Name : Suryal D . Khirade
Roll NO: T190424399
Assignment No :02

Data Wrangling II Perform the following operations using Python on any open source dataset
(eg. data.csv)
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. Reason
and document your approach properly.

Import all the required Python Libraries.

In [1]:
```
import numpy as np
import pandas as pd
import random as rd
import seaborn as sns
```

In [19]:
```
df = pd.read_csv("C:\\Users\\alisu\\Downloads\\archive (1)\\Student Perform
```

In [20]:
```
df.describe()
```

Out[20]:

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

In [21]:
```
df.isnull().sum()
```

Out[21]:
```
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
```

Create a DataFrame from the dictionary

In [22]:
```
data={'studentid':[i for  i in  range(1,101)],
      'Age':[rd.randint(15,18) for i in range(100)],
      'class':[rd.choice(['9 th','10 th' ,'11 th' ,'12 th']) for _ in range
      ,'attendence':[rd.uniform(10,100) for _ in range(100)],
      'score':[rd.randint(30,100) for _ in range(100) ]
      }
```

In [23]:
```
df.to_csv('StudentPerformance' , index =False)
df=pd.DataFrame(data )
```

In [24]:
```
df
```

Out[24]:

	studentid	Age	class	attendence	score
0	1	17	11 th	47.894979	50
1	2	18	11 th	46.373847	100
2	3	18	11 th	46.636907	76
3	4	16	11 th	25.126863	77
4	5	16	10 th	55.518510	41
...	...	...	...	...	...
95	96	17	10 th	34.682479	59
96	97	16	10 th	52.756471	87
97	98	15	10 th	72.759473	30
98	99	16	12 th	59.991836	82
99	100	15	12 th	35.561988	91

100 rows × 5 columns

Load the Dataset into pandas dataframe.

In [31]:
```
import pandas as pd
df = pd.read_csv("C:\\Users\\alisu\\Downloads\\archive (1)\\Student Perform
df.head()
```

Out[31]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

In [32]:
```
df.shape
```

Out[32]:
```
(1000, 8)
```

In [33]:
```
df.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                   1000 non-null   int64 
 6   reading score                1000 non-null   int64 
 7   writing score                1000 non-null   int64 
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

In [34]:
```
df.describe()
```

Out[34]:

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

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.

Data Preprocessing

In [35]:
```
df.isnull().sum()
```

Out[35]:
```
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
```

In [36]:
```
df.nunique()
```

Out[36]:
```
gender                         2
race/ethnicity                 5
parental level of education    6
lunch                          2
test preparation course        2
math score                    81
reading score                 72
writing score                 77
dtype: int64
```

In [37]:
```
df["gender"].value_counts() #categorical column
```

Out[37]:
```
gender
female    518
male      482
Name: count, dtype: int64
```

In [38]:
```
df['gender'].fillna('female',inplace=True)
df.isnull().sum()
```

Out[38]:
```
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
```

In [39]:
```
df['gender'].mode(0)
```

Out[39]:
```
0    female
Name: gender, dtype: object
```

In [40]:
```
df['race/ethnicity'].value_counts()
```

Out[40]:
```
race/ethnicity
group C    319
group D    262
group B    190
group E    140
group A     89
Name: count, dtype: int64
```

In [41]:
```
df['race/ethnicity'].fillna('Group C',inplace=True)
df.isnull().sum()
```

Out[41]:
```
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
```

In [42]:
```
df['lunch'].value_counts()
```

Out[42]:
```
lunch
standard        645
free/reduced    355
Name: count, dtype: int64
```

In [43]:
```
df['lunch'].mode()
```

Out[43]:
```
0    standard
Name: lunch, dtype: object
```

In [44]:
```
df['lunch'].mode()[0]
```

Out[44]:
```
'standard'
```

In [45]:
```
df['lunch'].fillna(df['lunch'].mode()[0],inplace=True)
df.isnull().sum()
```

Out[45]:
```
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
```

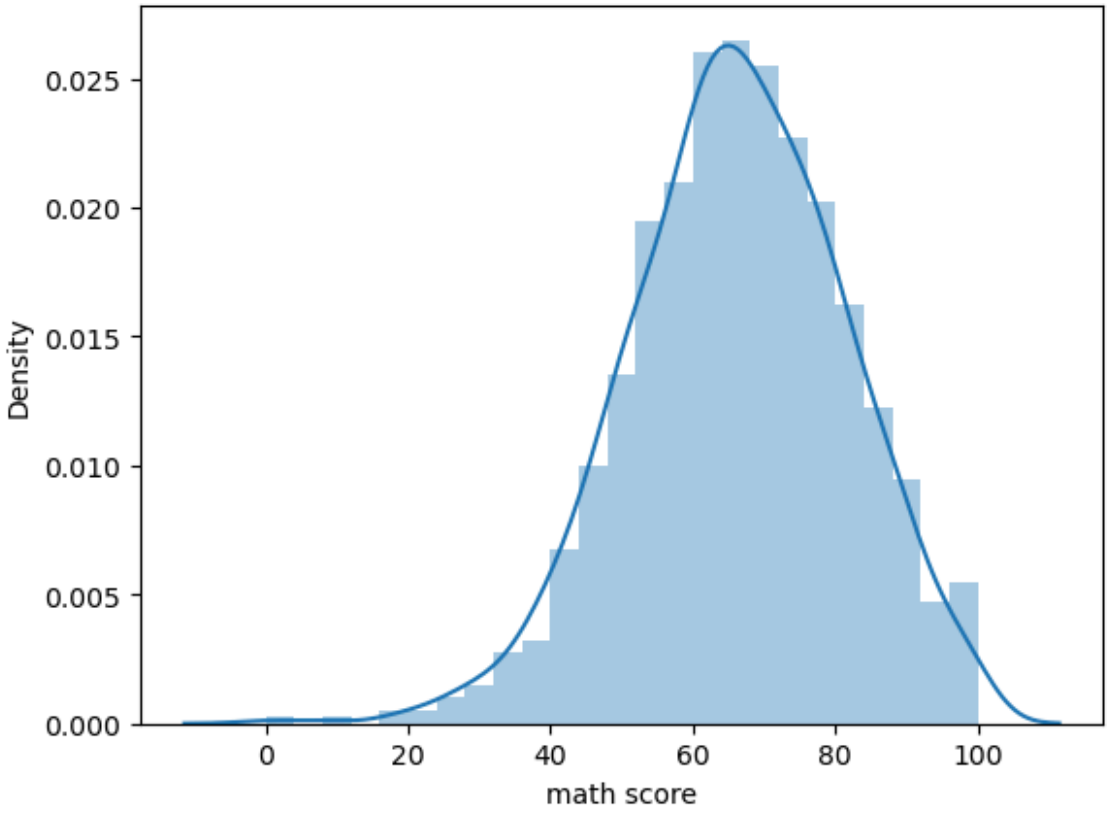
In [46]:
```
df.isnull().sum()
```

Out[46]:
```
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
```

In [47]:
```
sns.distplot(df['math score'])
```

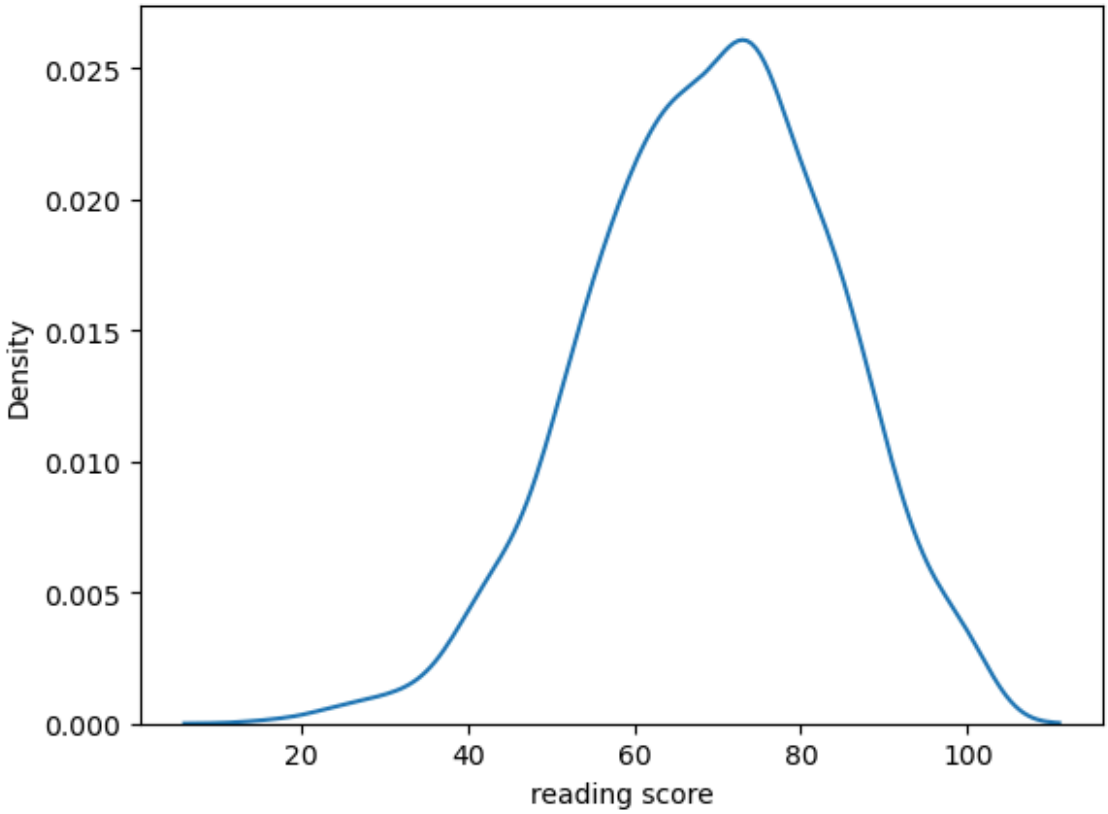
```
C:\Users\alisu\AppData\Local\Temp\ipykernel_17132\2354272343.py:1: UserWa
rning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function w
ith
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['math score'])
```

Out[47]:
```
<Axes: xlabel='math score', ylabel='Density'>
```



In [48]:
```
sns.kdeplot(df['reading score'])#normally distributed
```

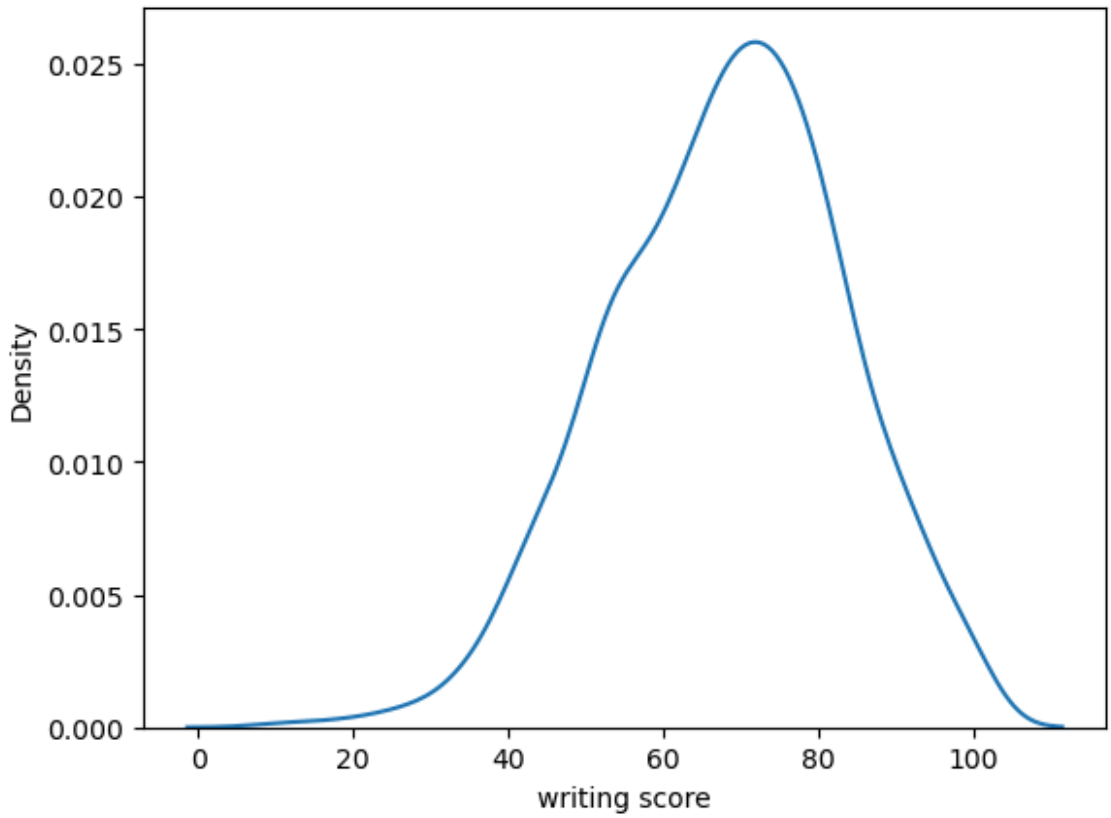
Out[48]:
```
<Axes: xlabel='reading score', ylabel='Density'>
```



In [49]:
```
sns.distplot(df['writing score'],hist=False,)
```

```
C:\Users\alisu\AppData\Local\Temp\ipykernel_17132\3897196859.py:1: UserWa
rning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function w
ith
similar flexibility) or `kdeplot` (an axes-level function for kernel dens
ity plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['writing score'],hist=False,)
```

Out[49]:
```
<Axes: xlabel='writing score', ylabel='Density'>
```



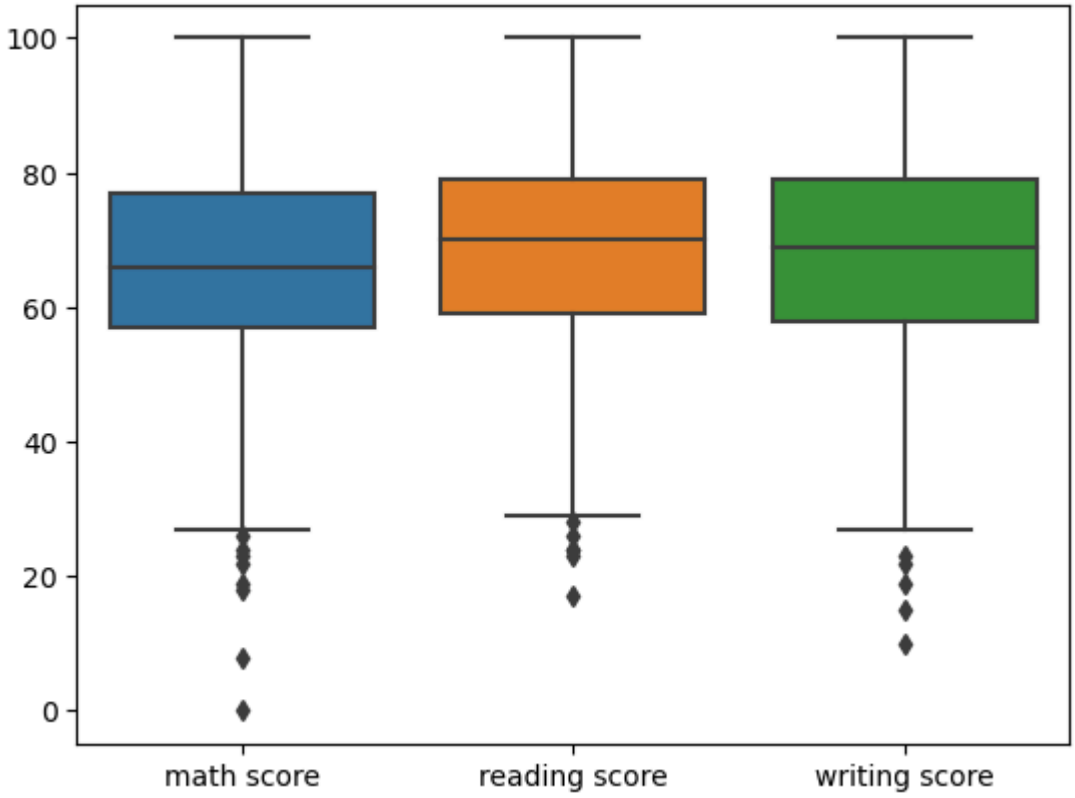
In [50]:
```
df['math score'].fillna(df['math score'].mean(),inplace=True)
df.isnull().sum()
```

Out[50]:
```
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
```

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

In [51]:
```
sns.boxplot(df)
```

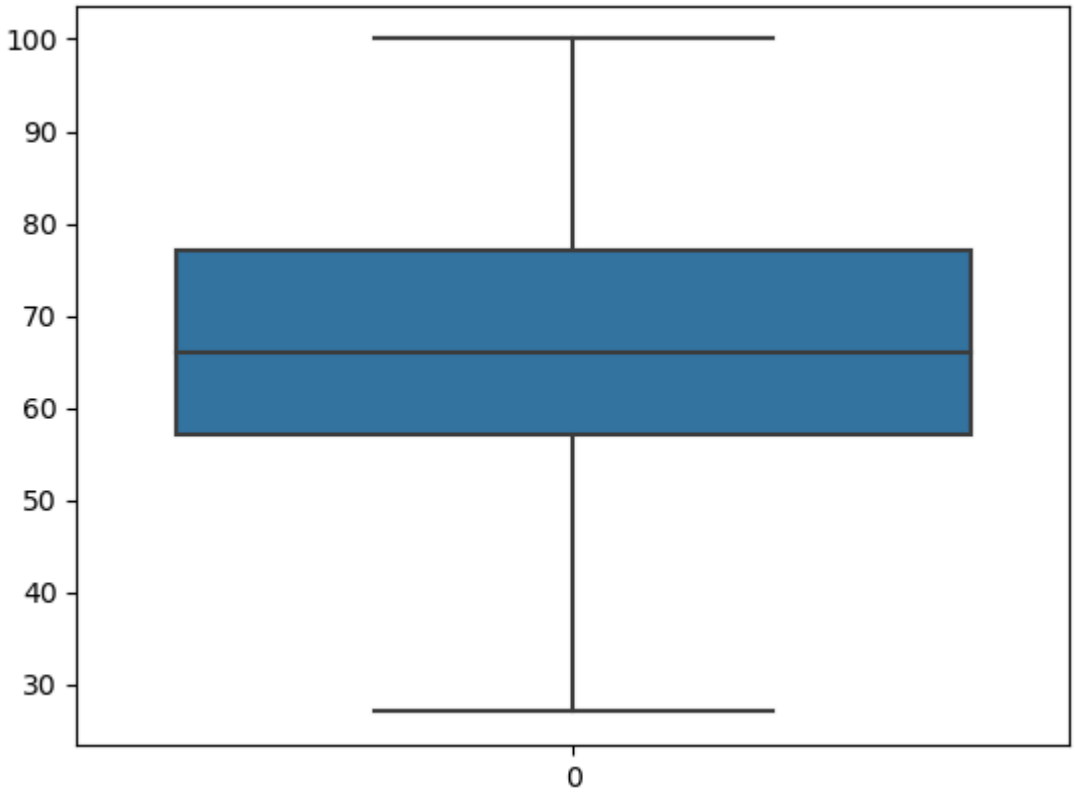
Out[51]: <Axes: >



In [52]:
```
Q1=df['math score'].quantile(0.25)
Q3=df['math score'].quantile(0.75)
IQR=Q3-Q1
lower= Q1-(1.5*IQR)
upper=Q3+(1.5*IQR)
```

In [53]:
```
np.clip(df['math score'],lower,upper,inplace=True)
sns.boxplot(df['math score'])
```

Out[53]: <Axes: >



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 [54]:
```
df['math score'].skew()
-0.12912399951580147
df['reading score'].skew()
-0.2595478399998487
df['writing score'].skew()
-0.3125529577143879
from sklearn.preprocessing import StandardScaler,MinMaxScaler
scaler=StandardScaler()
```

In [55]:
```
scaler.fit(df[['math score']])
```

Out[55]: StandardScaler()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [58]:
```
df.head()
```

Out[58]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

In [59]:
```
scaler.fit(df[['reading score']])
```

Out[59]:
```
StandardScaler()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [60]:
```
scaled_rscore=scaler.transform(df[['reading score']])
```