310256: Data Science and Big Data Analytics Laboratory

Title of the Assignment: Data Wrangling, II

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.

Reason and document your approach properly.

1. Import all the required Python Libraries.

In [8]:
```
import pandas as pd
import numpy as np
```

2. Creation of Dataset using Microsoft Excel.

3.Load the Dataset into pandas dataframe.

In [9]:
```
df=pd.read_csv("E:\DSBDL\Practical Programs\DSBDL Practical\DSBDALab_A2\AcademicPerformance.csv")
```

In [10]:
```
df
```

Out[10]:

	Rollno	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result
0	1	NaN	A	36.0	40.0	41.0	48.0	65.0	74.0	268	53.6	Pass
1	2	NaN	A	88.0	75.0	NaN	NaN	42.0	NaN	117	23.4	Pass
2	3	NaN	A	20.0	54.0	45.0	46.0	60.0	61.0	266	53.2	Fail
3	4	NaN	A	58.0	64.0	66.0	67.0	53.0	47.0	297	59.4	Pass
4	5	NaN	A	72.0	53.0	72.0	58.0	80.0	65.0	328	65.6	Pass
5	6	NaN	A	25.0	49.0	48.0	73.0	45.0	75.0	290	58.0	Pass
6	7	NaN	A	90.0	NaN	50.0	67.0	50.0	61.0	228	45.6	Pass
7	8	NaN	A	36.0	48.0	54.0	74.0	46.0	64.0	286	57.2	Pass
8	9	NaN	A	59.0	78.0	70.0	72.0	44.0	42.0	306	61.2	Pass
9	10	NaN	A	82.0	57.0	75.0	70.0	59.0	63.0	324	64.8	Pass
10	11	NaN	A	34.0	490.0	59.0	NaN	67.0	78.0	253	50.6	Pass
11	12	NaN	B	21.0	72.0	52.0	56.0	49.0	45.0	274	54.8	Pass
12	13	NaN	B	73.0	60.0	71.0	53.0	69.0	46.0	299	59.8	Pass
13	14	NaN	B	49.0	NaN	NaN	60.0	47.0	70.0	177	35.4	Pass
14	15	NaN	B	59.0	51.0	40.0	74.0	48.0	47.0	260	52.0	Fail
15	16	NaN	B	61.0	56.0	74.0	40.0	54.0	78.0	302	60.4	Pass
16	17	NaN	B	59.0	72.0	56.0	68.0	63.0	69.0	328	65.6	Pass
17	18	NaN	B	26.0	56.0	55.0	74.0	75.0	55.0	315	63.0	Pass
18	19	NaN	B	76.0	80.0	51.0	56.0	66.0	53.0	306	61.2	Pass
19	20	NaN	B	56.0	49.0	63.0	57.0	73.0	64.0	306	61.2	Pass
20	21	NaN	A	38.0	56.0	76.0	41.0	NaN	43.0	216	43.2	Pass
21	22	NaN	A	79.0	65.0	58.0	47.0	60.0	50.0	280	56.0	Pass
22	23	NaN	A	87.0	64.0	45.0	44.0	44.0	53.0	250	50.0	Pass
23	24	NaN	A	60.0	56.0	70.0	45.0	47.0	66.0	284	56.8	Pass
24	25	NaN	A	NaN	60.0	75.0	61.0	65.0	53.0	314	62.8	Pass
25	26	NaN	A	22.0	71.0	74.0	65.0	64.0	64.0	338	67.6	Pass
26	27	NaN	A	58.0	68.0	50.0	79.0	65.0	62.0	324	64.8	Pass
27	28	NaN	A	89.0	54.0	77.0	49.0	75.0	63.0	318	63.6	Pass
28	29	NaN	A	55.0	67.0	42.0	NaN	49.0	NaN	158	31.6	Pass
29	30	NaN	A	69.0	40.0	72.0	45.0	73.0	65.0	295	59.0	Fail

4. Data Preprocessing:

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

Out[7]:

	Rollno	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result
0	1	NaN	A	36.0	40.0	41.0	48.0	65.0	74.0	268	53.6	Pass
1	2	NaN	A	88.0	75.0	NaN	NaN	42.0	NaN	117	23.4	Pass
2	3	NaN	A	20.0	54.0	45.0	46.0	60.0	61.0	266	53.2	Fail
3	4	NaN	A	58.0	64.0	66.0	67.0	53.0	47.0	297	59.4	Pass
4	5	NaN	A	72.0	53.0	72.0	58.0	80.0	65.0	328	65.6	Pass

