DATA ANALYTICS-I

1. Problem Statement

Perform the following operations using Python on any open source dataset (e.g., data.csv)

1. Import all the required Python Libraries.
2. Locate an open source data from the web (e.g., https://www.kaggle.com). Provide a clear description of the data and its source (i.e., URL of the web site).
3. Load the Dataset into pandas dataframe.
4. Data Preprocessing: check for missing values in the data using pandas isnull(), describe() function to get some initial statistics. Provide variable descriptions. Types of variables etc. Check the dimensions of the data frame.
5. Data Formatting and Data Normalization: Summarize the types of variables by checking the data types (i.e., character, numeric, integer, factor, and logical) of the variables in the data set. If variables are not in the correct data type, apply proper type conversions.
6. Turn categorical variables into quantitative variables in Python.

Import all the required Python Libraries.

```
import pandas as pd
```

Locate an open source data from the web (e.g., https://www.kaggle.com). Provide a clear description of the data

```
!pip install -q kaggle
```

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

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Iris.csv to Iris.csv
{'Iris.csv':
b'Id,SepalLengthCm,SepalWidthCm,PetalLengthCm,PetalWidthCm,Species\n1,5.1,3.5,1.4,0.2,Iris-
setosa\n2,4.9,3.0,1.4,0.2,Iris-setosa\n3,4.7,3.2,1.3,0.2,Iris-setosa\n4,4.6,3.1,1.5,0.2,Iris-
setosa\n5,5.0,3.6,1.4,0.2,Iris-setosa\n6,5.4,3.9,1.7,0.4,Iris-setosa\n7,4.6,3.4,1.4,0.3,Iris-
setosa\n8,5.0,3.4,1.5,0.2,Iris-setosa\n9,4.4,2.9,1.4,0.2,Iris-setosa\n10,4.9,3.1,1.5,0.1,Iris-
setosa\n11,5.4,3.7,1.5,0.2,Iris-setosa\n12,4.8,3.4,1.6,0.2,Iris-setosa\n13,4.8,3.0,1.4,0.1,Iris-
setosa\n14,4.3,3.0,1.1,0.1,Iris-setosa\n15,5.8,4.0,1.2,0.2,Iris-setosa\n16,5.7,4.4,1.5,0.4,Iris-
setosa\n17,5.4,3.9,1.3,0.4,Iris-setosa\n18,5.1,3.5,1.4,0.3,Iris-setosa\n19,5.7,3.8,1.7,0.3,Iris-
setosa\n20,5.1,3.8,1.5,0.3,Iris-setosa\n21,5.4,3.4,1.7,0.2,Iris-setosa\n22,5.1,3.7,1.5,0.4,Iris-
setosa\n23,4.6,3.6,1.0,0.2,Iris-setosa\n24,5.1,3.3,1.7,0.5,Iris-setosa\n25,4.8,3.4,1.9,0.2,Iris-
setosa\n26,5.0,3.0,1.6,0.2,Iris-setosa\n27,5.0,3.4,1.6,0.4,Iris-setosa\n28,5.2,3.5,1.5,0.2,Iris-
setosa\n29,5.2,3.4,1.4,0.2,Iris-setosa\n30,4.7,3.2,1.6,0.2,Iris-setosa\n31,4.8,3.1,1.6,0.2,Iris-
setosa\n32,5.4,3.4,1.5,0.4,Iris-setosa\n33,5.2,4.1,1.5,0.1,Iris-setosa\n34,5.5,4.2,1.4,0.2,Iris-
setosa\n35,4.9,3.1,1.5,0.1,Iris-setosa\n36,5.0,3.2,1.2,0.2,Iris-setosa\n37,5.5,3.5,1.3,0.2,Iris-
setosa\n38,4.9,3.1,1.5,0.1,Iris-setosa\n39,4.4,3.0,1.3,0.2,Iris-setosa\n40,5.1,3.4,1.5,0.2,Iris-
setosa\n41,5.0,3.5,1.3,0.3,Iris-setosa\n42,4.5,2.3,1.3,0.3,Iris-setosa\n43,4.4,3.2,1.3,0.2,Iris-
setosa\n44,5.0,3.5,1.6,0.6,Iris-setosa\n45,5.1,3.8,1.9,0.4,Iris-setosa\n46,4.8,3.0,1.4,0.3,Iris-
setosa\n47,5.1,3.8,1.6,0.2,Iris-setosa\n48,4.6,3.2,1.4,0.2,Iris-setosa\n49,5.3,3.7,1.5,0.2,Iris-
setosa\n50,5.0,3.3,1.4,0.2,Iris-setosa\n51,7.0,3.2,4.7,1.4,Iris-versicolor\n52,6.4,3.2,4.5,1.5,Iris-
versicolor\n53,6.9,3.1,4.9,1.5,Iris-versicolor\n54,5.5,2.3,4.0,1.3,Iris-
versicolor\n55,6.5,2.8,4.6,1.5,Iris-versicolor\n56,5.7,2.8,4.5,1.3,Iris-
versicolor\n57,6.3,3.3,4.7,1.6,Iris-versicolor\n58,4.9,2.4,3.3,1.0,Iris-
versicolor\n59,6.6,2.9,4.6,1.3,Iris-versicolor\n60,5.2,2.7,3.9,1.4,Iris-
versicolor\n61,5.0,2.0,3.5,1.0,Iris-versicolor\n62,5.9,3.0,4.2,1.5,Iris-
versicolor\n63,6.0,2.2,4.0,1.0,Iris-versicolor\n64,6.1,2.9,4.7,1.4,Iris-
versicolor\n65,5.6,2.9,3.6,1.3,Iris-versicolor\n66,6.7,3.1,4.4,1.4,Iris-
versicolor\n67,5.6,3.0,4.5,1.5,Iris-versicolor\n68,5.8,2.7,4.1,1.0,Iris-
versicolor\n69,6.2,2.2,4.5,1.5,Iris-versicolor\n70,5.6,2.5,3.9,1.1,Iris-
versicolor\n71,5.9,3.2,4.8,1.8,Iris-versicolor\n72,6.1,2.8,4.0,1.3,Iris-
versicolor\n73,6.3,2.5,4.9,1.5,Iris-versicolor\n74,6.1,2.8,4.7,1.2,Iris-
versicolor\n75,6.4,2.9,4.3,1.3,Iris-versicolor\n76,6.6,3.0,4.4,1.4,Iris-
```

2. Data Collection

Loading the data

```
iris=pd.read_csv("/content/Iris.csv")
```

```
iris
```

	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

3. Exploratory Data Analysis

Data Preprocessing:

```
iris.head
```

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

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

[150 rows x 6 columns]>
```

```
iris.head()
```

	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

```
iris.tail()
```

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

```
iris.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
```

```
iris.describe(include="all")
```

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

```
iris.shape
```

```
(150, 6)
```

```
iris.columns
```

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

```
iris.PetalLengthCm
```

```
0      1.4
1      1.4
2      1.3
3      1.5
4      1.4
      ...
145    5.2
146    5.0
147    5.2
148    5.4
149    5.1
Name: PetalLengthCm, Length: 150, dtype: float64
```

```
iris[:6]
```

	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
5	6	5.4	3.9	1.7	0.4	Iris-setosa

```
iris.loc[0:2]
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa

```
iris.iloc[1:3]
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa

```
iris.isnull()
```

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

150 rows × 6 columns

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

```
Id               False
SepalLengthCm    False
SepalWidthCm     False
PetalLengthCm    False
PetalWidthCm     False
Species          False
dtype: bool
```

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

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

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

```
0
```

Data Formatting and Data Normalization:

```
iris.dtypes
```

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

```
from sklearn import preprocessing
```

```
iris.head()
```

	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

```
x=iris.iloc[:,:4]
x
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	1	5.1	3.5	1.4
1	2	4.9	3.0	1.4
2	3	4.7	3.2	1.3
3	4	4.6	3.1	1.5
4	5	5.0	3.6	1.4
...	...	...	...	...
145	146	6.7	3.0	5.2
146	147	6.3	2.5	5.0
147	148	6.5	3.0	5.2
148	149	6.2	3.4	5.4
149	150	5.9	3.0	5.1

150 rows × 4 columns

```
min_max_scaler = preprocessing.MinMaxScaler()

x_scaled = min_max_scaler.fit_transform(x)

df_normalized = pd.DataFrame(x_scaled)

df_normalized
```

	0	1	2	3
0	0.000000	0.222222	0.625000	0.067797
1	0.006711	0.166667	0.416667	0.067797
2	0.013423	0.111111	0.500000	0.050847
3	0.020134	0.083333	0.458333	0.084746
4	0.026846	0.194444	0.666667	0.067797
...	...	...	...	...
145	0.973154	0.666667	0.416667	0.711864
146	0.979866	0.555556	0.208333	0.677966
147	0.986577	0.611111	0.416667	0.711864
148	0.993289	0.527778	0.583333	0.745763
149	1.000000	0.444444	0.416667	0.694915

150 rows × 4 columns

Turn categorical variables into quantitative variables in Python

```
from sklearn import preprocessing

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

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

```
label_encoder = preprocessing.LabelEncoder()
```

```
iris['Species']= label_encoder.fit_transform(iris['Species'])
```

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

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

DATA ANALYTICS-I

1. Problem Statement

Create an “Academic performance” dataset of students and perform the following operations using Python.

1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them.
2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.
3. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution.

Loading the libraries

```
import pandas as pd
import numpy as np
```

Upload Dataset

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

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Academic_perf.xlsx to Academic_perf.xlsx
{'Academic_perf.xlsx':
b'PK\x03\x04\n\x00\x00\x00\x00\x00\x87N\xe2@\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\t\x0
\xdfO\xe4|\xd3\xd4\xac\x85\x88\xd6\xbb\x82O\x0e\xc6\x9c\x813\xbe\xb4nU\xf0\x9b\xe5\xf9\xe8\x983$\
{\x07\x05\xdf\x02\xf2\xb9\xda\xdf\x93\x8b\xe8\x03D\xb2\x80,E8,\xf8\x9a(\xcc\x84@\xb3\x86F\xe3A\x9
x\xab\xa3\xd5\x8eR|g\xcbK?
\xd7\x01)\xaa\x8f\xd7\xe7\xf7\xb7\xc7\xcf\xa7\x17)\x92\x9e\xdf\xfaqh\x1d\xce\xf6PMzC\x1av\x8d]@\x
t\xa4?\x80\'C\xe0\x9e!\xe3f\x9c\xeb5\x00\xe5\x9bC\xbe\xbeq\xba\xf4+
[\xfc\xfc\xb9\xfa\x02PK\x03\x04\x14\x00\x00\x00\x08\x00\x87N\xe2@n\xe9v\x16=\x01\x00\x00U\x02\x00
\xf4\xec}\x1fjl\xc8\xe3\xe8\xff\x10\xb32\xb9"\xd7\xb7$\x9d\x13O\x00:\xe4>\xff\x08\xf4\x1bPK\x03\x
^(8w_\xdat+\xb4II\xd2\xe9\x10\xff\xbb\x19:\xc5\x0bo\xf4\xf2\xf0\x1e\x1e\x9e\xf7\x9ct\xfd2\x0e\xe0
\r\x08\x04R5\xba\xed\xd5>\x83O\xdb\n\xc5\x10XW\xab\xb6\x1e\xb4\x92\x19<I\x0b\xd7\xf9\xf5U\xfa`\xf
[\xdd\x9c\xed\xecn{\x9a\xbcn\x9e~\xc2O\xa0\x1b]\xdff\xf0\xb5\x8c\x8a\xb2\x8cH\x84\xd8&)\x10%T\xa0
4}\x97\rT\x14\xc2F\x10BB"\xc6\x00\xc6\xe7\xba]\xa0\xb1\x1fK\xef\x9d\xc8\xbf\xb7\xdd\x98B\xdcc\xcf
\xc0\xf8\xc9\xd6:-
\xc8\x1f\xdd.\xc2\xc2\x81\xc8q\x0f@ZE\xbd8\xeeGZH\xc3\x99\xcc\x13\xde\xe3\xcc\x08\xed\xb7o\xc2R\x
\r%|\xe0\x15\xd5#j\xbc2\xd2m\x8c\xac\xacR/\x96\xf1K\xed\xa5\xd9\xe0\xae\x95q\xdb1\x99\xad\x9ff\x9
\xd2\xf9\xfb\xed\xb9\x80\x00\x0enS\x91\xf4
\x91=szf\xfat7f\x1a\xa3\x87_~\x9c\x8e\x83k\x94\xa4\x87\xf8\xfc\xa8\xe9\xc3\xb16\x88\xce\xdbxw8?
=j_\xbfx?/\xb4A\x9a\x85\xe7]x\x8c\xcf\xd1\xa3\xf6w\x94j\xbf\xac\xff\xfd\xaf\x87[\x9c|O\x9f\xa3(\x
<\x9c5\xa1a\x95\xa8\xe8\x88\xf7\xfb\xc36r\xe2\xed\xcb):gBI\x12\x1d\xc3\x8c\xe6\x9f>\x1f.i\xae\xed
{~\xd4\x96Z.\xfb=\xbe\x05\xd1\xe1\xe99\xa3\xa9L\x86\x13\xa6t\x1b\x1fI\x03\xfd\x1e\x9c\x0e\xcc\x9b
\xfd\xcc:\xe1\x148|t\xfa\x9b\xc3\x8d\xea\xf0M\xf0\x91X\x03\xb7\x0e\xa3k\xfd\x90\xc4\xb7\x01\xb923
\x132\xb3\xd2\xfaS\xfa~]\x8f\x1fFW2\xe6V\xb6Y\xe56\xbd\xdaf\x97\xdb\x8cj\x9bSn3\xabmn\xb9mRm\xf3\
\xeem#\xb2\xc9\xdd0F\xa3a\x0cn\x12X\xb6E}\xef\xe6Z\xde\xd5qS\xda\x021\x05+:B<\x83\x89\xb9B\xbc(\xe
=\xf0\xd0\xf9\xe3\xebo\xeflc\xe5\x19\xef\x1fF{F\xa81\x859\x06\xcdz\x02c\x15\x18\xe4\x8cL\x8f\xae\
N\xa4\xf8s\x98\xa6B\xf6\xa1\rx\xcc>\xdeq\\\x8b.\xb1^x8J}\xb9\x15\x98\xba;U\x15\xe6\xcdF\xe6M\xce<8
3
\xd3\x11b\x13\xc8t\xe5\x00\xd0\xdbk\x16\xfbL\\\xa1^\xb0M\xd9\xea\xba\xd6\xa7\xa0$h\xec\xcd\xd3\xd2
\xec\x07ae\x0b\xc8\x12\x16\xe30\xf1~\xb0\r\xb9W\xfe\xfe\xeb\'\xc7r\xbf\xfc\xe9\xba\x9f\xde\x99\xe3
\xb5\x84\xc6\xc4\xfd\xac\xcc!\xf5\xb8j0\xb4\xa4\'&1\x87\xb9\xf9emh\xa1\xb1\xf0[c\t\xa0\xa0\x0b\x9
\r\\\xdc\x96\x10H\xdb\x0e\x13\xf7s9\xc3\xd4&\xb5A\xba\xf1\xd4\xb4-
\x80K\xbf\x0ck#e\t\x0b\n\xba@\xf7\xd4V\x7f\xe0H#\xbd\x8e\x15:R5\xb0\xc2\xce\x18\xd75$\t\x8b\xfa\x
\xc0\xaf<\xb1.*>\xdc\xf7D|\xe7\xee\x97\xd5\xb5\xd06\x99"m]
I\x9bN\x87F\\\xf2F\x9a\xf1u\xbc\xe9T\xabi
\x8e\xc4\x8c9pr\x8b\xf5.\xa8\x03\xab\xd8\x1c\xd4\xcf\x9d\tVs\x14q\x0b\x98\x8d\xab\x88C\xd2=\x8e\xb
gz\xaa8\x18\xcf\xaf\xe0\xda\x08\x9a\xc0\x9a\x83NT\x1e^\xe3\xa1\x01\xa3m\x08\xa7LP\xfb9Jo.\x0f\x90
A\'*\xe7v\xde\x10\x84oR\xcc\xd0\x9b\xab\x19$\xe6\x04C\xb8X\xac{A0\x84\xa8\xcdQ\xfd\x04c\xb6t\x14q
\x88$[\xa2p\xbb\xebpy?
qX\x92p+\xb8\xb2A\xf9\x8b$j^\xb9\x06{\x0cS\xb1\x93\xb2zk\x85\x11\x89\xf5*z\xdbq\x0bpO\xbf\x82+\xc
r\xdd\n\x0e\rg\x8az_\x9d\xa5r\xa5\xa2\x9d%\x13\xec\xedw\x0e&\xf3\xa4>\x85\xa7G\xd0\x89\x92,\x99\x
t\xbdn3\x9a\xeb5$\xe6\xe4\xc1\xbe\xc0b\xdd\x0b\xf2\x80Z\x9b\xa3\xfa\xc9\xc3M\xab\xa3\x8a\xc3\x0c^
\xd6\xbd
\x17\xa8\xb79\xaa\x9f\\,G:\xaa8\x18\xcf\xad\xe0\xd0\xde\x92\xdc\xda1\x86\x83\xa8B\x0c\x9e\xe9w*\x
\xd6\xfd\xce\x1e\xd6Dm\x8e\xeag\xcf\x84|\xec\xfcC\x1c\xdb\x02\x17\xd5\xbf\x16\xf6p\x93\xe4U@\xed\
p\'*\x0fM\xb3\x81]\x95JP\x7f\xe2m\xae\x04\x19\xa2\x12d@\xf6\xb1H^b\x17\xb8\xb7%\nwQ\x0e\x97\xf7\x
\x04\x19Hk\xd0\x89\xca\xc3\xaf\xfeV\x90`|\x08\x95\xdc\xd9\x11}\xcdU
CT\x81\x0c\xd82X$/\xf1\x03\xec\xd9\x1c\xd5\xcfC\xed\x8c*G\x9b\xc1\x91\xc7U\xd47\x83yx\x8a8\xb3\x9
\xb0\xb9\xcac\x88*\x8f\x01&\xb5H^"\x10\xe8\xb59\xaa\x9f\xc0)\x1c\x8b\x1cE\x1c\x1e\xea]E\xdc\x04V\
\x7f\x0b\xd0\xea(\xe20\xf1\xbar\x96K\xf0\x07OQ\xdf\xa2\x96@\xbb\xee\xb4H\xfe\xcc\t\xb0\xce\xee_\x
\x84%\x7f\xb8d~\xe8d\x86|e\xfal\xae\xf0\xd0\xcdR\xce\x1e\xd8\xc5b7N\x0b\xf6\x80\x05[\xa2j5T)\xc7%
\xfd\xbd\x16\xb9\xb8\xef\xca/\x04\xd2U\x016\ry!\xb6\\A\xc3\xddr\xc0\x95\xed\xd7\xef\x0c\x83_k}\xc
\x89z\x8c\xb8v..V_\xc2\xa7\xe8\xb70y:\x9c\xd3\xc11\xda\xd3d\xc6C\x02\'\xe2N9\xff\x9c\xc5\x17.\xa5
\xd9\xec\xb8\x97\x8c\xee\rt\x93}t\xff\'\x8b\xf5\xff\x01PK\x03\x04\n\x00\x00\x00\x00\x00\x87N\xe2@
\t\x8f<\xaa\x84\x1c\x9d\x80\xdf\x84V\xb6\xaa\xd5f%A$\xf5\xbd\x14%\xe0\xf6\xdedB\x02\xec\x8d\x94K\
\x95\xbeG\x91\x90\xf0\xa2\xe3W\xf5\x8f_\xd9\xbdQA;\x85\x11\x95\x1bl\r\xbb\x81\xfe)\xec\n\x83p\xba
\xcb\xbf\x06\xc5\x94\xa4\xd35t\xb5\xd1\xdc\xee.V[B&\x8c\x1e8\xe1\xedF}\xd0\xda*\x9c/QP\reu\xa9)&,
AO\x18\x1f\x00@\x01)\x92$\xf5\xe4<\xc3\x13\x14@\x15w\x11%cN\xbcC\x12\xc5RM\x83v02\xde\xe7C\x81X\x
\xbe{\xf1m>\xf5*^\x98\xf8\xb7\xbf|\xf5\xf6\x8f?
\xdf\xe7\x1e6\x93A\xeb\xbb\x97o_\xbd|\xf3\xfd\xd7\x7f\xfd\xfc\xc2\xe1}\x8f\xa3\xb1\t\x1f\x91\x04\
\xa2\xf3\xc4FrI\xa6.\x9f\x87\x881\x13\xd9c\xd3n\x8c\x92\xcc\x85\x1d\x9246\xb1\x9f\x8b)\x94(\xf2\xe
```

2. Data Collection

Loading the data

```
    \n\xea\x1a\xca\xc7\x11\xcdbTt\x14S\xccs\xb8\x96\xf2\x92\x8e~*c`<\x15k\x86\x80\x1a!)\x1a\xe18R\r\x
```

```
df=pd.read_excel("/content/Academic_perf.xlsx")
```

```
    4h\xf7 PK\x03\x04\x14\x00\x00\x00\x08\x00\x87N\xe2@\xe9\xc8\xd6my\t\x00\x00\xb5K\x00\x00\r\x00\x0
```

```
df
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
0	1	A	50.0	67	88	54.0	259	64.75	Pass
1	2	B	68.0	34	28	28.0	158	39.50	Fail
2	3	C	98.0	59	67	67.0	291	72.75	Pass
3	4	D	50.0	79	91	72.0	292	73.00	Pass
4	5	E	56.0	67	35	78.0	236	59.00	Pass
5	6	F	79.0	72	43	72.0	266	66.50	Pass
6	7	G	71.0	69	72	83.0	295	73.75	Pass
7	8	H	75.0	93	87	200.0	455	113.75	Pass
8	9	I	30.0	89	93	60.0	272	68.00	Fail
9	10	J	57.0	56	70	32.0	215	53.75	Fail
10	11	K	68.0	45	63	65.0	241	60.25	Pass
11	12	L	64.0	88	59	NaN	211	52.75	False
12	13	M	63.0	41	59	66.0	229	57.25	Pass
13	14	N	45.0	36	80	58.0	219	54.75	Pass
14	15	O	75.0	78	36	84.0	273	68.25	Pass
15	16	P	NaN	82	36	33.0	151	37.75	Pass
16	17	Q	58.0	78	66	80.0	282	70.50	Pass
17	18	R	72.0	82	33	83.0	270	67.50	Fail

3. Exploratory Data Analysis

Data Preprocessing:

```
df.head()
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
0	1	A	50.0	67	88	54.0	259	64.75	Pass
1	2	B	68.0	34	28	28.0	158	39.50	Fail
2	3	C	98.0	59	67	67.0	291	72.75	Pass
3	4	D	50.0	79	91	72.0	292	73.00	Pass
4	5	E	56.0	67	35	78.0	236	59.00	Pass

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   RollNo      26 non-null     int64  
 1   Name        26 non-null     object 
 2   DSBDA       25 non-null     float64
 3   AI          26 non-null     int64  
 4   CC          26 non-null     int64  
 5   WT          25 non-null     float64
 6   TotalMarks  26 non-null     int64  
 7   Percentage  26 non-null     float64
 8   Result      26 non-null     object 
dtypes: float64(3), int64(4), object(2)
memory usage: 2.0+ KB
```

```
df.describe(include="all")
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
count	26.000000	26	25.000000	26.000000	26.000000	25.000000	26.000000	26.000000	26
unique	NaN	26	NaN	NaN	NaN	NaN	NaN	NaN	3
top	NaN	A	NaN	NaN	NaN	NaN	NaN	NaN	Pass
freq	NaN	1	NaN	NaN	NaN	NaN	NaN	NaN	19
mean	13.500000	NaN	62.040000	66.000000	59.269231	69.840000	252.076923	63.019231	NaN

```
df.shape
```

```
(26, 9)
```

```
df.dtypes
```

```
RollNo          int64
Name           object
DSBDA         float64
AI              int64
CC              int64
WT            float64
TotalMarks      int64
Percentage    float64
Result         object
dtype: object
```

```
df.columns
```

```
Index(['RollNo ', 'Name ', 'DSBDA', 'AI ', 'CC', 'WT', 'TotalMarks',
       'Percentage', 'Result'],
      dtype='object')
```

```
df[15:22]
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
15	16	P	NaN	82	36	33.0	151	37.75	Pass
16	17	Q	58.0	78	66	80.0	282	70.50	Pass
17	18	R	72.0	82	33	83.0	270	67.50	Fail
18	19	S	41.0	42	31	52.0	166	41.50	Fail
19	20	T	55.0	59	59	88.0	261	65.25	Pass
20	21	U	37.0	37	33	66.0	173	43.25	Fail
21	22	V	78.0	52	65	71.0	266	66.50	Pass

```
df.loc[0:6]
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
0	1	A	50.0	67	88	54.0	259	64.75	Pass
1	2	B	68.0	34	28	28.0	158	39.50	Fail
2	3	C	98.0	59	67	67.0	291	72.75	Pass
3	4	D	50.0	79	91	72.0	292	73.00	Pass
4	5	E	56.0	67	35	78.0	236	59.00	Pass
5	6	F	79.0	72	43	72.0	266	66.50	Pass
6	7	G	71.0	69	72	83.0	295	73.75	Pass

```
df.loc[0:6,'DSBDA':'CC']
```

	DSBDA	AI	CC
0	50.0	67	88
1	68.0	34	28
2	98.0	59	67
3	50.0	79	91
4	56.0	67	35
5	79.0	72	43
6	71.0	69	72

```
df.iloc[1:3]
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
1	2	B	68.0	34	28	28.0	158	39.50	Fail
2	3	C	98.0	59	67	67.0	291	72.75	Pass

```
df.iloc[1:5,1:5]
```

	Name	DSBDA	AI	CC
1	B	68.0	34	28
2	C	98.0	59	67
3	D	50.0	79	91
4	E	56.0	67	35

Identification and Handling of Null Values

```
df.isnull()
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False	False
10	False	False	False	False	False	False	False	False	False
11	False	False	False	False	False	True	False	False	False
12	False	False	False	False	False	False	False	False	False
13	False	False	False	False	False	False	False	False	False
14	False	False	False	False	False	False	False	False	False
15	False	False	True	False	False	False	False	False	False
16	False	False	False	False	False	False	False	False	False
17	False	False	False	False	False	False	False	False	False
18	False	False	False	False	False	False	False	False	False
19	False	False	False	False	False	False	False	False	False
20	False	False	False	False	False	False	False	False	False
21	False	False	False	False	False	False	False	False	False
22	False	False	False	False	False	False	False	False	False
23	False	False	False	False	False	False	False	False	False
24	False	False	False	False	False	False	False	False	False
25	False	False	False	False	False	False	False	False	False

```
df.isna()
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False	False
10	False	False	False	False	False	False	False	False	False
11	False	False	False	False	False	True	False	False	False
12	False	False	False	False	False	False	False	False	False
13	False	False	False	False	False	False	False	False	False
14	False	False	False	False	False	False	False	False	False
15	False	False	True	False	False	False	False	False	False
16	False	False	False	False	False	False	False	False	False
17	False	False	False	False	False	False	False	False	False
18	False	False	False	False	False	False	False	False	False
19	False	False	False	False	False	False	False	False	False
20	False	False	False	False	False	False	False	False	False

```
df.isnull().any()
```

```
RollNo        False
Name          False
DSBDA          True
AI            False
CC            False
WT             True
TotalMarks    False
Percentage    False
Result        False
dtype: bool
```

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

```
RollNo        0
Name          0
DSBDA         1
AI            0
CC            0
WT            1
TotalMarks    0
Percentage    0
Result        0
dtype: int64
```

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

```
1
```

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

```
1
```

```
cols_with_na = []
for col in df.columns:
    if df[col].isna().any():
       cols_with_na.append(col)
cols_with_na
```

```
['DSBDA', 'WT']
```

Filling missing values using dropna(), fillna(), replace() :

```
df.replace(np.nan,value=0)
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
0	1	A	50.0	67	88	54.0	259	64.75	Pass
1	2	B	68.0	34	28	28.0	158	39.50	Fail
2	3	C	98.0	59	67	67.0	291	72.75	Pass
3	4	D	50.0	79	91	72.0	292	73.00	Pass
4	5	E	56.0	67	35	78.0	236	59.00	Pass
5	6	F	79.0	72	43	72.0	266	66.50	Pass
6	7	G	71.0	69	72	83.0	295	73.75	Pass
7	8	H	75.0	93	87	200.0	455	113.75	Pass
8	9	I	30.0	89	93	60.0	272	68.00	Fail
9	10	J	57.0	56	70	32.0	215	53.75	Fail
10	11	K	68.0	45	63	65.0	241	60.25	Pass
11	12	L	64.0	88	59	0.0	211	52.75	False
12	13	M	63.0	41	59	66.0	229	57.25	Pass
13	14	N	45.0	36	80	58.0	219	54.75	Pass
14	15	O	75.0	78	36	84.0	273	68.25	Pass
15	16	P	0.0	82	36	33.0	151	37.75	Pass
16	17	Q	58.0	78	66	80.0	282	70.50	Pass
17	18	R	72.0	82	33	83.0	270	67.50	Fail
18	19	S	41.0	42	31	52.0	166	41.50	Fail
19	20	T	55.0	59	59	88.0	261	65.25	Pass
20	21	U	37.0	37	33	66.0	173	43.25	Fail
21	22	V	78.0	52	65	71.0	266	66.50	Pass
22	23	W	75.0	64	62	38.0	239	59.75	Pass
23	24	X	56.0	84	44	90.0	274	68.50	Pass
24	25	Y	85.0	78	96	81.0	340	85.00	Pass
25	26	Z	45.0	85	45	45.0	220	55.00	Pass

```
df.fillna(1)
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
0	1	A	50.0	67	88	54.0	259	64.75	Pass
1	2	B	68.0	34	28	28.0	158	39.50	Fail
2	3	C	98.0	59	67	67.0	291	72.75	Pass
3	4	D	50.0	79	91	72.0	292	73.00	Pass
4	5	E	56.0	67	35	78.0	236	59.00	Pass
5	6	F	79.0	72	43	72.0	266	66.50	Pass
6	7	G	71.0	69	72	83.0	295	73.75	Pass
7	8	H	75.0	93	87	200.0	455	113.75	Pass

```
df['DSBDA']=df['DSBDA'].fillna(df['DSBDA'].mean())

df['WT']=df['WT'].fillna(df['WT'].mean())
```

11	12	L	64.0	88	59	1.0	211	52.75	False

```
df.head(16)
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
0	1	A	50.00	67	88	54.00	259	64.75	Pass
1	2	B	68.00	34	28	28.00	158	39.50	Fail
2	3	C	98.00	59	67	67.00	291	72.75	Pass
3	4	D	50.00	79	91	72.00	292	73.00	Pass
4	5	E	56.00	67	35	78.00	236	59.00	Pass
5	6	F	79.00	72	43	72.00	266	66.50	Pass
6	7	G	71.00	69	72	83.00	295	73.75	Pass
7	8	H	75.00	93	87	200.00	455	113.75	Pass
8	9	I	30.00	89	93	60.00	272	68.00	Fail
9	10	J	57.00	56	70	32.00	215	53.75	Fail
10	11	K	68.00	45	63	65.00	241	60.25	Pass
11	12	L	64.00	88	59	69.84	211	52.75	False
12	13	M	63.00	41	59	66.00	229	57.25	Pass
13	14	N	45.00	36	80	58.00	219	54.75	Pass
14	15	O	75.00	78	36	84.00	273	68.25	Pass
15	16	P	62.04	82	36	33.00	151	37.75	Pass

```
df.dropna()
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
0	1	A	50.00	67	88	54.00	259	64.75	Pass
1	2	B	68.00	34	28	28.00	158	39.50	Fail
2	3	C	98.00	59	67	67.00	291	72.75	Pass
3	4	D	50.00	79	91	72.00	292	73.00	Pass
4	5	E	56.00	67	35	78.00	236	59.00	Pass
5	6	F	79.00	72	43	72.00	266	66.50	Pass
6	7	G	71.00	69	72	83.00	295	73.75	Pass
7	8	H	75.00	93	87	200.00	455	113.75	Pass
8	9	I	30.00	89	93	60.00	272	68.00	Fail
9	10	J	57.00	56	70	32.00	215	53.75	Fail

```
df.dropna(how="all")
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
0	1	A	50.00	67	88	54.00	259	64.75	Pass
1	2	B	68.00	34	28	28.00	158	39.50	Fail
2	3	C	98.00	59	67	67.00	291	72.75	Pass
3	4	D	50.00	79	91	72.00	292	73.00	Pass
4	5	E	56.00	67	35	78.00	236	59.00	Pass
5	6	F	79.00	72	43	72.00	266	66.50	Pass
6	7	G	71.00	69	72	83.00	295	73.75	Pass
7	8	H	75.00	93	87	200.00	455	113.75	Pass
8	9	I	30.00	89	93	60.00	272	68.00	Fail
9	10	J	57.00	56	70	32.00	215	53.75	Fail
10	11	K	68.00	45	63	65.00	241	60.25	Pass
11	12	L	64.00	88	59	69.84	211	52.75	False
12	13	M	63.00	41	59	66.00	229	57.25	Pass
13	14	N	45.00	36	80	58.00	219	54.75	Pass
14	15	O	75.00	78	36	84.00	273	68.25	Pass
15	16	P	62.04	82	36	33.00	151	37.75	Pass
16	17	Q	58.00	78	66	80.00	282	70.50	Pass
17	18	R	72.00	82	33	83.00	270	67.50	Fail
18	19	S	41.00	42	31	52.00	166	41.50	Fail
19	20	T	55.00	59	59	88.00	261	65.25	Pass
20	21	U	37.00	37	33	66.00	173	43.25	Fail
21	22	V	78.00	52	65	71.00	266	66.50	Pass
22	23	W	75.00	64	62	38.00	239	59.75	Pass
23	24	X	56.00	84	44	90.00	274	68.50	Pass
24	25	Y	85.00	78	96	81.00	340	85.00	Pass
25	26	Z	45.00	85	45	45.00	220	55.00	Pass

```
df.dropna(axis=1)
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
0	1	A	50.00	67	88	54.00	259	64.75	Pass
1	2	B	68.00	34	28	28.00	158	39.50	Fail
2	3	C	98.00	59	67	67.00	291	72.75	Pass
3	4	D	50.00	79	91	72.00	292	73.00	Pass
4	5	E	56.00	67	35	78.00	236	59.00	Pass
5	6	F	79.00	72	43	72.00	266	66.50	Pass
6	7	G	71.00	69	72	83.00	295	73.75	Pass
7	8	H	75.00	93	87	200.00	455	113.75	Pass
8	9	I	30.00	89	93	60.00	272	68.00	Fail
9	10	J	57.00	56	70	32.00	215	53.75	Fail
10	11	K	68.00	45	63	65.00	241	60.25	Pass
11	12	L	64.00	88	59	69.84	211	52.75	False
12	13	M	63.00	41	59	66.00	229	57.25	Pass
13	14	N	45.00	36	80	58.00	219	54.75	Pass
14	15	O	75.00	78	36	84.00	273	68.25	Pass
15	16	P	62.04	82	36	33.00	151	37.75	Pass
16	17	Q	58.00	78	66	80.00	282	70.50	Pass
17	18	R	72.00	82	33	83.00	270	67.50	Fail

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

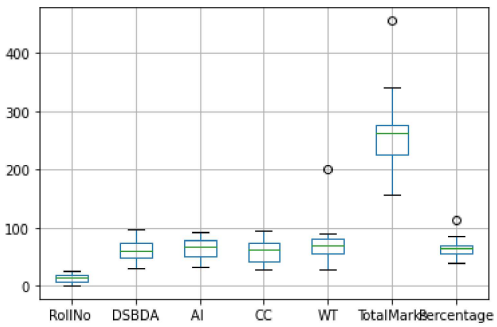
```
df
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
0	1	A	50.0	67	88	54.0	259	64.75	Pass
1	2	B	68.0	34	28	28.0	158	39.50	Fail
2	3	C	98.0	59	67	67.0	291	72.75	Pass
3	4	D	50.0	79	91	72.0	292	73.00	Pass
4	5	E	56.0	67	35	78.0	236	59.00	Pass
5	6	F	79.0	72	43	72.0	266	66.50	Pass
6	7	G	71.0	69	72	83.0	295	73.75	Pass
7	8	H	75.0	93	87	200.0	455	113.75	Pass
8	9	I	30.0	89	93	60.0	272	68.00	Fail
9	10	J	57.0	56	70	32.0	215	53.75	Fail
10	11	K	68.0	45	63	65.0	241	60.25	Pass
12	13	M	63.0	41	59	66.0	229	57.25	Pass
13	14	N	45.0	36	80	58.0	219	54.75	Pass
14	15	O	75.0	78	36	84.0	273	68.25	Pass
16	17	Q	58.0	78	66	80.0	282	70.50	Pass
17	18	R	72.0	82	33	83.0	270	67.50	Fail
18	19	S	41.0	42	31	52.0	166	41.50	Fail
19	20	T	55.0	59	59	88.0	261	65.25	Pass
20	21	U	37.0	37	33	66.0	173	43.25	Fail
21	22	V	78.0	52	65	71.0	266	66.50	Pass
22	23	W	75.0	64	62	38.0	239	59.75	Pass
23	24	X	56.0	84	44	90.0	274	68.50	Pass
24	25	Y	85.0	78	96	81.0	340	85.00	Pass
25	26	Z	45.0	85	45	45.0	220	55.00	Pass

Identification and Handling of Outliers

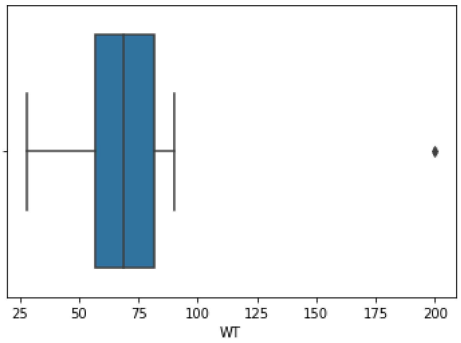
```
import seaborn as sns
import matplotlib.pyplot as plt


df.boxplot()
```

```
<AxesSubplot:>
```



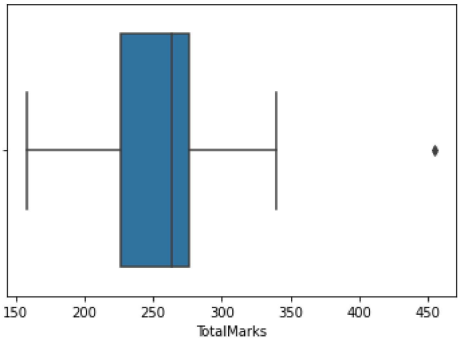

```
sns.boxplot(x=df.WT)
```

```
<AxesSubplot:xlabel='WT'>
```



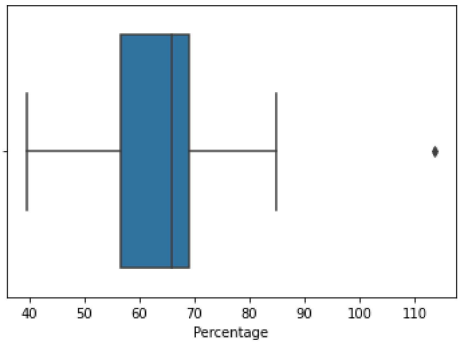

```
sns.boxplot(x=df.TotalMarks)
```

```
<AxesSubplot:xlabel='TotalMarks'>
```



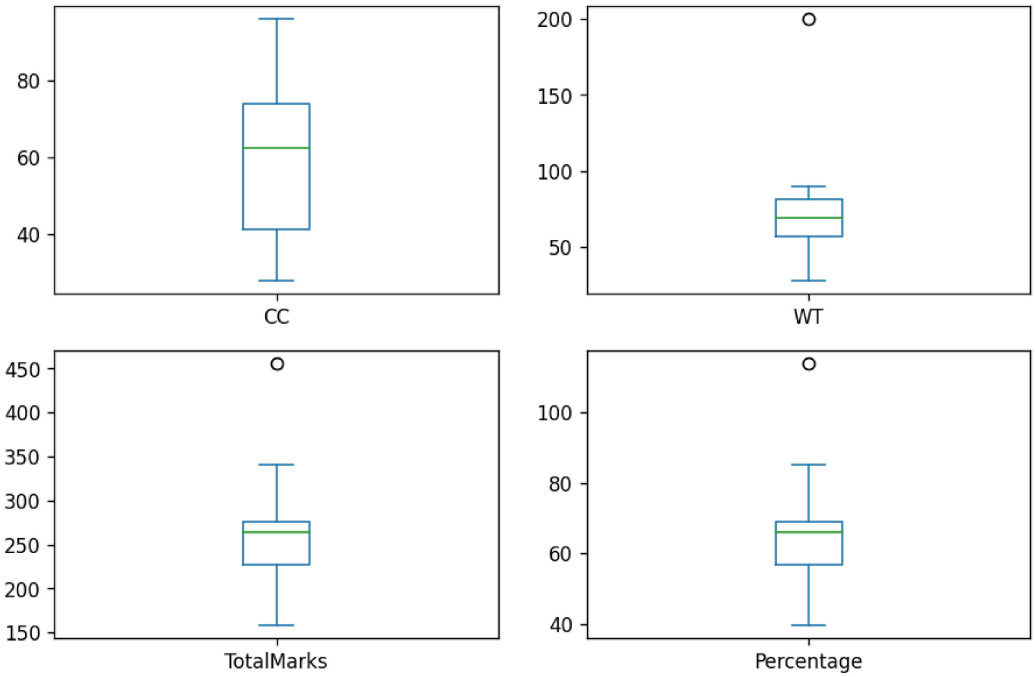

```
sns.boxplot(x=df.Percentage)
```

```
<AxesSubplot:xlabel='Percentage'>
```




```
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (9, 6)
df_list = ['CC','WT','TotalMarks','Percentage']
fig, axes = plt.subplots(2, 2)
fig.set_dpi(120)

count=0
```

```
for r in range(2):
   for c in range(2):
       _ = df[df_list[count]].plot(kind = 'box', ax=axes[r,c])
       count+=1
```



Detecting outliers using Inter Quantile Range(IQR):

```
Q1 = df['WT'].quantile(0.25)
Q3 = df['WT'].quantile(0.75)
IQR = Q3 - Q1
Lower_limit = Q1 - 1.5 * IQR
Upper_limit = Q3 + 1.5 * IQR
print(f'Q1 = {Q1}, Q3 = {Q3}, IQR = {IQR}, Lower_limit = {Lower_limit}, Upper_limit = {Upper_limit}')
```

```
Q1 = 57.0, Q3 = 81.5, IQR = 24.5, Lower_limit = 20.25, Upper_limit = 118.25
```

```
Q1 = df['TotalMarks'].quantile(0.25)
Q3 = df['TotalMarks'].quantile(0.75)
IQR = Q3 - Q1
Lower_limit = Q1 - 1.5 * IQR
Upper_limit = Q3 + 1.5 * IQR
print(f'Q1 = {Q1}, Q3 = {Q3}, IQR = {IQR}, Lower_limit = {Lower_limit}, Upper_limit = {Upper_limit}')
```

```
Q1 = 226.75, Q3 = 276.0, IQR = 49.25, Lower_limit = 152.875, Upper_limit = 349.875
```

```
Q1 = df['Percentage'].quantile(0.25)
Q3 = df['Percentage'].quantile(0.75)
IQR = Q3 - Q1
Lower_limit = Q1 - 1.5 * IQR
Upper_limit = Q3 + 1.5 * IQR
print(f'Q1 = {Q1}, Q3 = {Q3}, IQR = {IQR}, Lower_limit = {Lower_limit}, Upper_limit = {Upper_limit}')
```

```
Q1 = 56.6875, Q3 = 69.0, IQR = 12.3125, Lower_limit = 38.21875, Upper_limit = 87.46875
```

```
df[(df['WT'] < Lower_limit) | (df['WT'] > Upper_limit)]
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result

```
df[(df['TotalMarks'] < Lower_limit) | (df['TotalMarks'] > Upper_limit)]
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
0	1	A	50.0	67	88	54.0	259	64.75	Pass
1	2	B	68.0	34	28	28.0	158	39.50	Fail
2	3	C	98.0	59	67	67.0	291	72.75	Pass
3	4	D	50.0	79	91	72.0	292	73.00	Pass
4	5	E	56.0	67	35	78.0	236	59.00	Pass
5	6	F	79.0	72	43	72.0	266	66.50	Pass
6	7	G	71.0	69	72	83.0	295	73.75	Pass
7	8	H	75.0	93	87	200.0	455	113.75	Pass
8	9	I	30.0	89	93	60.0	272	68.00	Fail
9	10	J	57.0	56	70	32.0	215	53.75	Fail
10	11	K	68.0	45	63	65.0	241	60.25	Pass
12	13	M	63.0	41	59	66.0	229	57.25	Pass
13	14	N	45.0	36	80	58.0	219	54.75	Pass
14	15	O	75.0	78	36	84.0	273	68.25	Pass
16	17	Q	58.0	78	66	80.0	282	70.50	Pass
17	18	R	72.0	82	33	83.0	270	67.50	Fail
18	19	S	41.0	42	31	52.0	166	41.50	Fail
19	20	T	55.0	59	59	88.0	261	65.25	Pass
20	21	U	37.0	37	33	66.0	173	43.25	Fail
21	22	V	78.0	52	65	71.0	266	66.50	Pass
22	23	W	75.0	64	62	38.0	239	59.75	Pass
23	24	X	56.0	84	44	90.0	274	68.50	Pass
24	25	Y	85.0	78	96	81.0	340	85.00	Pass
25	26	Z	45.0	85	45	45.0	220	55.00	Pass

```
df[(df['Percentage'] < Lower_limit) | (df['Percentage'] > Upper_limit)]
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
7	8	H	75.0	93	87	200.0	455	113.75	Pass

Handling of Outliers

Removing the outlier:

```
outliers_WT=[]
for i in df.WT:
    if i<Lower_limit or i>Upper_limit:
       outliers_WT.append(i)
print("outliers are",outliers_WT)
```

```
outliers are [28.0, 200.0, 32.0, 88.0, 38.0, 90.0]
```

```
outliers_TotM=[]
for i in df.TotalMarks:
    if i<Lower_limit or i>Upper_limit:
       outliers_TotM.append(i)
print("outliers are",outliers_TotM)
```

```
outliers are [259, 158, 291, 292, 236, 266, 295, 455, 272, 215, 241, 229, 219, 273, 282, 270, 166, 261, 173, 266, 239, 274, 340, 220]
```

```
outliers_Perc=[]
for i in df.Percentage:
    if i<Lower_limit or i>Upper_limit:
       outliers_Perc.append(i)
print("outliers are",outliers_Perc)
```

```
outliers are [113.75]
```

```
df[df.WT<Lower_limit].index
```

```
Int64Index([1, 9, 22], dtype='int64')
```

```
df[df.TotalMarks<Lower_limit].index
```

```
Int64Index([], dtype='int64')
```

```
df[df.Percentage<Lower_limit].index
```

```
Int64Index([], dtype='int64')
```

```
df1=df.drop(df[df.WT<Lower_limit].index)
```

```
df1=df.drop(df[df.Percentage<Lower_limit].index)
```

```
df1=df.drop(df[df.TotalMarks<Lower_limit].index)
```

```
df1.shape
```

```
(24, 9)
```

```
df2=df[df.WT<Lower_limit]
df2
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result
1	2	B	68.0	34	28	28.0	158	39.50	Fail
9	10	J	57.0	56	70	32.0	215	53.75	Fail
22	23	W	75.0	64	62	38.0	239	59.75	Pass

```
df2=df[df.TotalMarks<Lower_limit]
df2
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result

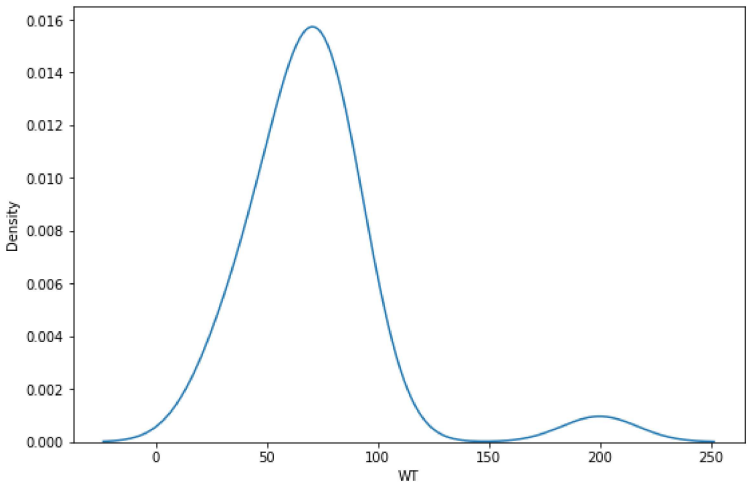
```
df2=df[df.Percentage<Lower_limit]
df2
```

	RollNo	Name	DSBDA	AI	CC	WT	TotalMarks	Percentage	Result

Mean/Median imputation

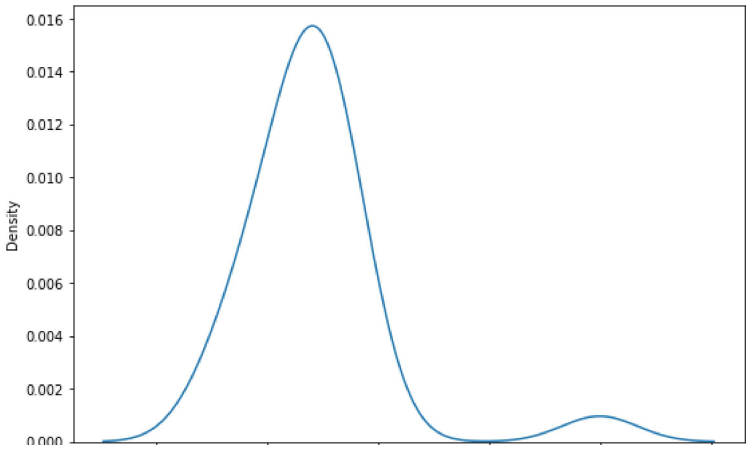
```
sns.kdeplot(df.WT)
```

```
<AxesSubplot:xlabel='WT', ylabel='Density'>
```



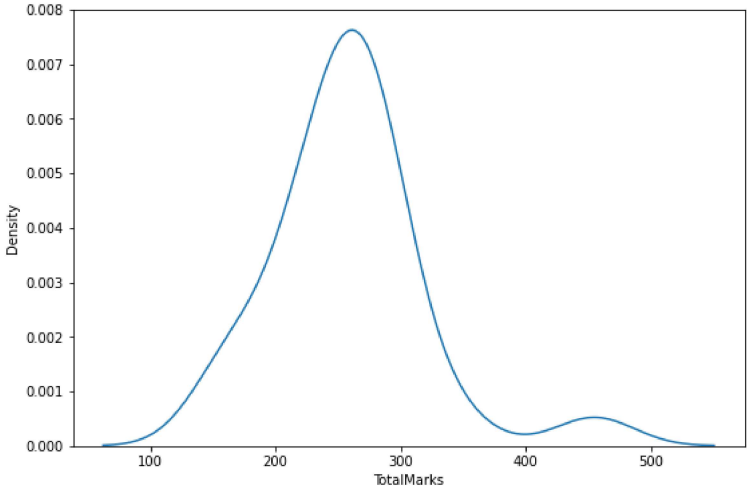

```
sns.kdeplot(df1.WT)
```

```
<AxesSubplot:xlabel='WT', ylabel='Density'>
```



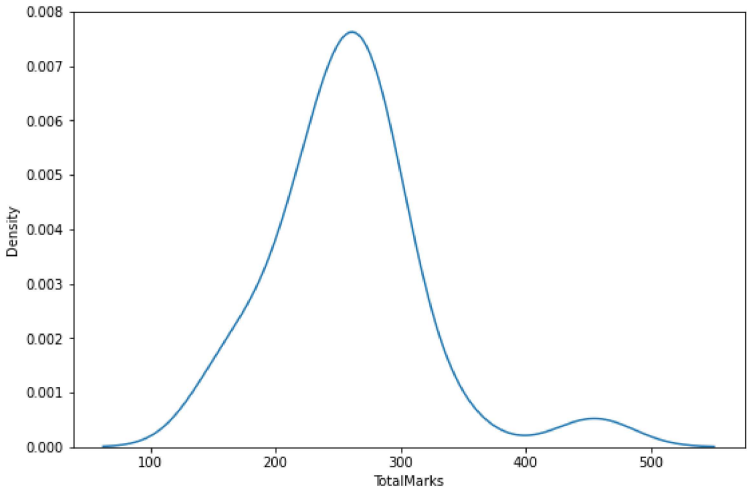

```
sns.kdeplot(df.TotalMarks)
```

```
<AxesSubplot:xlabel='TotalMarks', ylabel='Density'>
```



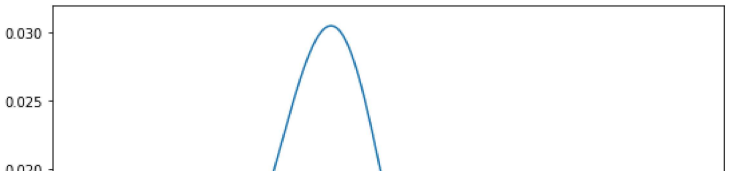

```
sns.kdeplot(df1.TotalMarks)
```

```
<AxesSubplot:xlabel='TotalMarks', ylabel='Density'>
```



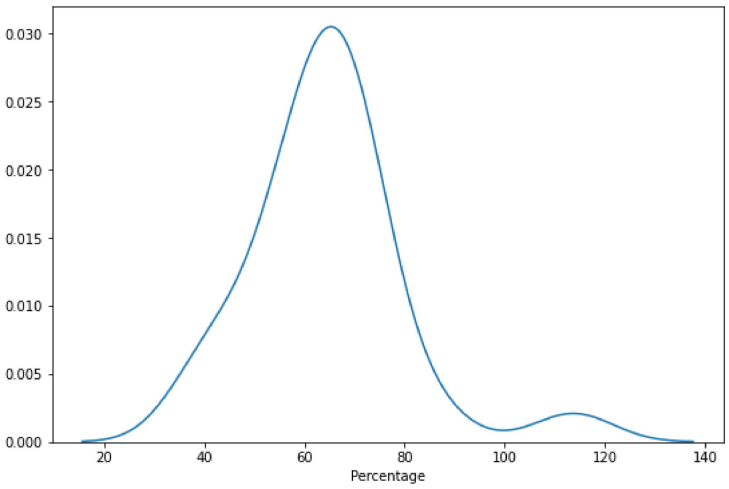

```
sns.kdeplot(df.Percentage)
```

```
<AxesSubplot:xlabel='Percentage', ylabel='Density'>
```




```
sns.kdeplot(df1.Percentage)
```

```
<AxesSubplot:xlabel='Percentage', ylabel='Density'>
```




```
df.WT
```

```
0      54.0
1      28.0
2      67.0
3      72.0
4      78.0
5      72.0
6      83.0
7     200.0
8      60.0
9      32.0
10     65.0
12     66.0
13     58.0
14     84.0
16     80.0
17     83.0
18     52.0
19     88.0
20     66.0
21     71.0
22     38.0
23     90.0
24     81.0
25     45.0
Name: WT, dtype: float64
```

```
df.TotalMarks
```

```
0     259
1     158
2     291
3     292
4     236
5     266
6     295
7     455
8     272
9     215
10    241
12    229
13    219
14    273
16    282
17    270
18    166
19    261
20    173
21    266
22    239
23    274
24    340
```

```
25    220
Name: TotalMarks, dtype: int64
```

```
df.Percentage
```

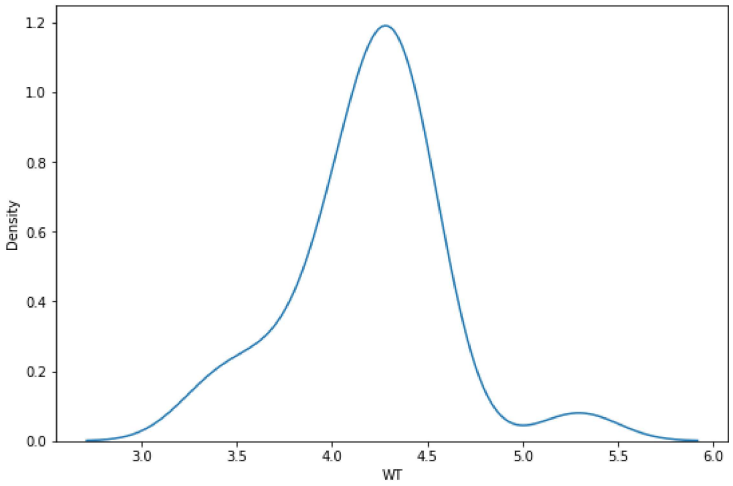
```
0      64.75
1      39.50
2      72.75
3      73.00
4      59.00
5      66.50
6      73.75
7     113.75
8      68.00
9      53.75
10     60.25
12     57.25
13     54.75
14     68.25
16     70.50
17     67.50
18     41.50
19     65.25
20     43.25
21     66.50
22     59.75
23     68.50
24     85.00
25     55.00
Name: Percentage, dtype: float64
```

```
log_wt=np.log(df.WT)
log_wt
```

```
0     3.988984
1     3.332205
2     4.204693
3     4.276666
4     4.356709
5     4.276666
6     4.418841
7     5.298317
8     4.094345
9     3.465736
10    4.174387
12    4.189655
13    4.060443
14    4.430817
16    4.382027
17    4.418841
18    3.951244
19    4.477337
20    4.189655
21    4.262680
22    3.637586
23    4.499810
24    4.394449
25    3.806662
Name: WT, dtype: float64
```

```
sns.kdeplot(log_wt)
```

```
<AxesSubplot:xlabel='WT', ylabel='Density'>
```

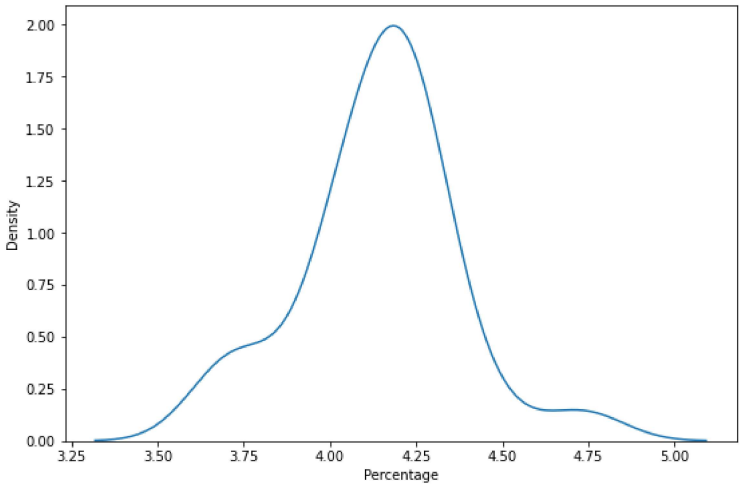



```
log_perc=np.log(df.Percentage)
log_perc
```

```
0     4.170534
1     3.676301
2     4.287029
3     4.290459
4     4.077537
5     4.197202
6     4.300681
7     4.734003
8     4.219508
9     3.984344
10    4.098503
12    4.047428
13    4.002777
14    4.223177
16    4.255613
17    4.212128
18    3.725693
19    4.178226
20    3.766997
21    4.197202
22    4.090169
23    4.226834
24    4.442651
25    4.007333
Name: Percentage, dtype: float64
```

```
sns.kdeplot(log_perc)
```

```
<AxesSubplot:xlabel='Percentage', ylabel='Density'>
```

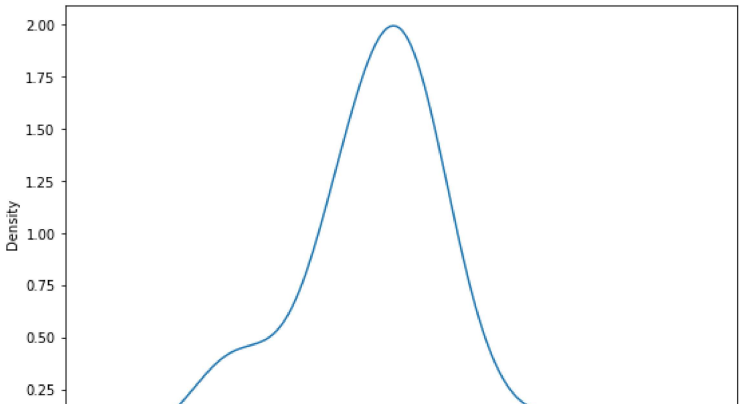



```
log_tm=np.log(df.TotalMarks)
log_tm
```

```
0     5.556828
1     5.062595
2     5.673323
3     5.676754
4     5.463832
5     5.583496
6     5.686975
7     6.120297
8     5.605802
9     5.370638
10    5.484797
12    5.433722
13    5.389072
14    5.609472
16    5.641907
17    5.598422
18    5.111988
19    5.564520
20    5.153292
21    5.583496
22    5.476464
23    5.613128
24    5.828946
25    5.393628
Name: TotalMarks, dtype: float64
```

```
sns.kdeplot(log_tm)
```

```
<AxesSubplot:xlabel='TotalMarks', ylabel='Density'>
```



DATA TRANSFORMATION

Checking the distribution with Skewness

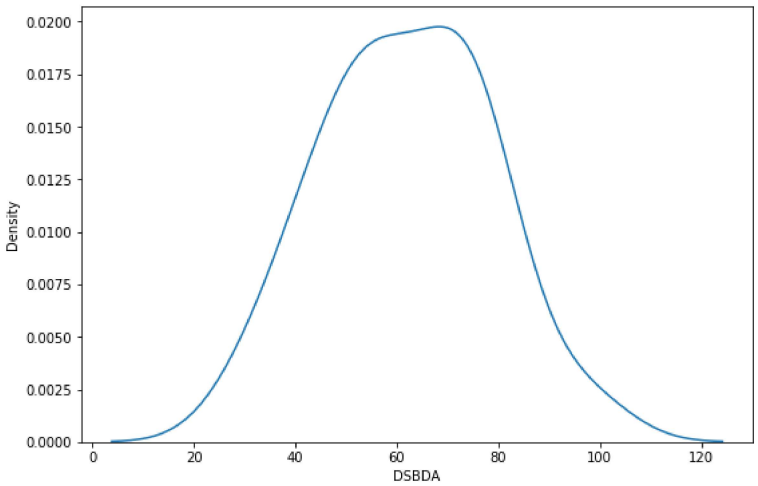
```
import seaborn as sns
```

```
df.skew()
```

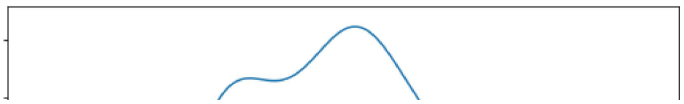
```
<ipython-input-55-9e0b1e29546f>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a f
  df.skew()
RollNo        0.015995
DSBDA         0.057892
AI           -0.275917
CC            0.086725
WT            2.691079
TotalMarks    1.247048
Percentage    1.247048
dtype: float64
```

Checking the distribution of variables using KDE plot

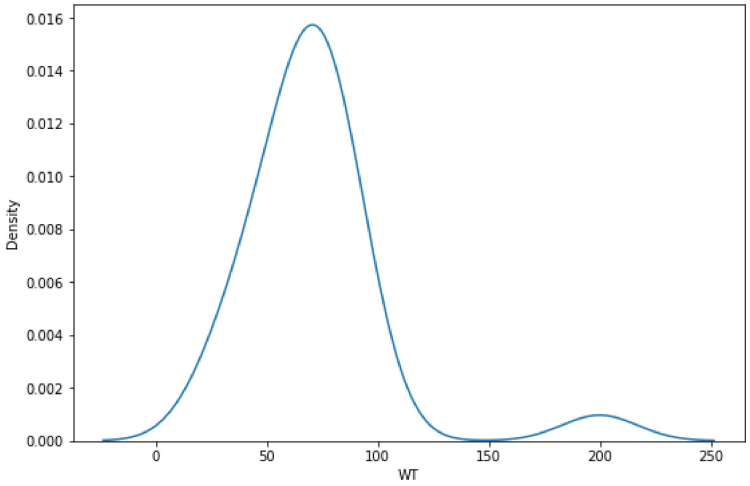
```
sns.kdeplot(df.DSBDA);
```



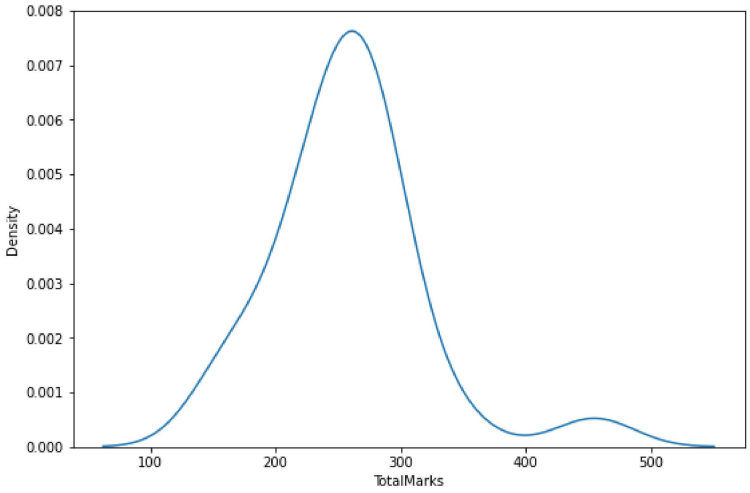

```
sns.kdeplot(df.CC);
```



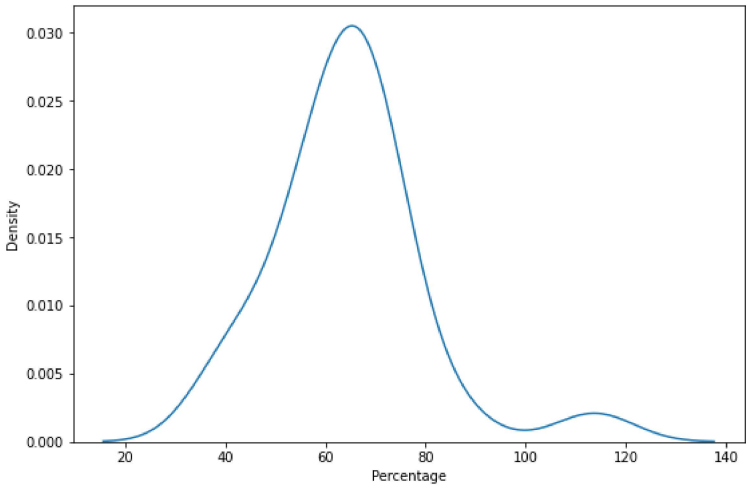

```
sns.kdeplot(df.WT);
```




```
sns.kdeplot(df.TotalMarks);
```




```
sns.kdeplot(df.Percentage);
```



DATA ANALYTICS-I

1. Problem Statement

Perform the following operations on any open source dataset (e.g., data.csv)

1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variable. For example, if your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable.
2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris-versicolor' of iris.csv dataset.

