Importing Required Library

In [1]:

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

Importing students Score dataset

In [2]:

```
data=pd.read_csv('student.csv')
```

In [3]:

```
data
```

Out[3]:

	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

Viewing first five rows of Data

In [4]:

```
1 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

Viewing last five rows of Data

In [5]:

```
1 data.tail()
```

Out[5]:

	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

Shape of Data

In [6]:

```
1 data.shape
```

Out[6]:

```
(1000, 8)
```

Fetching Information of Data

In [7]:

```
1 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 [8]:

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

Out[8]:

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

In [9]:

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

Out[9]:

```
True
```

Fetching all Columns available in Data

In [10]:

```
1 data.columns
```

Out[10]:

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

Categorical value

In [11]:

```
1 cat_col = [fea for fea in data.columns if data[fea].dtype == 'O']
```

Numerical value

In [12]:

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

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]:

```
1 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

Check memory usage

In [16]:

```
1 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]:

```
1 data.isnull()
```

Out[17]:

	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

To check null value

In [17]:

```
1 data.isnull().sum().sum()
```

Out[17]:

```
0
```

To check duplicate value

In [18]:

```
1 data.duplicated().sum() #so there is no duplicate value in my data
```

Out[18]:

```
0
```

to check unique value where n indicate number

In [18]:

```
1 data.nunique()
```

Out[18]:

```
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 [20]:

```
1 data['gender'].unique()
```

Out[20]:

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

Describe data with respect to statistics

In [19]:

```
1 data.describe()
```

Out[19]:

	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 transform mode

Where T is transpose

In [22]:

```
1 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

To find Correlation

In [21]:

```
1 data.corr()
```

Out[21]:

	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

To find covariance

In [24]:

```
1 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

skewness of data

In [25]:

```
1 data.skew()
```

Out[25]:

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

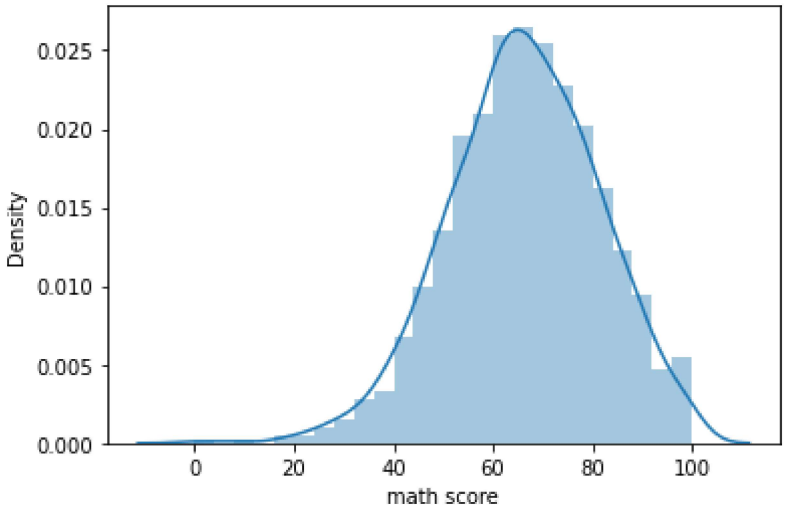
Plotting of Data

In [26]:

```
1 sns.distplot(data['math score'])
```

Out[26]:

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



To find the average score

In [22]:

```
1 (data['math score']+data['reading score']+data['writing score'])/3
```

Out[22]:

```
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
```

Adding averagecolumn in dataset

In [25]:

```
1 data['Average']= (data['math score']+data['reading score']+data['writing score'])/3
```

In [26]:

```
1 data.head()
```

Out[26]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Averag
0	female	group B	bachelor's degree	standard	none	72	72	74	72.6666
1	female	group C	some college	standard	completed	69	90	88	82.3333
2	female	group B	master's degree	standard	none	90	95	93	92.6666
3	male	group A	associate's degree	free/reduced	none	47	57	44	49.3333
4	male	group C	some college	standard	none	76	78	75	76.3333

Using groupby operation with mean

In [28]:

```
1 data.groupby('gender').mean()
```

Out[28]:

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 [29]:

```
1 data.groupby('gender').count()
```

Out[29]:

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

Question: you have to find out no of student whoever is having less than 30 marks

In [32]:

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

Out[32]:

```
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
```

Scipy.org for statistics lib

p value if p > 0.05 then the data will be normal distributed

In [37]:

```
1 data.columns
```

Out[37]:

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

In [38]:

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

In [39]:

```
1 data_num.head()
```

Out[39]:

	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 [40]:

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

Getting P value

In [41]:

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

Out[41]:

```
0.04508029386993784
```

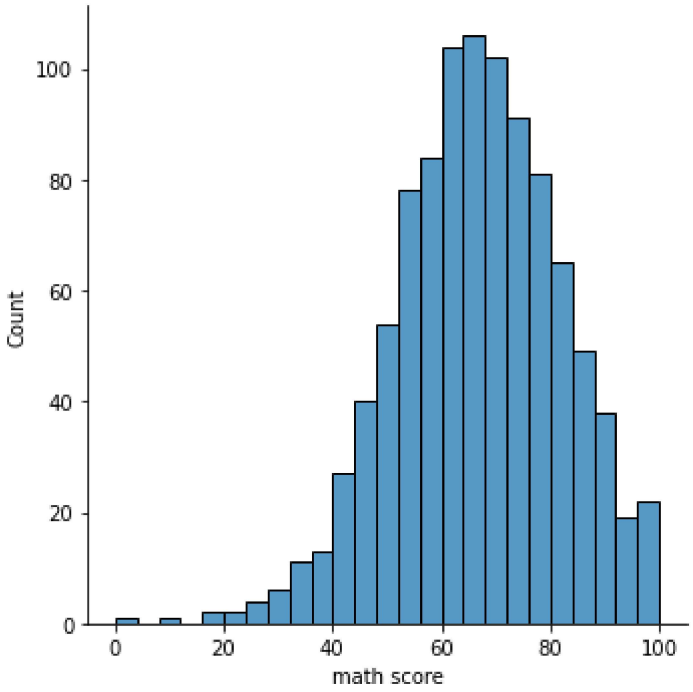
if p > 0.05 then my data will be normal distributed

In [42]:

```
1 sns.displot(data_num['math score'])
```

Out[42]:

```
<seaborn.axisgrid.FacetGrid at 0x1f37673ef10>
```



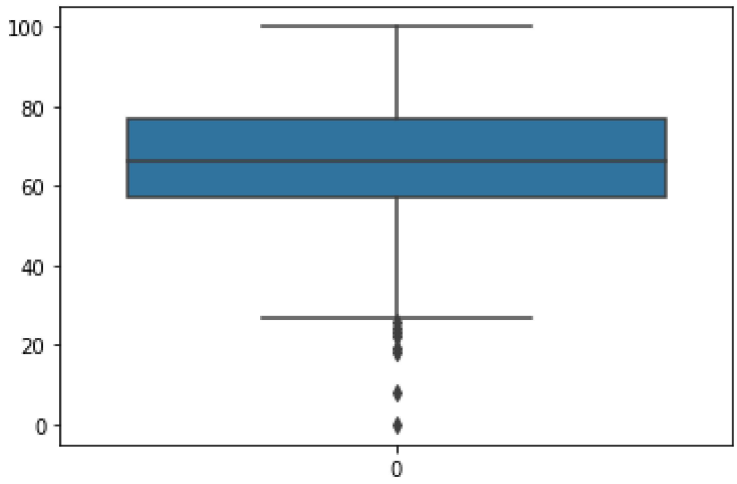
Outlier

In [44]:

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

Out[44]:

```
<AxesSubplot:>
```

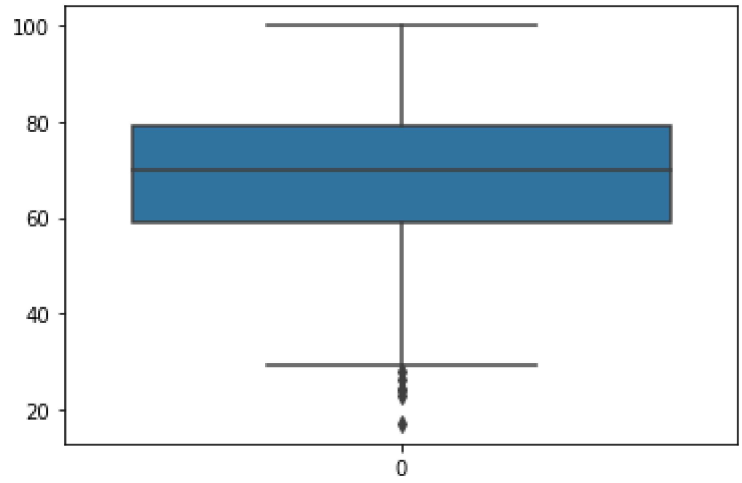


In [49]:

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

Out[49]:

<AxesSubplot:>



Quantile

In [50]:

```
1 q1 = data['math score'].quantile(0.25)
```

In [51]:

```
1 q3 = data['math score'].quantile(0.75)
```

Inter Quantile Range

In [52]:

```
1 IQR= q3-q1
```

In [53]:

```
1 IQR
```

Out[53]:

20.0

In [54]:

```
1 upper_limit=q3+(1.5*IQR)
```

In [55]:

```
lower_limit=q1-(1.5*IQR)
```

In [56]:

```
upper_limit
```

Out[56]:

107.0

In [57]:

```
lower_limit
```

Out[57]:

27.0

In [58]:

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

Out[58]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Ave
17	female	group B	some high school	free/reduced	none	18	32	28	26.00(
59	female	group C	some high school	free/reduced	none	0	17	10	9.00(
145	female	group C	some college	free/reduced	none	22	39	33	31.33
338	female	group B	some high school	free/reduced	none	24	38	27	29.66(
466	female	group D	associate's degree	free/reduced	none	26	31	38	31.66(
787	female	group B	some college	standard	none	19	38	32	29.66(
842	female	group B	high school	free/reduced	completed	23	44	36	34.33
980	female	group B	high school	free/reduced	none	8	24	23	18.33

In [59]:

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

Out[59]:

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

In [ ]:

```
1
```

In [60]:

```
1 data['math score'].quantile(1.00)
```

Out[60]:

```
100.0
```

In [61]:

```
1 data['math score'].min()
```

Out[61]:

```
0
```

In [62]:

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

Out[62]:

```
100
```

In [63]:

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

Out[63]:

```
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)
```

Graph Analysis

In [67]:

```
1 data
```

Out[67]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Ave
0	female	group B	bachelor's degree	standard	none	72	72	74	72.66
1	female	group C	some college	standard	completed	69	90	88	82.33
2	female	group B	master's degree	standard	none	90	95	93	92.66
3	male	group A	associate's degree	free/reduced	none	47	57	44	49.33
4	male	group C	some college	standard	none	76	78	75	76.33
...	...	...	...	...	...	...	...	...	
995	female	group E	master's degree	standard	completed	88	99	95	94.00
996	male	group C	high school	free/reduced	none	62	55	55	57.33
997	female	group C	high school	free/reduced	completed	59	71	65	65.00
998	female	group D	some college	standard	completed	68	78	77	74.33
999	female	group D	some college	free/reduced	none	77	86	86	83.00

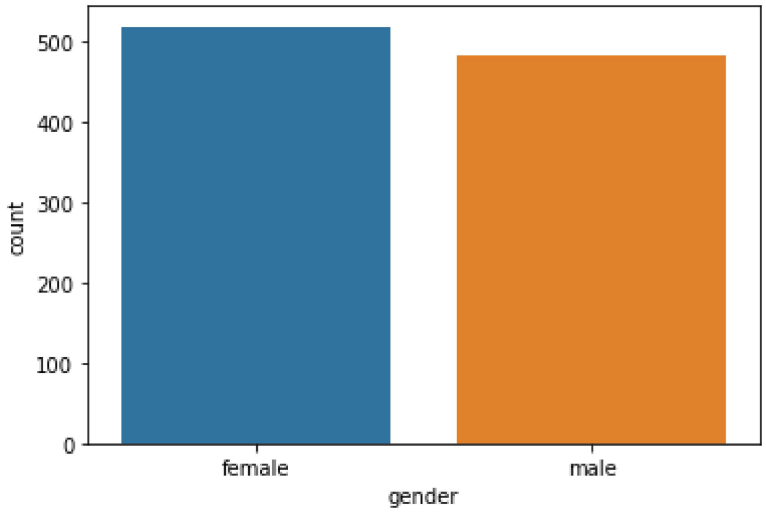
1000 rows × 9 columns

In [68]:

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

Out[68]:

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

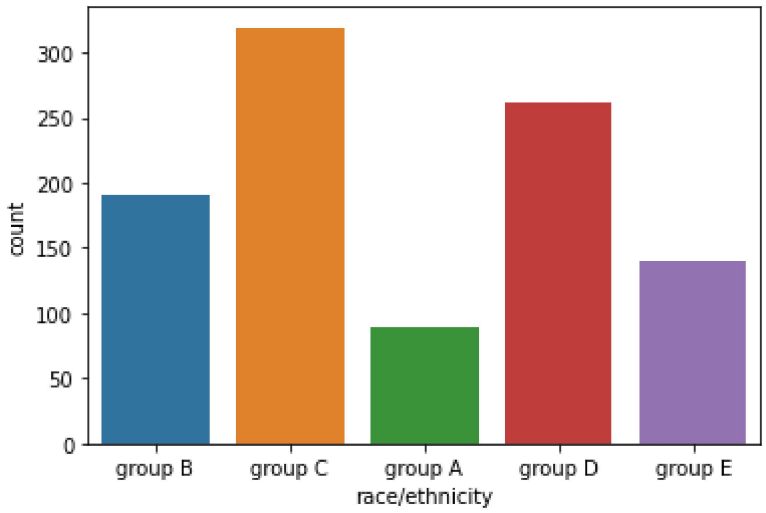


In [69]:

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

Out[69]:

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



In [70]:

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

In [71]:

```
1 df
```

Out[71]:

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 [72]:

```
1 df['Average']
```

Out[72]:

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

In [73]:

```
1 df['Average'][0]
```

Out[73]:

```
69.56949806949807
```

In [74]:

```
1 df['Average'][1]
```

Out[74]:

```
65.8374827109267
```

In [75]:

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

Out[75]:

```
gender
female    63.633205
male      68.728216
Name: math score, dtype: float64
```

In [76]:

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

Out[76]:

```
63.633204633204635
```

In [77]:

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

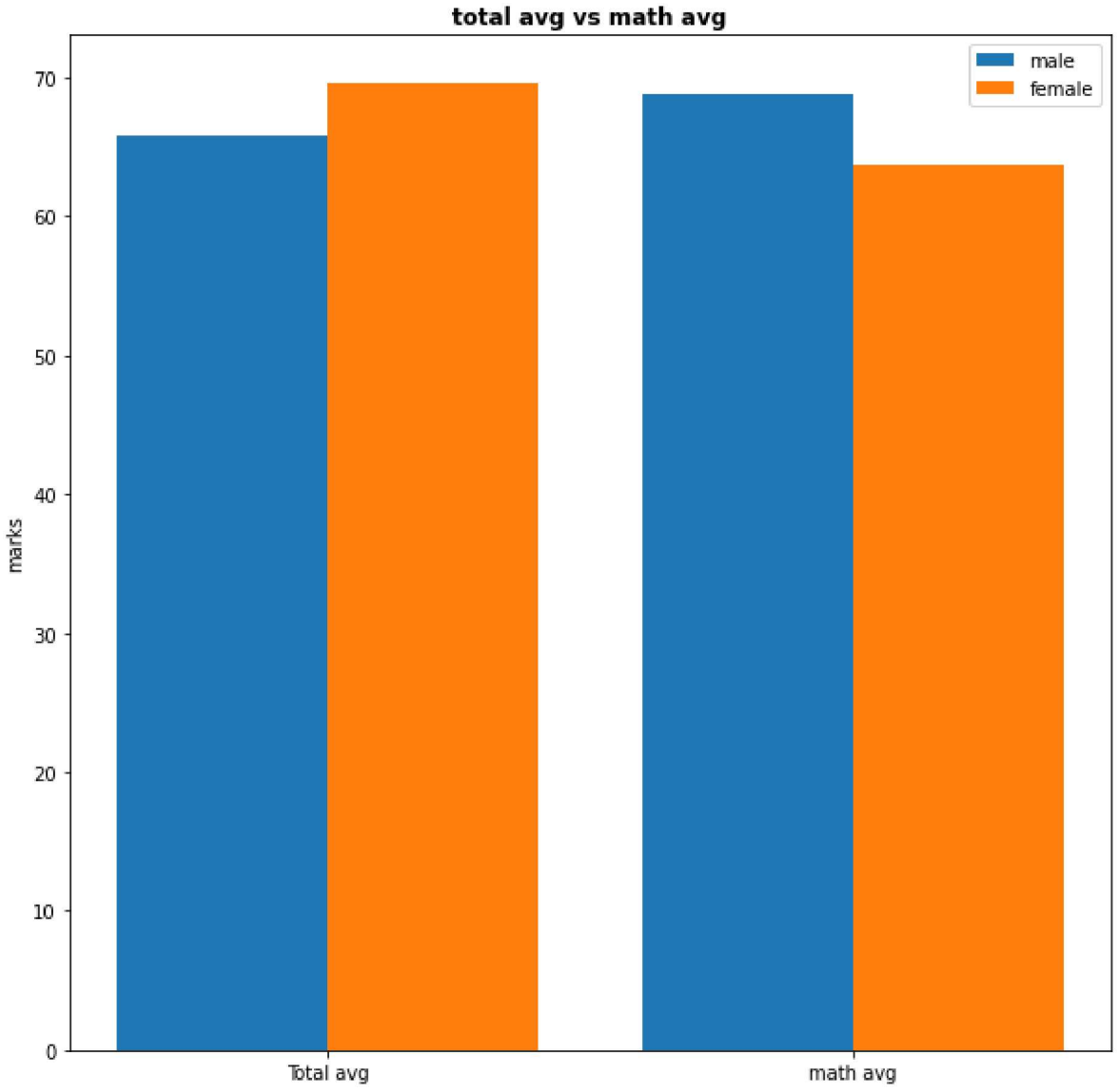
Out[77]:

```
68.72821576763485
```

In [78]:

```
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 [ ]:

```
1
```

In [ ]:

```
1
```