EDA Concepts Practical Implementation

In [11]:
```
#comment
#observations
```

In [10]:
```
## Importing All Necessary Library Required

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 [12]:
```
## Loading Dataset in Jupyter NOtebook

data = pd.read_csv('C:\\Users\\Lenovo\\Desktop\\DATA SCIENCE Project\\Dataset\\student.
```

In [13]:
```
# head() is use to show top 5 Rows from dataset
data.head()
```

Out[13]:

	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 [14]:
```
## Tail() is use to show Bottom 5 Row from dataset
data.tail()
```

Out[14]:

	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 [16]:
```
## To check all Rows And Columns available in shape
data.shape
```

Out[16]:
```
(1000, 8)
```

In [17]:
```
## to check all basic 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 [18]:
```
## Checking the data Type of Gender Column.
data['gender'].dtypes
```

Out[18]:
```
dtype('O')
```

In [21]:
```
## Checking the data Type of Gender Column in boolean value.
data['gender'].dtypes== 'O'
```

Out[21]:
```
True
```

In [22]:
```
 ## Extracting All Column from Dataset.
data.columns
```

Out[22]:
```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course', 'math score', 'reading score',
       'writing score'],
      dtype='object')
```

In [23]:
```
## Extracting all Columns from Lists Comprihensions
[fea for fea in data.columns]
```

Out[23]:
```
['gender',
 'race/ethnicity',
 'parental level of education',
 'lunch',
 'test preparation course',
 'math score',
```

```
'reading score',
'writing score']
```

In [26]:
```
## Extracting Categorical Column by Lists comprihensions
cat_col = [fea for fea in data.columns if data[fea].dtypes == 'O']
```

In [28]:
```
## Extracting Numerical Column by Lists comprihensions
num_col = [fea for fea in data.columns if data[fea].dtypes != 'O']
```

In [29]:
```
## Displaying All Numerical Column with data
data[num_col]
```

Out[29]:

	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 [30]:
```
## Displaying All Categorical Column with data
data[cat_col]
```

Out[30]:

	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

	gender	race/ethnicity	parental level of education	lunch	test preparation course
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 [31]:
```
## To check how much memory our data is consuming.
data.memory_usage()
```

Out[31]:
```
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 [32]:
```
 ## To Check the Null Value is Present in data or not.
data.isnull()
```

Out[32]:

	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 [33]:
```
## To check Sum of all Null Value present in dataset with respect to every columns.
data.isnull().sum()
```

Out[33]:
```
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 [34]:
```
## There is No Null value in our Dataset
data.isnull().sum().sum()
```

Out[34]:
```
0
```

In [35]:
```
## To check Duplicate Value is present or not in dataset.
data.duplicated()
```

Out[35]:
```
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 [36]:
```
## There is no Duplicate Value in my dataset
data.duplicated().sum()
```

Out[36]:
```
0
```

In [37]:
```
## It will give unique value with respect with each column
data.nunique()
```

Out[37]:
```
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 [38]:
```
## There are two unique  value in gender column
data['gender'].unique()
```

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

In [39]:
```
## To check all Statistics value of dataset
data.describe()
```

Out[39]:

	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 [40]:
```
## To Check Statistic value in Transpose Sequence of dataset
data.describe().T
```

Out[40]:

	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 [41]:
```
## To Check Correlation in the Data.
data.corr()
```

Out[41]:

	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 [42]:
```
## To check Relation with the two data is called CoVariance
data.cov()
```

Out[42]:

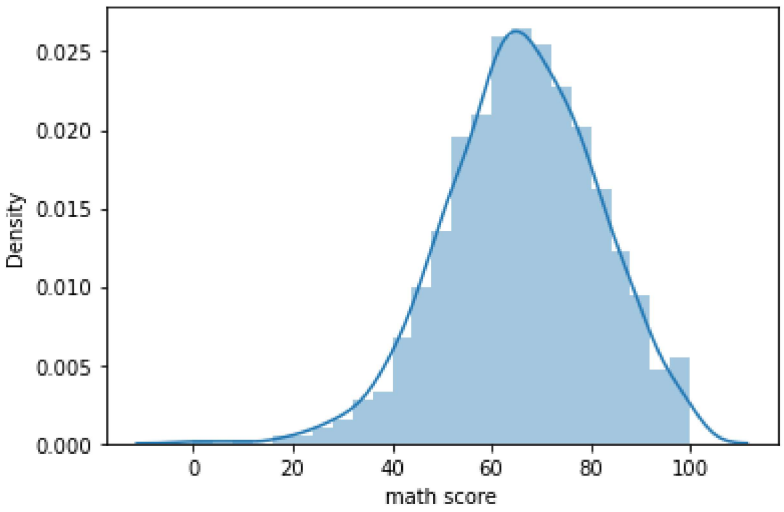
	math score	reading score	writing score
math score	229.918998	180.998958	184.939133
reading score	180.998958	213.165605	211.786661
writing score	184.939133	211.786661	230.907992

In [43]:
```
 ## To Check the Skewness Distribution of the Data.
data.skew()
```

Out[43]:
```
math score      -0.278935
reading score   -0.259105
writing score   -0.289444
dtype: float64
```

In [44]:
```
 ## This is Left Screw Distribution with the help of Distribution Plotting
sns.distplot(data['math score'])
```

Out[44]:
```
<AxesSubplot:xlabel='math score', ylabel='Density'>
```



In [45]:
```
## for Checking all columns in dataset
data.columns
```

Out[45]:
```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course', 'math score', 'reading score',
       'writing score'],
      dtype='object')
```

In [49]:
```
## Adding a New Column name as Average & Performing Average of all Numeric Data.
data['Average']= (data['math score'] + data['reading score'] + data['writing score'])/3
```

In [50]:
```
## Checking the Average of the Numeric dataset
data['Average']
```

Out[50]:
```
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 [51]:
```
## Checking that New Column 'Average' is added in the last of the dataset.
data.head()
```

Out[51]:

	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 [52]:
```
## Using Groupby() with gender to find mean of data.
data.groupby('gender').mean()
```

Out[52]:

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 [53]:
```
## Using Groupby() with gender to find count of data.
data.groupby('gender').count()
```

Out[53]:

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 [54]:
```
# Showing Top 3 value of dataset
data.head(3)
```

Out[54]:

	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

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

In [55]:
```
## Doing Comparision with math score column dataset
data[data['math score']< 30]
```

Out[55]:

	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
91	male	group C	high school	free/reduced	none	27	34	36	32.333333
145	female	group C	some college	free/reduced	none	22	39	33	31.333333
327	male	group A	some college	free/reduced	none	28	23	19	23.333333
338	female	group B	some high school	free/reduced	none	24	38	27	29.666667
363	female	group D	some high school	free/reduced	none	27	34	32	31.000000
466	female	group D	associate's degree	free/reduced	none	26	31	38	31.666667
528	female	group D	bachelor's degree	free/reduced	none	29	41	47	39.000000
601	female	group C	high school	standard	none	29	29	30	29.333333
683	female	group C	some high school	free/reduced	completed	29	40	44	37.666667
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 [56]:
```
 ## For Counting the number count() is used in this data comparision .
data[data['math score']< 30].count()
```

Out[56]:
```
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 [57]:
```
 # checking all Column name
data.columns
```

Out[57]:
```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course', 'math score', 'reading score',
       'writing score', 'Average'],
      dtype='object')
```

In [59]:
```
## Storing all Numeric data in data_num variable
data_num = data[num_col] `
```

In [60]:
```
 ## showing Top 5 Row from Data_num dataset
data_num.head()
```

Out[60]:

	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 [61]:
```
## Importing Scipy.stats Library
from scipy.stats import normaltest
```

In [62]:
```
 ## Checking the Statistics value & P value from NormalTest()
normaltest(data_num['math score'])
```

Out[62]:
```
NormaltestResult(statistic=15.408960513931822, pvalue=0.00045080293869937836)
```

In [63]:
```
## it is non normal Distributed
normaltest(data_num['math score'])[1]*100
```

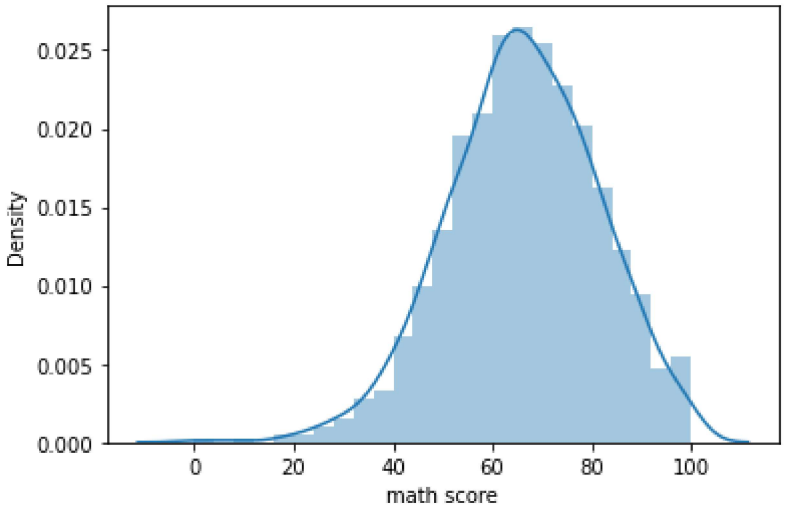
Out[63]:
```
0.04508029386993784
```

In [ ]:
```
## Rules with P-value for checking Distribution is Normal Distribution or NOt
```

```
if p >0.05 then my data will be Normal Distributed.
if p <0.005 then my data be non normally Distributed
```

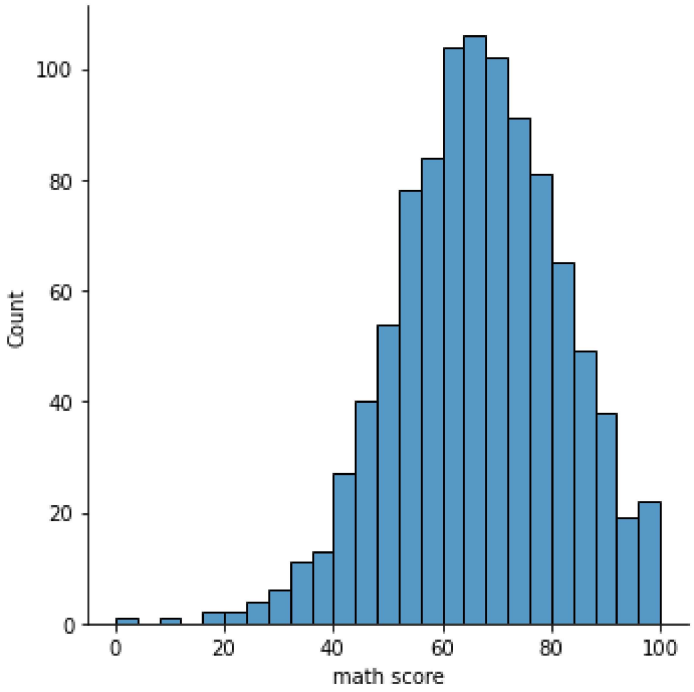
In [64]:
```
## Ploting distribution plot for math score data
sns.distplot(data_num['math score'])
```

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



In [65]:
```
## Distribution Ploting of 'math score column'
sns.displot(data_num['math score'])
```

