EDA

In [191…
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

#comment/#obeservation : it's always good to write comment and observartion

DATASET KAGGLE LINK ::- https://www.kaggle.com/datasets/spscientist/students-performance-in-exams

In [119…
```
#to load the dataset
data=pd.read_csv("student.csv")
```

In [9]:
```
#comment= its will show top 5 rows
data.head(5)
```

Out[9]:

	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 [120…
```
#for fetching the last 5 data
data.tail(5)
```

Out[120]:

	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 [121…
```
#for getting the knowledge of dimension
data.shape
```

Out[121]:
```
(1000, 8)
```

In [12]:
```
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 [13]:
```
data["gender"].dtypes
```

Out[13]:
```
dtype('O')
```

In [14]:
```
data["gender"].dtypes=="O"
```

Out[14]:
```
True
```

In [15]:
```
data.columns
```

Out[15]:
```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course', 'math score', 'reading score',
       'writing score'],
      dtype='object')
```

In [29]:
```
cat_col=[fea for fea in data.columns if data[fea].dtypes=="O"]
```

In [31]:
```
num_col=[fea for fea in data.columns if data[fea].dtypes!="O"]
```

In [32]:
```
data[cat_col]
```

Out[32]:

	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 [35]:
```
#for numerical value
data[num_col]
```

Out[35]:

	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 [40]:
```
#for memory checking
data.memory_usage()
```

Out[40]:
```
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 [41]:
```
data.isnull().sum()
```

Out[41]:
```
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 [42]:
```
#for duplicate value
```

In [43]:
```
data.duplicated()
```

Out[43]:
```
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 [45]:
```
#for knowing the sum
data.duplicated().sum()
```

Out[45]:
```
0
```

In [48]:
```
#for knowing the unique value
data.nunique()
```

Out[48]:
```
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 [49]:
```
data["gender"].unique()
```

Out[49]:
```
array(['female', 'male'], dtype=object)
```

In [51]:
```
#for statical analysis
data.describe()
```

Out[51]:

	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 [52]:
```
#for tanspose function
data.describe().T
```

Out[52]:

	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 [53]:
```
#for checking the correlation
data.corr()
```

Out[53]:

	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 [54]:
```
data.cov()
```

Out[54]:

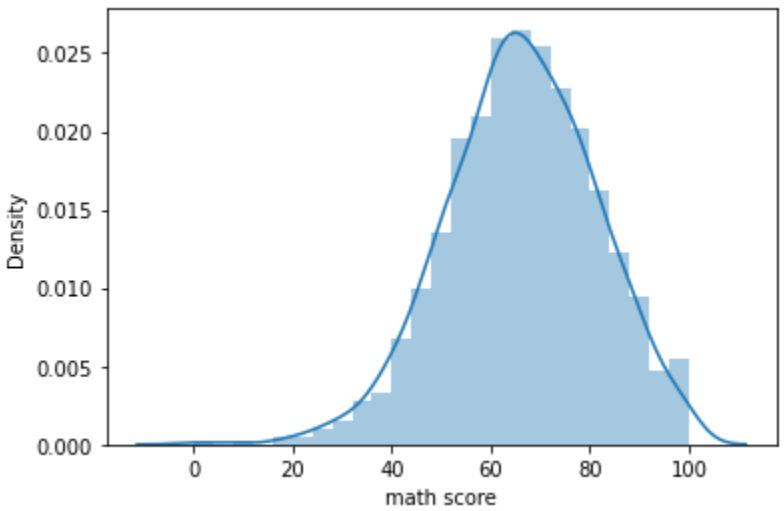
	math score	reading score	writing score
math score	229.918998	180.998958	184.939133
reading score	180.998958	213.165605	211.786661
writing score	184.939133	211.786661	230.907992

In [55]:
```
data.skew()
```

Out[55]:
```
math score      -0.278935
reading score   -0.259105
writing score   -0.289444
dtype: float64
```

In [58]:
```
sns.distplot(data["math score"])
```

Out[58]:
```
<AxesSubplot:xlabel='math score', ylabel='Density'>
```



In [59]:
```
data.columns
```

Out[59]:
```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course', 'math score', 'reading score',
       'writing score'],
      dtype='object')
```

In [63]:
```
#for finding the averarge
(data["math score"]+data["reading score"]+data["writing score"])/3
```

Out[63]:
```
0      72.666667
1      82.333333
2      92.666667
3      49.333333
4      76.333333
         ...
995    94.000000
996    57.333333
997    65.000000
998    74.333333
999    83.000000
Length: 1000, dtype: float64
```

In [177…
```
#feature adding
data["Average"]=(data["math score"]+data["reading score"]+data["writing score"])/3
```

In [179…
```
data["Average"]
```

Out[179]:
```
0      72.666667
1      82.333333
2      92.666667
3      49.333333
4      76.333333
         ...
995    94.000000
996    57.333333
997    65.000000
998    74.333333
999    83.000000
Name: Average, Length: 1000, dtype: float64
```

In [178…
```
data.head()
```

Out[178]:

	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 [66]:
```
#Group by Operations
data.groupby("gender").mean()
```

Out[66]:

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 [67]:
```
#for knowing the count
data.groupby("gender").count()
```

Out[67]:

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

In [68]:
```
#Question : you have to find out no. of student whoever is having less than 30 marks in
```

In [71]:
```
data[data["math score"]< 30].count()
```

Out[71]:
```
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 [73]:
```
data.columns
```