Basic Operations

```
import pandas as pd
import numpy as np
```

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

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Iris.csv to Iris.csv
{'Iris.csv':
b'Id,SepalLengthCm,SepalWidthCm,PetalLengthCm,PetalWidthCm,Species\n1,5.1,3.5,1.4,0.2,Iris-
setosa\n2,4.9,3.0,1.4,0.2,Iris-setosa\n3,4.7,3.2,1.3,0.2,Iris-setosa\n4,4.6,3.1,1.5,0.2,Iris-
setosa\n5,5.0,3.6,1.4,0.2,Iris-setosa\n6,5.4,3.9,1.7,0.4,Iris-setosa\n7,4.6,3.4,1.4,0.3,Iris-
setosa\n8,5.0,3.4,1.5,0.2,Iris-setosa\n9,4.4,2.9,1.4,0.2,Iris-setosa\n10,4.9,3.1,1.5,0.1,Iris-
setosa\n11,5.4,3.7,1.5,0.2,Iris-setosa\n12,4.8,3.4,1.6,0.2,Iris-setosa\n13,4.8,3.0,1.4,0.1,Iris-
setosa\n14,4.3,3.0,1.1,0.1,Iris-setosa\n15,5.8,4.0,1.2,0.2,Iris-setosa\n16,5.7,4.4,1.5,0.4,Iris-
setosa\n17,5.4,3.9,1.3,0.4,Iris-setosa\n18,5.1,3.5,1.4,0.3,Iris-setosa\n19,5.7,3.8,1.7,0.3,Iris-
setosa\n20,5.1,3.8,1.5,0.3,Iris-setosa\n21,5.4,3.4,1.7,0.2,Iris-setosa\n22,5.1,3.7,1.5,0.4,Iris-
setosa\n23,4.6,3.6,1.0,0.2,Iris-setosa\n24,5.1,3.3,1.7,0.5,Iris-setosa\n25,4.8,3.4,1.9,0.2,Iris-
setosa\n26,5.0,3.0,1.6,0.2,Iris-setosa\n27,5.0,3.4,1.6,0.4,Iris-setosa\n28,5.2,3.5,1.5,0.2,Iris-
setosa\n29,5.2,3.4,1.4,0.2,Iris-setosa\n30,4.7,3.2,1.6,0.2,Iris-setosa\n31,4.8,3.1,1.6,0.2,Iris-
setosa\n32,5.4,3.4,1.5,0.4,Iris-setosa\n33,5.2,4.1,1.5,0.1,Iris-setosa\n34,5.5,4.2,1.4,0.2,Iris-
setosa\n35,4.9,3.1,1.5,0.1,Iris-setosa\n36,5.0,3.2,1.2,0.2,Iris-setosa\n37,5.5,3.5,1.3,0.2,Iris-
setosa\n38,4.9,3.1,1.5,0.1,Iris-setosa\n39,4.4,3.0,1.3,0.2,Iris-setosa\n40,5.1,3.4,1.5,0.2,Iris-
setosa\n41,5.0,3.5,1.3,0.3,Iris-setosa\n42,4.5,2.3,1.3,0.3,Iris-setosa\n43,4.4,3.2,1.3,0.2,Iris-
setosa\n44,5.0,3.5,1.6,0.6,Iris-setosa\n45,5.1,3.8,1.9,0.4,Iris-setosa\n46,4.8,3.0,1.4,0.3,Iris-
setosa\n47,5.1,3.8,1.6,0.2,Iris-setosa\n48,4.6,3.2,1.4,0.2,Iris-setosa\n49,5.3,3.7,1.5,0.2,Iris-
setosa\n50,5.0,3.3,1.4,0.2,Iris-setosa\n51,7.0,3.2,4.7,1.4,Iris-
versicolor\n52,6.4,3.2,4.5,1.5,Iris-versicolor\n53,6.9,3.1,4.9,1.5,Iris-
versicolor\n54,5.5,2.3,4.0,1.3,Iris-versicolor\n55,6.5,2.8,4.6,1.5,Iris-
versicolor\n56,5.7,2.8,4.5,1.3,Iris-versicolor\n57,6.3,3.3,4.7,1.6,Iris-
versicolor\n58,4.9,2.4,3.3,1.0,Iris-versicolor\n59,6.6,2.9,4.6,1.3,Iris-
versicolor\n60,5.2,2.7,3.9,1.4,Iris-versicolor\n61,5.0,2.0,3.5,1.0,Iris-
versicolor\n62,5.9,3.0,4.2,1.5,Iris-versicolor\n63,6.0,2.2,4.0,1.0,Iris-
versicolor\n64,6.1,2.9,4.7,1.4,Iris-versicolor\n65,5.6,2.9,3.6,1.3,Iris-
versicolor\n66,6.7,3.1,4.4,1.4,Iris-versicolor\n67,5.6,3.0,4.5,1.5,Iris-
versicolor\n68,5.8,2.7,4.1,1.0,Iris-versicolor\n69,6.2,2.2,4.5,1.5,Iris-
versicolor\n70,5.6,2.5,3.9,1.1,Iris-versicolor\n71,5.9,3.2,4.8,1.8,Iris-
versicolor\n72,6.1,2.8,4.0,1.3,Iris-versicolor\n73,6.3,2.5,4.9,1.5,Iris-
versicolor\n74,6.1,2.8,4.7,1.2,Iris-versicolor\n75,6.4,2.9,4.3,1.3,Iris-
```

2. Data Collection

```
df=pd.read_csv("/content/Iris.csv")
```

```
df
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa

3. Exploratory Data Analysis

Data Preprocessing:

145	146	6.7	3.0	5.2	2.3	Iris-virginica

```
df.head()
```

	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

```
df.info()
```

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

```
df.describe()
```

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

Mean

```
df.mean()
```

```
<ipython-input-6-c61f0c8f89b5>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a fu
  df.mean()
Id               75.500000
SepalLengthCm     5.843333
SepalWidthCm      3.054000
PetalLengthCm     3.758667
PetalWidthCm      1.198667
dtype: float64
```

```
df.loc[:,'SepalLengthCm'].mean()
```

```
5.843333333333334
```

```
df.mean(axis=1)[0:4]
```

```
<ipython-input-8-6447142f9231>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a fu
  df.mean(axis=1)[0:4]
0    2.24
1    2.30
2    2.48
3    2.68
dtype: float64
```

Median

```
df.median()
```

```
<ipython-input-10-6d467abf240d>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a f
  df.median()
Id               75.50
SepalLengthCm     5.80
SepalWidthCm      3.00
PetalLengthCm     4.35
PetalWidthCm      1.30
dtype: float64
```

```
df.loc[:,'SepalWidthCm'].median()
```

```
3.0
```

```
df.loc[:,'PetalLengthCm'].median()
```

```
4.35
```

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

```
<ipython-input-14-ccd22b6541c9>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a f
  df.median(axis=1)[0:4]
0    1.4
1    2.0
2    3.0
3    3.1
dtype: float64
```

Mode

```
df.mode()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.0	3.0	1.5	0.2	Iris-setosa
1	2	NaN	NaN	NaN	NaN	Iris-versicolor
2	3	NaN	NaN	NaN	NaN	Iris-virginica
3	4	NaN	NaN	NaN	NaN	NaN
4	5	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...
145	146	NaN	NaN	NaN	NaN	NaN
146	147	NaN	NaN	NaN	NaN	NaN
147	148	NaN	NaN	NaN	NaN	NaN
148	149	NaN	NaN	NaN	NaN	NaN
149	150	NaN	NaN	NaN	NaN	NaN

150 rows × 6 columns

```
df.loc[:,'SepalWidthCm'].mode()
```

```
0    3.0
dtype: float64
```

Min

```
df.min()
```

```
Id                          1
SepalLengthCm             4.3
SepalWidthCm              2.0
PetalLengthCm             1.0
PetalWidthCm              0.1
Species           Iris-setosa
dtype: object
```

```
df.loc[:,'SepalWidthCm'].min(skipna = False)
```

```
2.0
```

Max

```
df.max()
```

```
Id                          150
SepalLengthCm               7.9
SepalWidthCm                4.4
PetalLengthCm               6.9
PetalWidthCm                2.5
Species           Iris-virginica
dtype: object
```

std

```
df.std()
```

```
<ipython-input-20-ce97bb7eaef8>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a f
  df.std()
Id               43.445368
SepalLengthCm     0.828066
SepalWidthCm      0.433594
PetalLengthCm     1.764420
PetalWidthCm      0.763161
dtype: float64
```

```
df.loc[:,'SepalWidthCm'].std()
```

```
0.4335943113621737
```

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

```
<ipython-input-22-18b37211c5c6>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a f
  df.std(axis=1)[0:4]
0    2.010721
1    1.772005
2    1.754138
3    1.813009
dtype: float64
```

groupby

```
df.groupby(['SepalLengthCm'])['PetalLengthCm'].mean()
```

```
SepalLengthCm
4.3    1.100000
4.4    1.333333
4.5    1.300000
4.6    1.325000
4.7    1.450000
4.8    1.580000
4.9    2.283333
5.0    1.840000
5.1    1.722222
5.2    2.075000
5.3    1.500000
5.4    2.033333
5.5    3.228571
5.6    4.200000
5.7    3.587500
5.8    4.071429
```

```
5.9    4.700000
6.0    4.650000
6.1    4.750000
6.2    4.750000
6.3    5.133333
6.4    5.157143
6.5    5.240000
6.6    4.500000
6.7    5.262500
6.8    5.400000
6.9    5.275000
7.0    4.700000
7.1    5.900000
7.2    5.966667
7.3    6.300000
7.4    6.100000
7.6    6.600000
7.7    6.600000
7.9    6.400000
Name: PetalLengthCm, dtype: float64
```

To create a list that contains a numeric value for each response to the categorical variable.

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

	0	1	2	3	4	5	6	7	8	9	...	33	34	35	36	37	38	39	40	41	42
0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
145	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
146	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
147	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
148	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
149	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

150 rows × 43 columns

Load all rows with Iris-versicolor species in variable irisVer

```
irisVer = (df['Species']== 'Iris-versicolor')
```

To display basic statistical details like percentile, mean,standard deviation etc. forIris- versicolor use describe

```
print(df[irisVer].describe())
```

```
                Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count    50.00000      50.000000     50.000000      50.000000     50.000000
mean     75.50000       5.936000      2.770000       4.260000      1.326000
std      14.57738       0.516171      0.313798       0.469911      0.197753
min      51.00000       4.900000      2.000000       3.000000      1.000000
25%      63.25000       5.600000      2.525000       4.000000      1.200000
50%      75.50000       5.900000      2.800000       4.350000      1.300000
75%      87.75000       6.300000      3.000000       4.600000      1.500000
max     100.00000       7.000000      3.400000       5.100000      1.800000
```

Percentile

```
ninetieth_percentile = np.percentile(df['SepalLengthCm'], 90)
```

```
ninetieth_percentile
```

```
6.9
```

```
np.percentile(df['SepalWidthCm'], 90)
```

```
3.6099999999999994
```

```
np.percentile(df['PetalLengthCm'], 90)
```

```
5.8
```

```
np.percentile(df['PetalWidthCm'], 90)
```

```
2.2
```

```
np.percentile(df['SepalLengthCm'], 25)
```

```
5.1
```

```
np.percentile(df['SepalLengthCm'], 75)
```

```
6.4
```

```
np.percentile(df['SepalLengthCm'], 50)
```

```
5.8
```

```
np.percentile(df['SepalWidthCm'], 75)
```

```
3.3
```

```
np.percentile(df['SepalWidthCm'], 25)
```

```
2.8
```

```
np.percentile(df['SepalWidthCm'], 50)
```

```
3.0
```

```
np.percentile(df['PetalLengthCm'], 75)
```

```
5.1
```

```
np.percentile(df['PetalLengthCm'], 25)
```

```
1.6
```

```
np.percentile(df['PetalLengthCm'], 50)
```

```
4.35
```

```
np.percentile(df['PetalWidthCm'], 75)
```

```
1.8
```

```
np.percentile(df['PetalWidthCm'], 50)
```

```
1.3
```

```
np.percentile(df['PetalWidthCm'], 25)
```

```
0.3
```

All obove on income dataset

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

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving income.csv to income.csv
{'income.csv':
b'ID,Gender,Age,Income\n1,Female,25,15632\n2,Female,26,36521\n3,Female,38,25638\n4,Male,26,25689\n5,Mal
```

```
df2 = pd.read_csv('/content/income.csv')
```

```
df2
```

	ID	Gender	Age	Income
0	1	Female	25	15632
1	2	Female	26	36521
2	3	Female	38	25638
3	4	Male	26	25689
4	5	Male	21	36689
5	6	Female	23	45789
6	7	Female	28	65897
7	8	Female	24	25896
8	9	Female	25	36486
9	10	Male	27	35649
10	11	Female	29	25635
11	12	Female	23	32165
12	13	Female	19	45645
13	14	Female	22	56898

```
df2.head()
```

	ID	Gender	Age	Income
0	1	Female	25	15632
1	2	Female	26	36521
2	3	Female	38	25638
3	4	Male	26	25689
4	5	Male	21	36689

```
df2.describe(include ="all")
```

	ID	Gender	Age	Income
count	14.0000	14	14.000000	14.000000
unique	NaN	2	NaN	NaN
top	NaN	Female	NaN	NaN
freq	NaN	11	NaN	NaN
mean	7.5000	NaN	25.428571	36444.928571
std	4.1833	NaN	4.535574	13503.606678
min	1.0000	NaN	19.000000	15632.000000
25%	4.2500	NaN	23.000000	25740.750000
50%	7.5000	NaN	25.000000	36067.500000
75%	10.7500	NaN	26.750000	43406.000000
max	14.0000	NaN	38.000000	65897.000000

```
df2.mean()
```

```
<ipython-input-11-4a56ca50fe8e>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a f
  df2.mean()
ID            7.500000
Age          25.428571
Income    36444.928571
dtype: float64
```

```
df2.mean()
```

```
<ipython-input-13-4a56ca50fe8e>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a f
  df2.mean()
ID            7.500000
Age          25.428571
Income    36444.928571
dtype: float64
```

```
df2.std()
```

```
<ipython-input-14-cefa5b2080f7>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a f
  df2.std()
ID            4.183300
Age           4.535574
Income    13503.606678
dtype: float64
```

```
df2.mean(axis=1)[0:4]
```

```
<ipython-input-15-577d107b385e>:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a f
  df2.mean(axis=1)[0:4]
0     5219.333333
1    12183.000000
2     8559.666667
3     8573.000000
dtype: float64
```

```
df2.min
```

```
<bound method NDFrame._add_numeric_operations.<locals>.min of     ID  Gender  Age  Income
0    1  Female   25   15632
1    2  Female   26   36521
2    3  Female   38   25638
3    4    Male   26   25689
4    5    Male   21   36689
5    6  Female   23   45789
6    7  Female   28   65897
7    8  Female   24   25896
8    9  Female   25   36486
9   10    Male   27   35649
10  11  Female   29   25635
11  12  Female   23   32165
12  13  Female   19   45645
13  14  Female   22   56898>
```

```
df2.max
```

```
<bound method NDFrame._add_numeric_operations.<locals>.max of     ID  Gender  Age  Income
0    1  Female   25   15632
1    2  Female   26   36521
2    3  Female   38   25638
3    4    Male   26   25689
4    5    Male   21   36689
5    6  Female   23   45789
6    7  Female   28   65897
7    8  Female   24   25896
8    9  Female   25   36486
9   10    Male   27   35649
10  11  Female   29   25635
11  12  Female   23   32165
12  13  Female   19   45645
13  14  Female   22   56898>
```

```
df2.quantile(0.25)
```

```
ID            4.25
Age          23.00
Income    25740.75
Name: 0.25, dtype: float64
```

```
df2.quantile(0.50)
```

```
ID            7.5
Age          25.0
Income    36067.5
Name: 0.5, dtype: float64
```

```
df2.quantile(0.75)
```

```
ID           10.75
Age          26.75
Income    43406.00
Name: 0.75, dtype: float64
```

```
df2.groupby(['Gender'])['Age'].mean()
```

```
Gender
Female    25.636364
```

```
Male      24.666667
Name: Age, dtype: float64
```

```
df2.groupby(['Gender'])['Income'].mean()
```

```
Gender
Female    37472.909091
Male      32675.666667
Name: Income, dtype: float64
```

DATA ANALYTICS-I

1. Problem Statement

Perform the following operations on any open source dataset (e.g., data.csv)

1. Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset.

