In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/students-performance-in-exams/StudentsPerformance.csv
```

1. Import the Libraries

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [3]:

```
data = pd.read_csv("/kaggle/input/students-performance-in-exams/StudentsPerformance.csv")
```

In [4]:

```
data.head()
```

Out[4]:

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

In [5]:

```
data.shape
```

Out[5]:

```
(1000, 8)
```

In [6]:

```
data.dtypes
```

Out[6]:

```
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 [7]:

```
duplicate_rows = data[data.duplicated()]
print('No. of duplicate rows: ', duplicate_rows.shape)
```

```
No. of duplicate rows:  (0, 8)
```

Hence, there are no duplicate rows.

In [8]:

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

Out[8]:

```
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
```

There are no Null values.

2. Exploratory Data Analysis

* Univariate Analysis

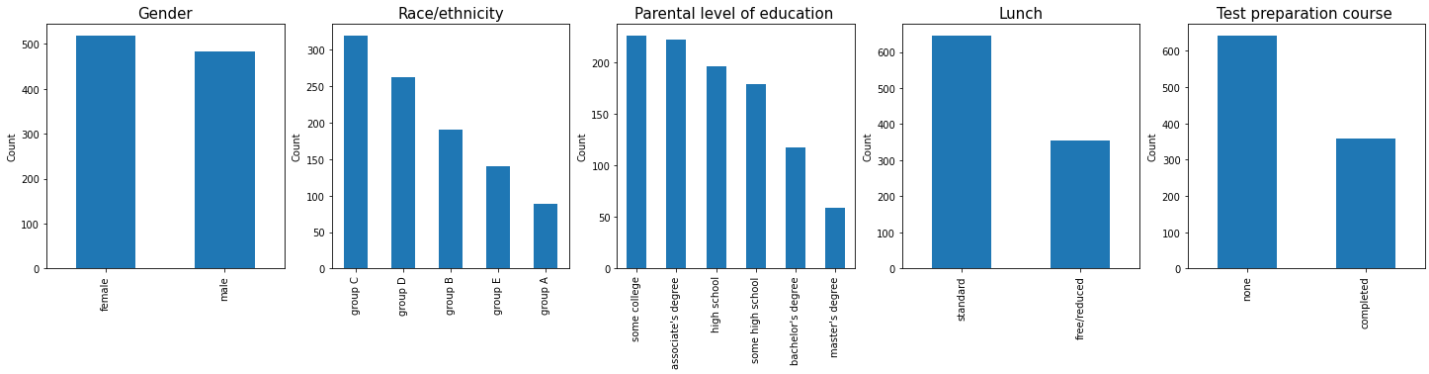
In [9]:

```
plt.figure(figsize=(25,10))
plt.subplot(2,5,1)
plt.title("Gender",fontsize=15)
plt.ylabel("Count")
data['gender'].value_counts().plot.bar()

plt.subplot(2,5,2)
plt.title("Race/ethnicity",fontsize=15)
plt.ylabel("Count")
data['race/ethnicity'].value_counts().plot.bar()

plt.subplot(2,5,3)
plt.title("Parental level of education ",fontsize=15)
plt.ylabel("Count")
data['parental level of education'].value_counts().plot.bar()

plt.subplot(2,5,4)
plt.title("Lunch",fontsize=15)
plt.ylabel("Count")
data['lunch'].value_counts().plot.bar()
```

```
plt.subplot(2,5,5)
plt.title("Test preparation course ",fontsize=15)
plt.ylabel("Count")
data['test preparation course'].value_counts().plot.bar()
```

Out[9]:

```
<AxesSubplot:title={'center':'Test preparation course '}, ylabel='Count'>
```



The following information can be inferred from the above graphs:

1) There are more females than males.

2) The maximum students belong to group C(more than 300) followed by groups D,B,E and A.

3) The students with their parents having masters degree are the least. Maximum parents have graduated from some college.

4) Over 300 people have free/reduced lunch whereas most of them have standard.

5) Majority of the students have not taken/completed any preparation course.

In [10]:

```
print("Q1 : 25 percentile")
Q1=data.quantile(0.25)
print(Q1)
print("Q2 : 25 Median")
Q2=data.quantile(0.250)
print(Q2)
print("Q3 : 75 percentile")
Q3=data.quantile(0.75)
print(Q3)
print("IQR : difference between 75th and 25th percentile")
IQR = Q3 - Q1
print(IQR)
```

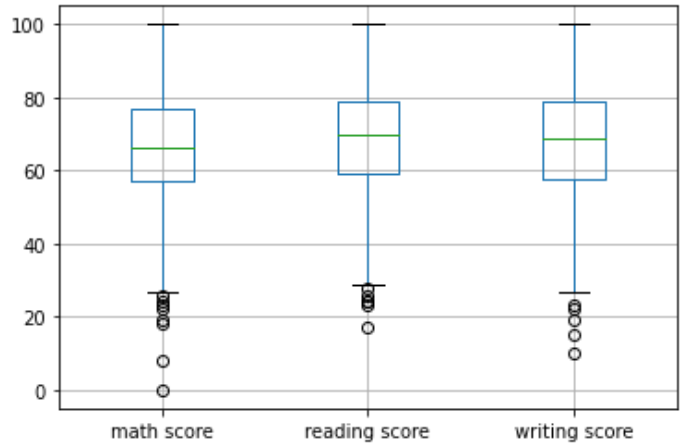
```
Q1 : 25 percentile
math score       57.00
reading score    59.00
writing score    57.75
Name: 0.25, dtype: float64
Q2 : 25 Median
math score       57.00
reading score    59.00
writing score    57.75
Name: 0.25, dtype: float64
Q3 : 75 percentile
math score       77.0
reading score    79.0
writing score    79.0
Name: 0.75, dtype: float64
IQR : difference between 75th and 25th percentile
math score       20.00
reading score    20.00
writing score    21.25
dtype: float64
```

In [11]:

```
data.boxplot(column=['math score','reading score','writing score'])
```

Out[11]:

<AxesSubplot:>



There are no outliers present in the data however there are few values which are exceptionally low than maximum values of the data.

In [12]:

```
data.describe()
```

Out[12]:

	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

The table shows the summary of the data.

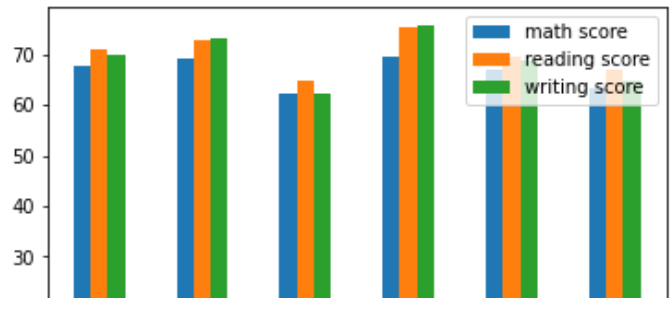
From a total of 1000 records, Maths has the least mean of scores and the least minimum score. The mean in two subjects are almost the same. Every subject has a student who has scored full marks.

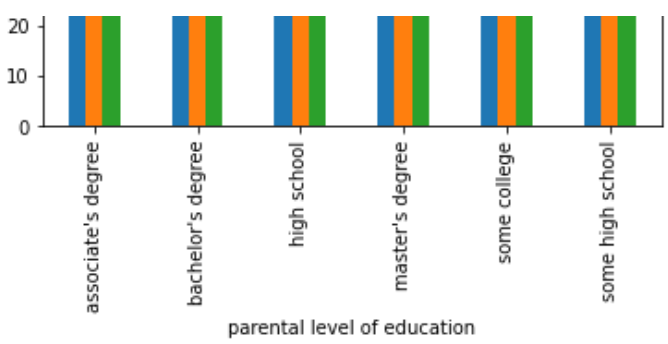
In [13]:

```
data.groupby('parental level of education').mean().plot.bar()
```

Out[13]:

<AxesSubplot:xlabel='parental level of education'>





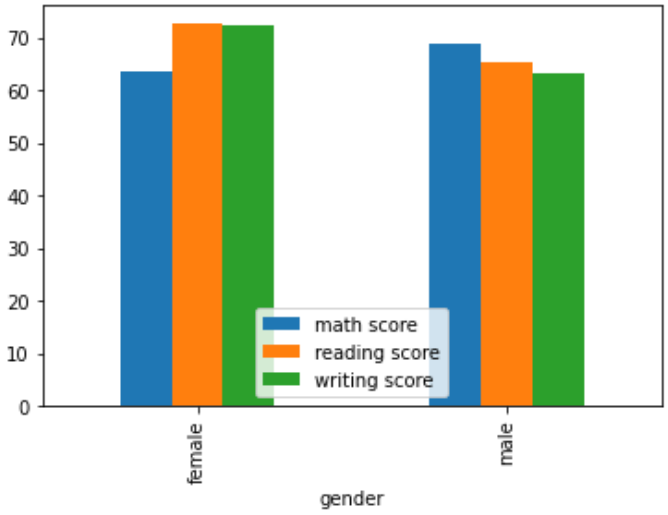
The mean of the subject score is greater in case of higher qualification of parents. In general, it can be concluded from the graph that Students with parents educated only till high school have less marks as compared to those whose parents have higher degree. Among the 3 subjects, maths has the least mean irrespective of the parent's degree which means that the students are better in reading and writing skills in comparison to maths if grouped on the basis of parent's level of education.

In [14]:

```
data.groupby('gender').mean().plot.bar()
```

Out[14]:

```
<AxesSubplot:xlabel='gender'>
```



The average maths score of female students is the lowest(compared to male students and the other 2 subjects of females) and their average reading score is highest, almost the same as their reading score.

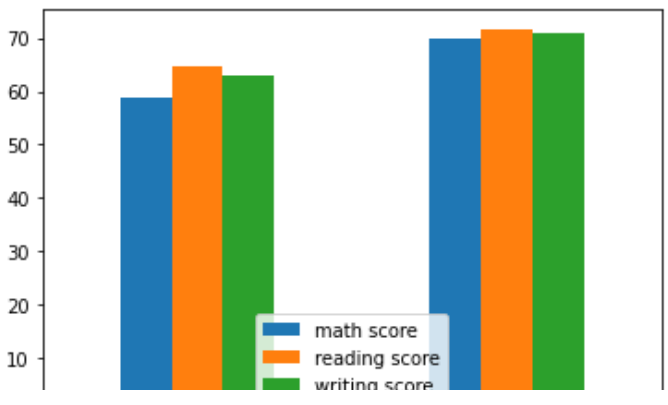
It means that the male students are better in maths compared to female students of the class. Also, among the math,reading and writing, male students are better in maths.

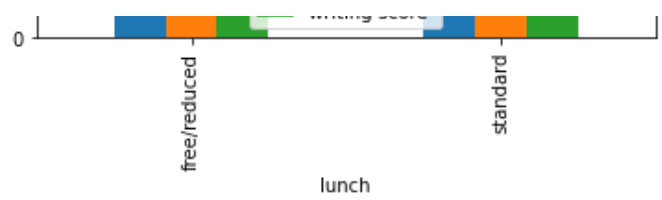
In [15]:

```
data.groupby('lunch').mean().plot.bar()
```

Out[15]:

```
<AxesSubplot:xlabel='lunch'>
```





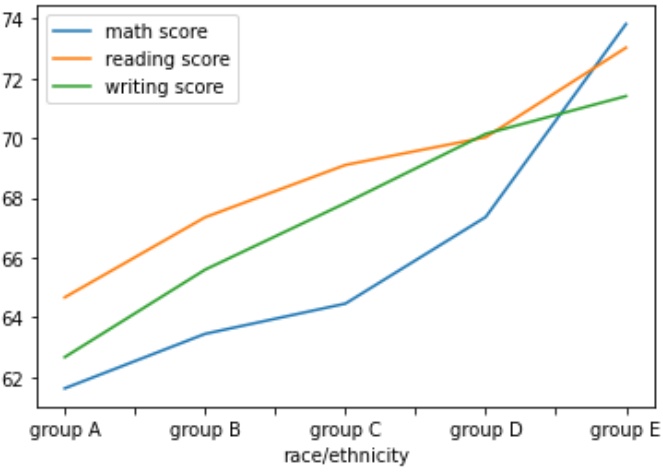
Between free and standard lunch, students with standard lunch are better in all 3 subjects. Students with free/reduced lunch are also poor in maths compared to their reading and writing average scores. The average score in all 3 subjects is almost the same for people with standard lunch.

In [16]:

```
data.groupby('race/ethnicity').mean().plot.line()
```

Out[16]:

```
<AxesSubplot:xlabel='race/ethnicity'>
```



Group E students are better in math,reading and writing followed by groups D,C,B and A.

Among the 3 average scores, students of all groups have more average in reading except group E whose maths mean is highest.

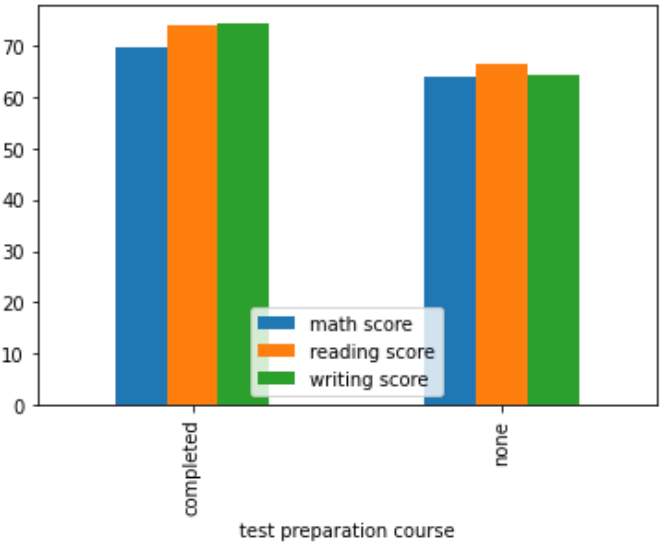
The writing average is between reading and maths in groups A,B and C. In group D, it is same as that of reading and it's lowest in group E.

In [17]:

```
data.groupby('test preparation course').mean().plot.bar()
```

Out[17]:

```
<AxesSubplot:xlabel='test preparation course'>
```



It can be seen clearly that students who have completed the test preparation course have performed better.

