Data Wrangling II

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.

Reason and document your approach properly.

In [2]:
```
import numpy as np
import pandas as pd
```

Dataset : Student's Academic Performance Dataset

In [3]:
```
df = pd.read_csv('StudentsPerformance.csv')
df1 = df.copy()
df
# we'll perform operation on df dataframe while df1 dataframe willbe our original datas
```

Out[3]:

	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	NaN	72.0	74.0
1	female	group C	some college	standard	completed	69.0	90.0	88.0
2	female	group B	master's degree	standard	none	90.0	95.0	NaN
3	male	NaN	associate's degree	free/reduced	none	47.0	57.0	44.0
4	male	group C	some college	standard	none	NaN	78.0	75.0
...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88.0	99.0	95.0
996	male	group C	high school	free/reduced	none	62.0	55.0	55.0
997	female	group C	high school	free/reduced	completed	59.0	71.0	65.0
998	female	group D	some college	standard	completed	68.0	78.0	77.0
999	female	group D	some college	free/reduced	none	77.0	86.0	86.0

1000 rows × 8 columns

In [4]:
```
df.shape
```

Out[4]:
```
(1000, 8)
```

In [5]:
```
df.dtypes
```

Out[5]:
```
gender                         object
race/ethnicity                 object
parental level of education    object
lunch                          object
test preparation course        object
math score                    float64
reading score                 float64
writing score                 float64
dtype: object
```

In [6]:
```
df.info
```

Out[6]:
```
<bound method DataFrame.info of      gender race/ethnicity parental level of education
lunch  \
0    female        group B           bachelor's degree      standard
1    female        group C                some college      standard
2    female        group B             master's degree      standard
3      male            NaN          associate's degree  free/reduced
4      male        group C                some college      standard
..      ...            ...                         ...           ...
995  female        group E             master's degree      standard
996    male        group C                 high school  free/reduced
997  female        group C                 high school  free/reduced
998  female        group D                some college      standard
999  female        group D                some college  free/reduced

    test preparation course  math score  reading score  writing score
0                      none         NaN           72.0           74.0
1                 completed        69.0           90.0           88.0
2                      none        90.0           95.0            NaN
3                      none        47.0           57.0           44.0
4                      none         NaN           78.0           75.0
..                      ...         ...            ...            ...
995               completed        88.0           99.0           95.0
996                    none        62.0           55.0           55.0
997               completed        59.0           71.0           65.0
998               completed        68.0           78.0           77.0
999                    none        77.0           86.0           86.0

[1000 rows x 8 columns]>
```

In [7]:
```
df.isnull()
```

Out[7]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	False	False	False	False	False	True	False	False
1	False	False	False	False	False	False	False	False

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
2	False	False	False	False	False	False	False	True
3	False	True	False	False	False	False	False	False
4	False	False	False	False	False	True	False	False
...	...	...	...	...	...	...	...	...
995	False	False	False	False	False	False	False	False
996	False	False	False	False	False	False	False	False
997	False	False	False	False	False	False	False	False
998	False	False	False	False	False	False	False	False
999	False	False	False	False	False	False	False	False

1000 rows × 8 columns

In [8]:
```
df.describe()
```

Out[8]:

	math score	reading score	writing score
count	993.000000	993.000000	995.000000
mean	66.499496	69.302115	68.157789
std	17.645885	15.301518	16.064390
min	0.000000	17.000000	10.000000
25%	57.000000	59.000000	57.500000
50%	66.000000	70.000000	69.000000
75%	77.000000	80.000000	79.000000
max	342.000000	212.000000	234.000000

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
gender                         0
race/ethnicity                 8
parental level of education    8
lunch                          6
test preparation course       13
math score                     7
reading score                  7
writing score                  5
dtype: int64
```

In [10]:
```
df.isnull().sum().sum()
```

Out[10]: 54

Unique values

In [11]:
```
df_col = df.columns
print("Unique data elements for each column in dataset :")
for i in df_col:
    print("Column " + str(i) + " : " + str(df[i].unique()))
```

```
Unique data elements for each column in dataset :
Column gender : ['female' 'male']
Column race/ethnicity : ['group B' 'group C' nan 'group D' 'group A' 'group E']
Column parental level of education : ["bachelor's degree" 'some college' "master's degre
e" "associate's degree"
 'high school' 'some high school' nan]
Column lunch : ['standard' 'free/reduced' nan]
Column test preparation course : ['none' 'completed' nan]
Column math score : [ nan  69.  90.  47.  71.  88.  40.  38.  58.  65.  78.  50.  46.  5
4.
  66.  44.  74.  73.  70.  62.  63.  97.  81.  75.  57.  55.  53.  59.
  82.  77.  33.  52.   0.  79.  39.  67.  45.  60.  61.  41.  49.  30.
  80.  72.  42.  76.  27.  43.  68.  85.  98.  87.  51.  99.  84.  91.
  83.  89.  22. 100.  96.  94.  48. 119. 342.  35.  34.  56.  86.  92.
  64.  37. 123.  28.  24.  26.  95.  36.  29.  32.  93.  19.  23.   8.]
Column reading score : [ 72.  90.  95.  57.  78.  83.  43.  64.  60.  nan  52.  81.  53.
89.
  32.  42.  58.  69.  54.  71.  74.  70.  65.  87.  56.  61.  73.  84.
  55.  44.  41.  85.  59.  17.  39.  80.  37.  63.  51.  49.  26.  68.
  45.  47.  86.  34.  79.  66.  67.  91. 100.  76.  77.  82.  92.  93.
  62.  75.  88.  50.  28. 212.  48.  46.  23.  38.  94.  97.  99.  31.
  96.  24.  29.  40.]
Column writing score : [ 74.  88.  nan  44.  75.  78.  39.  67.  50.  52.  43.  73.  70.
58.
  86.  28.  46.  61.  63.  53.  72.  55.  65.  38.  82.  79.  59.  54.
  68.  66.  57.  62.  76.  48.  42.  87.  49.  10.  34.  71.  37.  56.
  41.  22.  81.  45.  36.  92.  89.  47.  90. 100.  64.  98.  93.  51.
  40.  84.  69.  33.  60.  85.  91.  77. 234.  27.  94.  83.  80.  95.
  19.  35.  32.  96.  97.  99.  15.  30.  23.]
```

no. of unique values

In [12]:
```
for i in df_col:
    print("Column " + str(i) + " : " + str(df[i].nunique()))
```

```
Column gender : 2
Column race/ethnicity : 5
Column parental level of education : 6
Column lunch : 2
Column test preparation course : 2
Column math score : 83
Column reading score : 73
Column writing score : 78
```

1. Replacing missing values for scores with their respective mean values

In [13]:
```
df['math score'] = df['math score'].fillna(df['math score'].mean())
df['reading score'] = df['reading score'].fillna(df['reading score'].mean())
df['writing score'] = df['writing score'].fillna(df['writing score'].mean())
```

In [14]:
```
# Dropping missing values records
df.dropna(inplace = True)
```

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
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 [16]:
```
df.shape        # originally (1000,8)
```

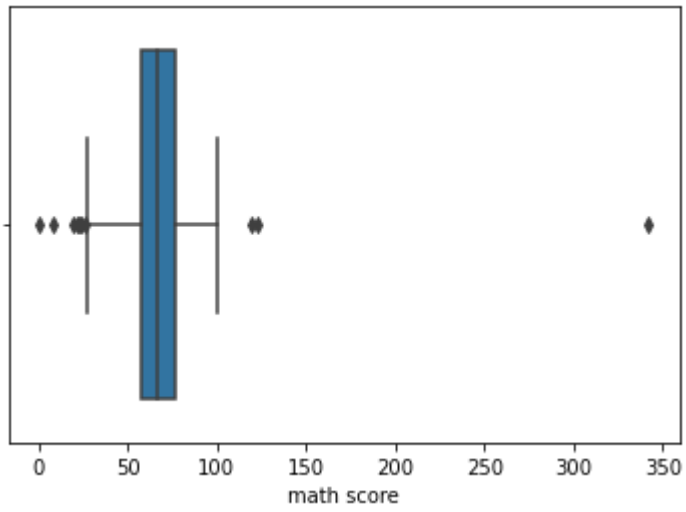
Out[16]:
```
(969, 8)
```

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

outliers are values within a dataset that vary greatly from the others

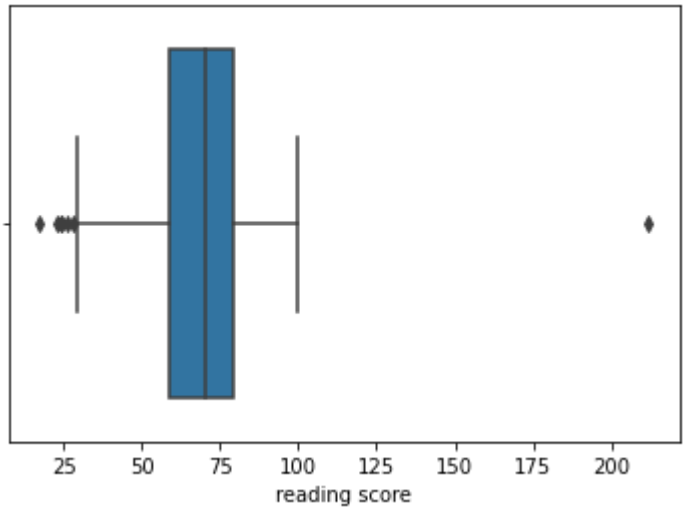
In [18]:
```
import seaborn as sb
sb.boxplot(x = df['math score'])
```

Out[18]:
```
<AxesSubplot:xlabel='math score'>
```



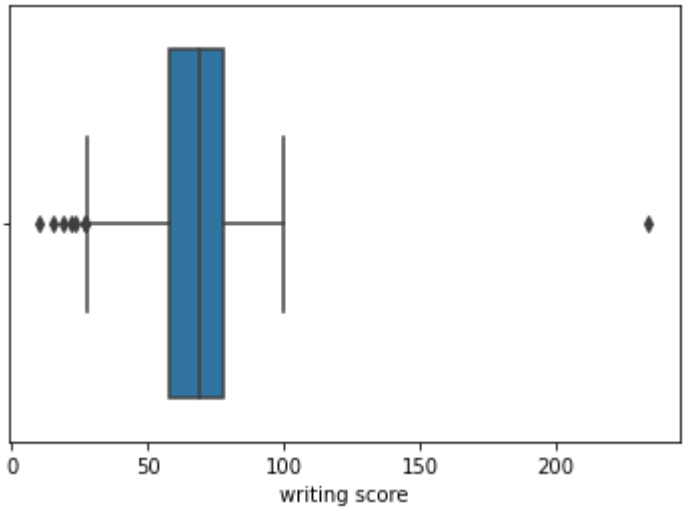
In [19]:
```
sb.boxplot(x = df['reading score'])
```

Out[19]:
```
<AxesSubplot:xlabel='reading score'>
```



In [20]:
```
sb.boxplot(x = df['writing score'])
```

Out[20]: <AxesSubplot:xlabel='writing score'>



Steps to perform Outlier Detection by identifying the lowerbound and upperbound of the data:

1. Arrange your data in ascending order
2. Calculate Q1 (the first Quartile) ==> 25%
3. Calculate Q3 (the third Quartile) ==> 75%
4. Find IQR = (Q3 - Q1)
5. Find the lower Range = Q1 -(1.5 * IQR)
6. Find the upper Range = Q3 + (1.5 * IQR)

Once you get the upperbound and lowerbound, all you have to do is to delete any values which is less than lowerbound or greater than upperbound.

In [21]:
```
# df2 will go throuh the outlier removing operations.
df2 = df.copy()
score_col = df2.select_dtypes(['float64']).columns
score_col
```

Out[21]:
```
Index(['math score', 'reading score', 'writing score'], dtype='object')
```

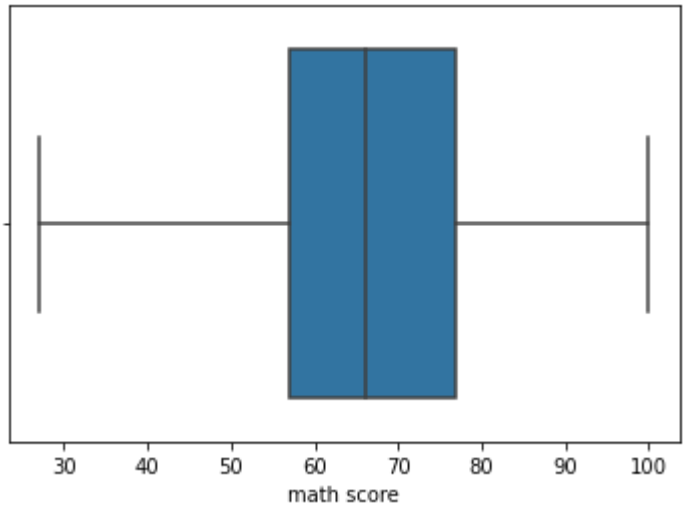
In [22]:
```
# iqr values
from scipy import stats
result_iqr = stats.iqr(df2[score_col], axis = 0)
result_iqr
```

Out[22]:
```
array([20., 20., 20.])
```

Dropping records with score less than lowerbound = Q1 - (1.5 *IQR) & greater than upperbound = Q3 + (1.5* IQR)

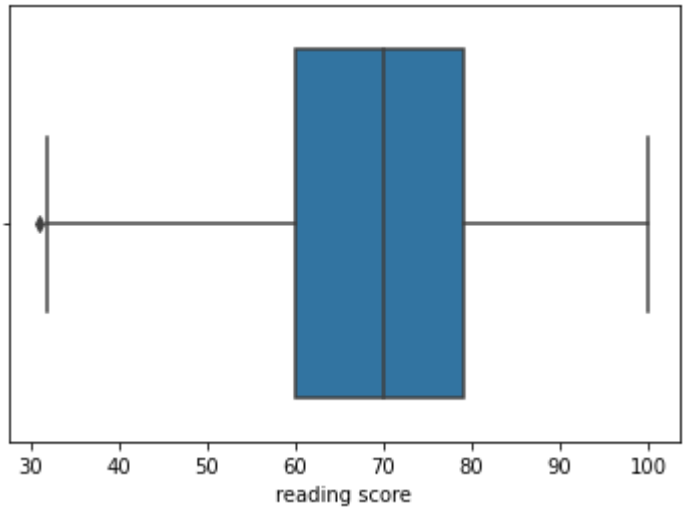
In [26]:
```
# for math score
df2.drop(df2[df2['math score'] < (df2['math score'].quantile(0.25) - (1.5 * result_iqr[
df2.drop(df2[df2['math score'] > (df2['math score'].quantile(0.75) + (1.5 * result_iqr[
sb.boxplot(x = df2['math score'])
```

Out[26]:
```
<AxesSubplot:xlabel='math score'>
```



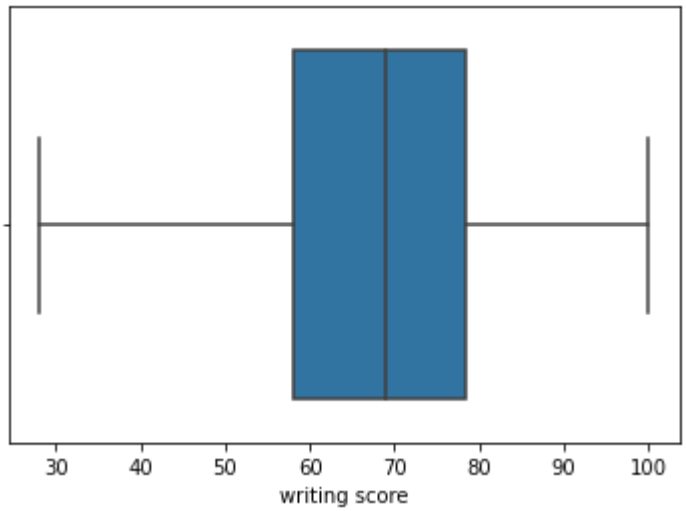
In [27]:
```
# for reading score
df2.drop(df2[df2['reading score'] < (df2['reading score'].quantile(0.25) - (1.5 * resul
df2.drop(df2[df2['reading score'] > (df2['reading score'].quantile(0.75) + (1.5 * resul
sb.boxplot(x = df2['reading score'])
```

Out[27]:
```
<AxesSubplot:xlabel='reading score'>
```



In [28]:
```
# for writing score
df2.drop(df2[df2['writing score'] < (df2['writing score'].quantile(0.25) - (1.5 * resul
df2.drop(df2[df2['writing score'] > (df2['writing score'].quantile(0.75) + (1.5 * resul
sb.boxplot(x = df2['writing score'])
```

Out[28]: <AxesSubplot:xlabel='writing score'>



In [29]:
```
df2.shape       # before = (969, 8)
```

Out[29]: (951, 8)

3. to change the scale for better understanding of the variable

In [38]:
```
df2.describe()
```

Out[38]:

	math score	reading score	writing score
count	951.000000	951.000000	951.000000
mean	66.645633	69.644388	68.496817
std	14.310691	13.874576	14.355463

	math score	reading score	writing score
min	27.000000	31.000000	28.000000
25%	57.000000	60.000000	58.000000
50%	66.000000	70.000000	69.000000
75%	77.000000	79.000000	78.500000
max	100.000000	100.000000	100.000000

In [43]:
```
df2_scale = df2.copy()
df2_feature = df2_scale[score_col]

from sklearn.preprocessing import MinMaxScaler
# x' = ((x - min) /(max - min))*(newmax - newmin) + newmin

scaling = MinMaxScaler(feature_range = (0, 10))                    # points out of 10
df2_scale[score_col] = scaling.fit_transform(df2_feature.values)
df2_scale.head()
```

Out[43]:

	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	5.410890	5.942029	6.388889
1	female	group C	some college	standard	completed	5.753425	8.550725	8.333333
2	female	group B	master's degree	standard	none	8.630137	9.275362	5.577471
4	male	group C	some college	standard	none	5.410890	6.811594	6.527778
5	female	group B	associate's degree	standard	none	6.027397	7.536232	6.944444

In [41]:
```
df2_scale.tail()
```

Out[41]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
995	female	group E	master's degree	standard	completed	8.356164	9.855072	9.305556
996	male	group C	high school	free/reduced	none	4.794521	3.478261	3.750000
997	female	group C	high school	free/reduced	completed	4.383562	5.797101	5.138889
998	female	group D	some college	standard	completed	5.616438	6.811594	6.805556
999	female	group D	some college	free/reduced	none	6.849315	7.971014	8.055556

In [ ]:
```

```