Out[73]:
```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course', 'math score', 'reading score',
       'writing score', 'Average'],
      dtype='object')
```

In [76]:
```
data_num=data[num_col]
```

In [77]:
```
data_num
```

Out[77]:

	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 [ ]:
```

```

In [78]:
```
from scipy.stats import normaltest
```

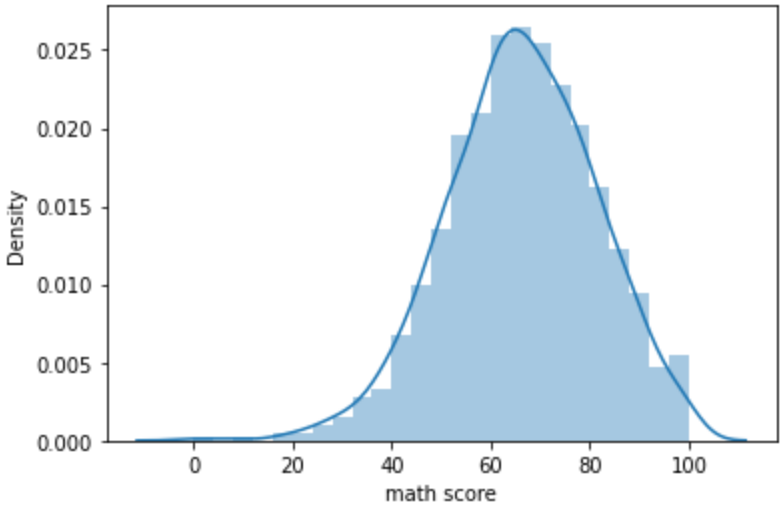
In [82]:
```
normaltest(data_num["math score"])[1]*100
```

Out[82]: 0.04508029386993784

In [87]:
```
#if p>0.05 then my data will be normally distributed
```

In [89]:
```
sns.distplot(data_num["math score"])
```

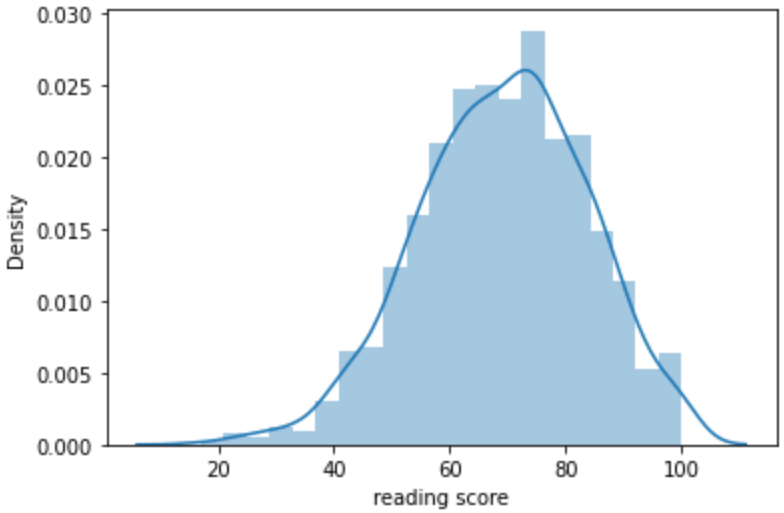
Out[89]: <AxesSubplot:xlabel='math score', ylabel='Density'>



OUTLIER

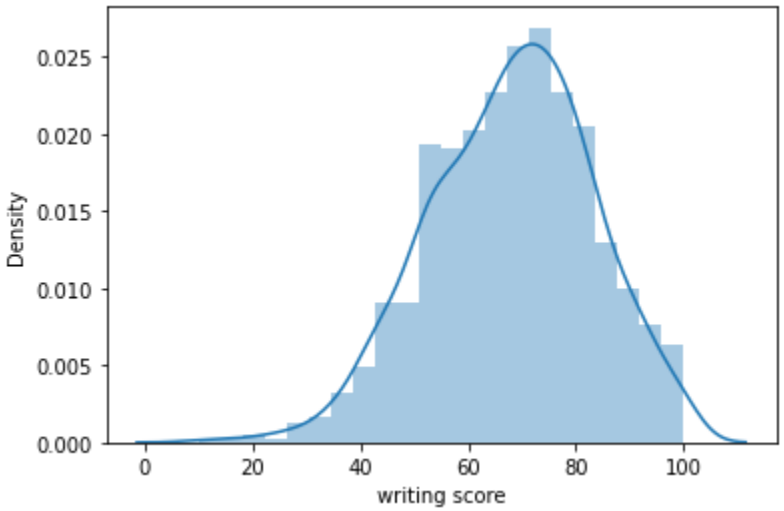
In [134…
```
sns.distplot(data_num['reading score'])
```

Out[134]: <AxesSubplot:xlabel='reading score', ylabel='Density'>



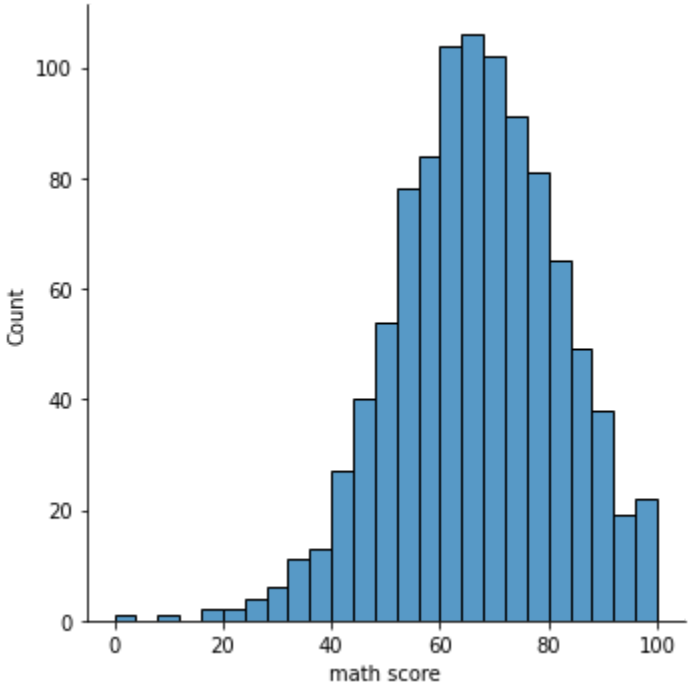
In [135…
```
sns.distplot(data_num['writing score'])
```

Out[135]: <AxesSubplot:xlabel='writing score', ylabel='Density'>



In [136…
```
sns.displot(data_num['math score'])
```

Out[136]:
```
<seaborn.axisgrid.FacetGrid at 0x1cadcd73340>
```

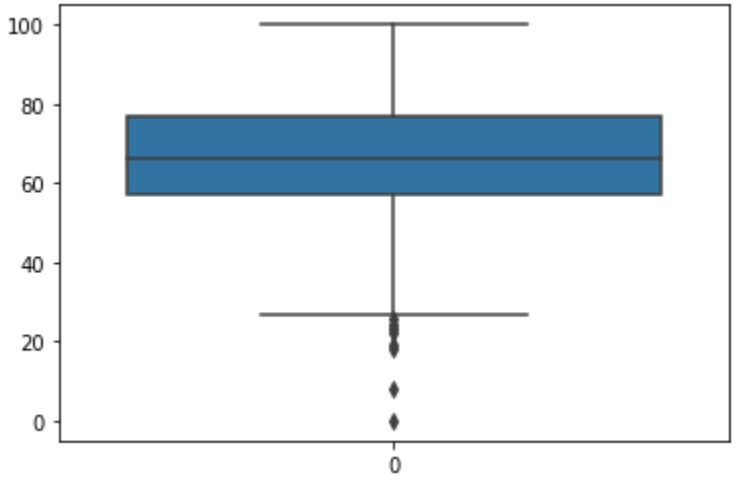


In [ ]:

In [ ]:

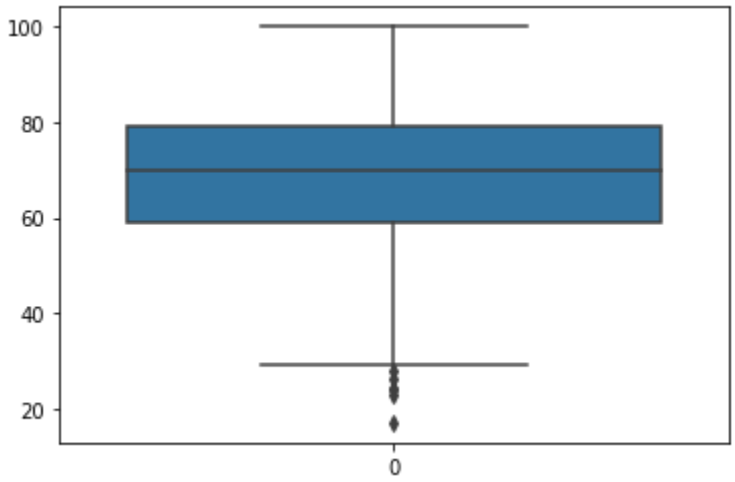
In [84]:
```
sns.boxplot(data=data["math score"])
```

Out[84]:
```
<AxesSubplot:>
```



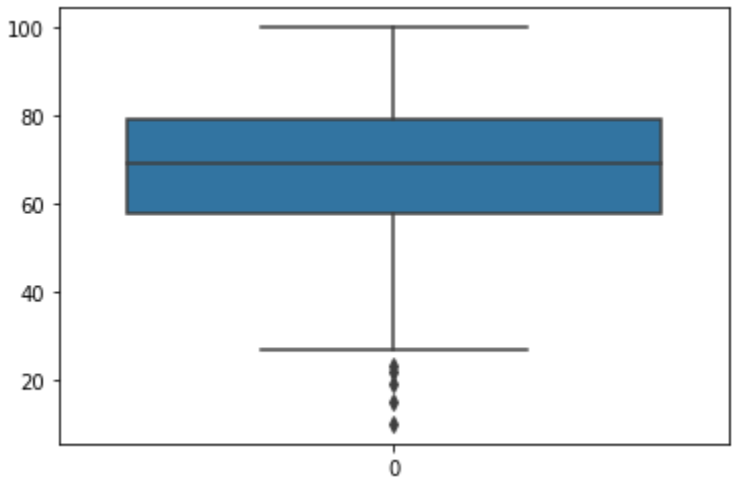
In [85]:
```
sns.boxplot(data=data["reading score"])
```

Out[85]: <AxesSubplot:>



In [86]:
```
sns.boxplot(data=data["writing score"])
```

Out[86]: <AxesSubplot:>



In [112…
```
q1=data["math score"].quantile(0.25)
```

In [113…
```
q3=data["math score"].quantile(0.75)
```

In [114…
```
iqr=q3-q1

iqr
```

Out[114]: 20.0

In [163…
```
upper_fence=q3+(1.5*iqr)
upper_fence
```

Out[163]: 107.0

In [116…
```
lower_fence=q1-(1.5*iqr)
lower_fence
```

Out[116]: 27.0

In [117…
```
#these are outliers
data[data["math score"]<lower_fence]
```

Out[117]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Average
17	female	group B	some high school	free/reduced	none	18	32	28	26.000000
59	female	group C	some high school	free/reduced	none	0	17	10	9.000000
145	female	group C	some college	free/reduced	none	22	39	33	31.333333
338	female	group B	some high school	free/reduced	none	24	38	27	29.666667
466	female	group D	associate's degree	free/reduced	none	26	31	38	31.666667
787	female	group B	some college	standard	none	19	38	32	29.666667
842	female	group B	high school	free/reduced	completed	23	44	36	34.333333
980	female	group B	high school	free/reduced	none	8	24	23	18.333333

In [118…
```
data[data["math score"]>upper_fence]
```

Out[118]:

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

In [139…
```
data['math score'].quantile(1.00)
```

Out[139]: 100.0

In [140…
```
data['math score'].min()
```

Out[140]: 0

In [141…
```
data['math score'].max()
```

Out[141]: 100

In [142…
```
data['math score'].unique()
```

Out[142]:
```
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 [143…
```
data_num.columns
```

Out[143]:
```
Index(['math score', 'reading score', 'writing score'], dtype='object')
```

In [ ]:
```
#function for getting outliers
```

In [159…
```
def outlier_threshold(data,variable):
    q1=df[variable].quantile(0.25)
    q3=df[variable].quantile(0.75)
    iqr=q3-q1
    upper_fence=q3+(1.5*iqr)
    lower_fence=q1-(1.5*iqr)
    return lower_fence,upper_fence
```

In [160…
```
for variable in data_num.columns:                          # only in numeric data
    lower_fence, upper_fence = outlier_threshold(data_num, variable)
```

In [164…
```
def replace_with_threshold(data,numeric_col):
    for variable in numeric_col:
        lower_fense,upper_fence=outlier_threshold(data_num,variable)
        data.loc[data[variable]<lower_fence,variable]=lower_fence
        data.loc[data[variable]>upper_fence,variable]=upper_fence
```

In [165…
```
replace_with_threshold(data_num, data_num.columns)
```

In [166…
```
data
```

Out[166]:

	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 [168…
```
data_num.loc[data_num['math score']<lower_fence, 'math_score']=lower_fence
```

In [169…
```
data_num
```

Out[169]:

	math score	reading score	writing score	math_score
0	71.275721	72.000000	74.000000	NaN
1	69.000000	76.175648	77.045171	NaN
2	71.275721	76.175648	77.045171	NaN
3	61.085699	58.733214	58.733214	NaN
4	71.275721	76.175648	75.000000	NaN
...	...	...	...	...
995	71.275721	76.175648	77.045171	NaN
996	62.000000	58.733214	58.733214	NaN
997	61.085699	71.000000	65.000000	NaN
998	68.000000	76.175648	77.000000	NaN
999	71.275721	76.175648	77.045171	NaN

1000 rows × 4 columns

In [170…
```
def identifying_outliers(df, col, remove_or_fill_with_quartile):
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3=q1
```

Graph

In [124…
```
data
```

Out[124]:

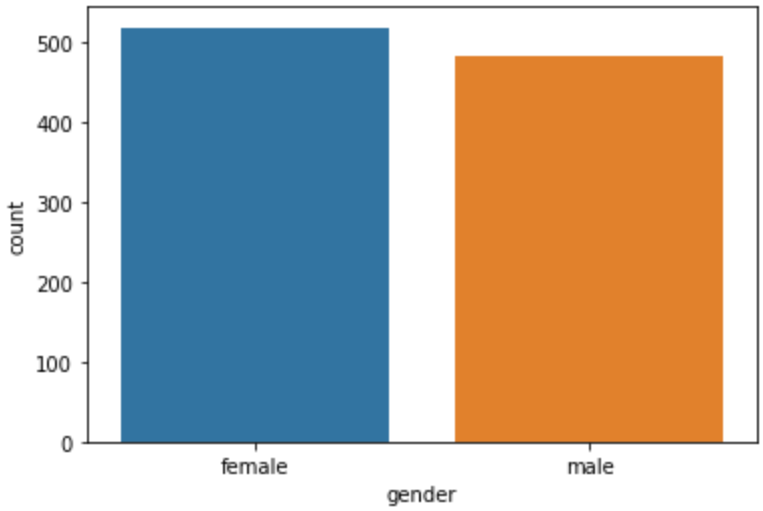
	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group 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

UNIVARIATE OPERATION

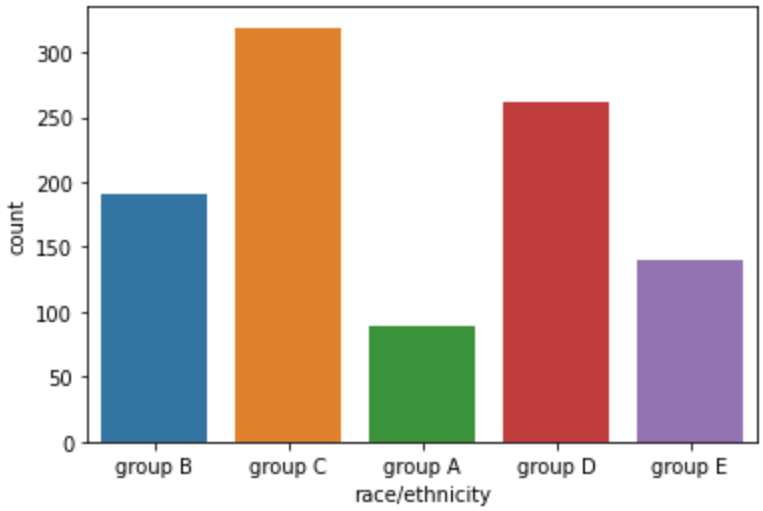
```
(data["gender"])
```

Out[126]:
```
<AxesSubplot:xlabel='gender', ylabel='count'>
```



In [127…
```
sns.countplot(data["race/ethnicity"])
```

Out[127]:
```
<AxesSubplot:xlabel='race/ethnicity', ylabel='count'>
```



In [171…
```
# Graph w.r.t group by
```

In [182…
```
df=data.groupby("gender").mean()
```

In [187…
```
df
```

Out[187]:

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 [184…
```
df['Average']
```

Out[184]:
```
gender
female    69.569498
male      65.837483
Name: Average, dtype: float64
```

In [185…
```
df['Average'][0]
```

Out[185]: 69.56949806949807

In [186…
```
df['Average'][1]
```

Out[186]: 65.8374827109267

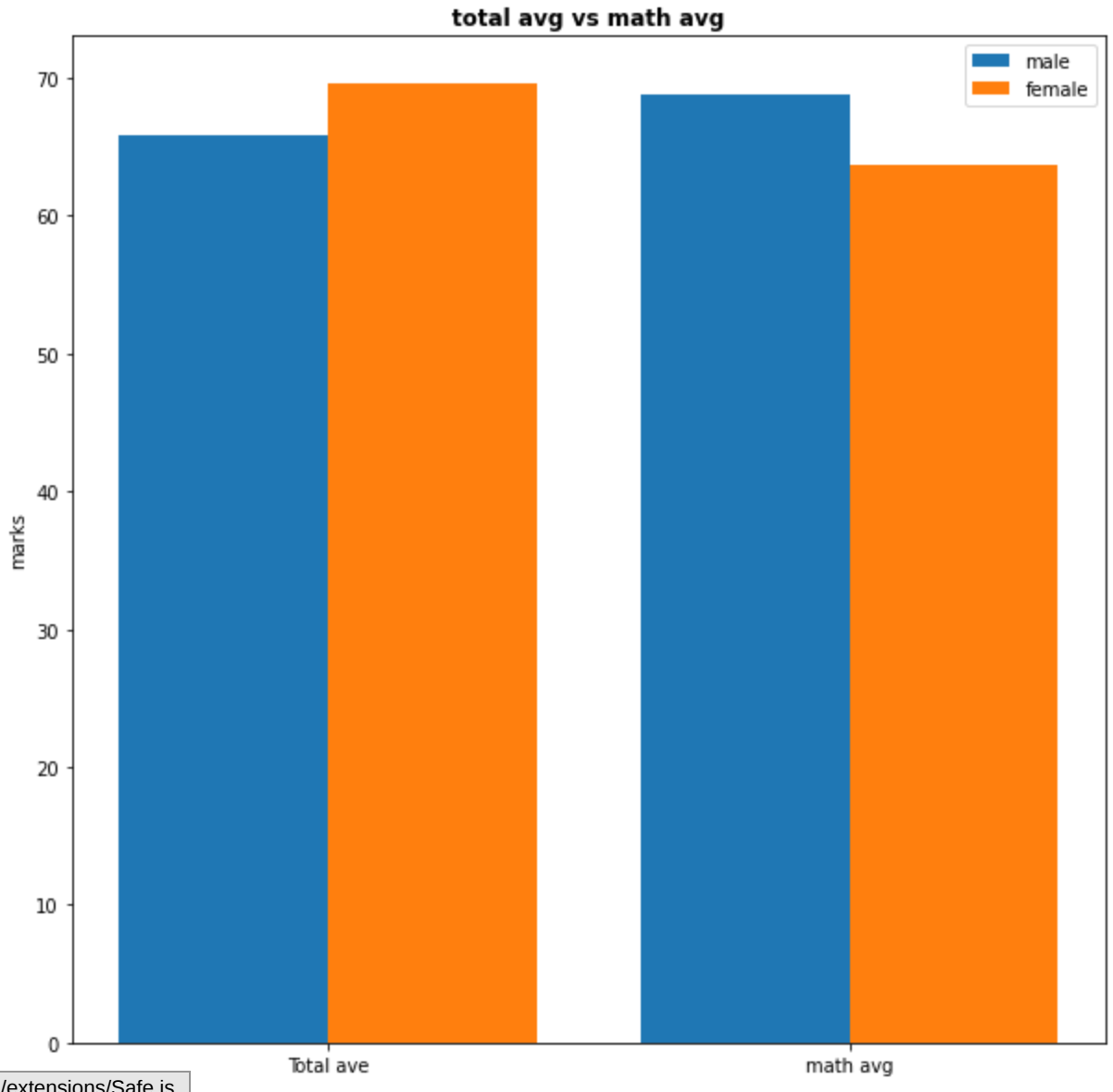
In [188…
```
female_score = df['Average'][0], df['math score'][0]
```

In [189…
```
female_score
```

Out[189]: (69.56949806949807, 63.633204633204635)

In [192…
```
plt.figure(figsize=(10,10))
X = ['Total ave', '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()
```

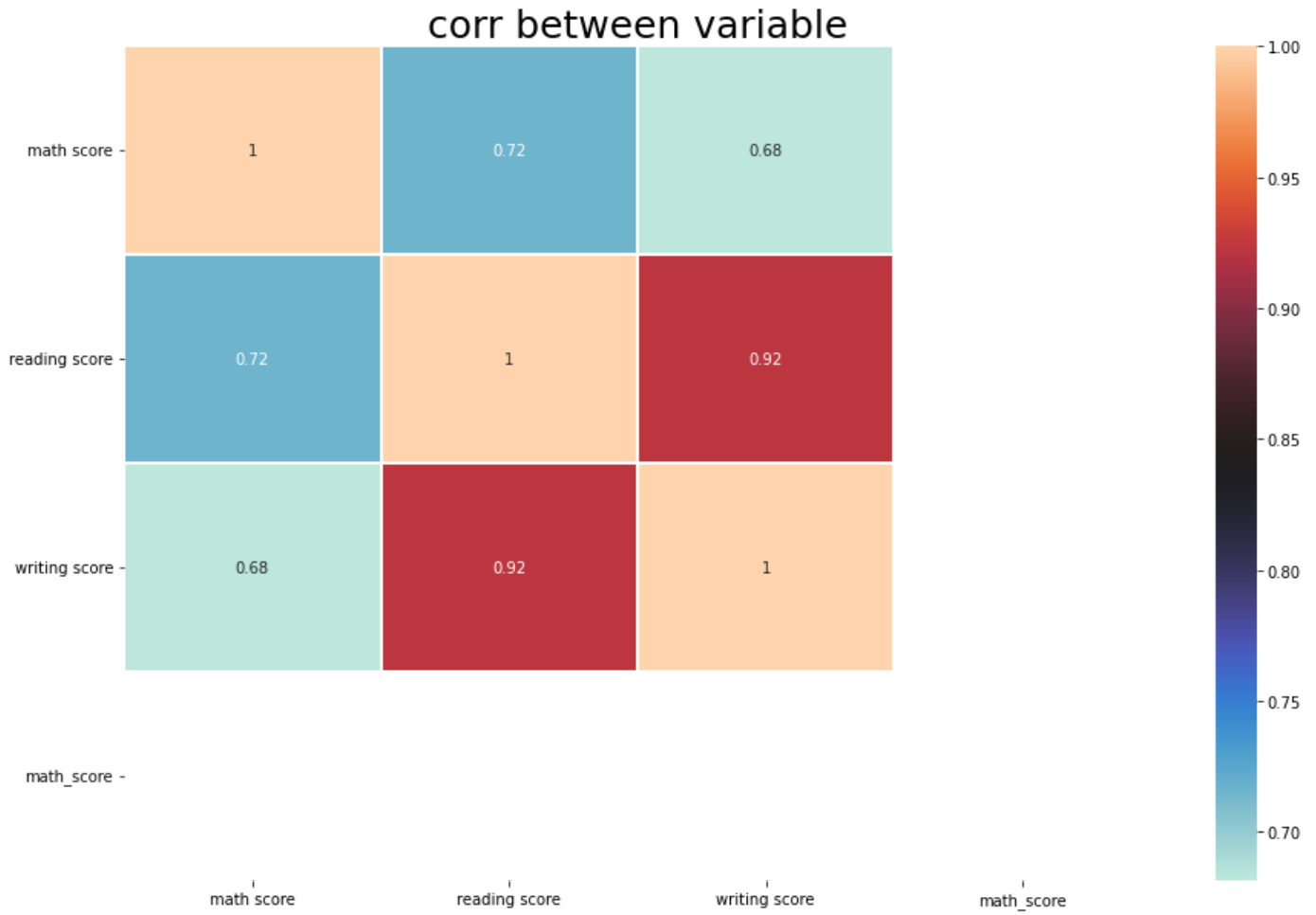


In [193…
```
data_num.corr()
```

Out[193]:

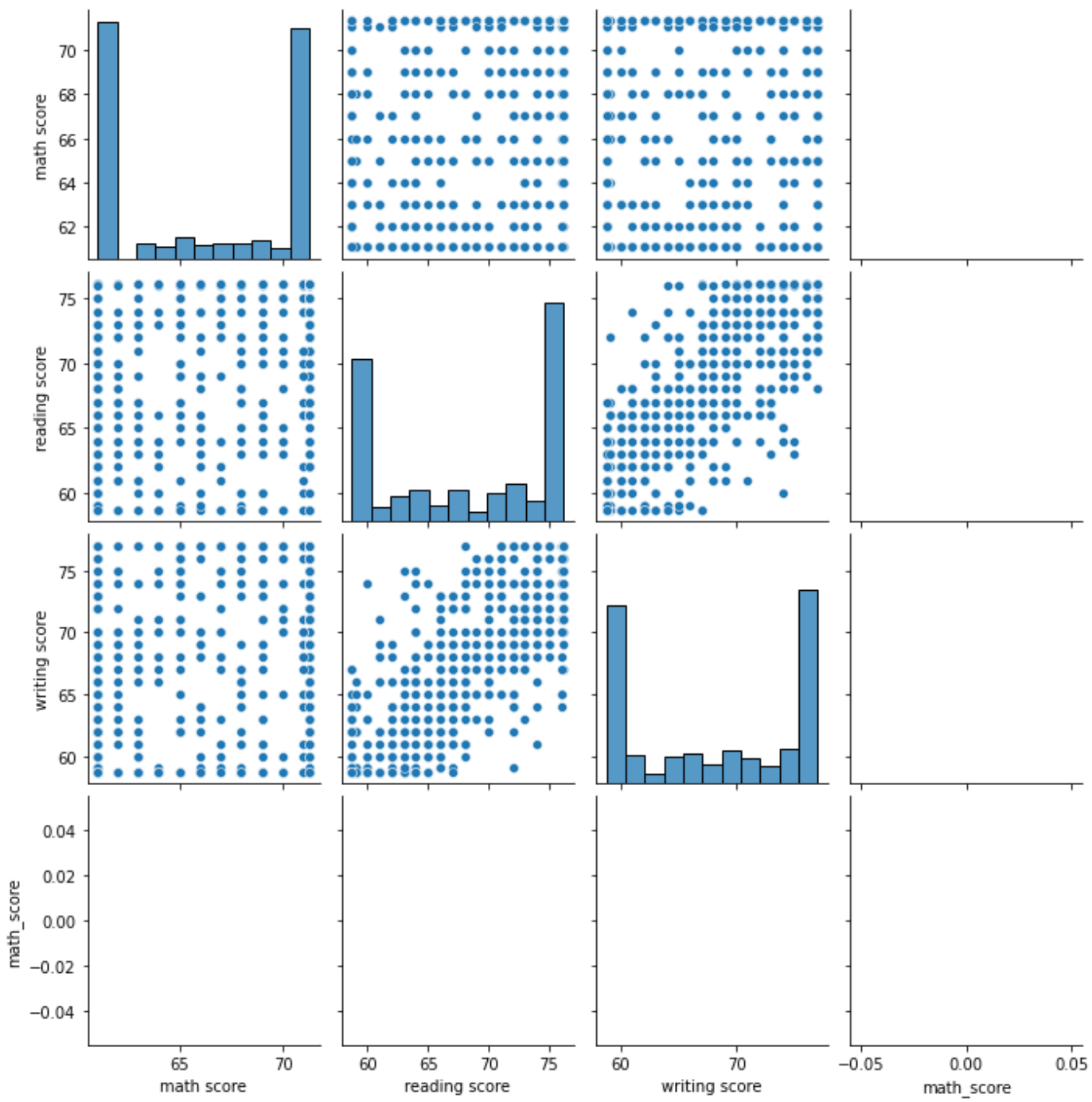
	math score	reading score	writing score	math_score
math score	1.000000	0.716269	0.681605	NaN
reading score	0.716269	1.000000	0.921747	NaN
writing score	0.681605	0.921747	1.000000	NaN
math_score	NaN	NaN	NaN	NaN

In [194…
```
sns.heatmap(data_num.corr(), annot=True, cmap='icefire', linewidths=0.3)
fig=plt.gcf()
fig.set_size_inches(15,10)
plt.title("corr between variable", color='black', size=25)
plt.show()
```



In [195…
```
sns.pairplot(data_num)
```

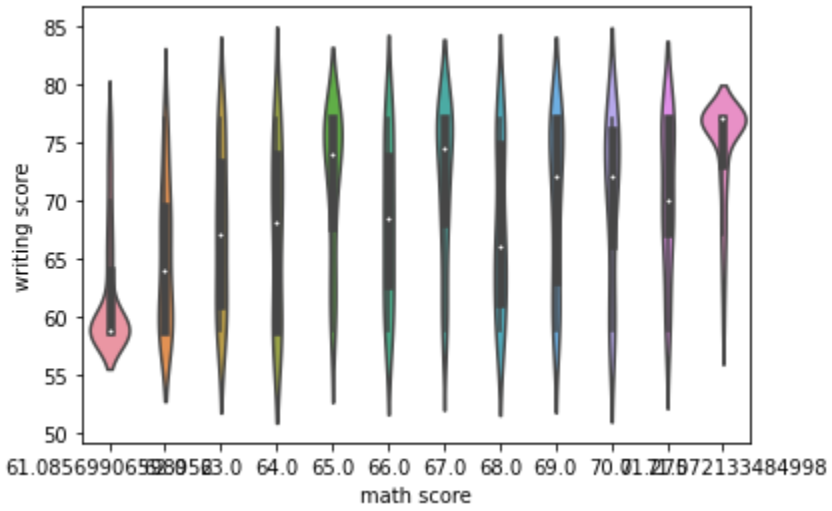
Out[195]:
```
<seaborn.axisgrid.PairGrid at 0x1cade4a4250>
```



In [196…
```
# For categorical data we do count plot, encoding
```

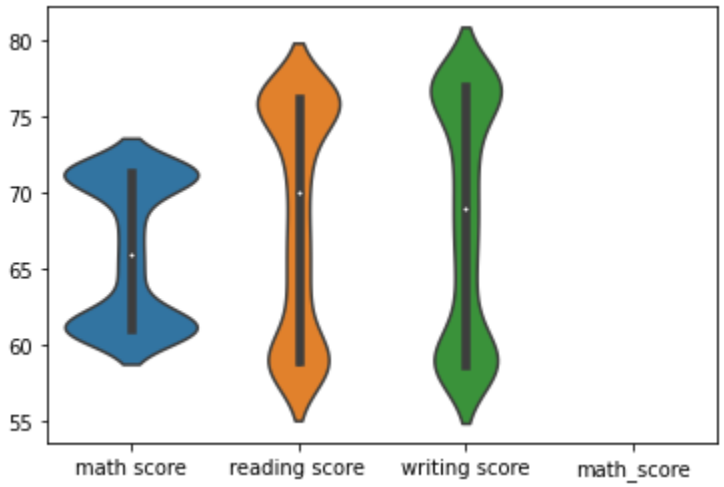
In [199…
```
sns.violinplot(data=data, x= data_num['math score'], y=data_num['writing score'])
```

Out[199]:
```
<AxesSubplot:xlabel='math score', ylabel='writing score'>
```



In [198…
```
sns.violinplot(data=data_num)
```

Out[198]:
```
<AxesSubplot:>
```



In [ ]:
```

```