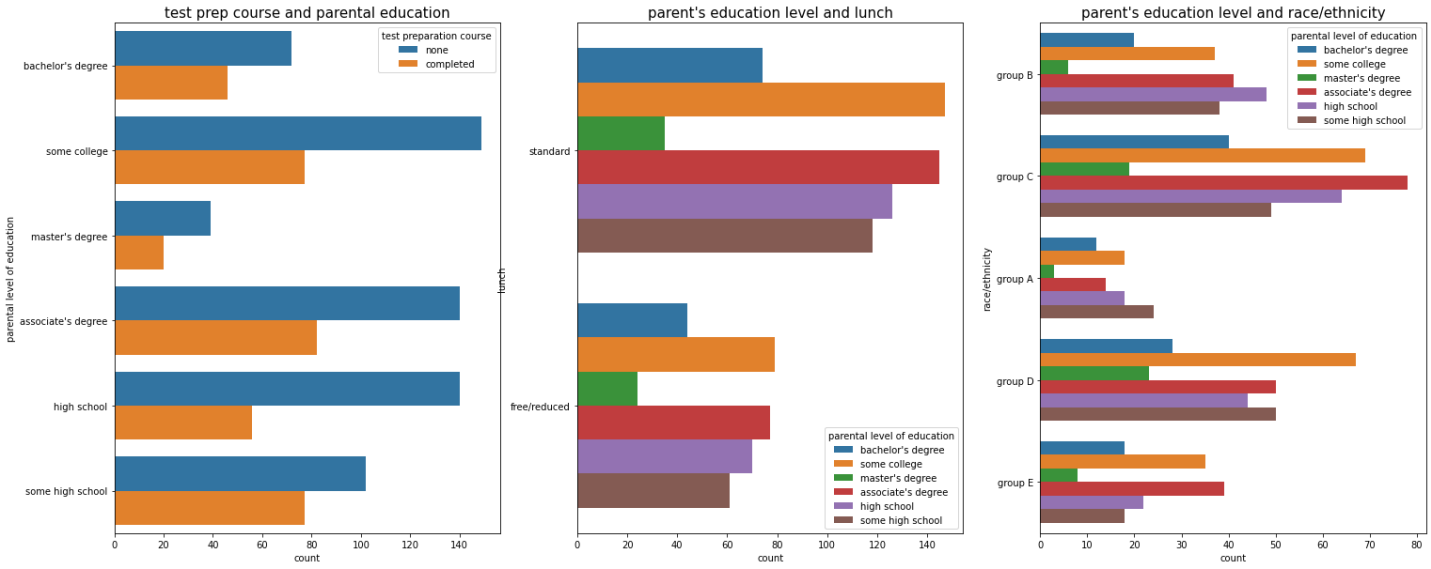
- **### Bivariate Analysis**

In [18]:

```
plt.figure(figsize=(25,10))
plt.subplot(1,3,1)
plt.title('test prep course and parental education',fontsize = 15)
sns.countplot(y='parental level of education', hue='test preparation course',data=data)
plt.subplot(132)
plt.title('parent\'s education level and lunch',fontsize = 15)
sns.countplot(y='lunch', hue='parental level of education',data=data)
plt.subplot(133)
plt.title('parent\'s education level and race/ethnicity',fontsize = 15)
sns.countplot(y='race/ethnicity', hue='parental level of education',data=data)
```

Out[18]:

```
<AxesSubplot:title={'center':"parent's education level and race/ethnicity"}, xlabel='coun
t', ylabel='race/ethnicity'>
```



1) The first graph suggests the relationship between parent's education and test preparation course. Most of the students who's parental level of education is some college, associate's degree, and high school have completed the test preparation course.

2) The second graph is lunch vs parental education.

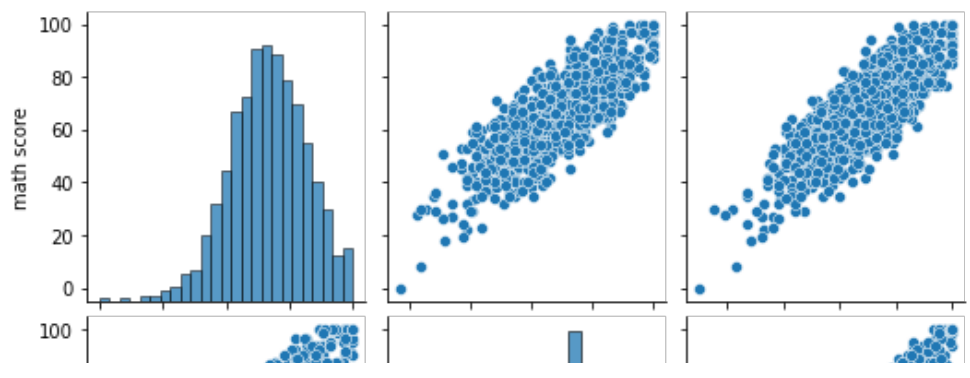
3) The graph of 'Race/ethnicity vs Parent's education' indicates that maximum group C students have their parents qualification associate degree followed by parents of some college.

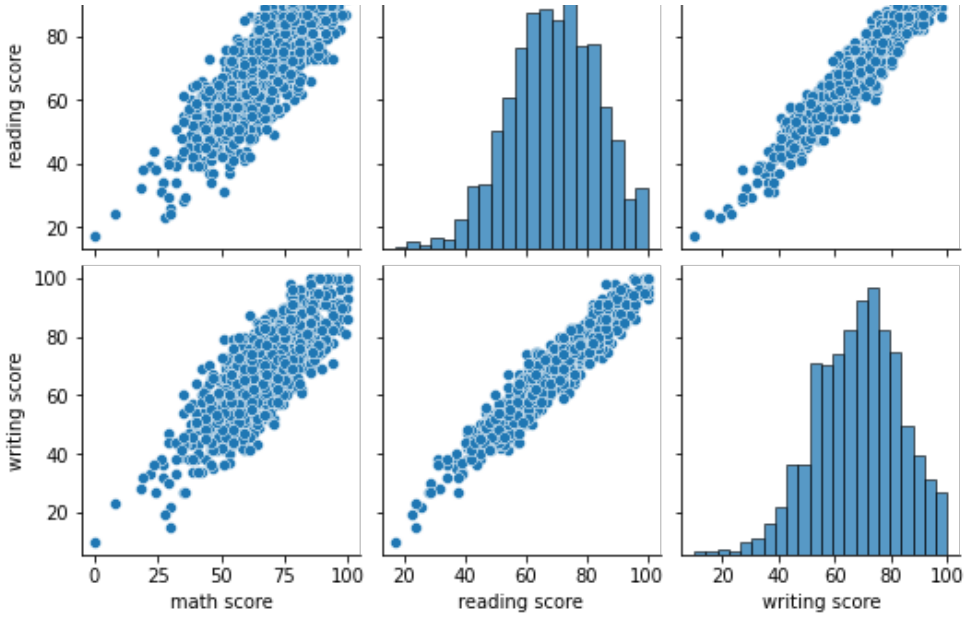
In [19]:

```
sns.pairplot(data)
```

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x7ff947c8c790>
```





It can be inferred that the student who scores good in reading also scores good in other 2 subjects and vice versa since there is a positive correlation between all 3 scores.

Now, converting certain categorical variables to numerical variables for furth

In [20]:

```
data['gender_male']=data.gender.map({'female':0,'male':1})
data.head()
```

Out[20]:

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

In [21]:

```
data['lunch_standard'] = data.lunch.map({'standard':1, 'free/reduced':0})
data.head()
```

Out[21]:

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

In [22]:

```
data['test_prep_course_completed'] = data['test preparation course'].map({'completed':1,
```

```
'none':0})
data.head()
```

Out[22]:

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

In [23]:

```
data['race_numerical'] = data['race/ethnicity'].map({'group A':1, 'group B':2,'group C':
3,'group D':4,'group E':5})
data.head()
```

Out[23]:

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

In [24]:

```
data['parent_edu_numerical'] = data['parental level of education'].map({'high school':1,
'some high school':2,'bachelor\'s degree':3,'some college':4,'associate\'s degree':5,'mas
ter\'s degree':6})
data.head()
```

Out[24]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	gender_male	lunch_standard	test_p
0	female	group B	bachelor's degree	standard	none	72	72	74	0	1	
1	female	group C	some college	standard	completed	69	90	88	0	1	
2	female	group B	master's degree	standard	none	90	95	93	0	1	
3	male	group A	associate's degree	free/reduced	none	47	57	44	1	0	

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	gender_male	lunch_standard	test_p
4	male	group C	some college	standard	none	76	78	75	1	1	

In [25]:

```
data.corr()
```

Out[25]:

	math score	reading score	writing score	gender_male	lunch_standard	test_prep_course_completed	race_nu
math score	1.000000	0.817580	0.802642	0.167982	0.350877	0.177702	
reading score	0.817580	1.000000	0.954598	-0.244313	0.229560	0.241780	
writing score	0.802642	0.954598	1.000000	-0.301225	0.245769	0.312946	
gender_male	0.167982	-0.244313	-0.301225	1.000000	0.021372	0.006028	-(
lunch_standard	0.350877	0.229560	0.245769	0.021372	1.000000	-0.017044	
test_prep_course_completed	0.177702	0.241780	0.312946	0.006028	-0.017044	1.000000	
race_numerical	0.216415	0.145253	0.165691	-0.001502	0.046563	0.017508	
parent_edu_numerical	0.153076	0.176331	0.211410	-0.045319	-0.007907	0.017294	

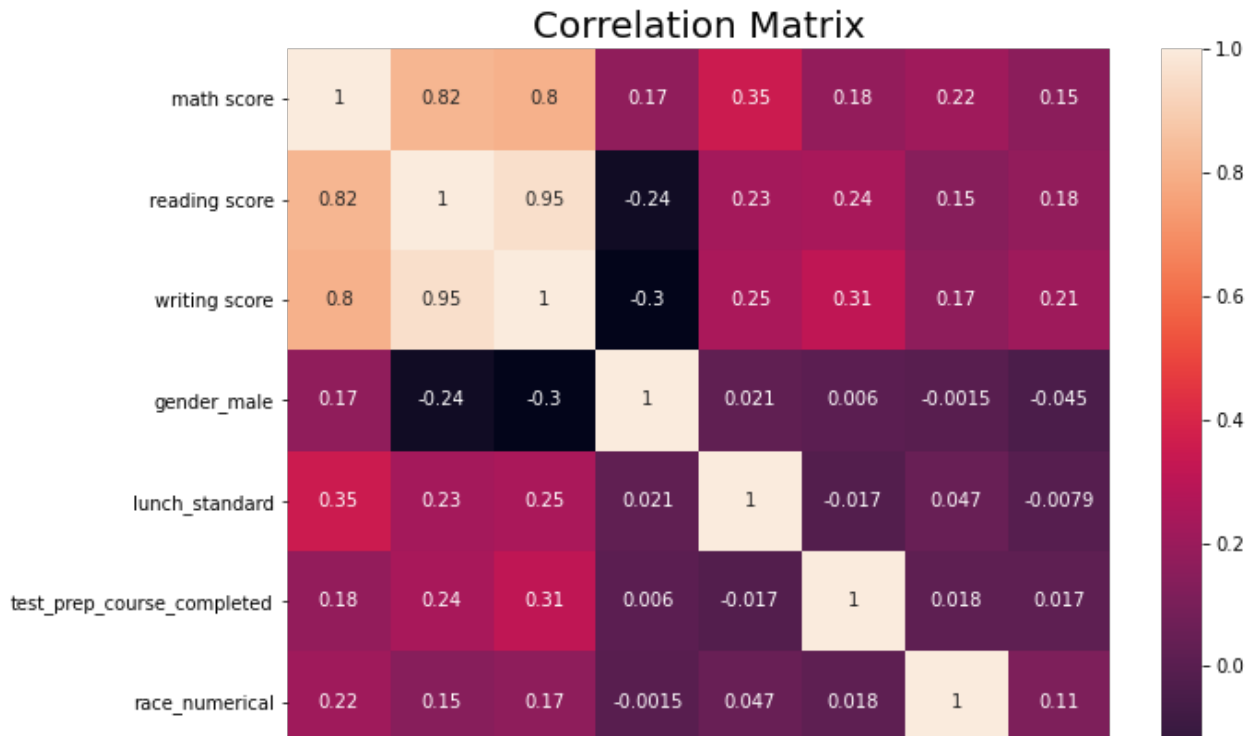
The above matrix shows all the possible correlations between numerical variables.

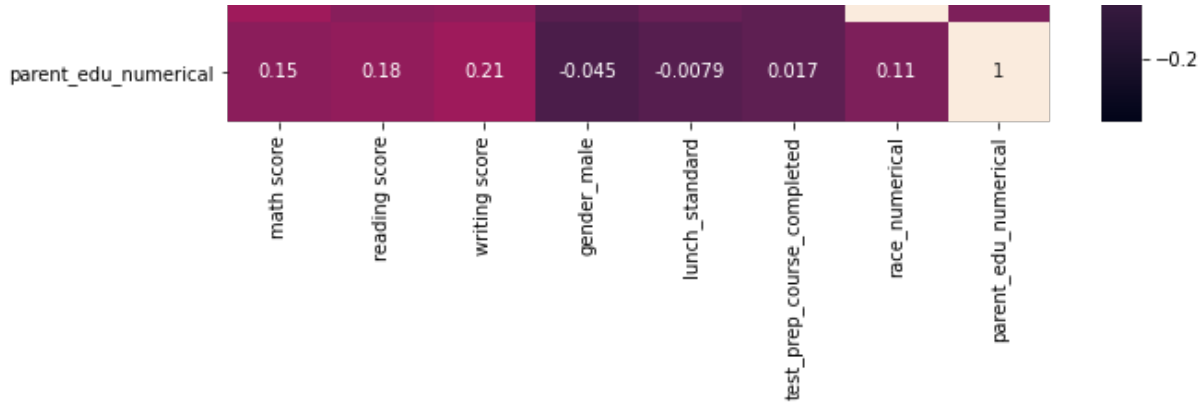
Negative correlation is a relationship between 2 variables in which as one variable increases, the other decreases and vice versa. A perfect negative correlation is -1, a 0 indicates no correlation, and a +1 indicates a perfect positive correlation.

All of them are positive except some like 'gender=male and writing score' and 'gender=male and reading score'. Interpretation: If gender is 1 ie. male, the reading and writing scores will be less.

In [26]:

```
plt.figure(figsize=(10,8))

plt.title("Correlation Matrix",fontsize=20)
sns.heatmap(data.corr(),annot=True)
plt.show()
```





In [27]:

```
maths_marks = ['0-10','11-20','21-30','31-40','41-50','51-60','61-70','71-80','81-90','91
-100']
data['math_grouping']=pd.cut(data['math score'],range(1,102,10),right=False,labels=maths
_marks)
print(data[['math score','math_grouping']].head(5))
reading_marks = ['0-10','11-20','21-30','31-40','41-50','51-60','61-70','71-80','81-90','
91-100']
data['reading_grouping'] = pd.cut(data['reading score'],range(1,102,10),right=False, lab
els=maths_marks)
print(data[['reading score','reading_grouping']].head(5))
maths_marks = ['0-10','11-20','21-30','31-40','41-50','51-60','61-70','71-80','81-90','91
-100']
data['writing_grouping'] = pd.cut(data['writing score'],range(1,102,10),right=False, lab
els=maths_marks)
print(data[['writing score','writing_grouping']].head(5))
```

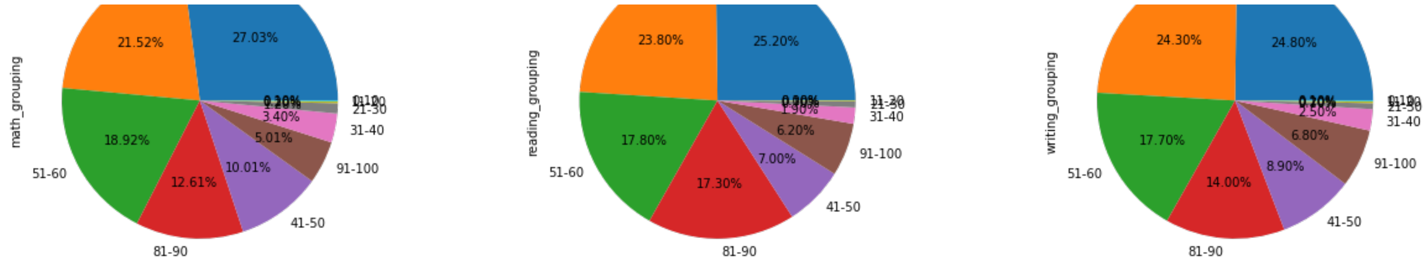
```
   math score math_grouping
0          72         71-80
1          69         61-70
2          90         81-90
3          47         41-50
4          76         71-80
   reading score reading_grouping
0             72            71-80
1             90            81-90
2             95           91-100
3             57            51-60
4             78            71-80
   writing score writing_grouping
0             74            71-80
1             88            81-90
2             93           91-100
3             44            41-50
4             75            71-80
```

In [28]:

```
plt.figure(figsize=(20,20))
plt.subplots_adjust(left=0.125, bottom=0.5, right=0.9, top=0.9, wspace=0.5, hspace=0.2)
plt.subplot(131)
data.math_grouping.value_counts().plot.pie(autopct="%0.2f%%")
plt.title("Maths",fontsize=15)
plt.subplot(132)
data.reading_grouping.value_counts().plot.pie(autopct="%0.2f%%")
plt.title("Reading",fontsize=15)
plt.subplot(133)
data.writing_grouping.value_counts().plot.pie(autopct="%0.2f%%")
plt.title("Writing",fontsize=15)
```

Out[28]:

```
Text(0.5, 1.0, 'Writing')
```



In maths, the maximum scores are from 61-70(27.03%).

The maximum scores in reading are from 71-80(25.20%). A similar trend is also seen in writing scores.

Now, let us see where the concentration of marks is in a given range.

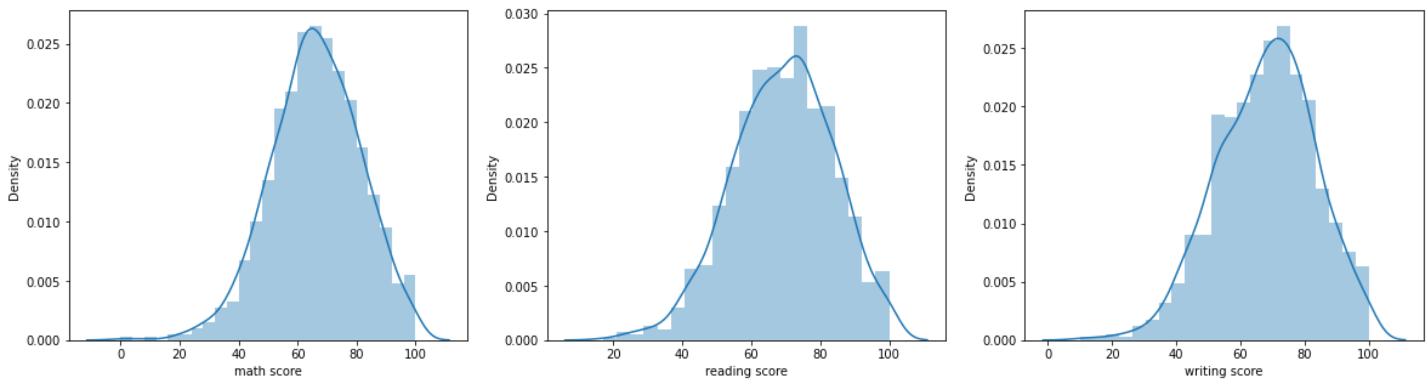
In [29]:

```
plt.figure(figsize=(20,5))
plt.subplot(1,3,1)
sns.distplot(data["math score"])
plt.subplot(132)
sns.distplot(data['reading score'])
plt.subplot(133)
sns.distplot(data['writing score'])
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[29]:

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

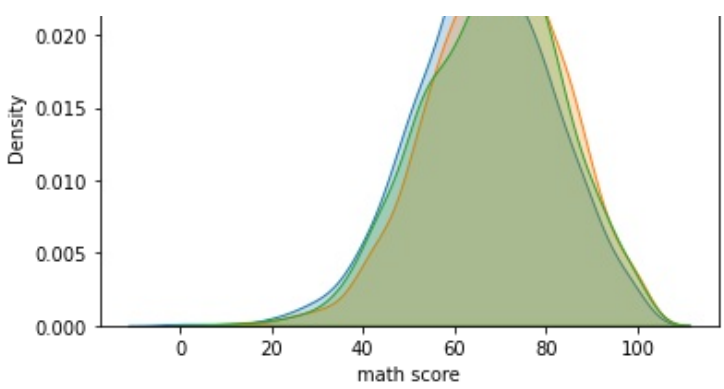


In [30]:

```
sns.kdeplot(data['math score'],shade=True)
sns.kdeplot(data['reading score'],shade=True)
sns.kdeplot(data['writing score'],shade=True)
```

Out[30]:

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



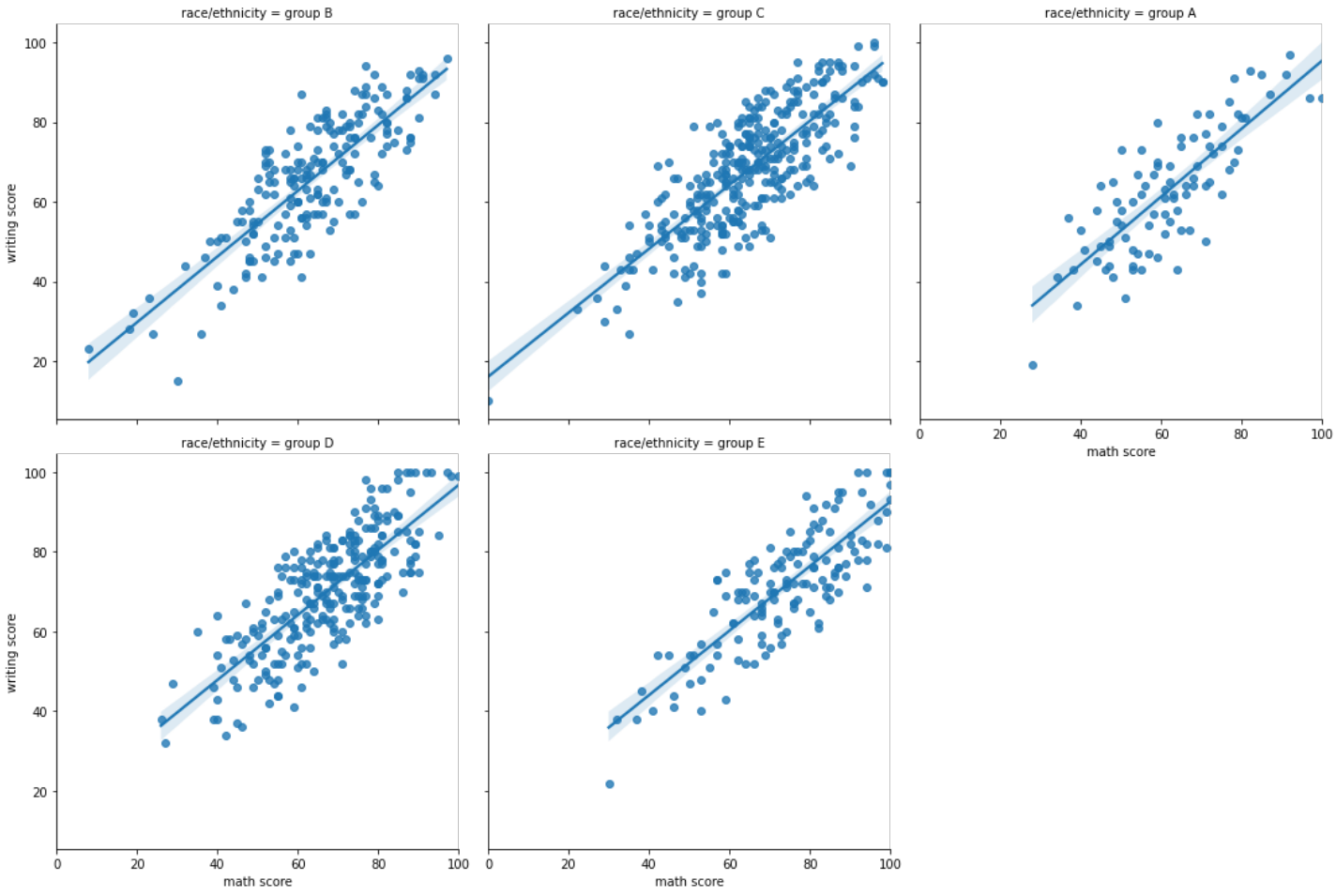
In [31]:

```
sns.lmplot('math score','writing score',data=data,col='race/ethnicity',col_wrap=3)
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will resu
lt in an error or misinterpretation.
  FutureWarning
```

Out[31]:

```
<seaborn.axisgrid.FacetGrid at 0x7ff944b6a5d0>
```



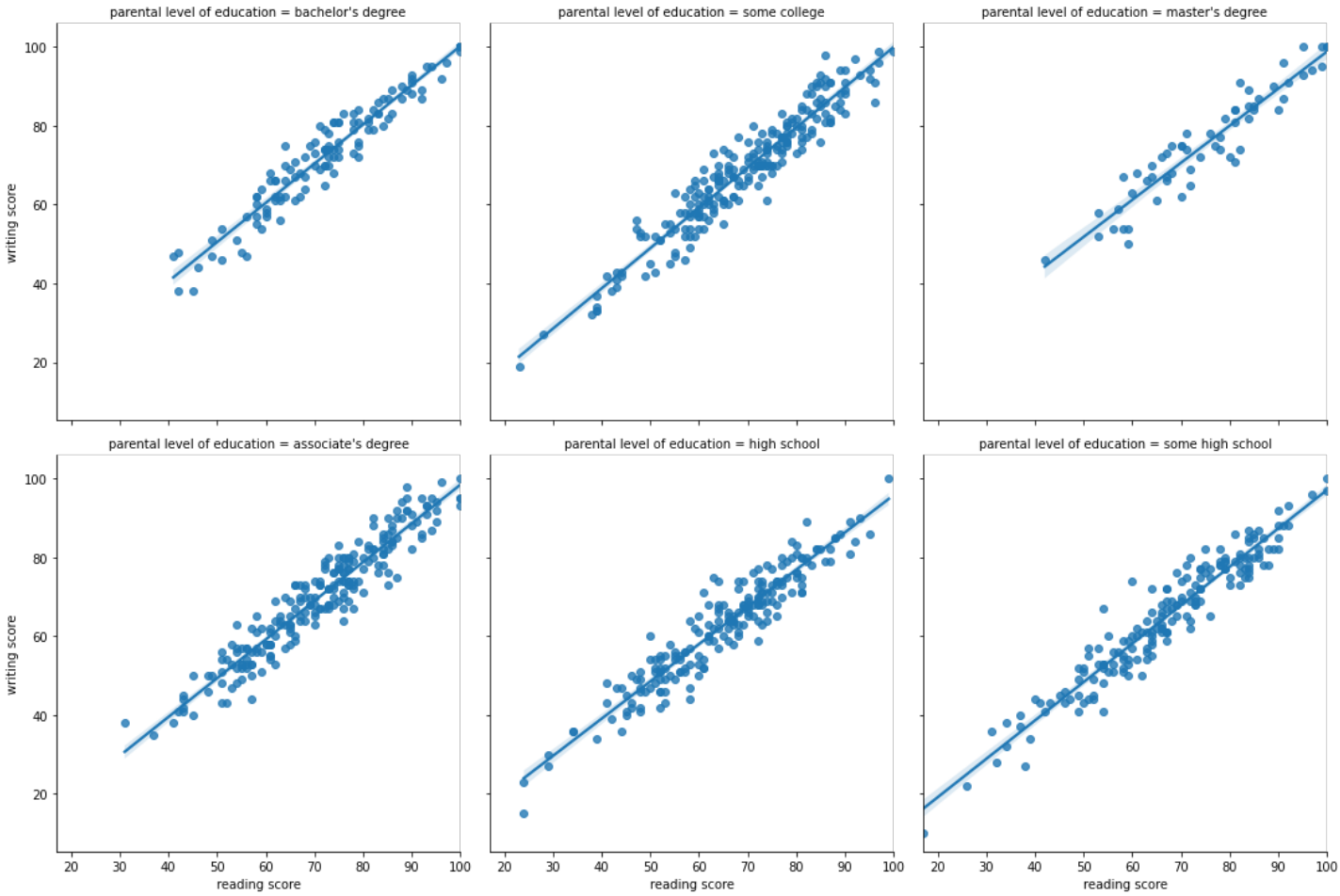
In [32]:

```
sns.lmplot('reading score','writing score',data=data,col='parental level of education',co
l_wrap=3)
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will resu
lt in an error or misinterpretation.
  FutureWarning
```

Out[32]:

```
<seaborn.axisgrid.FacetGrid at 0x7ff9448c9150>
```



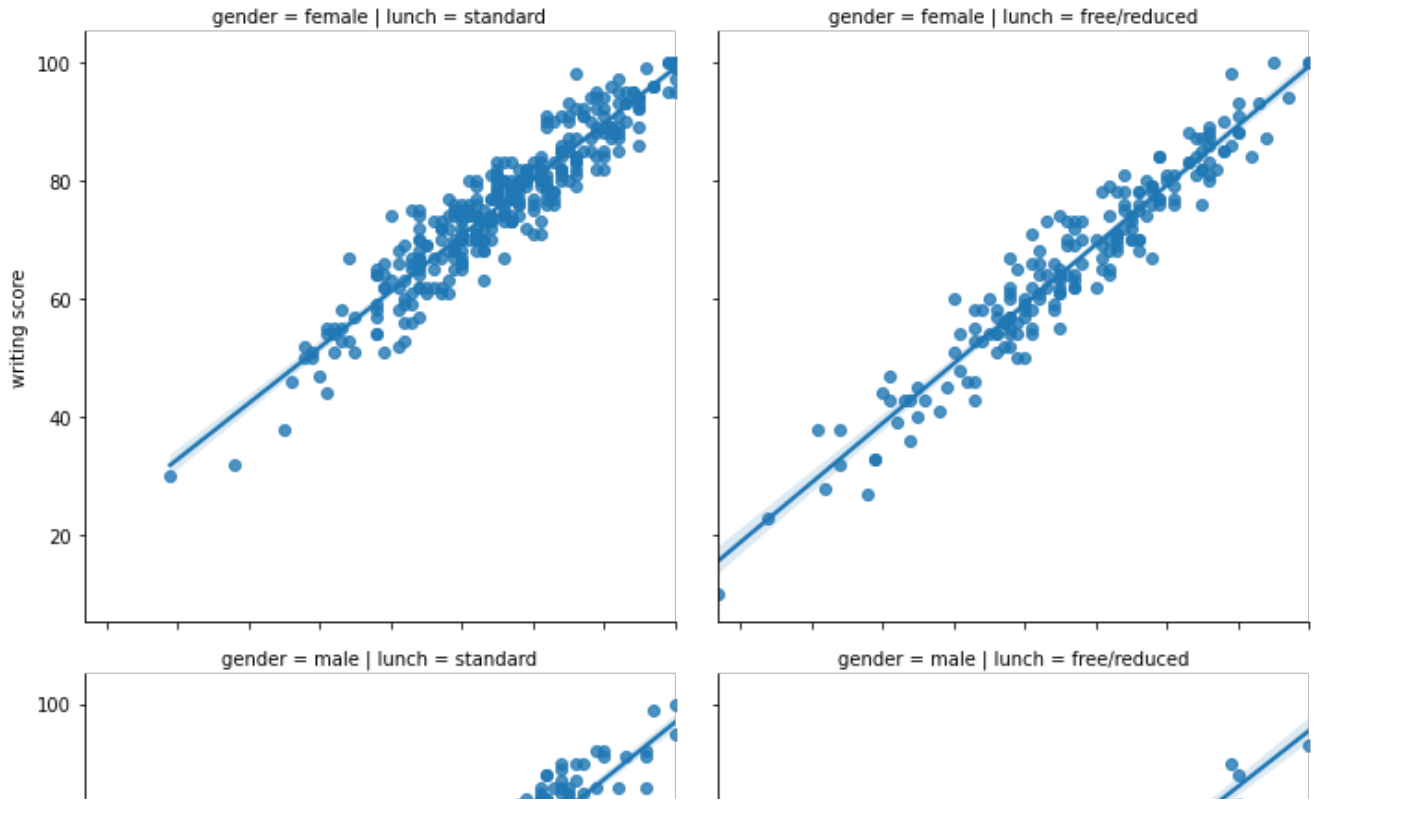
In [33]:

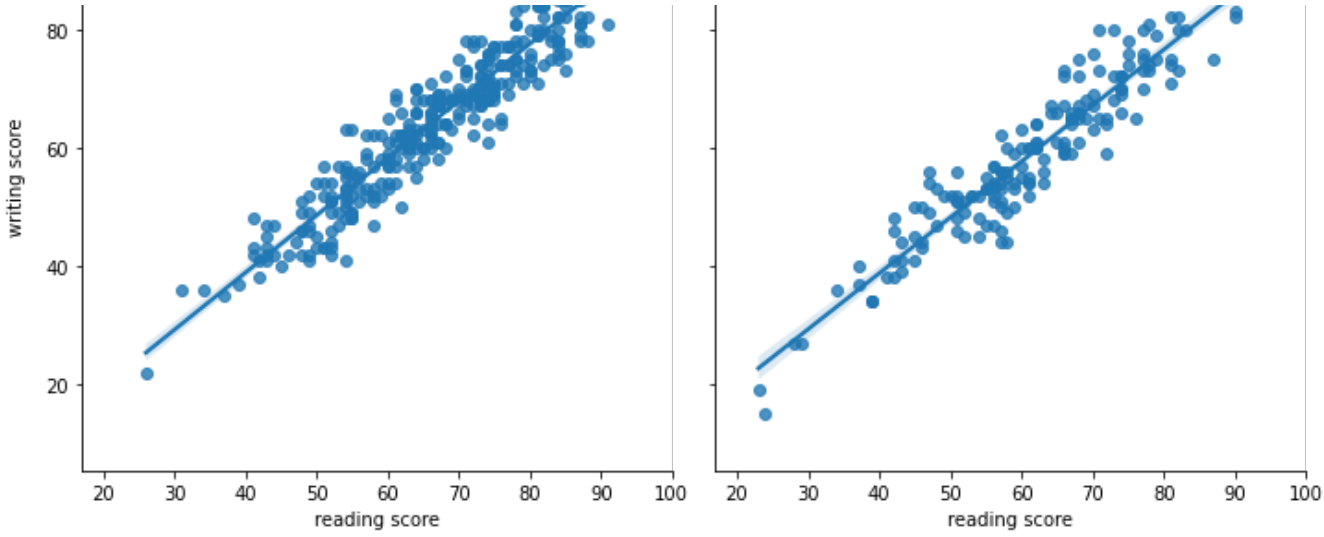
```
sns.lmplot('reading score','writing score',data=data,row='gender',col='lunch')
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[33]:

```
<seaborn.axisgrid.FacetGrid at 0x7ff9446552d0>
```





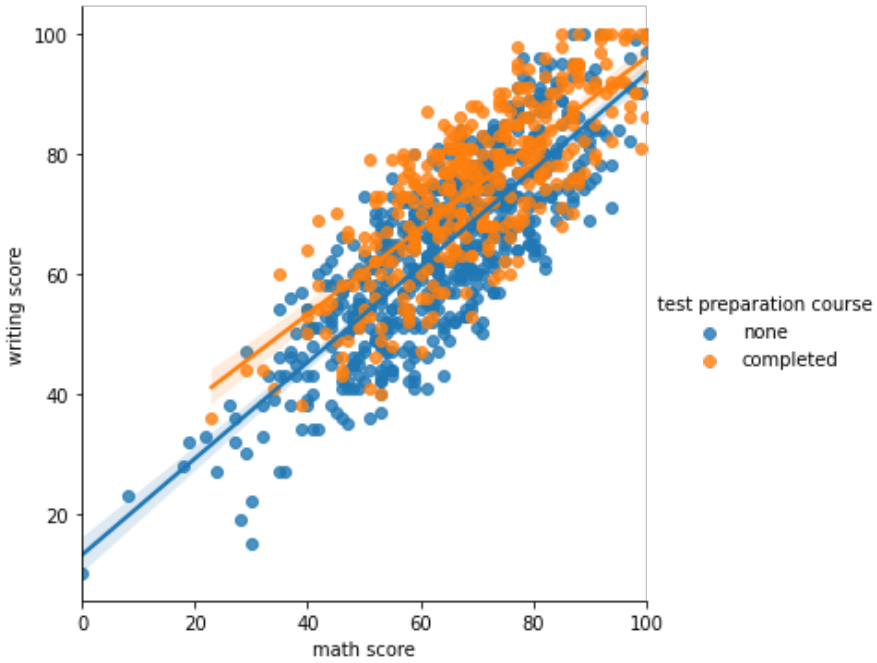
In [34]:

```
sns.lmplot('math score','writing score',data=data,hue='test preparation course')
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[34]:

```
<seaborn.axisgrid.FacetGrid at 0x7ff94443bdd0>
```



All the above lmplots suggest a regression line betweeen different variables to fit the model. The data is good fit for linear regression model with less variance. However, there are a few points in some graphs far away from the line which can be considered as outliers.

Grading

Let us grade the students according to the marks obtained -

Above 80 = A Grade

70 to 80 = B Grade

60 to 70 = C Grade

50 to 60 = D Grade

40 to 50 = E Grade

Below 40 = Fail

In [35]:

```
data['total_marks']=data['math score']+data['reading score']+data['writing score']
data['percentage']=data['total_marks']/300*100
data['percentage'].head()
```

Out[35]:

```
0    72.666667
1    82.333333
2    92.666667
3    49.333333
4    76.333333
Name: percentage, dtype: float64
```

In [36]:

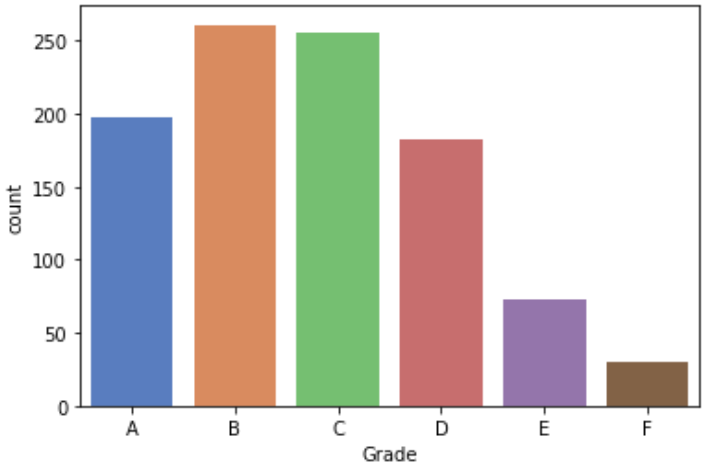
```
def grade(Percentage):
    if(Percentage>=80):
        return 'A'
    if ( Percentage >= 70):
        return 'B'
    if ( Percentage >= 60):
        return 'C'
    if ( Percentage >= 50):
        return 'D'
    if ( Percentage >= 40):
        return 'E'
    else:
        return 'F'
data['Grade']=data['percentage'].apply(grade)
data.Grade.value_counts()
```

Out[36]:

```
B    261
C    256
A    198
D    182
E     73
F     30
Name: Grade, dtype: int64
```

In [37]:

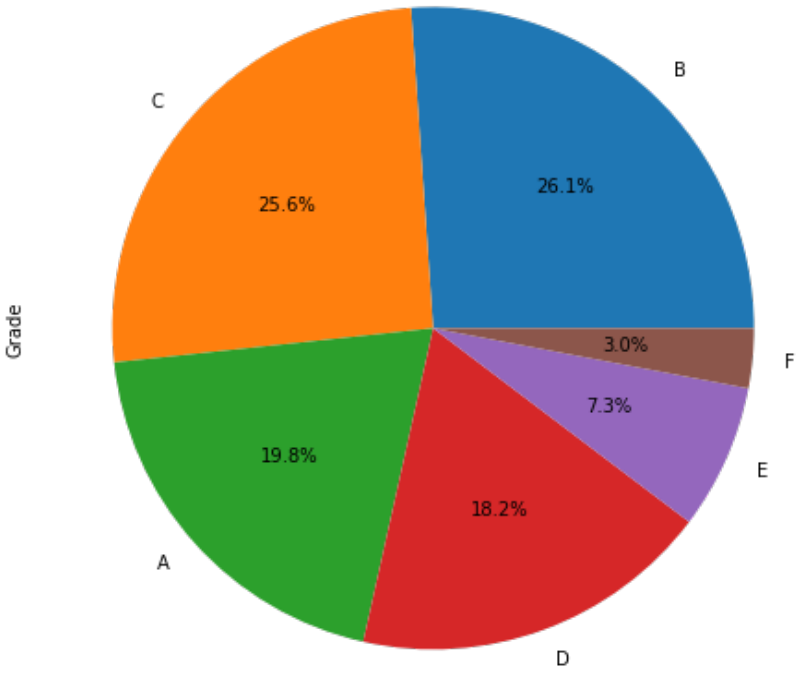
```
sns.countplot(x="Grade",data=data,order=['A','B','C','D','E','F'],palette="muted")
plt.show()
```



In [38]:

```
plt.figure(figsize=(10,8))
```

```
data['Grade'].value_counts().plot.pie(autopct="%1.1f%%")
plt.show()
```



Visualising grades as per race/ethnicity usimg cross tabs

In [39]:

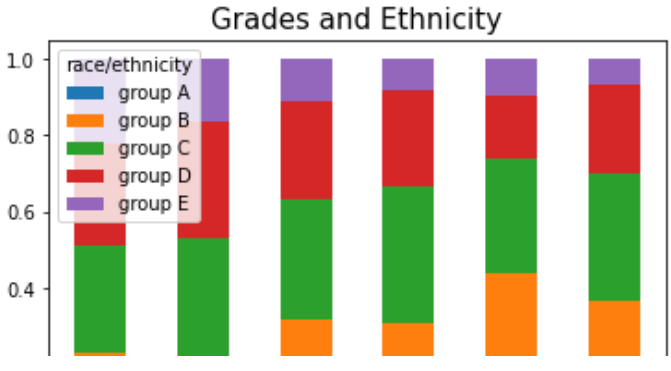
```
gr=pd.crosstab(data["Grade"],data["race/ethnicity"],normalize=0)
gr
```

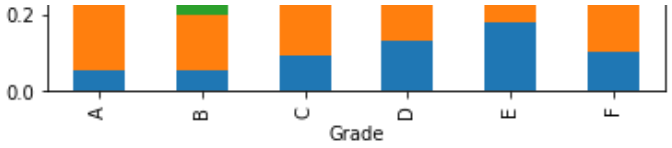
Out[39]:

race/ethnicity	group A	group B	group C	group D	group E
Grade					
A	0.055556	0.176768	0.277778	0.267677	0.222222
B	0.053640	0.145594	0.333333	0.302682	0.164751
C	0.093750	0.226562	0.312500	0.253906	0.113281
D	0.131868	0.175824	0.357143	0.252747	0.082418
E	0.178082	0.260274	0.301370	0.164384	0.095890
F	0.100000	0.266667	0.333333	0.233333	0.066667

In [40]:

```
gr.plot.bar(stacked=True)
plt.title('Grades and Ethnicity',fontsize=15)
plt.show()
```





3. Prediction and Training of Scores

In [41]:

```
X = data[['gender_male','race_numerical','parent_edu_numerical','lunch_standard','test_prep_course_completed','math score','reading score']]
y = data['writing score']
```

In [42]:

```
from sklearn import model_selection
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)
```

In [43]:

```
from sklearn.ensemble import RandomForestRegressor
model=RandomForestRegressor(n_estimators=100)
model.fit(X_train,y_train)
preds=model.predict(X_test)
```

In [44]:

```
from sklearn import metrics
from sklearn.metrics import r2_score

print('MAE:', metrics.mean_absolute_error(y_test, preds))
print('MSE:', metrics.mean_squared_error(y_test, preds))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, preds)))
print("R_square score: ", r2_score(y_test,preds))
```

```
MAE: 3.3613333333333335
MSE: 17.373915259259263
RMSE: 4.168202881249815
R_square score:  0.9216450435243129
```