```
import pandas as pd
import numpy as np


#Load the dataset in dataframe object df
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/StudentsPerformance.csv')

#Display the data frame
df
```

	math score	reading score	writing score	placement score	club join year	placement offer count
0	74.0	80	80	92	2019.0	3
1	76.0	79	66	89	2018.0	3
2	93.0	80	74	75	2021.0	2
3	72.0	85	68	66	2020.0	1
4	65.0	94	NaN	79	2017.0	2
5	73.0	93	68	82	2021.0	2
6	79.0	81	76	95	2020.0	2
7	65.0	90	75	86	2021.0	3
8	78.0	88	74	34	2020.0	1
9	61.0	91	76	83	2021.0	2
10	78.0	78	43	89	2020.0	3
11	74.0	86	71	94	2018.0	3
12	64.0	85	66	90	2021.0	3
13	66.0	74	67	90	2021.0	3
14	72.0	98	80	97	2020.0	3
15	NaN	78	72	89	2021.0	3
16	78.0	89	67	94	2021.0	3
17	70.0	77	na	86	2019.0	3
18	74.0	88	76	95	2019.0	3
19	55.0	75	87	84	2019.0	2
20	78.0	78	63	83	2021.0	2
21	64.0	80	78	95	2019.0	3
22	73.0	77	78	79	2020.0	2
23	69.0	82	71	96	2020.0	3
24	79.0	92	76	94	2021.0	1
25	63.0	77	64	70	2018.0	1
26	72.0	93	67	100	NaN	3
27	71.0	94	62	80	2019.0	2
28	64.0	77	73	98	2020.0	3
29	76.0	85	78	96	2015.0	3
30	64.0	84	70	83	2021.0	2

isnull()

```
#Use isnull() function to check null values in the dataset.
df.isnull()
```

	math score	reading score	writing score	placement score	club join year	placement offer count
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	True	False	False	False
5	False	False	False	False	False	False
6	False	False	False	False	False	False
7	False	False	False	False	False	False
8	False	False	False	False	False	False
9	False	False	False	False	False	False
10	False	False	False	False	False	False
11	False	False	False	False	False	False
12	False	False	False	False	False	False
13	False	False	False	False	False	False
14	False	False	False	False	False	False
15	True	False	False	False	False	False
16	False	False	False	False	False	False
17	False	False	False	False	False	False
18	False	False	False	False	False	False
19	False	False	False	False	False	False
20	False	False	False	False	False	False
21	False	False	False	False	False	False
22	False	False	False	False	False	False
23	False	False	False	False	False	False
24	False	False	False	False	False	False
25	False	False	False	False	False	False
26	False	False	False	False	True	False
27	False	False	False	False	False	False
28	False	False	False	False	False	False
29	False	False	False	False	False	False

```
#To create a series true for NaN values for specific columns. for example
#math score in dataset and display data with only math score as NaN

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

	math score	reading score	writing score	placement score	club join year	placement offer count
15	NaN	78	72	89	2021.0	3

notnull()

```
#Use notnull() function to check null values in the dataset.
df.notnull()
```

	math score	reading score	writing score	placement score	club join year	placement offer count
0	True	True	True	True	True	True
1	True	True	True	True	True	True
2	True	True	True	True	True	True
3	True	True	True	True	True	True
4	True	True	False	True	True	True
5	True	True	True	True	True	True
6	True	True	True	True	True	True
7	True	True	True	True	True	True
8	True	True	True	True	True	True
9	True	True	True	True	True	True
10	True	True	True	True	True	True
11	True	True	True	True	True	True
12	True	True	True	True	True	True
13	True	True	True	True	True	True
14	True	True	True	True	True	True
15	False	True	True	True	True	True
16	True	True	True	True	True	True
17	True	True	True	True	True	True
18	True	True	True	True	True	True
19	True	True	True	True	True	True
20	True	True	True	True	True	True
21	True	True	True	True	True	True
22	True	True	True	True	True	True
23	True	True	True	True	True	True
24	True	True	True	True	True	True
25	True	True	True	True	True	True
26	True	True	True	True	False	True
27	True	True	True	True	True	True
28	True	True	True	True	True	True
29	True	True	True	True	True	True

```
#To create a series true for NaN values for specific columns. for example
#math score in dataset and display data with only math score as NaN

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

	math score	reading score	writing score	placement score	club join year	placement offer count
0	74.0	80	80	92	2019.0	3
1	76.0	79	66	89	2018.0	3
2	93.0	80	74	75	2021.0	2
3	72.0	85	68	66	2020.0	1
4	65.0	94	NaN	79	2017.0	2
5	73.0	93	68	82	2021.0	2
6	79.0	81	76	95	2020.0	2
7	65.0	90	75	86	2021.0	3
8	78.0	88	74	34	2020.0	1
9	61.0	91	76	83	2021.0	2
10	78.0	78	43	89	2020.0	3
11	74.0	86	71	94	2018.0	3
12	64.0	85	66	90	2021.0	3
13	66.0	74	67	90	2021.0	3
14	72.0	98	80	97	2020.0	3
16	78.0	89	67	94	2021.0	3
17	70.0	77	na	86	2019.0	3
18	74.0	88	76	95	2019.0	3
19	55.0	75	87	84	2019.0	2
20	78.0	78	63	83	2021.0	2
21	64.0	80	78	95	2019.0	3
22	73.0	77	78	79	2020.0	2
23	69.0	82	71	96	2020.0	3
24	79.0	92	76	94	2021.0	1
25	63.0	77	64	70	2018.0	1
26	72.0	93	67	100	NaN	3
27	71.0	94	62	80	2019.0	2
28	64.0	77	73	98	2020.0	3
29	76.0	85	78	96	2015.0	3
30	64.0	84	70	83	2021.0	2

```
#Label Encoding
#from sklearn.preprocessing import LabelEncoder
#"""le = LabelEncoder()
#df['gender'] = le.fit_transform(df['gender'])
#newdf=df
#df"""
#no categorical values hence no need
```

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

```
#For replacing null values with NaN
missing_values = ["Na", "na"]
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/StudentsPerformance.csv", na_values = missing_values)
df
```

	math score	reading score	writing score	placement score	club join year	placement offer count
0	74.0	80	80.0	92	2019.0	3
1	76.0	79	66.0	89	2018.0	3
2	93.0	80	74.0	75	2021.0	2
3	72.0	85	68.0	66	2020.0	1
4	65.0	94	NaN	79	2017.0	2
5	73.0	93	68.0	82	2021.0	2
6	79.0	81	76.0	95	2020.0	2
7	65.0	90	75.0	86	2021.0	3
8	78.0	88	74.0	34	2020.0	1
9	61.0	91	76.0	83	2021.0	2
10	78.0	78	43.0	89	2020.0	3
11	74.0	86	71.0	94	2018.0	3
12	64.0	85	66.0	90	2021.0	3
13	66.0	74	67.0	90	2021.0	3
14	72.0	98	80.0	97	2020.0	3
15	NaN	78	72.0	89	2021.0	3
16	78.0	89	67.0	94	2021.0	3
17	70.0	77	NaN	86	2019.0	3
18	74.0	88	76.0	95	2019.0	3
19	55.0	75	87.0	84	2019.0	2
20	78.0	78	63.0	83	2021.0	2
21	64.0	80	78.0	95	2019.0	3
22	73.0	77	78.0	79	2020.0	2
23	69.0	82	71.0	96	2020.0	3
24	79.0	92	76.0	94	2021.0	1
25	63.0	77	64.0	70	2018.0	1
26	72.0	93	67.0	100	NaN	3
27	71.0	94	62.0	80	2019.0	2
28	64.0	77	73.0	98	2020.0	3
29	76.0	85	78.0	96	2015.0	3
30	64.0	84	70.0	83	2021.0	2

```
#Filling null values with a single value

#filling missing value using fillna()
ndf=df
ndf.fillna(0)
```

	math score	reading score	writing score	placement score	club join year	placement offer count
0	74.0	80	80.0	92	2019.0	3
1	76.0	79	66.0	89	2018.0	3
2	93.0	80	74.0	75	2021.0	2
3	72.0	85	68.0	66	2020.0	1
4	65.0	94	0.0	79	2017.0	2
5	73.0	93	68.0	82	2021.0	2
6	79.0	81	76.0	95	2020.0	2
7	65.0	90	75.0	86	2021.0	3
8	78.0	88	74.0	34	2020.0	1
9	61.0	91	76.0	83	2021.0	2
10	78.0	78	43.0	89	2020.0	3
11	74.0	86	71.0	94	2018.0	3
12	64.0	85	66.0	90	2021.0	3
13	66.0	74	67.0	90	2021.0	3
14	72.0	98	80.0	97	2020.0	3
15	0.0	78	72.0	89	2021.0	3
16	78.0	89	67.0	94	2021.0	3
17	70.0	77	0.0	86	2019.0	3
18	74.0	88	76.0	95	2019.0	3
19	55.0	75	87.0	84	2019.0	2
20	78.0	78	63.0	83	2021.0	2
21	64.0	80	78.0	95	2019.0	3
22	73.0	77	78.0	79	2020.0	2
23	69.0	82	71.0	96	2020.0	3
24	79.0	92	76.0	94	2021.0	1
25	63.0	77	64.0	70	2018.0	1
26	72.0	93	67.0	100	0.0	3
27	71.0	94	62.0	80	2019.0	2
28	64.0	77	73.0	98	2020.0	3
29	76.0	85	78.0	96	2015.0	3
30	64.0	84	70.0	83	2021.0	2

```
#filling missing values using mean, median and standard deviation of that column.
df['math score'] = df['math score'].fillna(df['math score'].mean())

df["math score"] = df["math score"].fillna(df["math score"].median())

df["math score"] = df["math score"].fillna(df["math score"].std())



#replacing missing values in forenoon column with minimum/maximum number of that column
df["math score"] = df["math score"].fillna(df["math score"].min())

df["math score"] = df["math score"].fillna(df["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
```

	math score	reading score	writing score	placement score	club join year	placement offer count
0	74.000000	80	80.0	92	2019.0	3
1	76.000000	79	66.0	89	2018.0	3
2	93.000000	80	74.0	75	2021.0	2
3	72.000000	85	68.0	66	2020.0	1
4	65.000000	94	NaN	79	2017.0	2
5	73.000000	93	68.0	82	2021.0	2
6	79.000000	81	76.0	95	2020.0	2
7	65.000000	90	75.0	86	2021.0	3
8	78.000000	88	74.0	34	2020.0	1
9	61.000000	91	76.0	83	2021.0	2
10	78.000000	78	43.0	89	2020.0	3
11	74.000000	86	71.0	94	2018.0	3
12	64.000000	85	66.0	90	2021.0	3
13	66.000000	74	67.0	90	2021.0	3
14	72.000000	98	80.0	97	2020.0	3
15	71.333333	78	72.0	89	2021.0	3
16	78.000000	89	67.0	94	2021.0	3
17	70.000000	77	NaN	86	2019.0	3
18	74.000000	88	76.0	95	2019.0	3
19	55.000000	75	87.0	84	2019.0	2
20	78.000000	78	63.0	83	2021.0	2
21	64.000000	80	78.0	95	2019.0	3
22	73.000000	77	78.0	79	2020.0	2
23	69.000000	82	71.0	96	2020.0	3
24	79.000000	92	76.0	94	2021.0	1
25	63.000000	77	64.0	70	2018.0	1
26	72.000000	93	67.0	100	NaN	3
27	71.000000	94	62.0	80	2019.0	2
28	64.000000	77	73.0	98	2020.0	3
29	76.000000	85	78.0	96	2015.0	3

```
#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)
```

	math score	reading score	writing score	placement score	club join year	placement offer count
0	74.000000	80	80.0	92	2019.0	3
1	76.000000	79	66.0	89	2018.0	3
2	93.000000	80	74.0	75	2021.0	2
3	72.000000	85	68.0	66	2020.0	1
4	65.000000	94	-99.0	79	2017.0	2
5	73.000000	93	68.0	82	2021.0	2
6	79.000000	81	76.0	95	2020.0	2
7	65.000000	90	75.0	86	2021.0	3
8	78.000000	88	74.0	34	2020.0	1
9	61.000000	91	76.0	83	2021.0	2
10	78.000000	78	43.0	89	2020.0	3
11	74.000000	86	71.0	94	2018.0	3
12	64.000000	85	66.0	90	2021.0	3
13	66.000000	74	67.0	90	2021.0	3
14	72.000000	98	80.0	97	2020.0	3
15	71.333333	78	72.0	89	2021.0	3
16	78.000000	89	67.0	94	2021.0	3
17	70.000000	77	-99.0	86	2019.0	3
18	74.000000	88	76.0	95	2019.0	3
19	55.000000	75	87.0	84	2019.0	2
20	78.000000	78	63.0	83	2021.0	2
21	64.000000	80	78.0	95	2019.0	3
22	73.000000	77	78.0	79	2020.0	2
23	69.000000	82	71.0	96	2020.0	3
24	79.000000	92	76.0	94	2021.0	1
25	63.000000	77	64.0	70	2018.0	1
26	72.000000	93	67.0	100	-99.0	3
27	71.000000	94	62.0	80	2019.0	2
28	64.000000	77	73.0	98	2020.0	3
29	76.000000	85	78.0	96	2015.0	3

```
#Deleting null values using dropna() method

#To drop rows with at least 1 null value
ndf.dropna()
```

	math score	reading score	writing score	placement score	club join year	placement offer count
0	74.000000	80	80.0	92	2019.0	3
1	76.000000	79	66.0	89	2018.0	3
2	93.000000	80	74.0	75	2021.0	2
3	72.000000	85	68.0	66	2020.0	1
5	73.000000	93	68.0	82	2021.0	2
6	79.000000	81	76.0	95	2020.0	2
7	65.000000	90	75.0	86	2021.0	3
8	78.000000	88	74.0	34	2020.0	1
9	61.000000	91	76.0	83	2021.0	2
10	78.000000	78	43.0	89	2020.0	3
11	74.000000	86	71.0	94	2018.0	3
12	64.000000	85	66.0	90	2021.0	3
13	66.000000	74	67.0	90	2021.0	3
14	72.000000	98	80.0	97	2020.0	3
15	71.333333	78	72.0	89	2021.0	3
16	78.000000	89	67.0	94	2021.0	3
18	74.000000	88	76.0	95	2019.0	3
19	55.000000	75	87.0	84	2019.0	2
20	78.000000	78	63.0	83	2021.0	2
21	64.000000	80	78.0	95	2019.0	3
22	73.000000	77	78.0	79	2020.0	2
23	69.000000	82	71.0	96	2020.0	3
24	79.000000	92	76.0	94	2021.0	1
25	63.000000	77	64.0	70	2018.0	1
27	71.000000	94	62.0	80	2019.0	2
28	64.000000	77	73.0	98	2020.0	3
29	76.000000	85	78.0	96	2015.0	3
30	64.000000	84	70.0	83	2021.0	2

```
#To Drop rows if all values in that row are missing
ndf.dropna(how = 'all')
```

