Module 2:Detection of Outlier

1.Outlier visualization- Boxplot, Scatterplot 2.Techniques of Detecting outlier a. Z-score b.IQR

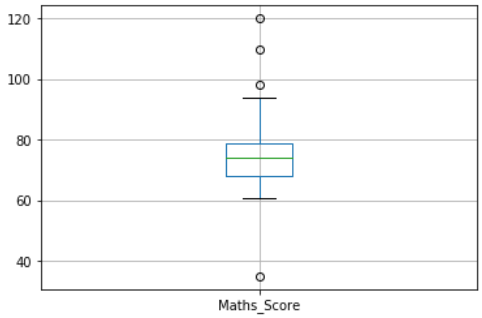
Outlier visualization- Boxplot,

In [124]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [125]:
```
df=pd.read_csv("C:\\Users\\Admin\\Desktop\\StudentPerformance.csv")
```

In [126]:
```
new_df = df

col = ['Maths_Score']
#******Identifying Outliers with Visualization
new_df.boxplot(col) #  outliers are seen in boxplot
```

Out[126]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xb9e2f10>
```



Detecting outlier by using IQR

InterQuantile Range 75%- 25% values in a dataset

Steps

1. Arrange the data in increasing order
2. Calculate first(q1) and third quartile(q3)
3. Find interquartile range (q3-q1) 4.Find lower bound q1*1.5 5.Find upper bound q*31.5 Percentiles are used in statistics to give you a number that describes the value that a given percent of the values are lower than. numpy.percentile(arr, n)

In [127]:
```
#Identifying Outliers with Interquartile Range (IQR) Calculate and print Quartile 1 and Quartile

q1 = np.percentile(df['Maths_Score'],25)
q3 = np.percentile(df['Maths_Score'],75)
print(q1,q3)
```

```
68.0 79.0
```

In [128]:
```
#Calculate value of IQR (Inter Quartile Range)
IQR = q3-q1

#Calculate and print Upper and Lower Bound to define the outlier base value.
lwr_bound = q1-(1.5*IQR)
upr_bound = q3+(1.5*IQR)
print(lwr_bound, upr_bound)
```

```
51.5 95.5
```

In [129]:
```
index_outliers = np.where((df[col] < lwr_bound) | ( df[col] > upr_bound))
index_outliers
```

Out[129]:
```
(array([ 3, 15, 19, 27], dtype=int32), array([0, 0, 0, 0], dtype=int32))
```

In [130]:
```
df
```

Out[130]:

	Maths_Score	Reading_Score	Writing_Score	Placement_Score	Club_Join_Date	Placement offer count
0	70	93	61	82	2020	2
1	77	84	65	88	2019	3
2	69	84	68	93	2020	3
3	35	81	73	91	2019	3
4	78	95	73	96	2020	3
5	70	94	60	80	2020	2
6	69	86	79	91	2018	3
7	76	92	61	79	2019	2
8	79	81	77	80	2018	2
9	79	85	78	76	2020	2
10	66	78	69	94	2018	3
11	75	60	60	90	2018	3
12	75	81	74	88	2019	3
13	94	75	80	83	2020	2
14	69	79	79	80	2020	2
15	110	88	61	81	2018	2
16	79	84	75	76	2018	2
17	68	80	66	89	2020	3
18	65	85	68	92	2020	3
19	120	75	75	84	2018	2
20	71	78	67	83	2018	2
21	67	89	95	78	2019	2
22	74	77	72	81	2020	2
23	64	76	67	82	2018	2
24	61	87	63	98	2020	3
25	76	91	60	88	2019	3
26	93	93	76	90	2019	3
27	98	88	99	91	2019	3
28	62	79	67	86	2020	3

In [131]:
```
sample_outliers= df[col][(df[col] < lwr_bound) | (df[col] > upr_bound)]
sample_outliers
```

Out[131]:

	Maths_Score
0	NaN
1	NaN
2	NaN
3	35.0
4	NaN
5	NaN
6	NaN
7	NaN
8	NaN
9	NaN
10	NaN
11	NaN
12	NaN
13	NaN
14	NaN
15	110.0
16	NaN
17	NaN
18	NaN
19	120.0
20	NaN
21	NaN
22	NaN
23	NaN
24	NaN
25	NaN
26	NaN
27	98.0
28	NaN

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 of detecting 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

Quantile based flooring and capping

the outlier is capped at a certain value above the 90th percentile value or floored at a factor below the 10th percentile value

In [132]:
```
df1=df
df[col] = np.where(df1[col]< lwr_bound,lwr_bound,df[col])
df[col] = np.where(df1[col]>upr_bound ,upr_bound,df[col])
```

In [133]:
```
df1
```

Out[133]:

	Maths_Score	Reading_Score	Writing_Score	Placement_Score	Club_Join_Date	Placement offer count
0	70.0	93	61	82	2020	2
1	77.0	84	65	88	2019	3
2	69.0	84	68	93	2020	3
3	51.5	81	73	91	2019	3
4	78.0	95	73	96	2020	3
5	70.0	94	60	80	2020	2
6	69.0	86	79	91	2018	3
7	76.0	92	61	79	2019	2
8	79.0	81	77	80	2018	2
9	79.0	85	78	76	2020	2
10	66.0	78	69	94	2018	3
11	75.0	60	60	90	2018	3
12	75.0	81	74	88	2019	3
13	94.0	75	80	83	2020	2
14	69.0	79	79	80	2020	2
15	95.5	88	61	81	2018	2
16	79.0	84	75	76	2018	2
17	68.0	80	66	89	2020	3
18	65.0	85	68	92	2020	3
19	95.5	75	75	84	2018	2
20	71.0	78	67	83	2018	2
21	67.0	89	95	78	2019	2
22	74.0	77	72	81	2020	2
23	64.0	76	67	82	2018	2
24	61.0	87	63	98	2020	3
25	76.0	91	60	88	2019	3
26	93.0	93	76	90	2019	3
27	95.5	88	99	91	2019	3
28	62.0	79	67	86	2020	3

In [146]:
```
df5=pd.read_csv("C:\\Users\\Admin\\Desktop\\StudentPerformance.csv")
df5
```

Out[146]:

	Maths_Score	Reading_Score	Writing_Score	Placement_Score	Club_Join_Date	Placement offer count
0	70	93	61	82	2020	2
1	77	84	65	88	2019	3
2	69	84	68	93	2020	3
3	35	81	73	91	2019	3
4	78	95	73	96	2020	3
5	70	94	60	80	2020	2
6	69	86	79	91	2018	3
7	76	92	61	79	2019	2
8	79	81	77	80	2018	2
9	79	85	78	76	2020	2
10	66	78	69	94	2018	3
11	75	60	60	90	2018	3
12	75	81	74	88	2019	3
13	94	75	80	83	2020	2
14	69	79	79	80	2020	2
15	110	88	61	81	2018	2
16	79	84	75	76	2018	2
17	68	80	66	89	2020	3
18	65	85	68	92	2020	3
19	120	75	75	84	2018	2
20	71	78	67	83	2018	2
21	67	89	95	78	2019	2
22	74	77	72	81	2020	2
23	64	76	67	82	2018	2
24	61	87	63	98	2020	3
25	76	91	60	88	2019	3
26	93	93	76	90	2019	3
27	98	88	99	91	2019	3
28	62	79	67	86	2020	3

In [147]:
```
ninetieth_percentile = np.percentile(df5['Maths_Score'], 90)
ninetieth_percentile
```

Out[147]:
```
94.80000000000001
```

In [148]:
```
tenth_percentile = np.percentile(df5['Maths_Score'], 10)
tenth_percentile
```

Out[148]:
```
63.6
```

In [150]:
```
df5[col] = np.where(df5[col]>upr_bound ,ninetieth_percentile,df5[col])
df5[col] = np.where(df5[col]<lwr_bound ,tenth_percentile,df5[col])
df5
```

Out[150]:

	Maths_Score	Reading_Score	Writing_Score	Placement_Score	Club_Join_Date	Placement offer count
0	70.0	93	61	82	2020	2
1	77.0	84	65	88	2019	3
2	69.0	84	68	93	2020	3
3	63.6	81	73	91	2019	3
4	78.0	95	73	96	2020	3
5	70.0	94	60	80	2020	2
6	69.0	86	79	91	2018	3
7	76.0	92	61	79	2019	2
8	79.0	81	77	80	2018	2
9	79.0	85	78	76	2020	2
10	66.0	78	69	94	2018	3
11	75.0	60	60	90	2018	3
12	75.0	81	74	88	2019	3
13	94.0	75	80	83	2020	2
14	69.0	79	79	80	2020	2
15	94.8	88	61	81	2018	2
16	79.0	84	75	76	2018	2
17	68.0	80	66	89	2020	3
18	65.0	85	68	92	2020	3
19	94.8	75	75	84	2018	2
20	71.0	78	67	83	2018	2
21	67.0	89	95	78	2019	2
22	74.0	77	72	81	2020	2
23	64.0	76	67	82	2018	2
24	61.0	87	63	98	2020	3
25	76.0	91	60	88	2019	3
26	93.0	93	76	90	2019	3
27	94.8	88	99	91	2019	3
28	62.0	79	67	86	2020	3

Mean/Median imputation: As the mean value is highly influenced by the outliers, it is advised to replace the outliers with the median value

In [137]:
```
#Calculate the median of reading score by using sorted_rscore
median = np.median(new_df[col])
median
```

Out[137]: 74.0

In [138]:
```
#Replace the lower bound and upper bound outliers using median value
for i in index_outliers:
   new_df.at[i,col] = median