Import libraries and create alias for Pandas, Numpy and Matplotlib

```
import pandas as pd
import numpy as np
```

Import the Boston Housing dataset

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

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving boston_housing.csv to boston_housing.csv
{'boston_housing.csv':
b'crim zn indus chas nox rm age dis rad tax ptratio black lstat medv\n0.00632 18.0 2.31 0 0.538 6
```

2. Data Collection

Loading the data

```
df=pd.read_csv("/content/boston_housing.csv")
```

```
df
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

506 rows × 14 columns

3. Exploratory Data Analysis

Data Preprocessing:

```
df.head()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64  
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64  
 9   tax      506 non-null    float64
 10  ptratio  506 non-null    float64
 11  black    506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

```
df.describe()
```

	crim	zn	indus	chas	nox	rm	age	dis	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.

Data Preprocessing

```
df.isna()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	False	False	False	False	False	False	False	False	False	False	False	False	False	False
502	False	False	False	False	False	False	False	False	False	False	False	False	False	False
503	False	False	False	False	False	False	False	False	False	False	False	False	False	False
504	False	False	False	False	False	False	False	False	False	False	False	False	False	False
505	False	False	False	False	False	False	False	False	False	False	False	False	False	False

506 rows × 14 columns

```
df.isna().sum()
```

```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
```

```
ptratio    0
black      0
lstat      0
medv       0
dtype: int64
```

Checking for Outliers

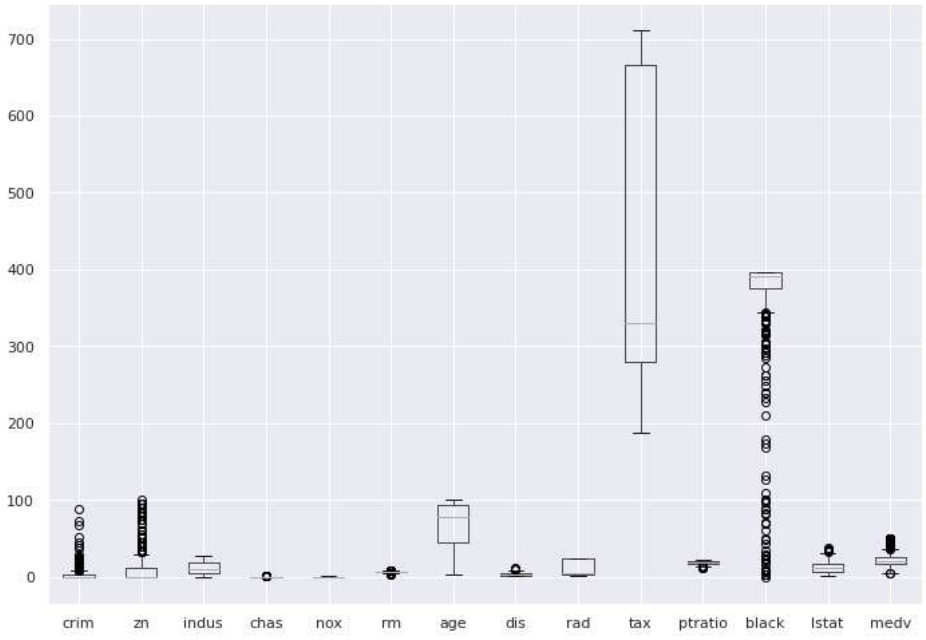
4. Feature Engineering

Outlier Detection and Removal:

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df.boxplot()
```

```
<AxesSubplot:>
```




```
Q1 = df['medv'].quantile(0.25)
Q3 = df['medv'].quantile(0.75)
IQR = Q3 - Q1
Lower_limit = Q1 - 1.5 * IQR
Upper_limit = Q3 + 1.5 * IQR
print(f'Q1 = {Q1}, Q3 = {Q3}, IQR = {IQR}, Lower_limit = {Lower_limit}, Upper_limit = {Upper_limit}')
```

```
Q1 = 17.025, Q3 = 25.0, IQR = 7.975000000000001, Lower_limit = 5.0624999999999964, Upper_limit = 36.962500000000006
```

```
outliers_medv=[]
for i in df.medv:
    if i<Lower_limit or i>Upper_limit:
       outliers_medv.append(i)
print("outliers are",outliers_medv)
```

```
outliers are [38.7, 43.8, 41.3, 50.0, 50.0, 50.0, 50.0, 37.2, 39.8, 37.9, 50.0, 37.0, 50.0, 42.3, 48.5, 50.0, 44.8, 50.0, 37.6, 46.7, 41.7, 48.3, 42.8
```

```
df[df.medv<Lower_limit].index
```

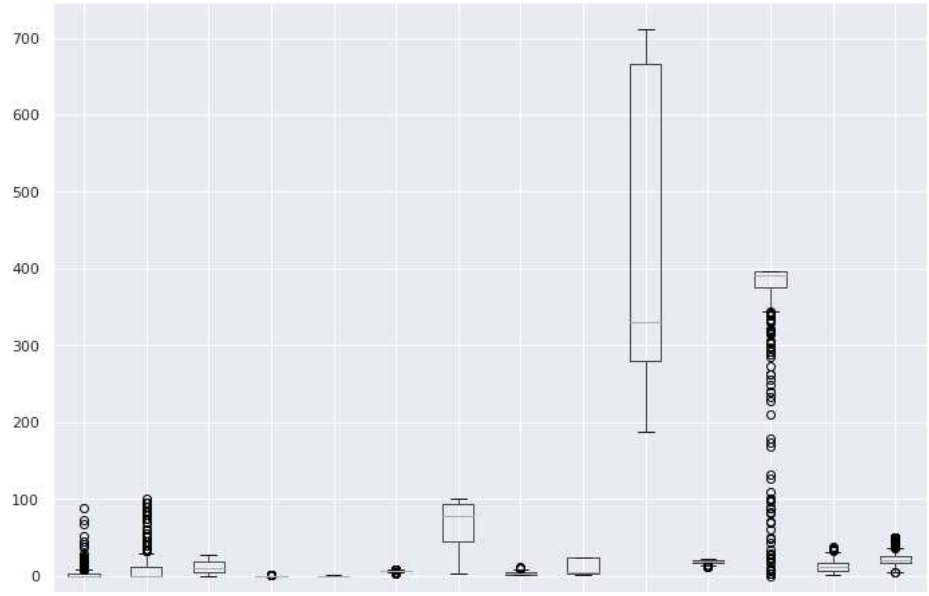
```
Int64Index([398, 405], dtype='int64')
```

```
df1=df.drop(df[df.medv<Lower_limit].index & df[df.medv>Upper_limit].index)
```

```
<ipython-input-66-b1eaccd07bbd>:1: FutureWarning: Index.__and__ operating as a set operation is deprecated, in the future this will be a logical operat
  df1=df.drop(df[df.medv<Lower_limit].index & df[df.medv>Upper_limit].index)
```

```
df1.boxplot()
```

```
<AxesSubplot:>
```




```
outliers_medv=[]
for i in df2.medv:
    if i<Lower_limit or i>Upper_limit:
       outliers_medv.append(i)
print("outliers are",outliers_medv)
```

```
outliers are []
```

```
df2
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

466 rows × 14 columns

Outlier Removal

```
X = df.drop(['medv'], axis = 1)
Y = df['medv']
```

```
X
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03

Y

```
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: medv, Length: 506, dtype: float64
```

506 rows × 13 columns

Double-click (or enter) to edit

5. Model Building and Training

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2,random_state = 0)
```

Training and testing the model

```
import sklearn
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
```

```
model=lm.fit(xtrain, ytrain)
```

```
model
```

```
▾ LinearRegression
LinearRegression()
```

Predict the y_pred for all values of train_x and test_x

```
ytrain_pred = lm.predict(xtrain)
ytest_pred = lm.predict(xtest)
ytrain_pred
```

```
array([307., 305., 300., 311., 666., 666., 273., 329., 403., 666., 311.,
       432., 311., 277., 666., 437., 666., 198., 398., 666., 233., 391.,
       304., 188., 222., 330., 284., 254., 666., 666., 711., 403., 666.,
       226., 307., 403., 330., 666., 666., 391., 193., 243., 398., 307.,
       402., 222., 666., 666., 304., 222., 437., 223., 437., 358., 188.,
       307., 345., 307., 666., 307., 216., 307., 216., 287., 287., 281.,
       666., 666., 265., 666., 666., 403., 666., 666., 391., 391., 666.,
       243., 666., 666., 224., 403., 384., 287., 284., 280., 193., 666.,
       666., 264., 270., 304., 264., 270., 329., 666., 304., 216., 270.,
       307., 264., 329., 403., 403., 403., 330., 666., 403., 247., 307.,
       666., 337., 304., 666., 437., 384., 666., 242., 666., 307., 307.,
       403., 315., 666., 307., 432., 300., 384., 666., 304., 296., 330.,
       432., 403., 666., 188., 264., 432., 666., 351., 666., 437., 398.,
       411., 193., 188., 285., 307., 666., 666., 666., 307., 437., 307.,
       666., 233., 284., 402., 223., 293., 276., 305., 279., 666., 307.,
       666., 352., 666., 276., 307., 666., 289., 403., 224., 666., 437.,
       277., 384., 334., 335., 334., 277., 666., 398., 277., 224., 307.,
       264., 345., 307., 276., 307., 437., 432., 403., 666., 666., 252.,
       254., 666., 398., 304., 304., 307., 293., 666., 391., 307., 403.,
       281., 273., 666., 384., 403., 403., 307., 370., 384., 666., 437.,
       307., 264., 666., 304., 273., 666., 330., 304., 666., 307., 430.,
       289., 223., 307., 666., 222., 187., 188., 216., 241., 398., 193.,
       666., 666., 254., 289., 666., 193., 666., 281., 222., 264., 311.,
       666., 247., 264., 279., 666., 284., 233., 223., 243., 277., 666.,
       666., 666., 188., 307., 284., 666., 437., 398., 307., 233., 666.,
       296., 348., 666., 666., 276., 335., 222., 391., 296., 666., 244.,
       270., 666., 276., 666., 403., 345., 222., 233., 284., 666., 666.,
       277., 216., 307., 403., 403., 224., 193., 666., 437., 666., 273.,
       264., 287., 224., 247., 188., 711., 666., 666., 666., 666., 270.,
```

```
            281., 264., 432., 666., 245., 256., 666., 287., 264., 666., 666.,
            233., 384., 666., 666., 223., 666., 279., 666., 243., 666., 666.,
            437., 245., 307., 193., 666., 403., 403., 330., 348., 307., 307.,
            437., 193., 666., 666., 293., 666., 666., 666., 403., 300., 711.,
            296., 276., 224., 666., 666., 224., 329., 300., 666., 432., 666.,
            264., 305., 224., 307., 403., 224., 711., 296., 711., 252., 265.,
            304., 666., 270., 666., 305., 247., 245., 300., 254., 277., 311.,
            666., 255., 330., 287., 398., 432., 233., 296.])
```

```
ytrain
```

```
     220    307.0
     71     305.0
     240    300.0
     6      311.0
     417    666.0
            ...
     323    287.0
     192    398.0
     117    432.0
     47     233.0
     172    296.0
     Name: tax, Length: 404, dtype: float64
```

6. Model Evaluation

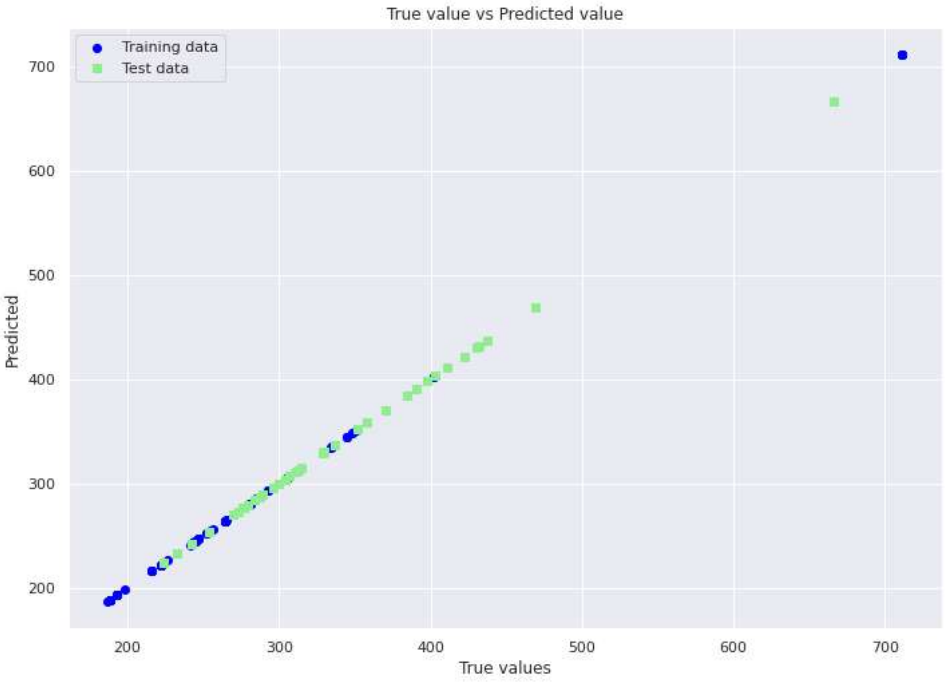
```
mse = mean_squared_error(ytrain, ytrain_pred)

print("The model performance for training set")
print("--------------------------------------")
print('MSE is {}'.format(mse))
print("\n")

# model evaluation for testing set
#y_test_predict = lin_model.predict(X_test)
mse = mean_squared_error(ytest, ytest_pred)


print("The model performance for testing set")
print("--------------------------------------")
print('MSE is {}'.format(mse))
print("\n\n\n")

rmse = (np.sqrt(mean_squared_error(ytrain, ytrain_pred)))
r2 = r2_score(ytrain, ytrain_pred)

print("The model performance for training set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("\n")

# model evaluation for testing set
#y_test_predict = lin_model.predict(X_test)
rmse = (np.sqrt(mean_squared_error(ytest, ytest_pred)))
r2 = r2_score(ytest, ytest_pred)

print("The model performance for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

```
     The model performance for training set
     --------------------------------------
     MSE is 6.432156251261397e-26


     The model performance for testing set
     --------------------------------------
     MSE is 5.362323766464221e-26




     The model performance for training set
     --------------------------------------
     RMSE is 2.5361696022272244e-13
     R2 score is 1.0


     The model performance for testing set
     --------------------------------------
```

```
RMSE is 2.315669183295451e-13
R2 score is 1.0
```

Plotting the linear regression model

```
plt.scatter(ytrain ,ytrain_pred,c='blue',marker='o',label='Training data')
plt.scatter(ytest,ytest_pred ,c='lightgreen',marker='s',label='Test data')
plt.xlabel('True values')
plt.ylabel('Predicted')
plt.title("True value vs Predicted value")
plt.legend(loc= 'upper left')
#plt.hlines(y=0,xmin=0,xmax=50)
plt.plot()
plt.show()
```



DATA ANALYTICS-II

1. Problem Statement

1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Import libraries and create alias for Pandas, Numpy

```
import pandas as pd
import numpy as np
```

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

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Social_Network_Ads.csv to Social_Network_Ads.csv
{'Social_Network_Ads.csv':
b'Age,EstimatedSalary,Purchased\n19,19000,0\n35,20000,0\n26,43000,0\n27,57000,0\n19,760
```

2. Data Collection

Loading the data

```
df=pd.read_csv("/content/Social_Network_Ads.csv")
```

3. Exploratory Data Analysis

Data Preprocessing:

```
df.head()
```

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

```
df.describe()
```

	Age	EstimatedSalary	Purchased
count	400.000000	400.000000	400.000000
mean	37.655000	69742.500000	0.357500
std	10.482877	34096.960282	0.479864
min	18.000000	15000.000000	0.000000
25%	29.750000	43000.000000	0.000000
50%	37.000000	70000.000000	0.000000
75%	46.000000	88000.000000	1.000000
max	60.000000	150000.000000	1.000000

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

```
Age               0
EstimatedSalary   0
Purchased         0
dtype: int64
```

```
df.info()
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Age              400 non-null    int64
 1   EstimatedSalary  400 non-null    int64
 2   Purchased        400 non-null    int64
dtypes: int64(3)
memory usage: 9.5 KB
```

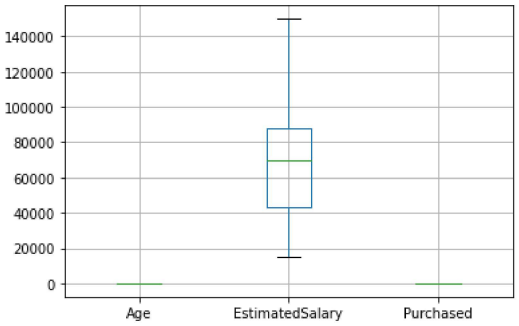
4. Feature Engineering

Outlier Detection and Removal:

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df.boxplot()
```

```
<Axes: >
```




```
X = df.drop(['Purchased'], axis = 1)
Y = df['Purchased']
```

5. Model Building and Training

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(X, Y, test_size =0.2,random_state = 0)
```

```
from sklearn.linear_model import LogisticRegression
```

```
logreg = LogisticRegression()
```

```
logreg.fit(xtrain,ytrain)
```

```
LogisticRegression
LogisticRegression()
```

Predict the y_pred for all values of and test_x

```
y_pred=logreg.predict(xtest)
```

```
print(xtrain)
print("------------\n")
print(xtest)
print("------------\n")
print(ytrain)
print("------------\n")
print(ytest)
print("------------\n")
print(y_pred)
```

```
300   58            38000
```

```
172   26           118000

[320 rows x 2 columns]
------------

     Age  EstimatedSalary
132   30            87000
309   38            50000
341   35            75000
196   30            79000
246   35            50000
..   ...              ...
14    18            82000
363   42            79000
304   40            60000
361   53            34000
329   47           107000

[80 rows x 2 columns]
------------

336    1
64     0
55     0
106    0
300    1
      ..
323    1
192    0
117    0
47     0
172    0
Name: Purchased, Length: 320, dtype: int64
------------

132    0
309    0
341    0
196    0
246    0
      ..
14     0
363    0
304    0
361    1
329    1
Name: Purchased, Length: 80, dtype: int64
------------

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0]
```

6. Model Evaluation

Find the Following Parameters for Logistic Regression on Social_Networking_Ads dataset: 1. Classification Report 2. Accuracy Score 3. Confusion Matrix 4. Error Rate 5. Precision 6. Recall

```
from sklearn.metrics import precision_score,confusion_matrix,accuracy_score,recall_score, classification_report
```

Confusion Matrix

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

```
array([[58,  0],
       [22,  0]])
```

accuracy_score

```
print ("Accuracy : ", accuracy_score(ytest, y_pred))
```

```
Accuracy :  0.725
```

Precision

```
ps = precision_score(ytest, y_pred)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 d
  _warn_prf(average, modifier, msg_start, len(result))
```

```
ps
```

```
0.0
```

Recall score

```
rs = recall_score(ytest, y_pred)
```

```
rs
```

```
0.0
```

*Error Rate**

```
error_rate = 1- accuracy_score(ytest, y_pred)
```

```
error_rate
```

```
0.275
```

Classification Report

```
print("classification report: ",classification_report(ytest, y_pred))
```

```
classification report:                precision    recall  f1-score   support

           0       0.72      1.00      0.84        58
           1       0.00      0.00      0.00        22

    accuracy                           0.73        80
   macro avg       0.36      0.50      0.42        80
weighted avg       0.53      0.72      0.61        80

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being
  _warn_prf(average, modifier, msg_start, len(result))
```

DATA ANALYTICS-II

1. Problem Statement

1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Import libraries and create alias for Pandas, Numpy

```
import pandas as pd
import numpy as np
```

Import the Iris Dataset

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

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

2. Data Collection

Loading the data

```
from sklearn.datasets import load_iris
df = load_iris()
```

3. Exploratory Data Analysis

Data Preprocessing:

```
df.head()
```

```
df.describe()
```

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

Check for Null Values

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

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

```
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
```

5. Model Building and Training

```
X = df.drop(['Species'], axis = 1)
Y = df['Species']
```

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(X, Y, test_size =0.2,random_state = 0)
```

```
from sklearn.naive_bayes import GaussianNB
gaussian = GaussianNB()
```

```
gaussian.fit(xtrain, ytrain)
```

```
GaussianNB()
```

Predict the y_pred for all values of train_x and test_x

```
y_pred = gaussian.predict(xtest)
```

```
print(xtrain)
print("------------\n")
print(xtest)
print("------------\n")
print(ytrain)
print("------------\n")
print(ytest)
print("------------\n")
print(y_pred)
```

```
24         Iris-setosa
8          Iris-setosa
126     Iris-virginica
22         Iris-setosa
44         Iris-setosa
97     Iris-versicolor
93     Iris-versicolor
26         Iris-setosa
Name: Species, dtype: object
------------

['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa']
```

6. Model Evaluation

```
from sklearn.metrics import precision_score,confusion_matrix,accuracy_score,recall_score, classification_report
cm= confusion_matrix(ytest, y_pred)
```

Confusion Matrix

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

```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  0,  6]])
```

Accuracy Score

```
print ("Accuracy : ", accuracy_score(ytest, y_pred))
```

```
Accuracy :  1.0
```

Error Rate

```
error_rate = 1- accuracy_score(ytest, y_pred)
```

```
error_rate
```

```
0.0
```

Classification

```
print("classification report: ",classification_report(ytest, y_pred))
```

```
classification report:                   precision    recall  f1-score   support

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

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

✓ 0s completed at 11:23 AM

TEXT ANALYSIS

1. Problem Statement

1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.

Import and install required packages

```
pip install nltk
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: nltk in /usr/local/lib/python3.9/dist-packages (3.8.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.9/dist-packages (from nltk) (2022.10.31)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from nltk) (4.65.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.9/dist-packages (from nltk) (1.2.0)
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packages (from nltk) (8.1.3)
```

```
import nltk
import re
```

Download the required packages

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

2. Data Collection

```
text= "Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentences is cal
```

3. Exploratory Data Analysis

Perform Tokenization

Sentence Tokenization

```
from nltk.tokenize import sent_tokenize
tokenized_text= sent_tokenize(text)
print(tokenized_text)
```

```
['Tokenization is the first step in text analytics.', 'The process of breaking down a text paragraph into smaller chunks such as words or sentences is
```

Word Tokenization

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

```
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analytics', '.', 'The', 'process', 'of', 'breaking', 'down', 'a', 'text', 'paragraph', 'i
```

Removing Punctuations and Stop Word

```
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
text= "How to remove stop words with NLTK library in Python?"
text= re.sub('[^a-zA-Z]', ' ',text)
tokens = word_tokenize(text.lower())
filtered_text=[]
for w in tokens:
  if w not in stop_words:
    filtered_text.append(w)
print("Tokenized Sentence:",tokens)
print("Filterd Sentence:",filtered_text)
```

```
{'hasn', 'won', 'can', 'below', 'didn', 'where', "she's", 'his', 'yourself', 'an', 'during', "wouldn't", 'how', 'between', 'she', 'him', 'over', 'own'
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'library', 'in', 'python']
Filterd Sentence: ['remove', 'stop', 'words', 'nltk', 'library', 'python']
```

Perform Stemming

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

```
wait
```

Perform Lemmatization

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

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

Apply POS Tagging to text

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

```
[('The', 'DT')]
[('pink', 'NN')]
[('sweater', 'NN')]
[('fit', 'NN')]
[('her', 'PRP$')]
[('perfectly', 'RB')]
```

Algorithm for Create representation of document by calculating TFIDF

Step 1: Import the necessary libraries.

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

Step 2: Initialize the Documents.

```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
```

Step 3: Create BagofWords (BoW) for Document A and B.

```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
```

Step 4: Create Collection of Unique words from Document A and B.

```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

Step 5: Create a dictionary of words and their occurrence for each document in the corpus

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

Step 6: Compute the term frequency for each of our documents.

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

Step 7: Compute the term Inverse Document Frequency.

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

Step 8: Compute the term TF/IDF for all words.a

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

Data Visualization I

1. Problem Statement

1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data.
2. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.

Import and install required packages

```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Load Titanic dataset using seaborn

2. Data Collection

Loading the data

```
dataset = sns.load_dataset('titanic')
dataset.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embar
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	South
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Ch
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	South
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	South
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	South

3.Exploratory Data Analysis

Draw distributional plot

```
sns.distplot(dataset['fare'])
```

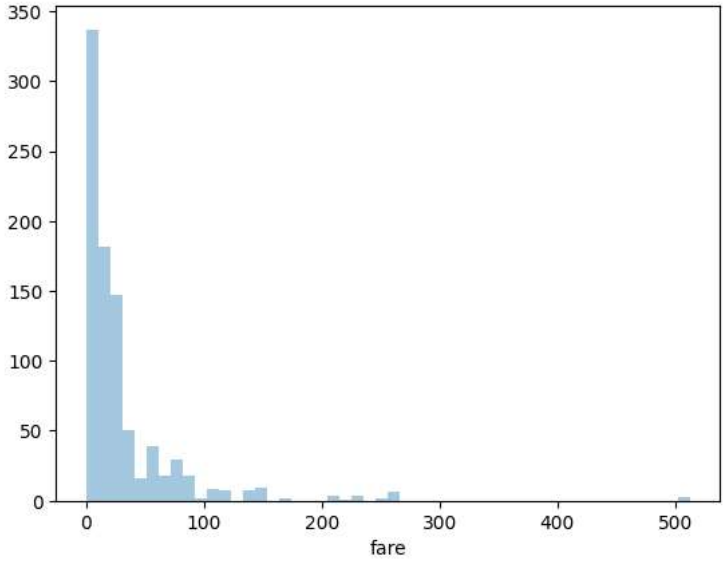
```
<ipython-input-5-fa36e370c0c4>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
```

Removal of Kernal Density line

```
  sns.distplot(dataset['fare'])
```

```
sns.distplot(dataset['fare'], kde=False)
```

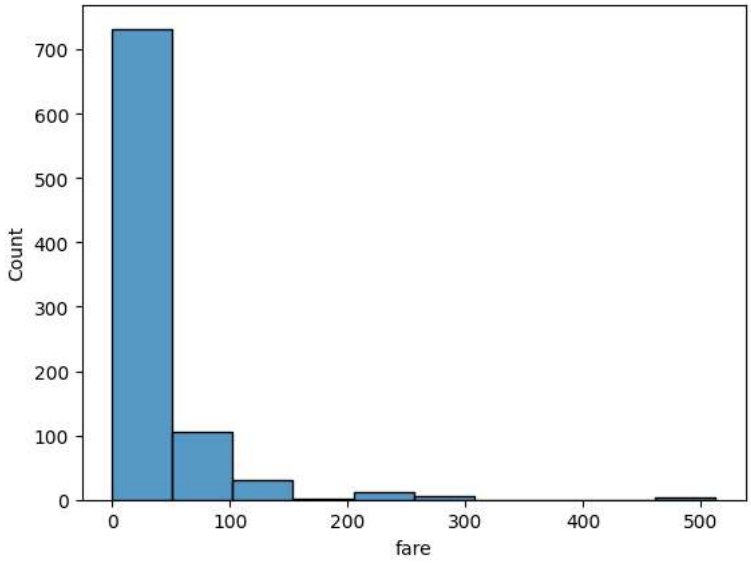
```
<ipython-input-6-716256732cc0>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataset['fare'], kde=False)
<Axes: xlabel='fare'>
```



Draw histogram

```
sns.histplot(dataset['fare'], kde=False, bins=10)
```

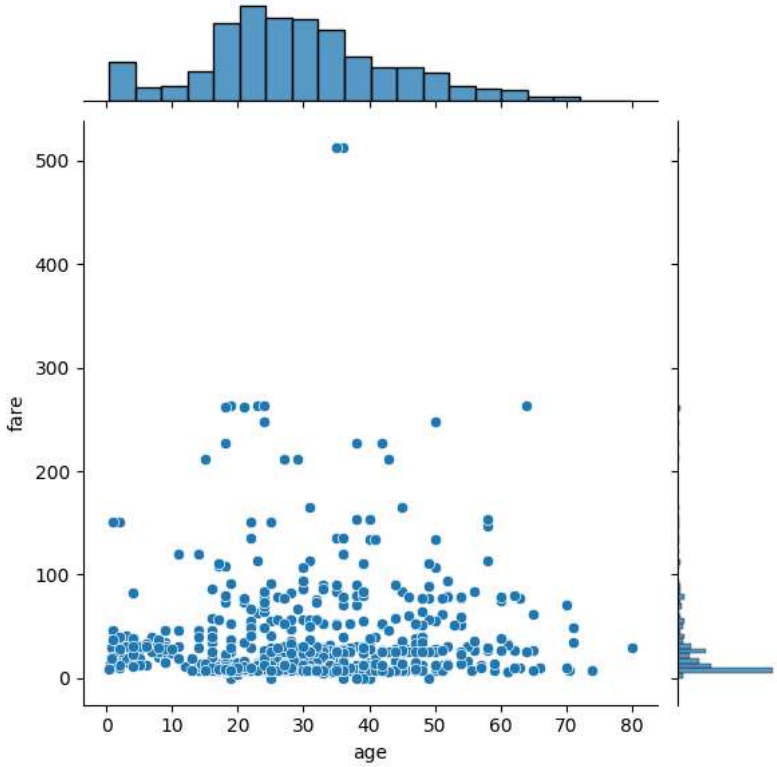
```
<Axes: xlabel='fare', ylabel='Count'>
```



The Joint Plot

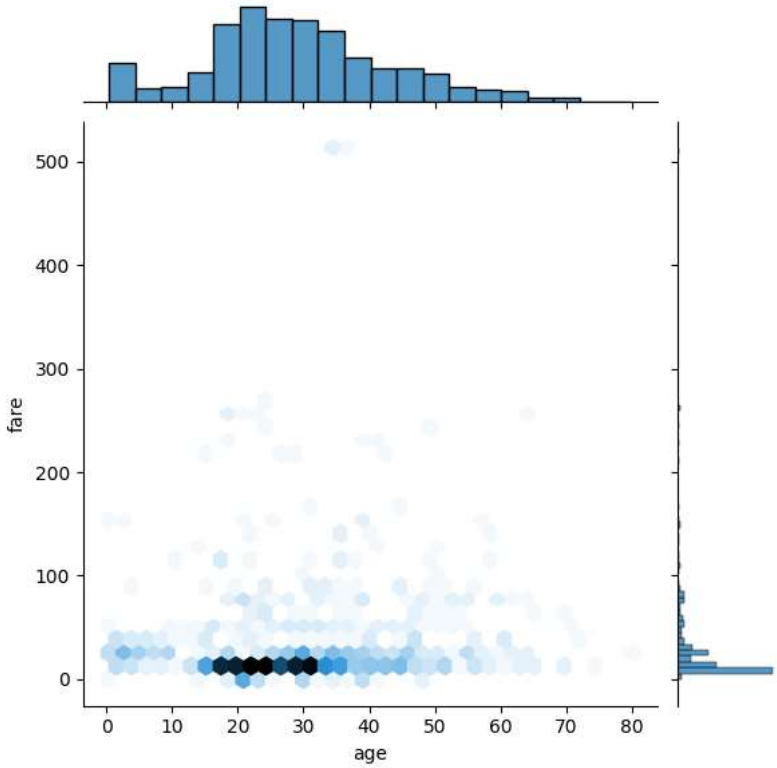
```
 sns.jointplot(x='age', y='fare', data=dataset)
```

```
<seaborn.axisgrid.JointGrid at 0x7fcf11cb4850>
```




```
sns.jointplot(x='age', y='fare', data=dataset, kind='hex')
```

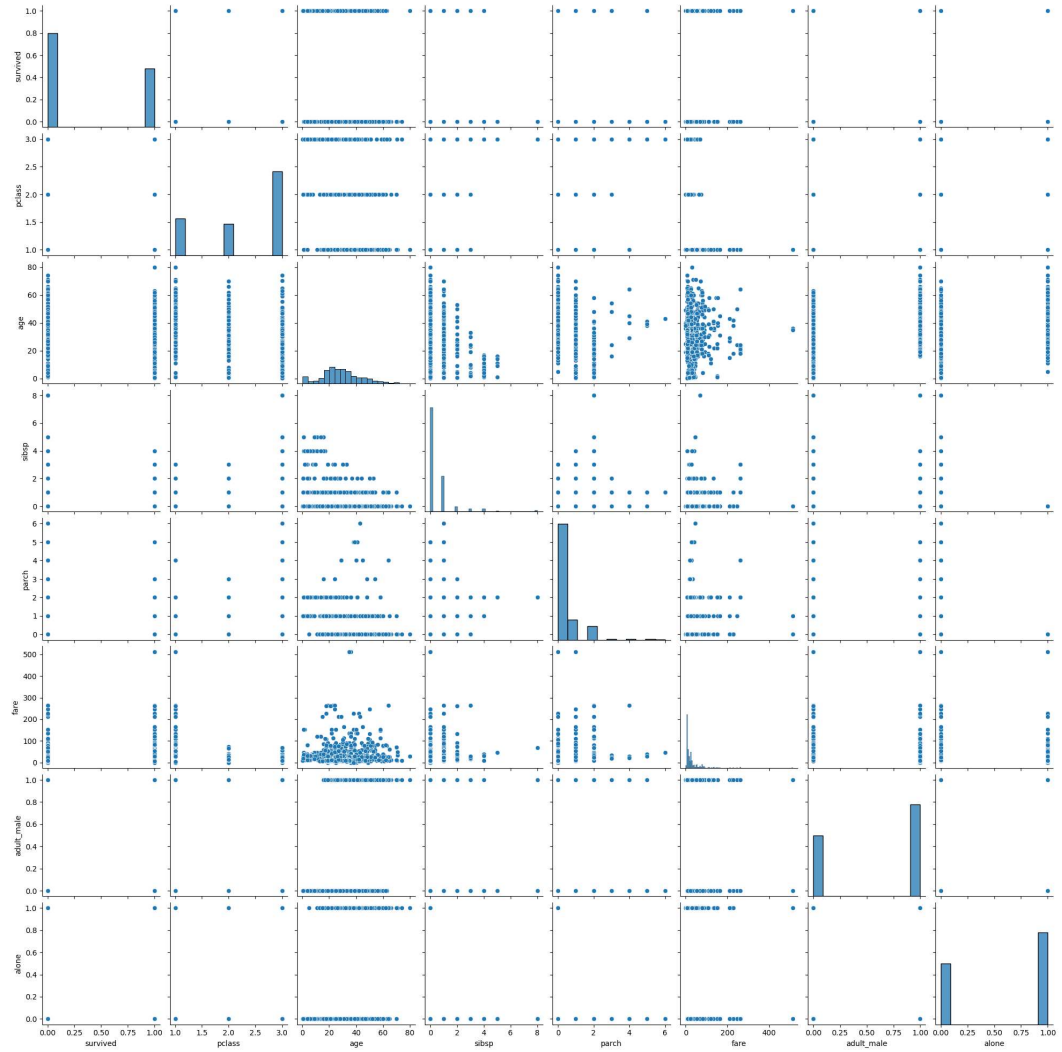
```
<seaborn.axisgrid.JointGrid at 0x7fcee1d1a820>
```



Pair Plot

