Student Performance Indicator

Part 1: Life cycle of Machine learning Project

- Understanding the Problem Statement
- Data Collection
- Data Checks to perform
- Exploratory data analysis
- Data Pre-Processing
- Model Training
- Choose best model

1) Problem statement

- This project understands how the student's performance (test scores) is affected by other variables such as Gender, Ethnicity, Parental level of education, Lunch and Test preparation course.

2) Data Collection

- Dataset Source - https://www.kaggle.com/datasets/spscientist/students-performance-in-exams?datasetId=74977
- The data consists of 8 column and 1000 rows.

2.1 Import Data and Required Packages

Importing Pandas, Numpy, Matplotlib, Seaborn and Warings Library.

In [7]:
```
pip install catboost
```

```
Requirement already satisfied: catboost in c:\users\ravi\anaconda3\lib\site-packag
es (1.2.2)
Requirement already satisfied: graphviz in c:\users\ravi\anaconda3\lib\site-packag
es (from catboost) (0.20.1)
Requirement already satisfied: matplotlib in c:\users\ravi\anaconda3\lib\site-pack
ages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in c:\users\ravi\anaconda3\lib\site-p
ackages (from catboost) (1.24.3)
Requirement already satisfied: pandas>=0.24 in c:\users\ravi\anaconda3\lib\site-pa
ckages (from catboost) (1.5.3)
Requirement already satisfied: scipy in c:\users\ravi\anaconda3\lib\site-packages
(from catboost) (1.10.1)
Requirement already satisfied: plotly in c:\users\ravi\anaconda3\lib\site-packages
(from catboost) (5.9.0)
Requirement already satisfied: six in c:\users\ravi\anaconda3\lib\site-packages (f
rom catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\ravi\anaconda3\l
ib\site-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\ravi\anaconda3\lib\site-pa
ckages (from pandas>=0.24->catboost) (2022.7)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\ravi\anaconda3\lib\sit
e-packages (from matplotlib->catboost) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\ravi\anaconda3\lib\site-pa
ckages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\ravi\anaconda3\lib\si
te-packages (from matplotlib->catboost) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\ravi\anaconda3\lib\si
te-packages (from matplotlib->catboost) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\ravi\anaconda3\lib\site
-packages (from matplotlib->catboost) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\ravi\anaconda3\lib\site-p
ackages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\ravi\anaconda3\lib\sit
e-packages (from matplotlib->catboost) (3.0.9)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\ravi\anaconda3\lib\site
-packages (from plotly->catboost) (8.2.2)
Note: you may need to restart the kernel to use updated packages.
```

In [58]:

```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\ravi\anaconda3\lib\site-package
s (2.0.3)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: numpy in c:\users\ravi\anaconda3\lib\site-packages
(from xgboost) (1.24.3)
Requirement already satisfied: scipy in c:\users\ravi\anaconda3\lib\site-packages
(from xgboost) (1.10.1)
```

In [59]:

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
# for modelling
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression, Ridge,Lasso
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.model_selection import RandomizedSearchCV
```

```python
from catboost import CatBoostRegressor
from xgboost import XGBRegressor
import warnings
```

Import the CSV Data as Pandas DataFrame

In [9]:
```python
df = pd.read_csv('stud.csv')
```

Show Top 5 Records

In [11]:
```python
df.head()
```

Out[11]:

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course	math_s
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	

Shape of the dataset

In [12]:
```python
df.shape
```

Out[12]:
```
(1000, 8)
```

2.2 Dataset information

- gender : sex of students -> (Male/female)
- race/ethnicity : ethnicity of students -> (Group A, B,C, D,E)
- parental level of education : parents' final education ->(bachelor's degree,some college,master's degree,associate's degree,high school)
- lunch : having lunch before test (standard or free/reduced)
- test preparation course : complete or not complete before test
- math score
- reading score
- writing score

3. Data Checks to perform

- Check Missing values
- Check Duplicates
- Check data type
- Check the number of unique values of each column
- Check statistics of data set
- Check various categories present in the different categorical column

3.1 Check Missing values

In [13]:
```
df.isna().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
```

There are no missing values in the data set

3.2 Check Duplicates

In [14]:
```
df.duplicated().sum()
```

Out[14]:
```
0
```

There are no duplicates values in the data set

3.3 Check data types

In [15]:
```
# Check Null and Dtypes
df.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
```

3.4 Checking the number of unique values of each column

In [16]:
```
df.nunique()
```

Out[16]:
```
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
```

3.5 Check statistics of data set

In [17]:
```
df.describe()
```

Out[17]:

	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

Insight

- From above description of numerical data, all means are very close to each other - between 66 and 68.05;
- All standard deviations are also close - between 14.6 and 15.19;
- While there is a minimum score 0 for math, for writing minimum is much higher = 10 and for reading myet higher = 17

3.7 Exploring Data

In [18]:
```
df.head()
```

Out[18]:

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course	math_s
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	

In [19]:
```
print("Categories in 'gender' variable:     ",end=" " )
print(df['gender'].unique())

print("Categories in 'race_ethnicity' variable:  ",end=" ")
print(df['race_ethnicity'].unique())

print("Categories in'parental level of education' variable:",end=" " )
print(df['parental_level_of_education'].unique())

print("Categories in 'lunch' variable:     ",end=" " )
print(df['lunch'].unique())
```

```
print("Categories in 'test preparation course' variable:     ",end=" " )
print(df['test_preparation_course'].unique())
```

```
Categories in 'gender' variable:      ['female' 'male']
Categories in 'race_ethnicity' variable:   ['group B' 'group C' 'group A' 'group
D' 'group E']
Categories in'parental level of education' variable: ["bachelor's degree" 'some co
llege' "master's degree" "associate's degree"
 'high school' 'some high school']
Categories in 'lunch' variable:      ['standard' 'free/reduced']
Categories in 'test preparation course' variable:      ['none' 'completed']
```

In [20]:

```
# define numerical & categorical columns
numeric_features = [feature for feature in df.columns if df[feature].dtype != 'O']
categorical_features = [feature for feature in df.columns if df[feature].dtype == '

# print columns
print('We have {} numerical features : {}'.format(len(numeric_features), numeric_fe
print('\nWe have {} categorical features : {}'.format(len(categorical_features), ca
```

```
We have 3 numerical features : ['math_score', 'reading_score', 'writing_score']

We have 5 categorical features : ['gender', 'race_ethnicity', 'parental_level_of_e
ducation', 'lunch', 'test_preparation_course']
```

In [21]:

```
df.head(2)
```

Out[21]:

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course	math_scor
0	female	group B	bachelor's degree	standard	none	7
1	female	group C	some college	standard	completed	6

3.8 Adding columns for "Total Score" and "Average"

In [22]:

```
df['total score'] = df['math_score'] + df['reading_score'] + df['writing_score']
df['average'] = df['total score']/3
df.head()
```

Out[22]:

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course	math_s
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	

In [23]:

```
reading_full = df[df['reading_score'] == 100]['average'].count()
writing_full = df[df['writing_score'] == 100]['average'].count()
math_full = df[df['math_score'] == 100]['average'].count()

print(f'Number of students with full marks in Maths: {math_full}')
print(f'Number of students with full marks in Writing: {writing_full}')
print(f'Number of students with full marks in Reading: {reading_full}')
```

```
Number of students with full marks in Maths: 7
Number of students with full marks in Writing: 14
Number of students with full marks in Reading: 17
```

In [24]:

```
reading_less_20 = df[df['reading_score'] <= 20]['average'].count()
writing_less_20 = df[df['writing_score'] <= 20]['average'].count()
math_less_20 = df[df['math_score'] <= 20]['average'].count()

print(f'Number of students with less than 20 marks in Maths: {math_less_20}')
print(f'Number of students with less than 20 marks in Writing: {writing_less_20}')
print(f'Number of students with less than 20 marks in Reading: {reading_less_20}')
```

```
Number of students with less than 20 marks in Maths: 4
Number of students with less than 20 marks in Writing: 3
Number of students with less than 20 marks in Reading: 1
```

Insights

- From above values we get students have performed the worst in Maths
- Best performance is in reading section

4. Exploring Data (Visualization)

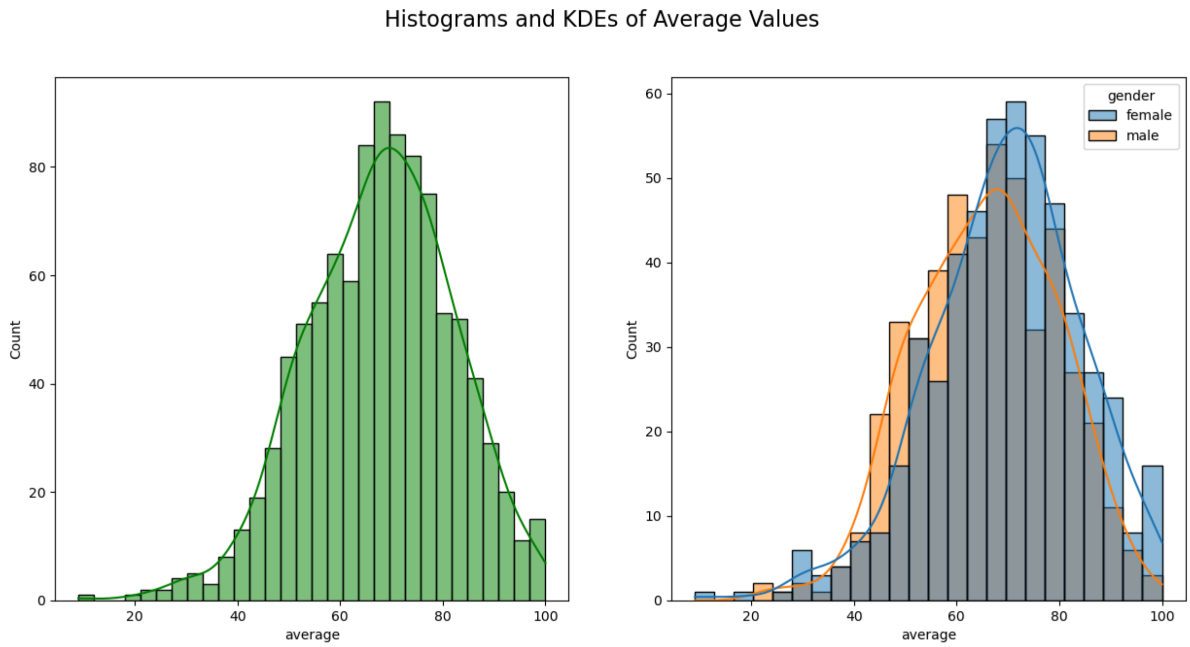
4.1 Visualize average score distribution to make some conclusion.

- Histogram
- Kernel Distribution Function (KDE)

4.1.1 Histogram & KDE

In [25]:

```
fig, axs = plt.subplots(1, 2, figsize=(15, 7))
plt.subplot(121)
sns.histplot(data=df,x='average',bins=30,kde=True,color='g')
plt.subplot(122)
sns.histplot(data=df,x='average',kde=True,hue='gender')
plt.suptitle('Histograms and KDEs of Average Values', fontsize=16)
plt.show()
```

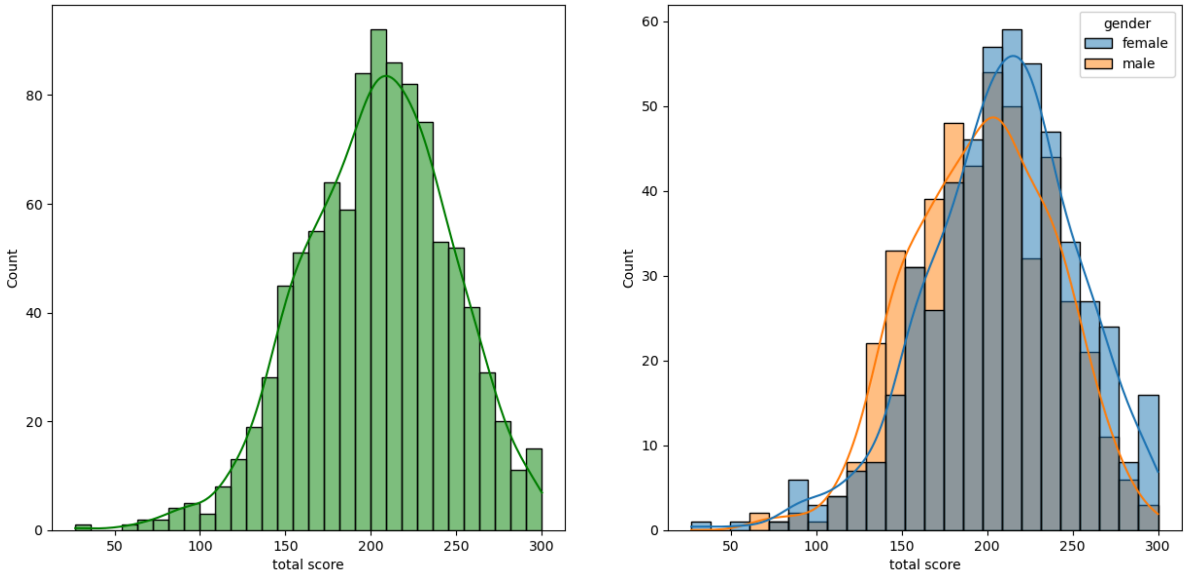


In [26]:

```
fig, axs = plt.subplots(1, 2, figsize=(15, 7))
plt.subplot(121)
```

```
sns.histplot(data=df,x='total score',bins=30,kde=True,color='g')
plt.subplot(122)
sns.histplot(data=df,x='total score',kde=True,hue='gender')
plt.suptitle('Histograms and KDEs of Total scores', fontsize=16)
plt.show()
```

Histograms and KDEs of Total scores

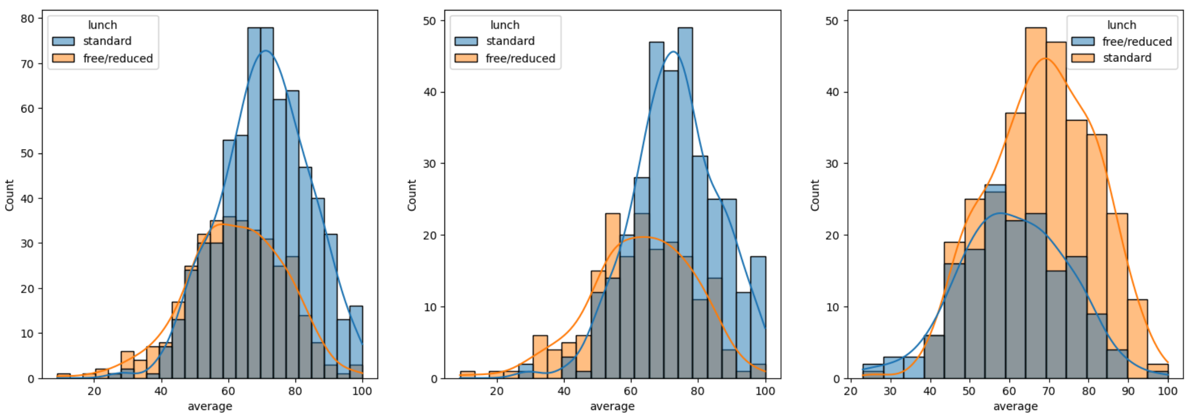


Insights

- Female students tend to perform well then male students.

In [27]:
```
plt.subplots(1,3,figsize=(25,6))
plt.subplot(141)
sns.histplot(data=df,x='average',kde=True,hue='lunch')
plt.subplot(142)
sns.histplot(data=df[df.gender=='female'],x='average',kde=True,hue='lunch')
plt.subplot(143)
sns.histplot(data=df[df.gender=='male'],x='average',kde=True,hue='lunch')
plt.suptitle('Histograms and KDEs of male and female performance', fontsize=16)
plt.show()
```

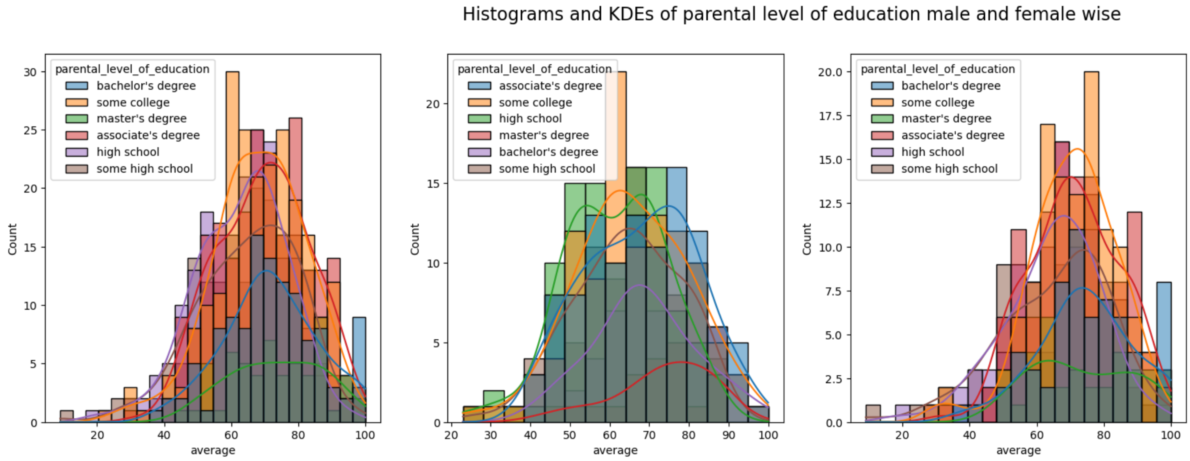
Histograms and KDEs of male and female performance



Insights

- Standard lunch helps perform well in exams.
- Standard lunch helps perform well in exams be it a male or a female.

In [28]:
```
plt.subplots(1,3,figsize=(25,6))
plt.subplot(141)
ax =sns.histplot(data=df,x='average',kde=True,hue='parental_level_of_education')
plt.subplot(142)
ax =sns.histplot(data=df[df.gender=='male'],x='average',kde=True,hue='parental_leve
plt.subplot(143)
ax =sns.histplot(data=df[df.gender=='female'],x='average',kde=True,hue='parental_le
plt.suptitle('Histograms and KDEs of parental level of education male and female wi
plt.show()
```



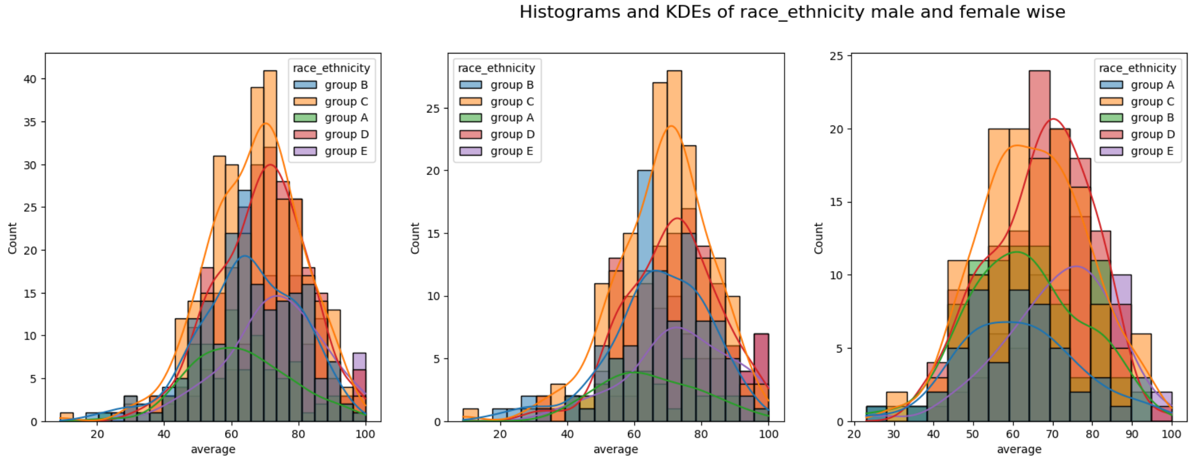
Insights

- In general parent's education don't help student perform well in exam.
- 2nd plot shows that parent's whose education is of associate's degree or master's degree their male child tend to perform well in exam
- 3rd plot we can see there is no effect of parent's education on female students.

In [29]:
```
df.columns
```

Out[29]:
```
Index(['gender', 'race_ethnicity', 'parental_level_of_education', 'lunch',
       'test_preparation_course', 'math_score', 'reading_score',
       'writing_score', 'total score', 'average'],
      dtype='object')
```

In [26]:
```
plt.subplots(1,3,figsize=(25,6))
plt.subplot(141)
ax =sns.histplot(data=df,x='average',kde=True,hue='race_ethnicity')
plt.subplot(142)
ax =sns.histplot(data=df[df.gender=='female'],x='average',kde=True,hue='race_ethnic
plt.subplot(143)
ax =sns.histplot(data=df[df.gender=='male'],x='average',kde=True,hue='race_ethnicit
plt.suptitle('Histograms and KDEs of race_ethnicity male and female wise', fontsize
plt.show()
```



Insights

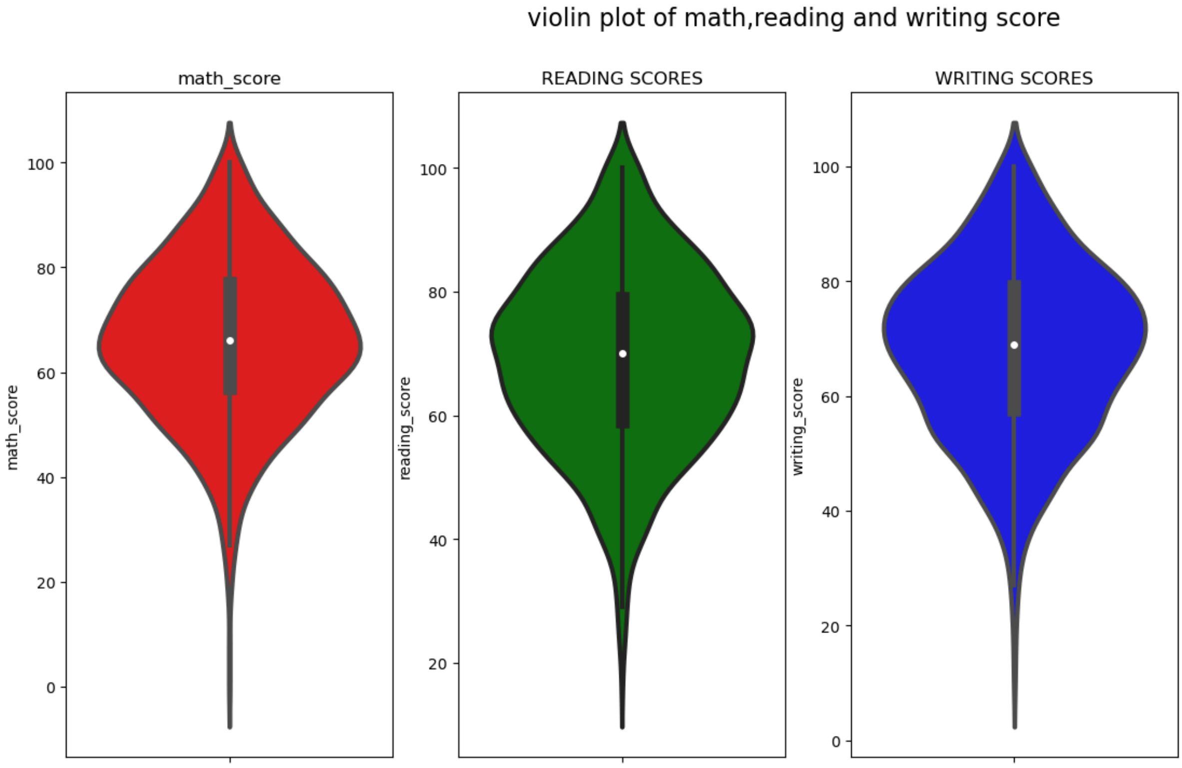
- Students of group A and group B tends to perform poorly in exam.
- Students of group A and group B tends to perform poorly in exam irrespective of whether they are male or female

4.2 Maximumum score of students in all three subjects

In [27]:
```
df.columns
```

Out[27]:
```
Index(['gender', 'race_ethnicity', 'parental_level_of_education', 'lunch',
       'test_preparation_course', 'math_score', 'reading_score',
       'writing_score', 'total score', 'average'],
      dtype='object')
```

In [30]:
```
plt.figure(figsize=(18,8))
plt.subplot(1, 4, 1)
plt.title('math_score')
sns.violinplot(y='math_score',data=df,color='red',linewidth=3)
plt.subplot(1, 4, 2)
plt.title('READING SCORES')
sns.violinplot(y='reading_score',data=df,color='green',linewidth=3)
plt.subplot(1, 4, 3)
plt.title('WRITING SCORES')
sns.violinplot(y='writing_score',data=df,color='blue',linewidth=3)
plt.suptitle('violin plot of math,reading and writing score', fontsize=16)
plt.show()
```



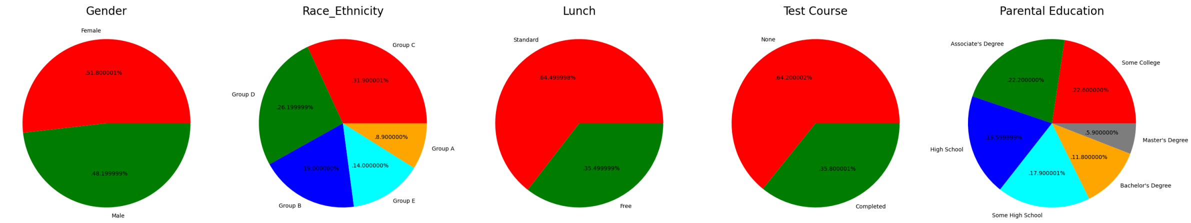
Insights

- From the above three plots its clearly visible that most of the students score in between 60-80 in Maths whereas in reading and writing most of them score from 50-80

4.3 Multivariate analysis using pieplot

In [31]:
```
plt.rcParams['figure.figsize'] = (30, 12)

plt.subplot(1, 5, 1)
size = df['gender'].value_counts()
labels = 'Female', 'Male'
color = ['red','green']


plt.pie(size, colors = color, labels = labels,autopct = '.%2f%%')
plt.title('Gender', fontsize = 20)
plt.axis('off')



plt.subplot(1, 5, 2)
size = df['race_ethnicity'].value_counts()
labels = 'Group C', 'Group D','Group B','Group E','Group A'
color = ['red', 'green', 'blue', 'cyan','orange']

plt.pie(size, colors = color,labels = labels,autopct = '.%2f%%')
plt.title('Race_Ethnicity', fontsize = 20)
plt.axis('off')



plt.subplot(1, 5, 3)
size = df['lunch'].value_counts()
labels = 'Standard', 'Free'
color = ['red','green']
```

```
plt.pie(size, colors = color,labels = labels,autopct = '.%2f%%')
plt.title('Lunch', fontsize = 20)
plt.axis('off')


plt.subplot(1, 5, 4)
size = df['test_preparation_course'].value_counts()
labels = 'None', 'Completed'
color = ['red','green']

plt.pie(size, colors = color,labels = labels,autopct = '.%2f%%')
plt.title('Test Course', fontsize = 20)
plt.axis('off')


plt.subplot(1, 5, 5)
size = df['parental_level_of_education'].value_counts()
labels = 'Some College', "Associate's Degree",'High School','Some High School',"Bac
color = ['red', 'green', 'blue', 'cyan','orange','grey']

plt.pie(size, colors = color,labels = labels,autopct = '.%2f%%')
plt.title('Parental Education', fontsize = 20)
plt.axis('off')


plt.tight_layout()
plt.grid()

plt.show()
```



Insights

- Number of Male and Female students is almost equal
- Number students are greatest in Group C
- Number of students who have standard lunch are greater
- Number of students who have not enrolled in any test preparation course is greater
- Number of students whose parental education is "Some College" is greater followed closely by "Associate's Degree"

4.4 Feature Wise Visualization

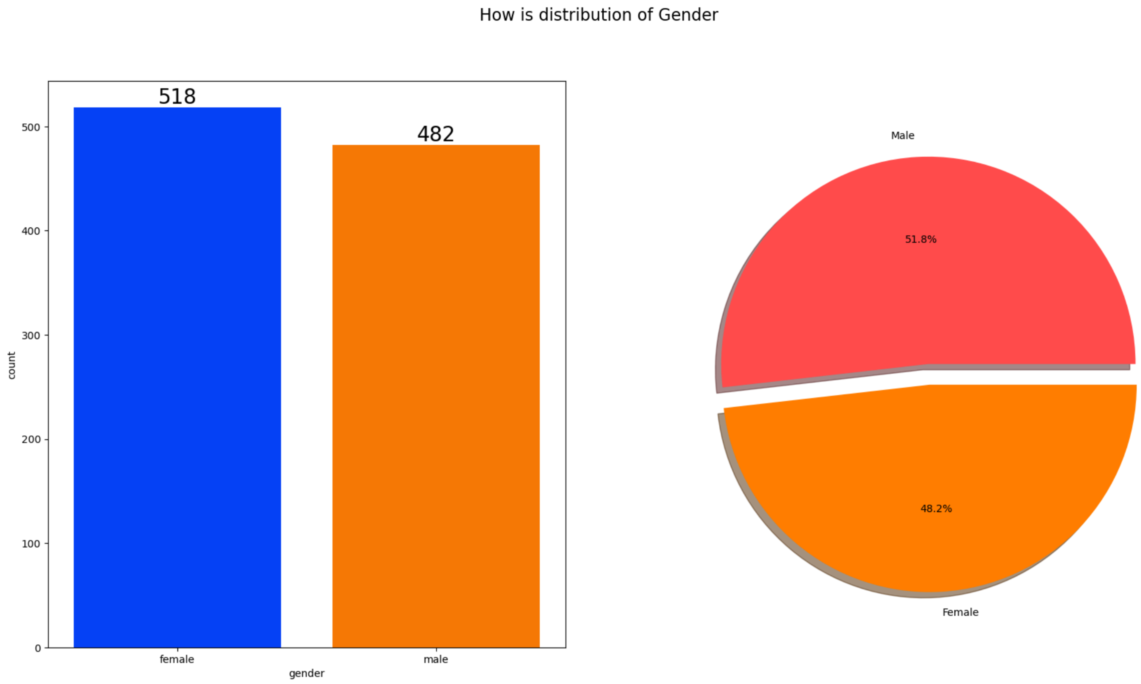
4.4.1 GENDER COLUMN

- How is distribution of Gender ?
- Is gender has any impact on student's performance ?

UNIVARIATE ANALYSIS (How is distribution of Gender ?)

In [32]:
```
f,ax=plt.subplots(1,2,figsize=(20,10))
sns.countplot(x=df['gender'],data=df,palette ='bright',ax=ax[0],saturation=0.95)
for container in ax[0].containers:
```

```
    ax[0].bar_label(container,color='black',size=20)
plt.suptitle('How is distribution of Gender', fontsize=16)

plt.pie(x=df['gender'].value_counts(),labels=['Male','Female'],explode=[0,0.1],autc
plt.show()
```



Insights

- Gender has balanced data with female students are 518 (48%) and male students are 482 (52%)

BIVARIATE ANALYSIS (Is gender has any impact on student's performance ?)

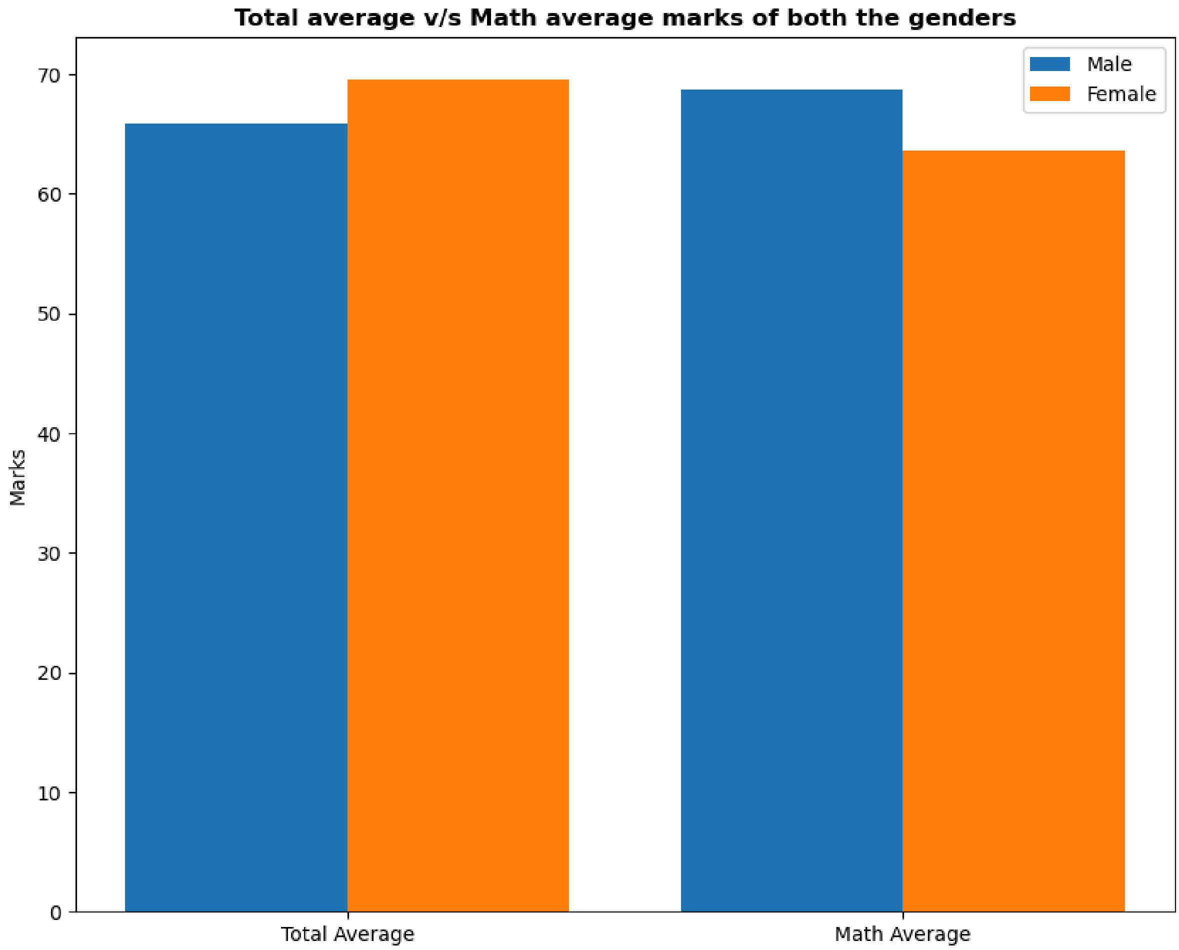
In [33]:
```
gender_group = df.groupby('gender').mean()
gender_group
```

Out[33]:

gender	math_score	reading_score	writing_score	total score	average
female	63.633205	72.608108	72.467181	208.708494	69.569498
male	68.728216	65.473029	63.311203	197.512448	65.837483

In [34]:
```
plt.figure(figsize=(10, 8))

X = ['Total Average','Math Average']


female_scores = [gender_group['average'][0], gender_group['math_score'][0]]
male_scores = [gender_group['average'][1], gender_group['math_score'][1]]

X_axis = np.arange(len(X))

plt.bar(X_axis - 0.2, male_scores, 0.4, label = 'Male')
plt.bar(X_axis + 0.2, female_scores, 0.4, label = 'Female')

plt.xticks(X_axis, X)
plt.ylabel("Marks")
```

```
plt.title("Total average v/s Math average marks of both the genders", fontweight='b
plt.legend()
plt.show()
```



Insights

- On an average females have a better overall score than men.
- whereas males have scored higher in Maths.

4.4.2 RACE/EHNICITY COLUMN

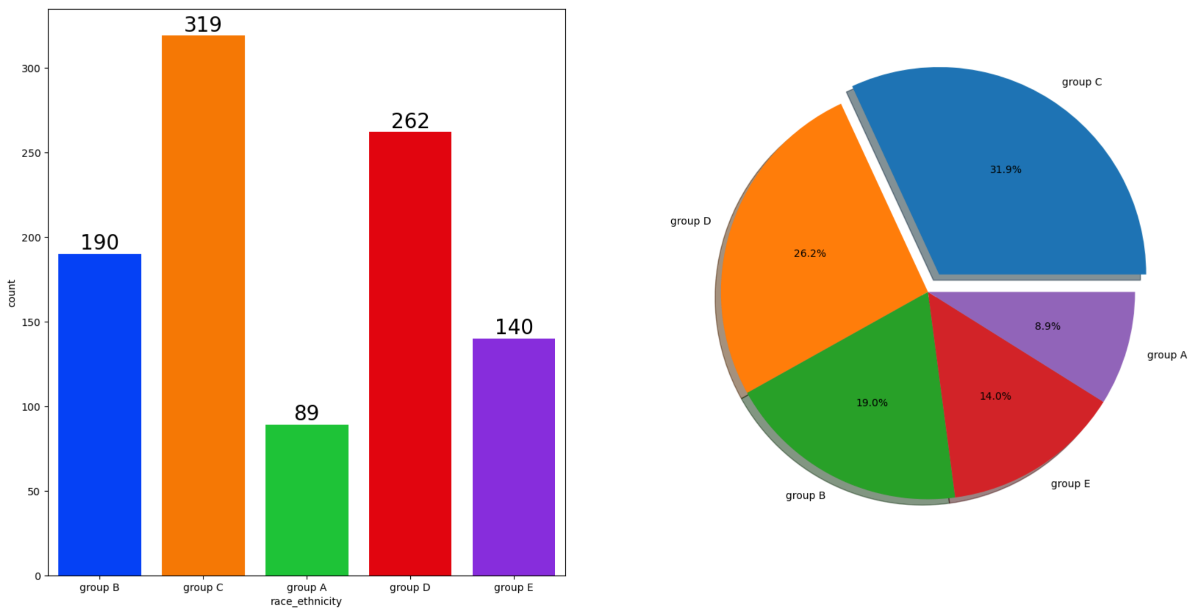
- How is Group wise distribution ?
- Is Race/Ehnicity has any impact on student's performance ?

UNIVARIATE ANALYSIS (How is Group wise distribution ?)

In [36]:

```
f,ax=plt.subplots(1,2,figsize=(20,10))
sns.countplot(x=df['race_ethnicity'],data=df,palette = 'bright',ax=ax[0],saturatior
for container in ax[0].containers:
    ax[0].bar_label(container,color='black',size=20)
plt.suptitle('How is Group wise distribution', fontsize=16)
plt.pie(x = df['race_ethnicity'].value_counts(),labels=df['race_ethnicity'].value_c
plt.show()
```

How is Group wise distribution



Insights

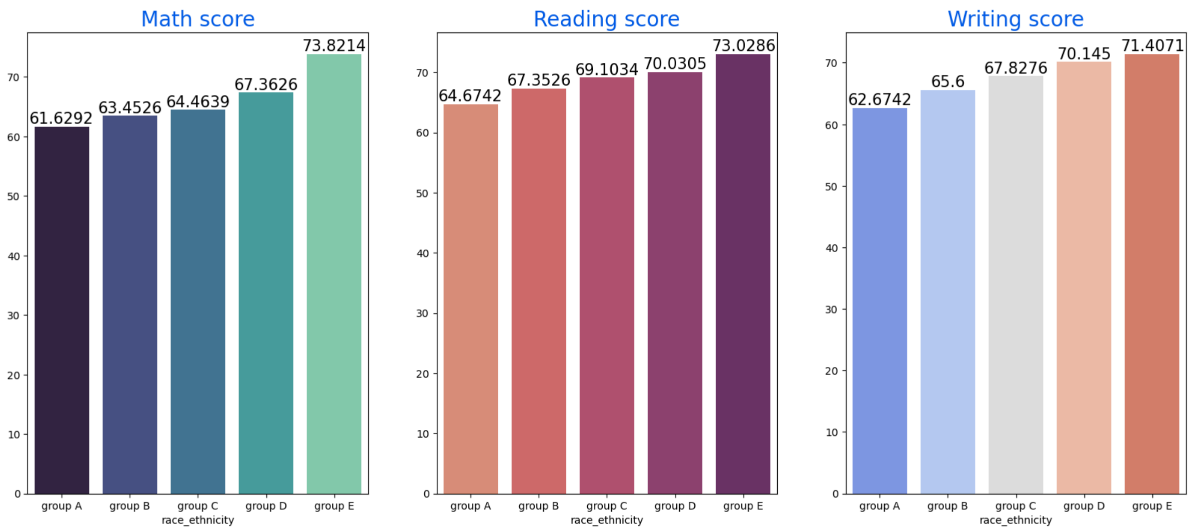
- Most of the student belonging from group C /group D.
- Lowest number of students belong to groupA.

BIVARIATE ANALYSIS (Is Race/Ehnicity has any impact on student's performance ?)

In [37]:

```
Group_data2=df.groupby('race_ethnicity')
f,ax=plt.subplots(1,3,figsize=(20,8))
sns.barplot(x=Group_data2['math_score'].mean().index,y=Group_data2['math_score'].me
ax[0].set_title('Math score',color='#005ce6',size=20)

for container in ax[0].containers:
    ax[0].bar_label(container,color='black',size=15)

sns.barplot(x=Group_data2['reading_score'].mean().index,y=Group_data2['reading_scor
ax[1].set_title('Reading score',color='#005ce6',size=20)

for container in ax[1].containers:
    ax[1].bar_label(container,color='black',size=15)

sns.barplot(x=Group_data2['writing_score'].mean().index,y=Group_data2['writing_scor
ax[2].set_title('Writing score',color='#005ce6',size=20)

plt.suptitle("Is Race/Ehnicity has any impact on student's performance", fontsize=1

for container in ax[2].containers:
    ax[2].bar_label(container,color='black',size=15)
```

Is Race/Ehnicity has any impact on student's performance



Insights

- Group E students have scored the highest marks.
- Group A students have scored the lowest marks.
- Students from a lower Socioeconomic status have a lower avg in all course subjects

4.4.3 PARENTAL LEVEL OF EDUCATION COLUMN

- What is educational background of student's parent ?
- Is parental education has any impact on student's performance ?

UNIVARIATE ANALYSIS (What is educational background of student's parent ?)

In [38]:
```
df.columns
```

Out[38]:
```
Index(['gender', 'race_ethnicity', 'parental_level_of_education', 'lunch',
       'test_preparation_course', 'math_score', 'reading_score',
       'writing_score', 'total score', 'average'],
      dtype='object')
```

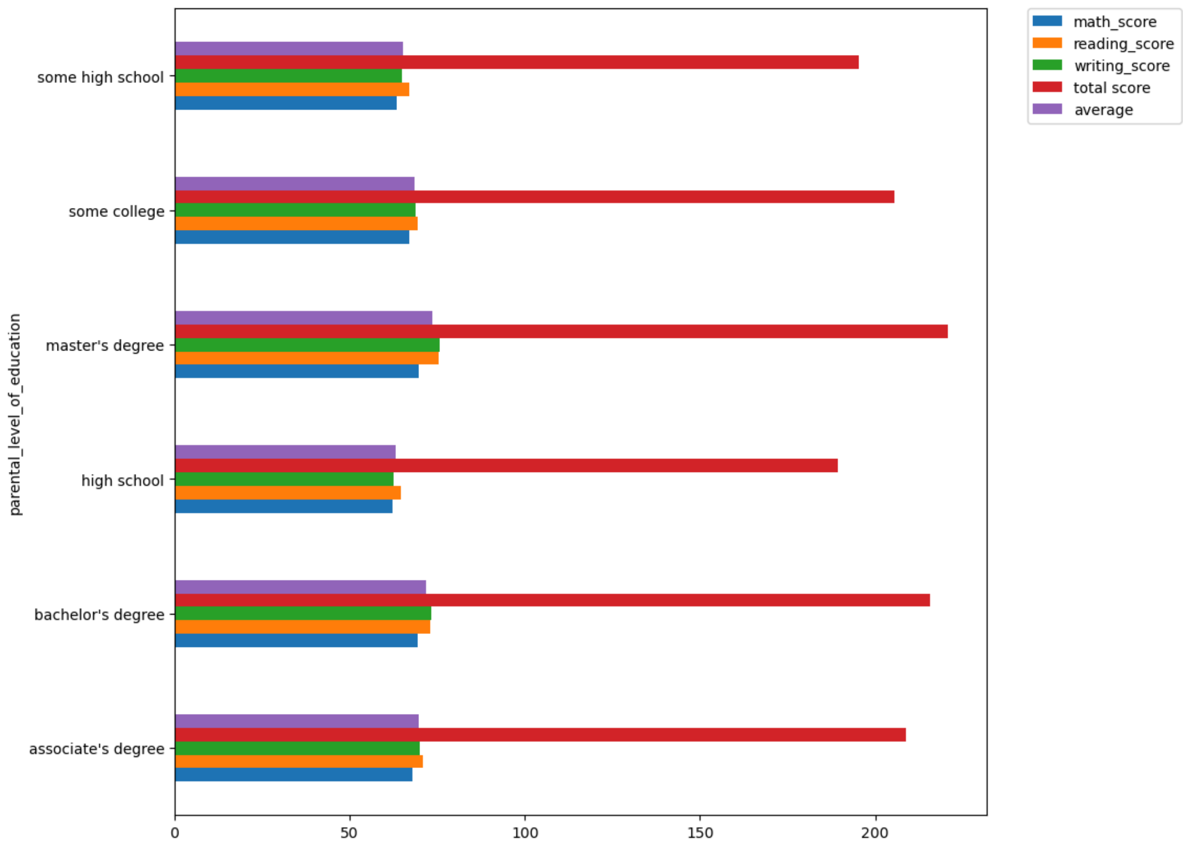
Insights

- Largest number of parents are from some college.

BIVARIATE ANALYSIS (Is parental education has any impact on student's performance ?)

In [39]:
```
df.groupby('parental_level_of_education').agg('mean').plot(kind='barh',figsize=(10,
plt.suptitle("Is parental education has any impact on student's performance", fonts
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.show()
```

Is parental education has any impact on student's performance



Insights

- The score of student whose parents possess master and bachelor level education are higher than others.

4.4.4 LUNCH COLUMN

- Which type of lunch is most common amoung students ?
- What is the effect of lunch type on test results?

UNIVARIATE ANALYSIS (Which type of lunch is most common amoung students ?)

Insights

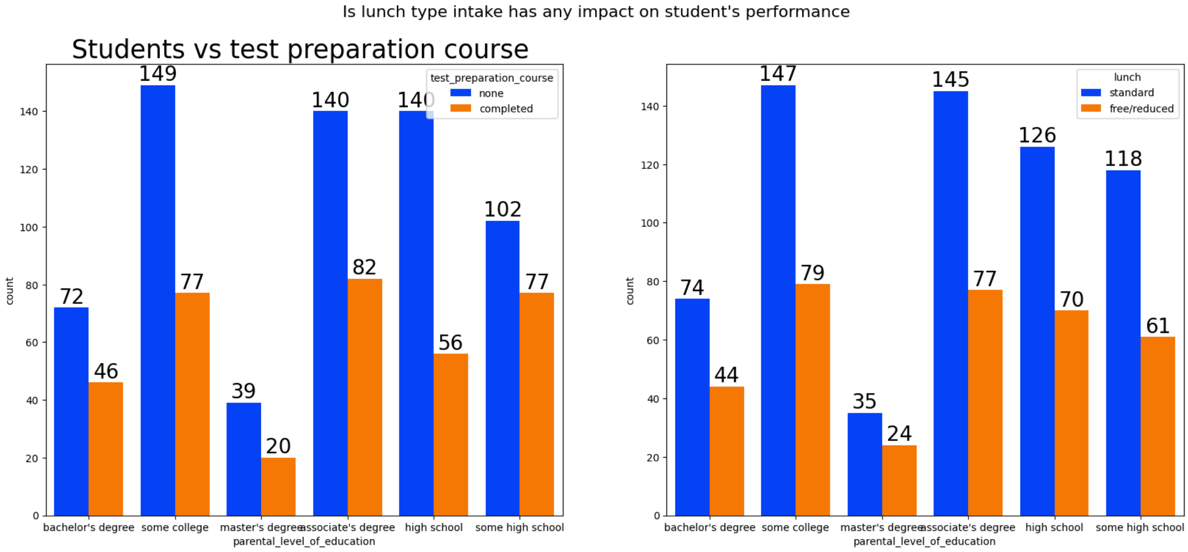
- Students being served Standard lunch was more than free lunch

BIVARIATE ANALYSIS (Is lunch type intake has any impact on student's performance ?)

In [37]:
```
f,ax=plt.subplots(1,2,figsize=(20,8))
sns.countplot(x=df['parental_level_of_education'],data=df,palette = 'bright',hue='t
ax[0].set_title('Students vs test preparation course ',color='black',size=25)
for container in ax[0].containers:
    ax[0].bar_label(container,color='black',size=20)

sns.countplot(x=df['parental_level_of_education'],data=df,palette = 'bright',hue='l
for container in ax[1].containers:
```

```
    ax[1].bar_label(container,color='black',size=20)
plt.suptitle("Is lunch type intake has any impact on student's performance", fontsi
```

Out[37]: Text(0.5, 0.98, "Is lunch type intake has any impact on student's performance")



Insights

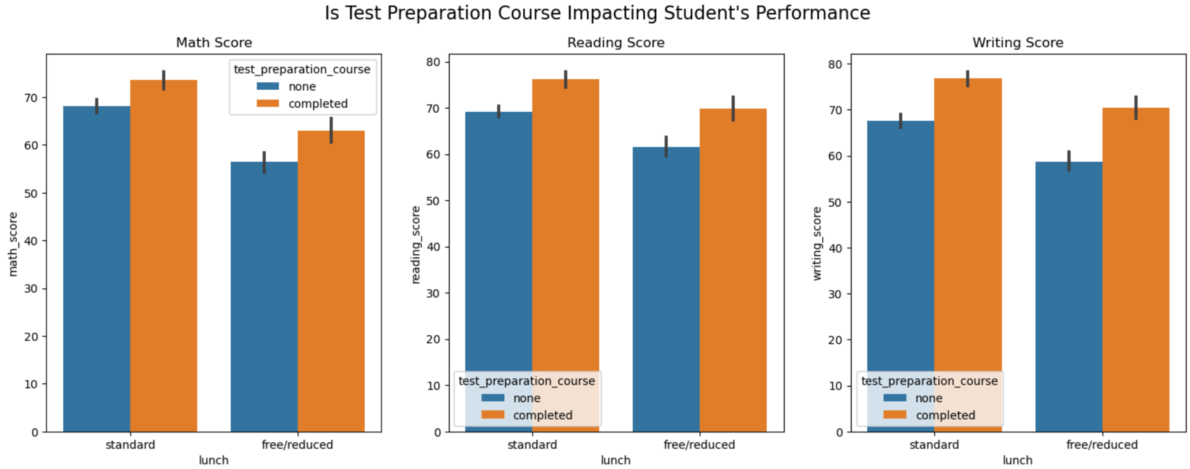
- Students who get Standard Lunch tend to perform better than students who got free/reduced lunch

4.4.5 TEST PREPARATION COURSE COLUMN

- Which type of lunch is most common amoung students ?
- Is Test prepration course has any impact on student's performance ?

BIVARIATE ANALYSIS (Is Test prepration course has any impact on student's performance ?)

In [40]:
```
plt.figure(figsize=(18, 6))  # Adjust the figure size as needed

# Subplot 1
plt.subplot(1, 3, 1)
sns.barplot(x=df['lunch'], y=df['math_score'], hue=df['test_preparation_course'])
plt.title('Math Score')

# Subplot 2
plt.subplot(1, 3, 2)
sns.barplot(x=df['lunch'], y=df['reading_score'], hue=df['test_preparation_course']
plt.title('Reading Score')

# Subplot 3
plt.subplot(1, 3, 3)
sns.barplot(x=df['lunch'], y=df['writing_score'], hue=df['test_preparation_course']
plt.title('Writing Score')

plt.suptitle("Is Test Preparation Course Impacting Student's Performance", fontsize
plt.show()
```

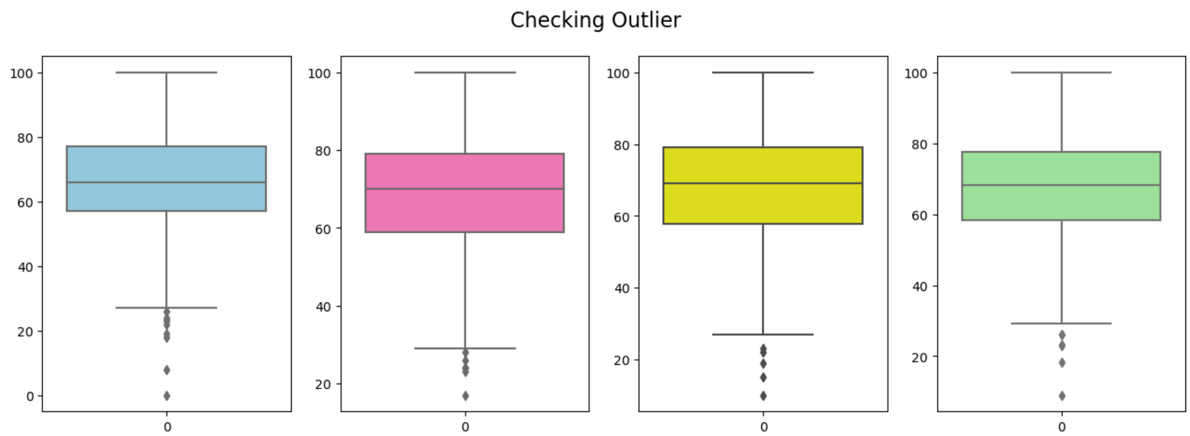


Insights

- Students who have completed the Test Prepration Course have scores higher in all three categories than those who haven't taken the course

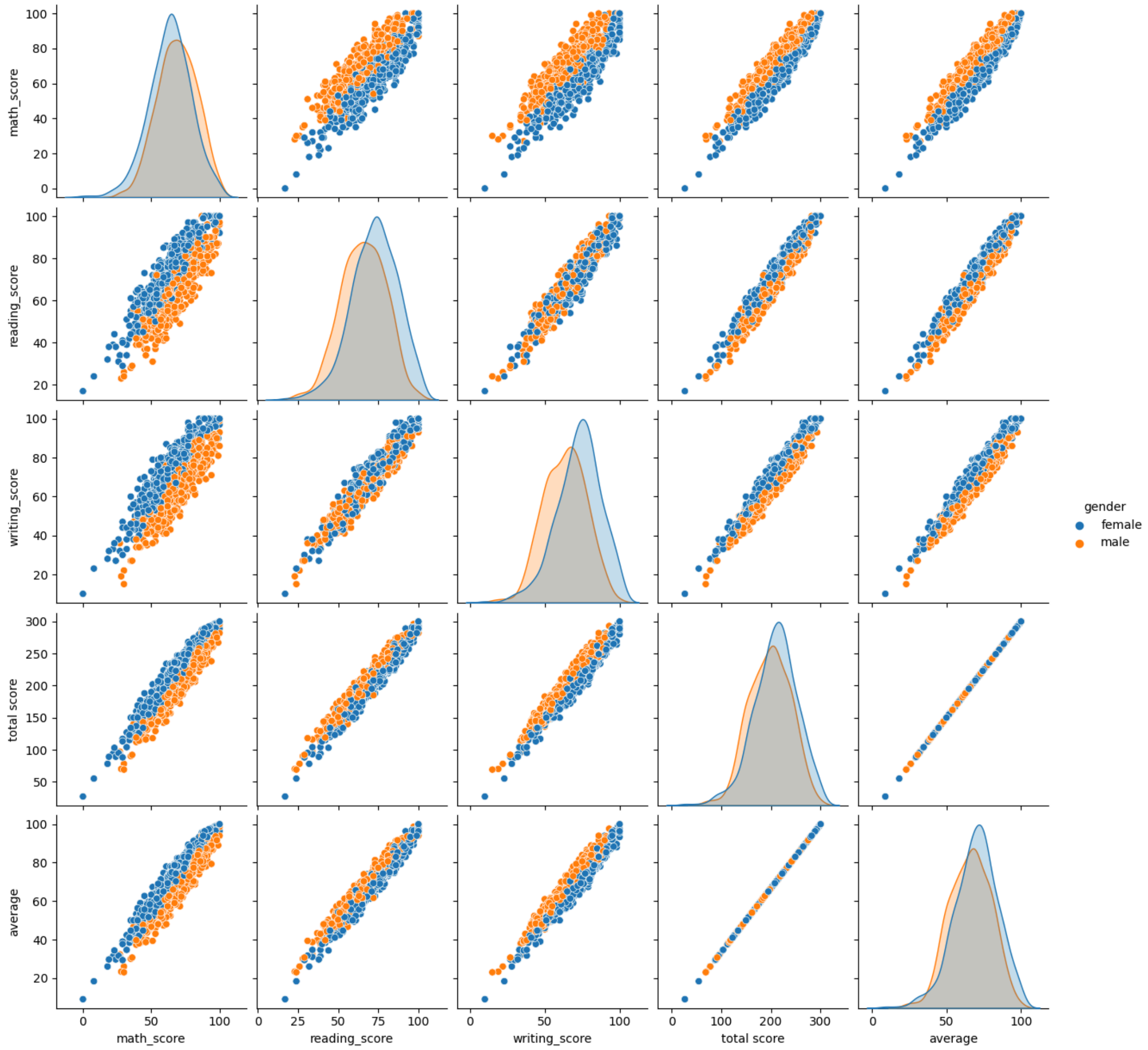
4.4.6 CHECKING OUTLIERS

In [41]:
```
plt.subplots(1,4,figsize=(16,5))
plt.subplot(141)
sns.boxplot(df['math_score'],color='skyblue')
plt.subplot(142)
sns.boxplot(df['reading_score'],color='hotpink')
plt.subplot(143)
sns.boxplot(df['writing_score'],color='yellow')
plt.subplot(144)
sns.boxplot(df['average'],color='lightgreen')

plt.suptitle("Checking Outlier", fontsize=16)
plt.show()
```



4.4.7 MUTIVARIATE ANALYSIS USING PAIRPLOT

In [40]:
```
sns.pairplot(df,hue = 'gender')
plt.show()
```



Insights

- From the above plot it is clear that all the scores increase linearly with each other.

5. Conclusions

- Student's Performance is related with lunch, race, parental level education
- Females lead in pass percentage and also are top-scorers
- Student's Performance is not much related with test preparation course
- Finishing preparation course is benefitial.

Part 2: Model Training

In [42]:
```
df = pd.read_csv('stud.csv')
```

In [43]:
```
X = df.drop(columns=['math_score'], axis=1)
```

In [44]:
```
X
```

Out[44]:

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course	reac
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 × 7 columns

In [45]:

```
df.head()
```

Out[45]:

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course	math_s
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	

In [46]:

```
df.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 [47]:

```
y = df['math_score']
```

In [48]:

```
categorical_features = df.select_dtypes(include='object').columns
numerical_features =df.select_dtypes(exclude='object').columns
```

```
print(categorical_features)
print(numerical_features)
```

```
Index(['gender', 'race_ethnicity', 'parental_level_of_education', 'lunch',
       'test_preparation_course'],
      dtype='object')
Index(['math_score', 'reading_score', 'writing_score'], dtype='object')
```

In [49]:
```
num_featuers = X.select_dtypes(exclude='object').columns
cat_features  = X.select_dtypes(include='object').columns

from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer

num_transformer = StandardScaler()
oh_transformer = OneHotEncoder()

preprocessor = ColumnTransformer(
    [
        ("OneHotEncoder", oh_transformer, cat_features),
        ("StandardScaler", num_transformer, num_featuers)
    ]
)
X = preprocessor.fit_transform(X)
```

In [50]:
```
X.shape
```

Out[50]:
```
(1000, 19)
```

In [51]:
```
X
```

Out[51]:
```
array([[ 1.        ,  0.        ,  0.        , ...,  1.        ,
         0.19399858,  0.39149181],
       [ 1.        ,  0.        ,  0.        , ...,  0.        ,
         1.42747598,  1.31326868],
       [ 1.        ,  0.        ,  0.        , ...,  1.        ,
         1.77010859,  1.64247471],
       ...,
       [ 1.        ,  0.        ,  0.        , ...,  0.        ,
         0.12547206, -0.20107904],
       [ 1.        ,  0.        ,  0.        , ...,  0.        ,
         0.60515772,  0.58901542],
       [ 1.        ,  0.        ,  0.        , ...,  1.        ,
         1.15336989,  1.18158627]])
```

In [52]:
```
X.max(), X.min()
```

Out[52]:
```
(2.112741202570347, -3.82234534162361)
```

In [53]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state
X_train.shape,X_test.shape
```

Out[53]:
```
((800, 19), (200, 19))
```

Evaluation function

In [54]:
```
def evaluate_model(true, predicted):
  mae = mean_absolute_error(true, predicted)
  mse = mean_squared_error(true, predicted)
  rmse = np.sqrt(mean_squared_error(true, predicted))
```

```
    r2_square = r2_score(true, predicted)
    return mae, rmse, r2_square
```

In [60]:
```
models = {
    "Linear Regression": LinearRegression(),
    "Lasso": Lasso(),
    "Ridge": Ridge(),
    "K-Neighbors Regressor": KNeighborsRegressor(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest Regressor": RandomForestRegressor(),
    "XGBRegressor": XGBRegressor(),
    "CatBoosting Regressor": CatBoostRegressor(verbose=False),
    "AdaBoost Regressor": AdaBoostRegressor()
}

model_list = []
r2_list = []

for i in range(len(list(models))):
  model = list(models.values())[i]
  model.fit(X_train,y_train)

  #make prediction
  y_train_pred = model.predict(X_train)
  y_test_pred = model.predict(X_test)

  # Evaluate train and test dataset
  model_train_mae, model_train_rmse, model_train_r2 = evaluate_model(y_train, y_tra
  model_test_mae, model_test_rmse, model_test_r2 = evaluate_model(y_test, y_test_pr

  print(list(models.keys())[i])
  model_list.append(list(models.keys())[i])
  r2_list.append(round(model_test_r2,3))

  print("Model performance for Train set")
  print("- Root mean squared error : {:.4f}".format(model_train_rmse))
  print("- Mean absolute error : {:.4f}".format(model_train_mae))
  print("- R2 score : {:.4f}".format(model_train_r2))

  print("-"*35)

  print("Model performance for Test set")
  print("- Root Mean squared error: {:.4f}".format(model_test_rmse))
  print("- Mean absolute error: {:.4f}".format(model_test_mae))
  print("- R2 score : {:.4f}".format(model_test_r2))
  print('='*35)
  print("\n")
```

```
Linear Regression
Model performance for Train set
- Root mean squared error : 5.3240
- Mean absolute error : 4.2691
- R2 score : 0.8743
----------------------------------
Model performance for Test set
- Root Mean squared error: 5.3773
- Mean absolute error: 4.2053
- R2 score : 0.8812
===================================


Lasso
Model performance for Train set
- Root mean squared error : 6.5938
- Mean absolute error : 5.2063
- R2 score : 0.8071
----------------------------------
Model performance for Test set
- Root Mean squared error: 6.5197
- Mean absolute error: 5.1579
- R2 score : 0.8253
===================================


Ridge
Model performance for Train set
- Root mean squared error : 5.3233
- Mean absolute error : 4.2650
- R2 score : 0.8743
----------------------------------
Model performance for Test set
- Root Mean squared error: 5.3904
- Mean absolute error: 4.2111
- R2 score : 0.8806
===================================


K-Neighbors Regressor
Model performance for Train set
- Root mean squared error : 5.7133
- Mean absolute error : 4.5213
- R2 score : 0.8552
----------------------------------
Model performance for Test set
- Root Mean squared error: 7.2488
- Mean absolute error: 5.6310
- R2 score : 0.7841
===================================


Decision Tree
Model performance for Train set
- Root mean squared error : 0.2795
- Mean absolute error : 0.0187
- R2 score : 0.9997
----------------------------------
Model performance for Test set
- Root Mean squared error: 7.5776
- Mean absolute error: 5.9700
- R2 score : 0.7640
===================================
```

```
Random Forest Regressor
Model performance for Train set
- Root mean squared error : 2.3223
- Mean absolute error : 1.8461
- R2 score : 0.9761
----------------------------------
Model performance for Test set
- Root Mean squared error: 5.9204
- Mean absolute error: 4.5573
- R2 score : 0.8560
===================================


XGBRegressor
Model performance for Train set
- Root mean squared error : 1.0073
- Mean absolute error : 0.6875
- R2 score : 0.9955
----------------------------------
Model performance for Test set
- Root Mean squared error: 6.4733
- Mean absolute error: 5.0577
- R2 score : 0.8278
===================================


CatBoosting Regressor
Model performance for Train set
- Root mean squared error : 3.0427
- Mean absolute error : 2.4054
- R2 score : 0.9589
----------------------------------
Model performance for Test set
- Root Mean squared error: 6.0086
- Mean absolute error: 4.6125
- R2 score : 0.8516
===================================


AdaBoost Regressor
Model performance for Train set
- Root mean squared error : 5.9072
- Mean absolute error : 4.8231
- R2 score : 0.8452
----------------------------------
Model performance for Test set
- Root Mean squared error: 6.0436
- Mean absolute error: 4.7032
- R2 score : 0.8499
===================================
```

In [62]:
```
print(model_list, r2_list)
```

```
['Linear Regression', 'Lasso', 'Ridge', 'K-Neighbors Regressor', 'Decision Tree',
'Random Forest Regressor', 'XGBRegressor', 'CatBoosting Regressor', 'AdaBoost Regr
essor'] [0.88, 0.825, 0.881, 0.783, 0.72, 0.85, 0.828, 0.852, 0.856]
```

In [63]:
```
results = pd.DataFrame(list(zip(model_list, r2_list)), columns=["Model Name", 'R2_S
```

In [64]:
```
results
```

Out[64]:

	Model Name	R2_Score
2	Ridge	0.881
0	Linear Regression	0.880
8	AdaBoost Regressor	0.856
7	CatBoosting Regressor	0.852
5	Random Forest Regressor	0.850
6	XGBRegressor	0.828
1	Lasso	0.825
3	K-Neighbors Regressor	0.783
4	Decision Tree	0.720

Part 3: Final Model (Linear Regression)

In [62]:

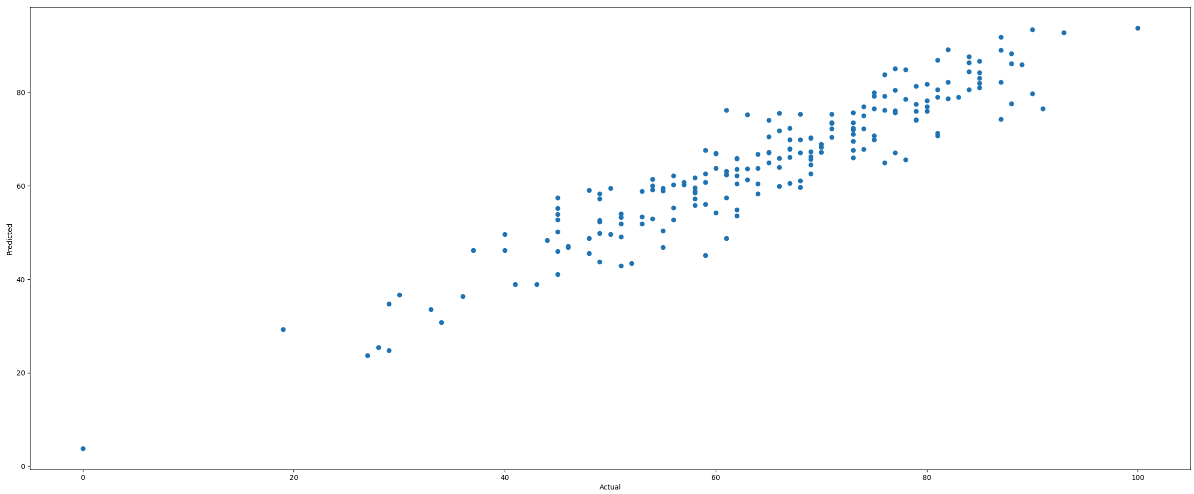
```
lin_model = LinearRegression(fit_intercept=True)
lin_model = lin_model.fit(X_train, y_train)
y_pred = lin_model.predict(X_test)
score = r2_score(y_test, y_pred)*100
print(" Accuracy of the model is %.2f" %score)
```

```
 Accuracy of the model is 88.12
```

Plot predicted and actual value

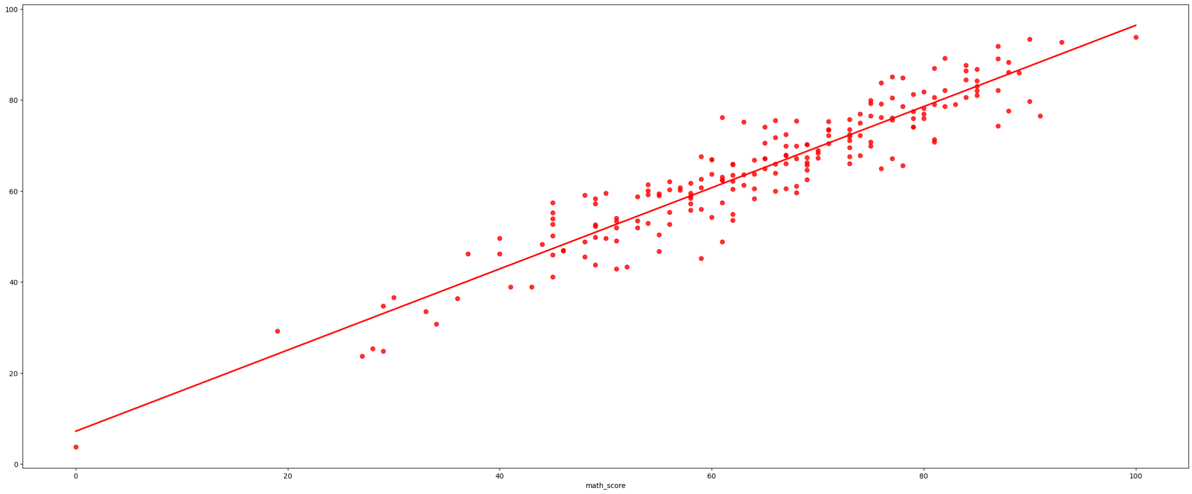
In [63]:

```
plt.scatter(y_test,y_pred);
plt.xlabel('Actual');
plt.ylabel('Predicted');
```



In [64]:

```
sns.regplot(x=y_test,y=y_pred,ci=None,color ='red');
```



Difference Between Actual and Predicted value

In [65]:
```
pred_df=pd.DataFrame({'Actual Value':y_test,'Predicted Value':y_pred,'Difference':y
pred_df
```

Out[65]:

	Actual Value	Predicted Value	Difference
521	91	76.507812	14.492188
737	53	58.796875	-5.796875
740	80	76.976562	3.023438
660	74	76.984375	-2.984375
411	84	87.664062	-3.664062
...	...	...	...
408	52	43.367188	8.632812
332	62	62.156250	-0.156250
208	74	67.812500	6.187500
613	65	67.125000	-2.125000
78	61	62.343750	-1.343750

200 rows × 3 columns

In [ ]: