In [47]:
```
import pandas as pd
import numpy as np

import fuzzywuzzy
from fuzzywuzzy import process
import chardet

import missingno as msno
from sklearn import preprocessing
import seaborn as sns
import matplotlib.pyplot as plt
```

In [28]:
```
df=pd.read_csv("UpdatedStudentsPerformance.csv")
```

In [29]:
```
df.head()
```

Out[29]:

	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 [30]:
```
df.isnull().sum()
```

Out[30]:
```
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 [31]:
```
gender = df['gender'].unique()

# sort them alphabetically and then take a closer look
gender.sort()
gender
```

Out[31]:
```
array(['f', 'female', 'male'], dtype=object)
```

In [32]:
```
maths = df['math score'].unique()

# sort them alphabetically and then take a closer look
maths.sort()
maths
```

Out[32]:
```
array([  0.,   8.,  19.,  22.,  23.,  24.,  26.,  27.,  28.,  29.,  30.,
        32.,  33.,  34.,  35.,  36.,  37.,  38.,  39.,  40.,  41.,  42.,
        43.,  44.,  45.,  46.,  47.,  48.,  49.,  50.,  51.,  52.,  53.,
        54.,  55.,  56.,  57.,  58.,  59.,  60.,  61.,  62.,  63.,  64.,
        65.,  66.,  67.,  68.,  69.,  70.,  71.,  72.,  73.,  74.,  75.,
        76.,  77.,  78.,  79.,  80.,  81.,  82.,  83.,  84.,  85.,  86.,
        87.,  88.,  89.,  90.,  91.,  92.,  93.,  94.,  95.,  96.,  97.,
        98.,  99., 100.,  nan])
```

In [33]:
```
reading = df['reading score'].unique()

# sort them alphabetically and then take a closer look
reading.sort()
reading
```

Out[33]:
```
array([ 17.,  23.,  24.,  26.,  28.,  29.,  31.,  32.,  34.,  37.,  38.,
        39.,  40.,  41.,  42.,  43.,  44.,  45.,  46.,  47.,  48.,  49.,
        50.,  51.,  52.,  53.,  54.,  55.,  56.,  57.,  58.,  59.,  60.,
        61.,  62.,  63.,  64.,  65.,  66.,  67.,  68.,  69.,  70.,  71.,
        72.,  73.,  74.,  75.,  76.,  77.,  78.,  79.,  80.,  81.,  82.,
        83.,  84.,  85.,  86.,  87.,  88.,  89.,  90.,  91.,  92.,  93.,
        94.,  95.,  96.,  97.,  99., 100.,  nan])
```

In [34]:
```
writing = df['writing score'].unique()

# sort them alphabetically and then take a closer look
writing.sort()
writing
```

Out[34]:
```
array([ 10.,  15.,  19.,  22.,  23.,  27.,  28.,  30.,  32.,  33.,  34.,
        35.,  36.,  37.,  38.,  39.,  40.,  41.,  42.,  43.,  44.,  45.,
        46.,  47.,  48.,  49.,  50.,  51.,  52.,  53.,  54.,  55.,  56.,
        57.,  58.,  59.,  60.,  61.,  62.,  63.,  64.,  65.,  66.,  67.,
        68.,  69.,  70.,  71.,  72.,  73.,  74.,  75.,  76.,  77.,  78.,
        79.,  80.,  81.,  82.,  83.,  84.,  85.,  86.,  87.,  88.,  89.,
        90.,  91.,  92.,  93.,  94.,  95.,  96.,  97.,  98.,  99., 100.,
        nan])
```

In [35]:
```
matches = fuzzywuzzy.process.extract("f", gender, limit=10, scorer=fuzzywuzzy.fuzz.token_sort_ratio)

# take a look at them
matches
```

Out[35]:
```
[('f', 100), ('female', 29), ('male', 0)]
```

In [36]:
```
# function to replace rows in the provided column of the provided dataframe
# that match the provided string above the provided ratio with the provided string
def replace_matches_in_column(df, column, string_to_match, min_ratio = 29):
    # get a list of unique strings
    strings = df[column].unique()

    # get the top 10 closest matches to our input string
    matches = fuzzywuzzy.process.extract(string_to_match, strings,
                                         limit=10, scorer=fuzzywuzzy.fuzz.token_sort_ratio)

    # only get matches with a ratio > 90
    close_matches = [matches[0] for matches in matches if matches[1] == min_ratio]

    # get the rows of all the close matches in our dataframe
    rows_with_matches = df[column].isin(close_matches)

    # replace all rows with close matches with the input matches
    df.loc[rows_with_matches, column] = string_to_match

    # let us know the function's done
    print("All done!")
```

In [37]:
```
# use the function we just wrote to replace close matches to "south korea" with "south korea"
replace_matches_in_column(df, column='gender', string_to_match="female")
```

```
All done!
```

In [38]:
```
gender = df['gender'].unique()

# sort them alphabetically and then take a closer look
gender.sort()
gender
```

Out[38]:
```
array(['female', 'male'], dtype=object)
```

In [39]:
```
df.head()
```

Out[39]:

	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 [40]:
```
col=["math score","reading score", "writing score"]
```

In [41]:
```
for i in col:
    df[i].fillna(value=df[i].mean(), inplace=True)
```

In [42]:
```
df.isnull().sum()
```

Out[42]:
```
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 [43]:
```
df.head(30)
```

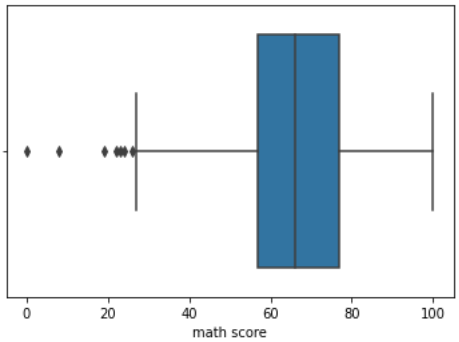
Out[43]:

	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	72.000000	72.000000	74.000000
1	female	group C	some college	standard	completed	69.000000	90.000000	88.000000
2	female	group B	master's degree	standard	none	90.000000	95.000000	93.000000
3	male	group A	associate's degree	free/reduced	none	47.000000	57.000000	44.000000
4	male	group C	some college	standard	none	76.000000	78.000000	75.000000
5	female	group B	associate's degree	standard	none	71.000000	83.000000	78.000000
6	female	group B	some college	standard	completed	88.000000	95.000000	92.000000
7	male	group B	some college	free/reduced	none	40.000000	43.000000	39.000000
8	male	group D	high school	free/reduced	completed	64.000000	64.000000	67.000000
9	female	group B	high school	free/reduced	none	38.000000	60.000000	50.000000
10	male	group C	associate's degree	standard	none	58.000000	54.000000	52.000000
11	male	group D	associate's degree	standard	none	66.208081	52.000000	43.000000
12	female	group B	high school	standard	none	65.000000	81.000000	73.000000
13	male	group A	some college	standard	completed	78.000000	72.000000	70.000000
14	female	group A	master's degree	standard	none	50.000000	53.000000	58.000000
15	female	group C	some high school	standard	none	69.000000	75.000000	78.000000
16	male	group C	high school	standard	none	88.000000	89.000000	86.000000
17	female	group B	some high school	free/reduced	none	66.208081	32.000000	28.000000
18	male	group C	master's degree	free/reduced	completed	46.000000	42.000000	46.000000
19	female	group C	associate's degree	free/reduced	none	54.000000	58.000000	61.000000
20	male	group D	high school	standard	none	66.000000	69.000000	63.000000
21	female	group B	some college	free/reduced	completed	66.208081	69.261929	70.000000
22	male	group D	some college	standard	none	66.208081	69.261929	53.000000
23	female	group C	some high school	standard	none	66.208081	69.261929	73.000000
24	male	group D	bachelor's degree	free/reduced	completed	66.208081	69.261929	80.000000
25	male	group A	master's degree	free/reduced	none	73.000000	74.000000	72.000000
26	male	group B	some college	standard	none	69.000000	54.000000	55.000000
27	female	group C	bachelor's degree	standard	none	67.000000	69.261929	68.142568
28	male	group C	high school	standard	none	70.000000	69.261929	68.142568
29	female	group D	master's degree	standard	none	62.000000	69.261929	68.142568

In [45]:
```
import seaborn as sns
sns.boxplot(df['math score'])
```

```
/usr/local/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

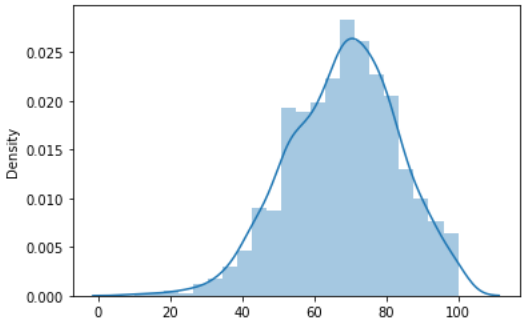
Out[45]: <AxesSubplot:xlabel='math score'>



In [49]:
```
sns.distplot(df[["writing score"]])
```

```
/usr/local/lib/python3.8/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated fu
nction and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

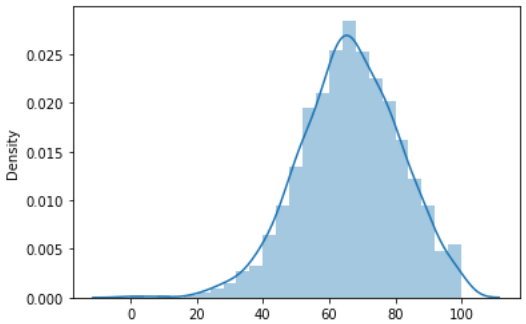
Out[49]: <AxesSubplot:ylabel='Density'>



In [50]:
```
sns.distplot(df[["math score"]])
```

```
/usr/local/lib/python3.8/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated fu
nction and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

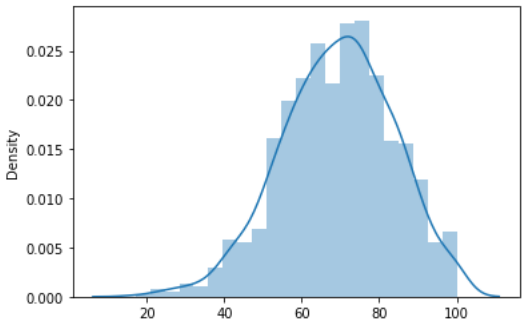
Out[50]: <AxesSubplot:ylabel='Density'>



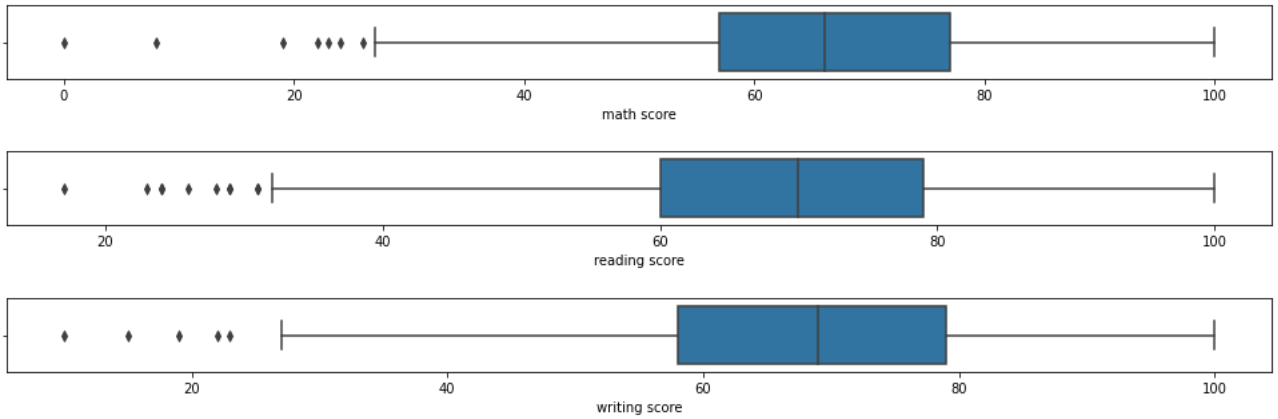
In [51]:
```
sns.distplot(df[["reading score"]])
```

```
/usr/local/lib/python3.8/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated fu
nction and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level fun
ction with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[51]: <AxesSubplot:ylabel='Density'>



In [55]:
```
for column in col:
    plt.figure(figsize=(17,1))
    sns.boxplot(data=df, x=column)
```



In [56]:
```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

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

In [58]:
```
print(IQR)
```

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

In [59]:
```
low = Q1 - 1.5 * IQR
high = Q3+1.5*IQR
print(low,high)
```

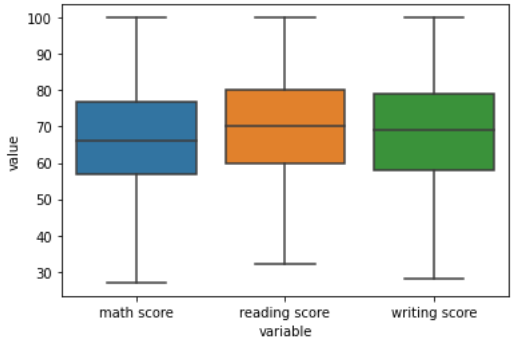
```
math score       27.0
reading score    31.5
writing score    26.5
dtype: float64 math score       107.0
reading score    107.5
writing score    110.5
dtype: float64
```

In [60]:
```
# IQR Score -
# we can use previously calculated IQR score to filter out the outliers
df_2 = df[~((df < low) |(df > high)).any(axis=1)]
print(df.shape,df_2.shape)
```

```
(1000, 8) (986, 8)

/tmp/ipykernel_14211/1102832448.py:3: FutureWarning: Automatic reindexing on DataFrame vs Series comparisons is de
precated and will raise ValueError in a future version. Do `left, right = left.align(right, axis=1, copy=False)` b
efore e.g. `left == right`
  df_2 = df[~((df < low) |(df > high)).any(axis=1)]
```

In [63]:
```
plt.figsize=(8,16)
sns.boxplot(x="variable", y="value", data=pd.melt(df_2[col]))
plt.show()
```



In [64]:
```
from sklearn.preprocessing import MinMaxScaler
df_min_max_scaled = df.copy()
# apply normalization techniques
scaler = MinMaxScaler()
df_scaled = scaler.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[64]:

	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

In [ ]: