In [1]:
```
#comment
#obervations
```

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"C:\Users\Balodi\Desktop\DATASET\student.csv")
```

In [4]:
```
data.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]:
```
72+72+74/3
```

Out[5]:
```
168.66666666666666
```

In [6]:
```
data.tail()
```

Out[6]:

	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	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

In [7]:
```
data.shape
```

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

In [8]:
```
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 [9]:
```
data['gender'].dtypes
```

Out[9]:
```
dtype('O')
```

In [10]:
```
data['gender'].dtypes=='O'
```

Out[10]:
```
True
```

In [11]:
```
data.columns
```

Out[11]:
```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course', 'math score', 'reading score',
       'writing score'],
      dtype='object')
```

In [12]:
```
cat_col=[fea for fea in data.columns if data[fea].dtype == 'O']
```

In [13]:
```
num_col=[fea for fea in data.columns if data[fea].dtype != 'O']
```

In [14]:
```
data[num_col]
```

Out[14]:

	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 [15]:
```
data[cat_col]
```

Out[15]:

	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 [16]:
```
data.memory_usage()
```

Out[16]:
```
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 [17]:
```
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]:
```
data.isnull().sum().sum()
```

Out[18]:
```
0
```

In [19]:
```
data.duplicated().sum()
```

Out[19]:
```
0
```

In [20]:
```
data.nunique()
```

Out[20]:
```
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 [21]:
```
data['gender'].unique()
```

Out[21]:
```
array(['female', 'male'], dtype=object)
```

In [22]:
```
data.describe().T
```

Out[22]:

	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 [23]:
```
data.corr()
```

Out[23]:

	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 [24]:
```
data.cov()
```

Out[24]:

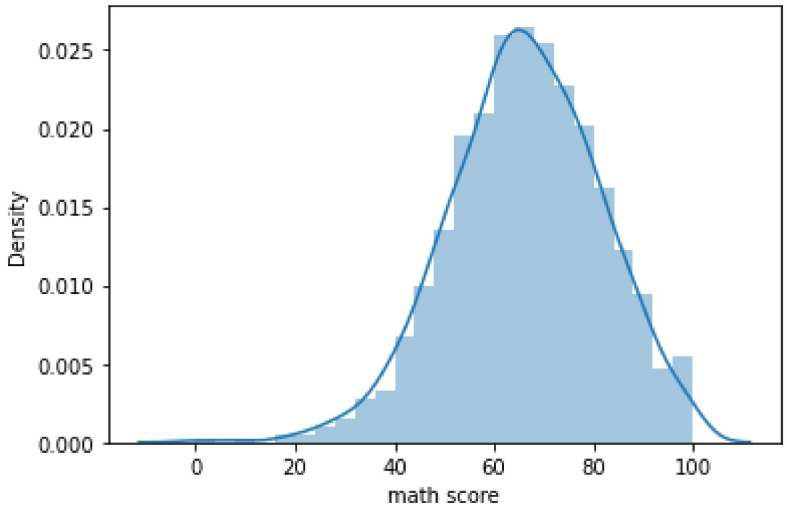
	math score	reading score	writing score
math score	229.918998	180.998958	184.939133
reading score	180.998958	213.165605	211.786661
writing score	184.939133	211.786661	230.907992

In [25]:
```
data.skew()
```

Out[25]:
```
math score      -0.278935
reading score   -0.259105
writing score   -0.289444
dtype: float64
```

In [26]:
```
sns.distplot(data['math score'])
```

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



In [27]:
```
data.columns
```

Out[27]:
```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course', 'math score', 'reading score',
       'writing score'],
      dtype='object')
```

In [28]:
```
data['Avarge']=(data['math score']+data['reading score']+data['writing score'])/3
```

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

Out[29]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Avarge
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 [30]:
```
data.groupby('gender').mean()
```

Out[30]:

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

In [31]:
```
data.groupby('gender').count()
```

Out[31]:

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

In [32]:
```
#question: you have to find out no of student whoever is having less than 30 mark
```

In [33]:
```
data[data["math score"] < 30].count()
```

Out[33]:
```
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
Avarge                         14
dtype: int64
```

In [34]:
```
data.columns
```

Out[34]:
```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course', 'math score', 'reading score',
       'writing score', 'Avarge'],
      dtype='object')
```

In [35]:
```
data_num=data[num_col]
```

In [36]:
```
data_num.head()
```

Out[36]:

	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

In [37]:
```
data.head()
```

Out[37]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Avarge
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 [38]:
```
data_num.head()
```

Out[38]:

	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

In [39]:
```
from scipy.stats import normaltest
```

In [40]:
```
normaltest(data_num['reading score'])[1]*100
```

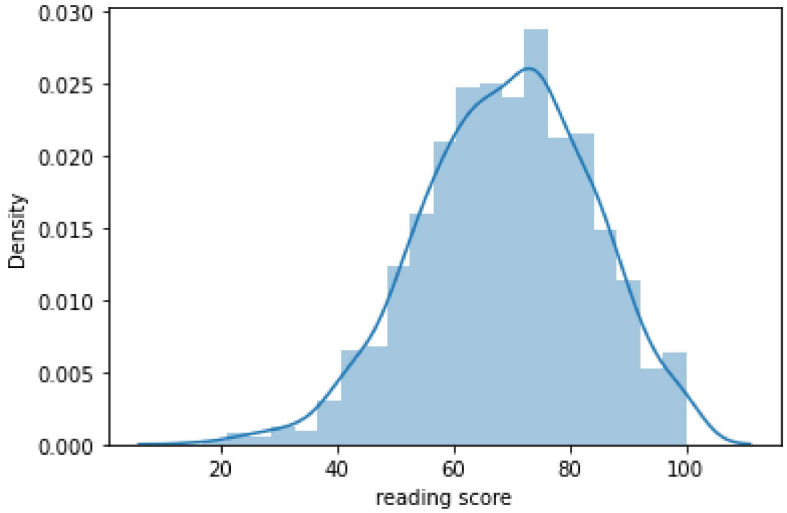
Out[40]:
```
0.3853758403576582
```

In [41]:
```
#if p>0.05 then my dala will be normal distributed
```

In [ ]:

In [42]:
```
sns.distplot(data_num['reading score'])
```

Out[42]: <AxesSubplot:xlabel='reading score', ylabel='Density'>

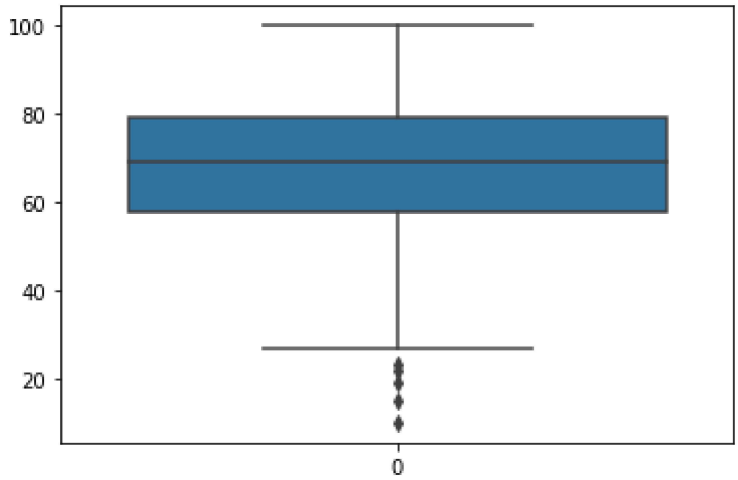


In [ ]:

In [43]:
```
#outlier
```

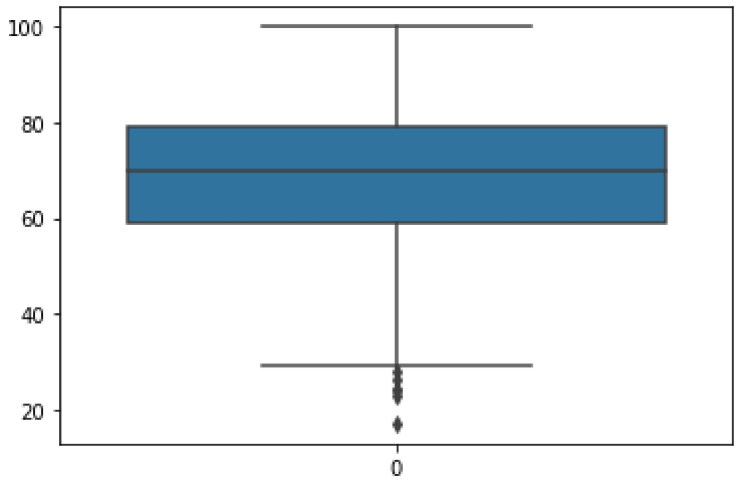
In [44]:
```
sns.boxplot(data=data['writing score'])
```

Out[44]: <AxesSubplot:>



In [45]:
```
sns.boxplot(data=data['reading score'])
```

Out[45]: <AxesSubplot:>



In [46]:
```
q1=data['math score'].quantile(0.25)
```

In [47]:
```
q3=data['math score'].quantile(0.75)
```

In [48]:
```
IQR=q3-q1
```

In [49]:
```
upper_limit=q3+(1.5*IQR)
```

In [50]:
```
upper_limit
```

Out[50]: 107.0

In [51]:
```
lower_limit=q1-(1.5*IQR)
```

In [52]:
```
lower_limit
```

Out[52]: 27.0

In [53]:
```
data_outlier=data[data['math score']<lower_limit]
```

In [54]:
```
data_num.drop(data_outlier.index)
```

Out[54]:

	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

992 rows × 3 columns

In [55]:
```
data
```

Out[55]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Avarge
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
...	...	...	...	...	...	...	...	...	..
995	female	group E	master's degree	standard	completed	88	99	95	94.000000
996	male	group C	high school	free/reduced	none	62	55	55	57.333333
997	female	group C	high school	free/reduced	completed	59	71	65	65.000000
998	female	group D	some college	standard	completed	68	78	77	74.333333
999	female	group D	some college	free/reduced	none	77	86	86	83.000000

1000 rows × 9 columns

In [56]:
```
data[data['math score']>upper_limit]
```

Out[56]:

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

In [57]:
```
data['math score'].quantile(1.00)
```

Out[57]: 100.0

In [58]:
```
data['math score'].min()
```

Out[58]: 0

In [59]:
```
data['math score'].max()
```

Out[59]:
```
100
```

In [60]:
```
data['math score'].unique()
```

Out[60]:
```
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 [61]:
```
def get_iqr(df, column_name, 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 [62]:
```
data_num.columns
```

Out[62]:
```
Index(['math score', 'reading score', 'writing score'], dtype='object')
```

In [63]:
```
def outlier_threshold(df,variable):
    q1=df[variable].quantile(0.25)
    q2=df[variable].quantile(0.75)
    iqr=q2-q1
    up_limit=q2
    +(1.5*iqr)
    lower_limit=q1-(1.5*iqr)
    return lower_limit,up_limit
```

In [64]:
```
def replace_with_threshold(data,numeric_col):
    for variable in numeric_col:
        low_limit,upper_limit=outlier_threshold(data_num,variable)
        data.loc[data[variable]<low_limit,variable]=low_limit
        data.loc[data[variable]>upper_limit,variable]=upper_limit
```

In [65]:
```
replace_with_threshold(data_num,data_num.columns)
```

In [66]:
```
data
```

Out[66]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Avarge
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
...	...	...	...	...	...	...	...	...	..
995	female	group E	master's degree	standard	completed	88	99	95	94.000000
996	male	group C	high school	free/reduced	none	62	55	55	57.333333
997	female	group C	high school	free/reduced	completed	59	71	65	65.000000
998	female	group D	some college	standard	completed	68	78	77	74.333333
999	female	group D	some college	free/reduced	none	77	86	86	83.000000

1000 rows × 9 columns

In [ ]:
```
```

In [67]:
```
data_num.loc[data_num['math score']<lower_limit,'math score']
```

Out[67]:
```
Series([], Name: math score, dtype: int64)
```

In [68]:
```
data_num.loc[data_num['math score']<lower_limit,'math score']=lower_limit
```

In [69]:
```
data_num
```

Out[69]:

	math score	reading score	writing score
0	72	72	74.0
1	69	79	79.0
2	77	79	79.0
3	47	57	44.0
4	76	78	75.0
...	...	...	...
995	77	79	79.0
996	62	55	55.0
997	59	71	65.0
998	68	78	77.0
999	77	79	79.0

1000 rows × 3 columns

In [70]:
```
def identifying_treating_outliers(df,col,remove_or_fill_with_quartile):
  q1=df[col].quantile(0.25)
  q3=df[col].quantile(0.75)
  iqr=q3-q1
  lower_fence=q1-1.5*(iqr)
  upper_fence=q3+1.5*(iqr)
  if remove_or_fill_with_quartile=="drop":
    df.drop(df.loc[df[col]<lower_fence].index,inplace=True)
    df.drop(df.loc[df[col]>upper_fence].index,inplace=True)
  elif remove_or_fill_with_quartile=="fill":
    df[col] = np.where(df[col] < lower_fence, lower_fence, df[col])
    df[col] = np.where(df[col] > upper_fence, upper_fence, df[col])
```

graph analysis

In [71]:
```
data
```

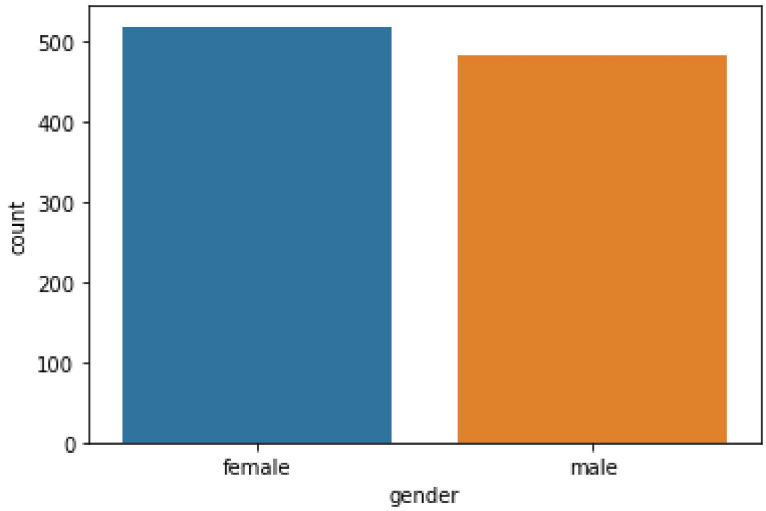
Out[71]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Avarge
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
...	...	...	...	...	...	...	...	...	..
995	female	group E	master's degree	standard	completed	88	99	95	94.000000
996	male	group C	high school	free/reduced	none	62	55	55	57.333333
997	female	group C	high school	free/reduced	completed	59	71	65	65.000000
998	female	group D	some college	standard	completed	68	78	77	74.333333
999	female	group D	some college	free/reduced	none	77	86	86	83.000000

1000 rows × 9 columns

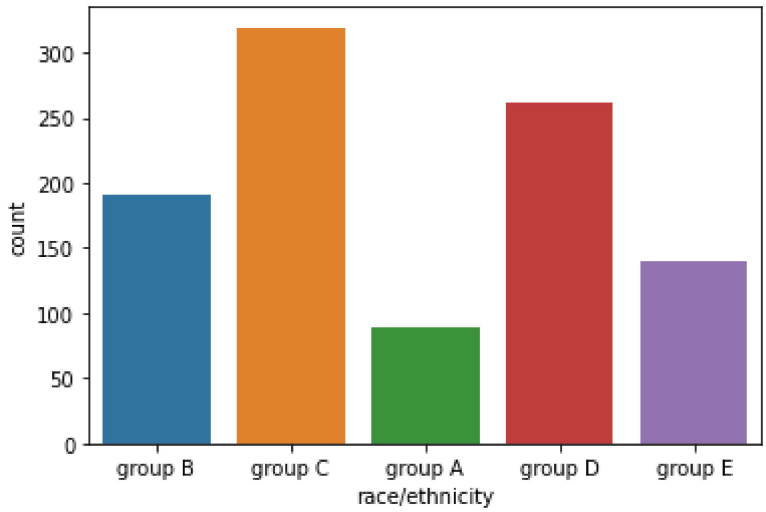
In [72]:
```
sns.countplot(data['gender'])
```

Out[72]: <AxesSubplot:xlabel='gender', ylabel='count'>



In [73]:
```
sns.countplot(data['race/ethnicity'])
```

Out[73]: <AxesSubplot:xlabel='race/ethnicity', ylabel='count'>



In [74]:
```
df=data.groupby('gender').mean()
```

In [75]:
```
df
```

Out[75]:

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

In [76]:
```
df['Avarge'][0]
```

Out[76]: 69.56949806949807

In [77]:
```
df['Avarge'][1]
```

Out[77]: 65.8374827109267

In [78]:
```
df['math score'][0]
```

Out[78]: 63.633204633204635

In [79]:
```
df['math score'][1]
```

Out[79]: 68.72821576763485

In [80]:
```
female_score=df['Avarge'][0],df['math score'][0]
```

In [81]:
```
female_score
```

Out[81]: (69.56949806949807, 63.633204633204635)

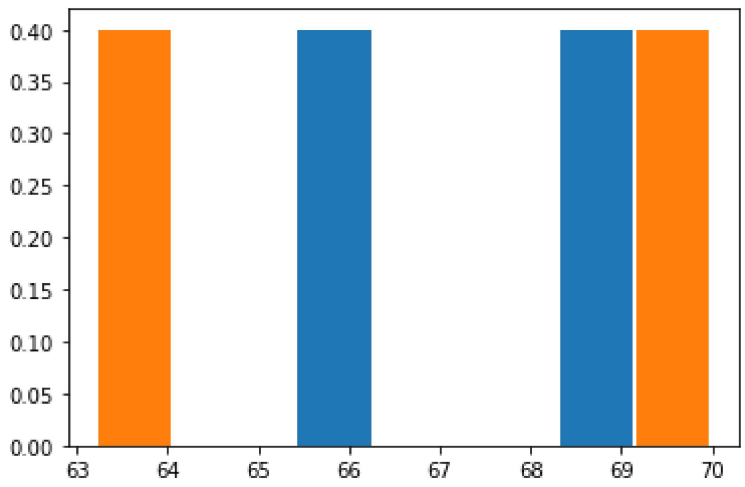
In [82]:
```
male_score = df['Avarge'][1],df['math score'][1]
```

In [83]:
```
male_score
```

Out[83]: (65.8374827109267, 68.72821576763485)

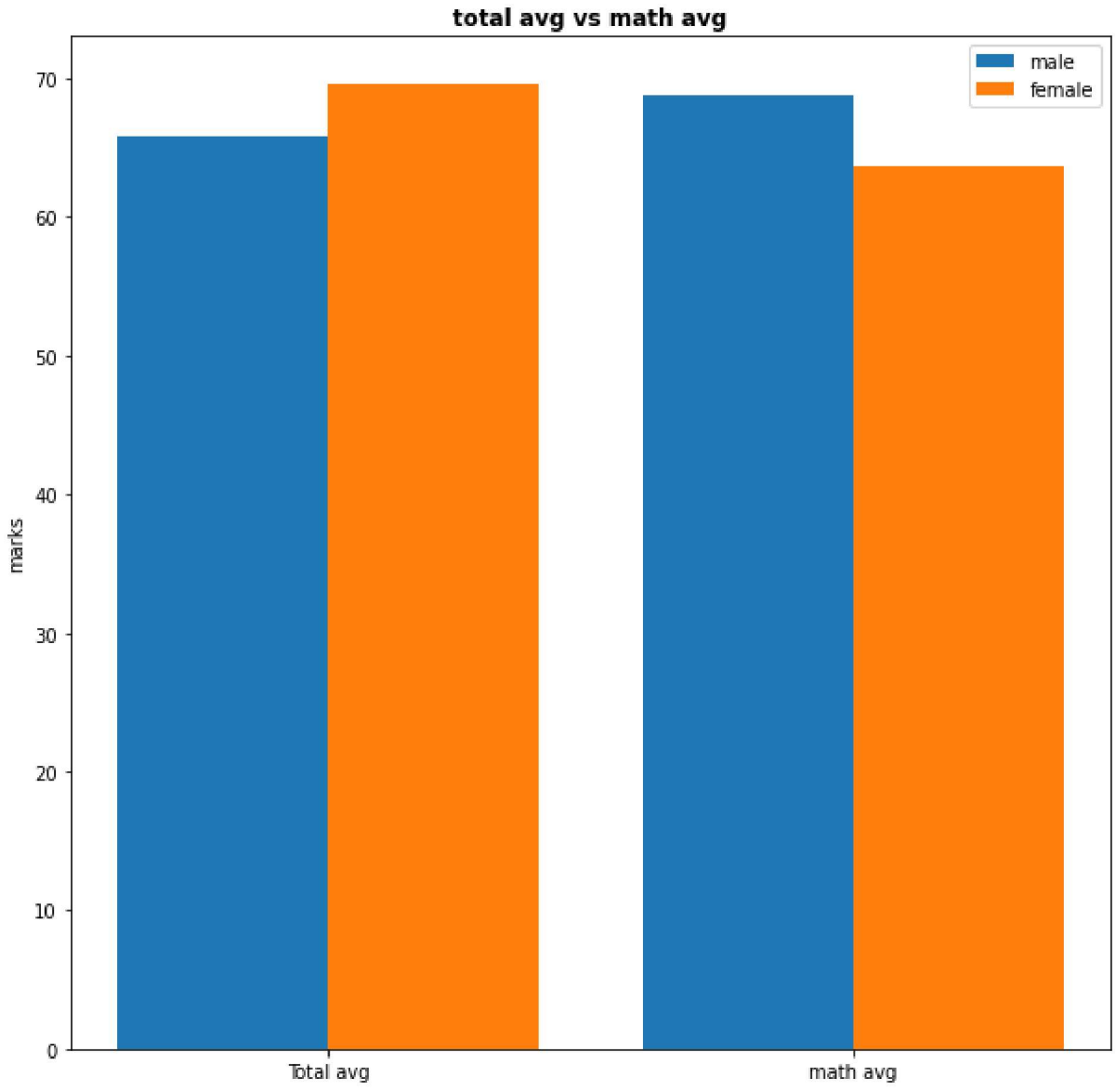
In [84]:
```
plt.bar(male_score,0.4,label='male')
plt.bar(female_score,0.4,label='female')
```

Out[84]: <BarContainer object of 2 artists>



In [85]:
```
plt.figure(figsize=(10,10))
X=['Total avg','math avg']
female_score=df['Avarge'][0],df['math score'][0]
male_score=df['Avarge'][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()
```



In [86]:
```
data_num.head()
```

Out[86]:

	math score	reading score	writing score
0	72	72	74.0
1	69	79	79.0
2	77	79	79.0
3	47	57	44.0
4	76	78	75.0

In [87]:
```
data_num.corr()
```

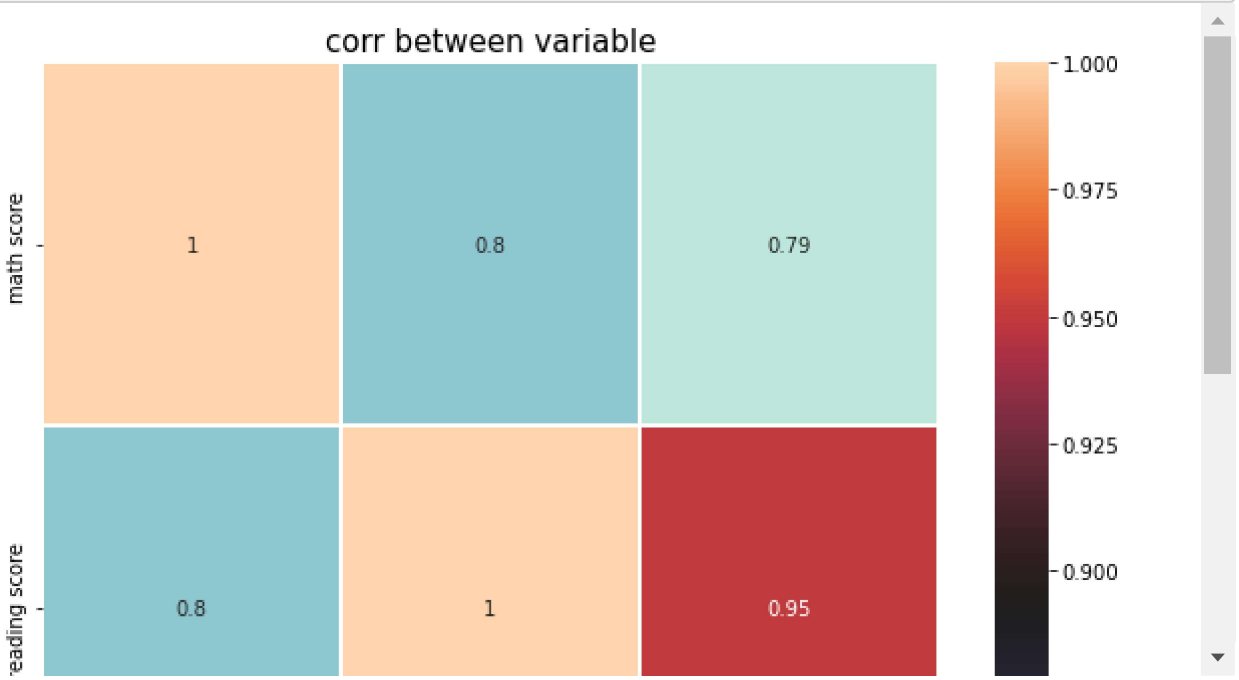
Out[87]:

	math score	reading score	writing score
math score	1.000000	0.799114	0.785275
reading score	0.799114	1.000000	0.949053
writing score	0.785275	0.949053	1.000000

In [88]:
```
sns.heatmap(data_num.corr())
```

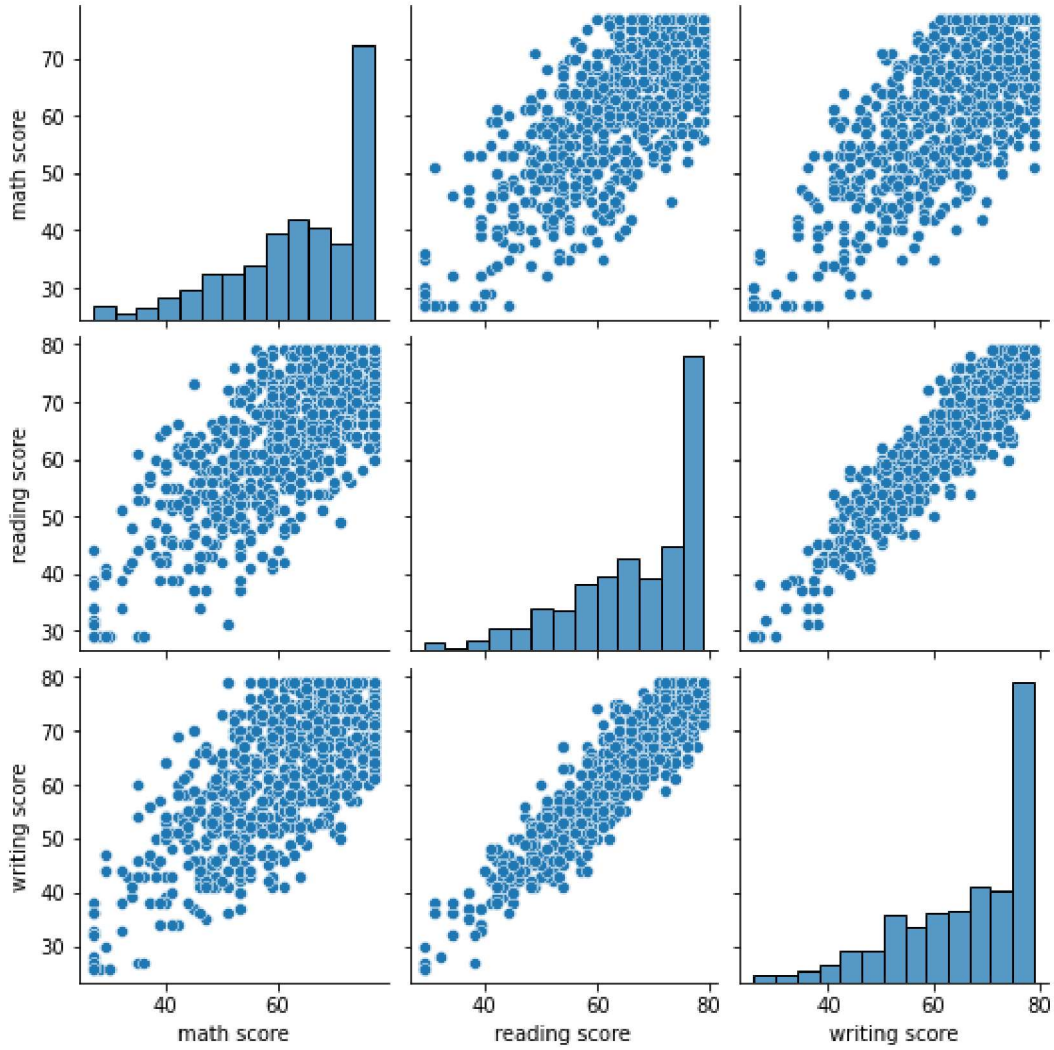
Out[88]: <AxesSubplot:>

In [89]:
```
sns.heatmap(data_num.corr(),annot=True,cmap='icefire',linewidths=0.3)
fig=plt.gcf()
fig.set_size_inches(10,10)
plt.title("corr between variable",color='black',size=15)
plt.show()
```



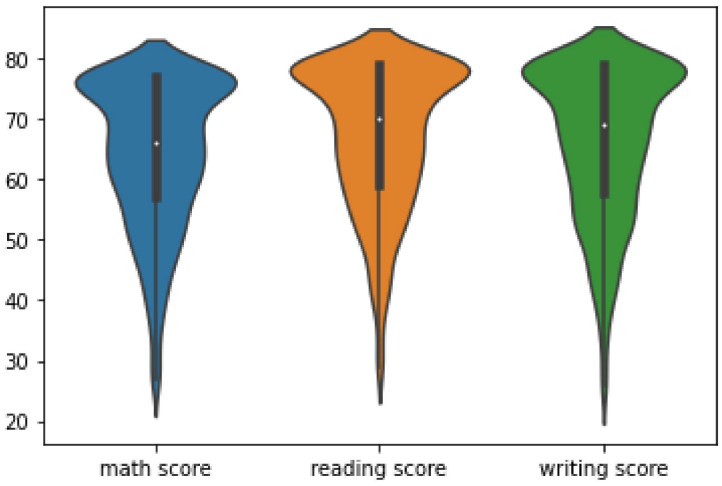
In [90]:
```
sns.pairplot(data_num)
```

Out[90]:
```
<seaborn.axisgrid.PairGrid at 0x1a6411bb160>
```



In [91]:
```
sns.violinplot(data=data_num)
```

Out[91]: <AxesSubplot:>



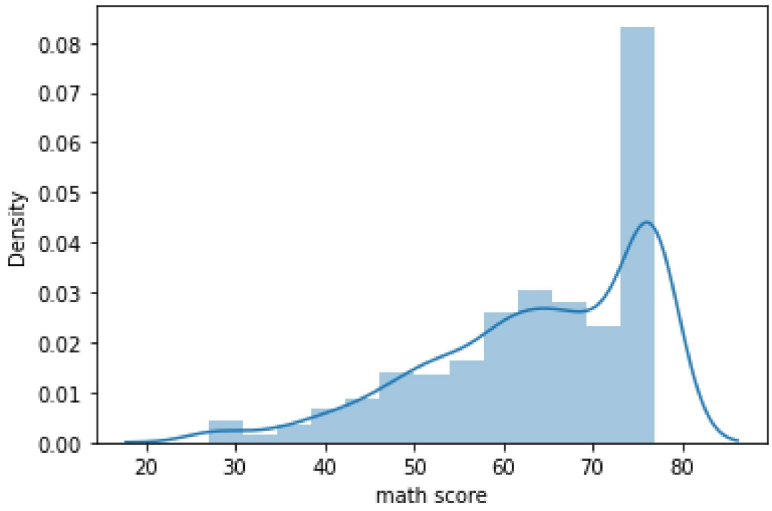
In [92]:
```
import numpy as np
data_num.head()
```

Out[92]:

	math score	reading score	writing score
0	72	72	74.0
1	69	79	79.0
2	77	79	79.0
3	47	57	44.0
4	76	78	75.0

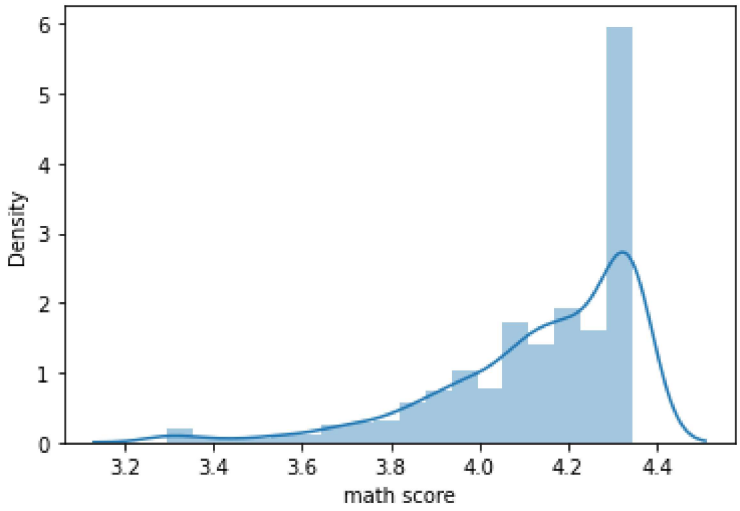
In [93]:
```
sns.distplot(data_num['math score'])
```

Out[93]: <AxesSubplot:xlabel='math score', ylabel='Density'>



In [94]:
```
sns.distplot(np.log(data_num['math score']))
```

Out[94]: <AxesSubplot:xlabel='math score', ylabel='Density'>



In [95]:
```
from sklearn.preprocessing import StandardScaler
```

In [96]:
```
scaler = StandardScaler()
```

In [97]:
```
scaler.fit(data_num)
```

Out[97]:
```
StandardScaler()
```

In [98]:
```
scaler.transform(data_num)
```

Out[98]:
```
array([[ 0.64111445,  0.41264193,  0.62435433],
       [ 0.39560215,  1.00503191,  1.02321994],
       [ 1.05030161,  1.00503191,  1.02321994],
       ...,
       [-0.42277218,  0.32801479, -0.09360379],
       [ 0.31376472,  0.92040477,  0.8636737 ],
       [ 1.05030161,  1.00503191,  1.02321994]])
```

In [ ]: