In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
df = pd.read_csv("C:/Users/ameya/OneDrive/Desktop/DSBDAL/UpdatedStudentsPerformance.csv"
```

In [3]:

```
df.head()
```

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	72.0	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	93.0
3	male	group A	associate's degree	free/reduced	none	47.0	57.0	44.0
4	male	group C	some college	standard	none	76.0	78.0	75.0

In [5]:

```
df.describe()
```

Out[5]:

	math score	reading score	writing score
count	990.000000	985.000000	989.000000
mean	66.208081	69.261929	68.142568
std	15.103724	14.634171	15.199780
min	0.000000	17.000000	10.000000
25%	57.000000	59.000000	58.000000
50%	66.000000	70.000000	69.000000
75%	77.000000	80.000000	79.000000
max	100.000000	100.000000	100.000000

In [16]:

```
df.isnull().sum()
```

Out[16]:

```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                    10
reading score                 15
writing score                 11
dtype: int64
```

In [17]:

```
df['math score'].fillna(df['math score'].mean(),inplace=True)
```

In [18]:

```
df['reading score'].fillna(df['reading score'].mean(),inplace=True)
```

In [19]:

```
df['writing score'].fillna(df['writing score'].mean(),inplace=True)
```

In [20]:

```
df.isnull().sum()
```

Out[20]:

```
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
```

A- Zscore

In [21]:

```
col = ["math score","reading score","writing score"]
```

In [22]:

```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
df_scaled = scalar.fit_transform(df[col].to_numpy())
df_scaled = pd.DataFrame(df_scaled,columns=col)
```

In [23]:

```
df_scaled
```

Out[23]:

	math score	reading score	writing score
0	0.385603	0.188616	0.387696
1	0.185875	1.428574	1.314339
2	1.583971	1.773006	1.645283
3	-1.278798	-0.844682	-1.597967
4	0.651907	0.601935	0.453885
...	...	...	...
995	1.450819	2.048553	1.777660
996	-0.280157	-0.982455	-0.869890
997	-0.479885	0.119730	-0.208003
998	0.119299	0.601935	0.586262
999	0.718483	1.153028	1.181961

1000 rows × 3 columns

Min Max

In [24]:

```
from sklearn.preprocessing import MinMaxScaler
scalar = MinMaxScaler()
df_scaled = scalar.fit_transform(df[col].to_numpy())
df_scaled = pd.DataFrame(df_scaled,columns=col)
print("Scaled Dataset using MinMaxScaler")
df_scaled.head()
```

```
Scaled Dataset using MinMaxScaler
```

Out[24]:

	math score	reading score	writing score
0	0.72	0.662651	0.711111
1	0.69	0.879518	0.866667
2	0.90	0.939759	0.922222
3	0.47	0.481928	0.377778
4	0.76	0.734940	0.722222

Box-Plot

In [25]:

```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [26]:

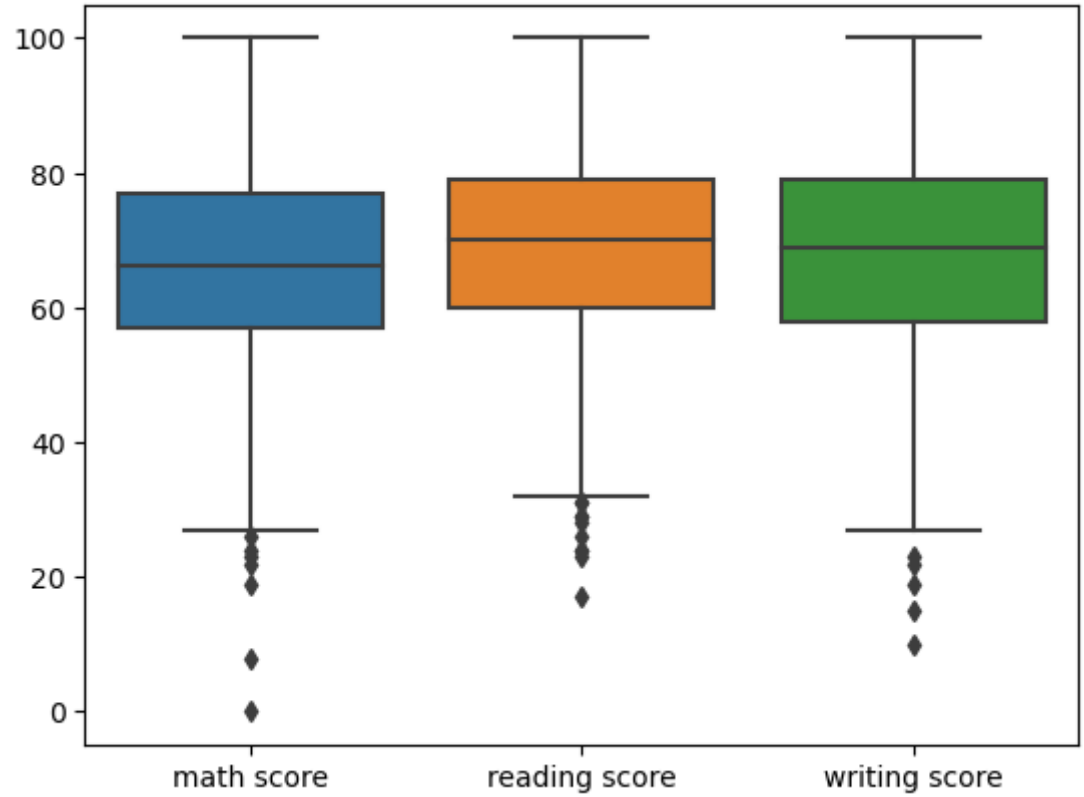
```
plt.figsize = (8,16)
```

In [27]:

```
sns.boxplot(data =df)
```

Out[27]:

```
<AxesSubplot:>
```



In [28]:

```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
```

In [29]:

```
print(IQR)
```

```
math score       20.0
reading score    19.0
writing score    21.0
dtype: float64
```

In [30]:

```
low = Q1 - 1.5*IQR
high = Q3 + 1.5*IQR
```

In [31]:

```
print(low,high)
```

```
math score       27.0
reading score    31.5
writing score    26.5
dtype: float64 math score       107.0
reading score    107.5
writing score    110.5
dtype: float64
```

In [34]:

```
df_2 = df[~((df< low) | (df>high)).any(axis=1)]
```

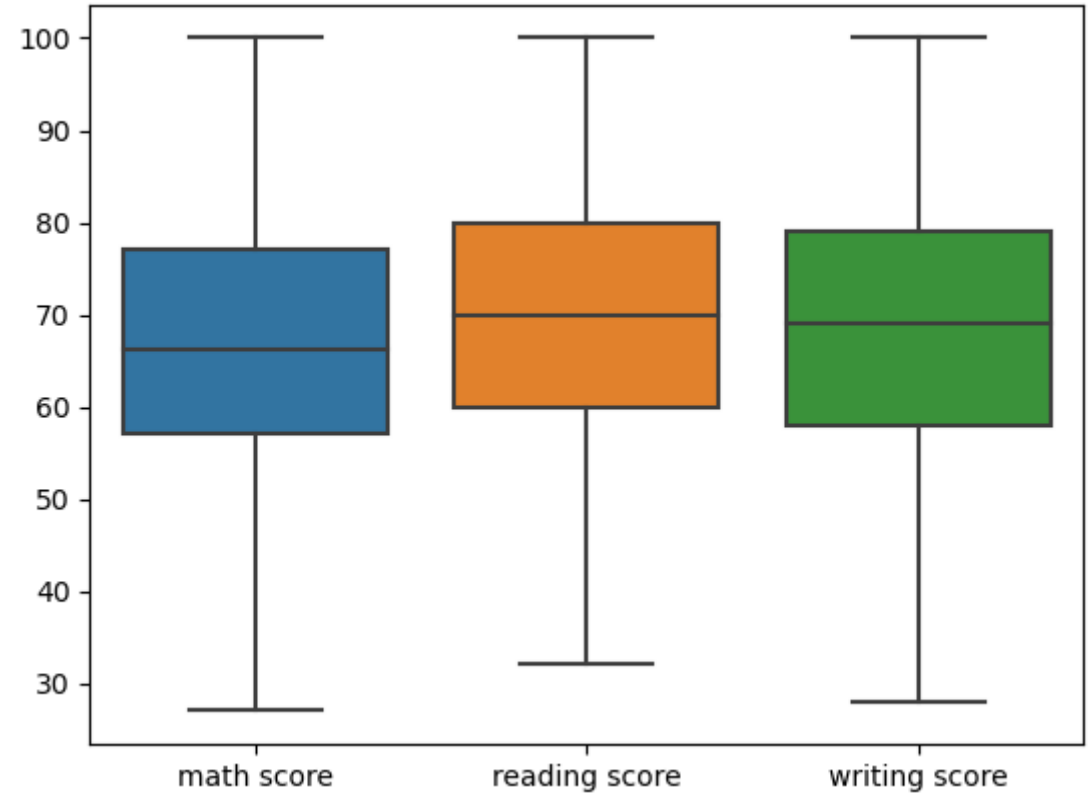
```
C:\Users\ameya\AppData\Local\Temp\ipykernel_3548\2007515236.py:1: FutureW
arning: Automatic reindexing on DataFrame vs Series comparisons is deprec
ated and will raise ValueError in a future version. Do `left, right = lef
t.align(right, axis=1, copy=False)` before e.g. `left == right`
  df_2 = df[~((df< low) | (df>high)).any(axis=1)]
```

In [35]:

```
sns.boxplot(data = df_2)
```

Out[35]:

```
<AxesSubplot:>
```



In [ ]:

```
```