In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
data = pd.read_csv(r'StudentsPerformance.csv')
```

In [4]:

```
data
```

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
...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95
996	male	group C	high school	free/reduced	none	62	55	55
997	female	group C	high school	free/reduced	completed	59	71	65
998	female	group D	some college	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

In [5]:

```
# top five rows
data.head()
```

Out[5]:

	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 [6]:

```
# to check data information
data.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 [7]:

```
data['gender'].dtypes
```

Out[7]:

```
dtype('O')
```

In [8]:

```
data['gender'].dtypes=='O'
```

Out[8]:

```
True
```

In [9]:

```
data.columns
```

Out[9]:

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

In [10]:

```
# iterate through columns
[i for i in data.columns]
```

Out[10]:

```
['gender',
 'race/ethnicity',
 'parental level of education',
 'lunch',
 'test preparation course',
 'math score',
 'reading score',
 'writing score']
```

In [11]:

```
# pull out categorical columns
cat_col = [i for i in data.columns if data[i].dtype == 'O']
cat_col
```

Out[11]:

```
['gender',
 'race/ethnicity',
 'parental level of education',
 'lunch',
 'test preparation course']
```

In [12]:

```
# pull out numeraical columns
num_col = [i for i in data.columns if data[i].dtype != 'O']
num_col
```

Out[12]:

```
['math score', 'reading score', 'writing score']
```

In [13]:

```
data[num_col]
```

Out[13]:

	math score	reading score	writing score
0	72	72	74
1	69	90	88
2	90	95	93
3	47	57	44
4	76	78	75
...	...	...	...
995	88	99	95
996	62	55	55
997	59	71	65
998	68	78	77
999	77	86	86

1000 rows × 3 columns

In [14]:

```
data[cat_col]
```

Out[14]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course
0	female	group B	bachelor's degree	standard	none
1	female	group C	some college	standard	completed
2	female	group B	master's degree	standard	none
3	male	group A	associate's degree	free/reduced	none
4	male	group C	some college	standard	none
...	...	...	...	...	...
995	female	group E	master's degree	standard	completed
996	male	group C	high school	free/reduced	none
997	female	group C	high school	free/reduced	completed
998	female	group D	some college	standard	completed
999	female	group D	some college	free/reduced	none

1000 rows × 5 columns

In [15]:

```
# to check memory usage of this dataset
data.memory_usage()
```

Out[15]:

```
Index                           128
gender                         8000
race/ethnicity                 8000
parental level of education    8000
lunch                          8000
test preparation course        8000
math score                     8000
reading score                  8000
writing score                  8000
dtype: int64
```

Missing value

In [16]:

```
data.isnull()
```

Out[16]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	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 [17]:

```
# sum of individual column
data.isnull().sum()
```

Out[17]:

```
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 [18]:

```
# sum of all
data.isnull().sum().sum()
```

Out[18]:

```
0
```

In [19]:

```
# to check duplicate data
data.duplicated()
```

Out[19]:

```
0      False
1      False
2      False
3      False
4      False
       ...  
995    False
996    False
997    False
998    False
999    False
Length: 1000, dtype: bool
```

In [20]:

```
# to check sum of duplicae values
data.duplicated().sum()
```

Out[20]:

```
0
```

In [21]:

```
# to check unique value column wise
data.nunique()
```

Out[21]:

```
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 [22]:

```
# sum of unique values
data.nunique().sum()
```

Out[22]:

```
247
```

In [23]:

```
# to check unique value of a column
data['gender'].unique()
```

Out[23]:

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

In [24]:

```
# to check unique value of a column
data['race/ethnicity'].unique()
```

Out[24]:

```
array(['group B', 'group C', 'group A', 'group D', 'group E'],
      dtype=object)
```

In [25]:

```
# statistic of the dataset
data.describe().T
```

Out[25]:

	count	mean	std	min	25%	50%	75%	max
math score	1000.0	66.089	15.163080	0.0	57.00	66.0	77.0	100.0
reading score	1000.0	69.169	14.600192	17.0	59.00	70.0	79.0	100.0
writing score	1000.0	68.054	15.195657	10.0	57.75	69.0	79.0	100.0

In [26]:

```
# to check correlation
data.corr()
```

Out[26]:

	math score	reading score	writing score
math score	1.000000	0.817580	0.802642
reading score	0.817580	1.000000	0.954598
writing score	0.802642	0.954598	1.000000

In [27]:

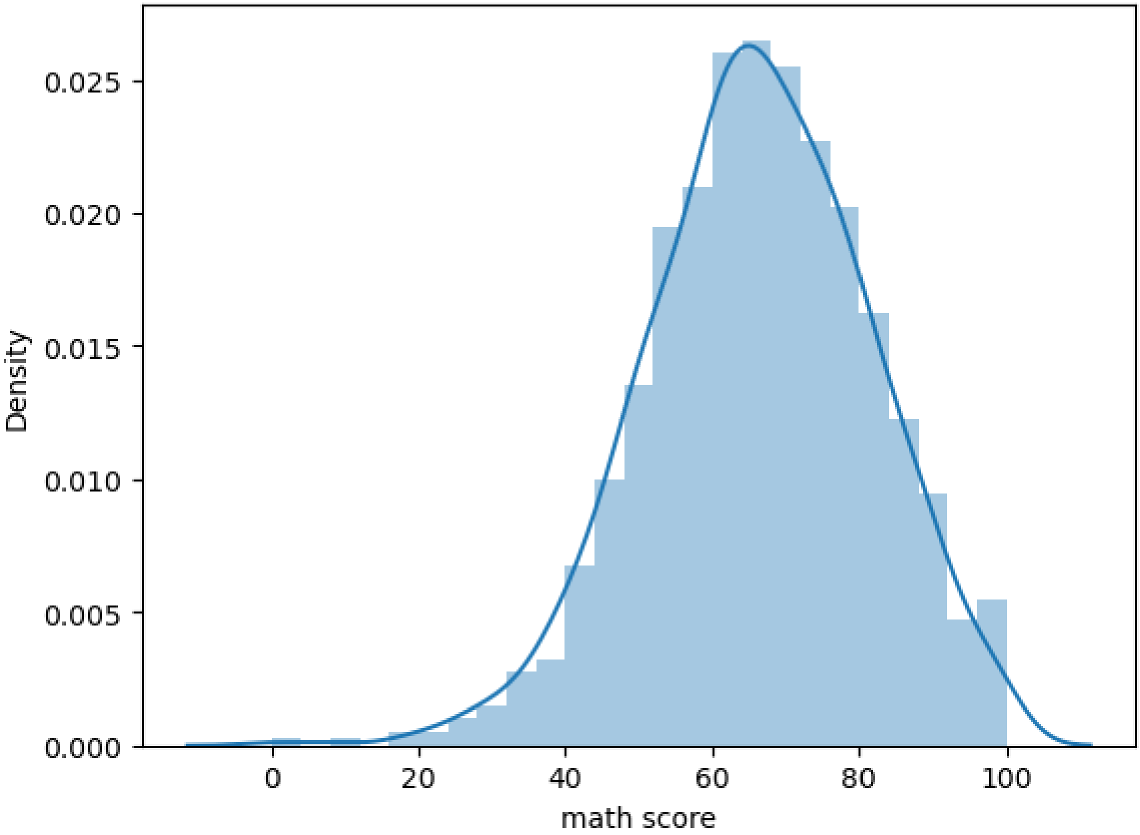
```
# to check skewness of the data
data.skew()
```

Out[27]:

```
math score      -0.278935
reading score   -0.259105
writing score   -0.289444
dtype: float64
```

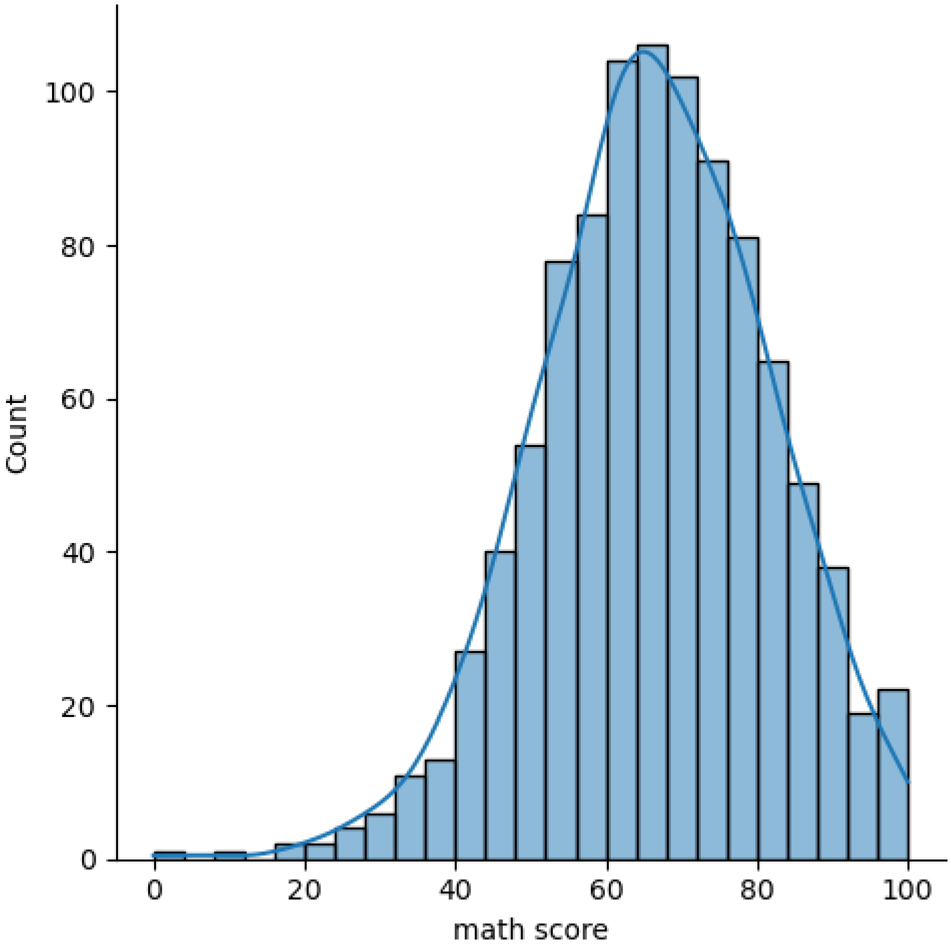
In [28]:
```
#to check distribution plot for a lleft skewed column
sns.distplot(data['math score'])
```

Out[28]:
```
<AxesSubplot:xlabel='math score', ylabel='Density'>
```



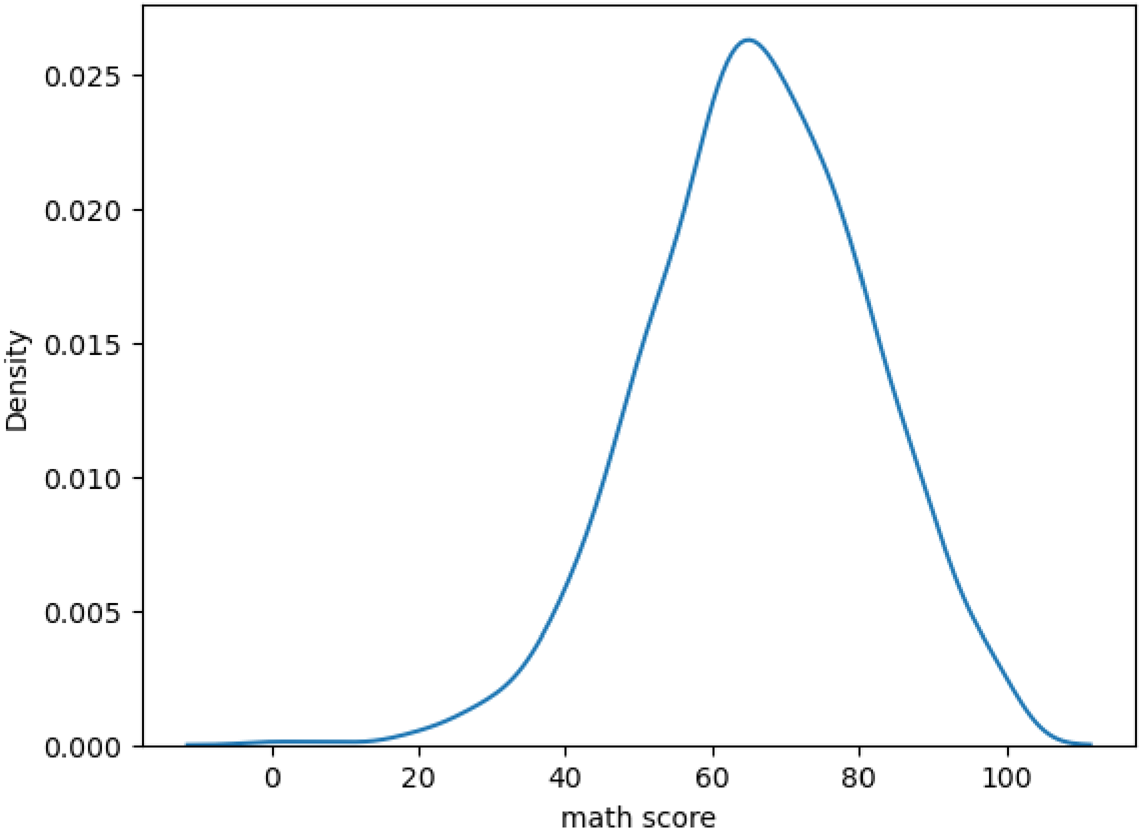
In [29]:
```
sns.displot(data['math score'],kde=True)
```

Out[29]:
```
<seaborn.axisgrid.FacetGrid at 0x20dc1af6ee0>
```



In [30]:
```
sns.kdeplot(data['math score'])
```

Out[30]:
```
<AxesSubplot:xlabel='math score', ylabel='Density'>
```



In [31]:
```
data
```

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
...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95
996	male	group C	high school	free/reduced	none	62	55	55
997	female	group C	high school	free/reduced	completed	59	71	65
998	female	group D	some college	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

In [32]:
```
data.columns
```

Out[32]:
```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course', 'math score', 'reading score',
       'writing score'],
      dtype='object')
```

In [33]:
```
data['math score'] + data['reading score'] + data['writing score']
```

Out[33]:
```
0      218
1      247
2      278
3      148
4      229
      ...
995    282
996    172
997    195
998    223
999    249
Length: 1000, dtype: int64
```

In [38]:
```
# adding a new column 'average'
data['average'] = (data['math score'] + data['reading score'] + data['writing scor
```

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

Out[39]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	average
0	female	group B	bachelor's degree	standard	none	72	72	74	72.666667
1	female	group C	some college	standard	completed	69	90	88	82.333333
2	female	group B	master's degree	standard	none	90	95	93	92.666667
3	male	group A	associate's degree	free/reduced	none	47	57	44	49.333333
4	male	group C	some college	standard	none	76	78	75	76.333333

In [41]:
```
data.groupby('gender').mean()
```

Out[41]:

	math score	reading score	writing score	average
gender				
female	63.633205	72.608108	72.467181	69.569498
male	68.728216	65.473029	63.311203	65.837483

In [42]:
```
data.groupby('gender').count()
```

Out[42]:

	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	average
gender								
female	518	518	518	518	518	518	518	518
male	482	482	482	482	482	482	482	482

In [48]:
```
#who get less than 30 in math
data[data['math score']<30]
```

Out[48]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	avera
17	female	group B	some high school	free/reduced	none	18	32	28	26.0000
59	female	group C	some high school	free/reduced	none	0	17	10	9.0000
91	male	group C	high school	free/reduced	none	27	34	36	32.3333
145	female	group C	some college	free/reduced	none	22	39	33	31.3333
327	male	group A	some college	free/reduced	none	28	23	19	23.3333
338	female	group B	some high school	free/reduced	none	24	38	27	29.6666
363	female	group D	some high school	free/reduced	none	27	34	32	31.0000
466	female	group D	associate's degree	free/reduced	none	26	31	38	31.6666
528	female	group D	bachelor's degree	free/reduced	none	29	41	47	39.0000
601	female	group C	high school	standard	none	29	29	30	29.3333
683	female	group C	some high school	free/reduced	completed	29	40	44	37.6666
787	female	group B	some college	standard	none	19	38	32	29.6666
842	female	group B	high school	free/reduced	completed	23	44	36	34.3333
980	female	group B	high school	free/reduced	none	8	24	23	18.3333

In [49]:

```
data[data['math score']<30].count()
```

Out[49]:

```
gender                         14
race/ethnicity                 14
parental level of education    14
lunch                          14
test preparation course        14
math score                     14
reading score                  14
writing score                  14
average                        14
dtype: int64
```

In [51]:

```
data_num = data[num_col]
data_num
```

Out[51]:

	math score	reading score	writing score
0	72	72	74
1	69	90	88
2	90	95	93
3	47	57	44
4	76	78	75
...	...	...	...
995	88	99	95
996	62	55	55
997	59	71	65
998	68	78	77
999	77	86	86

1000 rows × 3 columns

In [52]:

```
from scipy.stats import normaltest
```

In [55]:

```
normaltest(data_num['math score'])
```

Out[55]:

```
NormaltestResult(statistic=15.408960513931822, pvalue=0.00045080293869937836)
```

In [58]:

```
normaltest(data_num['math score'])[1]*100
```

Out[58]:

```
0.045080293869937844
```

In [ ]:

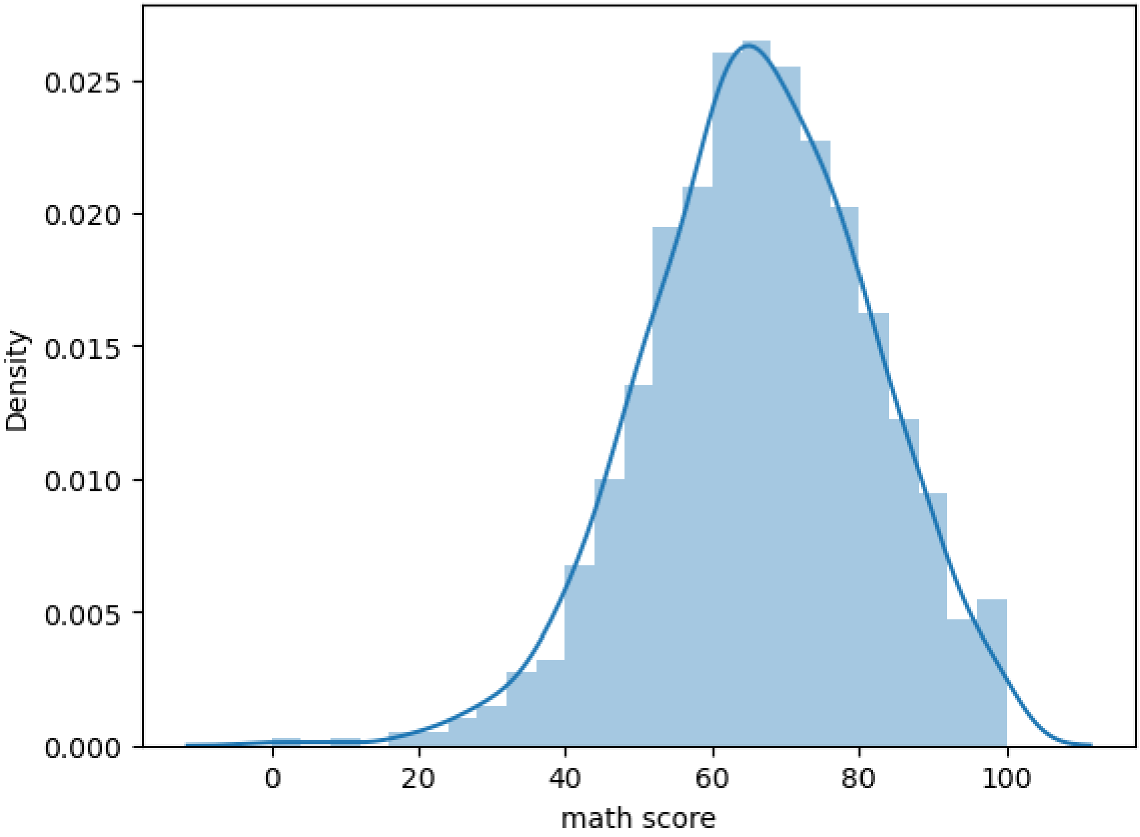
```
# if p value > 0.05 then my data will be normally distributed
# if p value < 0.05 then data  will be non normally distributed
```

In [61]:

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

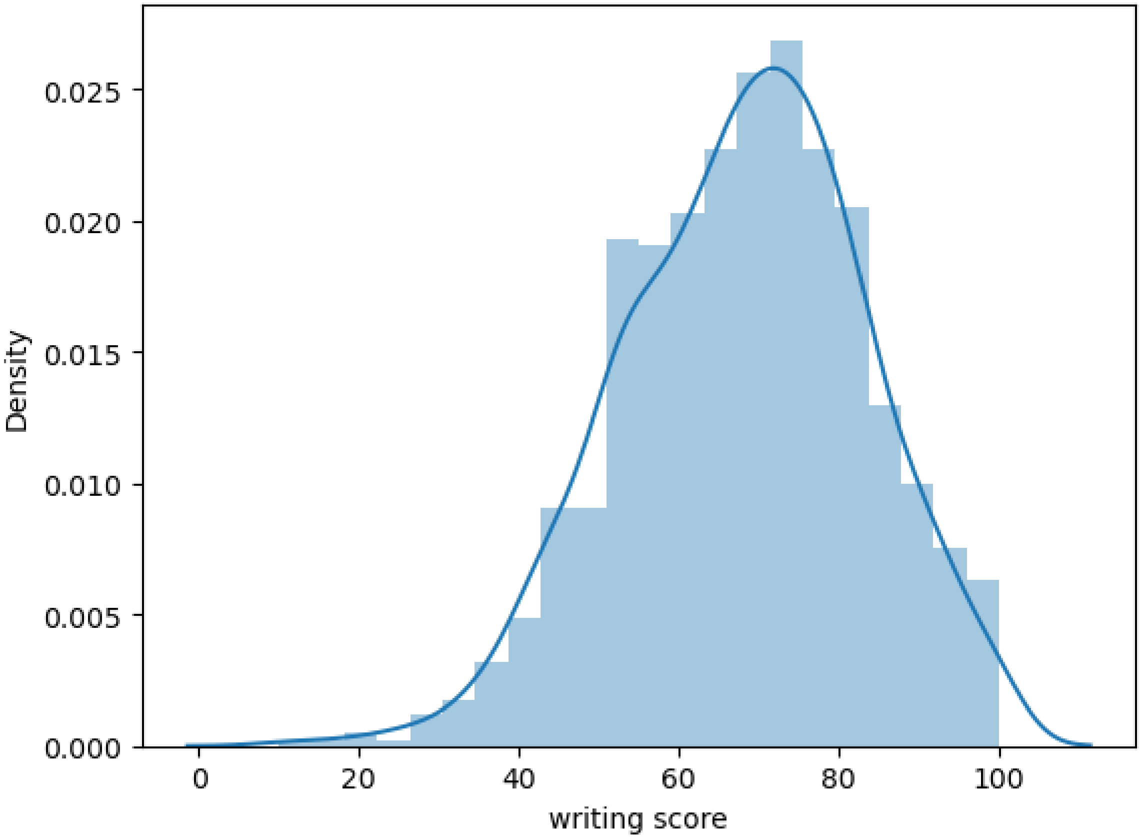
Out[61]:

```
<AxesSubplot:xlabel='math score', ylabel='Density'>
```



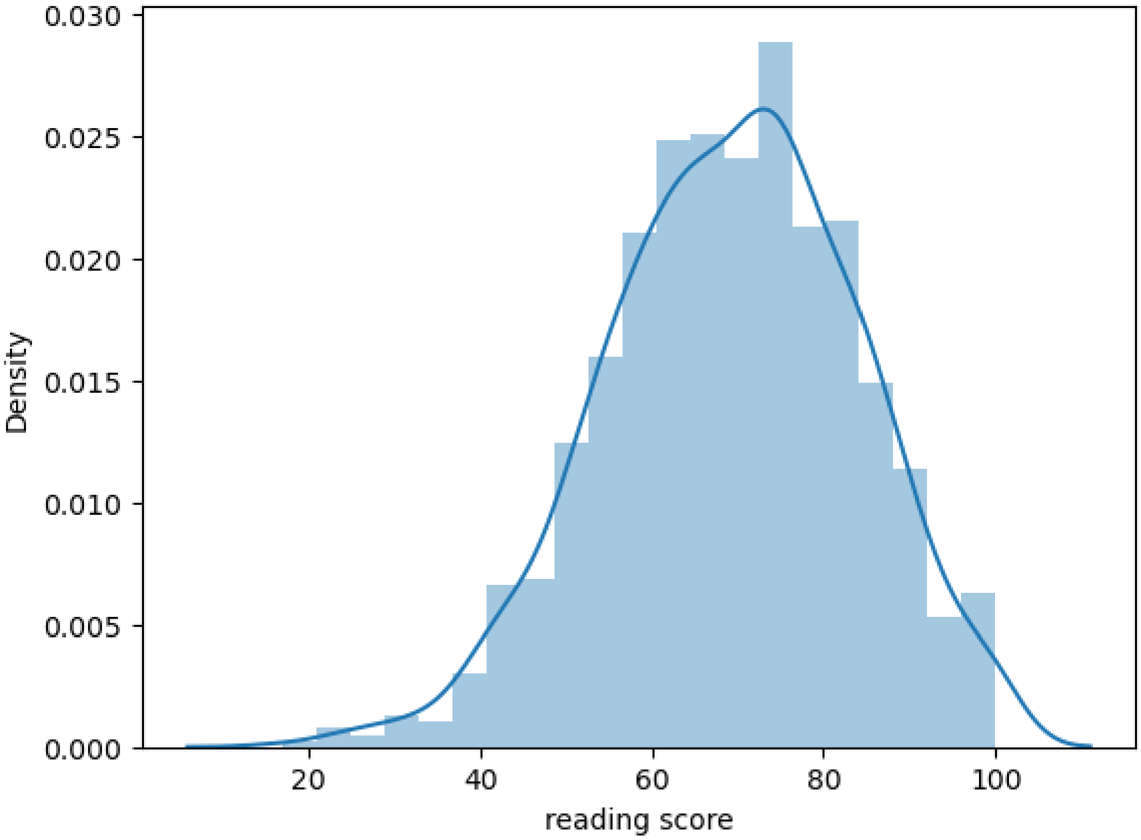
In [71]:
```
sns.distplot(data['writing score'])
```

Out[71]:
```
<AxesSubplot:xlabel='writing score', ylabel='Density'>
```



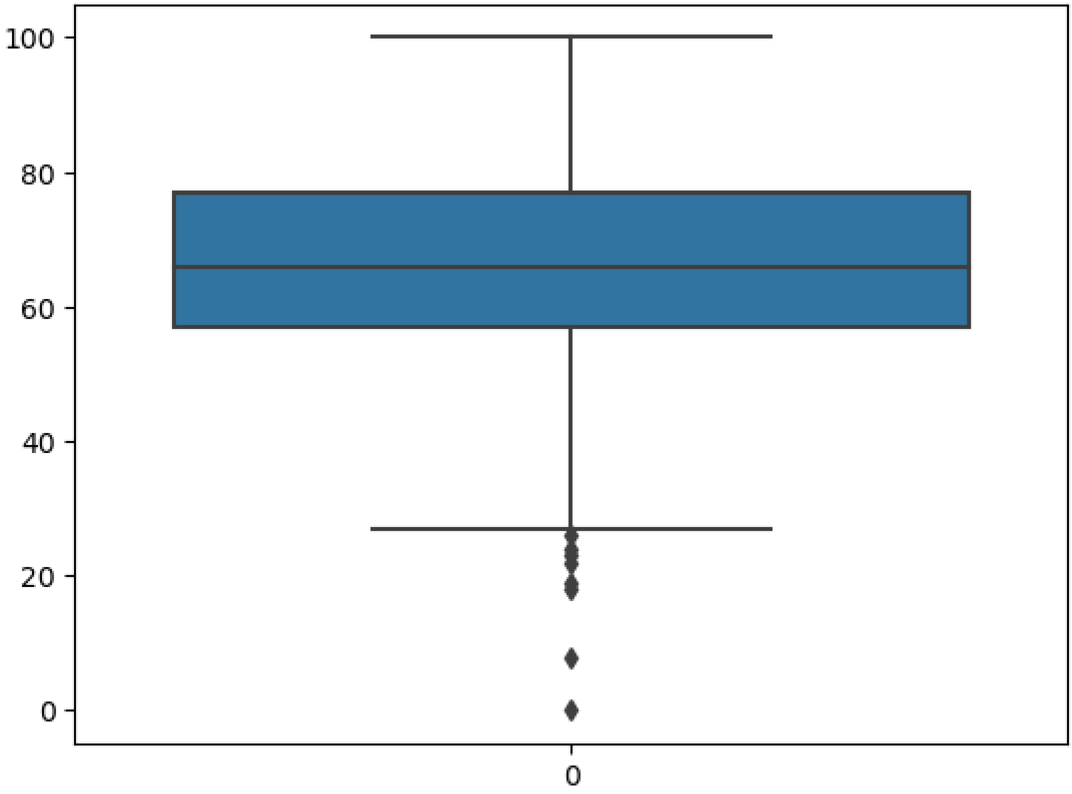
In [72]:
```
sns.distplot(data['reading score'])
```

Out[72]:
```
<AxesSubplot:xlabel='reading score', ylabel='Density'>
```



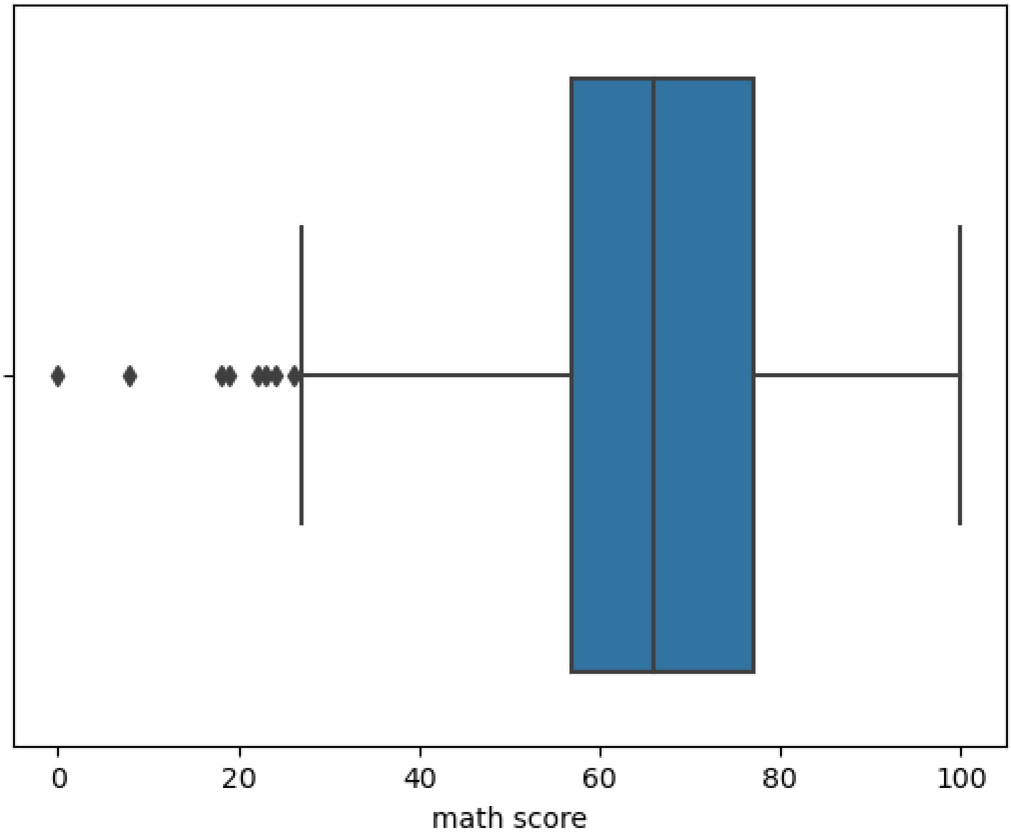
In [63]:
```
# outliers
sns.boxplot(data=data['math score'])
```

Out[63]:
```
<AxesSubplot:>
```



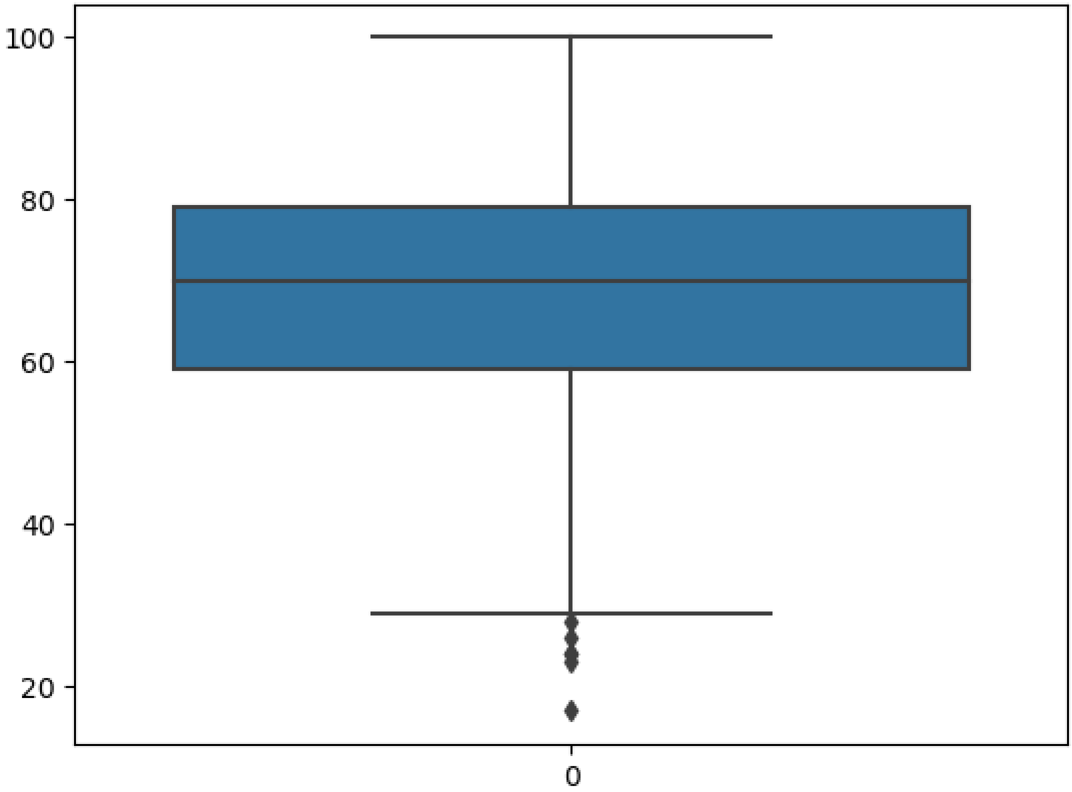
In [64]:
```
sns.boxplot(data['math score'])
```

Out[64]:
```
<AxesSubplot:xlabel='math score'>
```



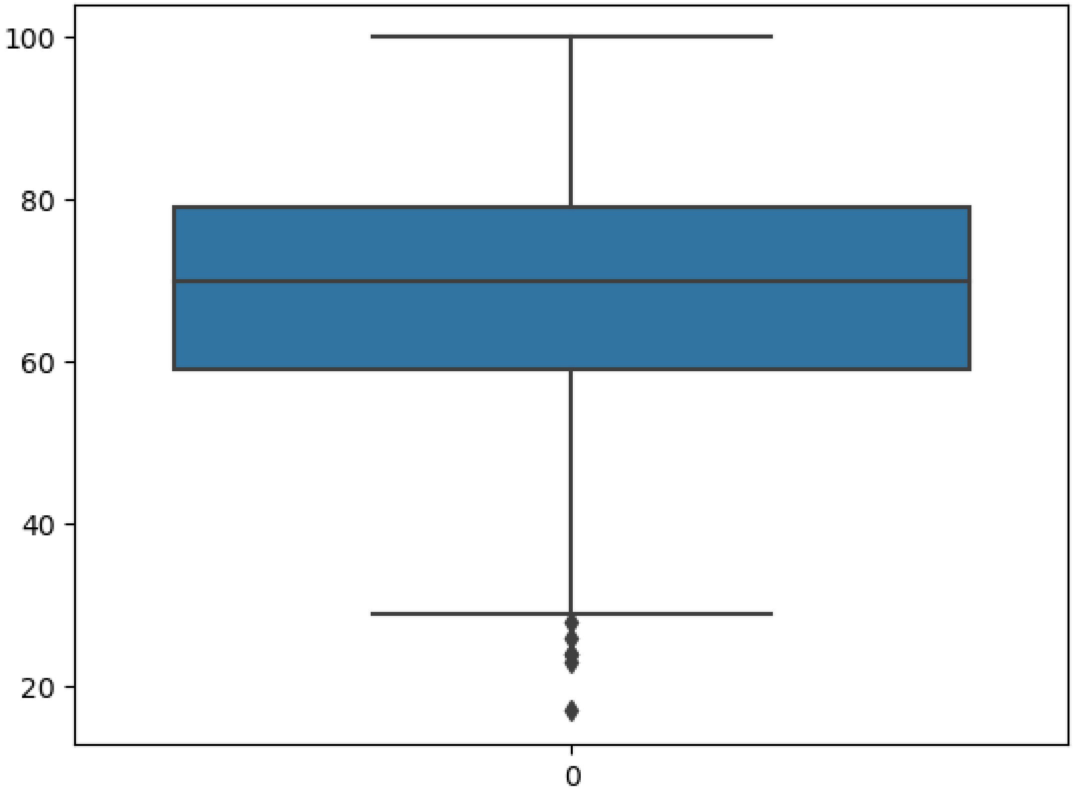
In [65]:
```
sns.boxplot(data=data['reading score'])
```

Out[65]:
```
<AxesSubplot:>
```



In [66]:
```
sns.boxplot(data=data['reading score'])
```

Out[66]:
```
<AxesSubplot:>
```



```
In [92]: q1 = data['math score'].quantile(0.25)
         q1
```

```
Out[92]: 57.0
```

```
In [93]: q3 = data['math score'].quantile(0.75)
         q3
```

```
Out[93]: 77.0
```

```
In [94]: #Interquartile range
         IQR = q3 - q1
         IQR
```

```
Out[94]: 20.0
```

```
In [95]: upper_limit = q3 + (1.5*IQR)
         upper_limit
```

```
Out[95]: 107.0
```

```
In [96]: lower_limit = q1 - (1.5*IQR)
         lower_limit
```

```
Out[96]: 27.0
```

```
In [97]: data['math score'].min()
```

```
Out[97]: 0
```

```
In [98]: data['math score'].max()
```

```
Out[98]: 100
```

In [99]:

```
data['math score'].unique()
```

Out[99]:

```
array([ 72,  69,  90,  47,  76,  71,  88,  40,  64,  38,  58,  65,  78,
        50,  18,  46,  54,  66,  44,  74,  73,  67,  70,  62,  63,  56,
        97,  81,  75,  57,  55,  53,  59,  82,  77,  33,  52,   0,  79,
        39,  45,  60,  61,  41,  49,  30,  80,  42,  27,  43,  68,  85,
        98,  87,  51,  99,  84,  91,  83,  89,  22, 100,  96,  94,  48,
        35,  34,  86,  92,  37,  28,  24,  26,  95,  36,  29,  32,  93,
        19,  23,   8], dtype=int64)
```

In [100…

```
# to check outlier in math score column which has less than lower limit 27
data[data['math score']<lower_limit]
```

Out[100]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	avera
17	female	group B	some high school	free/reduced	none	18	32	28	26.000
59	female	group C	some high school	free/reduced	none	0	17	10	9.000
145	female	group C	some college	free/reduced	none	22	39	33	31.333
338	female	group B	some high school	free/reduced	none	24	38	27	29.666
466	female	group D	associate's degree	free/reduced	none	26	31	38	31.666
787	female	group B	some college	standard	none	19	38	32	29.666
842	female	group B	high school	free/reduced	completed	23	44	36	34.333
980	female	group B	high school	free/reduced	none	8	24	23	18.333

In [102…

```
data[data['math score']>upper_limit]
```

Out[102]:

gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	average

In [103…

```
def outlier_thresold(df,variable):
    q1 = df[variable].quantile(0.25)
    q3 = df[variable].quantile(0.75)
    iqr = q3 - q1
    upper_limit = q3 + (1.5*iqr)
    lower_limit = q1 - (1.5*iqr)
    return upper_limit, lower_limit
```

In [104…

```
def get_iqr(df, column_name, variable, q1_range, q3_range):
    q1 = df[column_name].quantile(q1_range)
    q3 = df[column_name].quantile(q3_range)
    IQR = q3 - q1
    upper_fence = q3 + (1.5*IQR)
    lower_fence = q1 - (1.5*IQR)
    return IQR, upper_fence, lower_fence
```

In [112…

```
data_num.columns
```

Out[112]:

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

In [111…

```
# to check upper_limit, lower_limit for each columns
for variable in data_num.columns:
    upper_limit, lower_limit =  outlier_thresold(data_num, variable)
    print(upper_limit, lower_limit)
```

```
107.0 27.0
109.0 29.0
110.875 25.875
```

In [ ]:

```
def replace_with_thresold(data,numeric_col):
    for variable in numeric_col:
        upper_limit, lower_limit =  outlier_thresold(data_num, variable)
        data.loc[data[variable]<lower_limit, variable] = lower_limit
        data.loc[data[variable]>upper_limit, variable] = upper_limit
```

In [128…

```
#to check how many data points are lower than lower fence or lower limit
data.loc[data['math score']<lower_limit, 'math score']
```

Out[128]:

```
17     18
59      0
145    22
338    24
787    19
842    23
980     8
Name: math score, dtype: int64
```

In [133…

```
data[data['math score']<lower_limit]
```

Out[133]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	avera
17	female	group B	some high school	free/reduced	none	18	32	28	26.0000
59	female	group C	some high school	free/reduced	none	0	17	10	9.0000
145	female	group C	some college	free/reduced	none	22	39	33	31.3333
338	female	group B	some high school	free/reduced	none	24	38	27	29.6666
787	female	group B	some college	standard	none	19	38	32	29.6666
842	female	group B	high school	free/reduced	completed	23	44	36	34.3333
980	female	group B	high school	free/reduced	none	8	24	23	18.3333

In [129…

```
#to check how many data points are higher than upper fence or upper limit
data.loc[data['math score']>upper_limit, 'math score']
```

Out[129]:

```
Series([], Name: math score, dtype: int64)
```

In [130…

```
data['math score'].max()
```

Out[130]:

```
100
```

In [134…

```
data[data['math score']>upper_limit]
```

Out[134]:

gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	average

In [118…

```
data.head()
```

Out[118]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	average
0	female	group B	bachelor's degree	standard	none	72	72	74	72.666667
1	female	group C	some college	standard	completed	69	90	88	82.333333
2	female	group B	master's degree	standard	none	90	95	93	92.666667
3	male	group A	associate's degree	free/reduced	none	47	57	44	49.333333
4	male	group C	some college	standard	none	76	78	75	76.333333

In [135…

```
data
```

Out[135]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	avera
0	female	group B	bachelor's degree	standard	none	72	72	74	72.666
1	female	group C	some college	standard	completed	69	90	88	82.333
2	female	group B	master's degree	standard	none	90	95	93	92.666
3	male	group A	associate's degree	free/reduced	none	47	57	44	49.333
4	male	group C	some college	standard	none	76	78	75	76.333
...	...	...	...	...	...	...	...	...	
995	female	group E	master's degree	standard	completed	88	99	95	94.000
996	male	group C	high school	free/reduced	none	62	55	55	57.333
997	female	group C	high school	free/reduced	completed	59	71	65	65.000
998	female	group D	some college	standard	completed	68	78	77	74.333
999	female	group D	some college	free/reduced	none	77	86	86	83.000

1000 rows × 9 columns

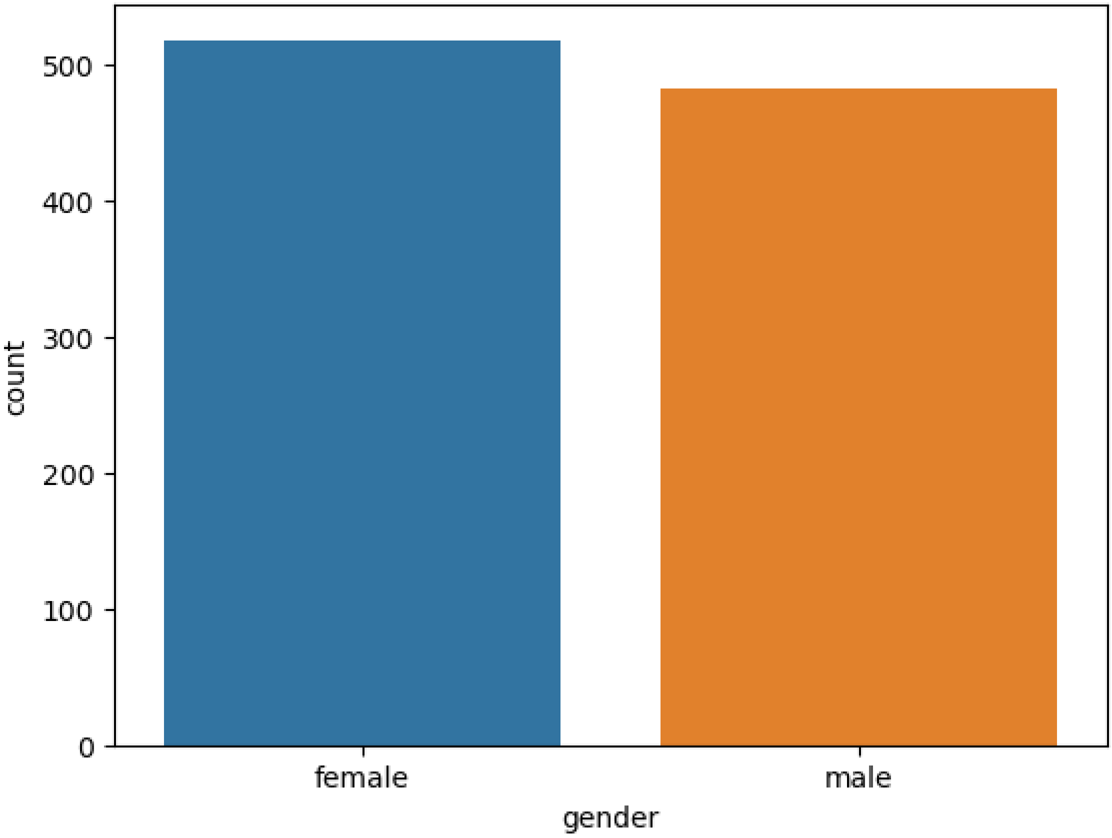
Graph Analysis

In [136…

```
sns.countplot(data['gender'])
```

Out[136]:

```
<AxesSubplot:xlabel='gender', ylabel='count'>
```

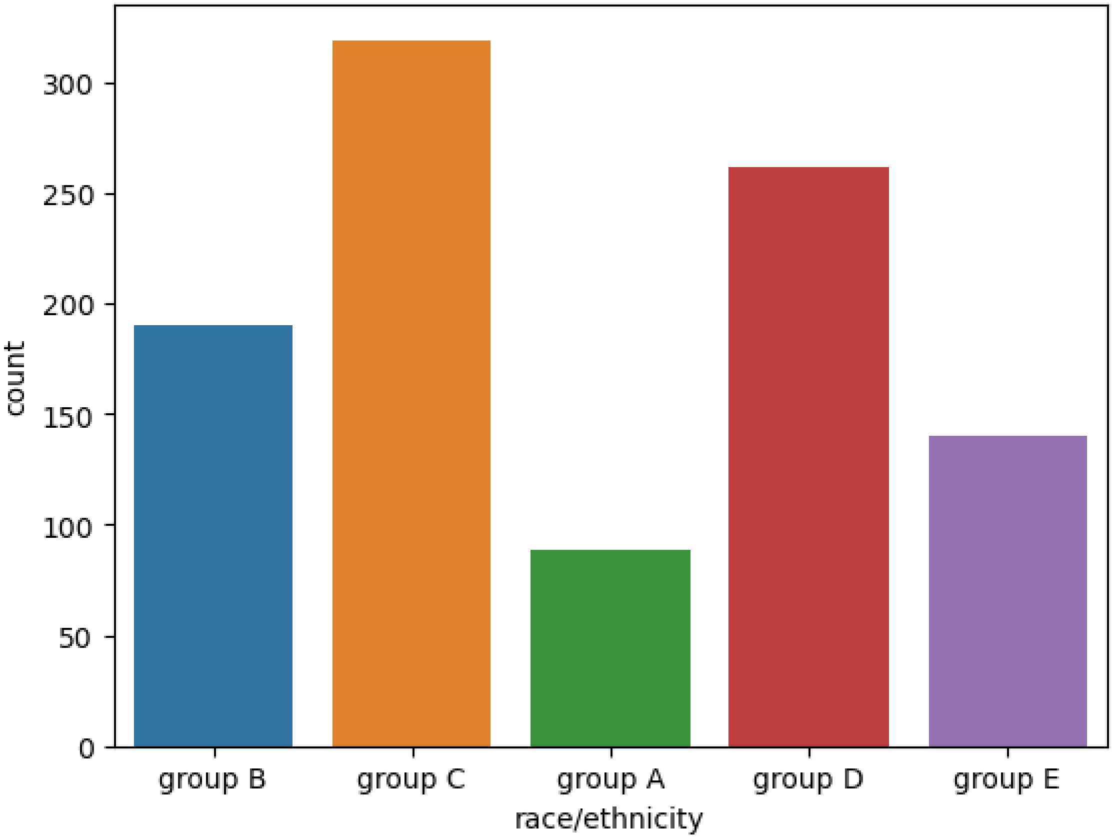


In [139…

```
sns.countplot(data['race/ethnicity'])
```

Out[139]:

```
<AxesSubplot:xlabel='race/ethnicity', ylabel='count'>
```



In [138…

```
data.head()
```

Out[138]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	average
0	female	group B	bachelor's degree	standard	none	72	72	74	72.666667
1	female	group C	some college	standard	completed	69	90	88	82.333333
2	female	group B	master's degree	standard	none	90	95	93	92.666667
3	male	group A	associate's degree	free/reduced	none	47	57	44	49.333333
4	male	group C	some college	standard	none	76	78	75	76.333333

In [141…

```
df = data.groupby('gender').mean()
df
```

Out[141]:

gender	math score	reading score	writing score	average
female	63.633205	72.608108	72.467181	69.569498
male	68.728216	65.473029	63.311203	65.837483

In [142…

```
df['average']
```

Out[142]:

```
gender
female    69.569498
male      65.837483
Name: average, dtype: float64
```

In [143…

```
df['average'][0]
```

Out[143]:

```
69.56949806949807
```

In [145…

```
df['average'][1]
```

Out[145]:

```
65.8374827109267
```

In [146…

```
df['math score'][0]
```

Out[146]:

```
63.633204633204635
```

In [147…

```
df['math score'][1]
```

Out[147]:

```
68.72821576763485
```

In [149…

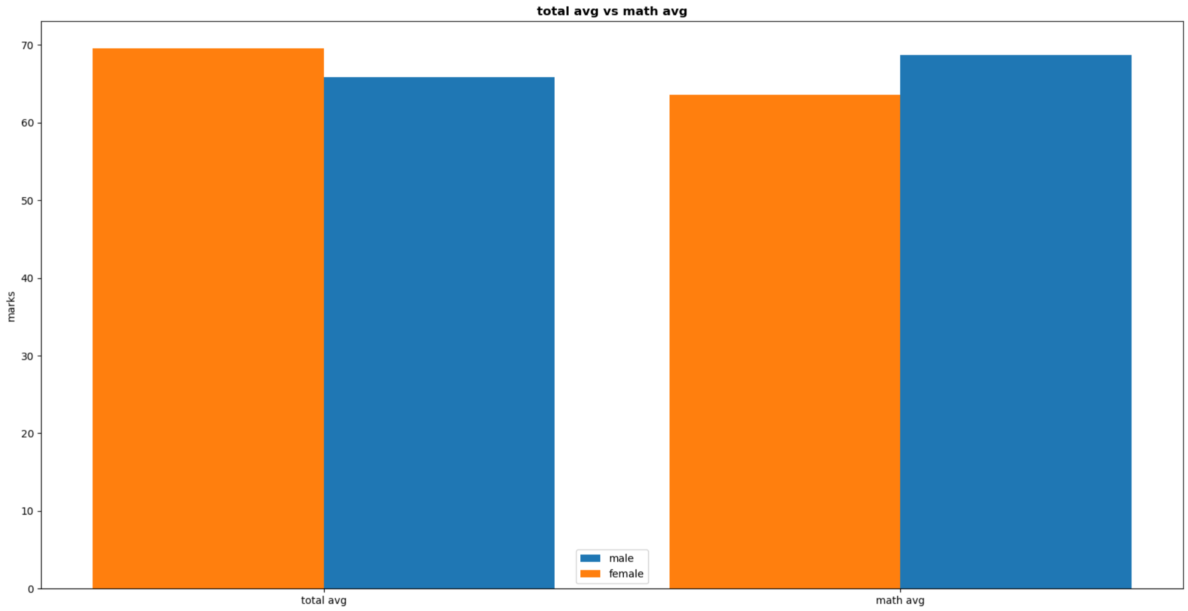
```
plt.figure(figsize = (20,10))
x = ['total avg', 'math avg']
female_score = df['average'][0],df['math score'][0]
male_score = df['average'][1], df['math score'][1]

x_axis = np.arange(len(x))
plt.bar(x_axis + 0.2, male_score, 0.4, label = 'male')
plt.bar(x_axis - 0.2, female_score, 0.4, label = 'female')

plt.xticks(x_axis,x)
plt.ylabel('marks')
plt.title('total avg vs math avg', fontweight = 'bold')
plt.legend()
plt.show
```

Out[149]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

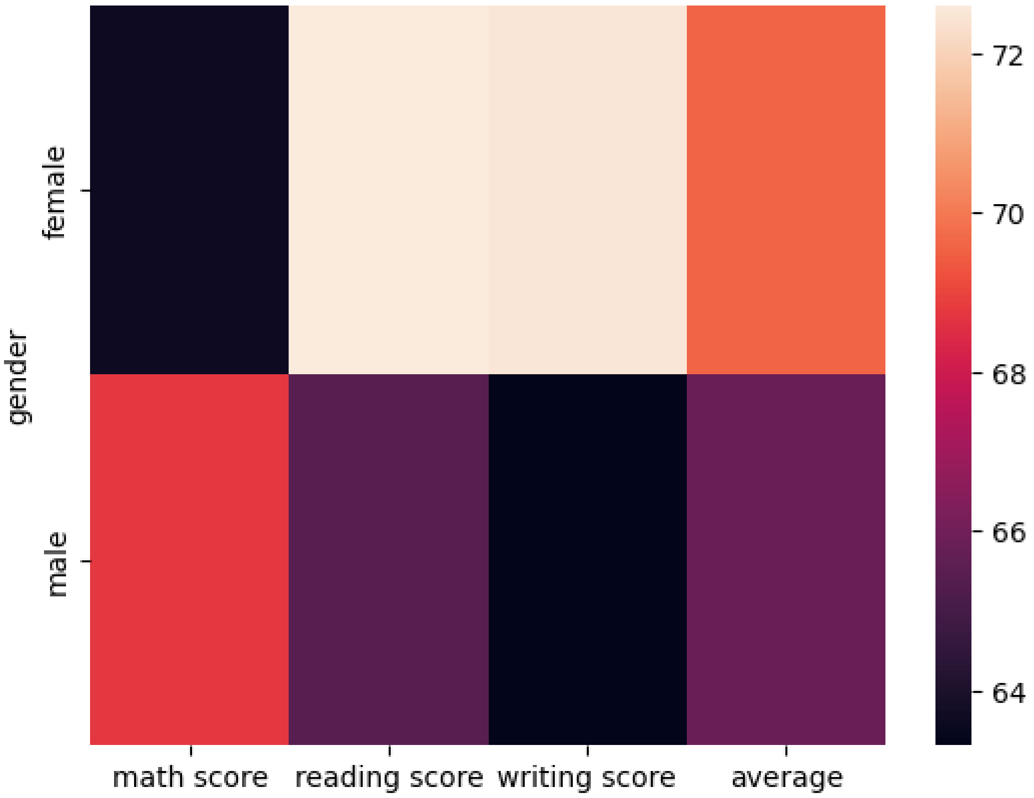


In [150…

```
sns.heatmap(df)
```

Out[150]:

```
<AxesSubplot:ylabel='gender'>
```



In [163…

```
data_num.corr()
```

Out[163]:

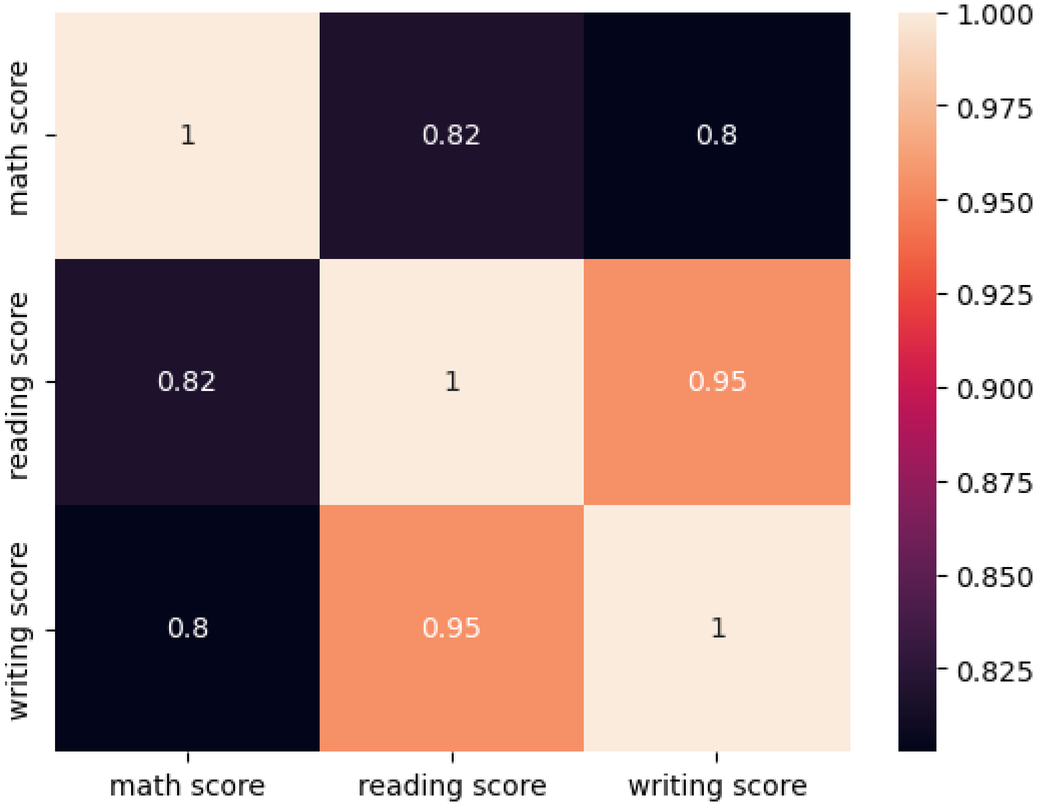
	math score	reading score	writing score
math score	1.000000	0.817580	0.802642
reading score	0.817580	1.000000	0.954598
writing score	0.802642	0.954598	1.000000

In [158…

```
sns.heatmap(data_num.corr(), annot = True)
```

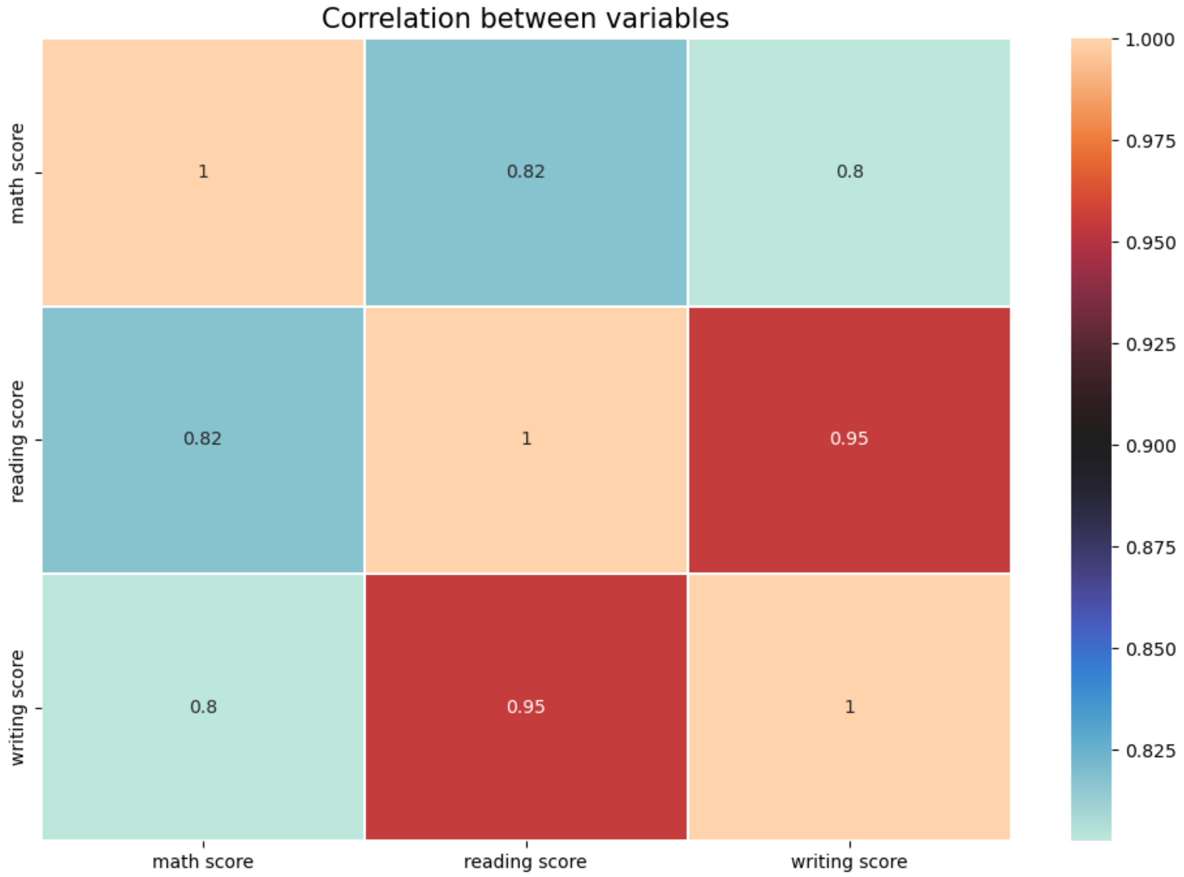
Out[158]:

```
<AxesSubplot:>
```

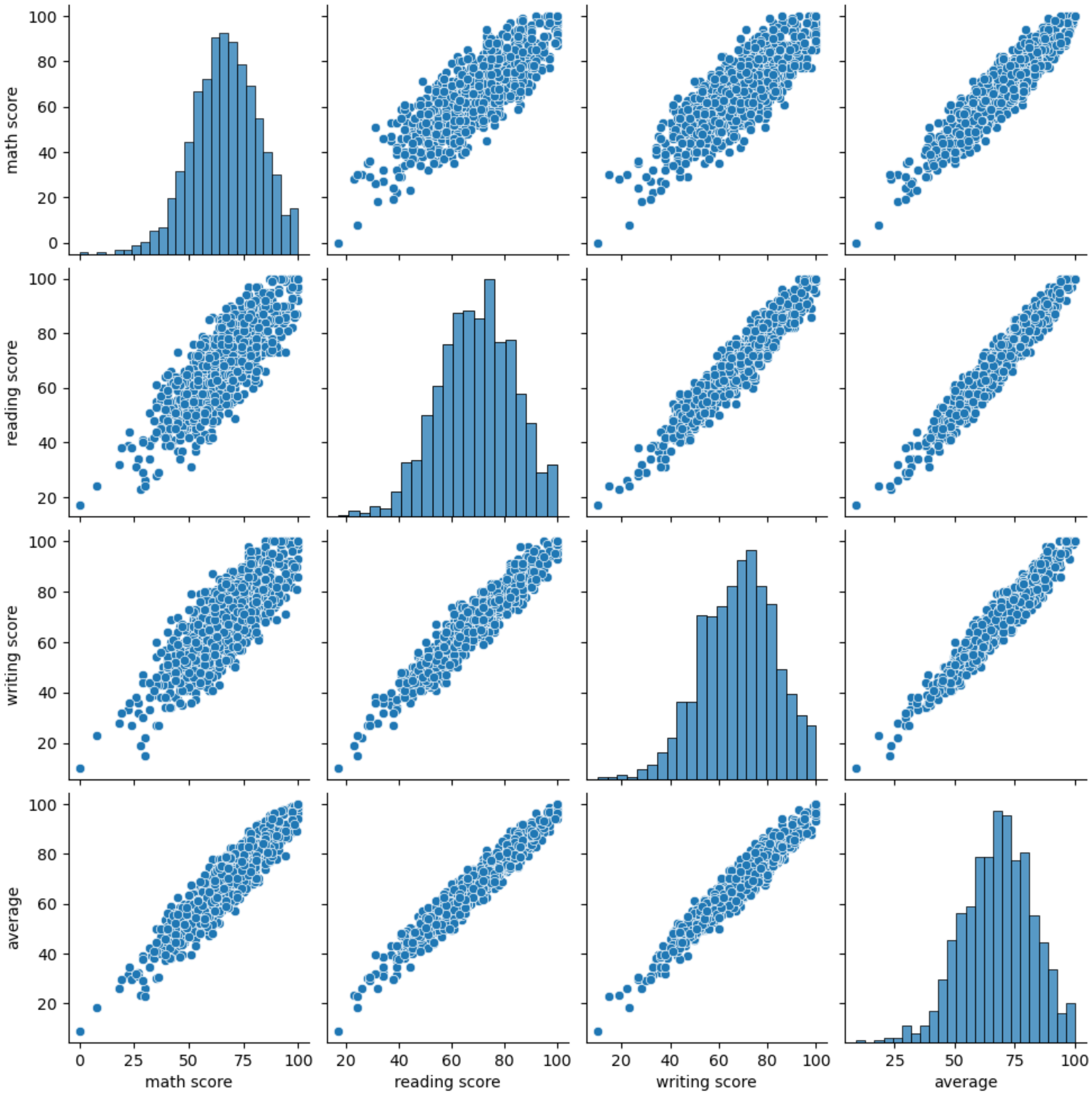


In [162…

```
sns.heatmap(data_num.corr(), annot = True, cmap = 'icefire', linewidths = 0.3)
fig = plt.gcf()
fig.set_size_inches(12,8)
plt.title('Correlation between variables', color = 'black', size = 15)
plt.show()
```



```
In [157…   sns.pairplot(data)
Out[157]:  <seaborn.axisgrid.PairGrid at 0x20dce2a0790>
```

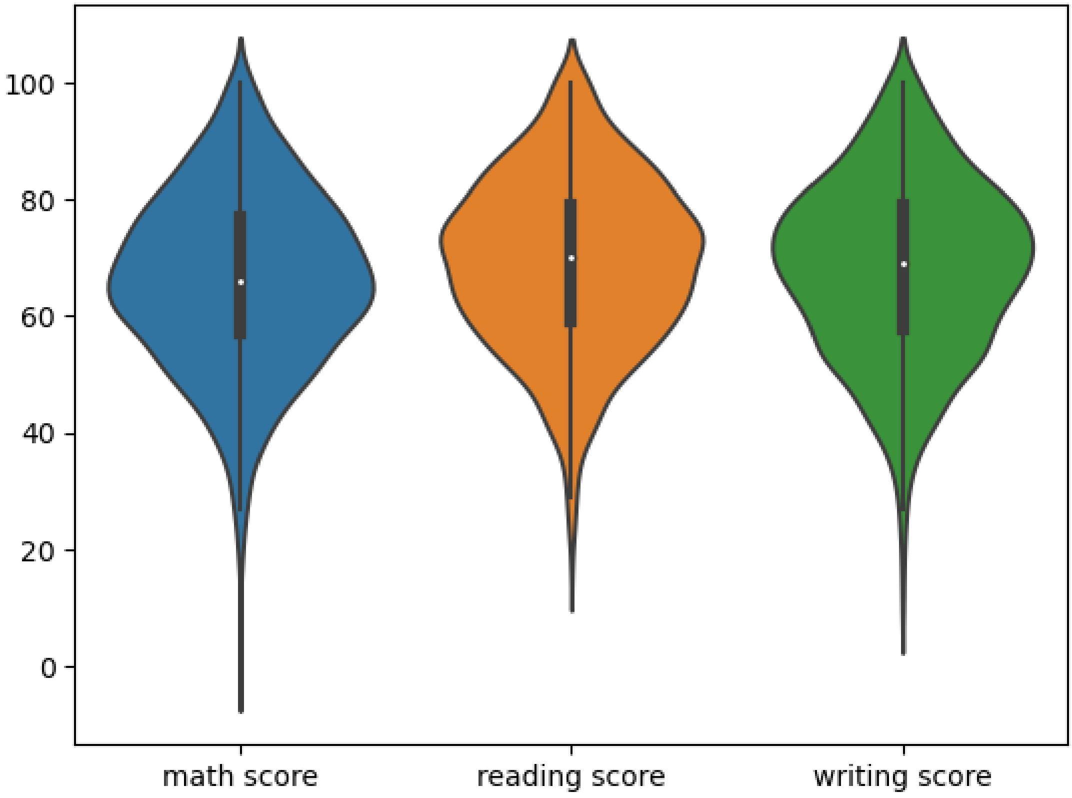


In [167…

```
sns.violinplot(data = data_num)
```

Out[167]:

```
<AxesSubplot:>
```



In [164…

```
df.head()
```

Out[164]:

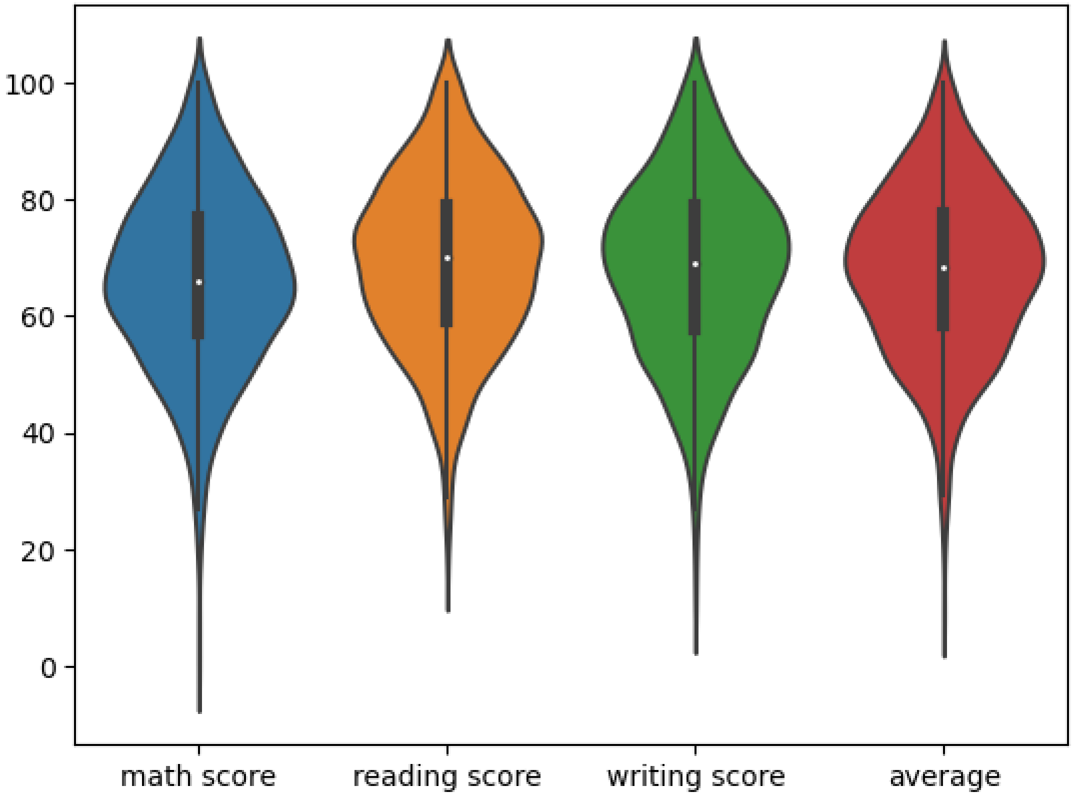
	math score	reading score	writing score	average
gender				
female	63.633205	72.608108	72.467181	69.569498
male	68.728216	65.473029	63.311203	65.837483

In [168…

```
sns.violinplot(data = data)
```

Out[168]:

```
<AxesSubplot:>
```



```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```