Data Wrangling II

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.

In [5]:

```
import pandas as pd
import numpy as np
student = pd.read_csv("/content/StudentsPerformance.csv")
```

In [6]:

```
student.info()
```

```
<class 'pandas.core.frame.DataFrame'> RangeIndex:
1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                        Non-Null Count  Dtype    0
---  ------                        --------------  -----
 gender                        1000 non-null   object
  1    race/ethnicity              1000 non-null   object
  2    parental level of education 1000 non-null   object
  3    lunch                       1000 non-null   object
  4    test_preparation_course     1000 non-null   object
  5    math_score                  991 non-null    float64
  6    reading_score               995 non-null    float64
  7    writing_score               994 non-null    float64 dtypes: float64(3),
 object(5) memory usage: 62.6+ KB
```

In [7]:

```
student.isnull().sum()
```

Out[7]:

```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test_preparation_course        0
math_score                     9
reading_score                  5
writing_score                  6
dtype: int64
```

In [8]:

```
#filling missing value by mean

student['math_score'].fillna(int(student['math_score'].mean()), inplace=True) In
```

[9]:

```
student.isnull().sum()
```

Out[9]:

```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test_preparation_course        0
math_score                     0
reading_score                  5
writing_score                  6
dtype: int64
```

In [10]:

```
# filling a missing value with previous ones
student['reading_score'].fillna(method ='pad',inplace=True) In
```

[11]:

```
student.isnull().sum()
```

Out[11]:

```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test_preparation_course        0
math_score                     0
reading_score                  0
writing_score                  6
dtype: int64
```

In [12]:

```
#filling missing value by median
student['writing_score'].fillna(int(student['writing_score'].median()), inplace=True)
```

In [13]:

```
student.isnull().sum()
```

Out[13]:

```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test_preparation_course        0
math_score                     0
reading_score                  0
writing_score                  0
dtype: int64
```

2.Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.

In [14]:

```
from numpy.random import seed
from numpy.random import randn
from numpy import mean
from numpy import std
seed(1)
#univariate dataset- single variable/ attribute
#multivariate detaset-muliple variables/attributes
data=5*randn(10000)+50

print('mean=%.3f stdv=%.3f' %(mean(data), std(data)))
```

```
mean=50.049 stdv=4.994
```

Standard Deviation Method

In [15]:

```
data_mean = mean(data)
data_std = std(data)
cut_off = data_std * 3
lower = data_mean - cut_off
upper = data_mean + cut_off
```

In [16]:

```
outliers=[x for x in data outliers Out[16]:
```

```
[65.15428556186015,
 69.79301352018982, 66.60539378085183,
 34.73117809786848, 34.23321274904475,
 34.91984007395351,
 67.1633171589778,
 34.679293219474495,
 68.70124451852294, 65.67523670043954,
 66.19171598376188, 33.73482882511691,
 65.66014864070253,
 65.06377284118616,
 34.0469182658796,
 33.6969245211173,
 67.02151137874486, 65.59239795391275,
 66.49270261640393,
 65.74492012609815,
 33.525707966507426,
 34.72183379792847, 70.1342452227369,
 33.90433947188079, 65.55945915508362,
 68.06638503541573, 66.99057828251213,
 67.80436660352774,
 31.717799503726024]
```