ASSIGNMENT 2

Data Wrangling 2

Create an "Academic Performance" dataset of students and perform the following operations using Python:

1. Scan all variables for missing values and inconsistencies. Deal with missing values if found.
2. Scan all numeric variables for outliers. If found, deal with them.
3. Apply data transformations on atleast one variable.

> Importing Required Libraries, Loading the dataset

In [13]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df = pd.read_csv("StudentsPerformance.csv")
```

In [3]:

```
df
```

Out[3]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writi sc
0	female	group B	bachelor's degree	standard	none	72	72	
1	female	group C	some college	standard	completed	69	90	
2	female	group B	master's degree	standard	none	90	95	
3	male	group A	associate's degree	free/reduced	none	47	57	
4	male	group C	some college	standard	none	76	78	
...	...	...	...	...	...	...	...	
995	female	group E	master's degree	standard	completed	88	99	
996	male	group C	high school	free/reduced	none	62	55	
997	female	group C	high school	free/reduced	completed	59	71	
998	female	group D	some college	standard	completed	68	78	
999	female	group D	some college	free/reduced	none	77	86	

1000 rows × 8 columns

> Data Preprocessing

In [4]:
```
#first 5 rows
df.head()
```

Out[4]:

	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 [5]:
```
#last 5 rows
df.tail()
```

Out[5]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writi sco
995	female	group E	master's degree	standard	completed	88	99	
996	male	group C	high school	free/reduced	none	62	55	
997	female	group C	high school	free/reduced	completed	59	71	
998	female	group D	some college	standard	completed	68	78	
999	female	group D	some college	free/reduced	none	77	86	

In [6]:
```
# checks total size(rows*columns)
df.size
```

Out[6]: 8000

In [7]:
```
# checks dimensions of dataframe
df.shape
```

Out[7]:
```
(1000, 8)
```

In [8]:
```
# prints information about the dataframe
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 [9]:
```
# checks initial statistics
df.describe()
```

Out[9]:

	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 [10]:
```
# checks the columns present
df.columns
```

Out[10]:
```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunc
h',
       'test preparation course', 'math score', 'reading score',
       'writing score'],
      dtype='object')
```

In [11]:
```
# checks datatypes of each column
df.dtypes
```

Out[11]:
```
gender                         object
race/ethnicity                 object
parental level of education    object
lunch                          object
test preparation course        object
math score                      int64
reading score                   int64
writing score                   int64
dtype: object
```

In [12]:
```
# checks for missing values
df.isnull().sum()
```

Out[12]:
```
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 this particular dataset, there are no null values. But incase there were, we would use replace(), interpolate() or fillna() to fill them, just like we did in assignment 1.

For example, if "math score" had null values, we would fill them by, df['math score'] = df['math score'].interpolate()

Scanning for outliers

When dealing with outliers, one common method is to use the Interquartile Range (IQR). The IQR is a measure of statistical dispersion that represents the range between the 25th and 75th percentiles of a dataset. Here's how you can use the IQR to identify and handle outliers:

To handle outliers using the IQR method:

1.Calculate the IQR (the range between the 25th and 75th percentiles).

2.Define a threshold for outliers (e.g., 1.5 times the IQR).

3.Identify potential outliers outside the threshold.

4.Choose a treatment: remove outliers, cap them (Winsorization), transform the data, or perform separate analyses:

Removal: You can choose to remove the outliers from your dataset. However, be cautious when removing outliers, as they may contain important information or represent rare events. Removing outliers can also affect the overall distribution and statistical properties of your data.

Winsorization: Instead of removing the outliers completely, you can cap them at a certain value. For example, you can replace values below Q1 - 1.5 * IQR with Q1 and values above Q3 + 1.5 * IQR with Q3.

Transformation: If the outliers are due to skewness or extreme values, you can consider transforming your data using techniques such as logarithmic transformation or Box-Cox transformation. These transformations can make the distribution more symmetric and reduce the influence of outliers.

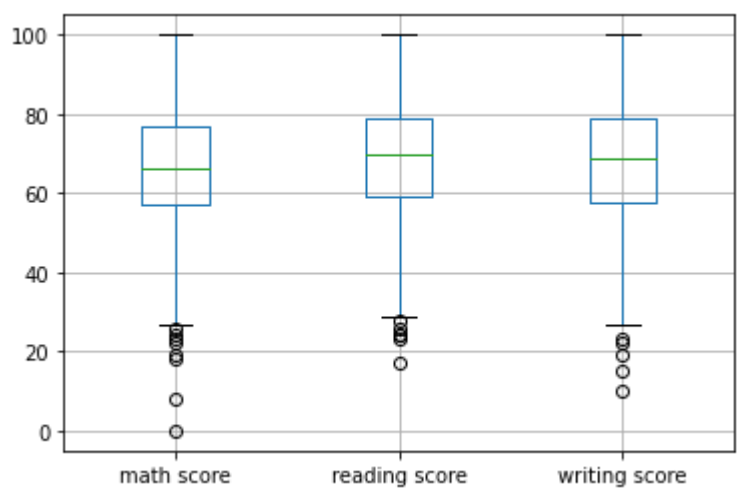
Separate analysis: In some cases, it may be appropriate to perform se

In [14]:
```
#Quartiles
q1 = df['math score'].quantile(0.25)
q3 = df['math score'].quantile(0.75)
q1,q3
```

Out[14]: (57.0, 77.0)

In [15]:
```
# Visual inspection for outliers
df.boxplot()
```

Out[15]: <AxesSubplot:>



The circles/dots that you see are the outliers.

> Dealing with outliers

We find the upper and lower limit of the interquartile range and read the dataset within the limit to avoid outliers.

In [16]:
```
#interquartile range
iqr=q3-q1
iqr
```

Out[16]: 20.0

In [17]:
```
#LIMITS
upper=q1-(1.5*iqr)
lower=q3+(1.5*iqr)
upper,lower
```

Out[17]: (27.0, 107.0)

In [21]:
```
# Reading dataset within the limit
final1=df[df['math score']< lower]
final2=df[df['math score']>upper]
```

In [22]:

```
final1
```

Out[22]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writi sc
0	female	group B	bachelor's degree	standard	none	72	72	
1	female	group C	some college	standard	completed	69	90	
2	female	group B	master's degree	standard	none	90	95	
3	male	group A	associate's degree	free/reduced	none	47	57	
4	male	group C	some college	standard	none	76	78	
...	...	...	...	...	...	...	...	
995	female	group E	master's degree	standard	completed	88	99	
996	male	group C	high school	free/reduced	none	62	55	
997	female	group C	high school	free/reduced	completed	59	71	
998	female	group D	some college	standard	completed	68	78	
999	female	group D	some college	free/reduced	none	77	86	

1000 rows × 8 columns

In [23]:
```
final2
```

Out[23]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writi sc
0	female	group B	bachelor's degree	standard	none	72	72	
1	female	group C	some college	standard	completed	69	90	
2	female	group B	master's degree	standard	none	90	95	
3	male	group A	associate's degree	free/reduced	none	47	57	
4	male	group C	some college	standard	none	76	78	
...	...	...	...	...	...	...	...	
995	female	group E	master's degree	standard	completed	88	99	
996	male	group C	high school	free/reduced	none	62	55	
997	female	group C	high school	free/reduced	completed	59	71	
998	female	group D	some college	standard	completed	68	78	
999	female	group D	some college	free/reduced	none	77	86	

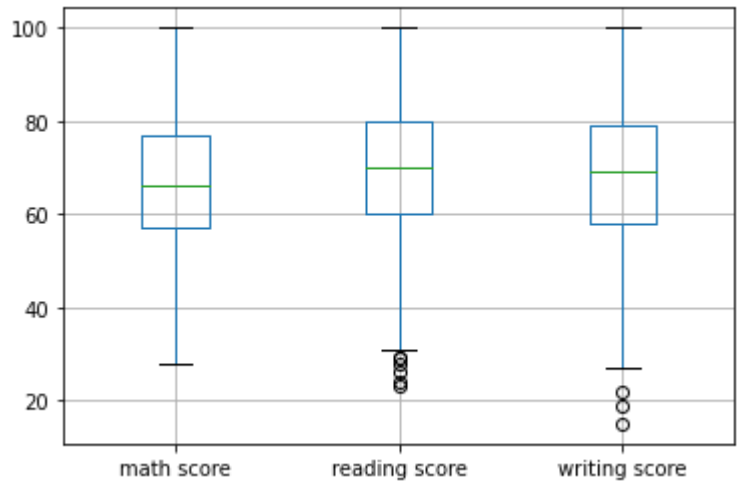
990 rows × 8 columns

In [25]:
```
final2["math score"].shape
```

Out[25]: (990,)

In [28]:
```
# Visual inspection for outliers
final2.boxplot()
```

Out[28]: <AxesSubplot:>

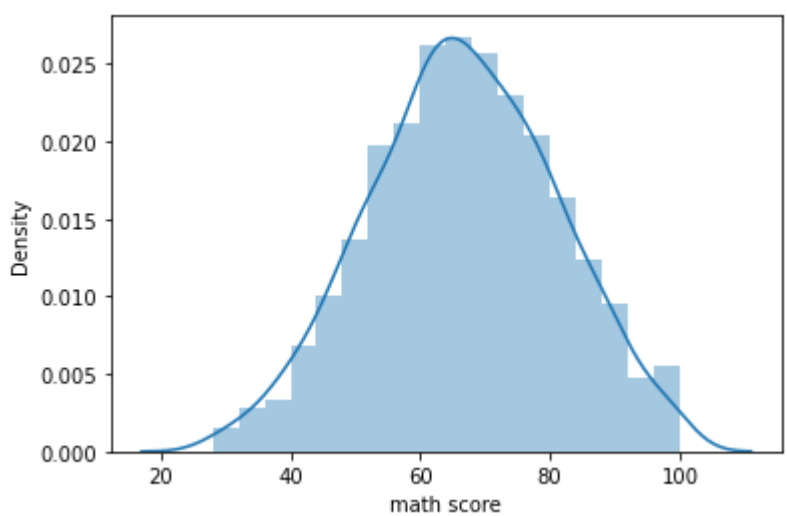


We scanned 'math score' for outliers and hence, as seen in the boxplot above, got rid of them.

In [27]:
```
sns.distplot(final2['math score'])
```

```
C:\Users\Shravani Sajekar\anaconda3\lib\site-packages\seaborn\distributi
ons.py:2551: FutureWarning: `distplot` is a deprecated function and will
be removed in a future version. Please adapt your code to use either `di
splot` (a figure-level function with similar flexibility) or `histplot`
(an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[27]: <AxesSubplot:xlabel='math score', ylabel='Density'>



Similarly, we can do for all numeric variables

> Data Transformations

In [29]:
```
df.dtypes
```

Out[29]:
```
gender                         object
race/ethnicity                 object
parental level of education    object
lunch                          object
test preparation course        object
math score                      int64
reading score                   int64
writing score                   int64
dtype: object
```

Let's convert 'int' to 'float'

In [30]:
```
df['reading score'] = df['reading score'].astype('float64')
```

In [31]:
```
df['reading score']
```

Out[31]:
```
0      72.0
1      90.0
2      95.0
3      57.0
4      78.0
       ...
995    99.0
996    55.0
997    71.0
998    78.0
999    86.0
Name: reading score, Length: 1000, dtype: float64
```

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

Out[32]:
```
gender                         object
race/ethnicity                 object
parental level of education    object
lunch                          object
test preparation course        object
math score                      int64
reading score                 float64
writing score                   int64
dtype: object
```

Similarly, we can convert other variables using 'astype'.