PROJECT REPORT

ANALYSIS ON STUDENT'S PERFORMANCE

Students of The US

SUBMITTED BY:
Ayisha Najeeha C O K

UNDER THE SUPERVISION OF
Shalini Kumari.

Table of Contents

1.Introduction

1.1.Background:

This EDA (Exploratory Data Analysis) is a journey through education, data visualization and exploratory data analysis of students Exam Scores at a public school to gain maximum insights on the test preparation course using Python libraries. EDA is a process of exploring data for analysis purpose. The steps involved in EDA are:

- Preparing the Data: know the dataset, produce all details.
- Cleaning the data for analysis: detect outliers and anomalies if any
- Extract important variables from the data set, statistics of the data
- Visualizations of data : this can be done in any step as per requirement
- conclusion of analysis

1.2.DATA PROVIDER:

Data is provided by the **kaggle** website under **Students Performance Data** set which gives the details of Marks secured by the students in high school , Students from the **United States.**

Data contains information about a student's test preparation course with their "math score ,reading score ,writing score , parent's education and so on "

1.3.PURPOSE:

"Student performance data set" is collected to gain on insight into

1. How effective is the test preparation course?
2. Which major factors contribute to test outcomes?
3. What would be the best way to improve student scores on each test?
4. To understand the influence of parents background on students' performance
5. To understand whether math's score is related to writing and reading score
6. To extract more closer details about the student such as percentage to know the pass and fail details.

1.4SOFTWARE USED:

- JUPYTER NOTEBOOK
- EXCEL

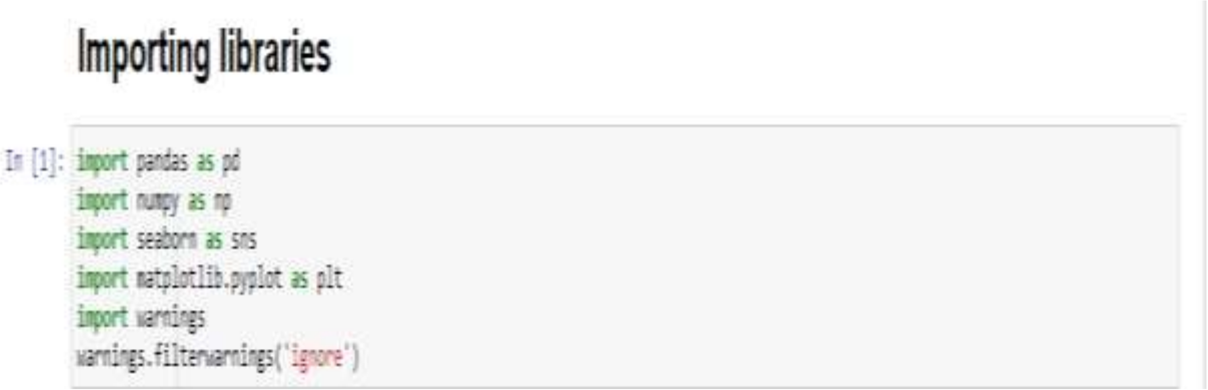
1.5.LIBRARIES USED:

- PANDAS
- NUMPY
- SEABORN
- MATPLOTLIB

2.DATA ANALYSIS : student’s performance

2.1.DATA MINING

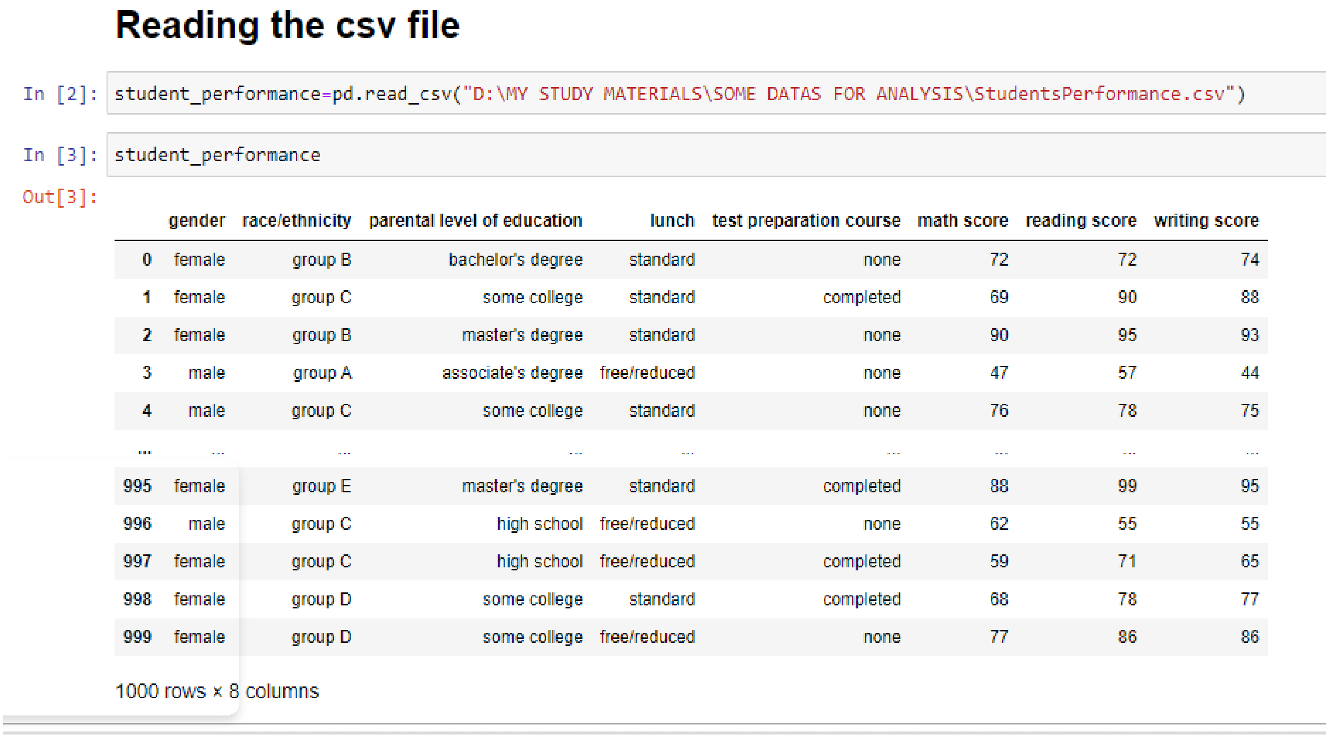
IMPORTING LIBRARIES:



Importing libraries

```
In [1]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        import warnings
        warnings.filterwarnings('ignore')
```

READING THE DATA:



Reading the csv file

```
In [2]: student_performance=pd.read_csv("D:\MY STUDY MATERIALS\SOME DATAS FOR ANALYSIS\StudentsPerformance.csv")

In [3]: student_performance
Out[3]:
```

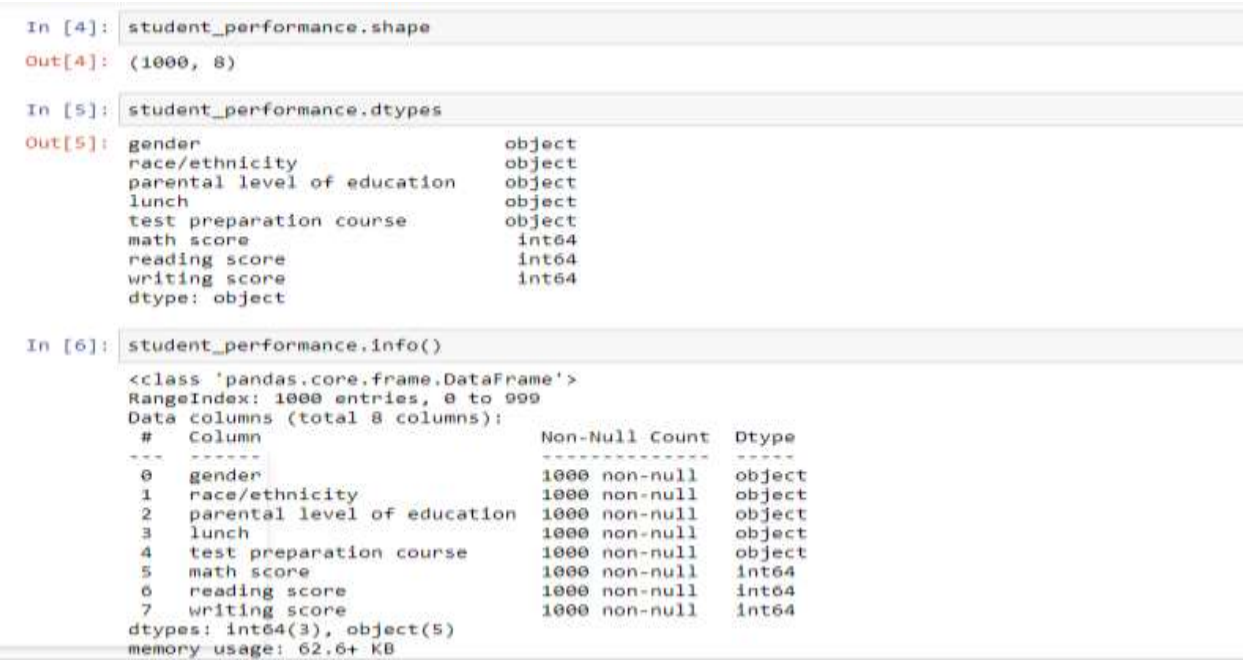
	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group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

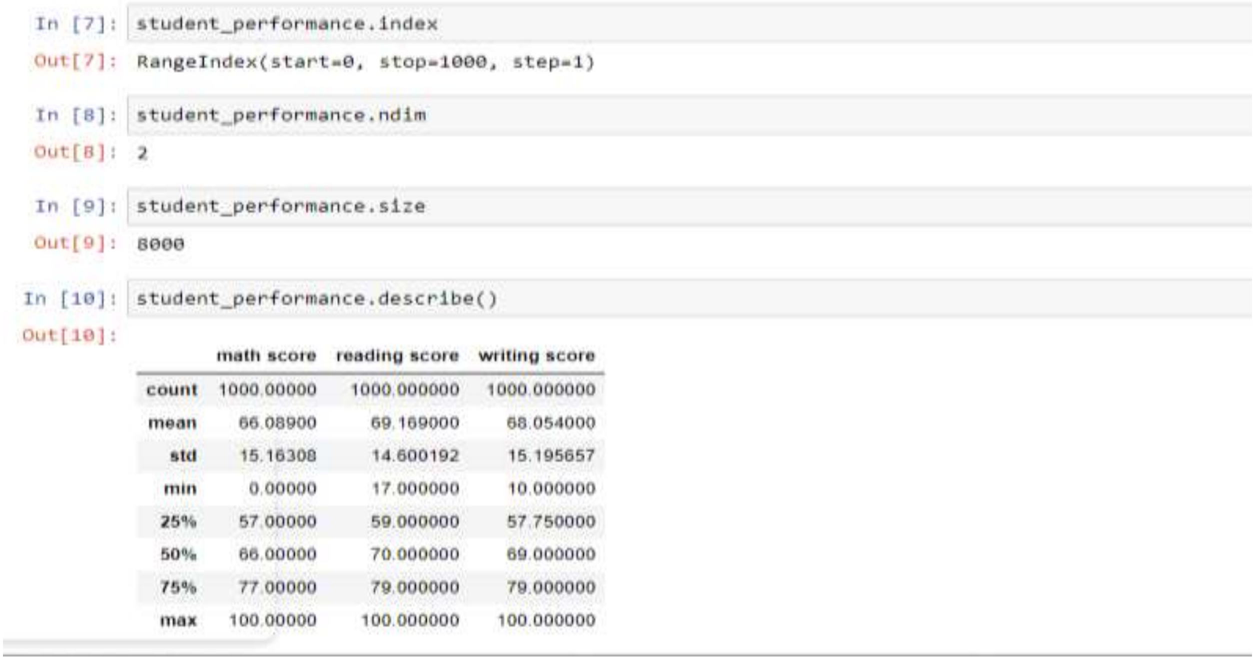
2.2.DATA EXPLORATION

Data preparing is basically understanding about the data such as

- Shape
- Attributes
- information of data
- index
- Dimensions of the data
- Size of the data
- To know a rough about the data roughly,
- Statistics and pairplot

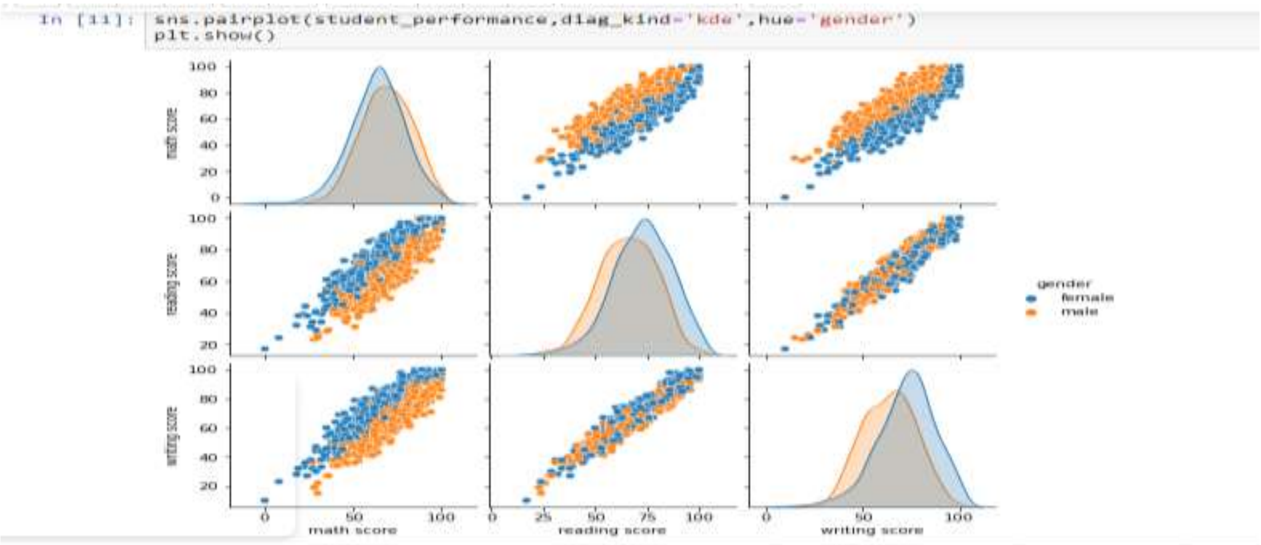


```
In [4]: student_performance.shape
Out[4]: (1000, 8)

In [5]: student_performance.dtypes
Out[5]: gender                          object
        race/ethnicity                  object
        parental level of education     object
        lunch                           object
        test preparation course         object
        math score                       int64
        reading score                    int64
        writing score                    int64
        dtype: object

In [6]: student_performance.info()
        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 1000 entries, 0 to 999
        Data columns (total 8 columns):
         #   Column                       Non-Null Count  Dtype
        ---  ------                       --------------  -----
         0   gender                       1000 non-null   object
         1   race/ethnicity               1000 non-null   object
         2   parental level of education  1000 non-null   object
         3   lunch                        1000 non-null   object
         4   test preparation course      1000 non-null   object
         5   math score                   1000 non-null   int64
         6   reading score                1000 non-null   int64
         7   writing score                1000 non-null   int64
        dtypes: int64(3), object(5)
        memory usage: 62.6+ KB
```



```
In [7]: student_performance.index
Out[7]: RangeIndex(start=0, stop=1000, step=1)

In [8]: student_performance.ndim
Out[8]: 2

In [9]: student_performance.size
Out[9]: 8000

In [10]: student_performance.describe()
Out[10]:
```

	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 [11]: sns.pairplot(student_performance,diag_kind='kde',hue='gender')
         plt.show()
```

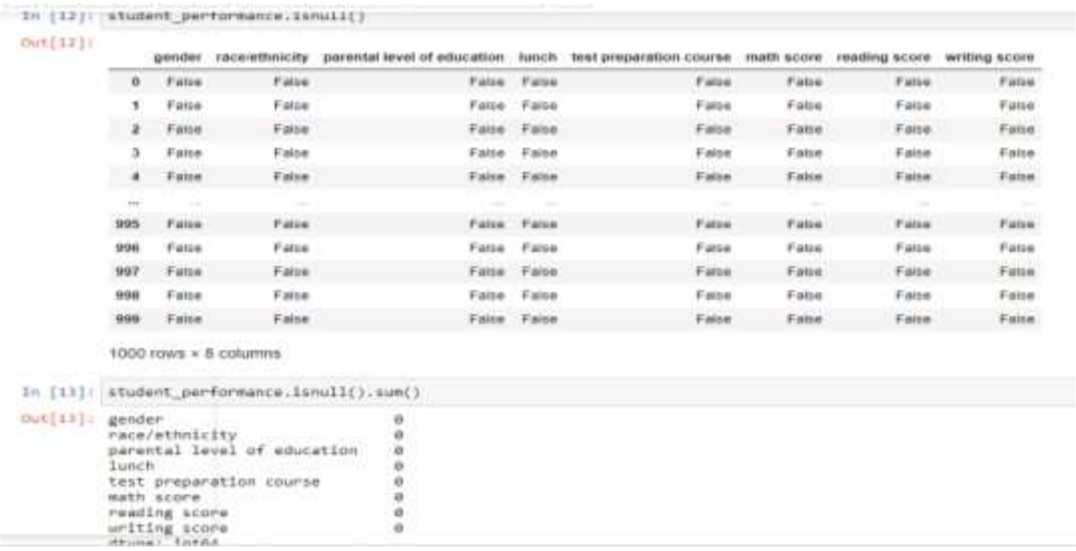
In short data contains

Shape	rows-1000, columns-8
Attributes	dtypes: object(5)
information of data	dtypes: int64(3), object(5)
Index	rangeindex(start=0,stop=1000)
Dimensions of the data	2
Size of the dataset	8000
Statistics	Mean, meadian, mode of numerical data

2.3.DATA CLEANING

After reviewing the data, next step is to clean the data and make it relevant for the purpose of analysis. Now, next step is to check for

- Null
- Column names
- duplicates
- renaming
- drop column
    - ➢ checking for null values and get the number of null values



```
In [12]: student_performance.isnull()
Out[12]:
```

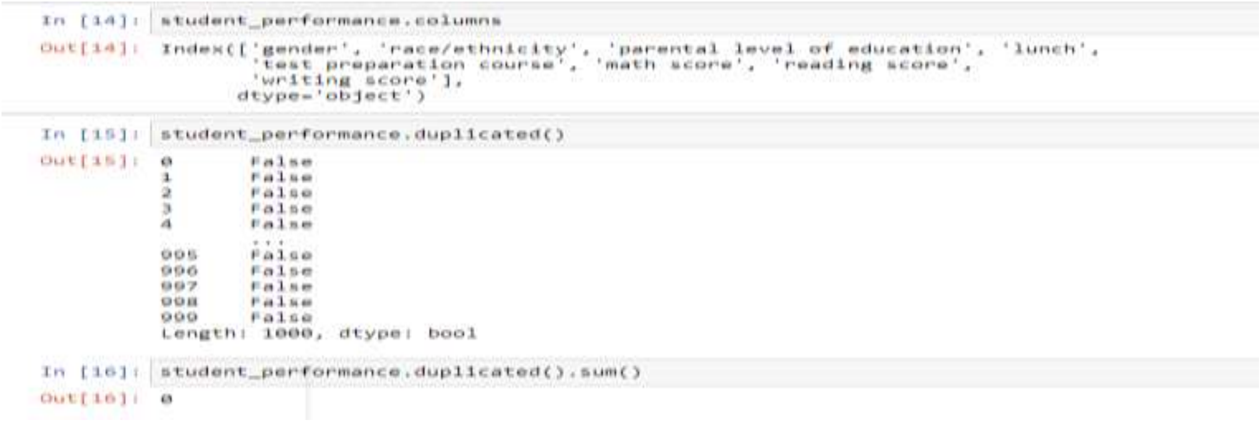
	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	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 [13]: student_performance.isnull().sum()
Out[13]: 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
```

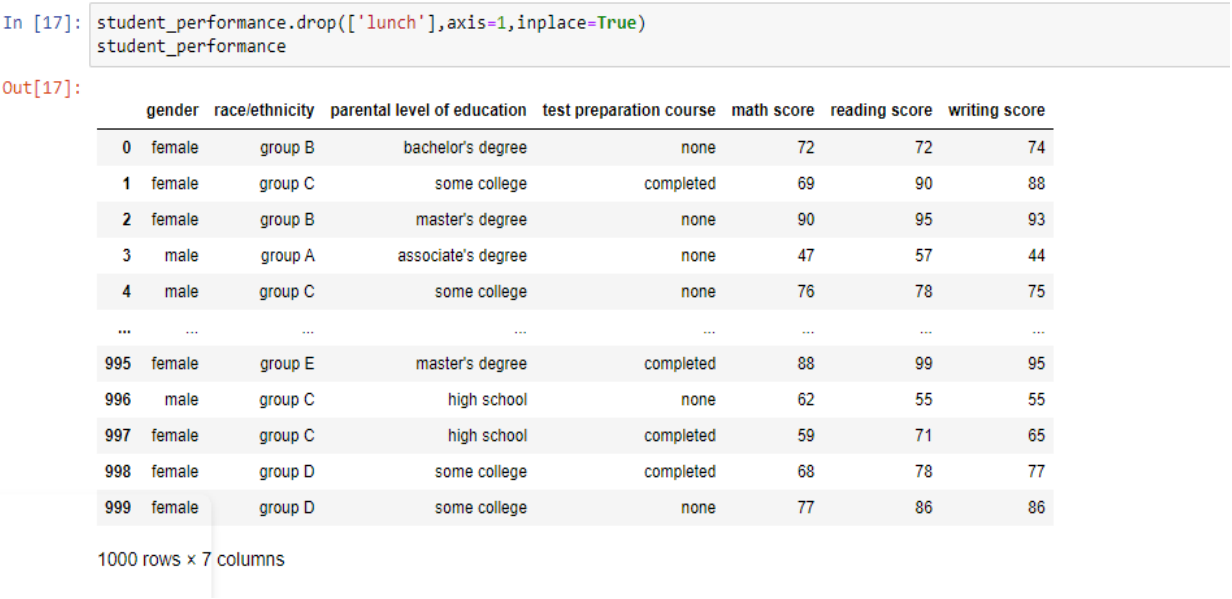
Data is free of null values

- ➢ Details of column names and check for duplicates data if any and hence find its sum.



```
In [14]: student_performance.columns

Out[14]: Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
                'test preparation course', 'math score', 'reading score',
                'writing score'],
               dtype='object')

In [15]: student_performance.duplicated()

Out[15]: 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 [16]: student_performance.duplicated().sum()

Out[16]: 0
```

- ➢ Dropping column “lunch” since it doesn’t give any information for my analysis



```
In [17]: student_performance.drop(['lunch'],axis=1,inplace=True)
         student_performance
```

Out[17]:

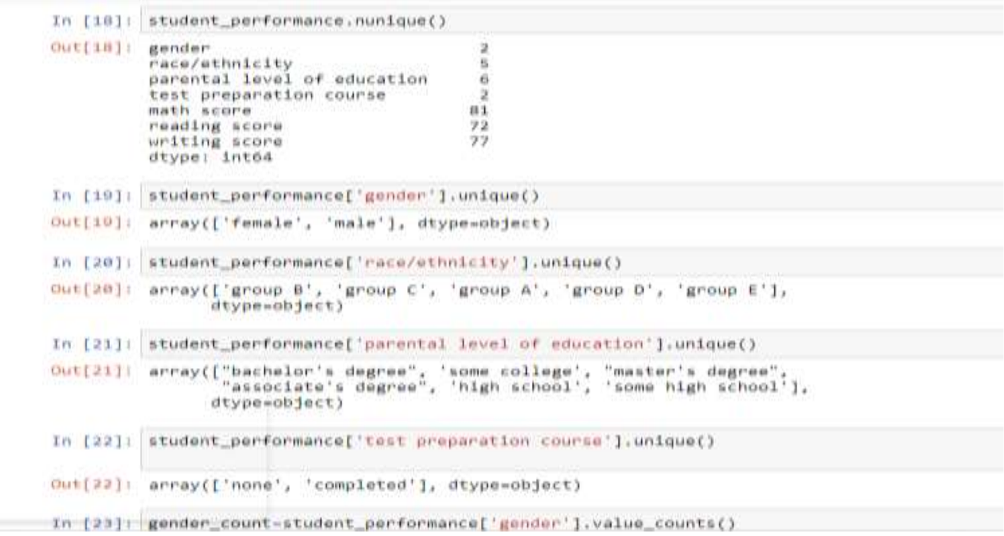
	gender	race/ethnicity	parental level of education	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	none	72	72	74
1	female	group C	some college	completed	69	90	88
2	female	group B	master's degree	none	90	95	93
3	male	group A	associate's degree	none	47	57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

1000 rows × 7 columns

2.4.DATA EXTRACTION:

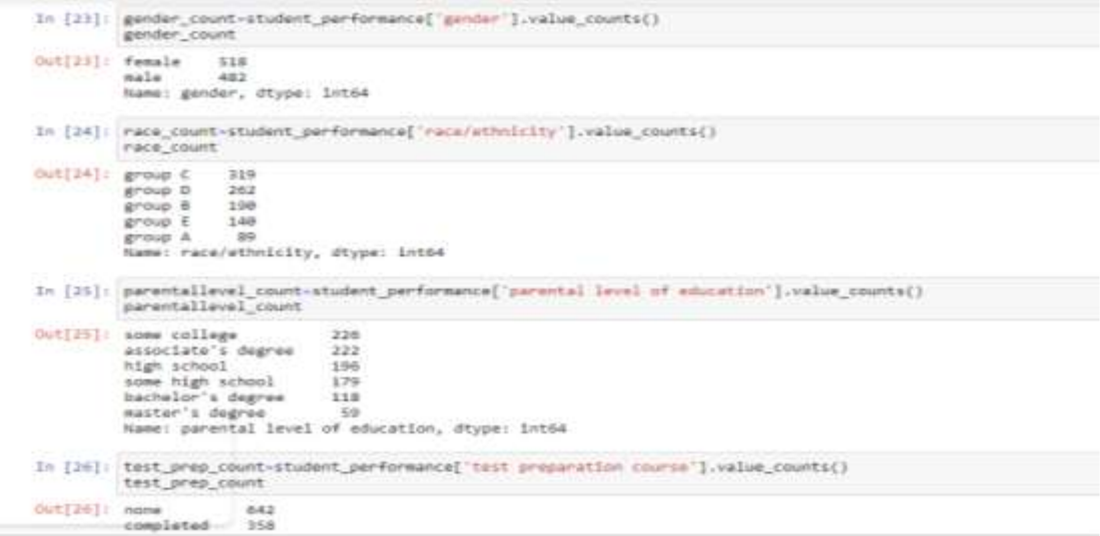
Here, we check for unique columns and their related information and move forward with those datas required for data analysis.

- Checking for unique values in the dataset



```
In [18]: student_performance.nunique()
Out[18]: gender                          2
         race/ethnicity                  5
         parental level of education     6
         test preparation course         2
         math score                     81
         reading score                  72
         writing score                  77
         dtype: int64

In [19]: student_performance['gender'].unique()
Out[19]: array(['female', 'male'], dtype=object)

In [20]: student_performance['race/ethnicity'].unique()
Out[20]: array(['group B', 'group C', 'group A', 'group D', 'group E'],
               dtype=object)

In [21]: student_performance['parental level of education'].unique()
Out[21]: array(["bachelor's degree", 'some college', "master's degree",
                "associate's degree", 'high school', 'some high school'],
               dtype=object)

In [22]: student_performance['test preparation course'].unique()

Out[22]: array(['none', 'completed'], dtype=object)

In [23]: gender_count=student_performance['gender'].value_counts()
```

- Now, for the count of each



```
In [23]: gender_count=student_performance['gender'].value_counts()
         gender_count
Out[23]: female    518
         male      482
         Name: gender, dtype: int64

In [24]: race_count=student_performance['race/ethnicity'].value_counts()
         race_count
Out[24]: group C    319
         group D    262
         group B    190
         group E    140
         group A     89
         Name: race/ethnicity, dtype: int64

In [25]: parentallevel_count=student_performance['parental level of education'].value_counts()
         parentallevel_count
Out[25]: some college          226
         associate's degree    222
         high school           196
         some high school      179
         bachelor's degree     118
         master's degree        59
         Name: parental level of education, dtype: int64

In [26]: test_prep_count=student_performance['test preparation course'].value_counts()
         test_prep_count
Out[26]: none         642
         completed    358
```

This gives a clear idea about student’s performance data set more closer which makes it ready for EDA.

2.5.DATA INSIGHT AND VISUALIZATION:

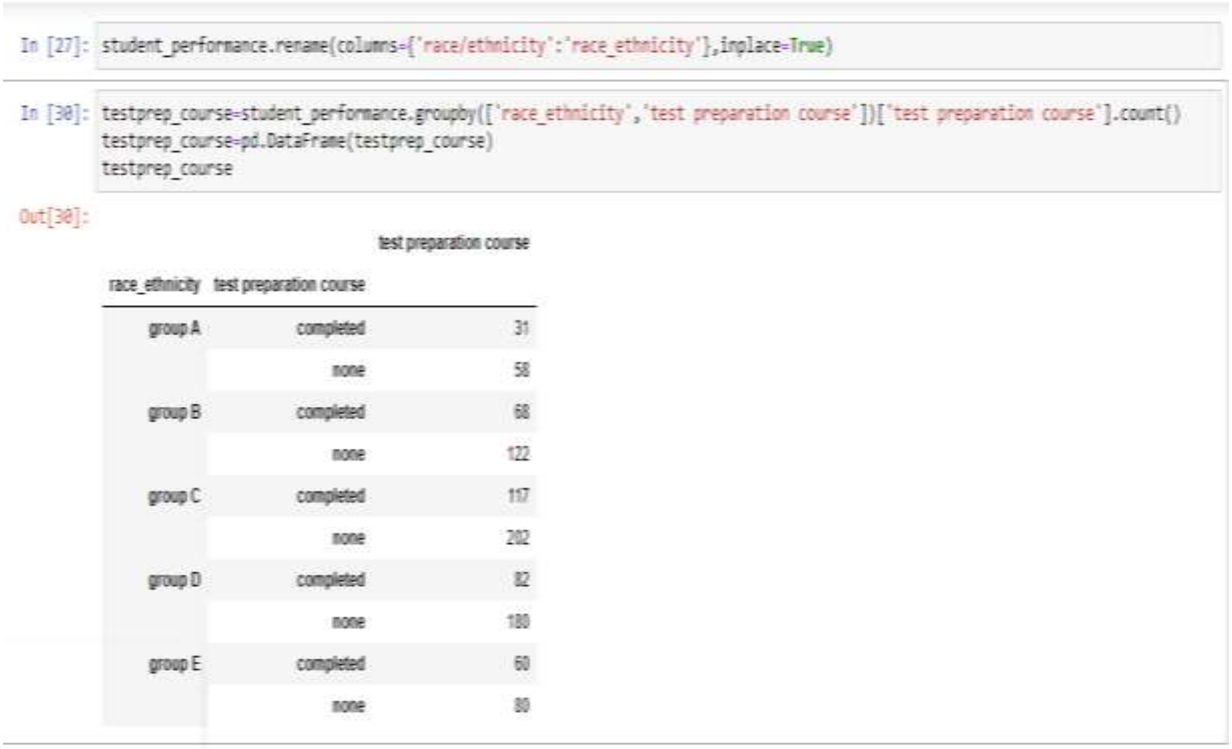
Now, a relevant data is prepared for the analysis and next is to resolve each purpose of the analysis one by one.

2.5.1.Data v/s test preparation course

Set of data variables are tested against test preparation course to conclude how effectiveness of test preparation course

From the above data frame "test_prep_count" , it's clear that number of students who completed the test preparation course is comparatively one third of the students.

- ➢ Compare the "test preparation course" with "race/ethnicity"

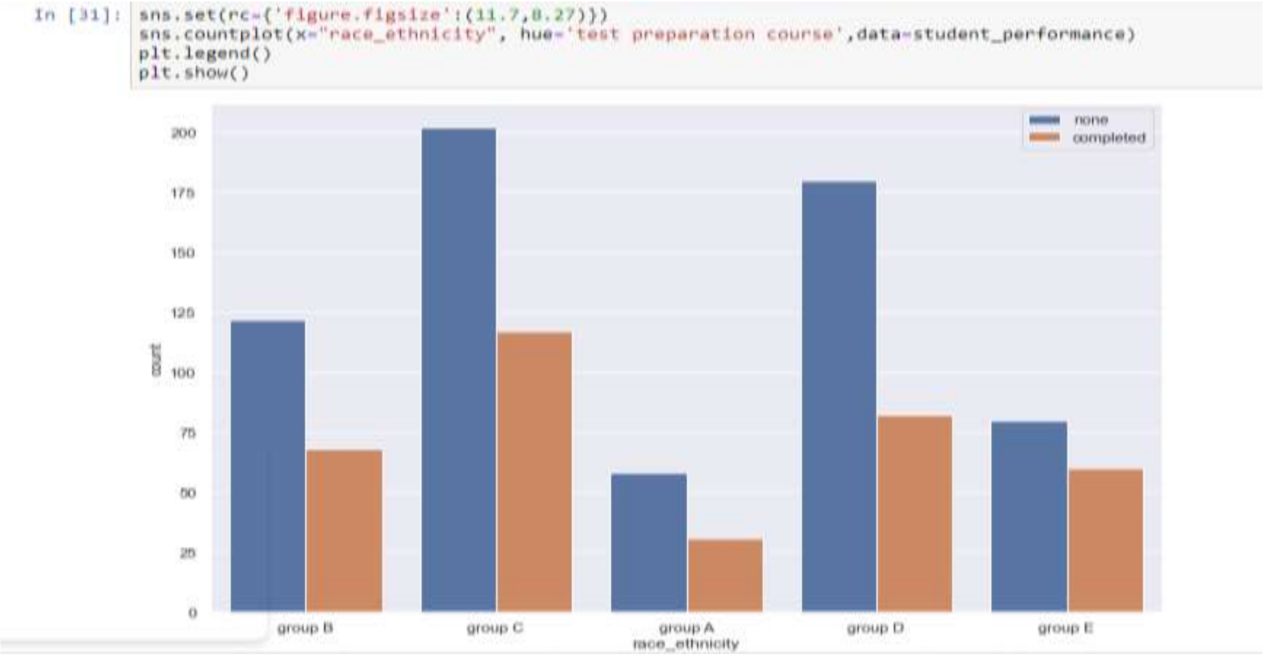


```
In [27]: student_performance.rename(columns={'race/ethnicity':'race_ethnicity'},inplace=True)
```

```
In [30]: testprep_course=student_performance.groupby(['race_ethnicity','test preparation course'])['test preparation course'].count()
         testprep_course=pd.DataFrame(testprep_course)
         testprep_course
```

Out[30]:

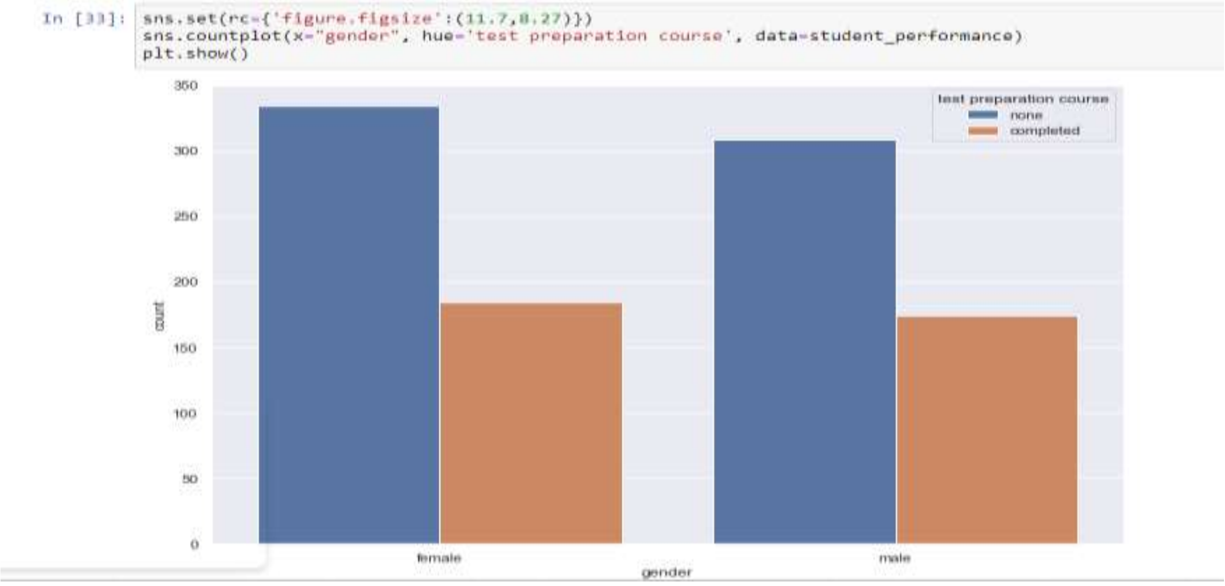
		test preparation course
race_ethnicity	test preparation course	
group A	completed	31
	none	58
group B	completed	68
	none	122
group C	completed	117
	none	202
group D	completed	82
	none	180
group E	completed	60
	none	80



```
In [31]: sns.set(rc={'figure.figsize':(11.7,8.27)})
         sns.countplot(x="race_ethnicity", hue='test preparation course',data=student_performance)
         plt.legend()
         plt.show()
```

Looking into the dataframe and visualization its clearly understood that students from group C has completed the maximum along with half of it members are not completed ones.

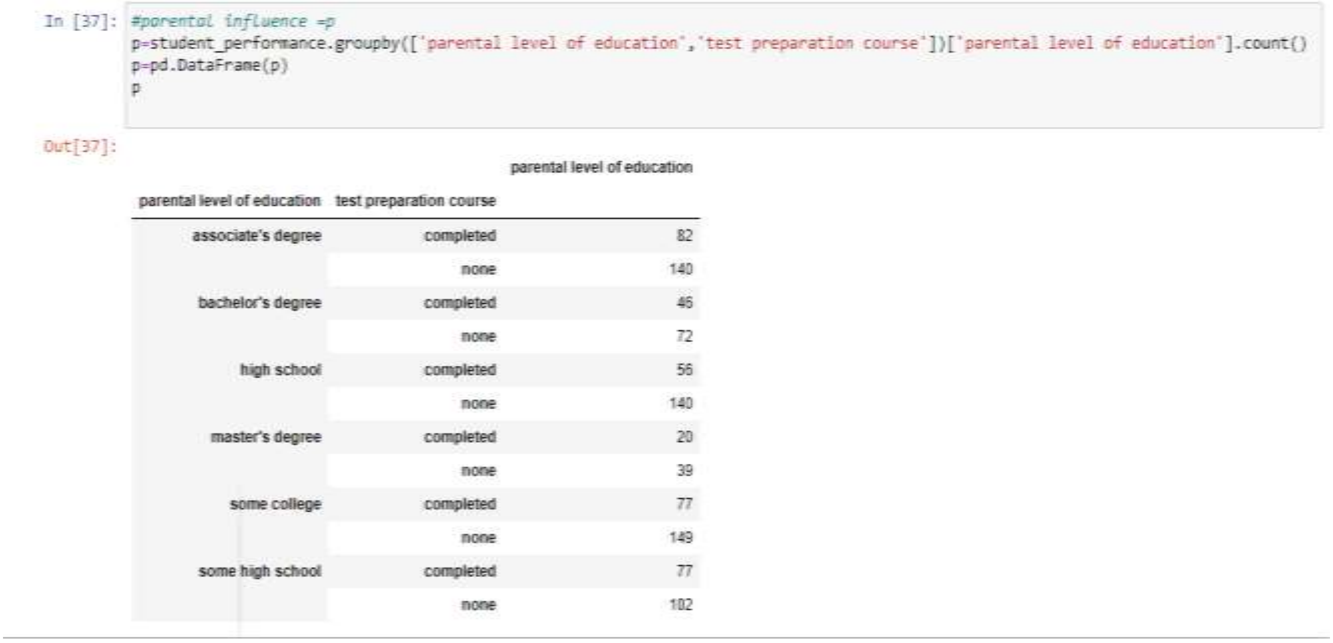
- Comparing " test preparation course" gender wise



```
In [33]: sns.set(rc={'figure.figsize':(11.7,8.27)})
         sns.countplot(x="gender", hue='test preparation course', data=student_performance)
         plt.show()
```

Students who did not complete are almost the same between males and females .

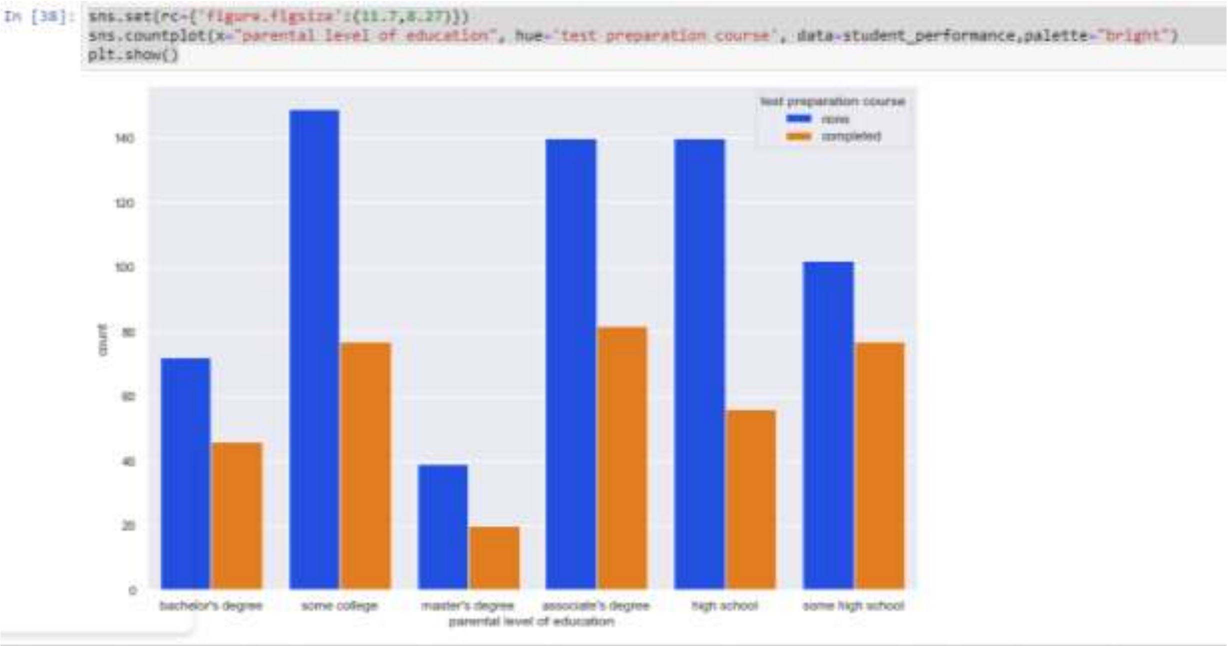
2.5.2. Test preparation course v/s parental education:



```
In [37]: #parental influence =p
         p=student_performance.groupby(['parental level of education','test preparation course'])['parental level of education'].count()
         p=pd.DataFrame(p)
         p
```

Out[37]:

		parental level of education
parental level of education	test preparation course	
associate's degree	completed	82
	none	140
bachelor's degree	completed	46
	none	72
high school	completed	56
	none	140
master's degree	completed	20
	none	39
some college	completed	77
	none	149
some high school	completed	77
	none	102

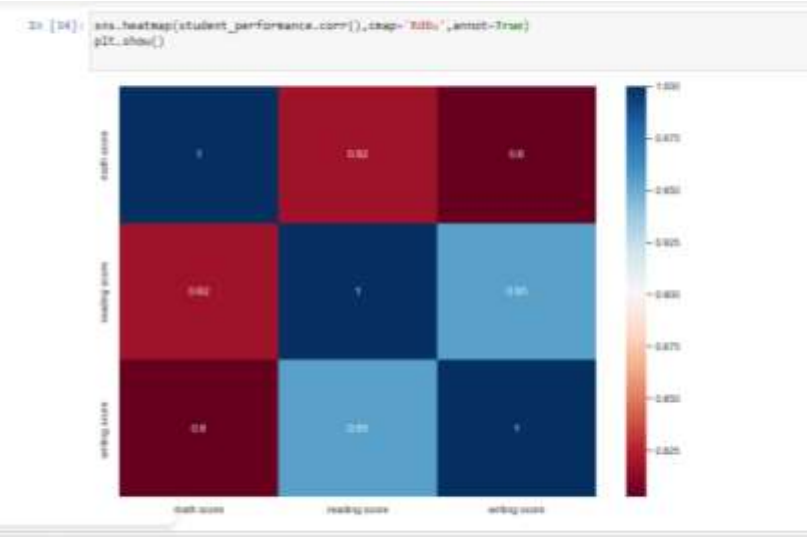


```
In [38]: sns.set(rc={'figure.figsize':(11.7,8.27)})
         sns.countplot(x="parental level of education", hue='test preparation course', data=student_performance,palette="bright")
         plt.show()
```

In short, test preparation course had no difference in gender involvement but it does had a relation in the difference in the race/ethnicity. So that is main reason in the difference in non-completion. Moreover, parental education did have a positive impact on completion.

2.5.3.Test scores v/s various variables in the data set :

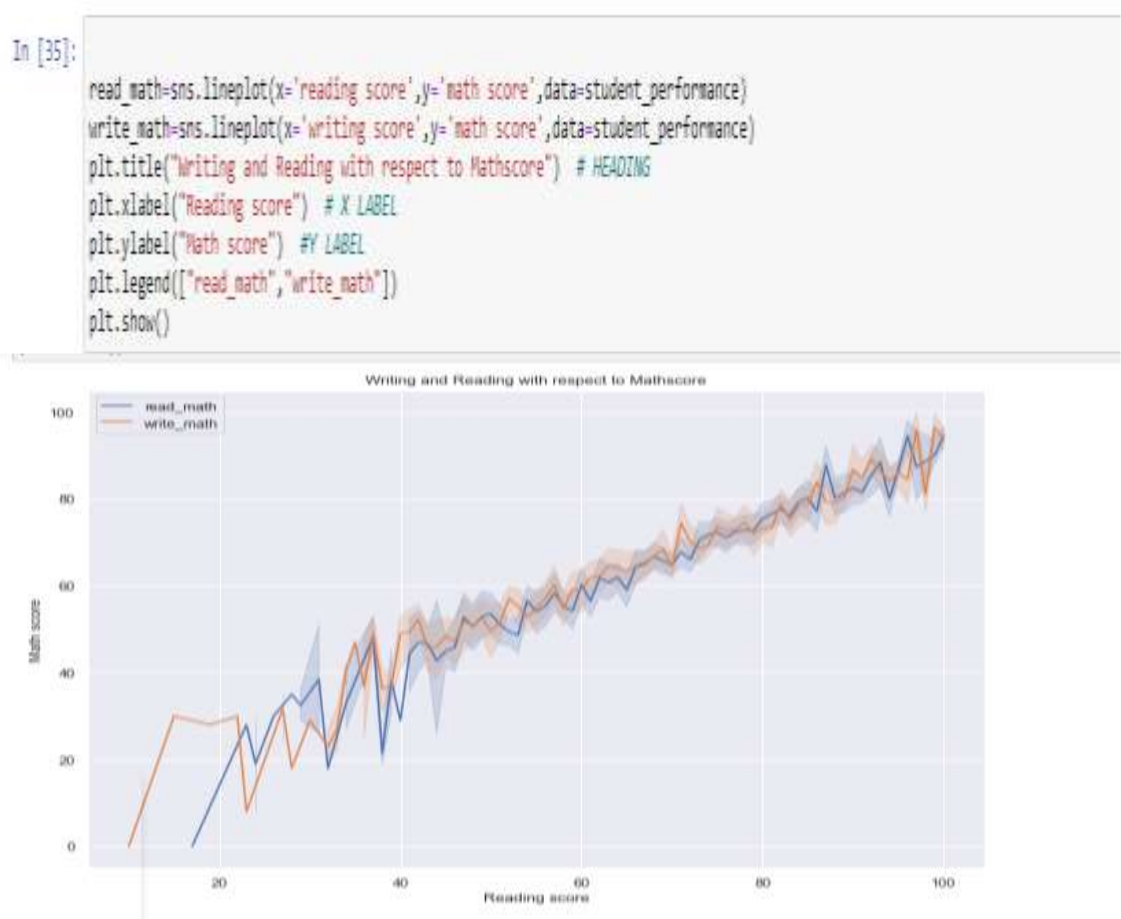
- ➢ For a start, its better to check the correlation between the entire test scores which can give rough idea about which data correlated.



```
In [34]: sns.heatmap(student_performance.corr(),cmap='RdBu',annot=True)
         plt.show()
```

Clearly,” reading and writing score” has a very good correlation between them (0.9).

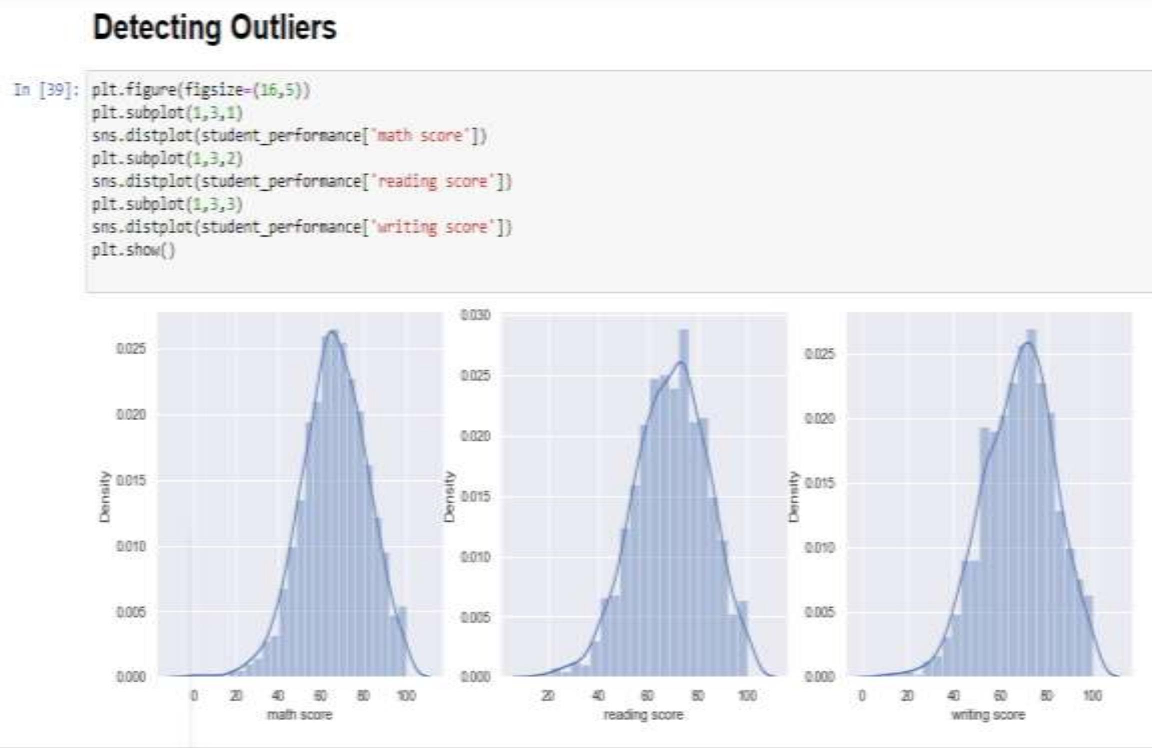
- Since its clear about reading and writing , next is to check whether these two have relation with maths score together with the help of a line plot together in one graph.



```
In [35]:
read_math=sns.lineplot(x='reading score',y='math score',data=student_performance)
write_math=sns.lineplot(x='writing score',y='math score',data=student_performance)
plt.title("Writing and Reading with respect to Mathscore")  # HEADING
plt.xlabel("Reading score")  # X LABEL
plt.ylabel("Math score")  #Y LABEL
plt.legend(["read_math","write_math"])
plt.show()
```

As certain writing and reading did have a good correlation with the math score. So it’s good habit to read and write to increase the score in the test preparation course.

- For a last confirmation it is better to check with outliers



Detecting Outliers

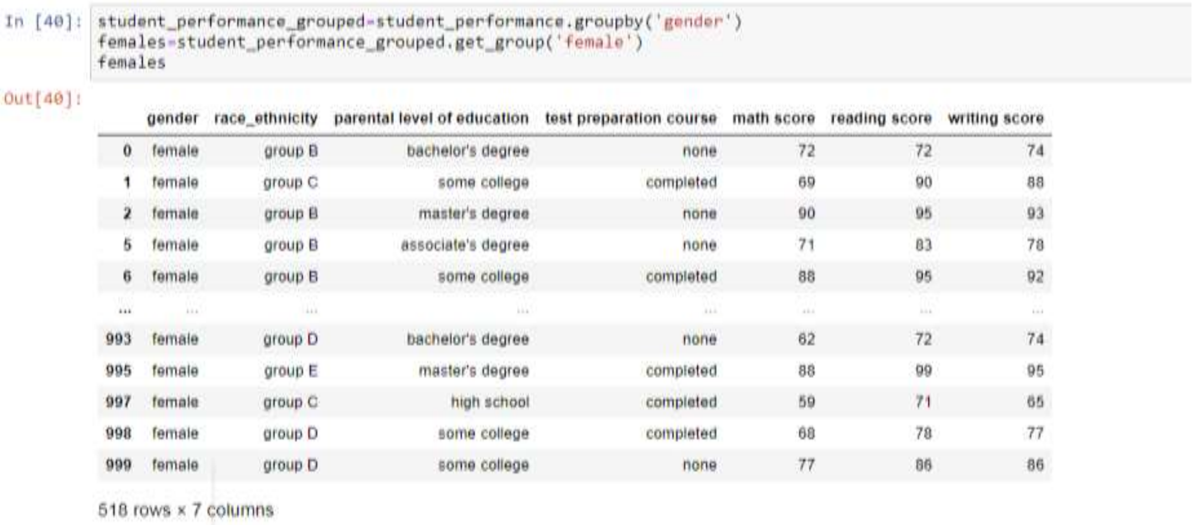
```
In [39]: plt.figure(figsize=(16,5))
         plt.subplot(1,3,1)
         sns.distplot(student_performance['math score'])
         plt.subplot(1,3,2)
         sns.distplot(student_performance['reading score'])
         plt.subplot(1,3,3)
         sns.distplot(student_performance['writing score'])
         plt.show()
```

There is no much skewness between the columns considered. Hence conclude that reading, writing and math score are related symmetrically.

2.5.4. Score v/s Gender

- Here, statistics of score is taken to understand whether the test score differ with gender and also to go into details of the marks and their percentage.
- Giving a rule for pass marks we calculate the students who passed and failed along with their gender.

  Grouping data with “female” and statistics scores

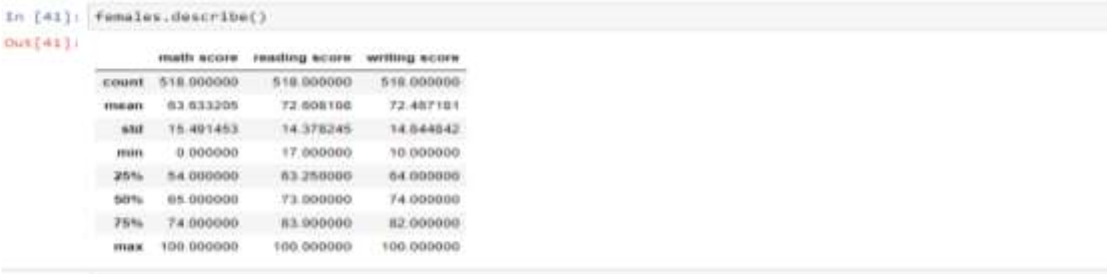


```
In [40]: student_performance_grouped=student_performance.groupby('gender')
         females=student_performance_grouped.get_group('female')
         females
```

Out[40]:

	gender	race_ethnicity	parental level of education	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	none	72	72	74
1	female	group C	some college	completed	69	90	88
2	female	group B	master's degree	none	90	95	93
5	female	group B	associate's degree	none	71	83	78
6	female	group B	some college	completed	88	95	92
...	...	...	...	...	...	...	...
993	female	group D	bachelor's degree	none	62	72	74
995	female	group E	master's degree	completed	88	99	95
997	female	group C	high school	completed	59	71	65
998	female	group D	some college	completed	68	78	77
999	female	group D	some college	none	77	86	86

518 rows × 7 columns

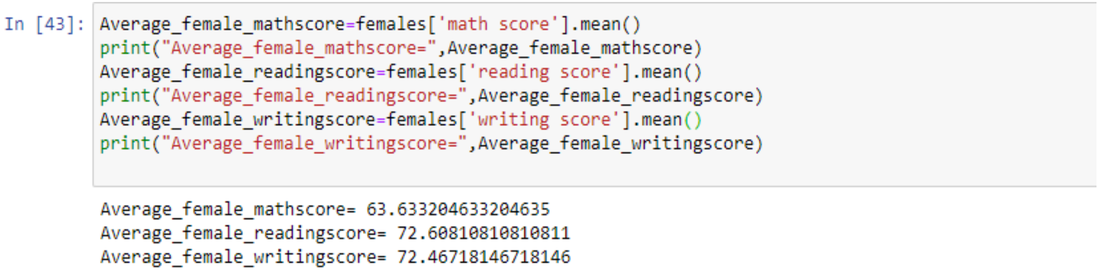


In [41]:

```
females.describe()
```

Out[41]:

	math score	reading score	writing score
count	518.000000	518.000000	518.000000
mean	63.633205	72.608108	72.467181
std	15.491453	14.378245	14.844842
min	0.000000	17.000000	10.000000
25%	54.000000	63.250000	64.000000
50%	65.000000	73.000000	74.000000
75%	74.000000	83.000000	82.000000
max	100.000000	100.000000	100.000000

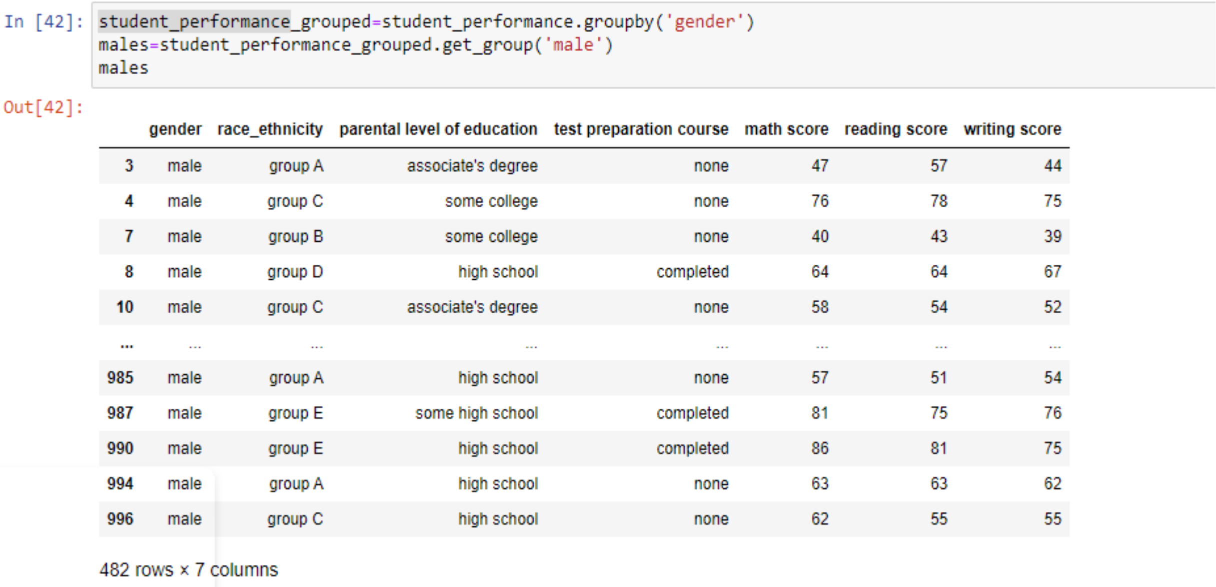


In [43]:

```
Average_female_mathscore=females['math score'].mean()
print("Average_female_mathscore=",Average_female_mathscore)
Average_female_readingscore=females['reading score'].mean()
print("Average_female_readingscore=",Average_female_readingscore)
Average_female_writingscore=females['writing score'].mean()
print("Average_female_writingscore=",Average_female_writingscore)
```

```
Average_female_mathscore= 63.633204633204635
Average_female_readingscore= 72.60810810810811
Average_female_writingscore= 72.46718146718146
```

- Grouping data with “male” and statistics scores



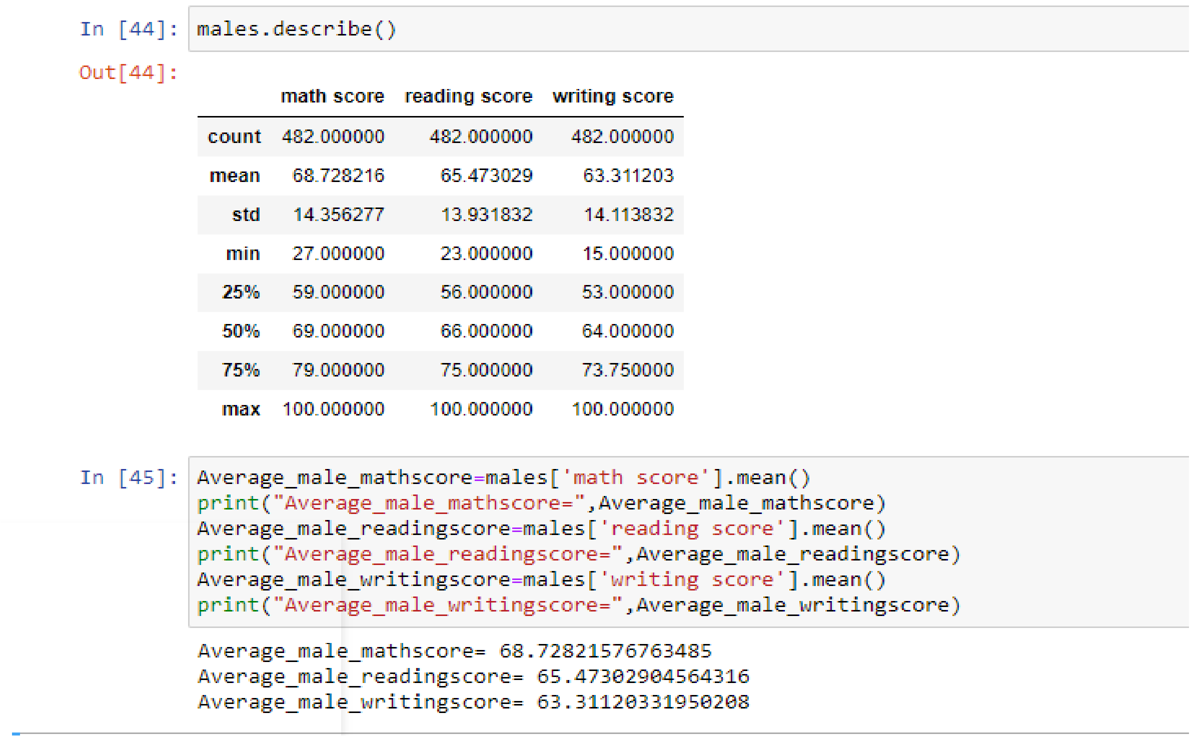
In [42]:

```
student_performance_grouped=student_performance.groupby('gender')
males=student_performance_grouped.get_group('male')
males
```

Out[42]:

	gender	race_ethnicity	parental level of education	test preparation course	math score	reading score	writing score
3	male	group A	associate's degree	none	47	57	44
4	male	group C	some college	none	76	78	75
7	male	group B	some college	none	40	43	39
8	male	group D	high school	completed	64	64	67
10	male	group C	associate's degree	none	58	54	52
...	...	...	...	...	...	...	...
985	male	group A	high school	none	57	51	54
987	male	group E	some high school	completed	81	75	76
990	male	group E	high school	completed	86	81	75
994	male	group A	high school	none	63	63	62
996	male	group C	high school	none	62	55	55

482 rows × 7 columns



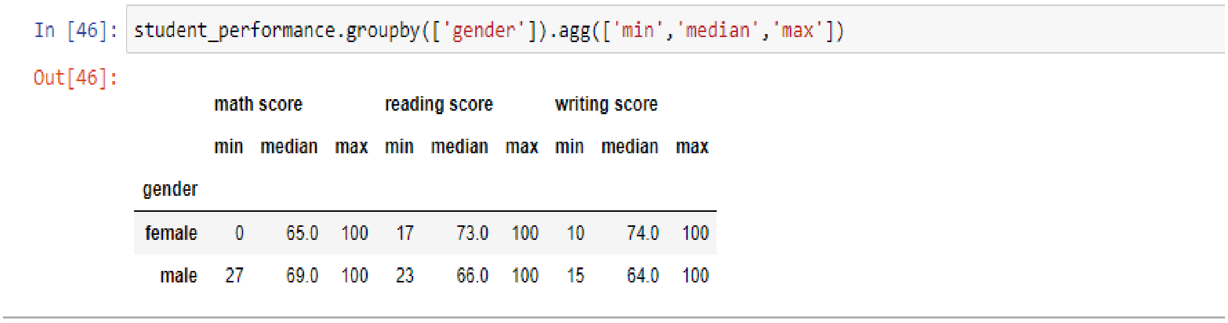
```
In [44]: males.describe()
```

Out[44]:

	math score	reading score	writing score
count	482.000000	482.000000	482.000000
mean	68.728216	65.473029	63.311203
std	14.356277	13.931832	14.113832
min	27.000000	23.000000	15.000000
25%	59.000000	56.000000	53.000000
50%	69.000000	66.000000	64.000000
75%	79.000000	75.000000	73.750000
max	100.000000	100.000000	100.000000

```
In [45]: Average_male_mathscore=males['math score'].mean()
         print("Average_male_mathscore=",Average_male_mathscore)
         Average_male_readingscore=males['reading score'].mean()
         print("Average_male_readingscore=",Average_male_readingscore)
         Average_male_writingscore=males['writing score'].mean()
         print("Average_male_writingscore=",Average_male_writingscore)

         Average_male_mathscore= 68.72821576763485
         Average_male_readingscore= 65.47302904564316
         Average_male_writingscore= 63.31120331950208
```

Looking at the details we clearly get an idea about the average score but it is always good to take only what is required and make an easy understandable form.



```
In [46]: student_performance.groupby(['gender']).agg(['min','median','max'])
```

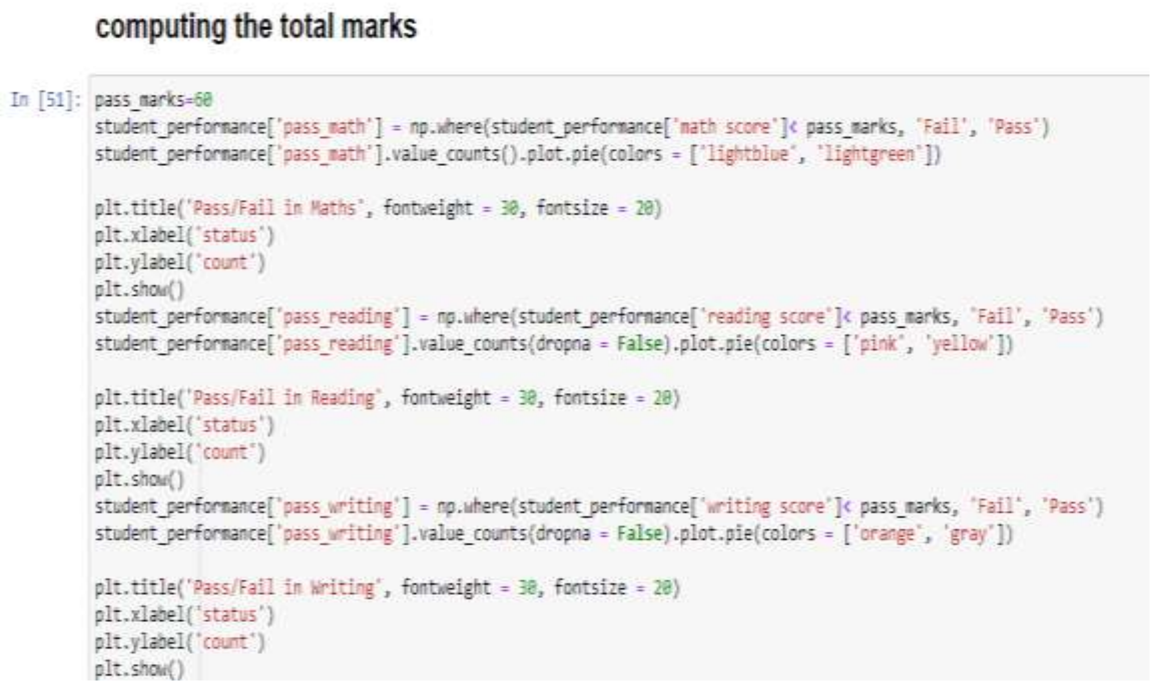
Out[46]:

	math score			reading score			writing score		
	min	median	max	min	median	max	min	median	max
gender									
female	0	65.0	100	17	73.0	100	10	74.0	100
male	27	69.0	100	23	66.0	100	15	64.0	100

Obviously, math score indicates well with males whereas reading and writing skill are good with females comparatively.

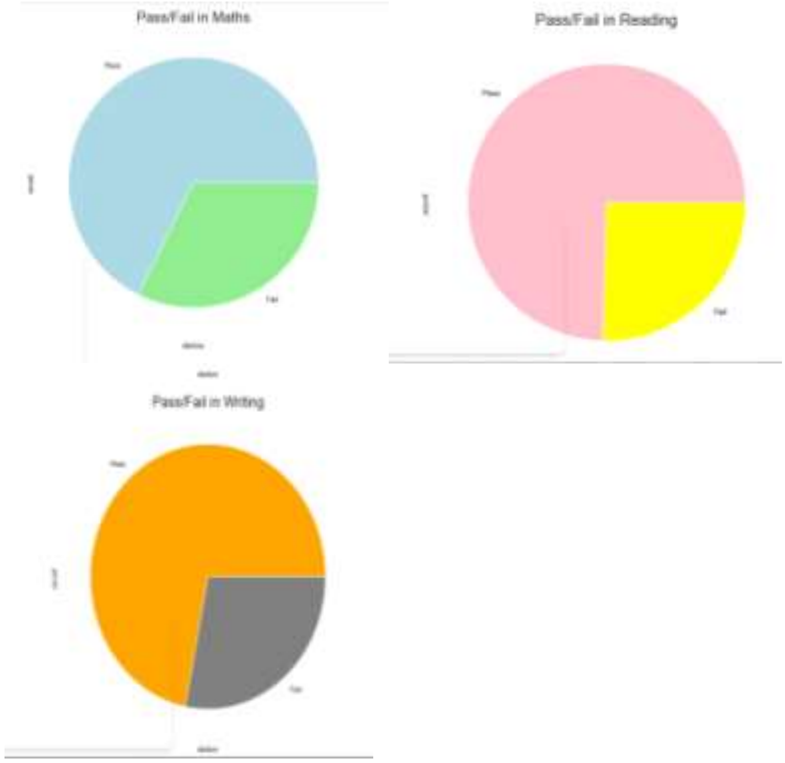
2.5.5. **Score to grades**:

Setting a pass mark for all three scores to be 60 and adding a new columns for the data set giving idea about those students who pass and fail in each test.

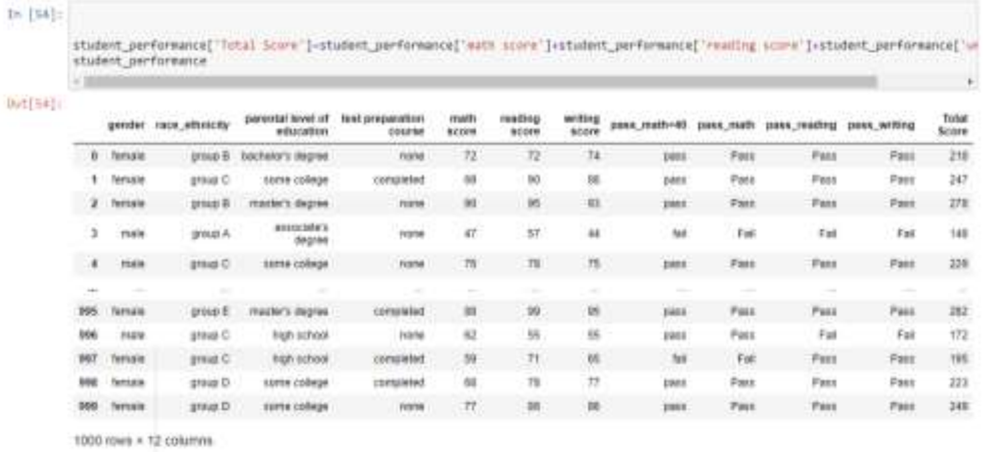


computing the total marks

```
In [51]: pass_marks=60
         student_performance['pass_math'] = np.where(student_performance['math score']< pass_marks, 'Fail', 'Pass')
         student_performance['pass_math'].value_counts().plot.pie(colors = ['lightblue', 'lightgreen'])

         plt.title('Pass/Fail in Maths', fontweight = 30, fontsize = 20)
         plt.xlabel('status')
         plt.ylabel('count')
         plt.show()
         student_performance['pass_reading'] = np.where(student_performance['reading score']< pass_marks, 'Fail', 'Pass')
         student_performance['pass_reading'].value_counts(dropna = False).plot.pie(colors = ['pink', 'yellow'])

         plt.title('Pass/Fail in Reading', fontweight = 30, fontsize = 20)
         plt.xlabel('status')
         plt.ylabel('count')
         plt.show()
         student_performance['pass_writing'] = np.where(student_performance['writing score']< pass_marks, 'Fail', 'Pass')
         student_performance['pass_writing'].value_counts(dropna = False).plot.pie(colors = ['orange', 'gray'])

         plt.title('Pass/Fail in Writing', fontweight = 30, fontsize = 20)
         plt.xlabel('status')
         plt.ylabel('count')
         plt.show()
```



Adding total marks and percentage to the column to make the data better for a conclusion



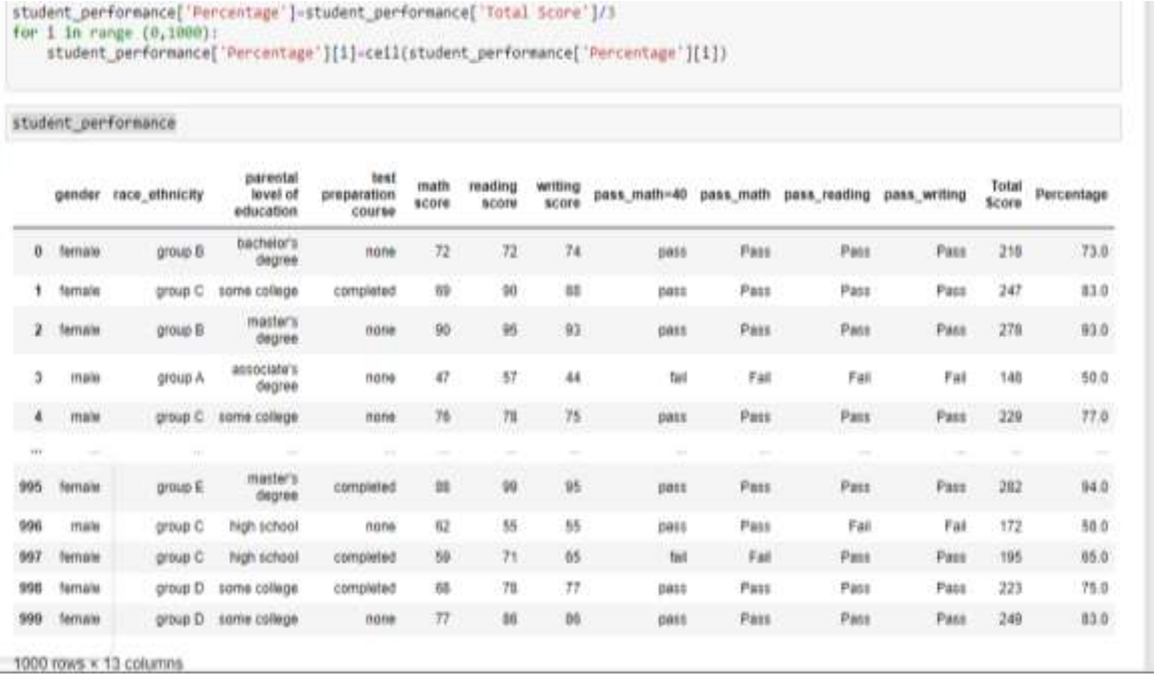
In [54]:

```
student_performance['Total Score']=student_performance['math score']+student_performance['reading score']+student_performance['w
student_performance
```

Out[54]:

	gender	race_ethnicity	parental level of education	test preparation course	math score	reading score	writing score	pass_math>40	pass_math	pass_reading	pass_writing	Total Score
0	female	group B	bachelor's degree	none	72	72	74	pass	Pass	Pass	Pass	218
1	female	group C	some college	completed	69	90	88	pass	Pass	Pass	Pass	247
2	female	group B	master's degree	none	90	95	93	pass	Pass	Pass	Pass	278
3	male	group A	associate's degree	none	47	57	44	fail	Fail	Fail	Fail	148
4	male	group C	some college	none	76	78	75	pass	Pass	Pass	Pass	229
...	...	...	...	...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	completed	88	99	95	pass	Pass	Pass	Pass	282
996	male	group C	high school	none	62	55	55	pass	Pass	Fail	Fail	172
997	female	group C	high school	completed	59	71	65	fail	Fail	Pass	Pass	195
998	female	group D	some college	completed	68	78	77	pass	Pass	Pass	Pass	223
999	female	group D	some college	none	77	86	86	pass	Pass	Pass	Pass	249

1000 rows × 12 columns



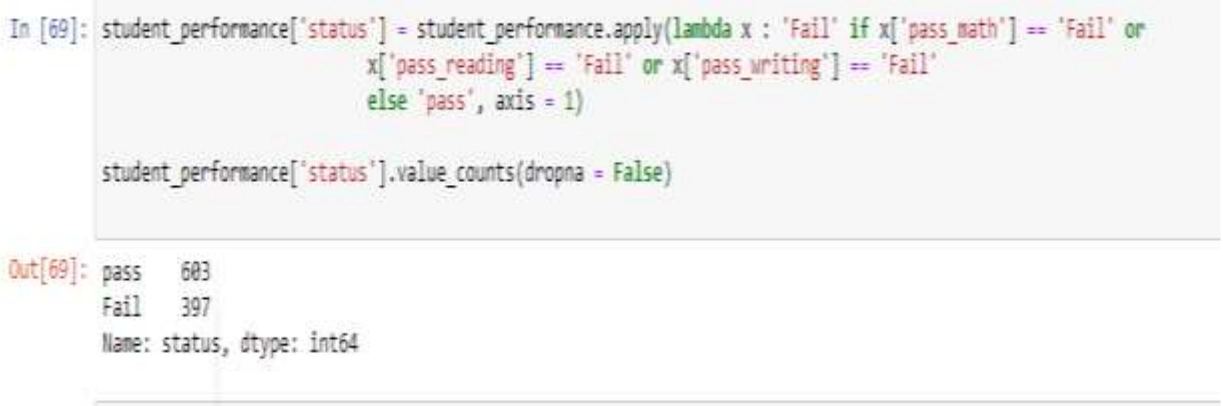
```
student_performance['Percentage']=student_performance['Total Score']/3
for i in range (0,1000):
    student_performance['Percentage'][i]=ceil(student_performance['Percentage'][i])
```

In [66]: `student_performance`

Out[66]:

	gender	race_ethnicity	parental level of education	test preparation course	math score	reading score	writing score	pass_math>40	pass_math	pass_reading	pass_writing	Total Score	Percentage
0	female	group B	bachelor's degree	none	72	72	74	pass	Pass	Pass	Pass	218	73.0
1	female	group C	some college	completed	69	90	88	pass	Pass	Pass	Pass	247	83.0
2	female	group B	master's degree	none	90	95	93	pass	Pass	Pass	Pass	278	93.0
3	male	group A	associate's degree	none	47	57	44	fail	Fail	Fail	Fail	148	50.0
4	male	group C	some college	none	76	78	75	pass	Pass	Pass	Pass	229	77.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	completed	88	99	95	pass	Pass	Pass	Pass	282	94.0
996	male	group C	high school	none	62	55	55	pass	Pass	Fail	Fail	172	58.0
997	female	group C	high school	completed	59	71	65	fail	Fail	Pass	Pass	195	65.0
998	female	group D	some college	completed	68	78	77	pass	Pass	Pass	Pass	223	75.0
999	female	group D	some college	none	77	86	86	pass	Pass	Pass	Pass	249	83.0

1000 rows × 13 columns

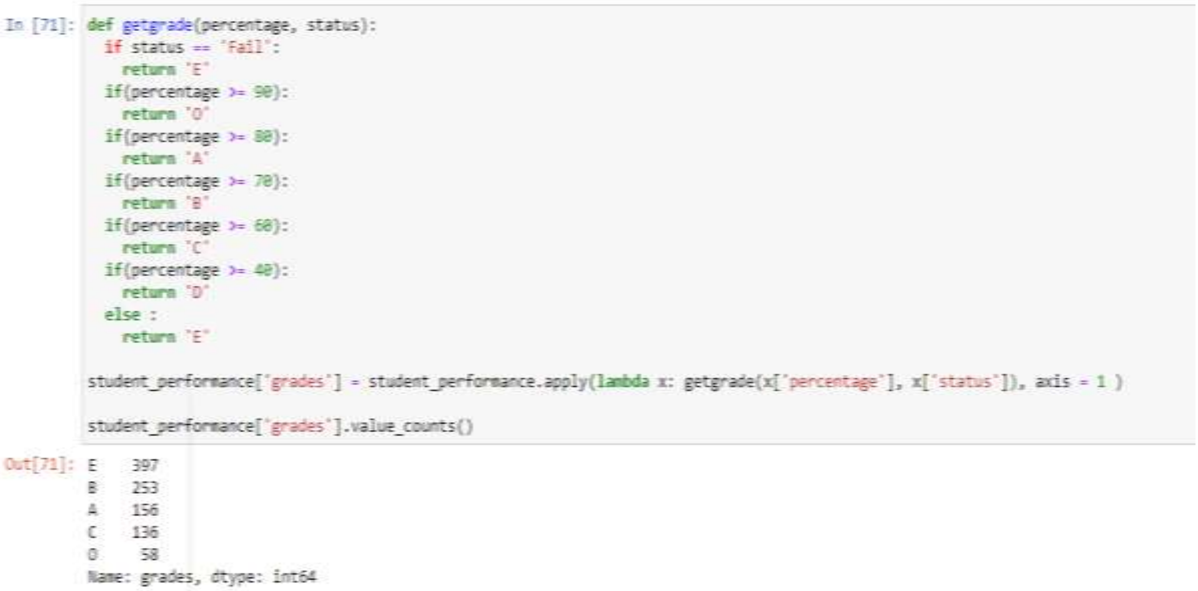


In [69]:

```
student_performance['status'] = student_performance.apply(lambda x : 'Fail' if x['pass_math'] == 'Fail' or
                                                  x['pass_reading'] == 'Fail' or x['pass_writing'] == 'Fail'
                                                  else 'pass', axis = 1)

student_performance['status'].value_counts(dropna = False)
```

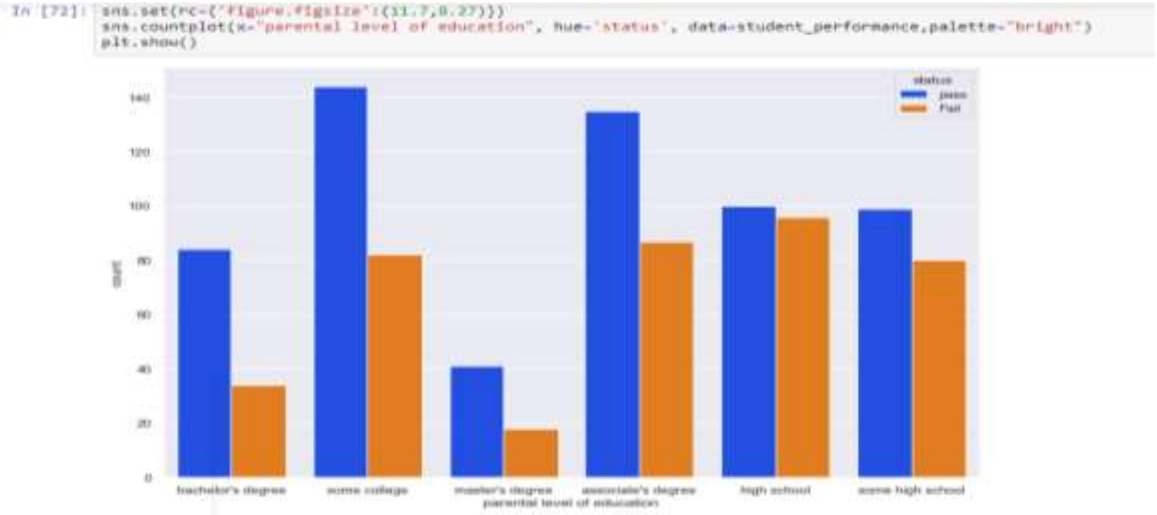
Out[69]:

```
pass    603
Fail    397
Name: status, dtype: int64
```



```
In [71]: def getgrade(percentage, status):
           if status == 'Fail':
             return 'E'
           if(percentage >= 90):
             return 'O'
           if(percentage >= 80):
             return 'A'
           if(percentage >= 70):
             return 'B'
           if(percentage >= 60):
             return 'C'
           if(percentage >= 40):
             return 'D'
           else :
             return 'E'

         student_performance['grades'] = student_performance.apply(lambda x: getgrade(x['percentage'], x['status']), axis = 1 )

         student_performance['grades'].value_counts()

Out[71]: E    397
         B    253
         A    156
         C    136
         O     58
         Name: grades, dtype: int64
```

- Clearly, the dataframe points out the students who scored the respective grades which gives an idea about the IQ level of children.



```
In [72]: sns.set(rc={'figure.figsize':(11.7,8.27)})
         sns.countplot(x="parental level of education", hue='status', data=student_performance,palette="bright")
         plt.show()
```

Surprisingly, there was no big effect of parents education with their children test score even though it can be seen that children whose parents completed their masters has no fail rate.

CONCLUSION:

- Test preparation course gives the result depending upon the race/ethnicity they are in and even the number of students in each group differs.
- Reading writing and math scores are correlated so to improve the scores students should make a habit of reading and writing to increase their skill
- In general, scores taken by males and females do not differ by completion rate and pass rate.
- Lastly, Parents education has no effect on their children which was actually a surprising fact