new_df
```

Out[138]:

	Maths_Score	Reading_Score	Writing_Score	Placement_Score	Club_Join_Date	Placement offer count
0	74.0	93	61	82	2020	2
1	77.0	84	65	88	2019	3
2	69.0	84	68	93	2020	3
3	74.0	81	73	91	2019	3
4	78.0	95	73	96	2020	3
5	70.0	94	60	80	2020	2
6	69.0	86	79	91	2018	3
7	76.0	92	61	79	2019	2
8	79.0	81	77	80	2018	2
9	79.0	85	78	76	2020	2
10	66.0	78	69	94	2018	3
11	75.0	60	60	90	2018	3
12	75.0	81	74	88	2019	3
13	94.0	75	80	83	2020	2
14	69.0	79	79	80	2020	2
15	74.0	88	61	81	2018	2
16	79.0	84	75	76	2018	2
17	68.0	80	66	89	2020	3
18	65.0	85	68	92	2020	3
19	74.0	75	75	84	2018	2
20	71.0	78	67	83	2018	2
21	67.0	89	95	78	2019	2
22	74.0	77	72	81	2020	2
23	64.0	76	67	82	2018	2
24	61.0	87	63	98	2020	3
25	76.0	91	60	88	2019	3
26	93.0	93	76	90	2019	3
27	74.0	88	99	91	2019	3
28	62.0	79	67	86	2020	3

Z-Score Z-Score is also called a standard score. This value/score helps to understand how far is the data point from the mean. And after setting up athreshold value one can utilize z score values of data points to define the outliers. Zscore = (data_point -mean) / std. deviation

In [157]:
```
from scipy import stats
df3=pd.read_csv("C:\\Users\\Admin\\Desktop\\StudentPerformance.csv")
df3
```

Out[157]:

	Maths_Score	Reading_Score	Writing_Score	Placement_Score	Club_Join_Date	Placement offer count
0	70	93	61	82	2020	2
1	77	84	65	88	2019	3
2	69	84	68	93	2020	3
3	35	81	73	91	2019	3
4	78	95	73	96	2020	3
5	70	94	60	80	2020	2
6	69	86	79	91	2018	3
7	76	92	61	79	2019	2
8	79	81	77	80	2018	2
9	79	85	78	76	2020	2
10	66	78	69	94	2018	3
11	75	60	60	90	2018	3
12	75	81	74	88	2019	3
13	94	75	80	83	2020	2
14	69	79	79	80	2020	2
15	110	88	61	81	2018	2
16	79	84	75	76	2018	2
17	68	80	66	89	2020	3
18	65	85	68	92	2020	3
19	120	75	75	84	2018	2
20	71	78	67	83	2018	2
21	67	89	95	78	2019	2
22	74	77	72	81	2020	2
23	64	76	67	82	2018	2
24	61	87	63	98	2020	3
25	76	91	60	88	2019	3
26	93	93	76	90	2019	3
27	98	88	99	91	2019	3
28	62	79	67	86	2020	3

In [158]:
```
z = np.abs(stats.zscore(df3['Maths_Score']))
print(z)
```

```
[0.35101647 0.09713663 0.41503834 2.59178199 0.16115851 0.35101647
 0.41503834 0.03311476 0.22518038 0.22518038 0.60710396 0.03090711
 0.03090711 1.18550846 0.41503834 2.20985841 0.22518038 0.47906021
 0.67112583 2.85007713 0.2869946  0.54308209 0.09492898 0.7351477
 0.92721332 0.03311476 1.12148658 1.44159594 0.86319145]
```

In [159]:
```
Upperthreshold = 1.4
Lowerthreshold = 0.18

index_outliers = np.where((z < Lowerthreshold) | ( z > Upperthreshold))

index_outliers
```

Out[159]:
```
(array([ 1,  3,  4,  7, 11, 12, 15, 19, 22, 25, 27], dtype=int32),)
```

In [160]:
```
sample_outliers= z[(z < Lowerthreshold) | (z > Upperthreshold)]
sample_outliers
```

Out[160]:
```
array([0.09713663, 2.59178199, 0.16115851, 0.03311476, 0.03090711,
       0.03090711, 2.20985841, 2.85007713, 0.09492898, 0.03311476,
       1.44159594])
```

In [161]:
```
##Trimming/removing the outlier:  In this technique, we remove the outliers from the dataset. Although it is not a good practice to follow.


new_df2=df3
for i in index_outliers:
  new_df2.drop(i,inplace=True)
new_df2
```

Out[161]:

	Maths_Score	Reading_Score	Writing_Score	Placement_Score	Club_Join_Date	Placement offer count
0	70	93	61	82	2020	2
2	69	84	68	93	2020	3
5	70	94	60	80	2020	2
6	69	86	79	91	2018	3
8	79	81	77	80	2018	2
9	79	85	78	76	2020	2
10	66	78	69	94	2018	3
13	94	75	80	83	2020	2
14	69	79	79	80	2020	2
16	79	84	75	76	2018	2
17	68	80	66	89	2020	3
18	65	85	68	92	2020	3
20	71	78	67	83	2018	2
21	67	89	95	78	2019	2
23	64	76	67	82	2018	2
24	61	87	63	98	2020	3
26	93	93	76	90	2019	3
28	62	79	67	86	2020	3

Module 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.

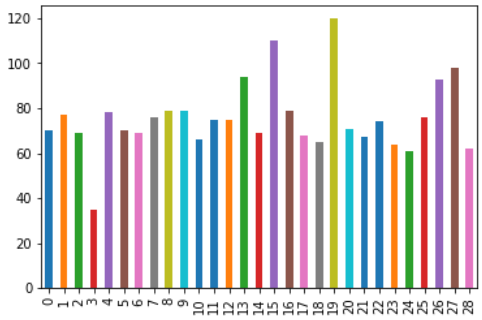
In [162]:
```
df4=pd.read_csv("C:\\Users\\Admin\\Desktop\\StudentPerformance.csv")
df4
```

Out[162]:

	Maths_Score	Reading_Score	Writing_Score	Placement_Score	Club_Join_Date	Placement offer count
0	70	93	61	82	2020	2
1	77	84	65	88	2019	3
2	69	84	68	93	2020	3
3	35	81	73	91	2019	3
4	78	95	73	96	2020	3
5	70	94	60	80	2020	2
6	69	86	79	91	2018	3
7	76	92	61	79	2019	2
8	79	81	77	80	2018	2
9	79	85	78	76	2020	2
10	66	78	69	94	2018	3
11	75	60	60	90	2018	3
12	75	81	74	88	2019	3
13	94	75	80	83	2020	2
14	69	79	79	80	2020	2
15	110	88	61	81	2018	2
16	79	84	75	76	2018	2
17	68	80	66	89	2020	3
18	65	85	68	92	2020	3
19	120	75	75	84	2018	2
20	71	78	67	83	2018	2
21	67	89	95	78	2019	2
22	74	77	72	81	2020	2
23	64	76	67	82	2018	2
24	61	87	63	98	2020	3
25	76	91	60	88	2019	3
26	93	93	76	90	2019	3
27	98	88	99	91	2019	3
28	62	79	67	86	2020	3

In [176]:
```
import matplotlib.pyplot as plt

df4['Maths_Score'].plot(kind = 'bar')
```

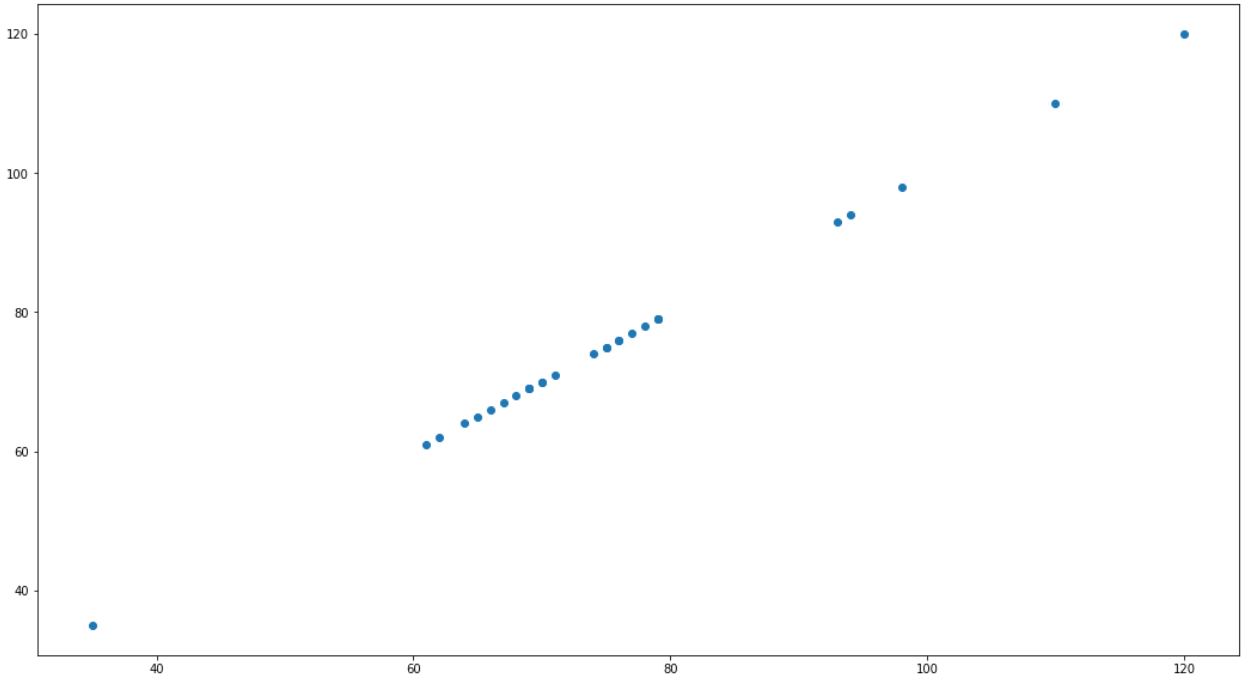
Out[176]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xd169d30>
```



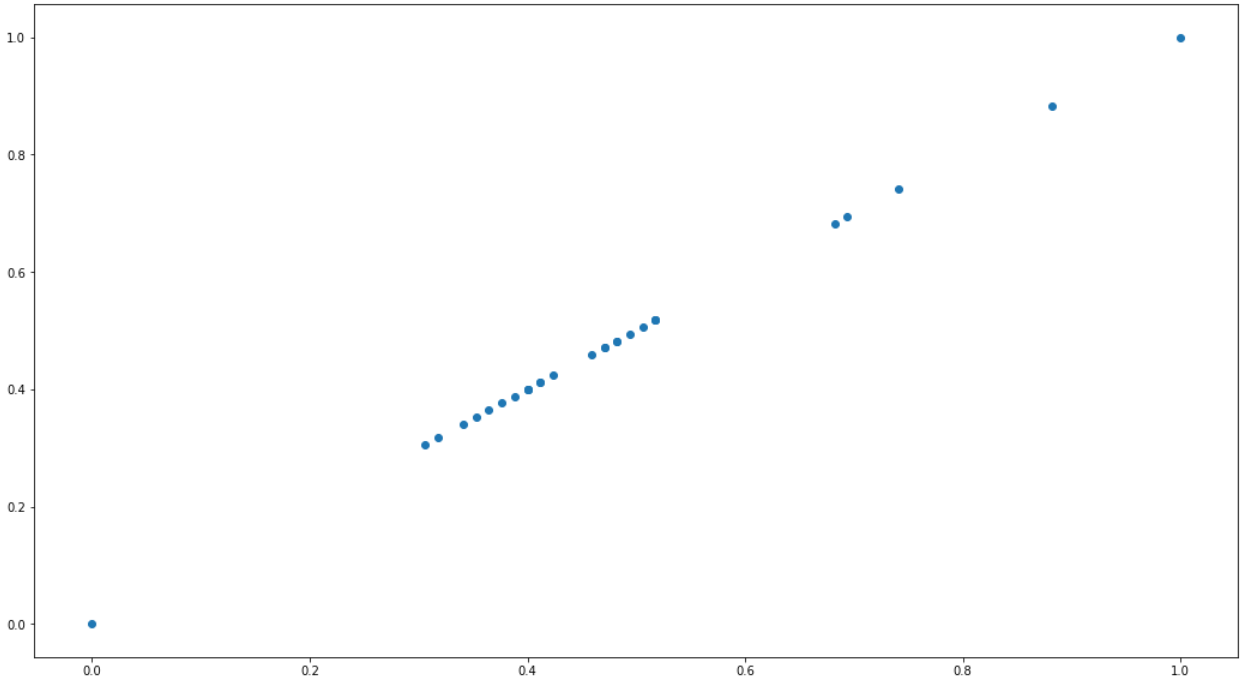
In [173]:
```
df_min_max_scaled = df4.copy()
colu=['Maths_Score']

# apply normalization techniques
df_min_max_scaled[colu] = (df_min_max_scaled[colu] - df_min_max_scaled[colu].min()) / (df_min_max_scaled[colu].max() - df_min
_max_scaled[colu].min())

# view normalized data
print(df_min_max_scaled)
```

```
    Maths_Score  Reading_Score  Writing_Score  Placement_Score  \
0      0.411765             93             61               82
1      0.494118             84             65               88
2      0.400000             84             68               93
3      0.000000             81             73               91
4      0.505882             95             73               96
5      0.411765             94             60               80
6      0.400000             86             79               91
7      0.482353             92             61               79
8      0.517647             81             77               80
9      0.517647             85             78               76
10     0.364706             78             69               94
11     0.470588             60             60               90
12     0.470588             81             74               88
13     0.694118             75             80               83
14     0.400000             79             79               80
15     0.882353             88             61               81
16     0.517647             84             75               76
17     0.388235             80             66               89
18     0.352941             85             68               92
19     1.000000             75             75               84
20     0.423529             78             67               83
21     0.376471             89             95               78
22     0.458824             77             72               81
23     0.341176             76             67               82
24     0.305882             87             63               98
25     0.482353             91             60               88
26     0.682353             93             76               90
27     0.741176             88             99               91
28     0.317647             79             67               86

    Club_Join_Date  Placement offer count
0             2020                      2
1             2019                      3
2             2020                      3
3             2019                      3
4             2020                      3
5             2020                      2
6             2018                      3
7             2019                      2
8             2018                      2
9             2020                      2
10            2018                      3
11            2018                      3
12            2019                      3
13            2020                      2
14            2020                      2
15            2018                      2
16            2018                      2
17            2020                      3
18            2020                      3
19            2018                      2
20            2018                      2
21            2019                      2
22            2020                      2
23            2018                      2
24            2020                      3
25            2019                      3
26            2019                      3
27            2019                      3
28            2020                      3
```

In [180]:



In [181]:
```
#fig,ax=plt.subplots(figsize=(18,10))
##plt.show()
```



In [182]:
```
#Skewness
df4.skew(axis = 1, skipna = True)
```

Out[182]:
```
0     2.440742
1     2.440979
2     2.440759
3     2.440198
4     2.439189
5     2.440771
6     2.440408
7     2.440791
8     2.441093
9     2.440993
10    2.441065
11    2.441954
12    2.441064
13    2.439931
14    2.441540
15    2.437675
16    2.441193
17    2.441507
18    2.440855
19    2.436555
20    2.441830
21    2.439648
22    2.441761
23    2.442259
24    2.440098
25    2.440431
26    2.438927
27    2.437555
28    2.441952
dtype: float64
```

In [ ]: