       [ 1.3827626 ,  0.60480202],
       [-0.89844474, -1.2297845 ],
       [ 1.11954637,  0.48249625],
       [ 1.82145632,  1.58324817],
       [-0.19653479, -1.38266671],
       [-0.10879604, -0.40422056],
       [-0.19653479,  1.36921307],
       [ 1.99693381,  0.54364914],
       [ 0.7685914 , -1.16863161],
```

```
[-0.63522851,  0.39076693],  
[-0.89844474,  0.2990376 ],  
[ 1.11954637, -1.29093738],  
[-1.16166097, -1.53554892],  
[-0.37201227, -1.5967018 ],  
[ 2.08467255, -0.86286719],  
[-1.51261594,  0.17673183],  
[-0.0210573 ,  0.87999 ],
```

Initializing logistic Regression model and training the model

```
In [26]: from sklearn.linear_model import LogisticRegression  
classifier = LogisticRegression(random_state = 0)  
classifier.fit(x_train, y_train)
```

```
Out[26]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,  
                           intercept_scaling=1, l1_ratio=None, max_iter=100,  
                           multi_class='auto', n_jobs=None, penalty='l2',  
                           random_state=0, solver='lbfgs', tol=0.0001, verbose=0,  
                           warm_start=False)
```

Prediction

```
In [29]: y_pred = classifier.predict(x_test)
```

```
In [30]: y_pred
```

```
Out[30]: array([0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,  
               0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,  
               1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,  
               0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,  
               0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1], dtype=int64)
```

```
In [31]: y_test
```

```
Out[31]: array([0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,  
               0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,  
               1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,  
               0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1,  
               1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1], dtype=int64)
```

Confusion Matrix

```
In [32]: from sklearn.metrics import confusion_matrix  
cm = confusion_matrix(y_test, y_pred)  
cm
```

```
Out[32]: array([[63,  5],  
                 [ 8, 24]], dtype=int64)
```

```
In [ ]: #The confusion matrix tells us that there were 87 correct predictions and 13 i
```

Calculating accuracy of model

```
In [34]: from sklearn.metrics import accuracy_score  
accuracy = accuracy_score(y_test, y_pred) * 100  
accuracy
```

```
Out[34]: 87.0
```

True Positive : Actually True, Predicted True

```
In [48]: tp = cm[0,[0]]  
print('True Positive : ',tp)  
  
True Positive : [63]
```

False Positive : Actually False, Predicted True

```
In [49]: fp = cm[0, [1]]  
print('False Positive : ',fp)  
  
False Positive : [5]
```

False Negative : Actually True, Predicted False

```
In [50]: fn = cm[1, [0]]  
print('False Negative : ',fn)  
  
False Negative : [8]
```

True Negative : Actually False, Predicted False

```
In [51]: tn = cm[1, [1]]  
print('True Negative : ',tn)  
  
True Negative : [24]
```

Accuracy

```
In [47]: accuracy_cm = ((tp + tn)/(tp + fp + fn + tn))  
print('Accuracy : ',accuracy_cm*100)  
  
Accuracy : [87.]
```

Error Rate

```
In [54]: error_rate_cm = ((fp + fn)/(tp + fp + fn + tn))  
print('Error Rate : ', error_rate_cm*100)  
  
Error Rate : [13.]
```

Precision

```
In [56]: precision_cm = (tp / (tp + fp))
print('Precision : ',precision_cm*100)

Precision : [92.64705882]
```

Recall

```
In [59]: #recall is also called sensitivity
recall_cm = (tp / (tp + fn))
print('Sensitivity : ', recall_cm*100)

Sensitivity : [88.73239437]
```

Specificity

```
In [62]: specificity_cm = (tn / (tn + fp))
print('Specificity : ', specificity_cm*100)

Specificity : [82.75862069]
```

Assignment No 6 : Data Analytics 3

Importing Required Libraries

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

Reading csv into dataframe

```
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	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

Data preprocessing

```
In [4]: df.shape
```

```
Out[4]: (150, 6)
```

```
In [5]: df.columns  
Out[5]: Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',  
               'Species'],  
              dtype='object')
```

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

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

```
In [7]: 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
```

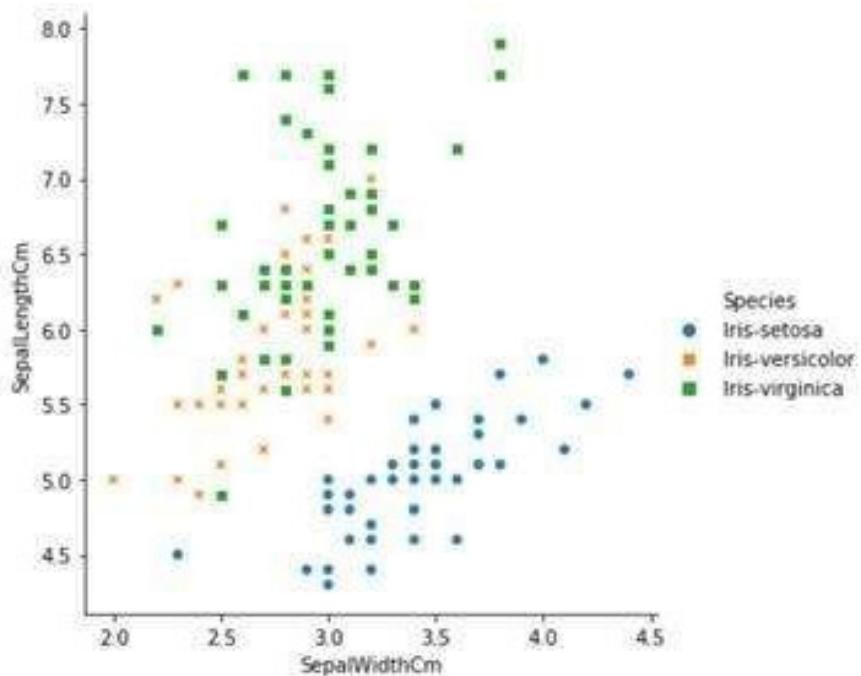
```
In [8]: df.describe()
```

```
Out[8]:      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  
count  150.000000  150.000000  150.000000  150.000000  150.000000  
mean   75.500000  5.843333  3.054000  3.758667  1.198667  
std    43.445368  0.828066  0.433594  1.764420  0.763161  
min    1.000000  4.300000  2.000000  1.000000  0.100000  
25%   38.250000  5.100000  2.800000  1.600000  0.300000  
50%   75.500000  5.800000  3.000000  4.350000  1.300000  
75%   112.750000 6.400000  3.300000  5.100000  1.800000  
max   150.000000  7.900000  4.400000  6.900000  2.500000
```

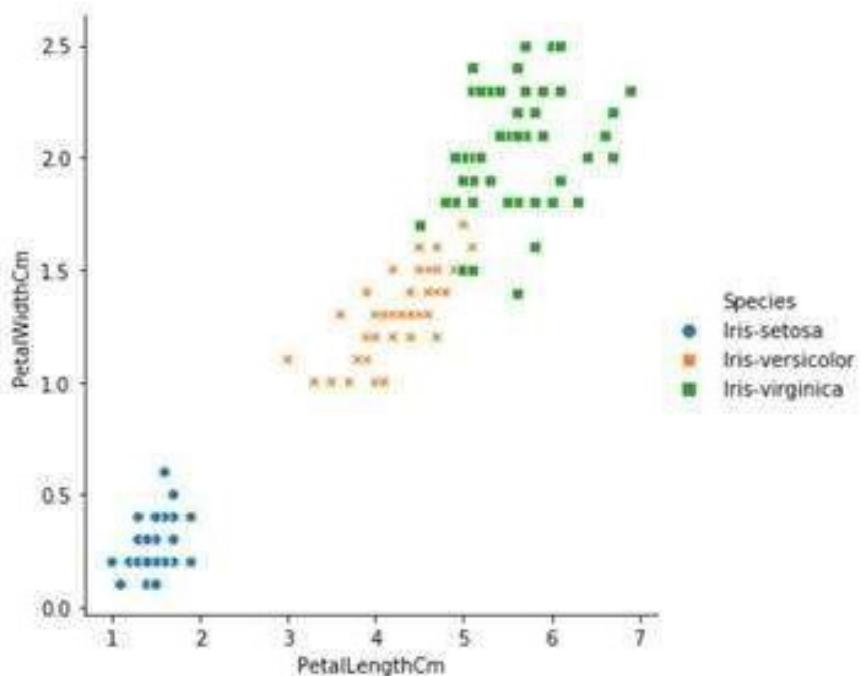
```
In [9]: df.isnull()
```

```
Out[9]:      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species  
0   False        False        False        False        False        False
```

```
In [12]: sns.relplot(data = df, x = 'SepalWidthCm', y = 'SepalLengthCm', hue = 'Species')
Out[12]: <seaborn.axisgrid.FacetGrid at 0x26edcc3cc8>
```

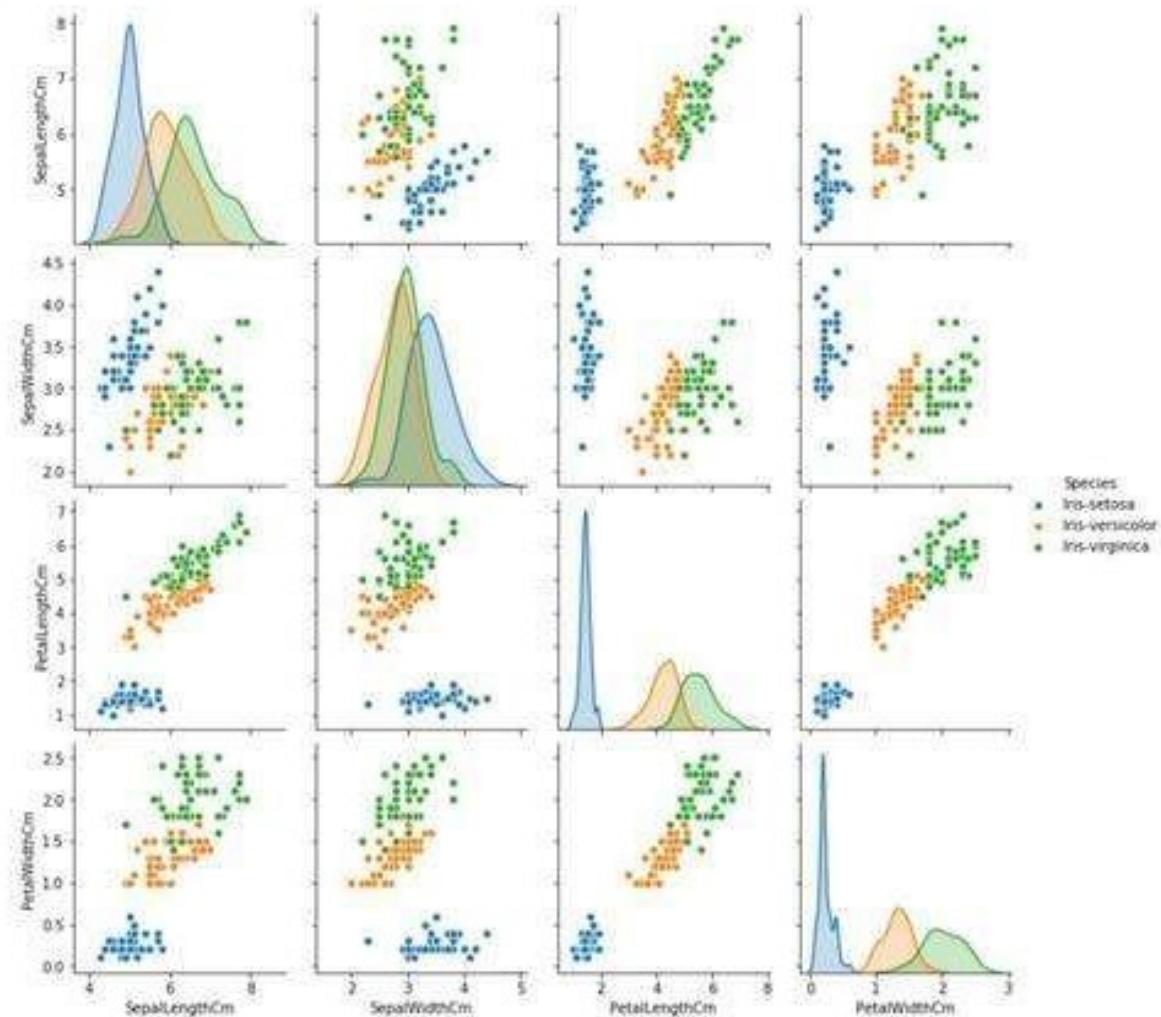


```
In [13]: sns.relplot(data = df, x = 'PetalLengthCm', y = 'PetalWidthCm', hue = 'Species')
Out[13]: <seaborn.axisgrid.FacetGrid at 0x26f0e53ac8>
```



```
In [14]: sns.pairplot(df, hue = 'Species')
```

```
Out[14]: <seaborn.axisgrid.PairGrid at 0x26f0ec9248>
```



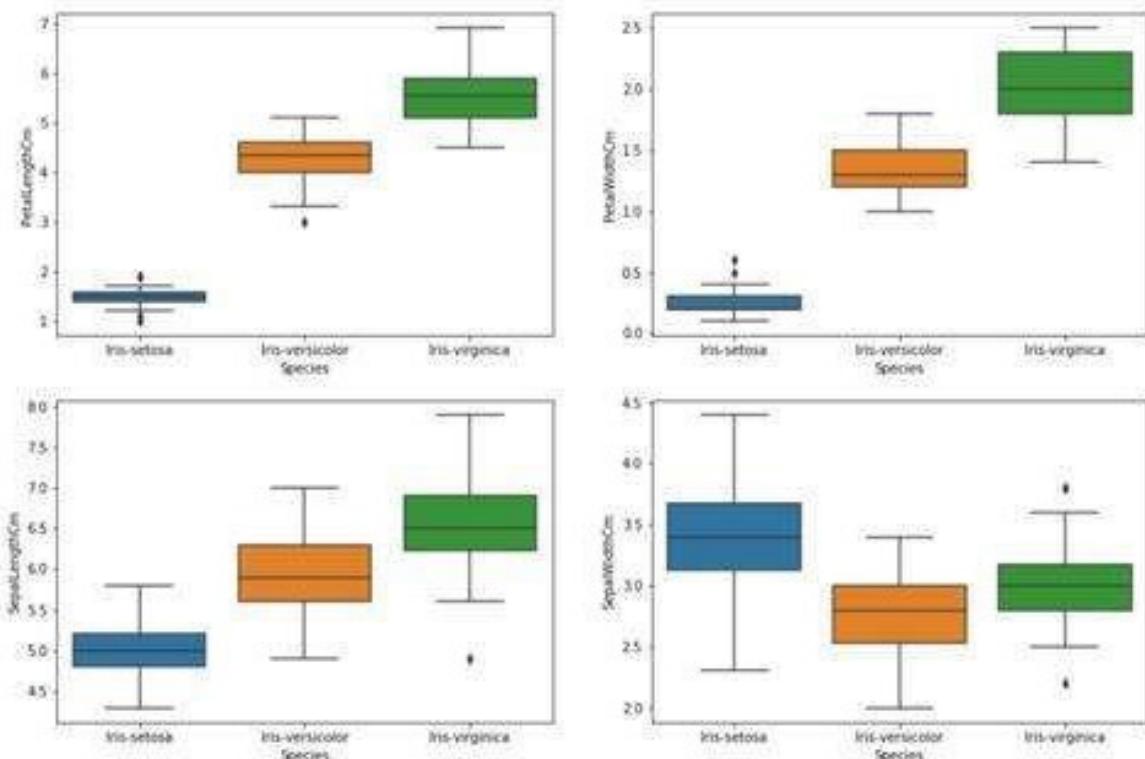
```
In [15]: plt.figure(figsize = (15, 10))
plt.subplot(2, 2, 1)
sns.boxplot(data = df, x = 'Species', y = 'PetalLengthCm')

plt.subplot(2, 2, 2)
sns.boxplot(data = df, x = 'Species', y = 'PetalWidthCm')

plt.subplot(2, 2, 3)
sns.boxplot(data = df, x = 'Species', y = 'SepalLengthCm')

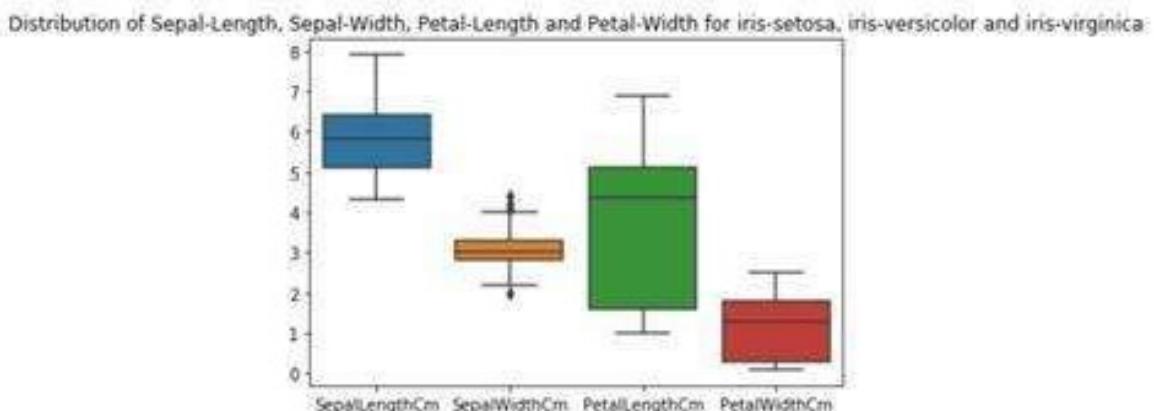
plt.subplot(2, 2, 4)
sns.boxplot(data = df, x = 'Species', y = 'SepalWidthCm')
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x26ef631688>



```
In [16]: sns.boxplot(data = df).set_title("Distribution of Sepal-Length, Sepal-Width, Petal-Length and Petal-Width for iris-setosa, iris-versicolor and iris-virginica")
```

```
Out[16]: Text(0.5, 1.0, 'Distribution of Sepal-Length, Sepal-Width, Petal-Length and Petal-Width for iris-setosa, iris-versicolor and iris-virginica')
```



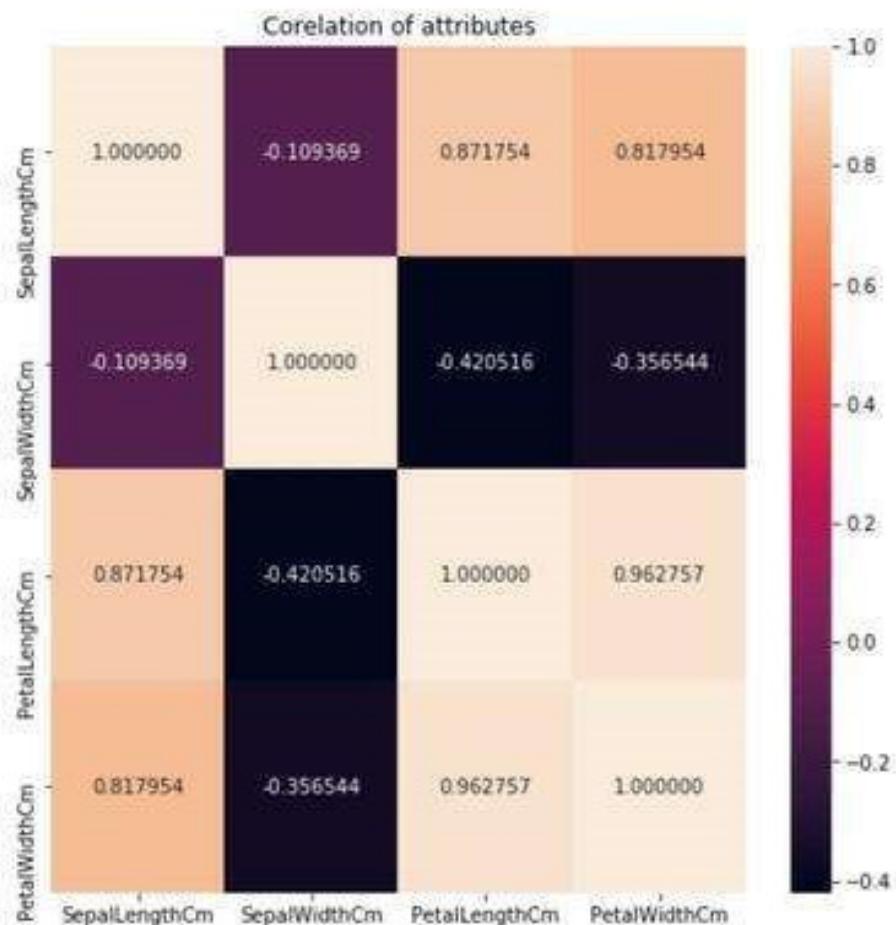
```
In [17]: df.corr()
```

```
Out[17]:
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

```
In [18]: plt.subplots(figsize = (8,8))
sns.heatmap(df.corr(), annot = True, fmt = "f").set_title("Corelation of attributes")
```

```
Out[18]: Text(0.5, 1, 'Corelation of attributes')
```



```
In [19]: x = df.iloc[:, 0:4].values
x
```

```
Out[19]:
```

ANSWER SHEET 4 5 6 4 4 6 31

```
In [20]: y = df.iloc[:, 4].values  
y
```

Label Encoding

```
In [21]: from sklearn.preprocessing import LabelEncoder  
le = LabelEncoder()
```

```
y = le.fit_transform(y)
```

Splitting data into 70% training and 30% testing

```
In [22]: from sklearn.model_selection import train_test_split  
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, ran
```

```
In [23]: x_train
```

Out[23]:

```
[5.4, 3.9, 1.7, 0.4],  
[5.5, 2.3, 4., 1.3],  
[6.8, 3.2, 5.9, 2.3],  
[7.6, 3., 6.6, 2.1],  
[5.1, 3.5, 1.4, 0.2],  
[4.9, 3.1, 1.5, 0.1],  
[5.2, 3.4, 1.4, 0.2],  
[5.7, 2.8, 4.5, 1.3],  
[6.6, 3., 4.4, 1.4],  
[5., 3.2, 1.2, 0.2],  
[5.1, 3.3, 1.7, 0.5],  
[5.4, 2.9, 1.3, 0.3]
```

```
In [24]: x_train.shape
```

Out[24]: (105, 4)

```
In [25]: x test
```

Out[25]:

```
array([[5.8, 2.8, 5.1, 2.4],  
       [6., 2.2, 4., 1.],  
       [5.5, 4.2, 1.4, 0.2],  
       [7.3, 2.9, 6.3, 1.8],  
       [5., 3.4, 1.5, 0.2],  
       [6.3, 3.3, 6., 2.5],  
       [5., 3.5, 1.3, 0.3],
```

```
In [26]: x_test.shape
```

```
Out[26]: (45, 4)
```

```
In [27]: y_train
```

```
Out[27]: array([1, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 1, 2, 1, 0, 2, 1, 1, 1, 1,  
                2, 0, 0, 2, 1, 0, 0, 1, 0, 2, 1, 0, 1, 2, 1, 0, 2, 2, 2, 2, 0, 0,  
                2, 2, 0, 2, 0, 2, 2, 0, 0, 2, 0, 0, 0, 1, 2, 2, 0, 0, 0, 1, 1, 0,  
                0, 1, 0, 2, 1, 2, 1, 0, 2, 0, 2, 0, 0, 2, 0, 2, 1, 1, 1, 2, 2, 1,  
                1, 0, 1, 2, 2, 0, 1, 1, 1, 0, 0, 2, 1, 2, 0])
```

```
In [28]: y_train.shape
```

```
Out[28]: (105,)
```

```
In [29]: y_test
```

```
Out[29]: array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,  
                0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 0, 2, 2, 1, 0, 1, 1, 1, 2, 0, 2, 0,  
                0])
```

```
In [30]: y_test.shape
```

```
Out[30]: (45,)
```

Initializing Naive Bayes Model and training it

```
In [31]: from sklearn.naive_bayes import GaussianNB  
model = GaussianNB()  
model.fit(x_train, y_train)
```

```
Out[31]: GaussianNB(priors=None, var_smoothing=1e-09)
```

Prediction

```
In [32]: prediction = model.predict(x_test)
```

```
In [33]: prediction
```

```
Out[33]: array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,  
                0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 0, 2, 2, 1, 0, 1, 1, 1, 2, 0, 2, 0,  
                0])
```