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Rollno           30 non-null     int64  
 1   Name             0 non-null      float64
 2   Term             30 non-null     object 
 3   Attendence       29 non-null     float64
 4   Subject1_marks   28 non-null     float64
 5   Subject2_marks   28 non-null     float64
 6   Subject3_marks   27 non-null     float64
 7   Subject4_marks2  29 non-null     float64
 8   Subject5_marks   28 non-null     float64
 9   Total_Marks      30 non-null     int64  
 10  Percentage       30 non-null     float64
 11  Result           30 non-null     object 
dtypes: float64(8), int64(2), object(2)
memory usage: 2.9+ KB
```

In [9]:
```
df.describe(include="all")
```

Out[9]:

	Rollno	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Re
count	30.000000	0.0	30	29.000000	28.000000	28.000000	27.000000	29.000000	28.000000	30.000000	30.000000	
unique	NaN	NaN	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	A	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	P
freq	NaN	NaN	21	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
mean	15.500000	NaN	NaN	56.448276	75.178571	60.035714	58.851852	58.517241	59.857143	276.900000	55.380000	N
std	8.803408	NaN	NaN	22.212587	81.975179	12.393194	11.937993	11.271395	10.630892	52.583825	10.516765	N
min	1.000000	NaN	NaN	20.000000	40.000000	40.000000	40.000000	42.000000	42.000000	117.000000	23.400000	N
25%	8.250000	NaN	NaN	36.000000	53.750000	50.000000	47.500000	48.000000	52.250000	261.500000	52.300000	N
50%	15.500000	NaN	NaN	59.000000	58.500000	58.500000	58.000000	60.000000	62.500000	292.500000	58.500000	N
75%	22.750000	NaN	NaN	73.000000	68.750000	72.000000	69.000000	66.000000	65.250000	312.000000	62.400000	N
max	30.000000	NaN	NaN	90.000000	490.000000	77.000000	79.000000	80.000000	78.000000	338.000000	67.600000	N

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

Out[10]:
```
(30, 12)
```

In [11]:
```
df.dtypes
```

Out[11]:
```
Rollno             int64
Name             float64
Term              object
Attendence       float64
Subject1_marks   float64
Subject2_marks   float64
Subject3_marks   float64
Subject4_marks2  float64
Subject5_marks   float64
Total_Marks        int64
Percentage       float64
Result            object
dtype: object
```

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

Out[12]:
```
Index(['Rollno', 'Name', 'Term', 'Attendence', 'Subject1_marks',
       'Subject2_marks', 'Subject3_marks', 'Subject4_marks2', 'Subject5_marks',
       'Total_Marks', 'Percentage', 'Result'],
      dtype='object')
```

In [13]:
```
df[0:5]
```

Out[13]:

	Rollno	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result
0	1	NaN	A	36.0	40.0	41.0	48.0	65.0	74.0	268	53.6	Pass
1	2	NaN	A	88.0	75.0	NaN	NaN	42.0	NaN	117	23.4	Pass
2	3	NaN	A	20.0	54.0	45.0	46.0	60.0	61.0	266	53.2	Fail
3	4	NaN	A	58.0	64.0	66.0	67.0	53.0	47.0	297	59.4	Pass
4	5	NaN	A	72.0	53.0	72.0	58.0	80.0	65.0	328	65.6	Pass

In [14]:
```
df.loc[0:2]
```

Out[14]:

	Rollno	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result
0	1	NaN	A	36.0	40.0	41.0	48.0	65.0	74.0	268	53.6	Pass
1	2	NaN	A	88.0	75.0	NaN	NaN	42.0	NaN	117	23.4	Pass
2	3	NaN	A	20.0	54.0	45.0	46.0	60.0	61.0	266	53.2	Fail

In [15]:
```
df.loc[0:2,'Subject1_marks':'Subject5_marks']
```

Out[15]:

	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks
0	40.0	41.0	48.0	65.0	74.0
1	75.0	NaN	NaN	42.0	NaN
2	54.0	45.0	46.0	60.0	61.0

In [16]:
```
df.iloc[1:3]
```

Out[16]:

	Rollno	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result
1	2	NaN	A	88.0	75.0	NaN	NaN	42.0	NaN	117	23.4	Pass
2	3	NaN	A	20.0	54.0	45.0	46.0	60.0	61.0	266	53.2	Fail

In [17]:
```
df.iloc[1:5,1:5]
```

Out[17]:

	Name	Term	Attendence	Subject1_marks
1	NaN	A	88.0	75.0
2	NaN	A	20.0	54.0
3	NaN	A	58.0	64.0
4	NaN	A	72.0	53.0

A. Identification and Handling of Null Values

check for missing values in the data using pandas isnull()

In [18]:
```
df.isnull()
```

Out[18]:

	RollNo	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result
0	False	True	False	False	False	False	False	False	False	False	False	False
1	False	True	False	False	False	True	True	False	True	False	False	False
2	False	True	False	False	False	False	False	False	False	False	False	False
3	False	True	False	False	False	False	False	False	False	False	False	False
4	False	True	False	False	False	False	False	False	False	False	False	False
5	False	True	False	False	False	False	False	False	False	False	False	False
6	False	True	False	False	True	False	False	False	False	False	False	False
7	False	True	False	False	False	False	False	False	False	False	False	False
8	False	True	False	False	False	False	False	False	False	False	False	False
9	False	True	False	False	False	False	False	False	False	False	False	False
10	False	True	False	False	False	False	True	False	False	False	False	False
11	False	True	False	False	False	False	False	False	False	False	False	False
12	False	True	False	False	False	False	False	False	False	False	False	False
13	False	True	False	False	True	True	False	False	False	False	False	False
14	False	True	False	False	False	False	False	False	False	False	False	False
15	False	True	False	False	False	False	False	False	False	False	False	False
16	False	True	False	False	False	False	False	False	False	False	False	False
17	False	True	False	False	False	False	False	False	False	False	False	False
18	False	True	False	False	False	False	False	False	False	False	False	False
19	False	True	False	False	False	False	False	False	False	False	False	False
20	False	True	False	False	False	False	False	True	False	False	False	False
21	False	True	False	False	False	False	False	False	False	False	False	False
22	False	True	False	False	False	False	False	False	False	False	False	False
23	False	True	False	False	False	False	False	False	False	False	False	False
24	False	True	False	True	False	False	False	False	False	False	False	False
25	False	True	False	False	False	False	False	False	False	False	False	False
26	False	True	False	False	False	False	False	False	False	False	False	False
27	False	True	False	False	False	False	False	False	False	False	False	False
28	False	True	False	False	False	False	True	False	True	False	False	False
29	False	True	False	False	False	False	False	False	False	False	False	False

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

Out[19]:

	Rollno	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result
0	False	True	False	False	False	False	False	False	False	False	False	False
1	False	True	False	False	False	True	True	False	True	False	False	False
2	False	True	False	False	False	False	False	False	False	False	False	False
3	False	True	False	False	False	False	False	False	False	False	False	False
4	False	True	False	False	False	False	False	False	False	False	False	False
5	False	True	False	False	False	False	False	False	False	False	False	False
6	False	True	False	False	True	False	False	False	False	False	False	False
7	False	True	False	False	False	False	False	False	False	False	False	False
8	False	True	False	False	False	False	False	False	False	False	False	False
9	False	True	False	False	False	False	False	False	False	False	False	False
10	False	True	False	False	False	False	True	False	False	False	False	False
11	False	True	False	False	False	False	False	False	False	False	False	False
12	False	True	False	False	False	False	False	False	False	False	False	False
13	False	True	False	False	True	True	False	False	False	False	False	False
14	False	True	False	False	False	False	False	False	False	False	False	False
15	False	True	False	False	False	False	False	False	False	False	False	False
16	False	True	False	False	False	False	False	False	False	False	False	False
17	False	True	False	False	False	False	False	False	False	False	False	False
18	False	True	False	False	False	False	False	False	False	False	False	False
19	False	True	False	False	False	False	False	False	False	False	False	False
20	False	True	False	False	False	False	False	True	False	False	False	False
21	False	True	False	False	False	False	False	False	False	False	False	False
22	False	True	False	False	False	False	False	False	False	False	False	False
23	False	True	False	False	False	False	False	False	False	False	False	False
24	False	True	False	True	False	False	False	False	False	False	False	False
25	False	True	False	False	False	False	False	False	False	False	False	False
26	False	True	False	False	False	False	False	False	False	False	False	False
27	False	True	False	False	False	False	False	False	False	False	False	False
28	False	True	False	False	False	False	True	False	True	False	False	False
29	False	True	False	False	False	False	False	False	False	False	False	False

In [21]:
```
df.isnull().any()
```

Out[21]:
```
Rollno             False
Name                True
Term               False
Attendence          True
Subject1_marks      True
Subject2_marks      True
Subject3_marks      True
Subject4_marks2     True
Subject5_marks      True
Total_Marks        False
Percentage         False
Result             False
dtype: bool
```

In [22]:
```
df.isnull().sum()
```

Out[22]:
```
Rollno              0
Name               30
Term                0
Attendence          1
Subject1_marks      2
Subject2_marks      2
Subject3_marks      3
Subject4_marks2     1
Subject5_marks      2
Total_Marks         0
Percentage          0
Result              0
dtype: int64
```

count of missing values of a specific column.

In [23]:
```
df.Attendence.isnull().sum()
```

Out[23]:
```
1
```

Make a list of column having missing value

In [24]:
```
cols_with_na = []
for col in df.columns:
    if df[col].isna().any():
        cols_with_na.append(col)

cols_with_na
```

Out[24]:
```
['Name',
 'Attendence',
 'Subject1_marks',
 'Subject2_marks',
 'Subject3_marks',
 'Subject4_marks2',
 'Subject5_marks']
```

In [ ]:

In [ ]:

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

1. replacing null values with NaN

In [25]:
```
df.replace(np.nan,value=0)
```

Out[25]:

	Rollno	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result
0	1	0.0	A	36.0	40.0	41.0	48.0	65.0	74.0	268	53.6	Pass
1	2	0.0	A	88.0	75.0	0.0	0.0	42.0	0.0	117	23.4	Pass
2	3	0.0	A	20.0	54.0	45.0	46.0	60.0	61.0	266	53.2	Fail
3	4	0.0	A	58.0	64.0	66.0	67.0	53.0	47.0	297	59.4	Pass
4	5	0.0	A	72.0	53.0	72.0	58.0	80.0	65.0	328	65.6	Pass
5	6	0.0	A	25.0	49.0	48.0	73.0	45.0	75.0	290	58.0	Pass
6	7	0.0	A	90.0	0.0	50.0	67.0	50.0	61.0	228	45.6	Pass
7	8	0.0	A	36.0	48.0	54.0	74.0	46.0	64.0	286	57.2	Pass
8	9	0.0	A	59.0	78.0	70.0	72.0	44.0	42.0	306	61.2	Pass
9	10	0.0	A	82.0	57.0	75.0	70.0	59.0	63.0	324	64.8	Pass
10	11	0.0	A	34.0	490.0	59.0	0.0	67.0	78.0	253	50.6	Pass
11	12	0.0	B	21.0	72.0	52.0	56.0	49.0	45.0	274	54.8	Pass
12	13	0.0	B	73.0	60.0	71.0	53.0	69.0	46.0	299	59.8	Pass
13	14	0.0	B	49.0	0.0	0.0	60.0	47.0	70.0	177	35.4	Pass
14	15	0.0	B	59.0	51.0	40.0	74.0	48.0	47.0	260	52.0	Fail
15	16	0.0	B	61.0	56.0	74.0	40.0	54.0	78.0	302	60.4	Pass
16	17	0.0	B	59.0	72.0	56.0	68.0	63.0	69.0	328	65.6	Pass
17	18	0.0	B	26.0	56.0	55.0	74.0	75.0	55.0	315	63.0	Pass
18	19	0.0	B	76.0	80.0	51.0	56.0	66.0	53.0	306	61.2	Pass
19	20	0.0	B	56.0	49.0	63.0	57.0	73.0	64.0	306	61.2	Pass
20	21	0.0	A	38.0	56.0	76.0	41.0	0.0	43.0	216	43.2	Pass
21	22	0.0	A	79.0	65.0	58.0	47.0	60.0	50.0	280	56.0	Pass
22	23	0.0	A	87.0	64.0	45.0	44.0	44.0	53.0	250	50.0	Pass
23	24	0.0	A	60.0	56.0	70.0	45.0	47.0	66.0	284	56.8	Pass
24	25	0.0	A	0.0	60.0	75.0	61.0	65.0	53.0	314	62.8	Pass
25	26	0.0	A	22.0	71.0	74.0	65.0	64.0	64.0	338	67.6	Pass
26	27	0.0	A	58.0	68.0	50.0	79.0	65.0	62.0	324	64.8	Pass
27	28	0.0	A	89.0	54.0	77.0	49.0	75.0	63.0	318	63.6	Pass
28	29	0.0	A	55.0	67.0	42.0	0.0	49.0	0.0	158	31.6	Pass
29	30	0.0	A	69.0	40.0	72.0	45.0	73.0	65.0	295	59.0	Fail

2. Filling null values with fillna()

In [26]:
```
df.fillna(1)
```

Out[26]:

	RollNo	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result
0	1	1.0	A	36.0	40.0	41.0	48.0	65.0	74.0	268	53.6	Pass
1	2	1.0	A	88.0	75.0	1.0	1.0	42.0	1.0	117	23.4	Pass
2	3	1.0	A	20.0	54.0	45.0	46.0	60.0	61.0	266	53.2	Fail
3	4	1.0	A	58.0	64.0	66.0	67.0	53.0	47.0	297	59.4	Pass
4	5	1.0	A	72.0	53.0	72.0	58.0	80.0	65.0	328	65.6	Pass
5	6	1.0	A	25.0	49.0	48.0	73.0	45.0	75.0	290	58.0	Pass
6	7	1.0	A	90.0	1.0	50.0	67.0	50.0	61.0	228	45.6	Pass
7	8	1.0	A	36.0	48.0	54.0	74.0	46.0	64.0	286	57.2	Pass
8	9	1.0	A	59.0	78.0	70.0	72.0	44.0	42.0	306	61.2	Pass
9	10	1.0	A	82.0	57.0	75.0	70.0	59.0	63.0	324	64.8	Pass
10	11	1.0	A	34.0	490.0	59.0	1.0	67.0	78.0	253	50.6	Pass
11	12	1.0	B	21.0	72.0	52.0	56.0	49.0	45.0	274	54.8	Pass
12	13	1.0	B	73.0	60.0	71.0	53.0	69.0	46.0	299	59.8	Pass
13	14	1.0	B	49.0	1.0	1.0	60.0	47.0	70.0	177	35.4	Pass
14	15	1.0	B	59.0	51.0	40.0	74.0	48.0	47.0	260	52.0	Fail
15	16	1.0	B	61.0	56.0	74.0	40.0	54.0	78.0	302	60.4	Pass
16	17	1.0	B	59.0	72.0	56.0	68.0	63.0	69.0	328	65.6	Pass
17	18	1.0	B	26.0	56.0	55.0	74.0	75.0	55.0	315	63.0	Pass
18	19	1.0	B	76.0	80.0	51.0	56.0	66.0	53.0	306	61.2	Pass
19	20	1.0	B	56.0	49.0	63.0	57.0	73.0	64.0	306	61.2	Pass
20	21	1.0	A	38.0	56.0	76.0	41.0	1.0	43.0	216	43.2	Pass
21	22	1.0	A	79.0	65.0	58.0	47.0	60.0	50.0	280	56.0	Pass
22	23	1.0	A	87.0	64.0	45.0	44.0	44.0	53.0	250	50.0	Pass
23	24	1.0	A	60.0	56.0	70.0	45.0	47.0	66.0	284	56.8	Pass
24	25	1.0	A	1.0	60.0	75.0	61.0	65.0	53.0	314	62.8	Pass
25	26	1.0	A	22.0	71.0	74.0	65.0	64.0	64.0	338	67.6	Pass
26	27	1.0	A	58.0	68.0	50.0	79.0	65.0	62.0	324	64.8	Pass
27	28	1.0	A	89.0	54.0	77.0	49.0	75.0	63.0	318	63.6	Pass
28	29	1.0	A	55.0	67.0	42.0	1.0	49.0	1.0	158	31.6	Pass
29	30	1.0	A	69.0	40.0	72.0	45.0	73.0	65.0	295	59.0	Fail

3. filling missing values using mean, median,max, min and standard deviation of that column

In [27]:
```
df['Subject1_marks']=df['Subject1_marks'].fillna(df['Subject1_marks'].mean())
```

In [28]:
```
df.head(10)
```

Out[28]:

	RollNo	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result
0	1	NaN	A	36.0	40.000000	41.0	48.0	65.0	74.0	268	53.6	Pass
1	2	NaN	A	88.0	75.000000	NaN	NaN	42.0	NaN	117	23.4	Pass
2	3	NaN	A	20.0	54.000000	45.0	46.0	60.0	61.0	266	53.2	Fail
3	4	NaN	A	58.0	64.000000	66.0	67.0	53.0	47.0	297	59.4	Pass
4	5	NaN	A	72.0	53.000000	72.0	58.0	80.0	65.0	328	65.6	Pass
5	6	NaN	A	25.0	49.000000	48.0	73.0	45.0	75.0	290	58.0	Pass
6	7	NaN	A	90.0	75.178571	50.0	67.0	50.0	61.0	228	45.6	Pass
7	8	NaN	A	36.0	48.000000	54.0	74.0	46.0	64.0	286	57.2	Pass
8	9	NaN	A	59.0	78.000000	70.0	72.0	44.0	42.0	306	61.2	Pass
9	10	NaN	A	82.0	57.000000	75.0	70.0	59.0	63.0	324	64.8	Pass

4.Deleting null values using dropna() method

```
In order to drop null values from a dataframe, dropna() function is used. This
function drops Rows/Columns of datasets with Null values in different ways.
1. Dropping rows with at least 1 null value
2. Dropping rows if all values in that row are missing
```

```
3. Dropping columns with at least 1 null value.
4. Dropping Rows with at least 1 null value in CSV file
```

In [27]:
```
df.dropna() #Dropping rows with at least 1 null value
```

Out[27]:

RollNo	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result

In [28]:
```
df.dropna(how="all")   #Dropping rows if all values in that row are missing
```

Out[28]:

	RollNo	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result
0	1	NaN	A	36.0	40.000000	41.0	48.0	65.0	74.0	268	53.6	Pass
1	2	NaN	A	88.0	75.000000	NaN	NaN	42.0	NaN	117	23.4	Pass
2	3	NaN	A	20.0	54.000000	45.0	46.0	60.0	61.0	266	53.2	Fail
3	4	NaN	A	58.0	64.000000	66.0	67.0	53.0	47.0	297	59.4	Pass
4	5	NaN	A	72.0	53.000000	72.0	58.0	80.0	65.0	328	65.6	Pass
5	6	NaN	A	25.0	49.000000	48.0	73.0	45.0	75.0	290	58.0	Pass
6	7	NaN	A	90.0	75.178571	50.0	67.0	50.0	61.0	228	45.6	Pass
7	8	NaN	A	36.0	48.000000	54.0	74.0	46.0	64.0	286	57.2	Pass
8	9	NaN	A	59.0	78.000000	70.0	72.0	44.0	42.0	306	61.2	Pass
9	10	NaN	A	82.0	57.000000	75.0	70.0	59.0	63.0	324	64.8	Pass
10	11	NaN	A	34.0	490.000000	59.0	NaN	67.0	78.0	253	50.6	Pass
11	12	NaN	B	21.0	72.000000	52.0	56.0	49.0	45.0	274	54.8	Pass
12	13	NaN	B	73.0	60.000000	71.0	53.0	69.0	46.0	299	59.8	Pass
13	14	NaN	B	49.0	75.178571	NaN	60.0	47.0	70.0	177	35.4	Pass
14	15	NaN	B	59.0	51.000000	40.0	74.0	48.0	47.0	260	52.0	Fail
15	16	NaN	B	61.0	56.000000	74.0	40.0	54.0	78.0	302	60.4	Pass
16	17	NaN	B	59.0	72.000000	56.0	68.0	63.0	69.0	328	65.6	Pass
17	18	NaN	B	26.0	56.000000	55.0	74.0	75.0	55.0	315	63.0	Pass
18	19	NaN	B	76.0	80.000000	51.0	56.0	66.0	53.0	306	61.2	Pass
19	20	NaN	B	56.0	49.000000	63.0	57.0	73.0	64.0	306	61.2	Pass
20	21	NaN	A	38.0	56.000000	76.0	41.0	NaN	43.0	216	43.2	Pass
21	22	NaN	A	79.0	65.000000	58.0	47.0	60.0	50.0	280	56.0	Pass
22	23	NaN	A	87.0	64.000000	45.0	44.0	44.0	53.0	250	50.0	Pass
23	24	NaN	A	60.0	56.000000	70.0	45.0	47.0	66.0	284	56.8	Pass
24	25	NaN	A	NaN	60.000000	75.0	61.0	65.0	53.0	314	62.8	Pass
25	26	NaN	A	22.0	71.000000	74.0	65.0	64.0	64.0	338	67.6	Pass
26	27	NaN	A	58.0	68.000000	50.0	79.0	65.0	62.0	324	64.8	Pass
27	28	NaN	A	89.0	54.000000	77.0	49.0	75.0	63.0	318	63.6	Pass
28	29	NaN	A	55.0	67.000000	42.0	NaN	49.0	NaN	158	31.6	Pass
29	30	NaN	A	69.0	40.000000	72.0	45.0	73.0	65.0	295	59.0	Fail

In [29]:
```
df.dropna(axis=1) #Dropping columns with at least 1 null value.
```

Out[29]:

	RollNo	Term	Subject1_marks	Total_Marks	Percentage	Result
0	1	A	40.000000	268	53.6	Pass
1	2	A	75.000000	117	23.4	Pass
2	3	A	54.000000	266	53.2	Fail
3	4	A	64.000000	297	59.4	Pass
4	5	A	53.000000	328	65.6	Pass
5	6	A	49.000000	290	58.0	Pass
6	7	A	75.178571	228	45.6	Pass
7	8	A	48.000000	286	57.2	Pass
8	9	A	78.000000	306	61.2	Pass
9	10	A	57.000000	324	64.8	Pass
10	11	A	490.000000	253	50.6	Pass
11	12	B	72.000000	274	54.8	Pass
12	13	B	60.000000	299	59.8	Pass
13	14	B	75.178571	177	35.4	Pass
14	15	B	51.000000	260	52.0	Fail
15	16	B	56.000000	302	60.4	Pass
16	17	B	72.000000	328	65.6	Pass
17	18	B	56.000000	315	63.0	Pass
18	19	B	80.000000	306	61.2	Pass
19	20	B	49.000000	306	61.2	Pass
20	21	A	56.000000	216	43.2	Pass
21	22	A	65.000000	280	56.0	Pass
22	23	A	64.000000	250	50.0	Pass
23	24	A	56.000000	284	56.8	Pass
24	25	A	60.000000	314	62.8	Pass
25	26	A	71.000000	338	67.6	Pass
26	27	A	68.000000	324	64.8	Pass
27	28	A	54.000000	318	63.6	Pass
28	29	A	67.000000	158	31.6	Pass
29	30	A	40.000000	295	59.0	Fail

In [30]:
```
df.dropna(axis=0,how='any',inplace=True)  #Dropping Rows with at least 1 null value in CSV file
```

In [31]:
```
df
```

Out[31]:

RollNo	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result

B. Identification and Handling of Outliers

Identification of Outliers

```
an Outlier is an observation in a given dataset that lies far from the rest
of the observations. That means an outlier is vastly larger or smaller than the remaining
values in the set.
```

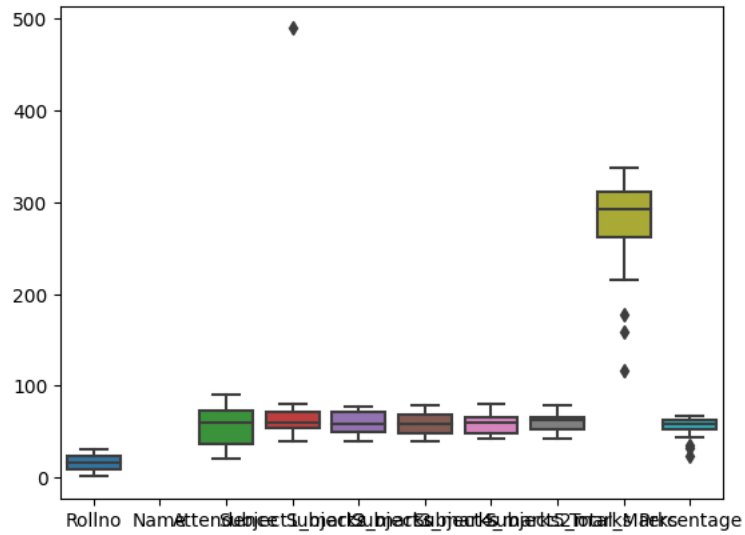
Detecting Outliers

1. Detecting outliers using Boxplot:

In [29]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

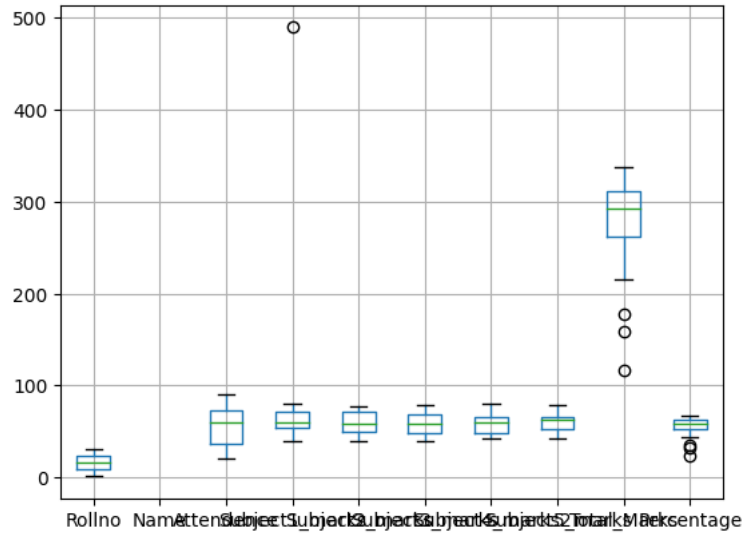
In [30]:
```
sns.boxplot(df)
```

Out[30]: <Axes: >



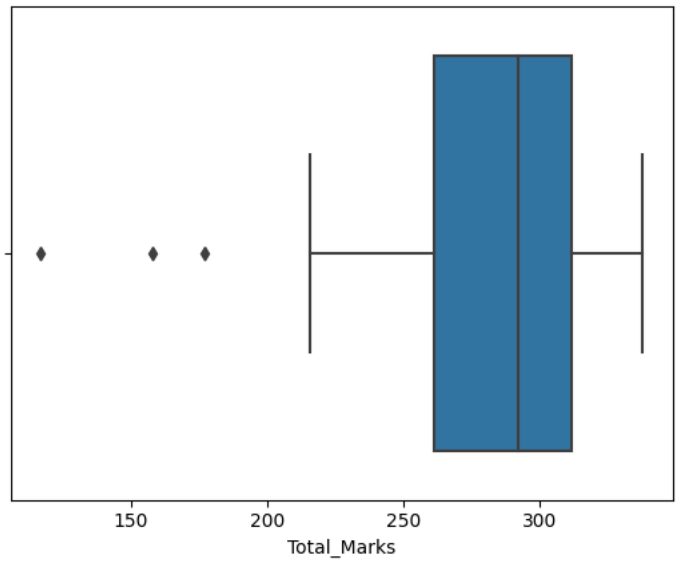
In [31]:
```
df.boxplot()
```

Out[31]: <Axes: >



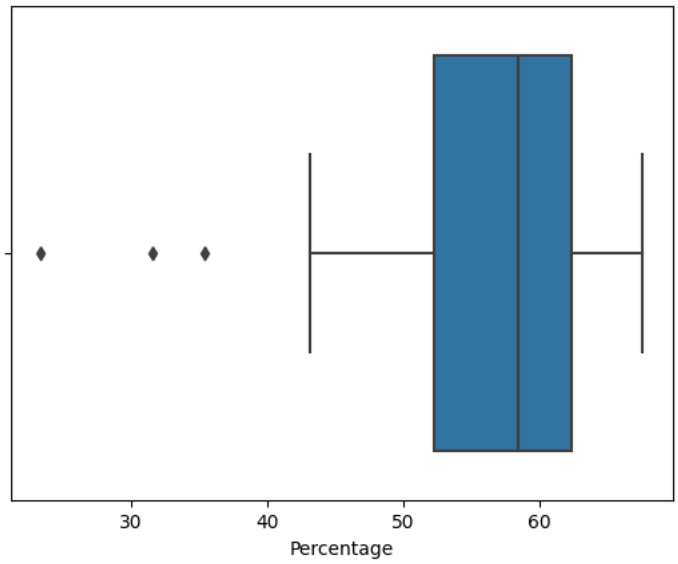
In [32]:
```
sns.boxplot(x=df.Total_Marks)
```

Out[32]:
```
<Axes: xlabel='Total_Marks'>
```



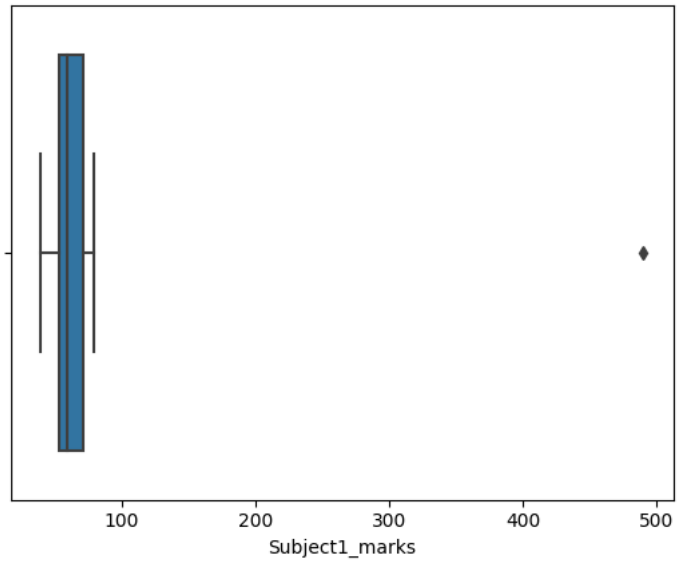
In [33]:
```
sns.boxplot(x=df.Percentage)
```

Out[33]:
```
<Axes: xlabel='Percentage'>
```



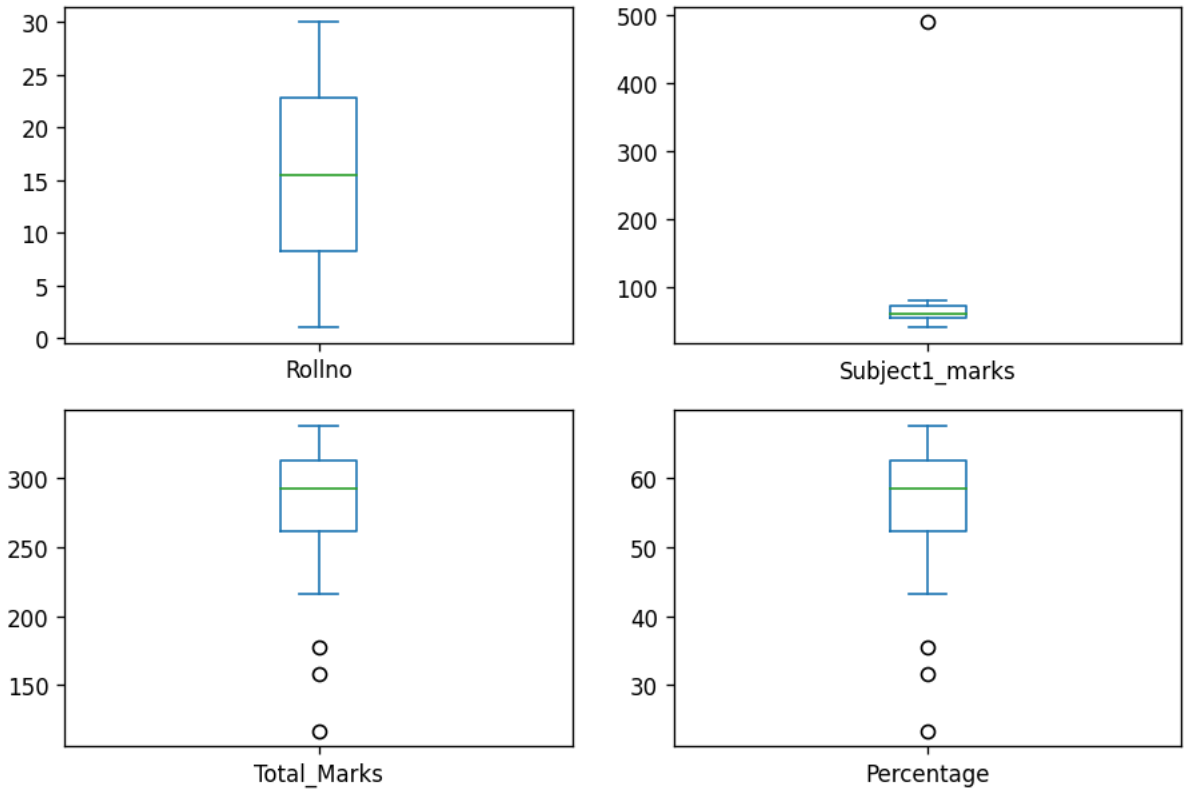
In [34]:
```
sns.boxplot(x=df.Subject1_marks)
```

Out[34]:
```
<Axes: xlabel='Subject1_marks'>
```



In [39]:
```
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (9, 6)
df_list = ['Rollno','Subject1_marks','Total_Marks','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
```



In [ ]:
```

```

2. Detect outlier using z-Score

In [ ]:

3.Detecting outliers using Inter Quantile Range(IQR):

In [40]:
```
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 = 52.3, Q3 = 62.4, IQR = 10.100000000000001, Lower_limit = 37.14999999999999, Upper_limit = 77.55
```

In [41]:
```
df[(df['Percentage'] < Lower_limit) | (df['Percentage'] > Upper_limit)]    # outlier data
```

Out[41]:

	Rollno	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result
1	2	NaN	A	88.0	75.000000	NaN	NaN	42.0	NaN	117	23.4	Pass
13	14	NaN	B	49.0	75.178571	NaN	60.0	47.0	70.0	177	35.4	Pass
28	29	NaN	A	55.0	67.000000	42.0	NaN	49.0	NaN	158	31.6	Pass

Handling of Outliers

1.removing the outlier:

In [42]:
```
outliers=[]
for i in df.Percentage:
    if i<Lower_limit or i>Upper_limit:
        outliers.append(i)
print("outliers are",outliers)
```

```
outliers are [23.4, 35.4, 31.6]
```

In [43]:
```
Upper_limit
```

Out[43]:
```
77.55
```

In [47]:
```
Lower_limit
```

Out[47]:
```
37.14999999999999
```

In [48]:
```
df[df.Percentage<Lower_limit].index
```

Out[48]:
```
Int64Index([1, 13, 28], dtype='int64')
```

In [49]:
```
df1=df.drop(df[df.Percentage<Lower_limit].index)       #normal data without outlier
```

In [51]:
```
df1.shape
```

Out[51]:
```
(27, 12)
```

In [55]:
```
#outlier data
df2=df[df.Percentage<Lower_limit]
df2
```

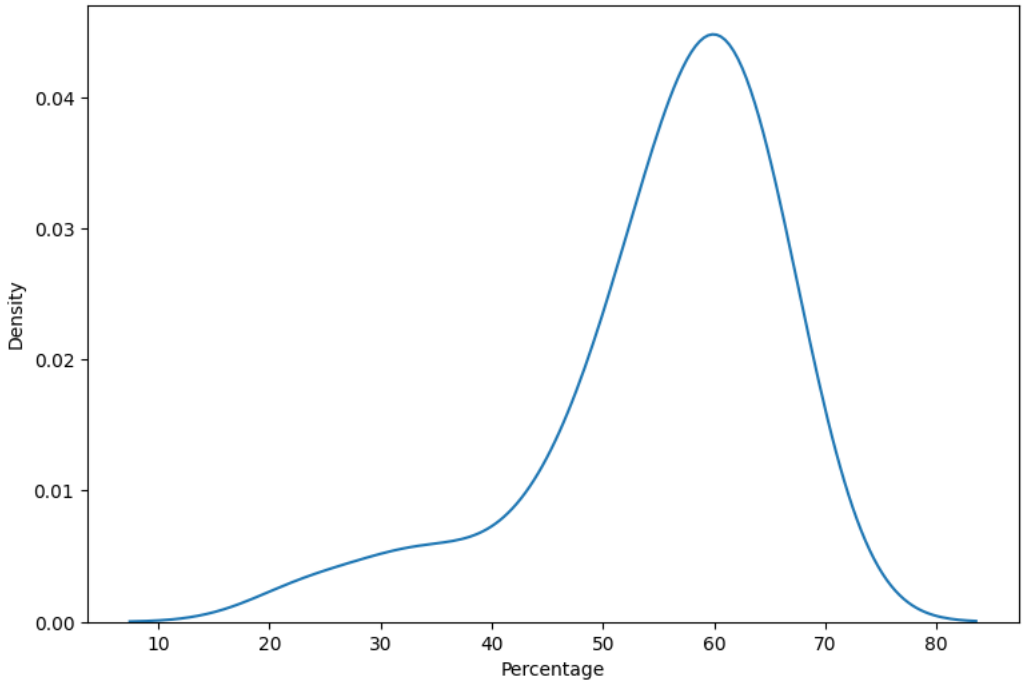
Out[55]:

	Rollno	Name	Term	Attendence	Subject1_marks	Subject2_marks	Subject3_marks	Subject4_marks2	Subject5_marks	Total_Marks	Percentage	Result
1	2	NaN	A	88.0	75.000000	NaN	NaN	42.0	NaN	117	23.4	Pass
13	14	NaN	B	49.0	75.178571	NaN	60.0	47.0	70.0	177	35.4	Pass
28	29	NaN	A	55.0	67.000000	42.0	NaN	49.0	NaN	158	31.6	Pass

2.Mean/Median imputation

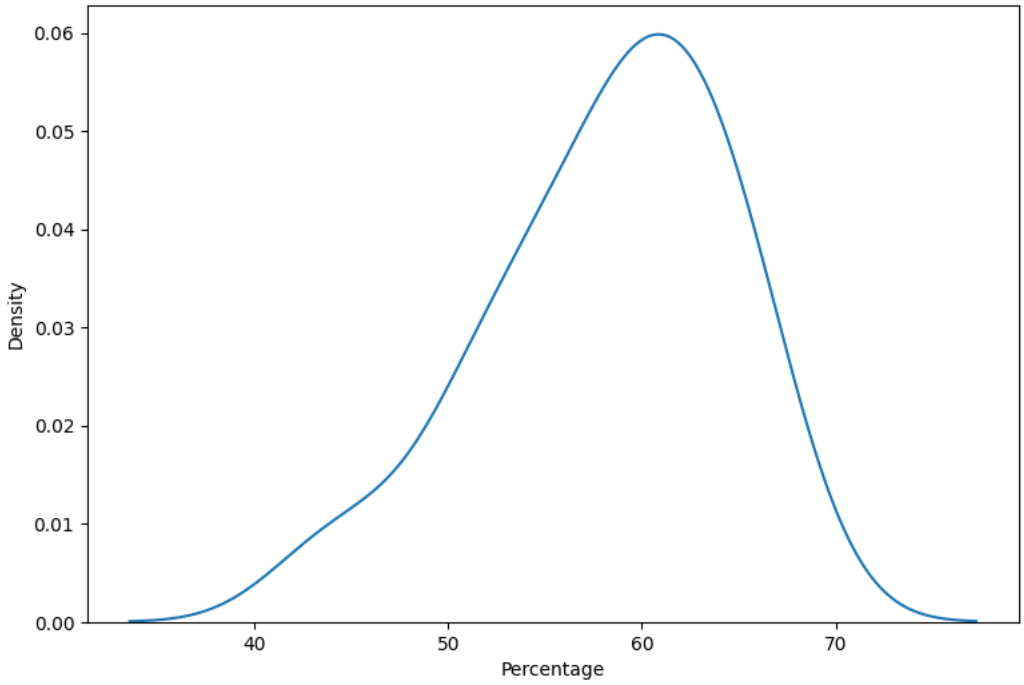
In [67]:
```
sns.kdeplot(df.Percentage)
```

Out[67]:
```
<Axes: xlabel='Percentage', ylabel='Density'>
```



In [52]:
```
sns.kdeplot(df1.Percentage)
```

Out[52]:
```
<Axes: xlabel='Percentage', ylabel='Density'>
```



In [56]:
```
df.Percentage
```

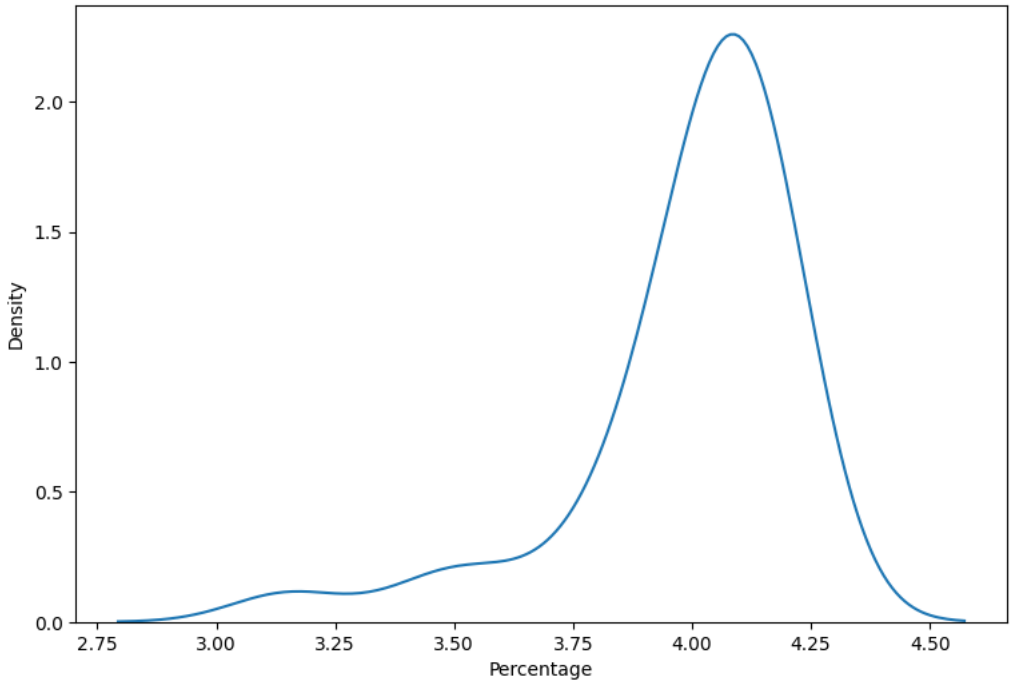
Out[56]:
```
0     53.6
1     23.4
2     53.2
3     59.4
4     65.6
5     58.0
6     45.6
7     57.2
8     61.2
9     64.8
10    50.6
11    54.8
12    59.8
13    35.4
14    52.0
15    60.4
16    65.6
17    63.0
18    61.2
19    61.2
20    43.2
21    56.0
22    50.0
23    56.8
24    62.8
25    67.6
26    64.8
27    63.6
28    31.6
29    59.0
Name: Percentage, dtype: float64
```

In [57]:
```
log_percentage=np.log(df.Percentage)
log_percentage
```

Out[57]:
```
0     3.981549
1     3.152736
2     3.974058
3     4.084294
4     4.183576
5     4.060443
6     3.819908
7     4.046554
8     4.114147
9     4.171306
10    3.923952
11    4.003690
12    4.091006
13    3.566712
14    3.951244
15    4.100989
16    4.183576
17    4.143135
18    4.114147
19    4.114147
20    3.765840
21    4.025352
22    3.912023
23    4.039536
24    4.139955
25    4.213608
26    4.171306
27    4.152613
28    3.453157
29    4.077537
Name: Percentage, dtype: float64
```

In [58]:
```
sns.kdeplot(log_percentage)
```

Out[58]: <Axes: xlabel='Percentage', ylabel='Density'>



C. Data Transformation

to change the scale for better understanding of the variable

In [ ]:
```

```

to convert a non-linear relation into a linear one

In [ ]:
```

```

decrease the skewness and convert the distribution into a normal distribution

1. Checking the distribution with Skewness

In [13]:
```
import seaborn as sns
```

In [11]:
```
#skewness in the data
df.skew()
```

```
C:\Users\P-One\AppData\Local\Temp\ipykernel_4484\3453824435.py:2: FutureWarning: The default value of numeric_only in DataFram
e.skew is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this warning.
  df.skew()
```

Out[11]:
```
Rollno            0.000000
Name                   NaN
Attendence       -0.174993
Subject1_marks    5.151540
Subject2_marks   -0.104589
Subject3_marks   -0.001637
Subject4_marks2   0.169546
Subject5_marks   -0.062086
Total_Marks      -1.563059
Percentage       -1.563059
dtype: float64
```

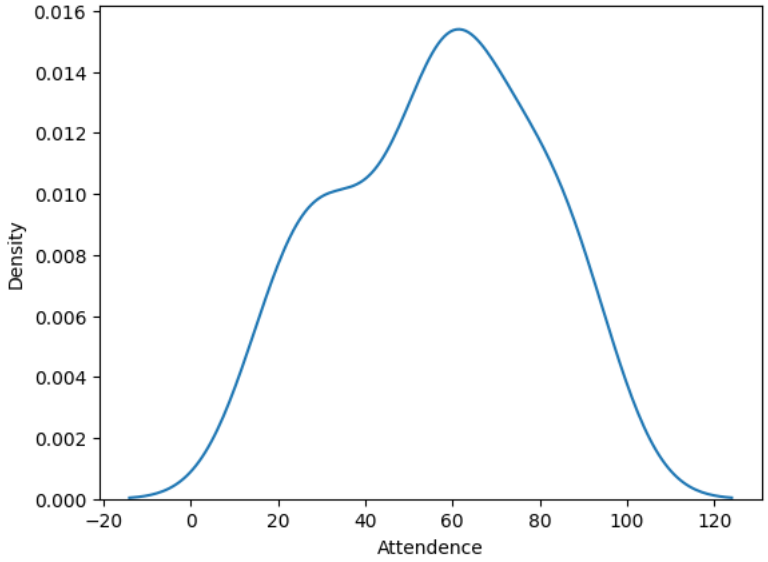
```
From the above result, we can check which variable is normally distributed and which is not.

The variables with skewness > 1 such as Subject1_marks is highly positively skewed.
The variables with skewness < -1 are highly negatively skewed.
The variables with 0.5 < skewness < 1 such as moderately positively skewed.
The variables with -0.5 < skewness < -1 such as stroke are moderately negatively skewed.
```

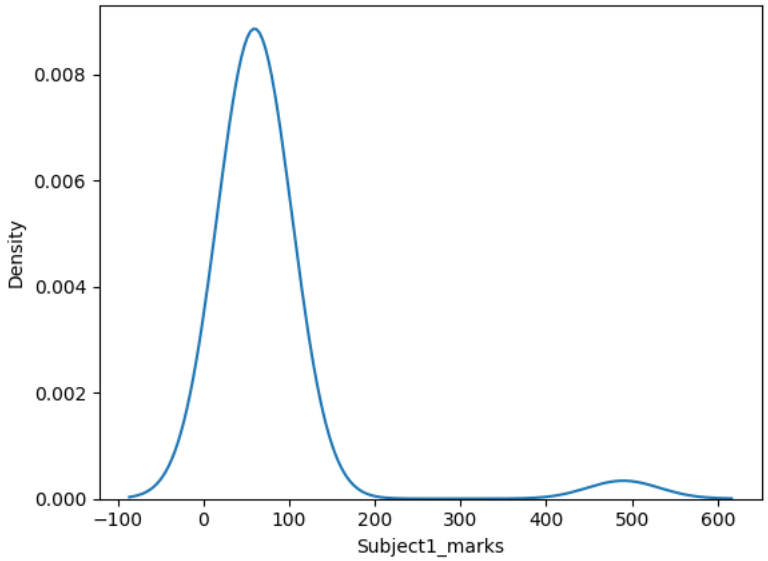
And, the variables with -0.5 < skewness < 0.5 are symmetric i.e normally distributed such as symboling, carheight, boreration, peakrpm, highwaympg.

Checking the distribution of variables using KDE plot

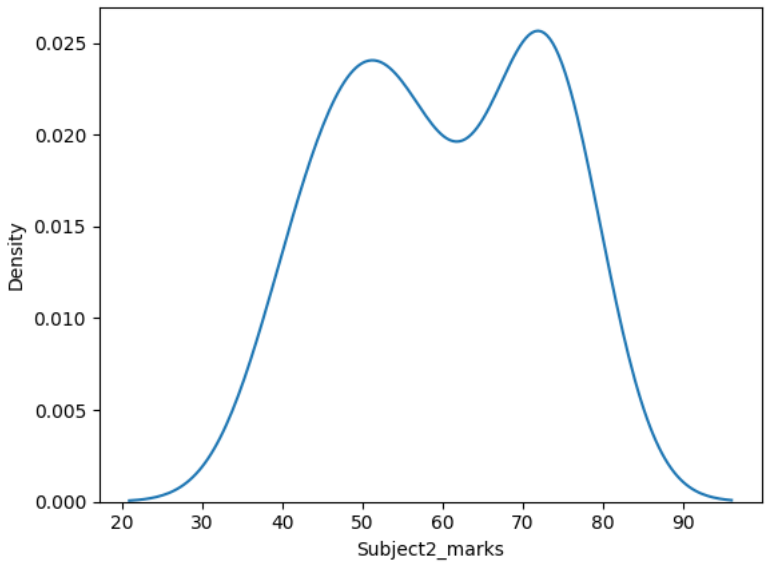
In [15]:
```
sns.kdeplot(df.Attendence);
```



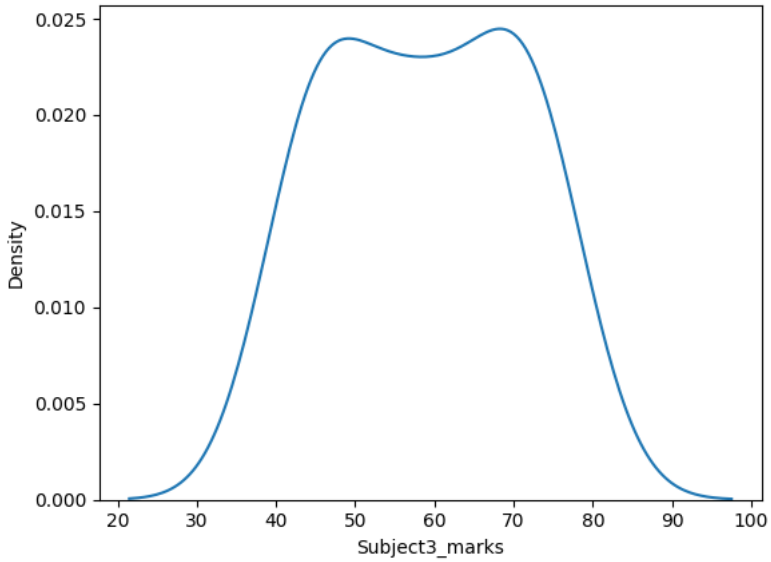
In [14]:
```
sns.kdeplot(df.Subject1_marks);
```



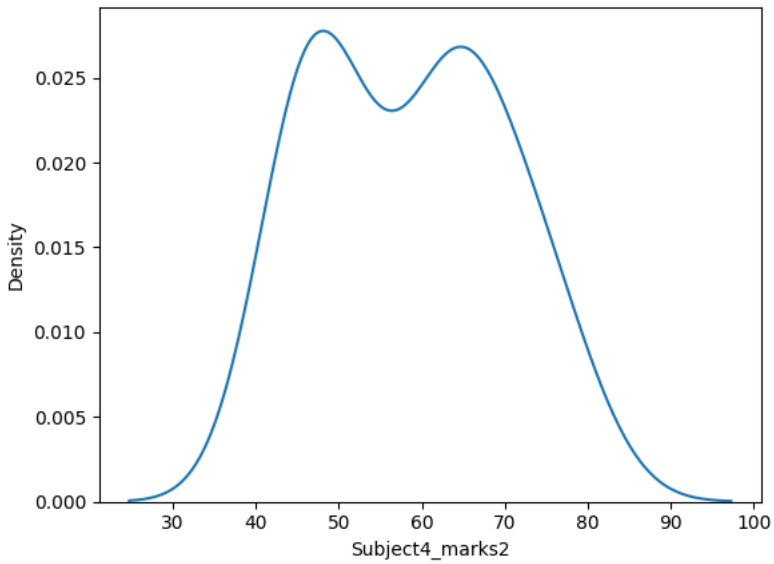
In [16]:
```
sns.kdeplot(df.Subject2_marks);
```



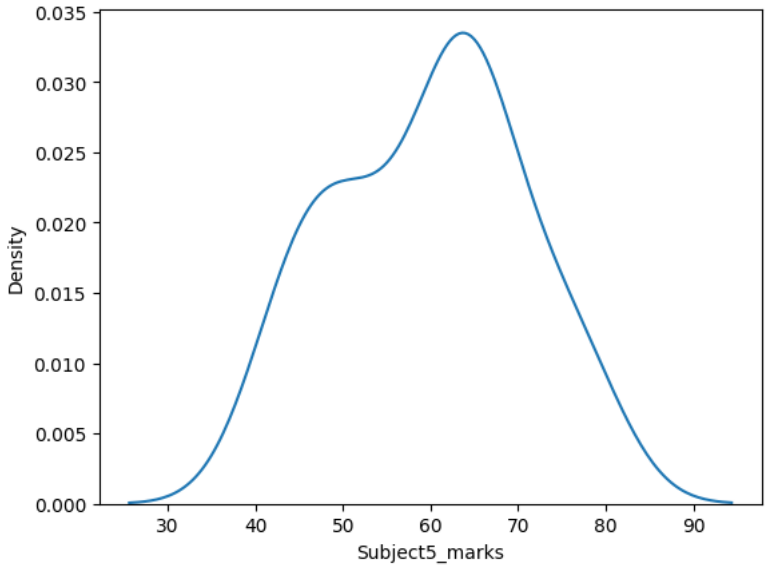
In [17]:
```
sns.kdeplot(df.Subject3_marks);
```



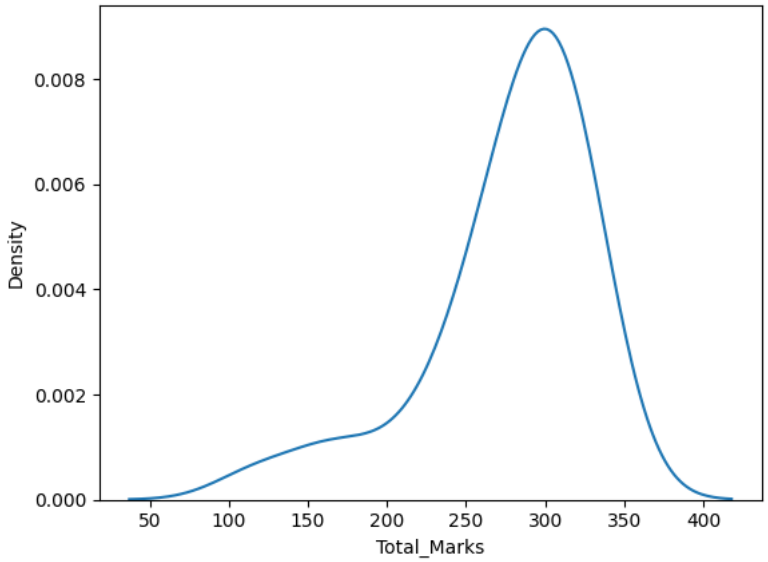
In [19]:
```
sns.kdeplot(df.Subject4_marks2);
```



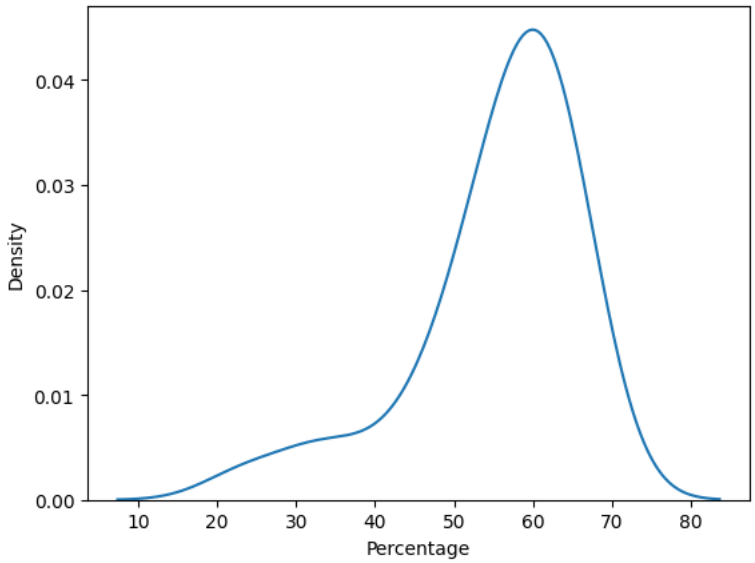
In [20]:
```
sns.kdeplot(df.Subject5_marks);
```



In [21]:
```
sns.kdeplot(df.Total_Marks);
```



In [22]:
```
sns.kdeplot(df.Percentage);
```



In [ ]:

*__Conclusion__

In this way we have explored the functions of the python library for Data Preprocessing, Data Wrangling Techniques and How to Handle missing values and outliers also applied data transformation.

In addition to the codes and outputs, explain every operation that you do in the above steps and explain everything that you do to import/read/scrape the data set.