	math score	reading score	writing score	placement score	club join year	placement offer count
0	74.000000	80	80.0	92	2019.0	3
1	76.000000	79	66.0	89	2018.0	3
2	93.000000	80	74.0	75	2021.0	2
3	72.000000	85	68.0	66	2020.0	1
4	65.000000	94	NaN	79	2017.0	2
5	73.000000	93	68.0	82	2021.0	2
6	79.000000	81	76.0	95	2020.0	2
7	65.000000	90	75.0	86	2021.0	3
8	78.000000	88	74.0	34	2020.0	1
9	61.000000	91	76.0	83	2021.0	2
10	78.000000	78	43.0	89	2020.0	3
11	74.000000	86	71.0	94	2018.0	3
12	64.000000	85	66.0	90	2021.0	3
13	66.000000	74	67.0	90	2021.0	3
14	72.000000	98	80.0	97	2020.0	3
15	71.333333	78	72.0	89	2021.0	3
16	78.000000	89	67.0	94	2021.0	3
17	70.000000	77	NaN	86	2019.0	3
18	74.000000	88	76.0	95	2019.0	3
19	55.000000	75	87.0	84	2019.0	2
20	78.000000	78	63.0	83	2021.0	2
21	64.000000	80	78.0	95	2019.0	3
22	73.000000	77	78.0	79	2020.0	2
23	69.000000	82	71.0	96	2020.0	3
24	79.000000	92	76.0	94	2021.0	1
25	63.000000	77	64.0	70	2018.0	1
26	72.000000	93	67.0	100	NaN	3
27	71.000000	94	62.0	80	2019.0	2
28	64.000000	77	73.0	98	2020.0	3
29	76.000000	85	78.0	96	2015.0	3

```
#To Drop columns with at least 1 null value.
ndf.dropna(axis = 1)
```

	math score	reading score	placement score	placement offer count
0	74.000000	80	92	3
1	76.000000	79	89	3
2	93.000000	80	75	2
3	72.000000	85	66	1
4	65.000000	94	79	2
5	73.000000	93	82	2
6	79.000000	81	95	2
7	65.000000	90	86	3
8	78.000000	88	34	1
9	61.000000	91	83	2
10	78.000000	78	89	3
11	74.000000	86	94	3
12	64.000000	85	90	3
13	66.000000	74	90	3
14	72.000000	98	97	3
15	71.333333	78	89	3
16	78.000000	89	94	3
17	70.000000	77	86	3
18	74.000000	88	95	3
19	55.000000	75	84	2
20	78.000000	78	83	2
21	64.000000	80	95	3
22	73.000000	77	79	2
23	69.000000	82	96	3
24	79.000000	92	94	1
25	63.000000	77	70	1
26	72.000000	93	100	3
27	71.000000	94	80	2
28	64.000000	77	98	3
29	76.000000	85	96	3
30	64.000000	84	83	2

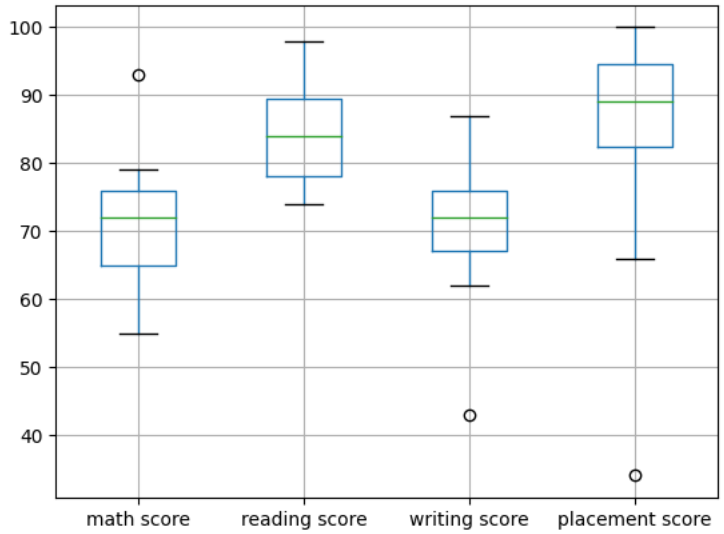
```
#To drop rows with at least 1 null value in CSV file.
#making new data frame with dropped NA value

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

	math score	reading score	writing score	placement score	club join year	placement offer count
0	74.000000	80	80.0	92	2019.0	3
1	76.000000	79	66.0	89	2018.0	3
2	93.000000	80	74.0	75	2021.0	2
3	72.000000	85	68.0	66	2020.0	1
5	73.000000	93	68.0	82	2021.0	2
6	79.000000	81	76.0	95	2020.0	2
7	65.000000	90	75.0	86	2021.0	3
8	78.000000	88	74.0	34	2020.0	1
9	61.000000	91	76.0	83	2021.0	2
10	78.000000	78	43.0	89	2020.0	3
11	74.000000	86	71.0	94	2018.0	3
12	64.000000	85	66.0	90	2021.0	3
13	66.000000	74	67.0	90	2021.0	3
14	72.000000	98	80.0	97	2020.0	3
15	71.333333	78	72.0	89	2021.0	3
16	78.000000	89	67.0	94	2021.0	3
18	74.000000	88	76.0	95	2019.0	3
19	55.000000	75	87.0	84	2019.0	2
20	78.000000	78	63.0	83	2021.0	2
21	64.000000	80	78.0	95	2019.0	3
22	73.000000	77	78.0	79	2020.0	2
23	69.000000	82	71.0	96	2020.0	3
24	79.000000	92	76.0	94	2021.0	1
25	63.000000	77	64.0	70	2018.0	1
27	71.000000	94	62.0	80	2019.0	2
28	64.000000	77	73.0	98	2020.0	3
29	76.000000	85	78.0	96	2015.0	3
30	64.000000	84	70.0	83	2021.0	2

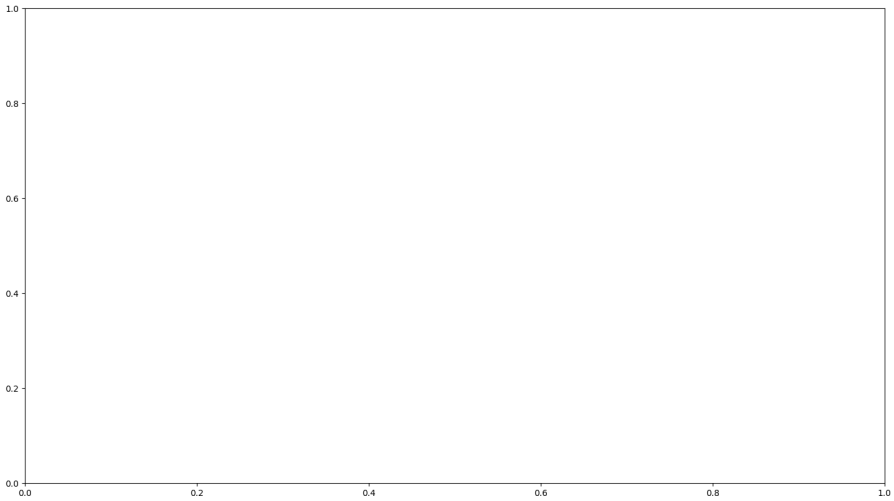
Identification and Handling of Outliers

```
#Detecting outliers using Boxplot:


#Select the columns for boxplot and draw the boxplot.

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

```
<Axes: >
```




```
#We can now print the outliers for each column with reference to the box plot.

print(np.where(df['math score']>90))

     (array([2]),)


print(np.where(df['reading score']<75))

     (array([13]),)


print(np.where(df['writing score']<60))

     (array([10]),)


#Detecting outliers using Scatterplot:


#Import pandas , numpy and matplotlib libraries
import matplotlib.pyplot as plt


#Draw the scatter plot with placement score and placement offer count
fig, ax = plt.subplots(figsize = (18,10))
```

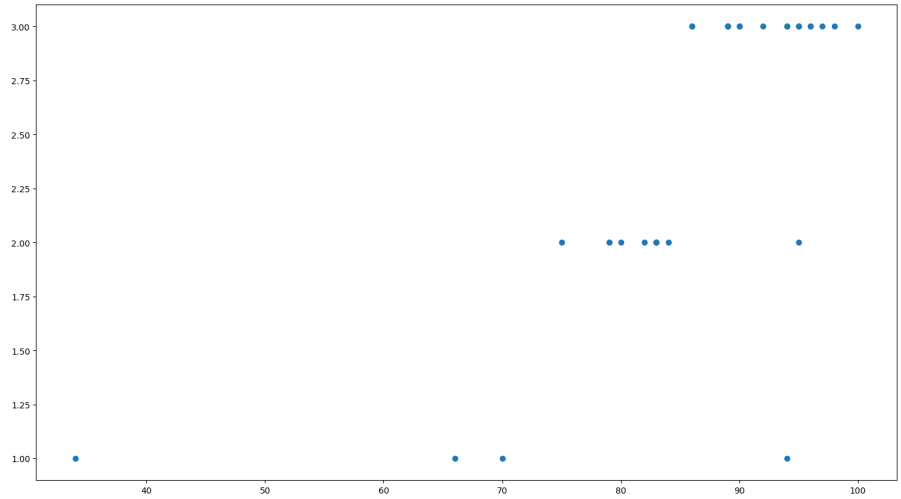



```
ax.scatter(df['placement score'], df['placement offer count'])
```

```
<matplotlib.collections.PathCollection at 0x7cfd196cafb0>
```

```
plt.show()
```

```
fig, ax = plt.subplots(figsize = (18,10))
ax.scatter(df['placement score'], df['placement offer count'])
plt.show()

#Labels to the axis can be assigned (Optional)
ax.set_xlabel('(Proportion non-retail business acres)/(town)')
ax.set_ylabel('(Full-value property-tax rate)/($10,000)')
```




```
Text(4.444444444444452, 0.5, '(Full-value property-tax rate)/($10,000)')
```

```
#We can now print the outliers with reference to scatter plot.
print(np.where((df['placement score']<80) & (df['placement offer count']>1)))
```

```
(array([ 2,  4, 22]),)
```

```
print(np.where((df['placement score']>85) & (df['placement offer count']<3)))
```

```
(array([ 6, 24]),)
```

```
#Detecting outliers using Z-Score:
```

```
#Import numpy and stats from scipy libraries
from scipy import stats
```

```
#Calculate Z-Score for mathscore column
z = np.abs(stats.zscore(df['math score']))
```

```
#Print Z-Score Value. It prints the z-score values of each data item of the column
print(z)
```

```
0     0.369953
1     0.647418
2     3.005869
3     0.092488
4     0.878639
5     0.231221
6     1.063615
7     0.878639
8     0.924883
```

```
9     1.433569
10    0.924883
11    0.369953
12    1.017371
13    0.739906
14    0.092488
15    0.000000
16    0.924883
17    0.184977
18    0.369953
19    2.265963
20    0.924883
21    1.017371
22    0.231221
23    0.323709
24    1.063615
25    1.156104
26    0.092488
27    0.046244
28    1.017371
29    0.647418
30    1.017371
Name: math score, dtype: float64
```

```
#Now to define an outlier threshold value is chosen.
threshold = 0.18
```

```
#Display the sample outliers
sample_outliers = np.where(z <threshold)
sample_outliers
```

```
(array([ 3, 14, 15, 26, 27]),)
```

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

```
#Sort Reading Score feature and store it into sorted_rscore.
sorted_rscore= sorted(df['reading score'])
```

```
#Print sorted_rscore
sorted_rscore
```

```
[74,
 75,
 77,
 77,
 77,
 77,
 78,
 78,
 78,
 79,
 80,
 80,
 80,
 81,
 82,
 84,
 85,
 85,
 85,
 86,
 88,
 88,
 89,
 90,
 91,
 92,
 93,
 93,
 94,
 94,
 98]
```

```
#Calculate and print Quartile 1 and Quartile 3
q1 = np.percentile(sorted_rscore, 75)
q3 = np.percentile(sorted_rscore, 100)

print(q1,q3)
```

```
     89.5 98.0
```

```
#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)
```

```
     76.75 110.75
```

```
#Print Outliers

r_outliers = []
for i in sorted_rscore:
  if (i<lwr_bound or i>upr_bound):
    r_outliers.append(i)
print(r_outliers)
```

```
     [74, 75]
```

Handling of Outliers:

```
#Trimming/removing the outlier:


'''new_df=df
for i in r_outliers:
  new_df.drop(i,inplace=True)
new_df'''
```

```
     'new_df=df\nfor i in r_outliers:\n  new_df.drop(i,inplace=True)\nnew_df'
```

```
#Quantile based flooring and capping:


df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/StudentsPerformance.csv')
df_stud=df
ninetieth_percentile = np.percentile(df_stud['math score'], 90)
b = np.where(df_stud['math score']>ninetieth_percentile,
ninetieth_percentile, df_stud['math score'])
print("New array:",b)
```

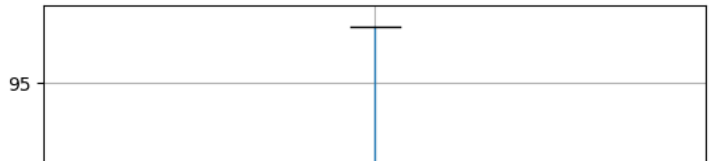
```
     New array: [74. 76. 93. 72. 65. 73. 79. 65. 78. 61. 78. 74. 64. 66. 72. nan 78. 70.
      74. 55. 78. 64. 73. 69. 79. 63. 72. 71. 64. 76. 64.]
```

```
df_stud.insert(1,"m score",b,True)
df_stud
```

	math score	m score	reading score	writing score	placement score	club join year	placement offer count
0	74.0	74.0	80	80	92	2019.0	3
1	76.0	76.0	79	66	89	2018.0	3
2	93.0	93.0	80	74	75	2021.0	2
3	72.0	72.0	85	68	66	2020.0	1
4	65.0	65.0	94	NaN	79	2017.0	2
5	73.0	73.0	93	68	82	2021.0	2
6	79.0	79.0	81	76	95	2020.0	2
7	65.0	65.0	90	75	86	2021.0	3
8	78.0	78.0	88	74	34	2020.0	1
9	61.0	61.0	91	76	83	2021.0	2
10	78.0	78.0	78	43	89	2020.0	3
11	74.0	74.0	86	71	94	2018.0	3
12	64.0	64.0	85	66	90	2021.0	3
13	66.0	66.0	74	67	90	2021.0	3
14	72.0	72.0	98	80	97	2020.0	3
15	NaN	NaN	78	72	89	2021.0	3
16	78.0	78.0	89	67	94	2021.0	3
17	70.0	70.0	77	na	86	2019.0	3
18	74.0	74.0	88	76	95	2019.0	3
19	55.0	55.0	75	87	84	2019.0	2
20	78.0	78.0	78	63	83	2021.0	2
21	64.0	64.0	80	78	95	2019.0	3
22	73.0	73.0	77	78	79	2020.0	2
23	69.0	69.0	82	71	96	2020.0	3
24	79.0	79.0	92	76	94	2021.0	1
25	63.0	63.0	77	64	70	2018.0	1
26	72.0	72.0	93	67	100	NaN	3
27	71.0	71.0	94	62	80	2019.0	2
28	64.0	64.0	77	73	98	2020.0	3
29	76.0	76.0	85	78	96	2015.0	3
30	64.0	64.0	84	70	83	2021.0	2

```
#Mean/Median imputation:


#Plot the box plot for reading score
col = ['reading score']
df.boxplot(col)
```

```
<Axes: >
```




```
#Outliers are seen in box plot.

#Calculate the median of reading score by using sorted_rscore
median=np.median(sorted_rscore)
median
```

```
84.0
```


```
#Replace the upper bound outliers using median value
refined_df=df
refined_df['reading score'] = np.where(refined_df['reading score'] >upr_bound, median,refined_df['reading score'])

#Display redefined_df
refined_df
```