ASSIGNMENT 2

TITLE: DATA WRANGLING II

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.
4. Reason and document your approach properly.

OBJECTIVE:
Students should be able to perform the data wrangling operation using Python on any open-source dataset.

PREREQUISITE: -
1. Basic of Python Programming
2. Concept of Data Preprocessing, Data Formatting, Data Normalization and Data Cleaning

THEORY:
1. Creation of Dataset using Microsoft Excel.

The dataset is created in “CSV” format.
- The name of dataset is Students Performance
- The features of the dataset are: Math_Score, Reading_Score, Writing_Score, Placement_Score, Club_Join_Date .
- Number of Instances: 30
- The response variable is: Placement_Offer_Count .
- Range of Values:
  Math_Score [60-80], Reading_Score[75-,95], ,Writing_Score [60,80], Placement_Score[75-100], Club_Join_Date [2018-2021].
- The response variable is the number of placement offers facilitated to particular students, which is largely depend on Placement_Score

To fill the values in the dataset the RANDBETWEEN is used. Returns a random integer number between the numbers you specify.

Syntax: RANDBETWEEN (bottom, top)

Bottom The smallest integer and Top The largest integer RANDBETWEEN will return.

2 Identification and Handling of Null Values

Missing Data can occur when no information is provided for one or more items or for a whole unit. Missing Data is a very big problem in real-life scenarios. Missing Data can also refer to as NA(Not Available) values in pandas. In DataFrame sometimes many datasets simply arrive with missing data, either because it exists and was not collected, or it never existed. For Example, Suppose different users being surveyed may choose not to share their income, some users may choose not to share the address in this way many datasets went missing.

In Pandas missing data is represented by two values:

1. **None**: None is a Python singleton object that is often used for missing data in Python code.
2. **NaN:** NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation.

Pandas treat None and NaN as essentially interchangeable for indicating missing or null values. To facilitate this convention, there are several useful functions for detecting, removing, and replacing null values in Pandas DataFrame :

- isnull()
- notnull()
- dropna()
- fillna()
- replace()

1. Checking for missing values using isnull() and notnull()

- **Checking for missing values using isnull()**

In order to check null values in Pandas DataFrame, isnull() function is used. This function returns dataframe of Boolean values which are True for NaN values.

Algorithm:

Step 1 : Import pandas and numpy in order to check missing values in PandasDataFrame

```
import pandas as pd

import numpy as np
```

Step 2: Load the dataset in dataframe object df

```
df=pd.read_csv("/content/StudentsPerformanceTest1.csv")
```

Step 3: Display the data frame

```
df
```

Step 4: Use isnull() function to check null values in the dataset.

```
df.isnull()
```

Step 5: To create a series true for NaN values for specific columns. for examplemath score in dataset and display data with only math score as NaN

```
series = pd.isnull(df["math score"])df[series]
```

- **Checking for missing values using notnull()**

In order to check null values in Pandas Dataframe, notnull() function is used. This function returns dataframe of Boolean values which are False for NaN values.

1. **Algorithm:**

Step 1 : Import pandas and numpy in order to check missing values in PandasDataFrame

```
import pandas as pd

import numpy as np
```

Step 2: Load the dataset in dataframe object df

```
df=pd.read_csv("/content/StudentsPerformanceTest1.csv")
```

Step 3: Display the data frame

```
df
```

Step 4: Use notnull() function to check null values in the dataset.

```
df.notnull()
```

Step 5: To create a series true for NaN values for specific columns. for examplemath score in dataset and display data with only math score as NaN

```
series1 = pd.notnull(df["math  score"])df[series1]
```

See that there are also categorical values in the dataset, for this, you need to useLabel Encoding

```
from sklearn.preprocessing import LabelEncoderle =
LabelEncoder()
df['gender'] = le.fit_transform(df['gender'])newdf=df
```

1. **Filling missing values using dropna(), fillna(), replace()**

In order to fill null values in a datasets, fillna(), replace() functions are used. These functions replace NaN values with some value of their own. All these functions help in filling null values in datasets of a DataFrame.

For replacing null values with NaN

```
missing_values = ["Na", "na"]
df  =  pd.read_csv("StudentsPerformanceTest1.csv",  na_values  =missing_values)

df
```

- **Filling null values with a single value**

Step 1 : Import pandas and numpy in order to check missing values in PandasDataFrame

```
import pandas as pd

import numpy as np
```

Step 2: Load the dataset in dataframe object df

```
df=pd.read_csv("/content/StudentsPerformanceTest1.csv")
```

Step 3: Display the data frame

```
df
```

Step 4: filling missing value using fillna()

```
ndf=df
ndf.fillna(0)
```

Step 5: filling missing values using mean, median and standard deviation of thatcolumn.

```
data['math score'] = data['math score'].fillna(data['math score'].mean())

data["math score"] = data["math score"].fillna(data["mathscore"].median())

data['math score"] = data["math score"].fillna(data["math score"].std())
```

replacing missing values in forenoon column with minimum/maximum numberof that column

```
data["math score"] = data["math score"].fillna(data["math score"].min())

data["math score"] = data["math score"].fillna(data["math score"].max())
```

- **Filling null values in dataset**

To fill null values in dataset use inplace=true

```
m_v=df['math score'].mean()
df['math score'].fillna(value=m_v, inplace=True)
df
```

- **Filling a null values using replace() method**

Following line will replace Nan value in dataframe with value -99

```
ndf.replace(to_replace = np.nan, value = -99)
```

- **Deleting null values using dropna() method**

In order to drop null values from a dataframe, dropna() function is used. Thisfunction 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

Algorithm:

Step 1 : Import pandas and numpy in order to check missing values in PandasDataFrame

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

Step 2: Load the dataset in dataframe object df

```
df=pd.read_csv("/content/StudentsPerformanceTest1.csv")
```

Step 3: Display the data frame

```
df
```

Step 4:To drop rows with at least 1 null value

```
ndf.dropna()
```

Step 5: To Drop rows if all values in that row are missing

```
ndf.dropna(how = 'all')
```

Step 6: To Drop columns with at least 1 null value.

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

Step 7 : To drop rows with at least 1 null value in CSV file.making new data frame with dropped NA values

```
new_data = ndf.dropna(axis = 0, how ='any')
new_data
```

3. Identification and Handling of Outliers

3.1 Identification of Outliers

One of the most important steps as part of data preprocessing is detecting and treating theoutliers as they can negatively affect the statistical analysis and the training process of a machine learning algorithm resulting in lower accuracy.

1. What are Outliers?

We all have heard of the idiom ‘odd one out' which means something unusual in comparison to the others in a group.

Similarly, 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.

2. Why do they occur?

An outlier may occur due to the variability in the data, or due to Assignment al error/human error. They may indicate an Assignment al error or heavy skewness in the data(heavy-tailed distribution).

3. What do they affect?

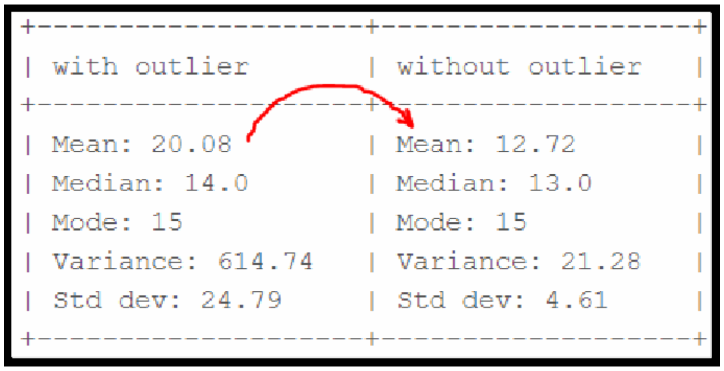
In statistics, we have three measures of central tendency namely Mean, Median, and Mode. They help us describe the data.

Mean is the accurate measure to describe the data when we do not have any outliers present. Median is used if there is an outlier in the dataset. Mode is used if thereis an outlier AND about ½ or more of the data is the same.

'Mean' is the only measure of central tendency that is affected by the outliers which in turn impacts Standard deviation.

Example:

Consider a small dataset, sample= [15, 101, 18, 7, 13, 16, 11, 21, 5, 15, 10, 9]. Bylooking at it, one can quickly say '101' is an outlier that is much larger than the other values.



```
+--------------------+-------------------+
| with outlier       | without outlier   |
+--------------------+-------------------+
| Mean: 20.08        | Mean: 12.72       |
| Median: 14.0       | Median: 13.0      |
| Mode: 15           | Mode: 15          |
| Variance: 614.74   | Variance: 21.28   |
| Std dev: 24.79     | Std dev: 4.61     |
+--------------------+-------------------+
```

Fig.1 Computation with and without outlier

From the above calculations, we can clearly say the Mean is more affected than the Median.

4. Detecting Outliers

If our dataset is small, we can detect the outlier by just looking at the dataset. But what if we have a huge dataset, how do we identify the outliers then? We need to use visualization and mathematical techniques.

Below are some of the techniques of detecting outliers

- Boxplots
- Scatterplots
- Z-score
- Inter Quantile Range(IQR)

Detecting outliers using Boxplot:

It captures the summary of the data effectively and efficiently with only a simple box and whiskers. Boxplot summarizes sample data using 25th, 50th, and 75th percentiles. One can just get insights(quartiles, median, and outliers) into the dataset by just looking at its boxplot.

2. Algorithm:

Step 1 : Import pandas and numpy libraries

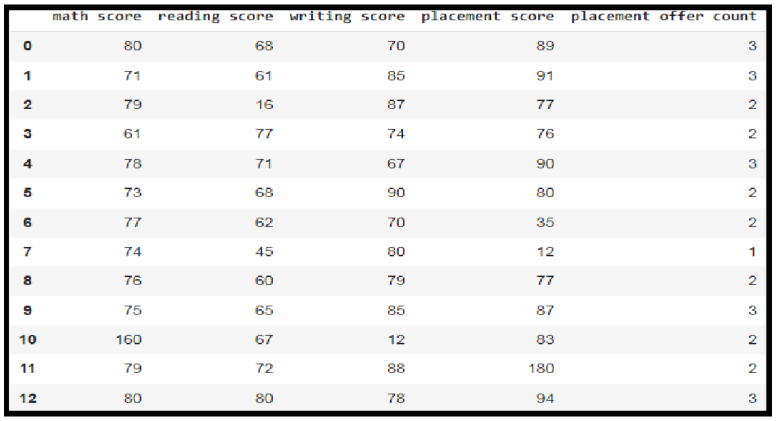
```
import pandas as pd
import numpy as np
```

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

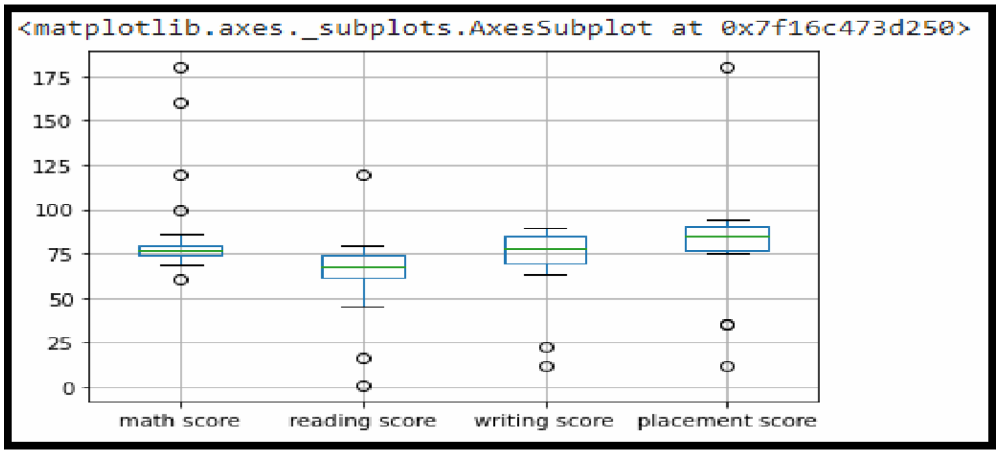
df



	math score	reading score	writing score	placement score	placement offer count
0	80	68	70	89	3
1	71	61	85	91	3
2	79	16	87	77	2
3	61	77	74	76	2
4	78	71	67	90	3
5	73	68	90	80	2
6	77	62	70	35	2
7	74	45	80	12	1
8	76	60	79	77	2
9	75	65	85	87	3
10	160	67	12	83	2
11	79	72	88	180	2
12	80	80	78	94	3

Step 4:Select the columns for boxplot and draw the boxplot.

```
col = ['math score', 'reading score' , 'writingscore','placement
score']
df.boxplot(col)
```



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f16c473d250>
```

Step 5: We can now print the outliers for each column with reference to the box plot.

```
print(np.where(df['math score']>90))
print(np.where(df['reading score']<25))
print(np.where(df['writing score']<30))
```

Handling of Outliers:

For removing the outlier, one must follow the same process of removing an entry from the dataset using its exact position in the dataset because in all the above methods ofdetecting the outliers end result is the list of all those data items that satisfy the outlier definition according to the method used.

Below are some of the methods of treating the outliers

- Trimming/removing the outlier
- Quantile based flooring and capping
- Mean/Median imputation

4. Data Transformation for the purpose of:

Data transformation is the process of converting raw data into a format or structure that would be more suitable for model building and also data discovery in general.The process of data transformation can also be referred to as extract/transform/load (ETL). The extraction phase involves identifying and pulling data from the various source systems that create data and then moving the data to a single repository. Next, the raw data is cleansed, if needed. It's then transformed into a target format that can be fed into operational systems or into a data warehouse, a date lake or another repository for use in business intelligence and analytics applications. The transformation The data are transformed in ways that are ideal for mining the data. The data transformation involves steps that are.

- **Smoothing:** It is a process that is used to remove noise from the dataset using some algorithms It allows for highlighting important features present in the dataset. It helps in predicting the patterns
- **Aggregation**: Data collection or aggregation is the method of storing and presenting data in a summary format. The data may be obtained from multiple data sources to integrate these data sources into a data analysis description. This is a crucial step since the accuracy of data analysis insights is highly dependent on the quantity and quality of the data used.
- **Generalization**:It converts low-level data attributes to high-level data attributes using concept hierarchy. For Example Age initially in Numerical form (22, 25) is converted into categorical value (young, old).
- **Normalization:** Data normalization involves converting all data variables into a given range. Some of the techniques that are used for accomplishing normalization are:

  Min–max normalization: This transforms the original data linearly.

  Z-score normalization: In z-score normalization (or zero-mean normalization) the values of an attribute (A), are normalized based on the mean of A and its standard deviation.

  Normalization by decimal scaling: It normalizes the values of an attribute by changing the position of their decimal points
- **Attribute or feature construction**.

  New attributes constructed from the given ones: Where new attributes are created & applied to assist the mining process from the given set of attributes. This simplifies the original data & makes the mining more efficient.

In this assignment , The purpose of this transformation should be one of the following reasons:

a. To change the scale for better understanding (Attribute or feature construction)
Here the Club_Join_Date is transferred to Duration.

Algorithm:

Step 1 : Import pandas and numpy libraries

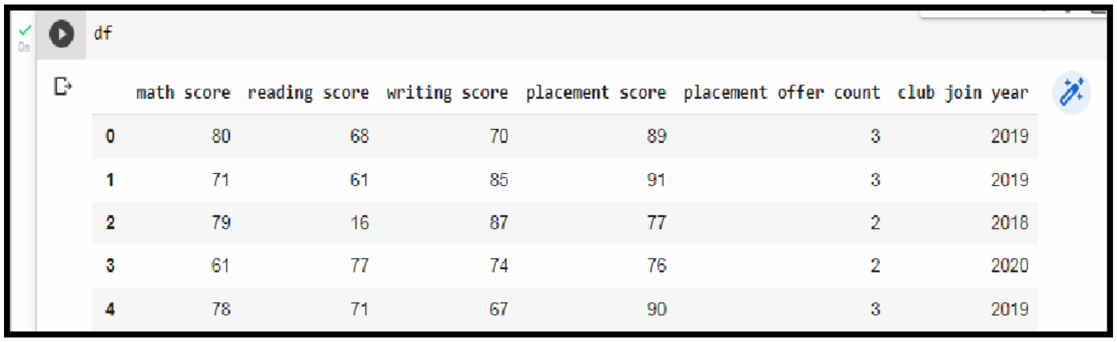
```
import pandas as pd

import numpy as np
```

Step 2: Load the dataset in dataframe object df

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

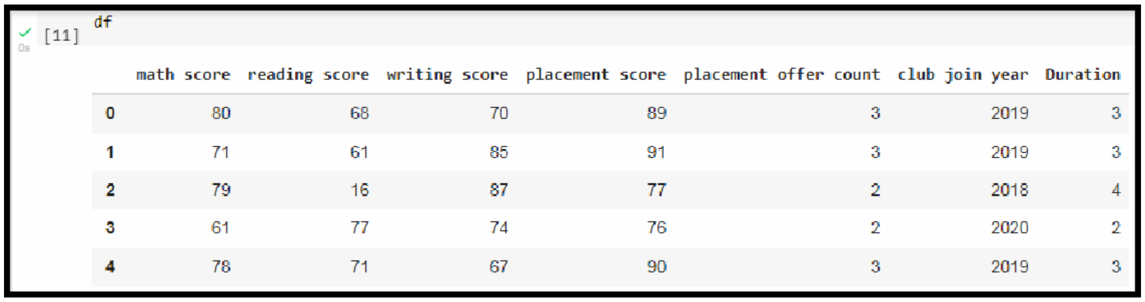
Step 3: Display the data frame

Step 3: Change the scale of Joining year to duration.



```
df
```

	math score	reading score	writing score	placement score	placement offer count	club join year
0	80	68	70	89	3	2019
1	71	61	85	91	3	2019
2	79	16	87	77	2	2018
3	61	77	74	76	2	2020
4	78	71	67	90	3	2019



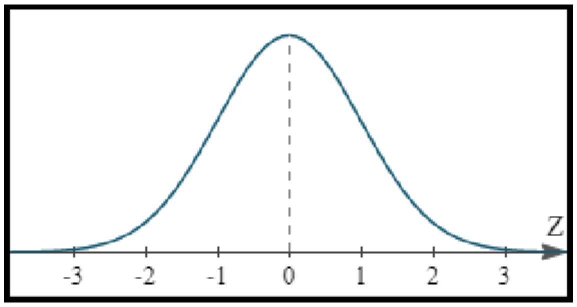
```
[11] df
```

	math score	reading score	writing score	placement score	placement offer count	club join year	Duration
0	80	68	70	89	3	2019	3
1	71	61	85	91	3	2019	3
2	79	16	87	77	2	2018	4
3	61	77	74	76	2	2020	2
4	78	71	67	90	3	2019	3

b. To decrease the skewness and convert distribution into normal distribution (Normalization by decimal scaling)

Data Skewness: It is asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right. Skewness can be quantified to define the extent to which a distribution differs from a normal distribution.

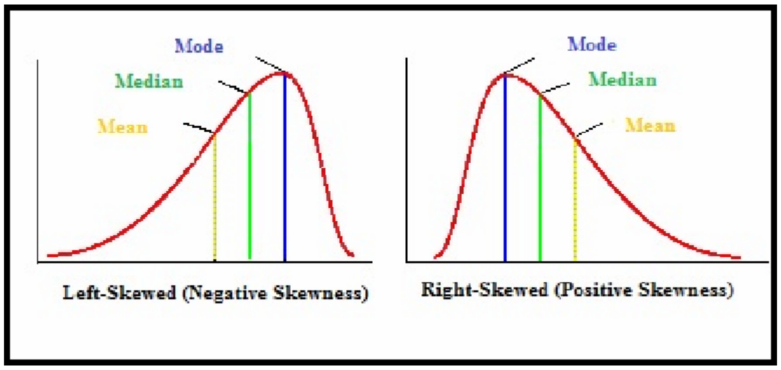
Normal Distribution: In a normal distribution, the graph appears as a classical, symmetrical "bell-shaped curve." The mean, or average, and the mode, or maximum point on the curve, are equal.



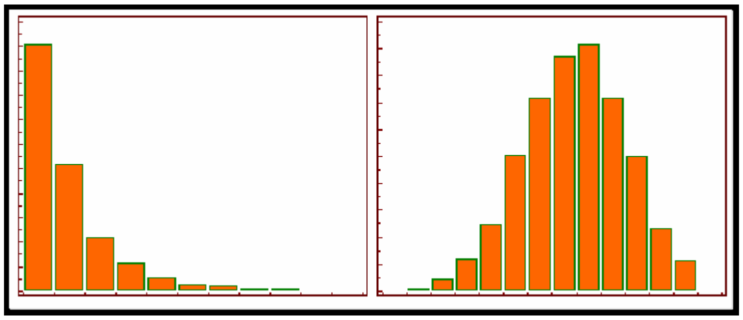
Positively Skewed Distribution

A positively skewed distribution means that the extreme data results are larger. This skews the data in that it brings the mean (average) up. The mean will be larger than the median in a Positively skewed distribution.

A negatively skewed distribution means the opposite: that the extreme data results are smaller. This means that the mean is brought down, and the median is larger than the mean in a negatively skewed distribution.



Reducing skewness A data transformation may be used to reduce skewness. A distribution that is symmetric or nearly so is often easier to handle and interpret than a skewed distribution. The logarithm, x to log base 10 of x, or x to log base e of x (ln x), or x to log base 2 of x, is a strong transformation with a major effect on distribution shape. It is commonly used for reducing right skewness and is often appropriate for measured variables. It can not be applied to zero or negative values.



1. **Algorithm:**

   Step 1 : Detecting outliers using Z-Score for the Math_score variable andremove the outliers.

Step 2: Observe the histogram for math_score variable.

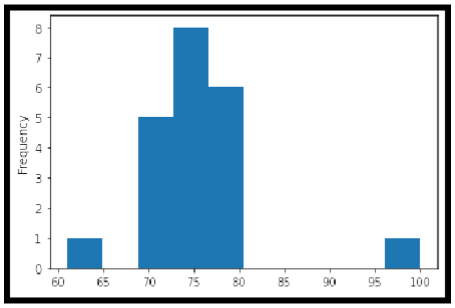
```
import matplotlib.pyplot as plt new_df['math
score'].plot(kind = 'hist')
```

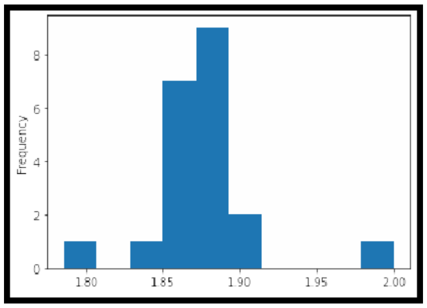
Step 3: Convert the variables to logarithm at the scale 10.

```
df['log_math'] = np.log10(df['math score'])
```

Step 4: Observe the histogram for math_score variable.

```
df['log_math'].plot(kind = 'hist')
```





It is observed that skewness is reduced at some level.

Conclusion: In this way we have explored the functions of the python library for Data Identifying and handling the outliers. Data Transformations Techniques are explored with the purpose of creating the new variable and reducing the skewness from datasets.

Assignment Question:

1. Explain the methods to detect the outlier.
2. Explain data transformation methods.
3. Write the algorithm to display the statistics of Null values present in the dataset.
4. Write an algorithm to replace the outlier value with the mean of the variable.

.

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

```
np.random.seed(56)
```

Generating Data

```
n=100
data = {
    'Student_ID': np.arange(1, n+1),
    'Math_Score': np.random.normal(70, 10, n).astype(int),
    'Science_Score': np.random.normal(75, 12, n).astype(int),
    'English_Score': np.random.normal(65, 8, n).astype(int),
    'Attendance': np.random.uniform(70, 100, n).astype(int)
}
```

```
df = pd.DataFrame(data)
df
```

	Student_ID	Math_Score	Science_Score	English_Score	Attendance
0	1	59	90	62	81
1	2	75	88	64	76
2	3	81	71	55	75
3	4	64	86	64	79
4	5	67	86	72	98
...	...	...	...	...	...
95	96	82	105	62	71
96	97	71	65	62	75
97	98	70	67	65	73
98	99	68	46	70	95
99	100	73	81	66	71

Next steps: Generate code with df View recommended plots New interactive sheet

Introduce missing values

```
missing_indices = np.random.choice(df.index, size=5, replace=False)
df.loc[missing_indices, 'Math_Score'] = np.nan

df.loc[np.random.choice(df.index, size=5, replace=False), 'Science_Score'] = np.nan

df.loc[np.random.choice(df.index, size=3, replace=False), 'English_Score'] = np.nan
```

Introducting Outliers

```
df.loc[np.random.choice(df.index, size=3, replace=False), 'Math_Score'] = [650, -53, 1600]
df.loc[np.random.choice(df.index,size=3,replace=False),"Science_Score"]=[1600,-452,3140]
df.loc[np.random.choice(df.index,size=3,replace=False),"English_Score"]=[-130,777,144]
```

```
df
```

	Student_ID	Math_Score	Science_Score	English_Score	Attendance
0	1	59.0	90.0	62.0	81
1	2	75.0	88.0	64.0	76
2	3	81.0	71.0	55.0	75
3	4	64.0	86.0	64.0	79
4	5	67.0	86.0	72.0	98
...	...	...	...	...	...
95	96	82.0	105.0	62.0	71
96	97	71.0	65.0	62.0	75
97	98	70.0	67.0	65.0	73
98	99	68.0	46.0	70.0	95
99	100	73.0	81.0	66.0	71

Next steps: Generate code with df View recommended plots New interactive sheet

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

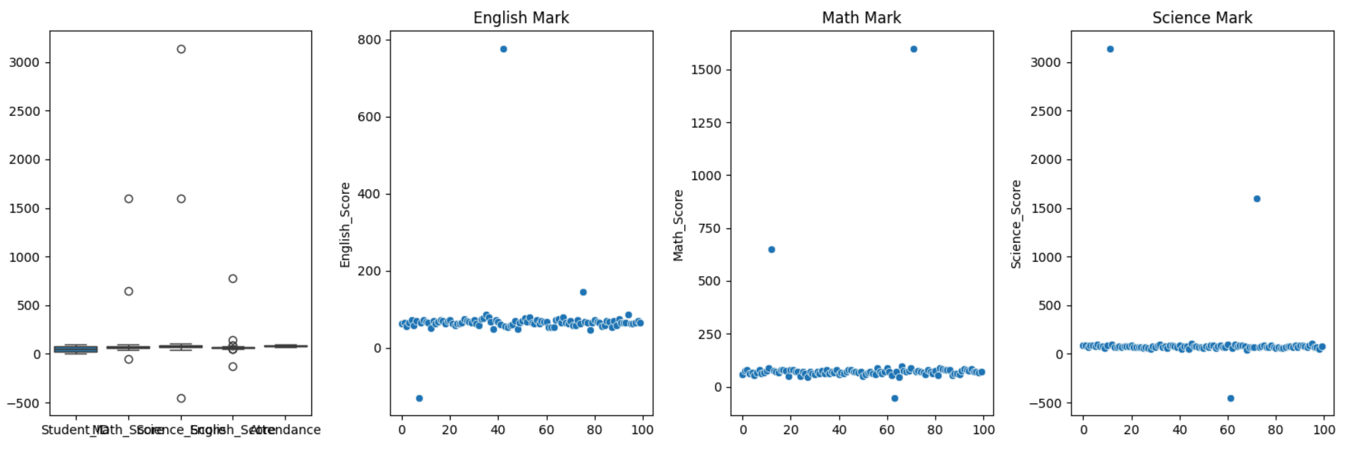
	0
Student_ID	0
Math_Score	5
Science_Score	5
English_Score	3
Attendance	0

```
df["Math_Score"]=df['Math_Score'].fillna(df['Math_Score'].median())
df["Science_Score"]=df["Science_Score"].fillna(df["Science_Score"].median())
df["English_Score"]=df["English_Score"].fillna(df["English_Score"].median())
```

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

	0
Student_ID	0
Math_Score	0
Science_Score	0
English_Score	0
Attendance	0

```
fig,ax=plt.subplots(1,4,figsize=(15,5))
sns.boxplot(df,ax=ax[0])
sns.scatterplot(df['English_Score'],  ax=ax[1]).set_title('English Mark')
sns.scatterplot(df['Math_Score'],  ax=ax[2]).set_title('Math Mark')
sns.scatterplot(df['Science_Score'],  ax=ax[3]).set_title('Science Mark')
plt.tight_layout()
plt.show()
```



Removing Outliers

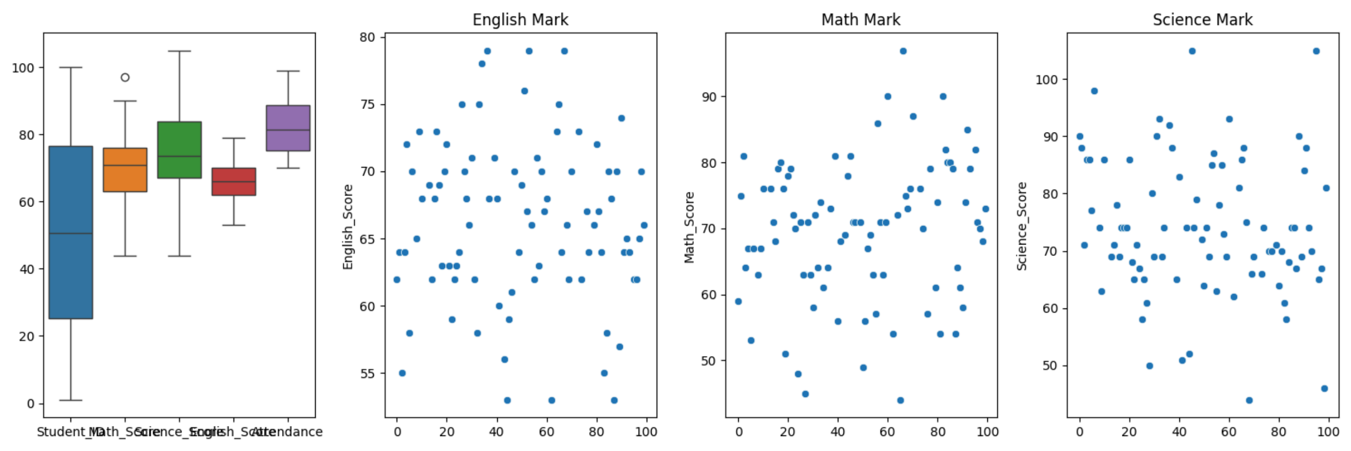
```
df.describe()
```

	Student_ID	Math_Score	Science_Score	English_Score	Attendance
count	100.000000	100.000000	100.000000	100.000000	100.000000
mean	50.500000	89.840000	115.310000	71.460000	82.910000
std	29.011492	164.018336	345.955147	74.721222	8.366715
min	1.000000	-53.000000	-452.000000	-130.000000	70.000000
25%	25.750000	63.750000	67.000000	62.000000	75.750000
50%	50.500000	71.000000	74.000000	66.000000	82.500000
75%	75.250000	78.000000	85.000000	70.000000	89.000000

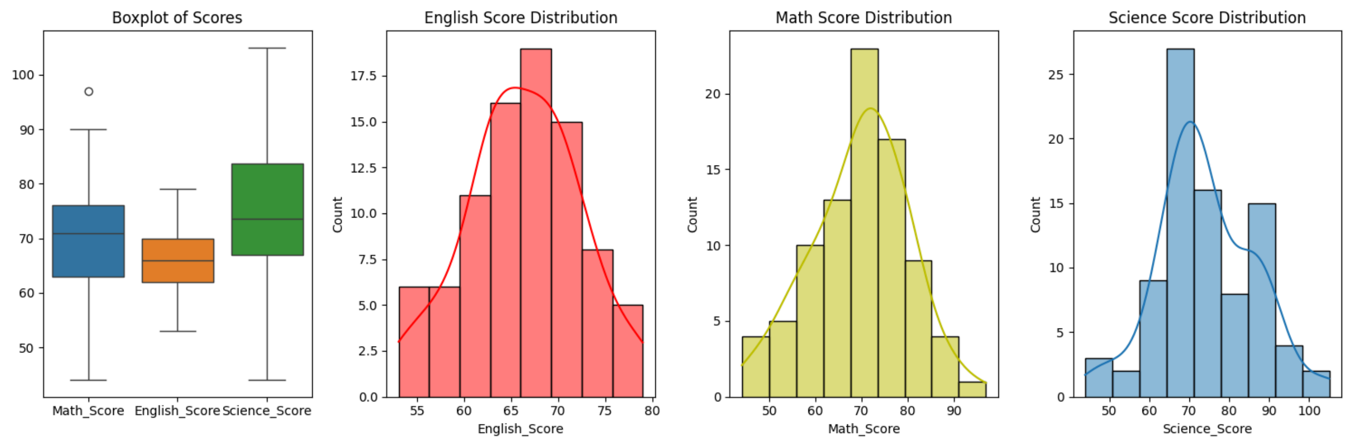
```
cols = ['Math_Score', 'English_Score', 'Science_Score']

for col in cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    print(f"{col} - Lower Bound: {lower_bound}, Upper Bound: {upper_bound}")


    df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]


print(df.shape)
```

```
Math_Score - Lower Bound: 42.375, Upper Bound: 99.375
English_Score - Lower Bound: 50.0, Upper Bound: 82.0
Science_Score - Lower Bound: 40.0, Upper Bound: 112.0
(86, 5)
```

```
fig,ax=plt.subplots(1,4,figsize=(15,5))
sns.boxplot(df,ax=ax[0])
sns.scatterplot(df['English_Score'],ax=ax[1]).set_title('English Mark')
sns.scatterplot(df['Math_Score'],ax=ax[2]).set_title('Math Mark')
sns.scatterplot(df['Science_Score'],ax=ax[3]).set_title('Science Mark')
plt.tight_layout()
plt.show()
```



```
fig, ax = plt.subplots(1, 4, figsize=(15, 5))

sns.boxplot(data=df[['Math_Score', 'English_Score', 'Science_Score']], ax=ax[0])
ax[0].set_title('Boxplot of Scores')

sns.histplot(df['English_Score'], kde=True, ax=ax[1],color='r')
ax[1].set_title('English Score Distribution')

sns.histplot(df['Math_Score'], kde=True, ax=ax[2],color='y')
ax[2].set_title('Math Score Distribution')

sns.histplot(df['Science_Score'], kde=True, ax=ax[3])
ax[3].set_title('Science Score Distribution')

plt.tight_layout()
plt.show()
```



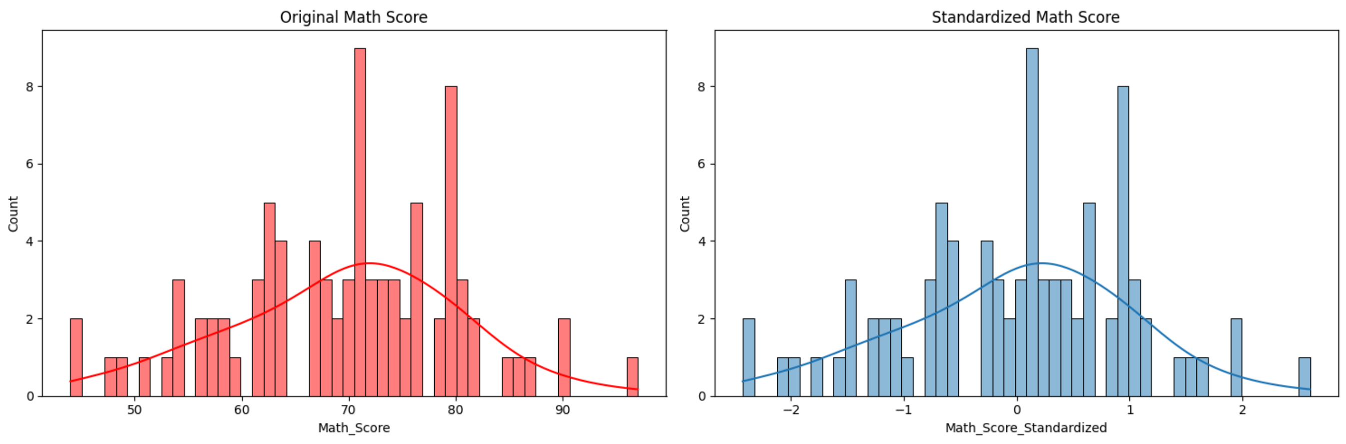
```
print(df[['Math_Score', 'English_Score', 'Science_Score']].skew())
```

```
Math_Score      -0.207735
English_Score   -0.085036
Science_Score    0.141484
dtype: float64
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df[['Math_Score_Standardized', 'English_Score_Standardized', 'Science_Score_Standardized']] = scaler.fit_transform(df[['Math_Score', 'Eng
```

```
fig, axes = plt.subplots(1, 2, figsize=(15, 5))


sns.histplot(df['Math_Score'], bins=50, kde=True, ax=axes[0],color='red').set_title('Original Math Score')
sns.histplot(df['Math_Score_Standardized'], bins=50, kde=True, ax=axes[1]).set_title('Standardized Math Score')
```

```
plt.tight_layout()

plt.show()
```


Original Math Score
Count
Math_Score
Standardized Math Score
Count
Math_Score_Standardized