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.

1.Import all the required Python Libraries.

```
# Pandas is a popular Python library used for data manipulation and analysis
import pandas as pd


# NumPy is a fundamental library in Python for numerical computing
import numpy as np
```

2. Creation of Dataset using Microsoft Excel.

```
# pd.read_csv() is a function provided by the pandas library in Python, used to read data from a CSV (Comma-Separated Values) file and crea

data=pd.read_csv("/content/sample_data/Academic-Performance-Dataset.csv")
```

```
data
```

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS
0	1	Mohammed	M	Comp	72.0	62.0	98.0	63.0	
1	2	Reyansh	M	IT	58.0	62.0	83.0	83.0	
2	3	Aarav	M	IT	57.0	-20.0	100.0	NaN	
3	4	Atharv	M	IT	60.0	89.0	83.0	70.0	
4	5	Vivaan	M	Comp	85.0	90.0	NaN	78.0	
5	6	Advik	M	ENTC	94.0	99.0	84.0	100.0	
6	7	Ansh	M	ENTC	98.0	88.0	95.0	81.0	
7	8	Ishaan	M	ENTC	75.0	66.0	51.0	83.0	
8	9	Dhruv	M	ENTC	63.0	NaN	NaN	97.0	
9	10	Siddharth	M	ENTC	96.0	67.0	78.0	95.0	
10	11	Vihaan	M	ENTC	82.0	54.0	70.0	88.0	
11	12	NaN	M	IT	75.0	64.0	67.0	71.0	
12	13	Aarush	M	IT	67.0	56.0	81.0	NaN	
13	14	Leo	M	IT	98.0	-34.0	70.0	94.0	
14	15	Maryam	F	IT	64.0	87.0	60.0	90.0	
15	16	Saanvi	F	Comp	66.0	90.0	95.0	67.0	
16	17	Zaranew	F	Comp	93.0	54.0	NaN	75.0	
17	18	Inaya	F	Comp	74.0	67.0	93.0	93.0	
18	19	Aarya	F	Comp	72.0	88.0	84.0	81.0	

Next steps: Generate code with data View recommended plots

```
# In pandas, data.head() is a method used to display the first few rows of a DataFrame or Series

data.head()
```

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS_
0	1	Mohammed	M	Comp	72.0	62.0	98.0	63.0	
1	2	Reyansh	M	IT	58.0	62.0	83.0	83.0	
2	3	Aarav	M	IT	57.0	-20.0	100.0	NaN	
3	4	Atharv	M	IT	60.0	89.0	83.0	70.0	

Next steps: Generate code with data View recommended plots

```
# data.tail() displays the last few rows.

data.tail()
```

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS_ma
15	16	Saanvi	F	Comp	66.0	90.0	95.0	67.0	
16	17	Zaranew	F	Comp	93.0	54.0	NaN	75.0	
17	18	Inaya	F	Comp	74.0	67.0	93.0	93.0	
18	19	Aarya	F	Comp	72.0	88.0	84.0	81.0	

```
# In pandas, data.info() is a method used to print a concise summary of a DataFrame, including information about the data types, non-null v

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Rollno       20 non-null     int64  
 1   Name         18 non-null     object 
 2   Gender       20 non-null     object 
 3   Branch       20 non-null     object 
 4   Attendence   20 non-null     float64
 5   Phy_marks    19 non-null     float64
 6   Che_marks    17 non-null     float64
 7   EM1_marks    18 non-null     float64
 8   PPS_marks    19 non-null     float64
 9   SME_marks    20 non-null     float64
 10  Total Marks  20 non-null     int64  
 11  Percentage   20 non-null     float64
dtypes: float64(7), int64(2), object(3)
memory usage: 2.0+ KB
```

```
# In pandas, data.describe() is a method used to generate descriptive statistics of numerical columns in a DataFrame
data.describe()
```

	Rollno	Attendence	Phy_marks	Che_marks	EM1_marks	PPS_marks	SME_marks	
count	20.00000	20.000000	19.000000	17.000000	18.000000	19.000000	20.000000	2(
mean	10.50000	75.100000	63.421053	80.764706	83.444444	60.736842	62.700000	32
std	5.91608	14.660724	34.940133	13.690916	11.078449	43.598983	24.891554	7
min	1.00000	53.000000	-34.000000	51.000000	63.000000	-99.000000	23.000000	19
25%	5.75000	63.750000	59.000000	70.000000	75.750000	56.000000	42.750000	27
50%	10.50000	73.000000	67.000000	83.000000	83.000000	66.000000	60.500000	33
75%	15.25000	87.000000	88.000000	93.000000	93.000000	87.500000	80.250000	38

```
# When you use data.describe(include="all"), it provides descriptive statistics for all columns in the DataFrame, including both numerical
data.describe(include="all")
```

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_mar
count	20.00000	18	20	20	20.000000	19.000000	17.000000	18.0000
unique	NaN	18	2	3	NaN	NaN	NaN	Na
top	NaN	Mohammed	M	Comp	NaN	NaN	NaN	Na
freq	NaN	1	14	7	NaN	NaN	NaN	Na
mean	10.50000	NaN	NaN	NaN	75.100000	63.421053	80.764706	83.4444
std	5.91608	NaN	NaN	NaN	14.660724	34.940133	13.690916	11.0784
min	1.00000	NaN	NaN	NaN	53.000000	-34.000000	51.000000	63.0000
25%	5.75000	NaN	NaN	NaN	63.750000	59.000000	70.000000	75.7500
50%	10.50000	NaN	NaN	NaN	73.000000	67.000000	83.000000	83.0000
75%	15.25000	NaN	NaN	NaN	87.000000	88.000000	93.000000	93.0000

```
data.shape
```

```
(20, 12)
```

```
# data.dtypes is a pandas attribute that returns a Series containing the data types of each column in the DataFrame.
data.dtypes
```

```
Rollno           int64
Name            object
Gender          object
Branch          object
Attendence     float64
Phy_marks      float64
Che_marks      float64
EM1_marks      float64
PPS_marks      float64
SME_marks      float64
Total Marks      int64
Percentage     float64
dtype: object
```

```
data.columns
```

```
Index(['Rollno', 'Name', 'Gender', 'Branch', 'Attendence', 'Phy_marks',
       'Che_marks', 'EM1_marks', 'PPS_marks', 'SME_marks', 'Total Marks',
       'Percentage'],
      dtype='object')
```

```
data[0:4]
```

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS_
0	1	Mohammed	M	Comp	72.0	62.0	98.0	63.0	
1	2	Reyansh	M	IT	58.0	62.0	83.0	83.0	
2	3	Aarav	M	IT	57.0	-20.0	100.0	NaN	

```
# data.loc[0:2] is a method used to access rows of a DataFrame by label, particularly when the labels are integers
data.loc[0:2]
```

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS_
0	1	Mohammed	M	Comp	72.0	62.0	98.0	63.0	
1	2	Reyansh	M	IT	58.0	62.0	83.0	83.0	

```
data.loc[0:2,'Attendence']
```

```
0    72.0
1    58.0
2    57.0
Name: Attendence, dtype: float64
```

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

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS_mar
1	2	Reyansh	M	IT	58.0	62.0	83.0	83.0	8

A.Identification and Handling of Null Values

```
# Check for missing values in the data using pandas isnull()
data.isnull()
```

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS_mark
0	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	True	Fals
3	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	True	False	Fals
5	False	False	False	False	False	False	False	False	Fals
6	False	False	False	False	False	False	False	False	Fals
7	False	False	False	False	False	False	False	False	Fals
8	False	False	False	False	False	True	True	False	Fals
9	False	False	False	False	False	False	False	False	Tru
10	False	False	False	False	False	False	False	False	Fals
11	False	True	False	False	False	False	False	False	Fals
12	False	False	False	False	False	False	False	True	Fals
13	False	False	False	False	False	False	False	False	Fals
14	False	False	False	False	False	False	False	False	Fals
15	False	False	False	False	False	False	False	False	Fals
16	False	False	False	False	False	False	True	False	Fals
17	False	False	False	False	False	False	False	False	Fals
18	False	False	False	False	False	False	False	False	Fals

```
data.isna()
```

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS_mark
0	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	True	Fals
3	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	True	False	Fals
5	False	False	False	False	False	False	False	False	Fals
6	False	False	False	False	False	False	False	False	Fals
7	False	False	False	False	False	False	False	False	Fals
8	False	False	False	False	False	True	True	False	Fals
9	False	False	False	False	False	False	False	False	Tru
10	False	False	False	False	False	False	False	False	Fals
11	False	True	False	False	False	False	False	False	Fals
12	False	False	False	False	False	False	False	True	Fals
13	False	False	False	False	False	False	False	False	Fals
14	False	False	False	False	False	False	False	False	Fals
15	False	False	False	False	False	False	False	False	Fals
16	False	False	False	False	False	False	True	False	Fals
17	False	False	False	False	False	False	False	False	Fals
18	False	False	False	False	False	False	False	False	Fals

```
data.isnull().any()
```

```
Rollno         False
Name            True
Gender         False
Branch         False
Attendence     False
Phy_marks       True
Che_marks       True
EM1_marks       True
PPS_marks       True
SME_marks      False
Total Marks    False
Percentage     False
dtype: bool
```

```
data.isnull().sum()
```

```
Rollno         0
Name           2
Gender         0
Branch         0
Attendence     0
Phy_marks      1
Che_marks      3
EM1_marks      2
PPS_marks      1
SME_marks      0
Total Marks    0
Percentage     0
dtype: int64
```

```
data['Name'].isnull().any()
```

```
True
```

Make a list of column having missing value

```
col_NAN_val=[]
for col in data:
  if data[col].isnull().any():
    col_NAN_val.append(col)
col_NAN_val
```

```
['Name', 'Phy_marks', 'Che_marks', 'EM1_marks', 'PPS_marks']
```

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

1. replacing null values with NaN

```
# np.nan: This is a constant from the NumPy library (numpy.nan) representing a missing value (NaN). It's the value that you want to replace
data.replace(np.nan,value=0,inplace=True)
```

```
data['Name'].isnull().any()
```

```
False
```

2. Filling null values with fillna()

```
data.fillna(1)
```

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS
0	1	Mohammed	M	Comp	72.0	62.0	98.0	63.0	
1	2	Reyansh	M	IT	58.0	62.0	83.0	83.0	
2	3	Aarav	M	IT	57.0	-20.0	100.0	0.0	
3	4	Atharv	M	IT	60.0	89.0	83.0	70.0	
4	5	Vivaan	M	Comp	85.0	90.0	0.0	78.0	
5	6	Advik	M	ENTC	94.0	99.0	84.0	100.0	
6	7	Ansh	M	ENTC	98.0	88.0	95.0	81.0	
7	8	Ishaan	M	ENTC	75.0	66.0	51.0	83.0	
8	9	Dhruv	M	ENTC	63.0	0.0	0.0	97.0	
9	10	Siddharth	M	ENTC	96.0	67.0	78.0	95.0	
10	11	Vihaan	M	ENTC	82.0	54.0	70.0	88.0	
11	12	0	M	IT	75.0	64.0	67.0	71.0	
12	13	Aarush	M	IT	67.0	56.0	81.0	0.0	
13	14	Leo	M	IT	98.0	-34.0	70.0	94.0	
14	15	Maryam	F	IT	64.0	87.0	60.0	90.0	
15	16	Saanvi	F	Comp	66.0	90.0	95.0	67.0	
16	17	Zaranew	F	Comp	93.0	54.0	0.0	75.0	
17	18	Inaya	F	Comp	74.0	67.0	93.0	93.0	
18	19	Aarya	F	Comp	72.0	88.0	84.0	81.0	

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

```
data['Phy_marks']=data['Phy_marks'].fillna(data['Phy_marks'].mean())
```

```
data
```

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS
0	1	Mohammed	M	Comp	72.0	62.0	98.0	63.0	
1	2	Reyansh	M	IT	58.0	62.0	83.0	83.0	
2	3	Aarav	M	IT	57.0	-20.0	100.0	0.0	
3	4	Atharv	M	IT	60.0	89.0	83.0	70.0	
4	5	Vivaan	M	Comp	85.0	90.0	0.0	78.0	
5	6	Advik	M	ENTC	94.0	99.0	84.0	100.0	
6	7	Ansh	M	ENTC	98.0	88.0	95.0	81.0	
7	8	Ishaan	M	ENTC	75.0	66.0	51.0	83.0	
8	9	Dhruv	M	ENTC	63.0	0.0	0.0	97.0	
9	10	Siddharth	M	ENTC	96.0	67.0	78.0	95.0	
10	11	Vihaan	M	ENTC	82.0	54.0	70.0	88.0	
11	12	0	M	IT	75.0	64.0	67.0	71.0	
12	13	Aarush	M	IT	67.0	56.0	81.0	0.0	
13	14	Leo	M	IT	98.0	-34.0	70.0	94.0	
14	15	Maryam	F	IT	64.0	87.0	60.0	90.0	
15	16	Saanvi	F	Comp	66.0	90.0	95.0	67.0	
16	17	Zaranew	F	Comp	93.0	54.0	0.0	75.0	
17	18	Inaya	F	Comp	74.0	67.0	93.0	93.0	
18	19	Aarya	F	Comp	72.0	88.0	84.0	81.0	

Next steps: Generate code with `data` View recommended plots

```
data.isnull().any()
```

```
Rollno         False
Name           False
Gender         False
Branch         False
Attendence     False
Phy_marks      False
Che_marks      False
EM1_marks      False
PPS_marks      False
SME_marks      False
Total Marks    False
Percentage     False
dtype: bool
```

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

```
data.dropna() ##Dropping rows with at least 1 null value
```

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS
0	1	Mohammed	M	Comp	72.0	62.0	98.0	63.0	
1	2	Reyansh	M	IT	58.0	62.0	83.0	83.0	
2	3	Aarav	M	IT	57.0	-20.0	100.0	0.0	
3	4	Atharv	M	IT	60.0	89.0	83.0	70.0	
4	5	Vivaan	M	Comp	85.0	90.0	0.0	78.0	
5	6	Advik	M	ENTC	94.0	99.0	84.0	100.0	
6	7	Ansh	M	ENTC	98.0	88.0	95.0	81.0	
7	8	Ishaan	M	ENTC	75.0	66.0	51.0	83.0	
8	9	Dhruv	M	ENTC	63.0	0.0	0.0	97.0	
9	10	Siddharth	M	ENTC	96.0	67.0	78.0	95.0	
10	11	Vihaan	M	ENTC	82.0	54.0	70.0	88.0	
11	12	0	M	IT	75.0	64.0	67.0	71.0	
12	13	Aarush	M	IT	67.0	56.0	81.0	0.0	
13	14	Leo	M	IT	98.0	-34.0	70.0	94.0	
14	15	Maryam	F	IT	64.0	87.0	60.0	90.0	
15	16	Saanvi	F	Comp	66.0	90.0	95.0	67.0	
16	17	Zaranew	F	Comp	93.0	54.0	0.0	75.0	
17	18	Inaya	F	Comp	74.0	67.0	93.0	93.0	
18	19	Aarya	F	Comp	72.0	88.0	84.0	81.0	

```
data.dropna(how="all") #Dropping rows if all values in that row are missing
```

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS
0	1	Mohammed	M	Comp	72.0	62.0	98.0	63.0	
1	2	Reyansh	M	IT	58.0	62.0	83.0	83.0	
2	3	Aarav	M	IT	57.0	-20.0	100.0	0.0	
3	4	Atharv	M	IT	60.0	89.0	83.0	70.0	
4	5	Vivaan	M	Comp	85.0	90.0	0.0	78.0	
5	6	Advik	M	ENTC	94.0	99.0	84.0	100.0	
6	7	Ansh	M	ENTC	98.0	88.0	95.0	81.0	
7	8	Ishaan	M	ENTC	75.0	66.0	51.0	83.0	
8	9	Dhruv	M	ENTC	63.0	0.0	0.0	97.0	
9	10	Siddharth	M	ENTC	96.0	67.0	78.0	95.0	
10	11	Vihaan	M	ENTC	82.0	54.0	70.0	88.0	
11	12	0	M	IT	75.0	64.0	67.0	71.0	
12	13	Aarush	M	IT	67.0	56.0	81.0	0.0	
13	14	Leo	M	IT	98.0	-34.0	70.0	94.0	
14	15	Maryam	F	IT	64.0	87.0	60.0	90.0	
15	16	Saanvi	F	Comp	66.0	90.0	95.0	67.0	
16	17	Zaranew	F	Comp	93.0	54.0	0.0	75.0	
17	18	Inaya	F	Comp	74.0	67.0	93.0	93.0	
18	19	Aarya	F	Comp	72.0	88.0	84.0	81.0	

```
data.dropna(axis=1) #Dropping columns with at least 1 null value.
```

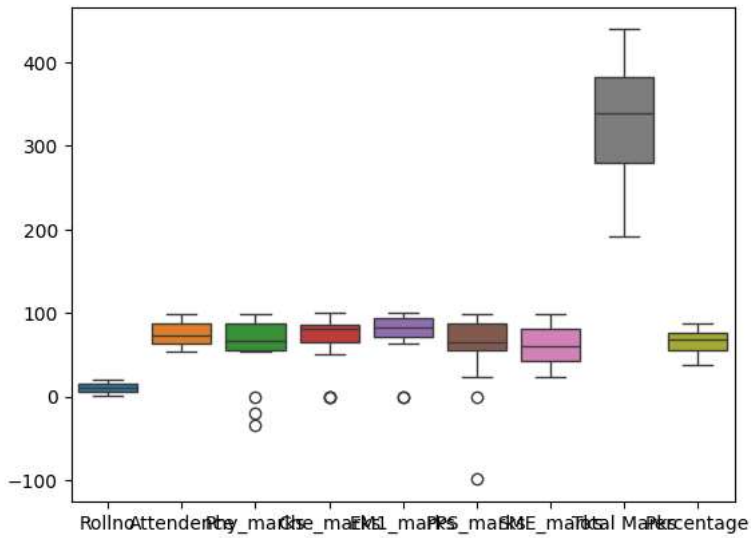
	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS
0	1	Mohammed	M	Comp	72.0	62.0	98.0	63.0	
1	2	Reyansh	M	IT	58.0	62.0	83.0	83.0	
2	3	Aarav	M	IT	57.0	-20.0	100.0	0.0	
3	4	Atharv	M	IT	60.0	89.0	83.0	70.0	
4	5	Vivaan	M	Comp	85.0	90.0	0.0	78.0	
5	6	Advik	M	ENTC	94.0	99.0	84.0	100.0	
6	7	Ansh	M	ENTC	98.0	88.0	95.0	81.0	
7	8	Ishaan	M	ENTC	75.0	66.0	51.0	83.0	
8	9	Dhruv	M	ENTC	63.0	0.0	0.0	97.0	
9	10	Siddharth	M	ENTC	96.0	67.0	78.0	95.0	
10	11	Vihaan	M	ENTC	82.0	54.0	70.0	88.0	
11	12	0	M	IT	75.0	64.0	67.0	71.0	
12	13	Aarush	M	IT	67.0	56.0	81.0	0.0	
13	14	Leo	M	IT	98.0	-34.0	70.0	94.0	
14	15	Maryam	F	IT	64.0	87.0	60.0	90.0	
15	16	Saanvi	F	Comp	66.0	90.0	95.0	67.0	
16	17	Zaranew	F	Comp	93.0	54.0	0.0	75.0	
17	18	Inaya	F	Comp	74.0	67.0	93.0	93.0	
18	19	Aarya	F	Comp	72.0	88.0	84.0	81.0	

```
data.dropna(axis=0,how="any",inplace=True) ##Dropping Rows with at least 1 null value in CSV file
```

B .Identification and Handling of Outliers

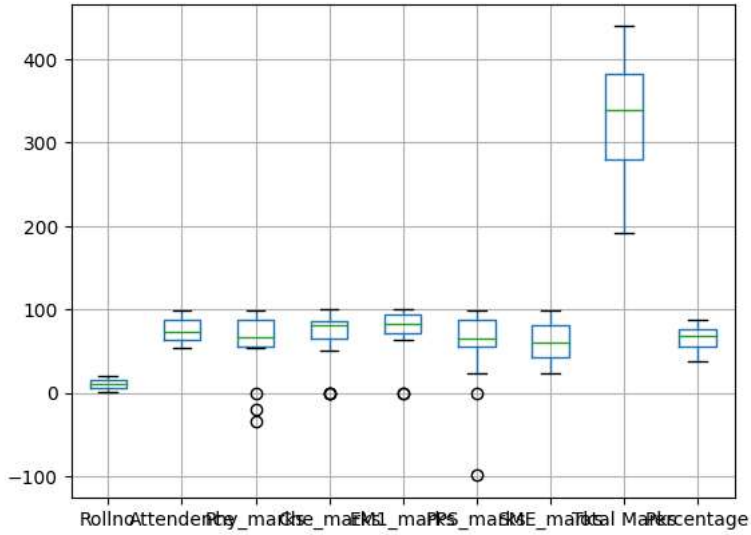
```
#1. Detecting outliers using Boxplot:

import seaborn as sns #Seaborn is a popular Python visualization library built on top of Matplotlib that provides a high-level interface fo

# Matplotlib is a powerful Python plotting library that provides a wide range of functionalities for creating static, interactive, and publ
# plotting functions more conveniently and concisely in your code.
import matplotlib.pyplot as plt

# A box plot, also known as a box-and-whisker plot, is a graphical representation of the distribution of a dataset based on five summary st
sns.boxplot(data)
```

```
<Axes: >
```



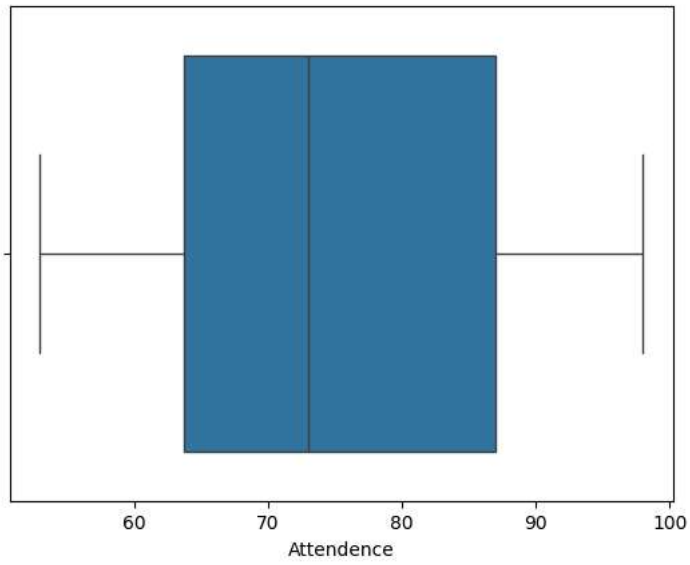

```
data.boxplot()
```

```
<Axes: >
```



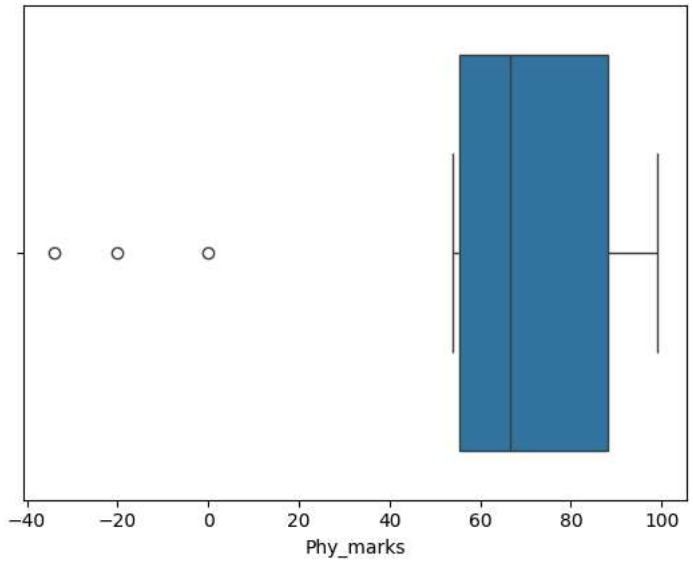

```
sns.boxplot(x=data.Attendence)
```

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



```
sns.boxplot(x=data.Phy_marks)
```

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




```
# 2 . Detecting outliers using Inter Quantile Range(IQR):


Q1=data['Phy_marks'].quantile(0.25)
Q3=data['Phy_marks'].quantile(0.75)
IQR=Q3-Q1

Lower_limit=Q1-1.5*IQR
Upper_limit=Q3+1.5*IQR

print(f'Q1={Q1},Q2={Q3},IQR={IQR},Lower_limit={Lower_limit},Upper_limit={Upper_limit}')
```

```
Q1=55.5,Q2=88.0,IQR=32.5,Lower_limit=6.75,Upper_limit=136.75
```

```
# 3 Handling of Outliers

# 1.Removing the outlier:

data[data['Phy_marks'] < Lower_limit].index
```

```
Index([2, 8, 13], dtype='int64')
```

```
data[data['Phy_marks'] < Upper_limit].index
```

```
RangeIndex(start=0, stop=20, step=1)
```

```
data1=data.drop(data[data['Phy_marks'] < Lower_limit].index)

data1
```

	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS
0	1	Mohammed	M	Comp	72.0	62.0	98.0	63.0	
1	2	Reyansh	M	IT	58.0	62.0	83.0	83.0	
3	4	Atharv	M	IT	60.0	89.0	83.0	70.0	
4	5	Vivaan	M	Comp	85.0	90.0	0.0	78.0	
5	6	Advik	M	ENTC	94.0	99.0	84.0	100.0	
6	7	Ansh	M	ENTC	98.0	88.0	95.0	81.0	
7	8	Ishaan	M	ENTC	75.0	66.0	51.0	83.0	
9	10	Siddharth	M	ENTC	96.0	67.0	78.0	95.0	
10	11	Vihaan	M	ENTC	82.0	54.0	70.0	88.0	
11	12	0	M	IT	75.0	64.0	67.0	71.0	
12	13	Aarush	M	IT	67.0	56.0	81.0	0.0	
14	15	Maryam	F	IT	64.0	87.0	60.0	90.0	
15	16	Saanvi	F	Comp	66.0	90.0	95.0	67.0	
16	17	Zaranew	F	Comp	93.0	54.0	0.0	75.0	
17	18	Inaya	F	Comp	74.0	67.0	93.0	93.0	
18	19	Aarya	F	Comp	72.0	88.0	84.0	81.0	

Next steps: Generate code with `data1` View recommended plots

```
data2=data[data['Phy_marks'] < Lower_limit]
```

```
data2
```

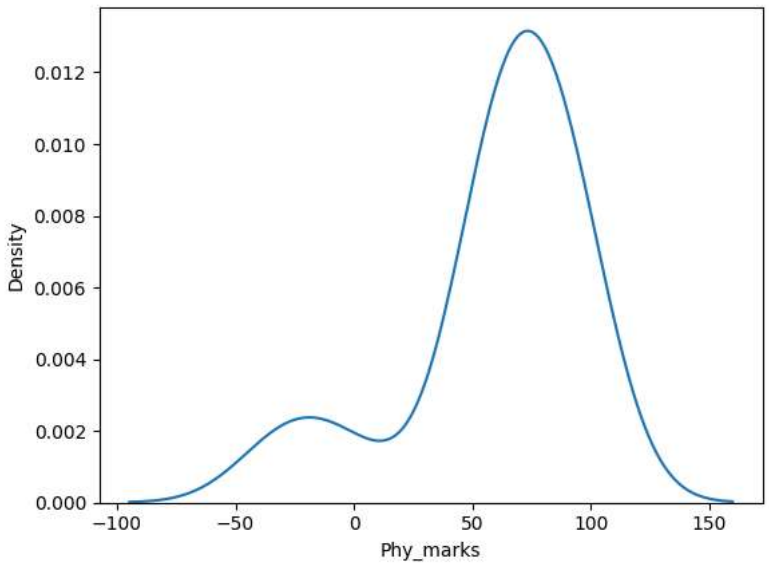
	Rollno	Name	Gender	Branch	Attendence	Phy_marks	Che_marks	EM1_marks	PPS_mark
2	3	Aarav	M	IT	57.0	-20.0	100.0	0.0	56.
8	9	Dhruv	M	ENTC	63.0	0.0	0.0	97.0	56.

Next steps: Generate code with `data2` View recommended plots

```
# he sns.kdeplot() function in Seaborn is used to create a kernel density estimate (KDE) plot, which visualizes the distribution of a univa
```

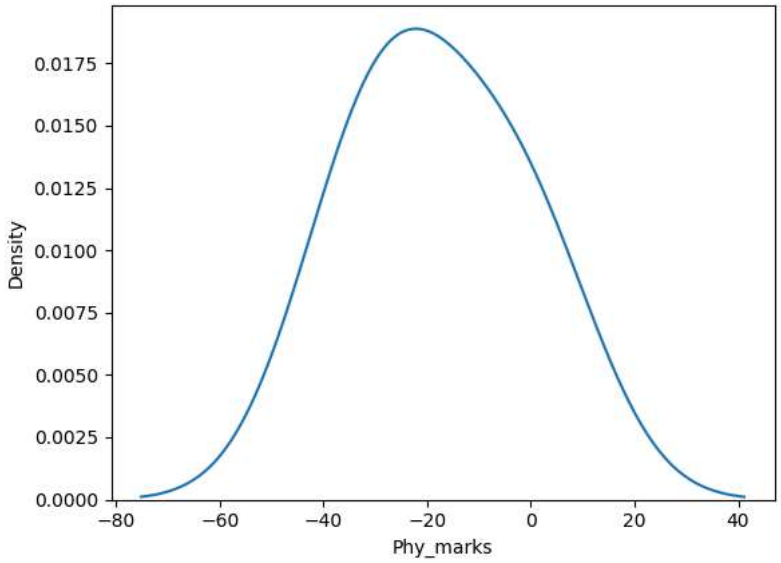
```
sns.kdeplot(data.Phy_marks)
```

```
<Axes: xlabel='Phy_marks', ylabel='Density'>
```




```
sns.kdeplot(data2.Phy_marks)
```

```
<Axes: xlabel='Phy_marks', ylabel='Density'>
```




```
log_phy_marks=np.log(data1.Phy_marks)
```

```
log_phy_marks
```

```
0     4.127134
1     4.127134
3     4.488636
4     4.499810
5     4.595120
6     4.477337
7     4.189655
9     4.204693
10    3.988984
11    4.158883
12    4.025352
14    4.465908
15    4.499810
16    3.988984
17    4.204693
18    4.477337
19    4.330733
Name: Phy_marks, dtype: float64
```

```
sns.kdeplot(log_phy_marks)
```

```
<Axes: xlabel='Phy_marks', ylabel='Density'>
```