Assignment No 7

```
In [14]: from textblob import TextBlob  
import nltk
```

```
In [4]: b = TextBlob("I havv good spelling")
```

```
In [5]: b.correct()
```

```
Out[5]: TextBlob("I have good spelling")
```

```
In [22]: import nltk  
nltk.download('punkt')  
b1 = TextBlob("Beautiful is better than ugly."  
             "Explicit is better than implicit."  
             "Simple is better than complex.")  
b1.words
```

```
[nltk_data] Downloading package punkt to  
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...  
[nltk_data] Package punkt is already up-to-date!
```

```
Out[22]: WordList(['Beautiful', 'is', 'better', 'than', 'ugly.Explicit', 'is', 'better', 'than', 'implicit.Simple', 'is', 'better', 'than', 'complex'])
```

```
In [23]: b1.sentences
```

```
Out[23]: [Sentence("Beautiful is better than ugly.Explicit is better than implicit.Simple is better than complex.")]
```

```
In [24]: sentence = TextBlob("Use 4 spaces per indentation level")  
sentence.words
```

```
Out[24]: WordList(['Use', '4', 'spaces', 'per', 'indentation', 'level'])
```

```
In [27]: sentence.words[2].singularize()
```

```
Out[27]: 'space'
```

```
In [28]: sentence.words[5].pluralize()
```

```
Out[28]: 'levels'
```

```
In [30]: animals = TextBlob("cat dog octopus")  
animals.words
```

```
Out[30]: WordList(['cat', 'dog', 'octopus'])
```

```
In [31]: animals.words.pluralize()
```

```
Out[31]: WordList(['cats', 'dogs', 'octopodes'])
```

```
In [32]: sen = TextBlob("We are no longer the knights who say Ni." "We are now the knig  
sen.word_counts['ekki']  
Out[32]: 3  
  
In [33]: sen.words.count('ekki')  
Out[33]: 3  
  
In [34]: sen.words.count('ekki', case_sensitive=True)  
Out[34]: 2  
  
In [35]: b = TextBlob("And now for something completely different.")  
print(b.parse())  
  
And/CC/O/O now/RB/B-ADVP/O for/IN/B-PP/B-PNP something/NN/B-NP/I-PNP complete  
ly/RB/B-ADJP/O different/JJ/I-ADJP/O ././O/O  
  
In [36]: b1[0:19]  
TextBlob("Beautiful is better")  
Out[36]: TextBlob("Beautiful is better")  
  
In [37]: b1.upper()  
Out[37]: TextBlob("BEAUTIFUL IS BETTER THAN UGLY.EXPLICIT IS BETTER THAN IMPLICIT.SIMP  
LE IS BETTER THAN COMPLEX.")  
  
In [39]: b1.find("Simple")  
Out[39]: 63  
  
In [40]: apple_blob = TextBlob('apples')  
banana_blob = TextBlob('bananas')  
apple_blob < banana_blob  
Out[40]: True  
  
In [41]: blob = TextBlob("Now is better then never.")  
blob.ngrams(n=3)  
Out[41]: [WordList(['Now', 'is', 'better']),  
WordList(['is', 'better', 'then']),  
WordList(['better', 'then', 'never'])]
```

```
In [42]: import nltk
from nltk import tokenize
from nltk.tokenize import sent_tokenize
text = """ Good Day everyone, how are you all today? Its fun learning data ana
text

Out[42]: ' Good Day everyone, how are you all today? Its fun learning data analysis. H
ope you all are practicing well.'

In [43]: tokenized_text = sent_tokenize(text)

In [45]: print(tokenized_text)

[' Good Day everyone, how are you all today?', 'Its fun learning data analysi
s.', 'Hope you all are practicing well.']

In [47]: from nltk.tokenize import word_tokenize
tokenizer_word = word_tokenize(text)
print(tokenizer_word)

['Good', 'Day', 'everyone', ',', 'how', 'are', 'you', 'all', 'today', '?', 'I
ts', 'fun', 'learning', 'data', 'analysis', '.', 'Hope', 'you', 'all', 'are',
'practicing', 'well', '.']

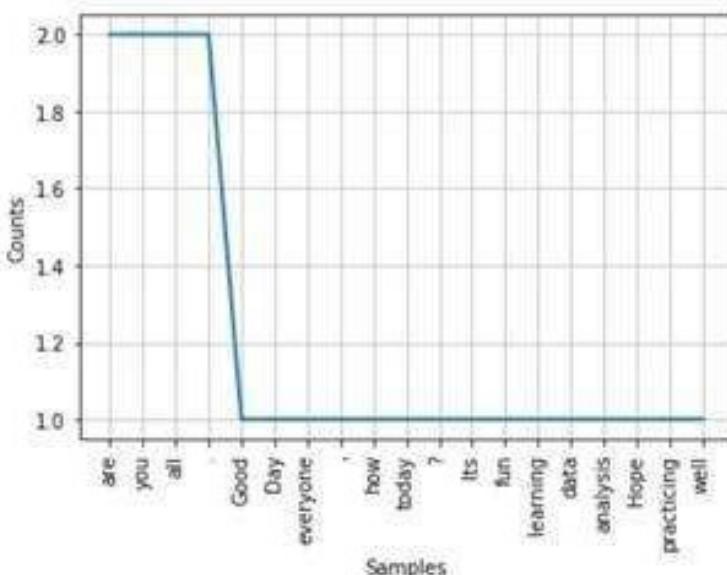
In [48]: from nltk.probability import FreqDist
fdist = FreqDist(tokenizer_word)
print(fdist)

<FreqDist with 19 samples and 23 outcomes>

In [49]: fdist.most_common(4)

Out[49]: [('are', 2), ('you', 2), ('all', 2), ('.', 2)]
```

```
In [50]: import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)
plt.show()
```



```
In [52]: nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]      C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]  Unzipping corpora\stopwords.zip.
```

```
Out[52]: True
```

```
In [57]: from nltk.corpus import stopwords
stop_words = set(stopwords.words("english"))
print(stop_words)
```

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

```
In [60]: filtered_sent=[]
for w in tokenizer_word:
    if w not in stop_words:
        filtered_sent.append(w)

print("Tokenized sentence : ",tokenizer_word)
print("Filtered sentence : ", filtered_sent)

Tokenized sentence : ['Good', 'Day', 'everyone', ',', 'how', 'are', 'you', 'all', 'today', '?', 'Its', 'fun', 'learning', 'data', 'analysis', '.', 'Hope', 'you', 'all', 'are', 'practicing', 'well', '.']
Filtered sentence : ['Good', 'Day', 'everyone', ',', 'today', '?', 'Its', 'fun', 'learning', 'data', 'analysis', '.', 'Hope', 'practicing', 'well', '.']
```

```
In [61]: from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
ps = PorterStemmer()
stemmed_words = []
for w in filtered_sent:
    stemmed_words.append(ps.stem(w))
print("Filtered Sentence : ", filtered_sentence)
print("Stemmed Sentence : ", stemmed_words)

Filtered Sentence : ['Good', 'Day', 'everyone', ',', 'today', '?', 'Its', 'fun', 'learning', 'data', 'analysis', '.', 'Hope', 'practicing', 'well', '.']
Stemmed Sentence : ['good', 'day', 'everyon', ',', 'today', '?', 'it', 'fun', 'learn', 'data', 'analysi', '.', 'hope', 'practic', 'well', '.']
```

```
In [62]: nltk.download('wordnet')

[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]     Unzipping corpora\wordnet.zip.
```

```
Out[62]: True
```

```
In [64]: from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()
from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()
word = "flying"
print("Lemmatizer Word : ", lem.lemmatize(word, "v"))
print("Stemmed Word : ", stem.stem(word))

Lemmatizer Word : fly
Stemmed Word : fli
```

```
In [65]: sent = "Albert Einstein was born in Ulm, Germany in 1879."
tokens = nltk.word_tokenize(sent)
print(tokens)

['Albert', 'Einstein', 'was', 'born', 'in', 'Ulm', ',', 'Germany', 'in', '1879', '.']
```

```
In [66]: nltk.download('averaged_perceptron_tagger')

[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]     Unzipping taggers\averaged_perceptron_tagger.zip.
```

```
Out[66]: True
```

```
In [67]: nltk.pos_tag(tokens)
```

```
Out[67]: [('Albert', 'NNP'),
           ('Einstien', 'NNP'),
           ('was', 'VBD'),
           ('born', 'VBN'),
           ('in', 'IN'),
           ('Ulm', 'NNP'),
           ('.', '.'),
           ('Germany', 'NNP'),
           ('in', 'IN'),
           ('1879', 'CD'),
           ('..', '..')]
```

```
In [68]: from collections import Counter
sentence = "Texas A&M University is located in Texas"
term_frequency = Counter(sentence.split())
```

```
In [69]: term_frequency
```

```
Out[69]: Counter({'Texas': 2,
                  'A&M': 1,
                  'University': 1,
                  'is': 1,
                  'located': 1,
                  'in': 1})
```

Assignment No 8 : Data Visualization 1

Import Required Libraries

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

Reading csv into data frame

```
In [2]: df = pd.read_csv('train.csv')  
df
```

Out[2]:

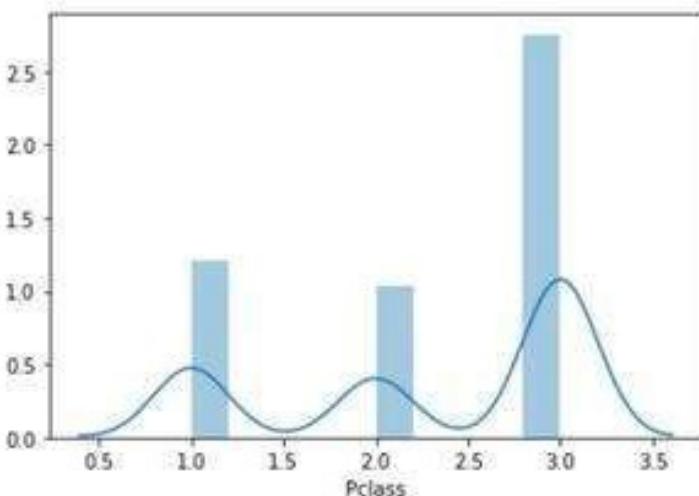
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2- 3101282	7.9250
3	4	1	1	Futrelle, 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
...
886	887	0	2	Montville, Rev. Juozas	male	27.0	0	0	211536	13.0000
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W/C- 6807	23.4500
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500

Data Visualization

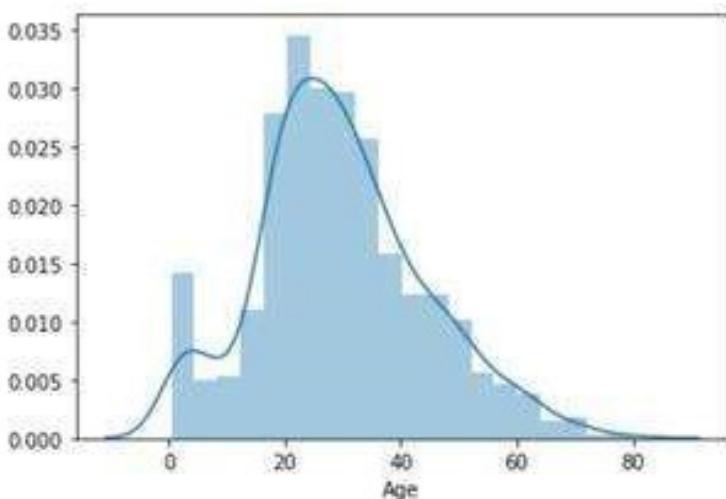
```
In [3]: import seaborn as sns
sns.distplot(df['Pclass'])
```

```
Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x4f90eff688>
```



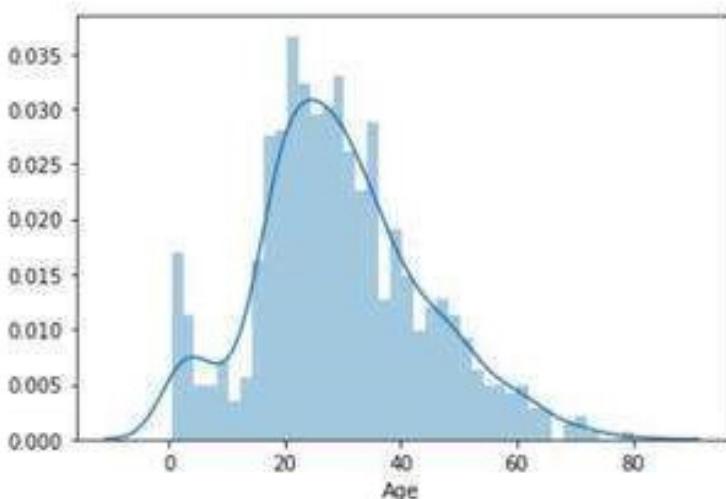
```
In [4]: sns.distplot(df['Age'])
```

```
Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x4f914a8b08>
```



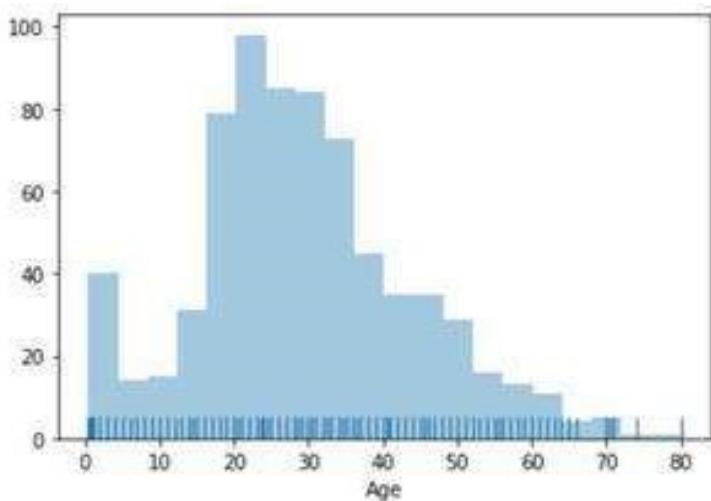
```
In [5]: sns.distplot(df['Age'], bins=40)
```

```
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x4f91918cc8>
```



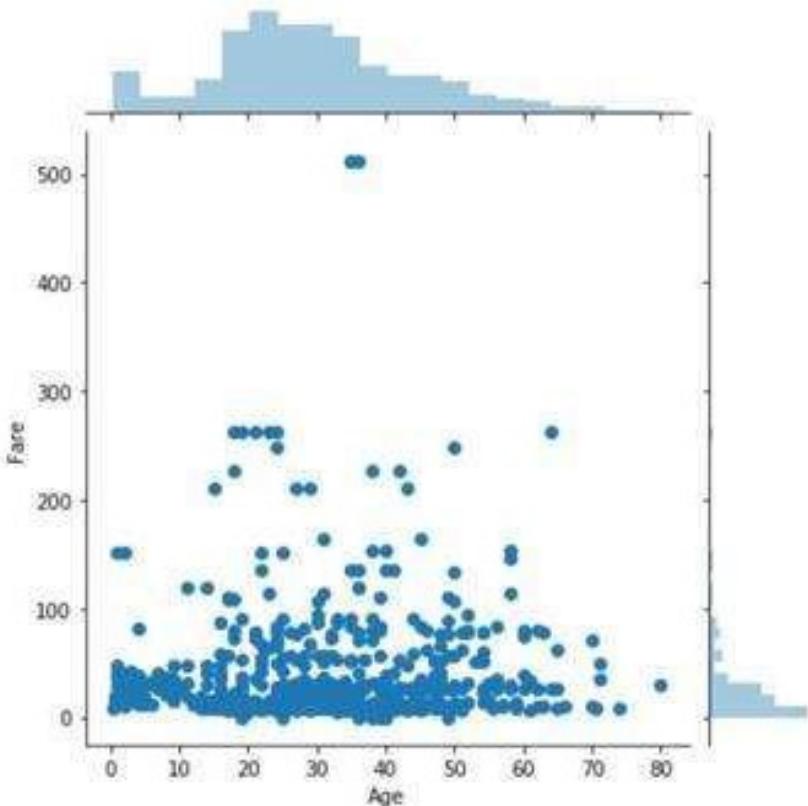
```
In [6]: sns.distplot(df['Age'], bins=20, kde=False, rug=True)
```

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x4f914b4b88>
```



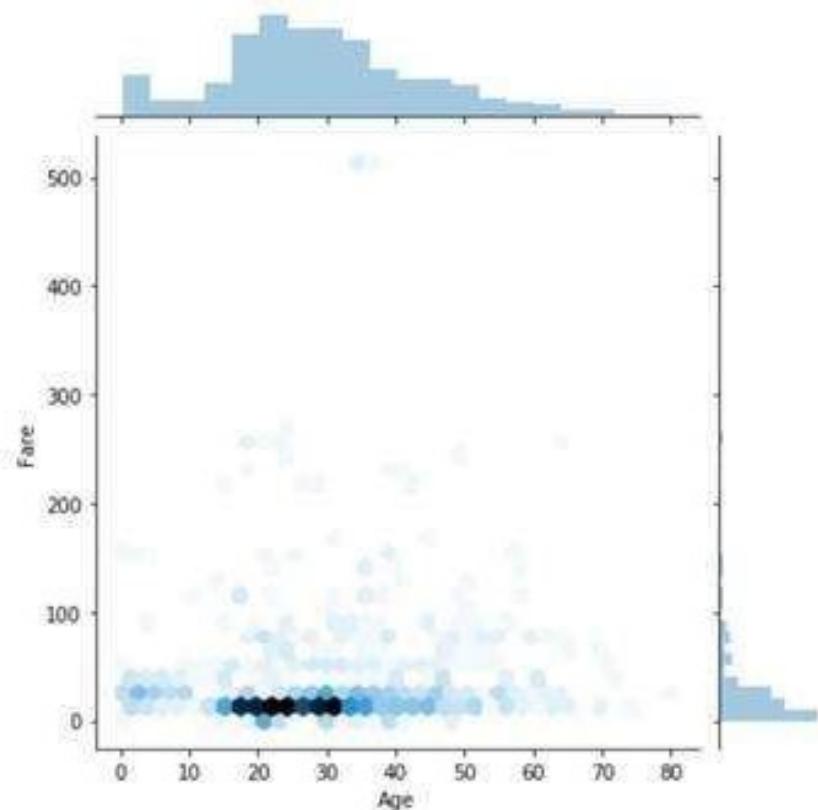
```
In [7]: sns.jointplot(x = df['Age'], y = df['Fare'], kind = 'scatter')
```

```
Out[7]: <seaborn.axisgrid.JointGrid at 0x4f91b27388>
```



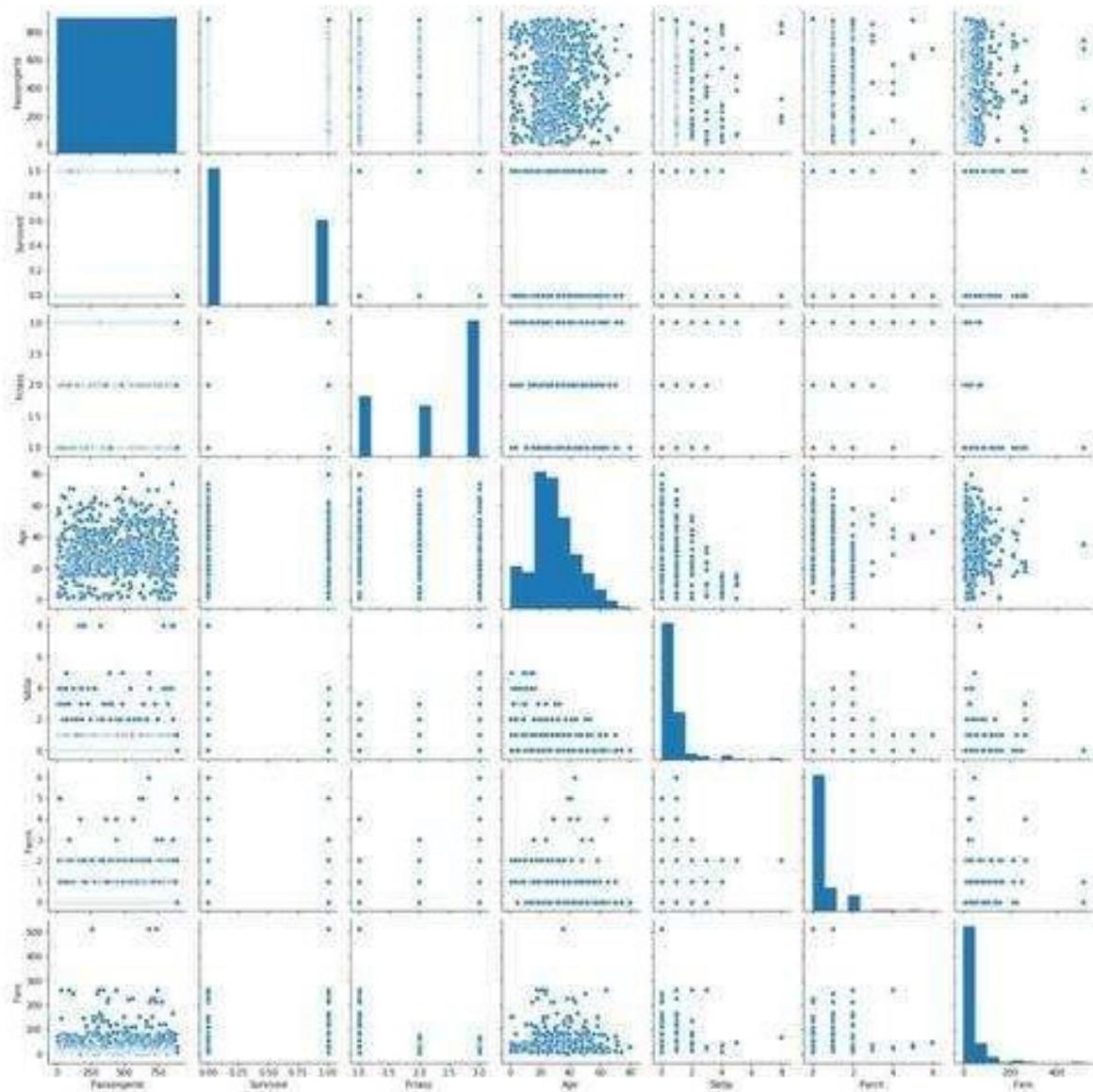
```
In [8]: sns.jointplot(x = df['Age'], y = df['Fare'], kind = 'hex')
```

```
Out[8]: <seaborn.axisgrid.JointGrid at 0x4f91cca788>
```



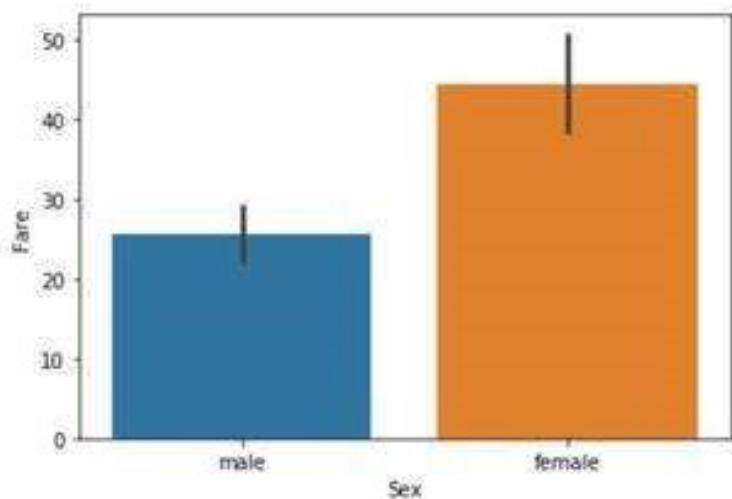
```
In [9]: sns.pairplot(df)
```

```
Out[9]: <seaborn.axisgrid.PairGrid at 0x4f91ccff88>
```



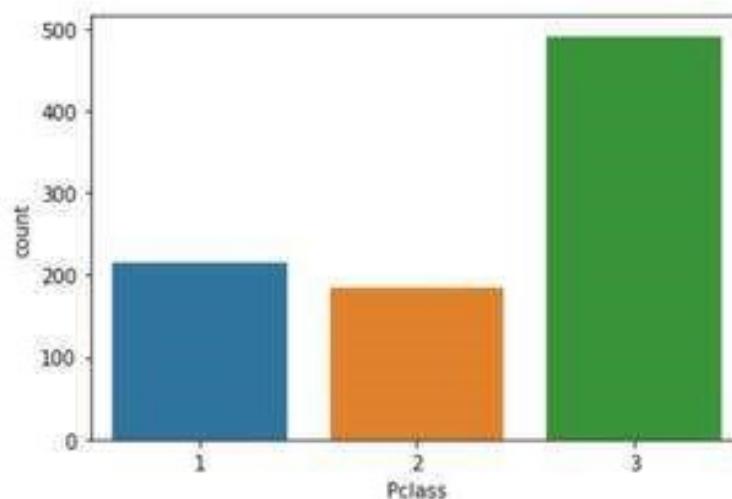
```
In [10]: sns.barplot(x = df['Sex'], y = df['Fare'])
```

```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x4f94808a48>
```



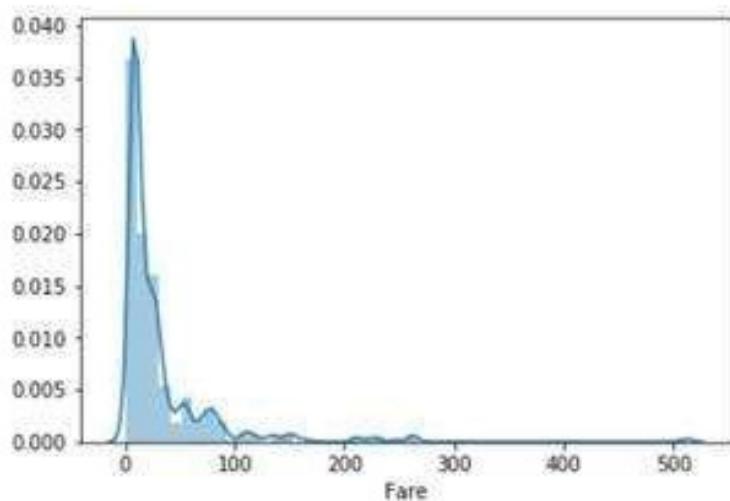
```
In [11]: sns.countplot(df['Pclass'])
```

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x4f91514648>
```



```
In [24]: sns.distplot(df['Fare'], hist = True)
```

```
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x4f955c6a48>
```



Assignment No 9 : Data Visualization 2

Import required Libraries

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

Read csv into dataframe

```
In [2]: df = pd.read_csv('train.csv')
```

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

	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
...
886	887	0	2	Montville, Rev. Juozas	male	27.0	0	0	211536	13.0000

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W/C- 6607	23.4500
889	890	1	1	Behr, Mr. Karl Howell Dooley.	male	26.0	0	0	111369	30.0000

Data preprocessing

In [4]: `df.shape`

Out[4]: (891, 12)

In [5]: `df.columns`

Out[5]: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
 dtype='object')

In [6]: `df.dtypes`

Out[6]: PassengerId int64
 Survived int64
 Pclass int64
 Name object
 Sex object
 Age float64
 SibSp int64
 Parch int64
 Ticket object
 Fare float64
 Cabin object
 Embarked object
 dtype: object

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

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

In [8]: df.isnull()

```
Out[8]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	False	True
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	True
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	True
..
886	False	False	False	False	False	False	False	False	False	False	False	True
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	False	True	False	False	False	False	True
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	False	True

891 rows × 12 columns

In [9]: df.isnull().sum()

```
Out[9]:
```

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 [10]: df.describe()

```
Out[10]:
```

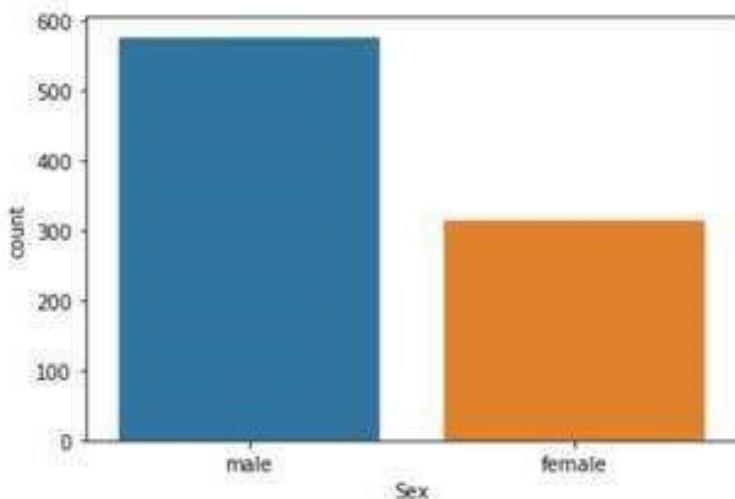
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
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

Data Visualization

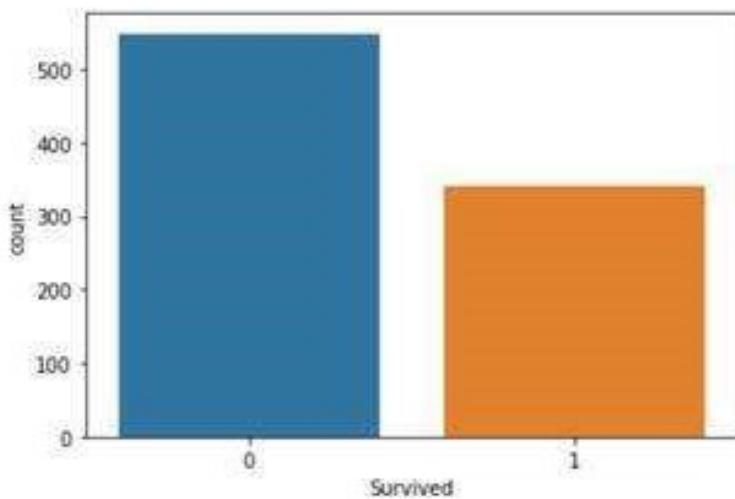
```
In [12]: sns.countplot(df['Sex'])
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x8345203508>
```



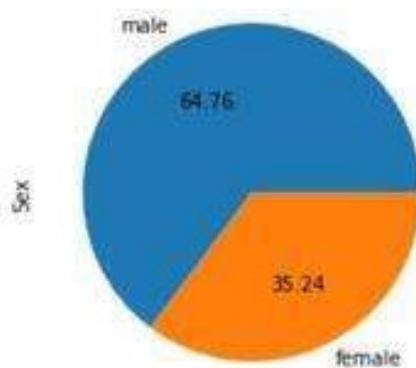
```
In [13]: sns.countplot(df['Survived'])
```

```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x8345288b88>
```



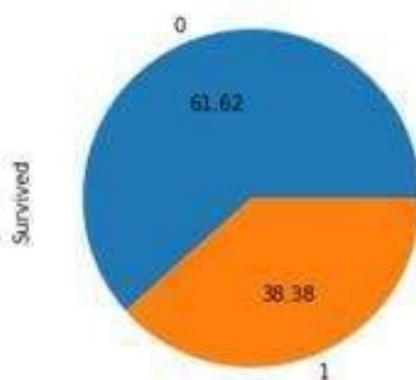
```
In [15]: df['Sex'].value_counts().plot(kind = 'pie', autopct = '%.2f')
```

```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x8345593d08>
```



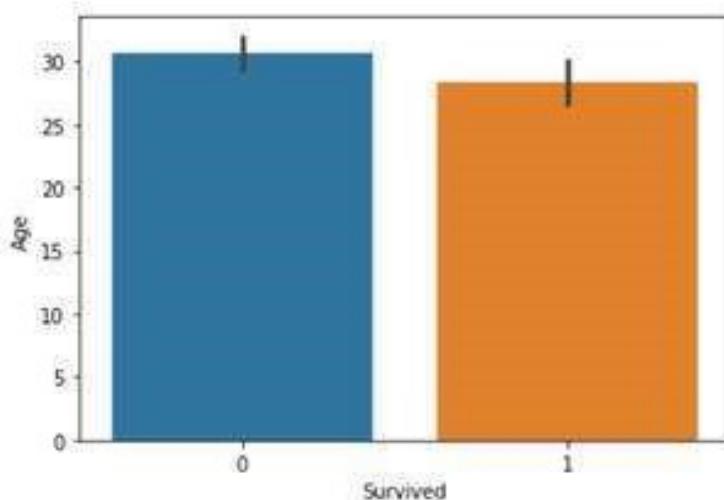
```
In [16]: df['Survived'].value_counts().plot(kind = 'pie', autopct = '%.2f')
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x83455b8ec8>
```



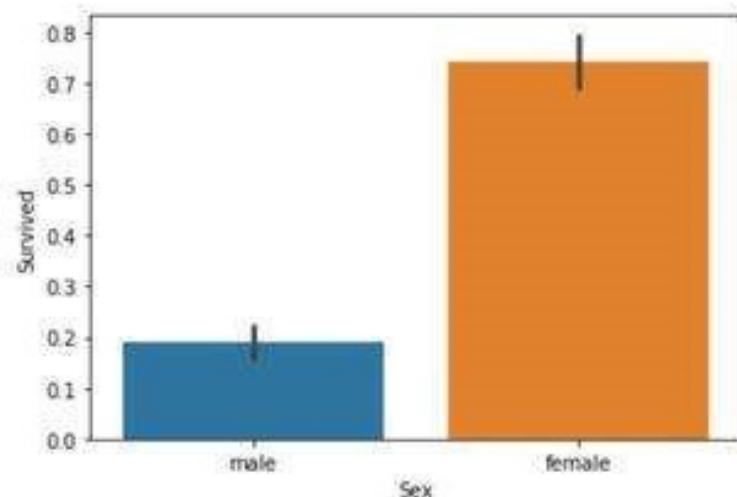
```
In [17]: sns.barplot(df['Survived'], df['Age'])
```

```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x8345bd2788>
```



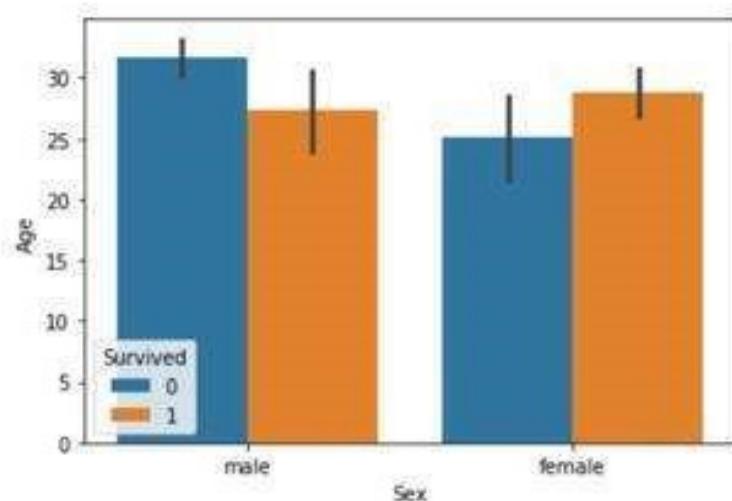
```
In [19]: sns.barplot(df['Sex'], df['Survived'])
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x8345bcda08>
```



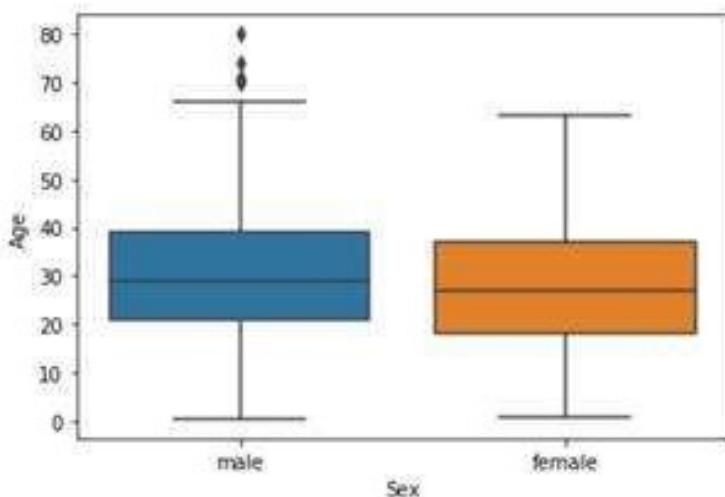
```
In [20]: sns.barplot(df['Sex'],df['Age'], hue = df['Survived'])
```

```
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x8345cf55c8>
```



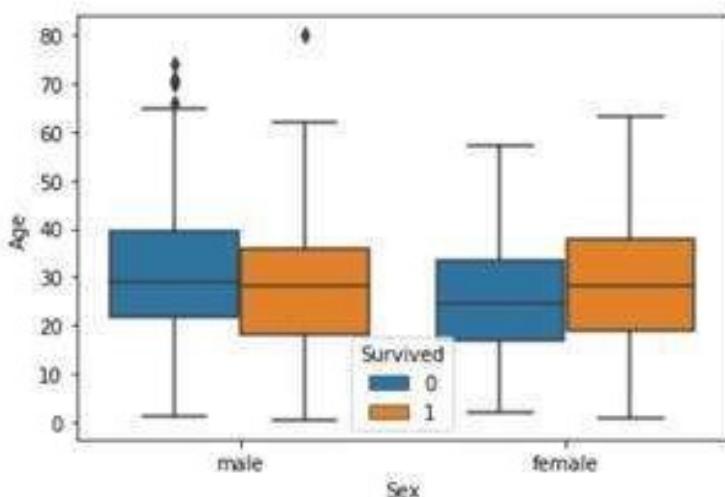
```
In [21]: sns.boxplot(df['Sex'], df['Age'])
```

```
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x8345d66c48>
```



```
In [4]: sns.boxplot(df['Sex'], df['Age'], df['Survived'])
```

```
Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x44b8ecf788>
```



```
In [24]: pd.crosstab(df['Sex'], df['Survived'])
```

```
Out[24]: Survived   0   1
```

	Sex	
Survived	female	male
0	81	468
1	233	109

```
In [25]: pd.crosstab(df['Age'], df['Survived'])
```

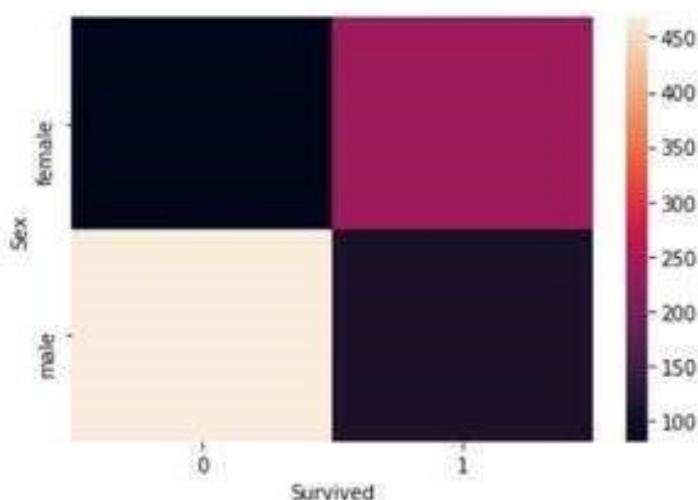
```
Out[25]: Survived   0   1
```

Age

Survived	0	1
Age		
0.42	0	1
0.67	0	1
0.75	0	2
0.83	0	2
0.92	0	1
...
70.00	2	0
70.50	1	0
71.00	2	0
74.00	1	0

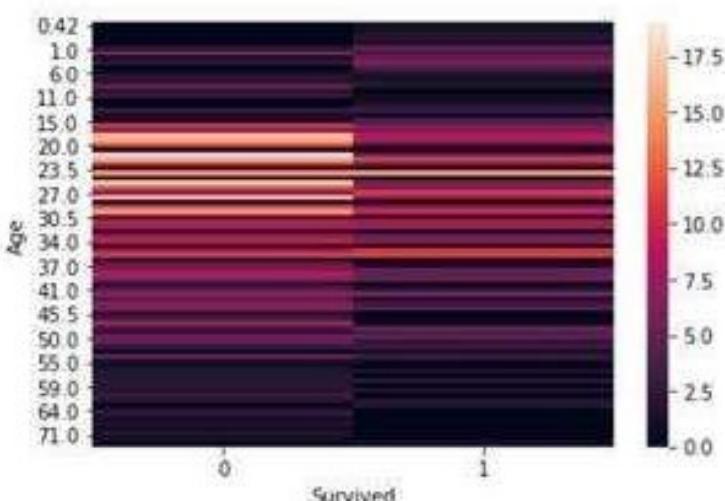
```
In [26]: sns.heatmap(pd.crosstab(df['Sex'], df['Survived']))
```

```
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x8345eaa848>
```



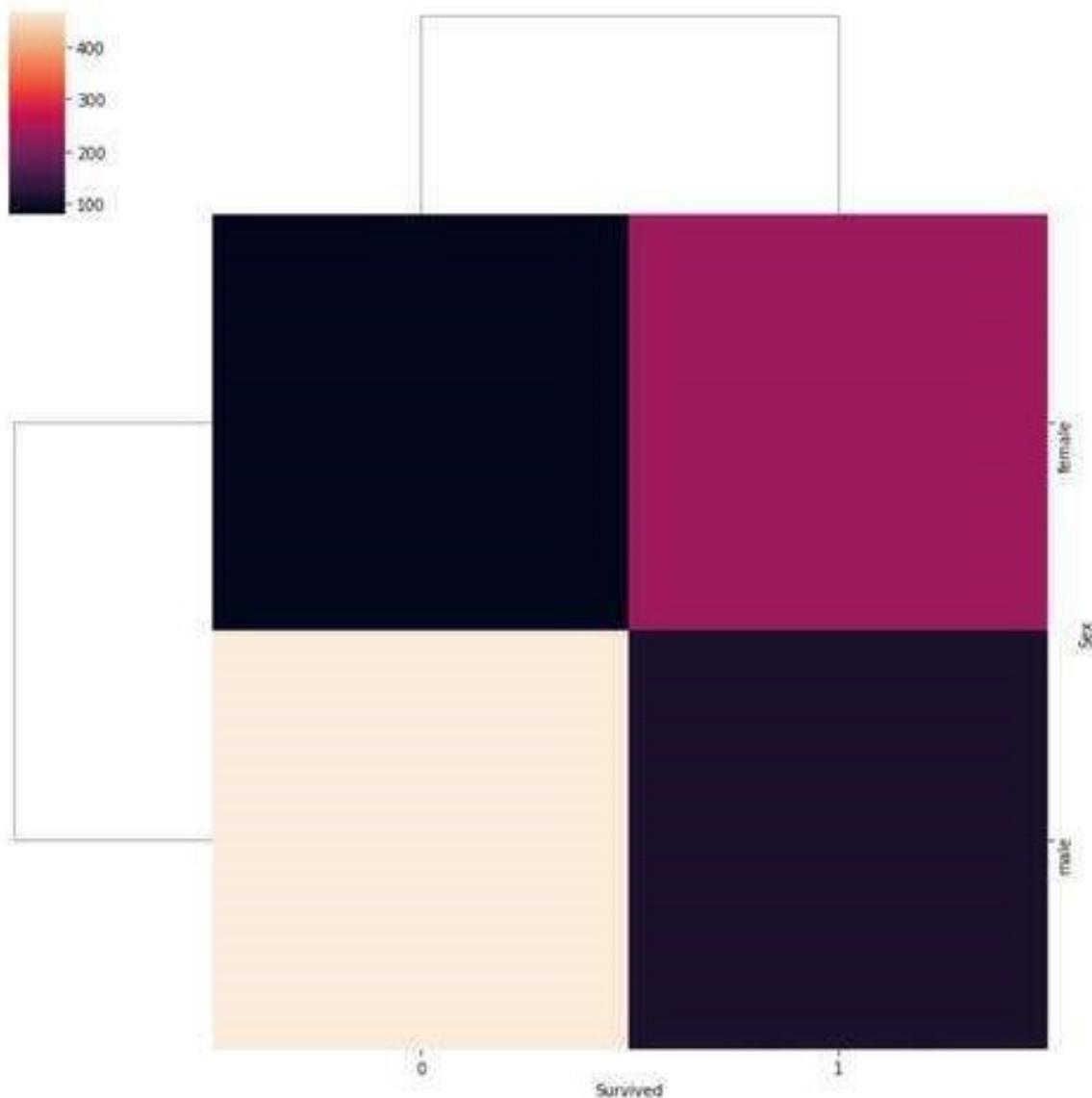
```
In [27]: sns.heatmap(pd.crosstab(df['Age'], df['Survived']))
```

```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x8346973308>
```



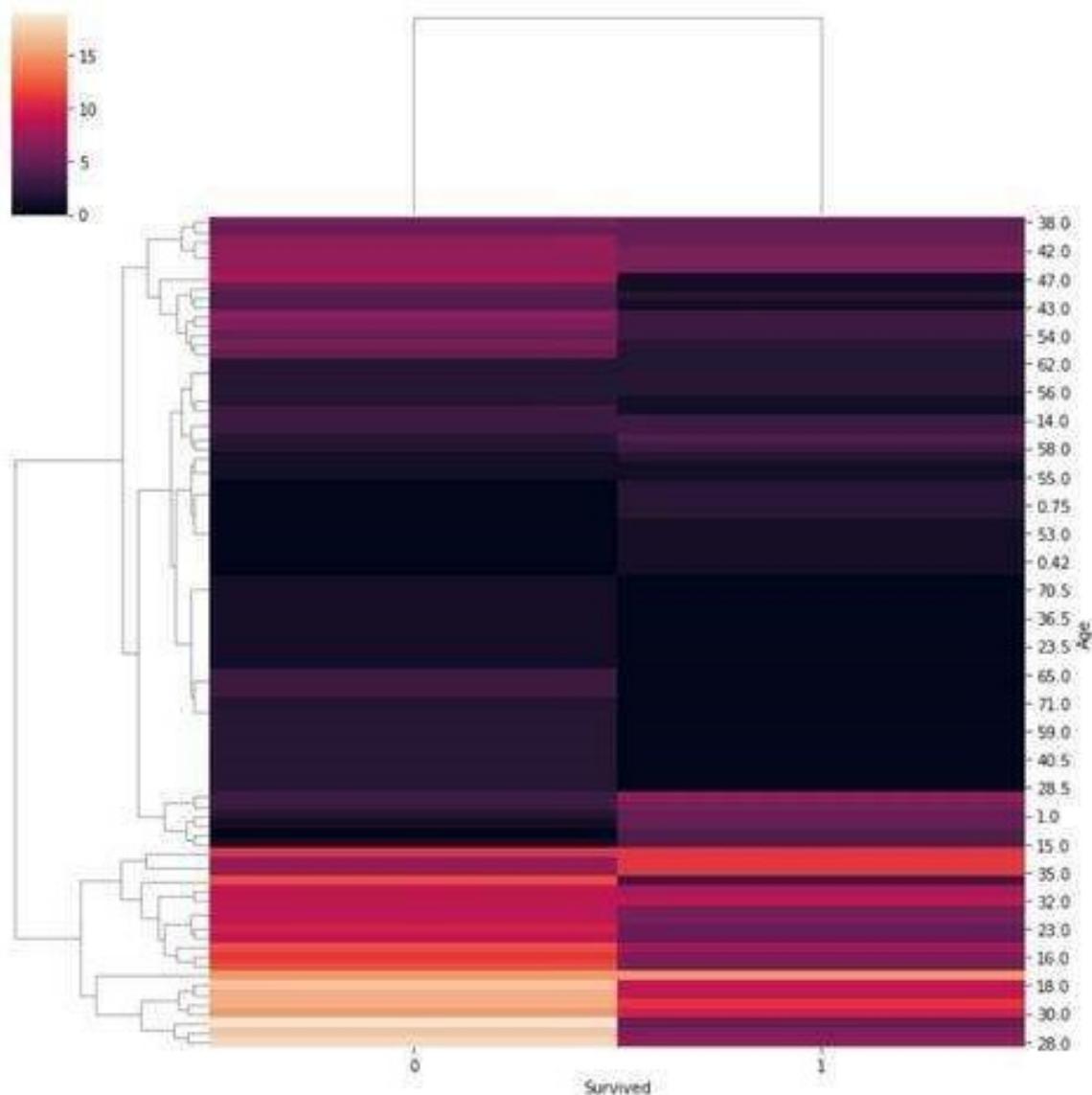
```
In [28]: sns.clustermap(pd.crosstab(df['Sex'], df['Survived']))b
```

```
Out[28]: <seaborn.matrix.ClusterGrid at 0x8345c8bd48>
```



```
In [29]: sns.clustermap(pd.crosstab(df['Age'], df['Survived']))
```

```
Out[29]: <seaborn.matrix.ClusterGrid at 0x83469736c8>
```



Assignment No 10 : Data Visualization 3

Importing Required Libraries

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

Reading csv into Dataframe

```
In [2]: df = pd.read_csv('iris.csv')  
df
```

Out[2]:

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

	145	6.7	3.0	5.2	2.3	Iris-virginica
	146	6.3	2.5	5.0	1.9	Iris-virginica
	147	6.5	3.0	5.2	2.0	Iris-virginica
	148	6.2	3.4	5.4	2.3	Iris-virginica
	149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

Data preprocessing

```
In [3]: df.shape
```

```
Out[3]: (150, 6)
```

```
In [4]: df.columns
```

```
Out[4]:
```

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
       ...)
```

```
In [5]: df.dtypes
```

```
Out[5]: Id          int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species         object
dtype: object
```

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

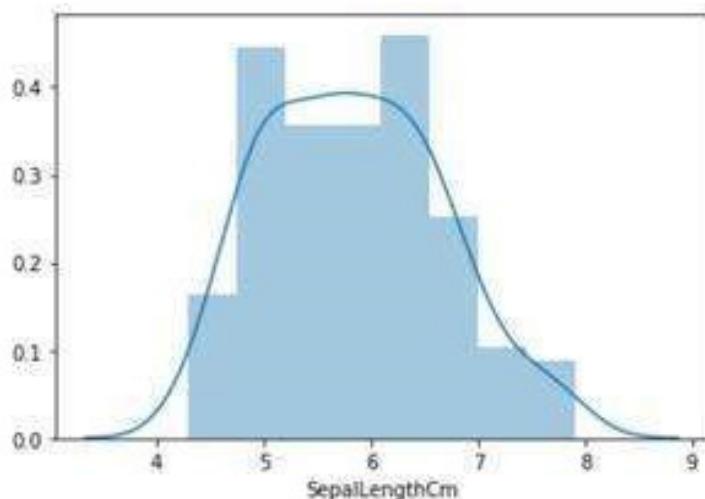
```
Out[7]:
```

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

Data Visualization

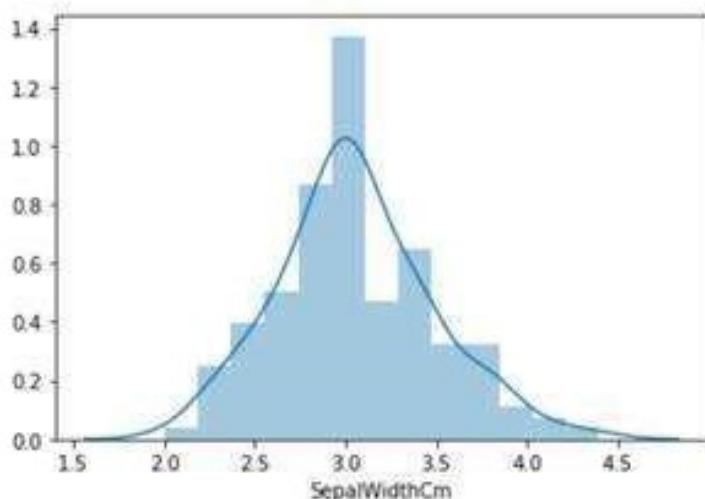
```
In [9]: sns.distplot(df['SepalLengthCm'])
```

```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x491aa5fec8>
```



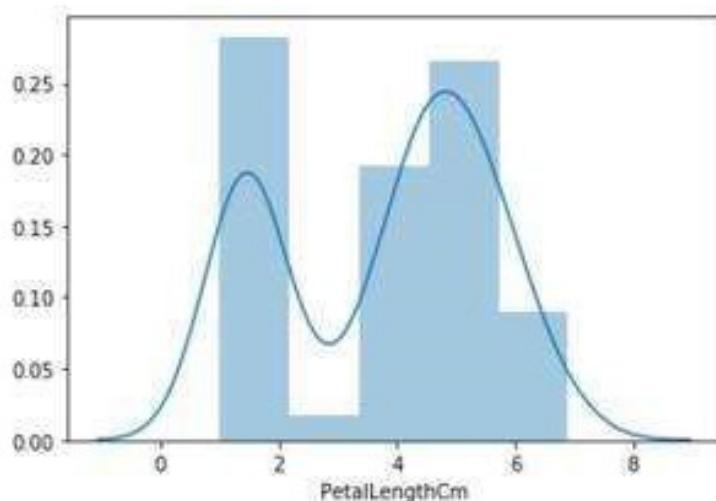
```
In [10]: sns.distplot(df['SepalWidthCm'])
```

```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x4919d7a808>
```



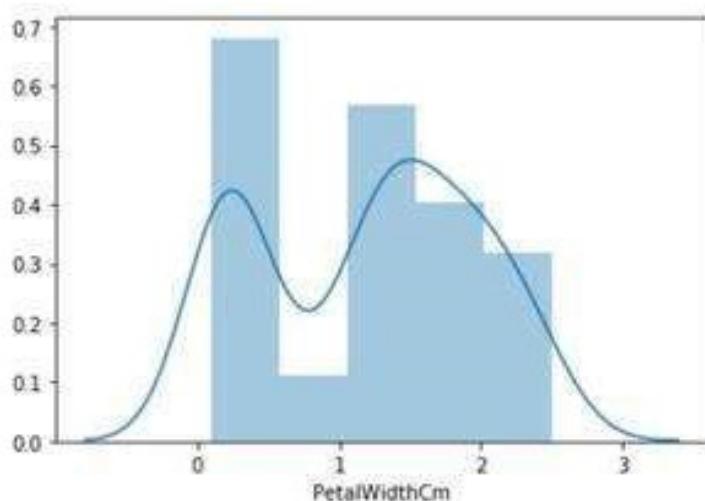
```
In [11]: sns.distplot(df['PetalLengthCm'])
```

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x491ad8f0c8>
```



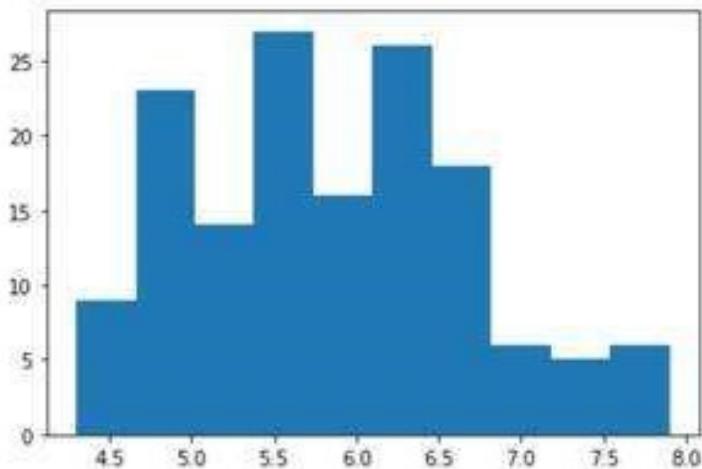
```
In [18]: sns.distplot(df['PetalWidthCm'])
```

```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x49203185c8>
```



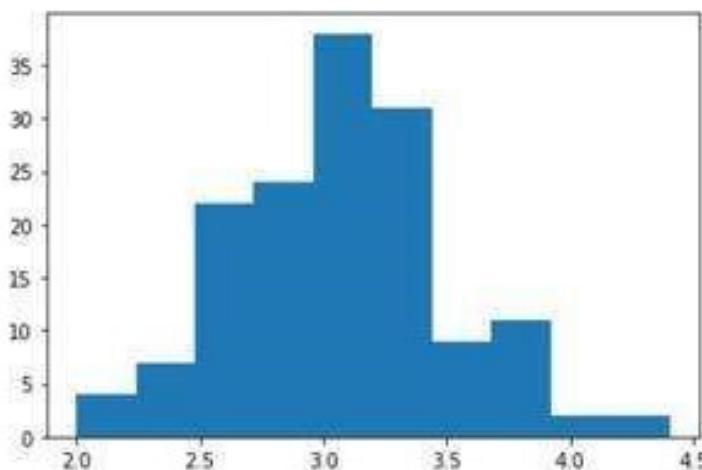
```
In [13]: plt.hist(df['SepalLengthCm'])
```

```
Out[13]: (array([ 9., 23., 14., 27., 16., 26., 18., 6., 5., 6.]),  
 array([4.3 , 4.66, 5.02, 5.38, 5.74, 6.1 , 6.46, 6.82, 7.18, 7.54, 7.9 ]),  
 <a list of 10 Patch objects>)
```



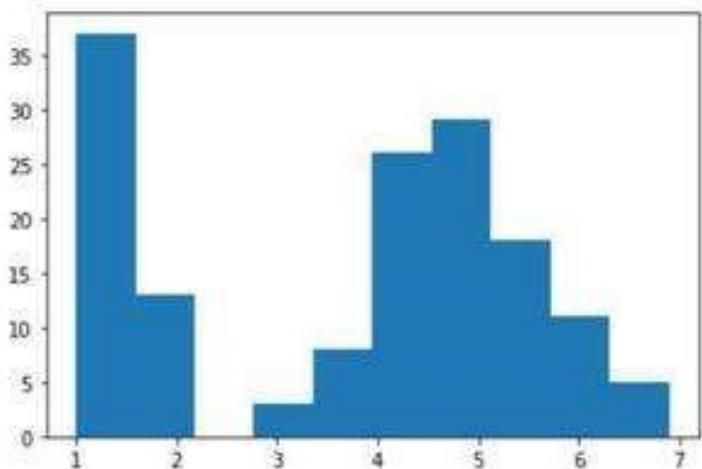
```
In [14]: plt.hist(df['SepalWidthCm'])
```

```
Out[14]: (array([ 4., 7., 22., 24., 38., 31., 9., 11., 2., 2.]),  
 array([2. , 2.24, 2.48, 2.72, 2.96, 3.2 , 3.44, 3.68, 3.92, 4.16, 4.4 ]),  
 <a list of 10 Patch objects>)
```



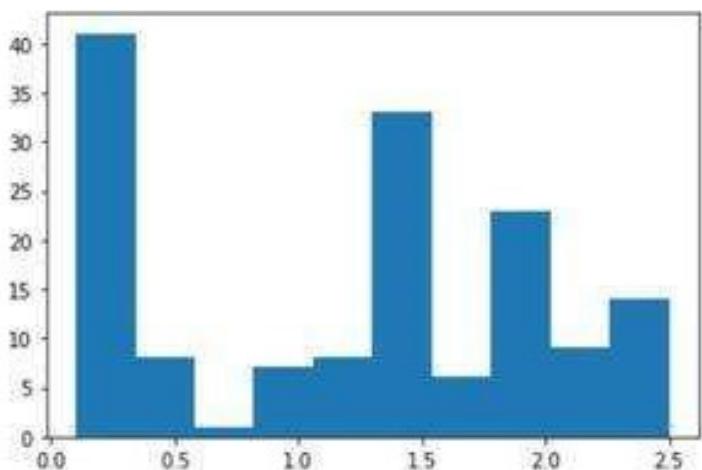
```
In [15]: plt.hist(df['PetalLengthCm'])
```

```
Out[15]: (array([37., 13., 0., 3., 8., 26., 29., 18., 11., 5.]),  
 array([1. , 1.59, 2.18, 2.77, 3.36, 3.95, 4.54, 5.13, 5.72, 6.31, 6.9 ]),  
 <a list of 10 Patch objects>)
```



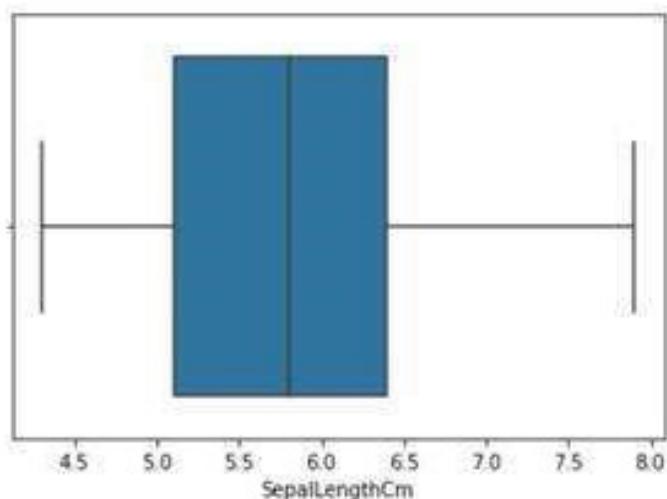
```
In [16]: plt.hist(df['PetalWidthCm'])
```

```
Out[16]: (array([41., 8., 1., 7., 8., 33., 6., 23., 9., 14.]),  
 array([0.1 , 0.34, 0.58, 0.82, 1.06, 1.3 , 1.54, 1.78, 2.02, 2.26, 2.5 ]),  
 <a list of 10 Patch objects>)
```



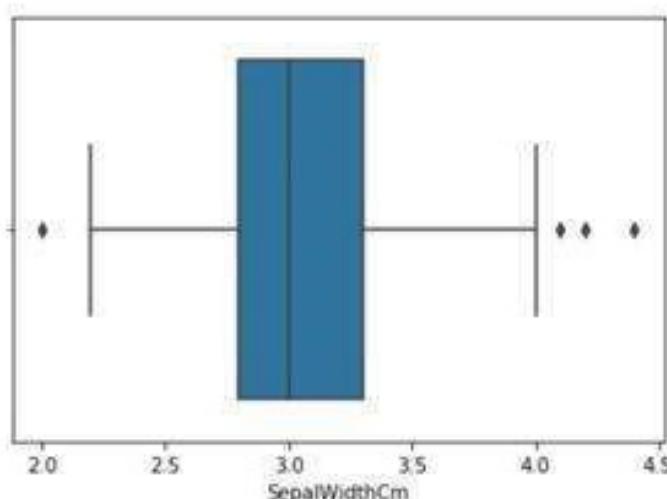
```
In [19]: sns.boxplot(df['SepalLengthCm'])
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x492033df88>
```

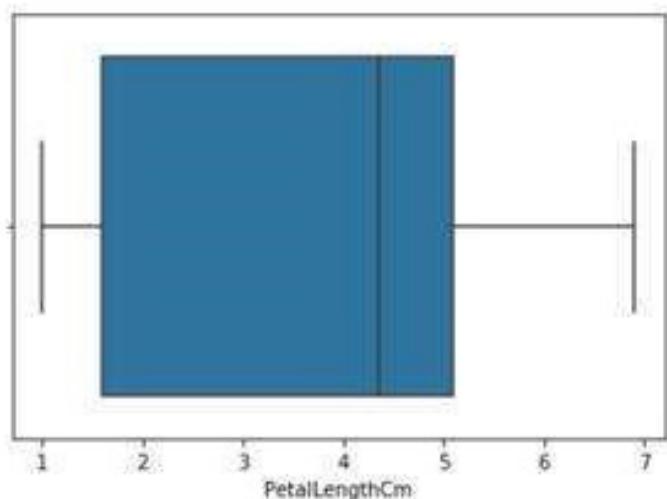


```
In [20]: sns.boxplot(df['SepalWidthCm'])
```

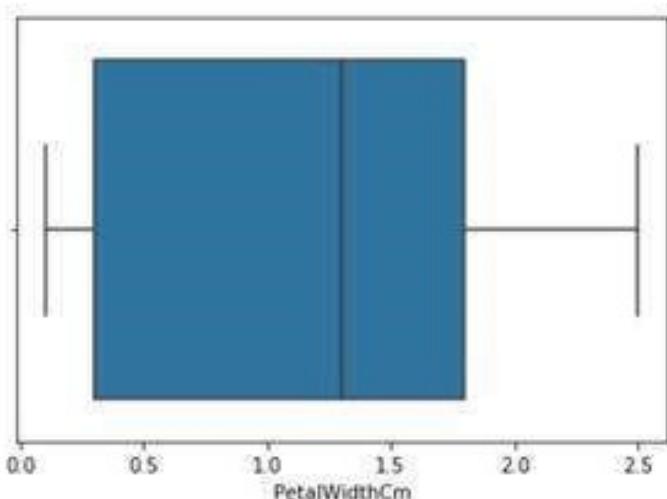
```
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x49203d7d48>
```



```
In [21]: sns.boxplot(df['PetalLengthCm'])  
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x492043fe88>
```

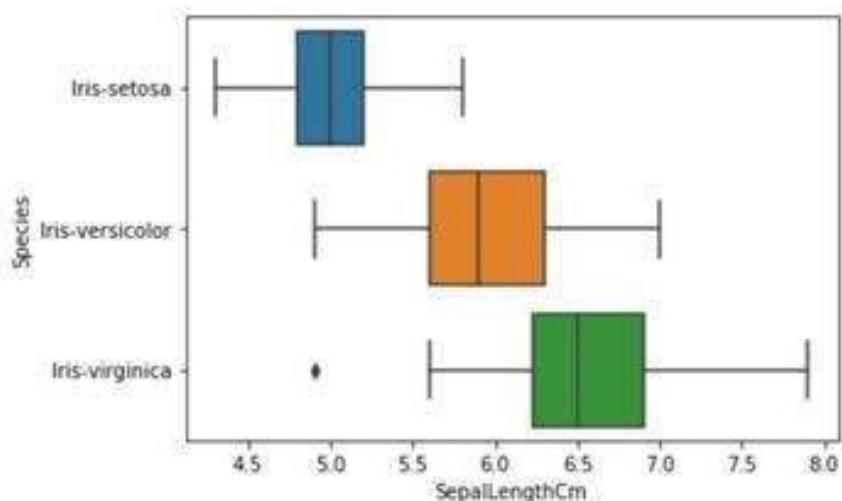


```
In [22]: sns.boxplot(df['PetalWidthCm'])  
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x492047cf88>
```



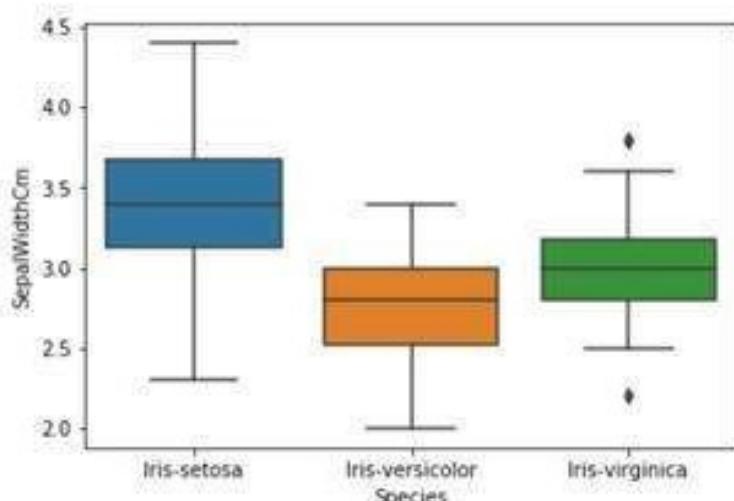
```
In [27]: sns.boxplot(data = df, x = df['SepalLengthCm'], y = df['Species'])
```

```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x49206243c8>
```



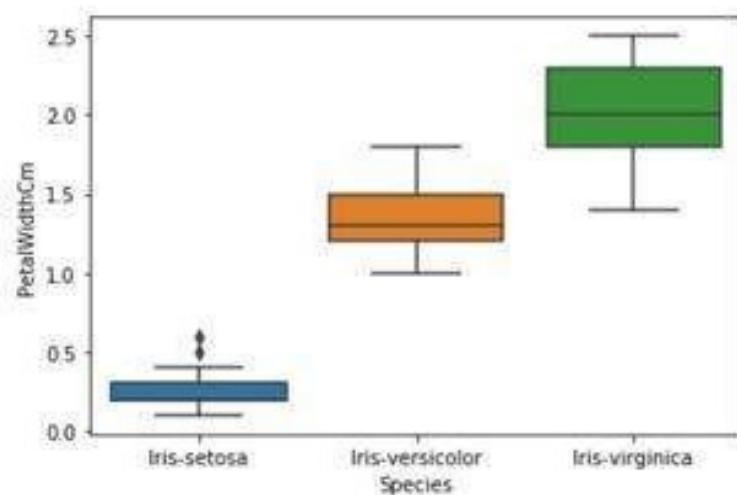
```
In [28]: sns.boxplot(data = df, x = df['Species'], y = df['SepalWidthCm'])
```

```
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x49206b7688>
```



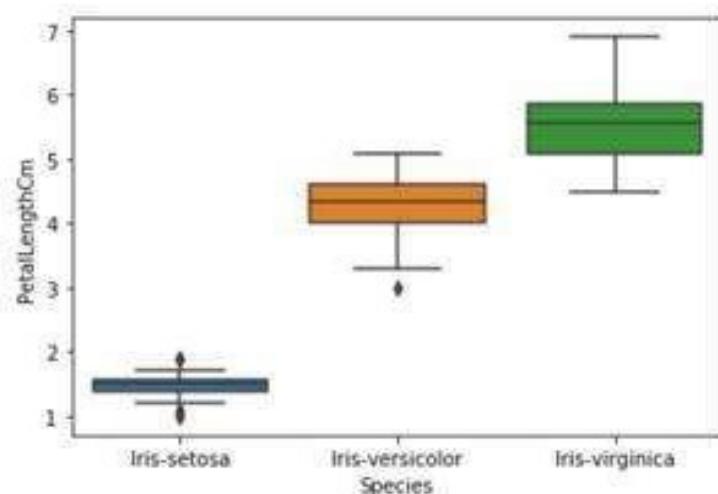
```
In [29]: sns.boxplot(data = df, x = df['Species'], y = df['PetalWidthCm'])
```

```
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x49207549c8>
```



```
In [30]: sns.boxplot(data = df, x = df['Species'], y = df['PetalLengthCm'])
```

```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x49207d9ac8>
```



Assignment No. 11

Word Count Application using Hadoop Map-Reduce Framework

```
package com.javatpoint;

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WC_Mapper extends MapReduceBase implements 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,OutputCollector<Text,IntWritable> output,
        Reporter reporter) throws IOException{
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()){
            word.set(tokenizer.nextToken());
            output.collect(word, one);
        }
    }
}
```

```
codegyani@ubuntu64server:~$ hdfs dfs -cat /r_output/part-00000
HDFS      1
Hadoop    2
MapReduce   1
a          2
is          2
of          2
processing  1
storage    1
tool        1
unit        1
codegyani@ubuntu64server:~$
```

Assignment No. 12

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

```
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);
    }
}
```

Browse Directory

/										Go!			
Show 25 entries										Search:			
	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name					
<input type="checkbox"/>	drwxr-xr-x	hduser	supergroup	0 B	Feb 18 06:03	0	0 B	input3000					
<input type="checkbox"/>	drwxr-xr-x	hduser	supergroup	0 B	Feb 17 23:22	0	0 B	inputfiles					
<input type="checkbox"/>	drwxr-xr-x	hduser	supergroup	0 B	Feb 18 06:05	0	0 B	output3000					
<input type="checkbox"/>	drwxr-xr-x	hduser	supergroup	0 B	Feb 17 23:34	0	0 B	outputfiles					
<input type="checkbox"/>	drwx----	hduser	supergroup	0 B	Feb 17 23:20	0	0 B	tmp					

Showing 1 to 5 of 5 entries

Previous 1 Next

Hadoop, 2017.

Browse Directory

/inputfiles									Go!			
Show 25 entries									Search:			
	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name				
<input type="checkbox"/>	-rw-r--r--	hduser	supergroup	120.74 KB	Feb 17 23:22	1	128 MB	SalesJan2009.csv				
Showing 1 to 1 of 1 entries												
Previous 1 Next												

Hadoop, 2017.

Browse Directory

/outputfiles									Go!			
Show 25 entries									Search:			
	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name				
<input type="checkbox"/>	-rw-r--r--	hduser	supergroup	0 B	Feb 17 23:34	1	128 MB	_SUCCESS				
<input type="checkbox"/>	-rw-r--r--	hduser	supergroup	661 B	Feb 17 23:34	1	128 MB	part-00000				
Showing 1 to 2 of 2 entries												
Previous 1 Next												

Hadoop, 2017.

Assignment 13

Locate dataset for working on weather data which reads the text input files and finds the average for temperature, dew point and wind speed.

```
// importing Libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;

public class MyMaxMin
{
    public static class MaxTemperatureMapper extends Mapper<LongWritable, Text, Text, Text>
    {
        public static final int MISSING = 9999;
        @Override
        public void map(LongWritable arg0, Text value, Context context) throws IOException, InterruptedException
        {
            String line = Value.toString();
            if (!(line.length() == 0))
            {
                String date = line.substring(6, 14);
                float temp_Max = Float.parseFloat(line.substring(39, 45).trim());
                float temp_Min = Float.parseFloat(line.substring(47, 53).trim());
                if (temp_Max > 30.0)
                {
                    context.write(new Text("The Day is Hot Day :" + date),
                                new Text(String.valueOf(temp_Max)));
                }
                if (temp_Min < 15)
                {
                    context.write(new Text("The Day is Cold Day :" + date),
                                new Text(String.valueOf(temp_Min)));
                }
            }
        }
    }

    public static class MaxTemperatureReducer extends Reducer<Text, Text, Text, Text>
    {
        public void reduce(Text key, Iterator<Text> values, Context context) throws IOException, InterruptedException
        {
            String temperature = Values.next().toString();
            context.write(key, new Text(temperature));
        }
    }

    public static void main(String[] args) throws Exception
    {
        Configuration conf = new Configuration();
        Job job = new Job(conf, "weather example");
        job.setJarByClass(MyMaxMin.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);
        job.setMapperClass(MaxTemperatureMapper.class);
        job.setReducerClass(MaxTemperatureReducer.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        Path outputPath = new Path(args[1]);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        outputPath.getFileSystem(conf).delete(outputPath);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

1	The Day is Cold Day	:20200101	-21.8
2	The Day is Cold Day	:20200102	-23.4
3	The Day is Cold Day	:20200103	-25.4
4	The Day is Cold Day	:20200104	-26.8
5	The Day is Cold Day	:20200105	-28.8
6	The Day is Cold Day	:20200106	-30.0
7	The Day is Cold Day	:20200107	-31.4
8	The Day is Cold Day	:20200108	-33.6
9	The Day is Cold Day	:20200109	-26.6
10	The Day is Cold Day	:20200110	-24.3