```
sns.pairplot(dataset)
```

```
<__array_function__ internals>:180: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'>
<__array_function__ internals>:180: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'>
<seaborn.axisgrid.PairGrid at 0x7fcee17ca520>
```



Data Visualization II

1. Problem Statement

1. Use the inbuilt dataset 'titanic' as used in the above problem. Plot a box plot for distribution of age with respect to each gender along with the information about whether they survived or not. (Column names : 'sex' and 'age')
2. Write observations on the inference from the above statistics.

Import all Packages

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from seaborn import load_dataset
```

2. Data Collection

Loading the data

```
data = load_dataset("titanic")
```

Basic Operations

```
data.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embar
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	South
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Ch
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	South
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	South
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	South

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

```
data.describe()
```

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429

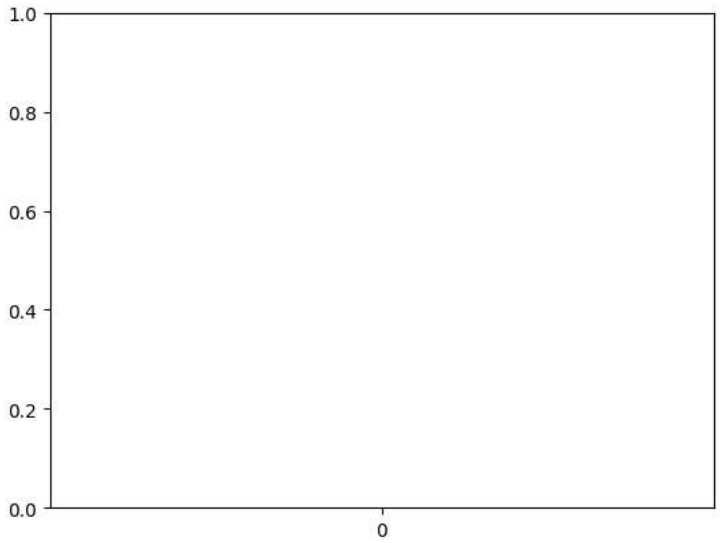
```
data.isna().sum()
```

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

3.Exploratory Data Analysis

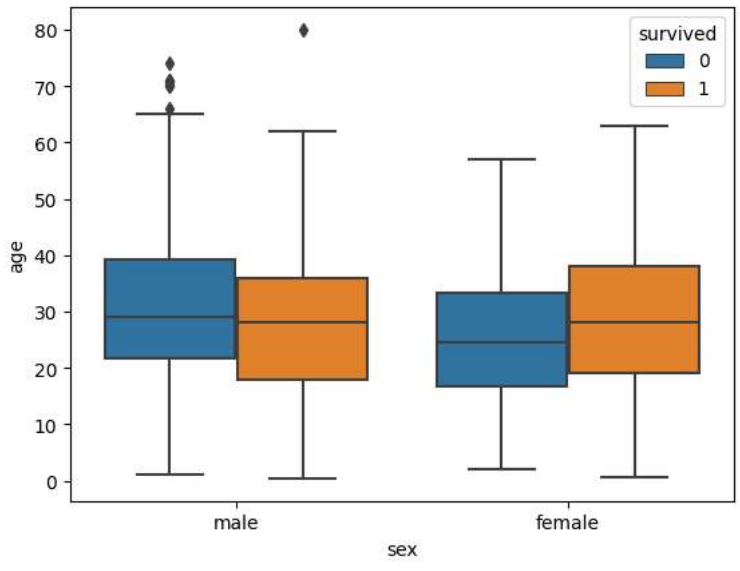
```
sns.boxplot(data)
```

```
<Axes: >
```




```
sns.boxplot(x = data['sex'], y = data["age"], hue = data["survived"])
```

```
<Axes: xlabel='sex', ylabel='age'>
```



Data Visualization III

1. Problem Statement

Download the Iris flower dataset or any other dataset into a DataFrame. (e.g., https://archive.ics.uci.edu/ml/datasets/Iris). Scan the dataset and give the inference as:

1. List down the features and their types (e.g., numeric, nominal) available in the dataset.
2. Create a histogram for each feature in the dataset to illustrate the feature distributions.
3. Create a boxplot for each feature in the dataset.
4. Compare distributions and identify outliers.

Importing Libraries

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

Upload dataset

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

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Iris.csv to Iris.csv
{'Iris.csv':
b'Id,SepalLengthCm,SepalWidthCm,PetalLengthCm,PetalWidthCm,Species\n1,5.1,3.5,1.4,0.2,Iris-
setosa\n2,4.9,3.0,1.4,0.2,Iris-setosa\n3,4.7,3.2,1.3,0.2,Iris-setosa\n4,4.6,3.1,1.5,0.2,Iris-
setosa\n5,5.0,3.6,1.4,0.2,Iris-setosa\n6,5.4,3.9,1.7,0.4,Iris-setosa\n7,4.6,3.4,1.4,0.3,Iris-
setosa\n8,5.0,3.4,1.5,0.2,Iris-setosa\n9,4.4,2.9,1.4,0.2,Iris-setosa\n10,4.9,3.1,1.5,0.1,Iris-
setosa\n11,5.4,3.7,1.5,0.2,Iris-setosa\n12,4.8,3.4,1.6,0.2,Iris-setosa\n13,4.8,3.0,1.4,0.1,Iris-
setosa\n14,4.3,3.0,1.1,0.1,Iris-setosa\n15,5.8,4.0,1.2,0.2,Iris-setosa\n16,5.7,4.4,1.5,0.4,Iris-
setosa\n17,5.4,3.9,1.3,0.4,Iris-setosa\n18,5.1,3.5,1.4,0.3,Iris-setosa\n19,5.7,3.8,1.7,0.3,Iris-
setosa\n20,5.1,3.8,1.5,0.3,Iris-setosa\n21,5.4,3.4,1.7,0.2,Iris-setosa\n22,5.1,3.7,1.5,0.4,Iris-
setosa\n23,4.6,3.6,1.0,0.2,Iris-setosa\n24,5.1,3.3,1.7,0.5,Iris-setosa\n25,4.8,3.4,1.9,0.2,Iris-
setosa\n26,5.0,3.0,1.6,0.2,Iris-setosa\n27,5.0,3.4,1.6,0.4,Iris-setosa\n28,5.2,3.5,1.5,0.2,Iris-
setosa\n29,5.2,3.4,1.4,0.2,Iris-setosa\n30,4.7,3.2,1.6,0.2,Iris-setosa\n31,4.8,3.1,1.6,0.2,Iris-
setosa\n32,5.4,3.4,1.5,0.4,Iris-setosa\n33,5.2,4.1,1.5,0.1,Iris-setosa\n34,5.5,4.2,1.4,0.2,Iris-
setosa\n35,4.9,3.1,1.5,0.1,Iris-setosa\n36,5.0,3.2,1.2,0.2,Iris-setosa\n37,5.5,3.5,1.3,0.2,Iris-
setosa\n38,4.9,3.1,1.5,0.1,Iris-setosa\n39,4.4,3.0,1.3,0.2,Iris-setosa\n40,5.1,3.4,1.5,0.2,Iris-
setosa\n41,5.0,3.5,1.3,0.3,Iris-setosa\n42,4.5,2.3,1.3,0.3,Iris-setosa\n43,4.4,3.2,1.3,0.2,Iris-
setosa\n44,5.0,3.5,1.6,0.6,Iris-setosa\n45,5.1,3.8,1.9,0.4,Iris-setosa\n46,4.8,3.0,1.4,0.3,Iris-
setosa\n47,5.1,3.8,1.6,0.2,Iris-setosa\n48,4.6,3.2,1.4,0.2,Iris-setosa\n49,5.3,3.7,1.5,0.2,Iris-
setosa\n50,5.0,3.3,1.4,0.2,Iris-setosa\n51,7.0,3.2,4.7,1.4,Iris-
versicolor\n52,6.4,3.2,4.5,1.5,Iris-versicolor\n53,6.9,3.1,4.9,1.5,Iris-
versicolor\n54,5.5,2.3,4.0,1.3,Iris-versicolor\n55,6.5,2.8,4.6,1.5,Iris-
versicolor\n56,5.7,2.8,4.5,1.3,Iris-versicolor\n57,6.3,3.3,4.7,1.6,Iris-
versicolor\n58,4.9,2.4,3.3,1.0,Iris-versicolor\n59,6.6,2.9,4.6,1.3,Iris-
versicolor\n60,5.2,2.7,3.9,1.4,Iris-versicolor\n61,5.0,2.0,3.5,1.0,Iris-
versicolor\n62,5.9,3.0,4.2,1.5,Iris-versicolor\n63,6.0,2.2,4.0,1.0,Iris-
versicolor\n64,6.1,2.9,4.7,1.4,Iris-versicolor\n65,5.6,2.9,3.6,1.3,Iris-
versicolor\n66,6.7,3.1,4.4,1.4,Iris-versicolor\n67,5.6,3.0,4.5,1.5,Iris-
versicolor\n68,5.8,2.7,4.1,1.0,Iris-versicolor\n69,6.2,2.2,4.5,1.5,Iris-
versicolor\n70,5.6,2.5,3.9,1.1,Iris-versicolor\n71,5.9,3.2,4.8,1.8,Iris-
versicolor\n72,6.1,2.8,4.0,1.3,Iris-versicolor\n73,6.3,2.5,4.9,1.5,Iris-
versicolor\n74,6.1,2.8,4.7,1.2,Iris-versicolor\n75,6.4,2.9,4.3,1.3,Iris-
```

Laoding the dataset into dataframe

2. Data Collection

Loading the data

```
iris=pd.read_csv("/content/Iris.csv")
iris
```

	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
147	148	6.5	3.0	5.2	2.0	Iris-virginica

Basic Operations

```
iris.head()
```

	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

```
iris.describe()
```

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

```
iris.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
```

3.Exploratory Data Analysis

Check for null values

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

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

Statistical Measures

```
iris.min()
```

```
sepal_length            4.3
sepal_width             2.0
petal_length            1.0
petal_width             0.1
species         Iris-setosa
dtype: object
```

```
iris.max()
```

```
sepal_length               7.9
sepal_width                4.4
petal_length               6.9
petal_width                2.5
species         Iris-virginica
dtype: object
```

```
iris.mean()
```

```
<ipython-input-17-7eed97565d6e>:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it will defau
  iris.mean()
sepal_length    5.843333
sepal_width     3.054000
petal_length    3.758667
petal_width     1.198667
dtype: float64
```

```
iris.std()
```

```
-c5ab3f85284a>:1: FutureWarning: The default value of numeric_only in DataFrame.std is deprecated. In a future version, it will default to False. In ad

.828066
.433594
.764420
.763161
```

```
percentile = np.percentile(iris['sepal_length'], 90)
percentile
```

```
6.9
```

```
percentile1 = np.percentile(iris['sepal_length'], 75)
percentile1
```

```
6.4
```

```
percentile2 = np.percentile(iris['sepal_length'], 50)
percentile2
```

```
5.8
```

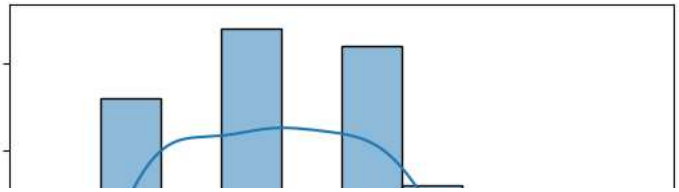
```
percentile3 = np.percentile(iris['sepal_length'], 25)
percentile3
```

```
5.1
```

Histogram Plots

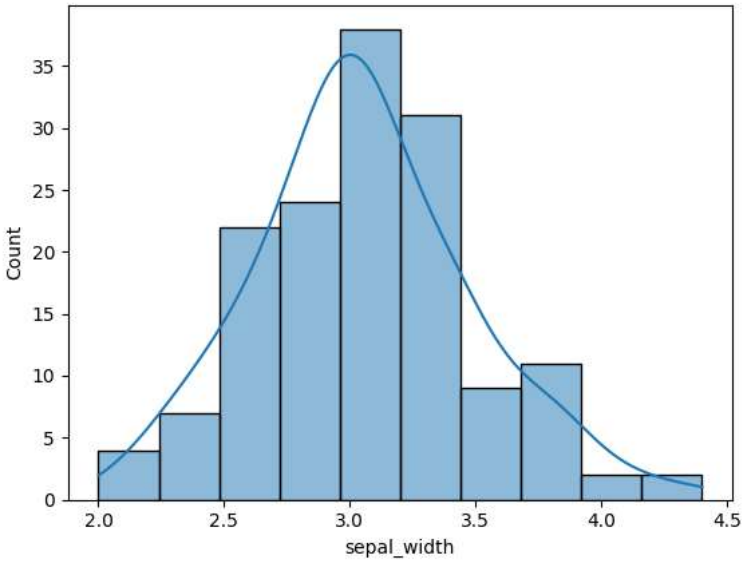
```
sns.histplot(iris['sepal_length'], kde=True, bins=10)
```

```
<Axes: xlabel='sepal_length', ylabel='Count'>
```



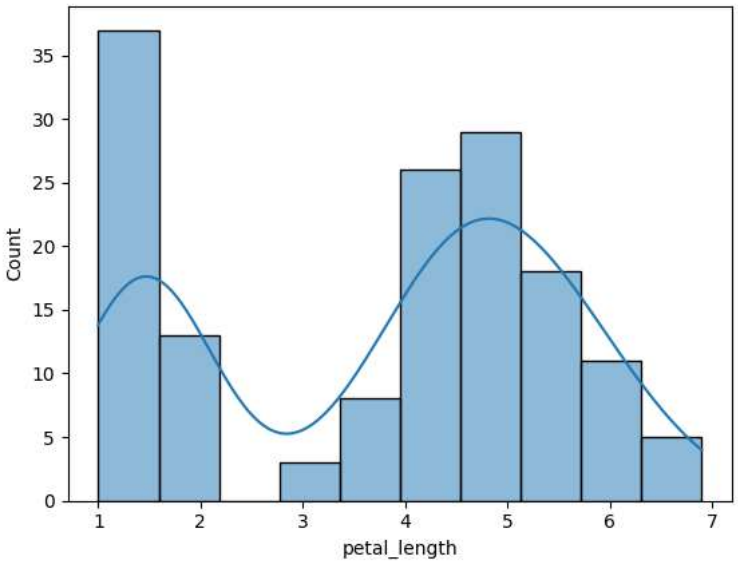

```
sns.histplot(iris['sepal_width'], kde=True, bins=10)
```

```
<Axes: xlabel='sepal_width', ylabel='Count'>
```



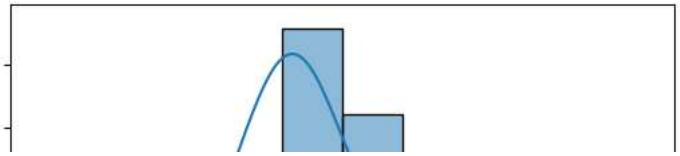

```
sns.histplot(iris['petal_length'], kde=True, bins=10)
```

```
<Axes: xlabel='petal_length', ylabel='Count'>
```




```
sns.histplot(iris['sepal_width'], kde=True, bins=10)
```

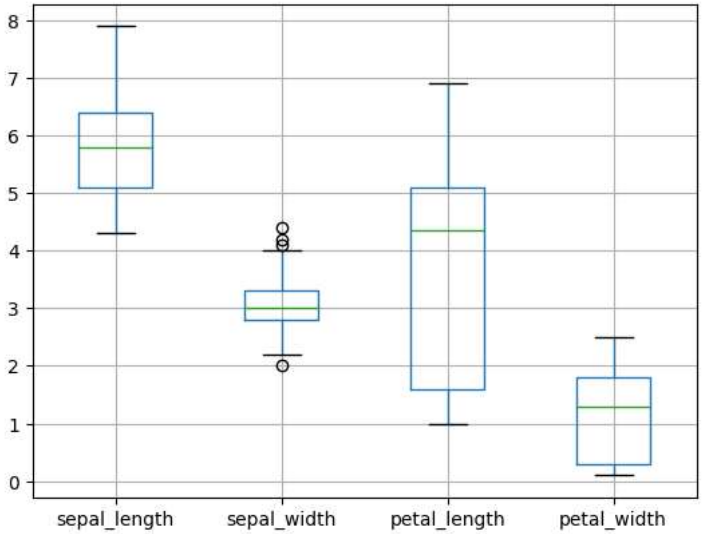
```
<Axes: xlabel='sepal_width', ylabel='Count'>
```



Box Plot

```
iris.boxplot()
```

```
<Axes: >
```




```
sns.boxplot(x=iris.sepal_width)
```

```
<Axes: xlabel='sepal_width'>
```