Out[65]:
```
<seaborn.axisgrid.FacetGrid at 0x21277d43460>
```

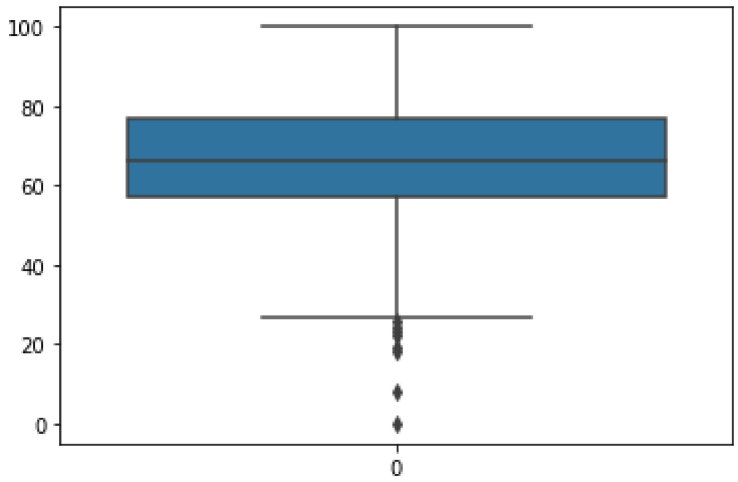


Outlier

In [66]:
```
## Box Ploting of 'math score column' for checking outlier
```

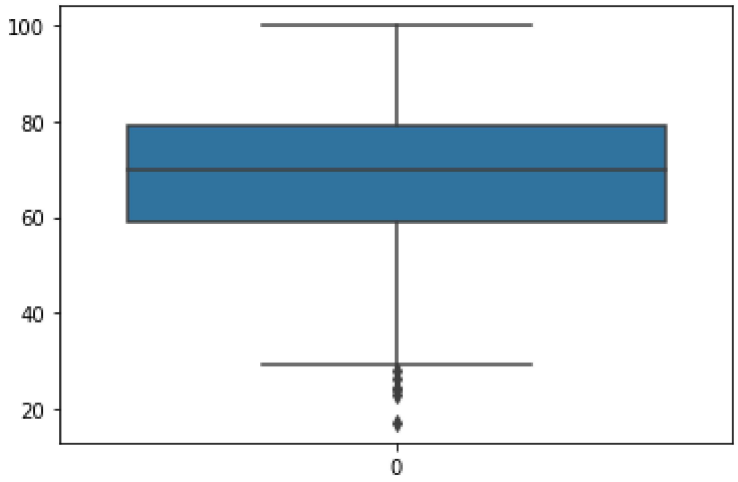
```
sns.boxplot(data= data['math score'])
```

Out[66]: <AxesSubplot:>



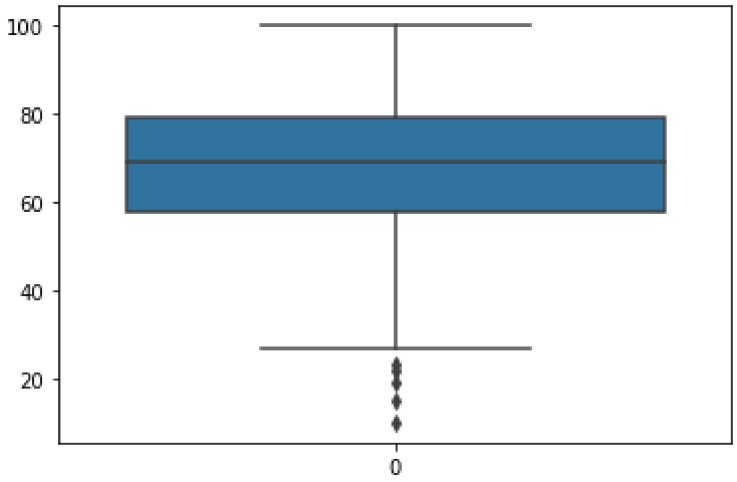
In [67]:
```
## Box Ploting of 'reading score column' for checking outlier
sns.boxplot(data= data['reading score'])
```

Out[67]: <AxesSubplot:>



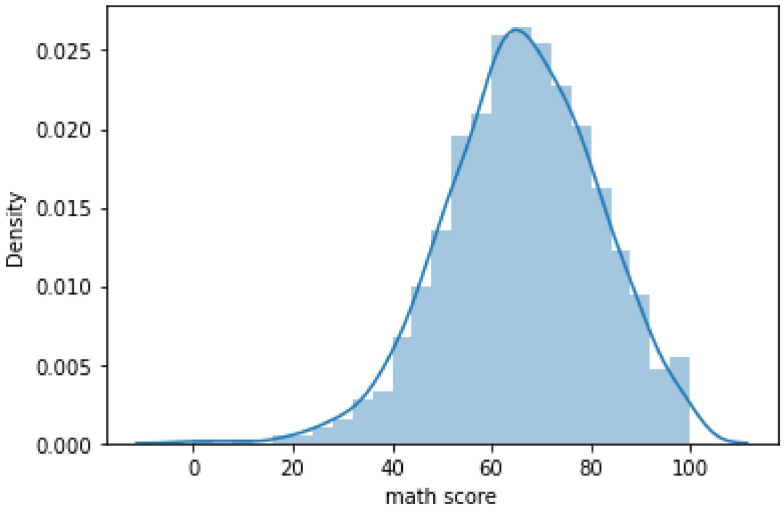
In [68]:
```
 ## Box Ploting of 'writing score column' for checking outlier
sns.boxplot(data= data['writing score'])
```

Out[68]: <AxesSubplot:>



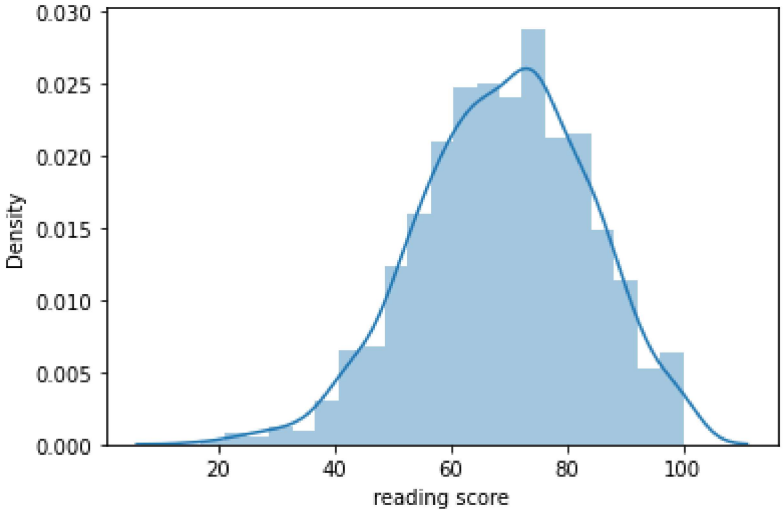
In [69]:
```
 ## Distribution Plotting of 'math score columns'
sns.distplot(data_num['math score'])
```

Out[69]:
```
<AxesSubplot:xlabel='math score', ylabel='Density'>
```



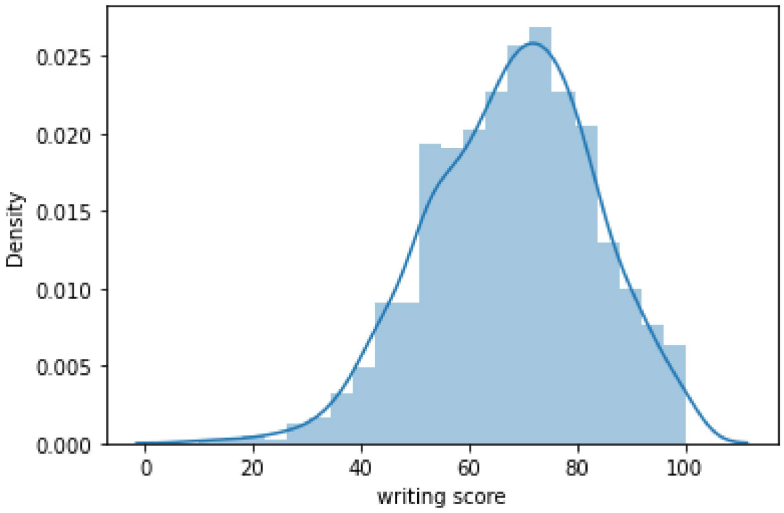
In [80]:
```
## Distribution Plotting of 'reading score columns'
sns.distplot(data_num['reading score'])
```

Out[80]:
```
<AxesSubplot:xlabel='reading score', ylabel='Density'>
```



In [72]:
```
## Distribution Plotting of 'writing score columns'
sns.distplot(data_num['writing score'])
```

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



In [ ]:
```
## To Check Outlier we are using IQR function
```

In [97]:
```
 ## Checking the Value for quantile-1( q1) for data
q1 = data['math score'].quantile(0.25)
```

In [98]:
```
## Checking the Value for quantile-3( q3) for data
q3 = data['math score'].quantile(0.75)
```

In [99]:
```
## Checking IQR (Inter Quantitle range) for the data with the formula
IQR  = q3 - q1
```

In [100…
```
 ## showing value of IQR
IQR
```

Out[100… 20.0

In [101…
```
## Calculating Upper_limit or upper fence
upper_limit = q3 + (1.5 * IQR)
```

In [102…
```
## Calculating lower_limit or lower fence
lower_limit = q1 -(1.5 * IQR)
```

In [103…
```
## Displaying upper_limit value
upper_limit
```

Out[103… 107.0

In [104…
```
## Displaying lower_limit value
lower_limit
```

Out[104… 27.0

In [119…
```
##  Checking Outlier with the help of lower_limit condition.
data_outlier = data[data['math score'] < lower_limit]
```

In [120…
```
## Displaying the data_outlier value
data_outlier
```

Out[120…

	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
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 [121…
```
## Displaying all numeric data which are store in data_num
data_num
```

Out[121…

	math score	reading score	writing score
0	72.0	72	74.0
1	69.0	90	88.0
2	90.0	95	93.0
3	47.0	57	44.0
4	76.0	78	75.0
...	...	...	...
995	88.0	99	95.0
996	62.0	55	55.0
997	59.0	71	65.0
998	68.0	78	77.0
999	77.0	86	86.0

1000 rows × 3 columns

In [106…

```
##  Checking Outlier with the help of upper_limit condition.
data[data['math score'] > upper_limit]
```

Out[106…

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

In [79]:

```
 ## Calculating 100 Quantile of Math score data
data['math score'].quantile(1.00)
```

Out[79]:

```
100.0
```

In [76]:

```
 ## Calculating Minimum value from math score data
data['math score'].min()
```

Out[76]:

```
0
```

In [78]:

```
 ## Calculating maximum value from math score data
data['math score'].max()
```

Out[78]:

```
100
```

In [107…

```
 ## Showing All Unique marks obtained in Math score columns
data['math score'].unique()
```

Out[107…

```
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 [156…
```
## Showing all Numeric column
data_num.columns
```

Out[156…
```
Index(['math score', 'reading score', 'writing score'], dtype='object')
```

In [108…
```
## Definning Outlier_threshold function for finding upper_limit & lower_limit

def outlier_threshold(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 lower_limit,upper_limit
```

In [114…
```
## Defining replace_with_threshold function for finding upper_limit & lower_limit with

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

In [115…
```
# replace_with_threshold(data_num,data_num.columns)
```

In [117…
```
## Showing Top 3 Row of dataset
data.head(3)
```

Out[117…

	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

In [118…
```
# data_num.loc[data_num['math score'] <Lower_limit,'math score'] = Lower_limit
```

Graph Ananlysis

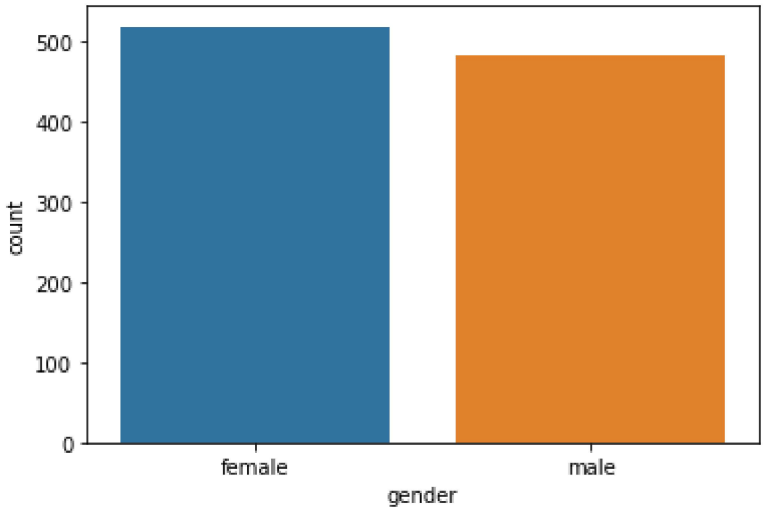
In [122…
```
## Showing top 5 Rows of Dataset
data.head()
```

Out[122…

	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

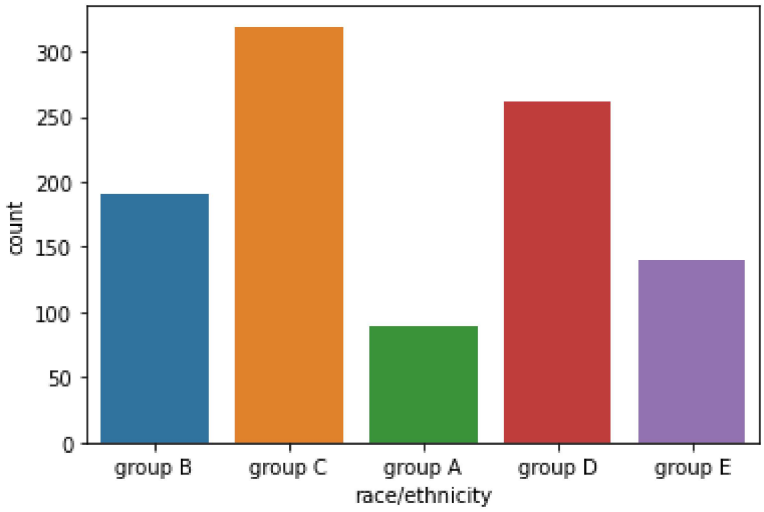
In [123…
```
## Plotting Countplot graph with Respect to gender Column.
sns.countplot(data['gender'])
```

Out[123… `<AxesSubplot:xlabel='gender', ylabel='count'>`



In [124…
```
## count-plot graph is useful for identifing Imbalanced data in dataset
sns.countplot(data['race/ethnicity'])
```

Out[124… `<AxesSubplot:xlabel='race/ethnicity', ylabel='count'>`



In [126…
```
## checking Mean of gender column with the help of Groupby()
df = data.groupby('gender').mean()
```

In [155…
```
## Checking Mean value of Df Dataframe
df
```

Out[155…

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 [127…
```
## Checking Average of gender data
df['Average']
```

Out[127…
```
gender
female    69.569498
male      65.837483
Name: Average, dtype: float64
```

In [129…
```
## Checking the Average of only Female Data
df['Average'][0]
```

Out[129…
```
69.56949806949807
```

In [130…
```
## Checking the Average of only Male Data
df['Average'][1]
```

Out[130…
```
65.8374827109267
```

In [131…
```
## Checking the Mean of Math Score of Gender data
df['math score']
```

```
gender
```

Out[131…
```
female    63.633205
male      68.728216
Name: math score, dtype: float64
```

In [132…
```
## Checking the Mean of Math Score of only female data
df['math score'][0]
```

Out[132…
```
63.633204633204635
```

In [133…
```
## Checking the Mean of Math Score of only Male data
df['math score'][1]
```

Out[133…
```
68.72821576763485
```

In [134…
```
## Storing Average & Mean ('math score') Value of only female data in female_score
female_score = df['Average'][0],df['math score'][0]
```

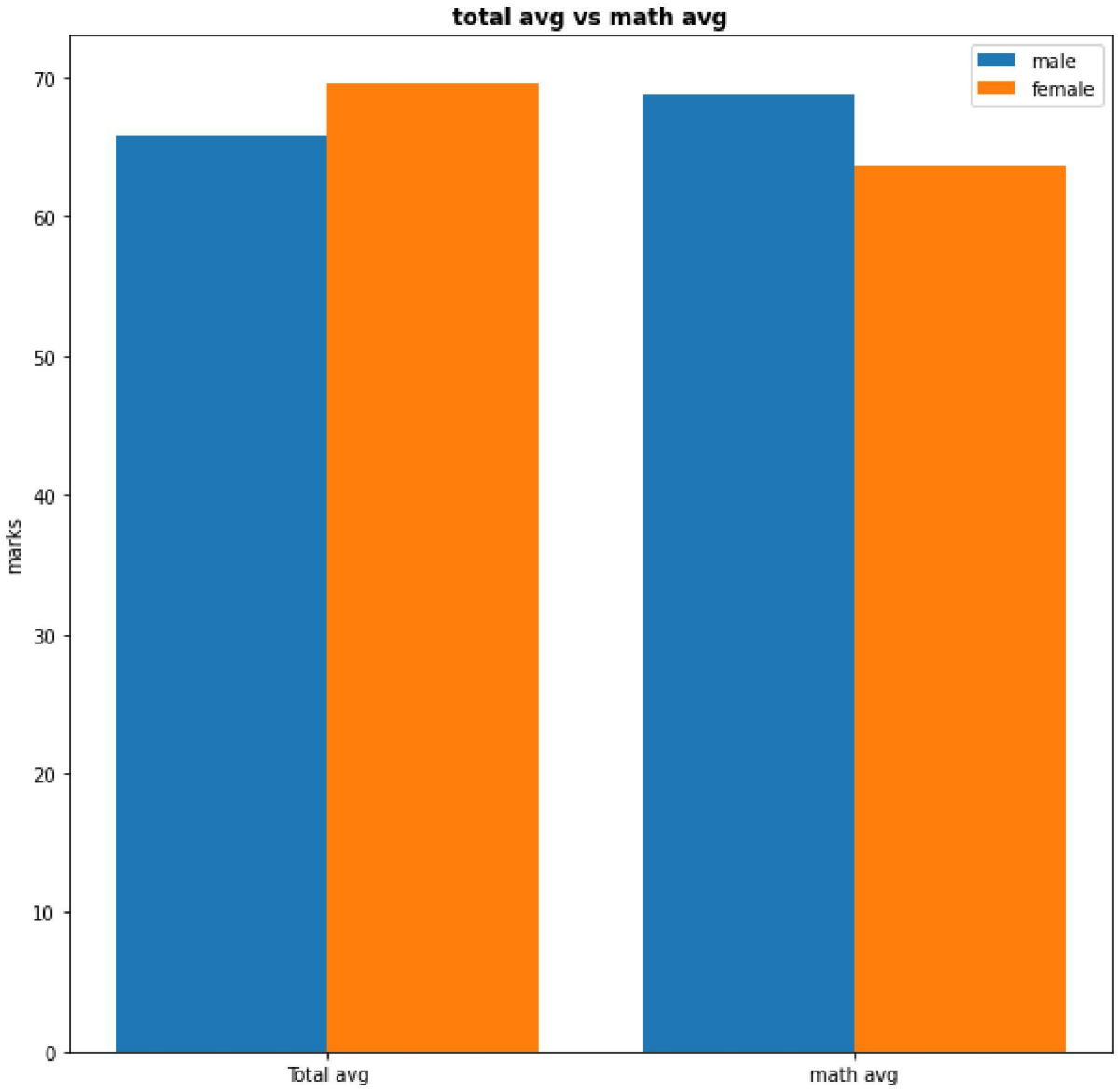
In [135…
```
## Displaying Average & Mean Value which is store in female_score variable.
female_score
```

Out[135…
```
(69.56949806949807, 63.633204633204635)
```

In [139…
```
 ## Plotting Bar chart with Respect to Dataset

plt.figure(figsize=(10,10))
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()
```



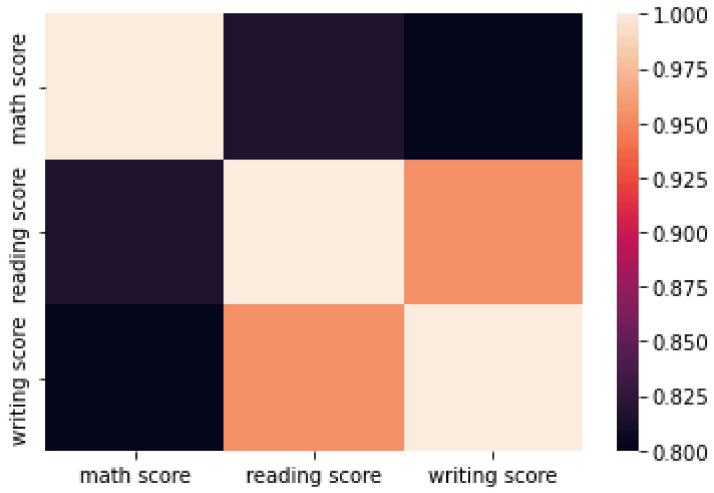
In [144…
```
## Checking Correlation with dataset
data_num.corr()
```

Out[144…

	math score	reading score	writing score
math score	1.000000	0.815727	0.799954
reading score	0.815727	1.000000	0.954040
writing score	0.799954	0.954040	1.000000

In [145…
```
## Plotting Heatmap for Correlation() with dataset
sns.heatmap(data_num.corr())
```

Out[145… <AxesSubplot:>



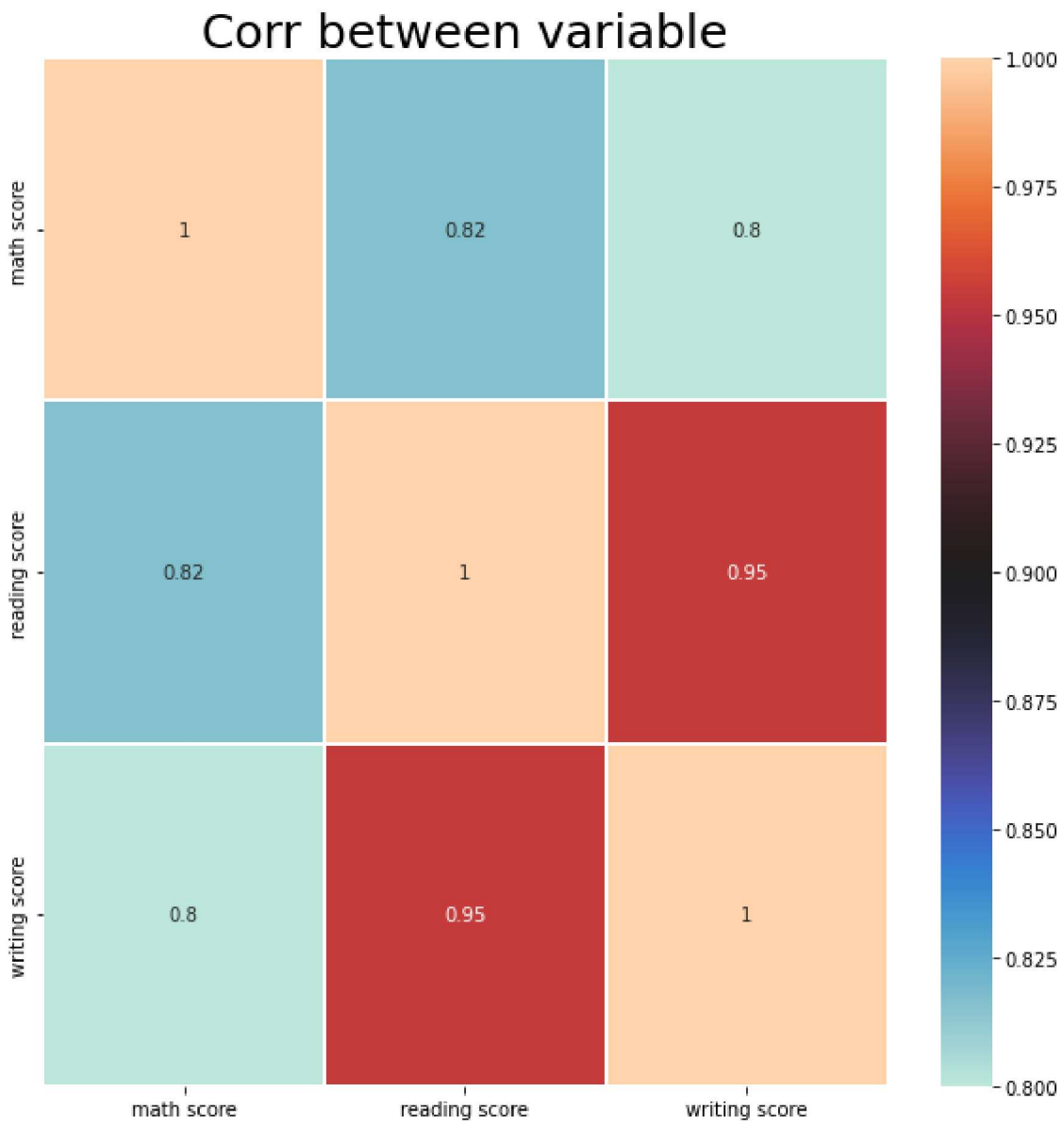
In [146…

```
## Plotting Heatmap with respect to Correlation between Variable.

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 =25)
plt.show
```

Out[146…

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

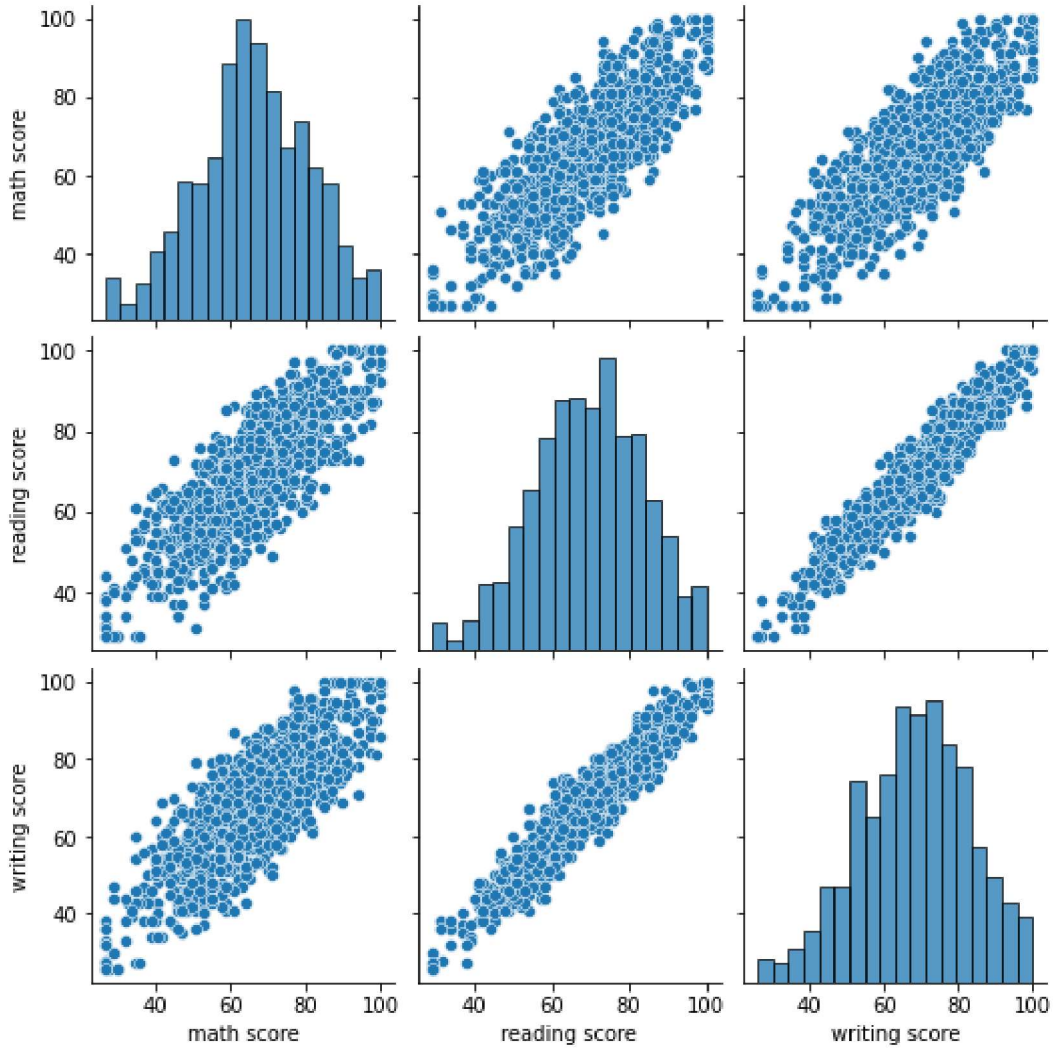


In [151…

```
## Plotting Pair-plot for data_num dataset
sns.pairplot(data=data_num)
```

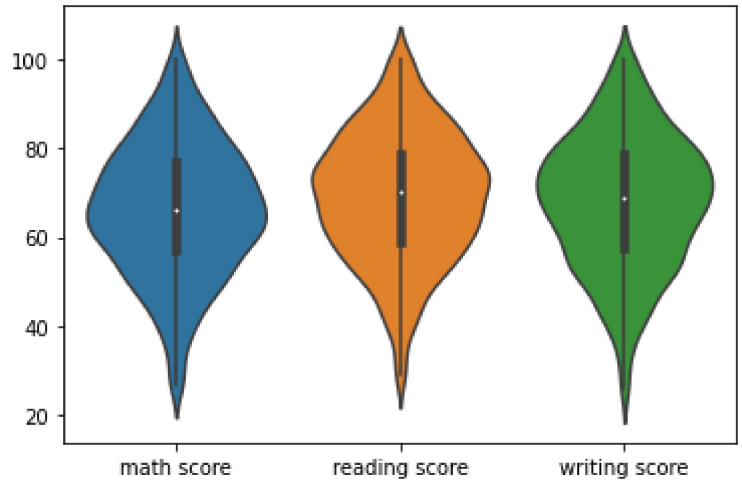
Out[151…

```
<seaborn.axisgrid.PairGrid at 0x2127ec0be20>
```



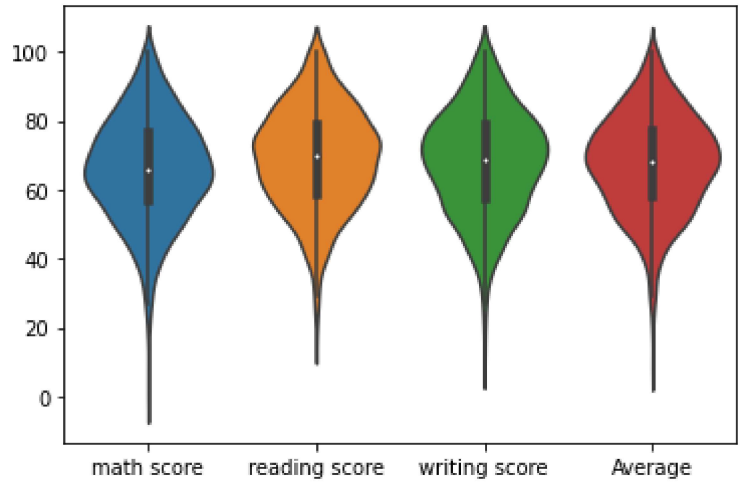
In [148…
```
## Plotting ViolinPlot for only data_num dataset
sns.violinplot(data= data_num)
```

Out[148…
```
<AxesSubplot:>
```



In [150…
```
## Plotting ViolinPlot for whole dataset
sns.violinplot(data= data)
```

Out[150… <AxesSubplot:>



In [152…
```
## Importing numpy Library
import numpy as np
data_num
```

Out[152…

	math score	reading score	writing score
0	72.0	72	74.0
1	69.0	90	88.0
2	90.0	95	93.0
3	47.0	57	44.0
4	76.0	78	75.0
...	...	...	...
995	88.0	99	95.0
996	62.0	55	55.0
997	59.0	71	65.0
998	68.0	78	77.0
999	77.0	86	86.0

1000 rows × 3 columns

In [153…
```
 ## for Checking Natural Log for 'Math score'  column
np.log(data_num['math score'])
```

Out[153…
```
0      4.276666
1      4.234107
2      4.499810
3      3.850148
4      4.330733
         ...
995    4.477337
996    4.127134
997    4.077537
```

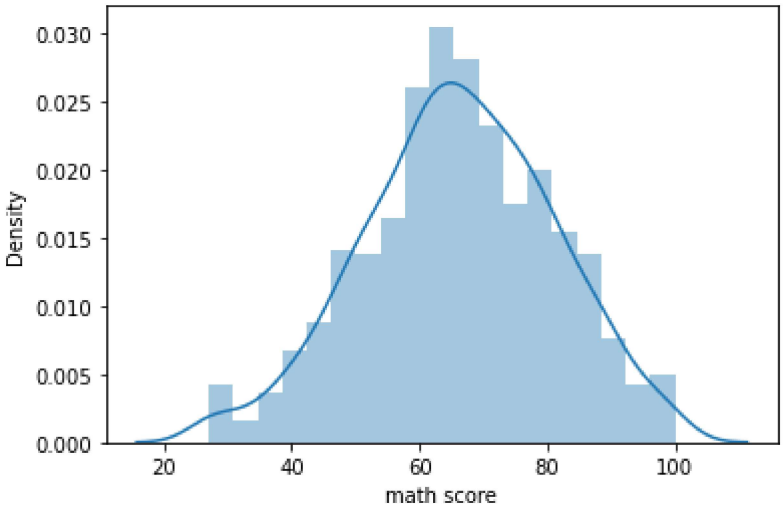
```
998    4.219508
999    4.343805
Name: math score, Length: 1000, dtype: float64
```

In [154…

```
## Distribution Plotting for math_score column data.
sns.distplot(data_num['math score'])
```

Out[154…

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



Homework for EDA & Preprocessing Tasks:- 10 dataset pick anyone perform detail EDA in one ipynb perform missing value handle if it is there in one ipynb all the method(10-12) perform outlier handle if it is there in one ipynb all the method(10-12) perform encoding if it is required in one ipynb all the method(10-12) perform scaling one ipynb all the method(10-12) perform feature selection in one ipynb all the method(10-12) perform transformation one ipynb all the method(10-12) You can Refer :- book, different artical,research paper, Google, Youtube, GitHub Resource :- https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing Submit Date :- friday (30th September,2022) till 12AM ist sunny.savita@ineuron.ai krish.naik@ineuron.ai Note :- All 10 Dataset you can solve it and you can keep solution with GitHub.

Thank You