In [ ]:

```
#If you don't have all of this installed, then you need to do the followin
#Go to the command prompt (cmd) and type: pip install <name>
```

In [1]:

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

In [ ]:

```
#We have run this and checked it. There are no errors in it. Everything is
```

In [2]:

```
#Now here we will write to call the dataset

df = pd.read_csv("Student_score.csv")    #CSV because our file is a CSV fi
```

In [3]:

```
# Now this thing has also run

#So now we print the values starting from 3.

print(df.head())     #Called the dataset head so that all columns come to
```

```
   Unnamed: 0  Gender EthnicGroup          ParentEduc     LunchType TestPrep
\
0           0  female         NaN   bachelor's degree      standard     none
1           1  female     group C        some college      standard      NaN
2           2  female     group B     master's degree      standard     none
3           3    male     group A  associate's degree  free/reduced     none
4           4    male     group C        some college      standard     none

  ParentMaritalStatus PracticeSport IsFirstChild  NrSiblings TransportMeans
\
0             married     regularly          yes         3.0     school_bus
1             married     sometimes          yes         0.0            NaN
2              single     sometimes          yes         4.0     school_bus
3             married         never           no         1.0            NaN
4             married     sometimes          yes         0.0     school_bus

  WklyStudyHours  MathScore  ReadingScore  WritingScore
0            < 5         71            71            74
1         5 - 10         69            90            88
2            < 5         87            93            91
3         5 - 10         45            56            42
4         5 - 10         76            78            75
```

In [4]:

```
# Now let's describe the data.

df.describe()  # As soon as we made the call, all the columns in our data
```

Out[4]:

	Unnamed: 0	NrSiblings	MathScore	ReadingScore	WritingScore

count	30641.000000	29069.000000	30641.000000	30641.000000	30641.000000
mean	499.556607	2.145894	66.558402	69.377533	68.418622
std	288.747894	1.458242	15.361616	14.758952	15.443525
min	0.000000	0.000000	0.000000	10.000000	4.000000
25%	249.000000	1.000000	56.000000	59.000000	58.000000
50%	500.000000	2.000000	67.000000	70.000000	69.000000
75%	750.000000	3.000000	78.000000	80.000000	79.000000
max	999.000000	7.000000	100.000000	100.000000	100.000000

In [5]:

```
df.info() #will tell you what data types are in the columns .........you c
          #it will tell you the count of null values.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30641 entries, 0 to 30640
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Unnamed: 0           30641 non-null  int64  
 1   Gender               30641 non-null  object 
 2   EthnicGroup          28801 non-null  object 
 3   ParentEduc           28796 non-null  object 
 4   LunchType            30641 non-null  object 
 5   TestPrep             28811 non-null  object 
 6   ParentMaritalStatus  29451 non-null  object 
 7   PracticeSport        30010 non-null  object 
 8   IsFirstChild         29737 non-null  object 
 9   NrSiblings           29069 non-null  float64
 10  TransportMeans       27507 non-null  object 
 11  WklyStudyHours       29686 non-null  object 
 12  MathScore            30641 non-null  int64  
 13  ReadingScore         30641 non-null  int64  
 14  WritingScore         30641 non-null  int64  
dtypes: float64(1), int64(4), object(10)
memory usage: 3.5+ MB
```

In [6]:

```
df.isnull() #If we run this, we will get to know the count of null values.
```

Out[6]:

	Unnamed: 0	Gender	EthnicGroup	ParentEduc	LunchType	TestPrep	Parent
0	False	False	True	False	False	False	
1	False	False	False	False	False	True	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
...	...	...	...	...	...	...	
30636	False	False	False	False	False	False	
30637	False	False	False	False	False	False	

30638	False	False	True	False	False	False
30639	False	False	False	False	False	False
30640	False	False	False	False	False	False

30641 rows × 15 columns

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
Unnamed: 0                0
Gender                    0
EthnicGroup            1840
ParentEduc             1845
LunchType                 0
TestPrep               1830
ParentMaritalStatus    1190
PracticeSport           631
IsFirstChild            904
NrSiblings             1572
TransportMeans         3134
WklyStudyHours          955
MathScore                 0
ReadingScore              0
WritingScore              0
dtype: int64
```

Drop unnamed column" means to delete the column.

In [10]:
```
df = df.drop("Unnamed: 0", axis = 1)
print(df.head())
```

```
   Gender EthnicGroup          ParentEduc     LunchType TestPrep  \
0  female         NaN   bachelor's degree      standard     none
1  female     group C        some college      standard      NaN
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

  ParentMaritalStatus PracticeSport IsFirstChild  NrSiblings TransportMeans
\
0             married     regularly          yes         3.0     school_bus
1             married     sometimes          yes         0.0            NaN
2              single     sometimes          yes         4.0     school_bus
3             married         never           no         1.0            NaN
4             married     sometimes          yes         0.0     school_bus

  WklyStudyHours  MathScore  ReadingScore  WritingScore
0            < 5         71            71            74
1         5 - 10         69            90            88
2            < 5         87            93            91
3         5 - 10         45            56            42
4         5 - 10         76            78            75
```

In [ ]:
```
# Removing unnamed successfully.
```

Change weakly studyhours columns

In [12]:
```
df["WklyStudyHours"]= df["WklyStudyHours"].str.replace("05-Oct","5 - 10")

#str.replace() gives us the data after replacing the values.

df.head() #We made a call after making the changes.
```

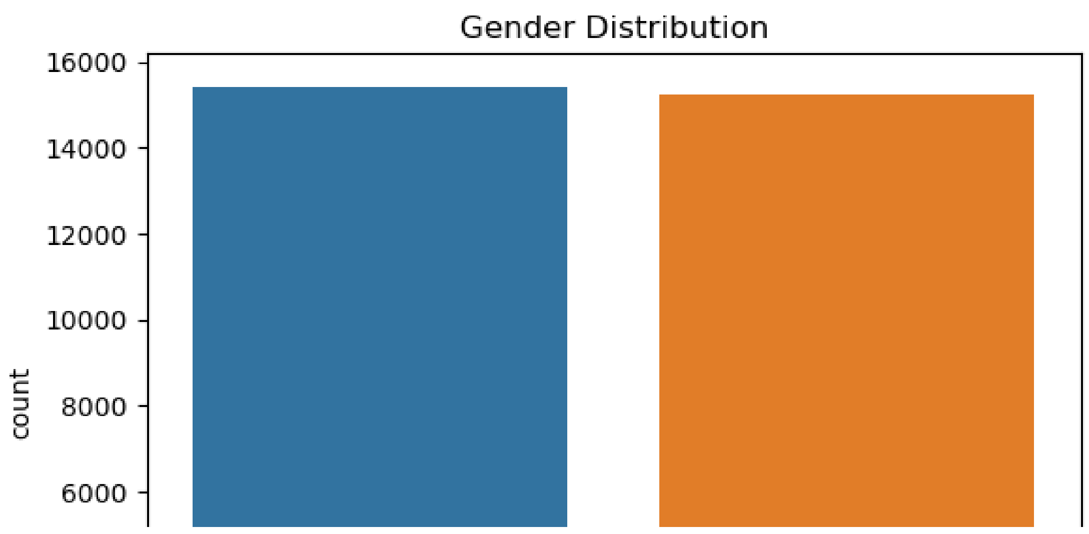
Out[12]:

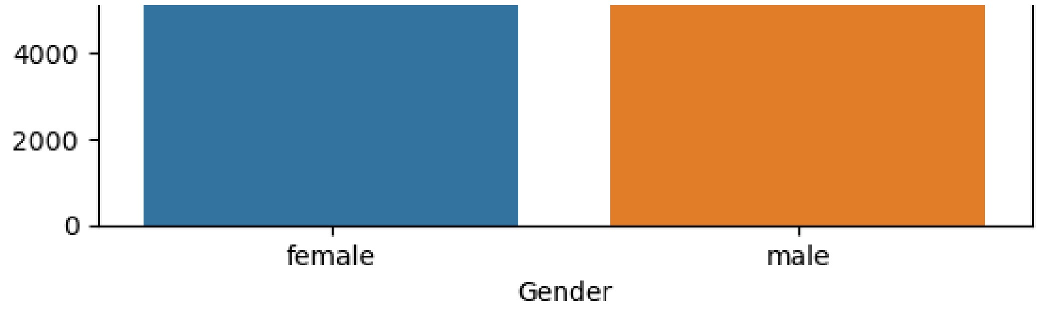
	Gender	EthnicGroup	ParentEduc	LunchType	TestPrep	ParentMaritalStatus
0	female	NaN	bachelor's degree	standard	none	married
1	female	group C	some college	standard	NaN	married
2	female	group B	master's degree	standard	none	single
3	male	group A	associate's degree	free/reduced	none	married
4	male	group C	some college	standard	none	married

In [ ]:
```
# Successfully changed.
```

Gender Distribution

In [38]:
```
sns.countplot(data=df,x="Gender") #countplot automatically counts the valu

plt.title("Gender Distribution")  #This will add a title to the chart, tel

plt.show()   #As we ran it, we got the values.
```



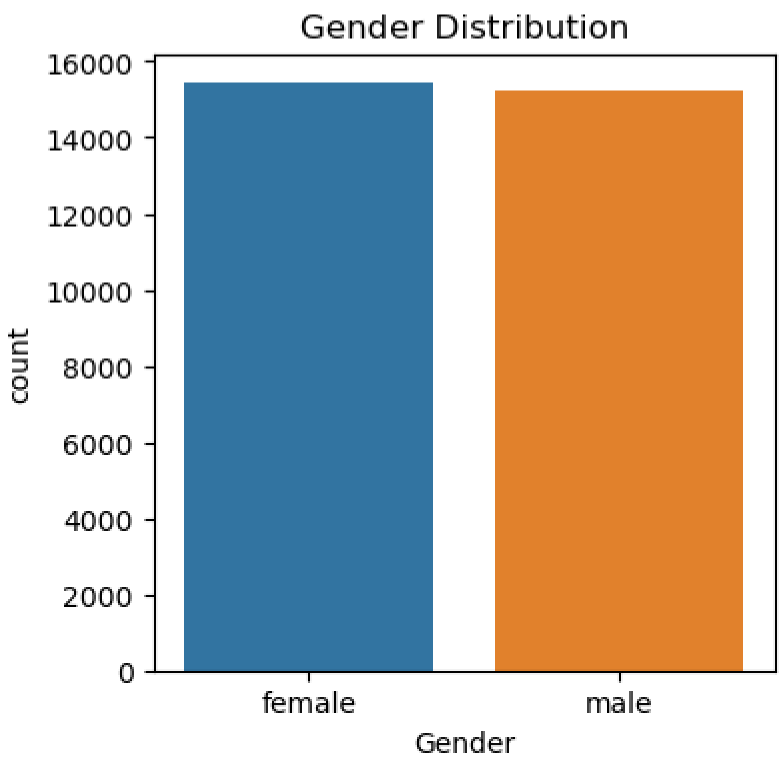


In [ ]:

```
#Now we can also define the figure size in this.
```

In [39]:

```
plt.figure(figsize= (4,4))   #We have set the height and width from this.

plt.title("Gender Distribution")  #From this, a title will come above the


sns.countplot(data=df,x="Gender") #countplot automatically counts the valu
plt.show()   #As we ran it, the values came out.
```

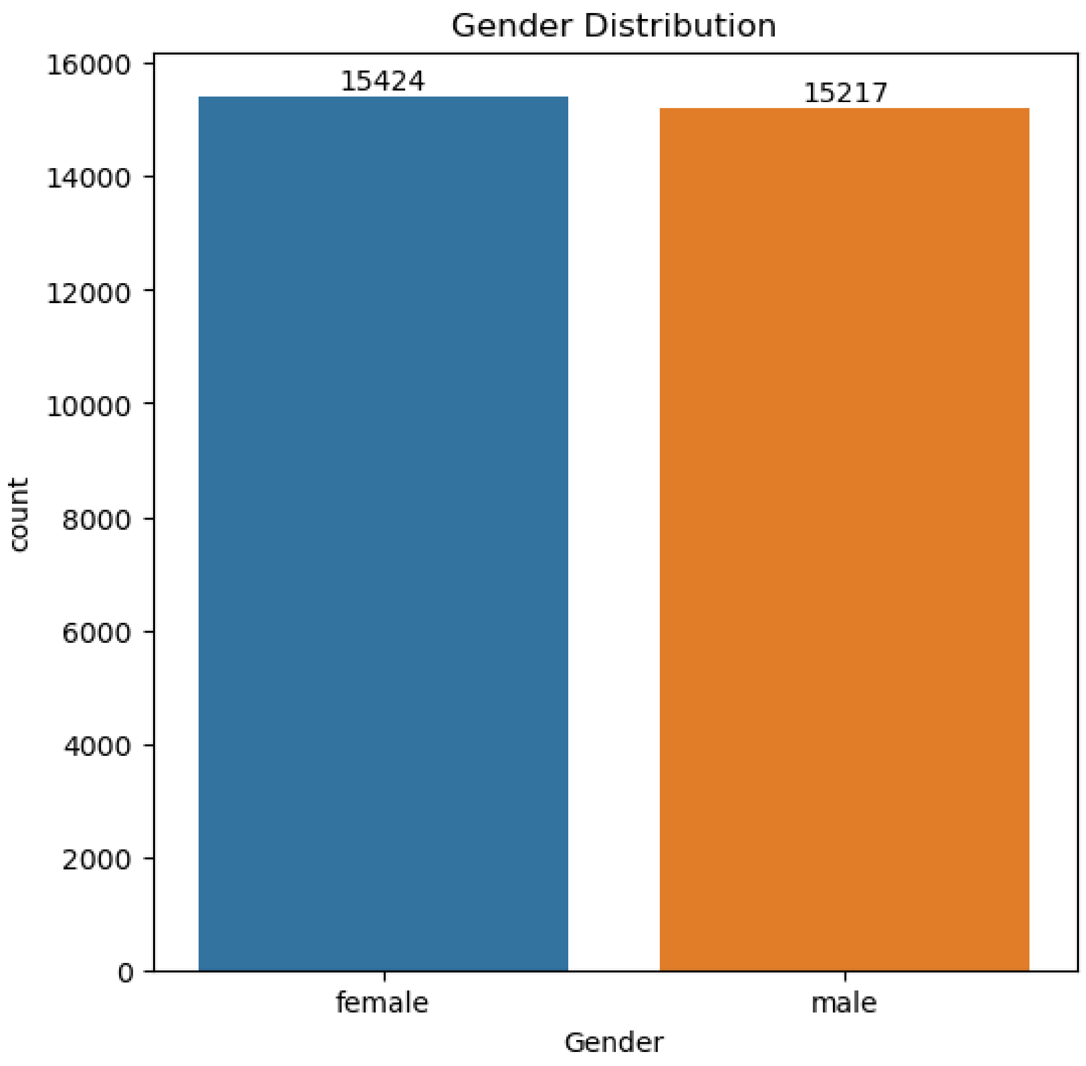


In [ ]:

```
# Now let's see if the distance between our values is very less.....so if 
# which will show us the exact count of the values.
```

In [40]:

```
plt.figure(figsize= (6,6))  # We have set the height and width using this.

plt.title("Gender Distribution")  # This will add a title to the top of th



ax=sns.countplot(data=df,x="Gender") # countplot automatically counts the 
# We have stored the values inside the variable 'ax'.
# Now we will call 'ax'.
ax.bar_label(ax.containers[0])


plt.show()  # As we ran it, we got the values.
```



In [ ]:

```
#From the above chart, we have analyzed that the data shows a higher numbe
```

In [ ]:

```
#Now we will create a chart on the impact of parents' education on student
```

In [28]:

```
gb=df.groupby("ParentEduc").agg({"MathScore" :'mean',"ReadingScore" : 'mea
#gb >> group by ...

#.agg({}) >> We will use this to calculate the mean value of students' mat
#We will pass these three things in the format of a dictionary.

#Now we will print all the values.

print(gb)
```

```
                   MathScore  ReadingScore  WritingScore
ParentEduc
associate's degree  68.365586     71.124324     70.299099
bachelor's degree   70.466627     73.062020     73.331069
high school         64.435731     67.213997     65.421136
master's degree     72.336134     75.832921     76.356896
some college        66.390472     69.179708     68.501432
some high school    62.584013     65.510785     63.632409
```

In [ ]:

```
#All our values have been printed.
```

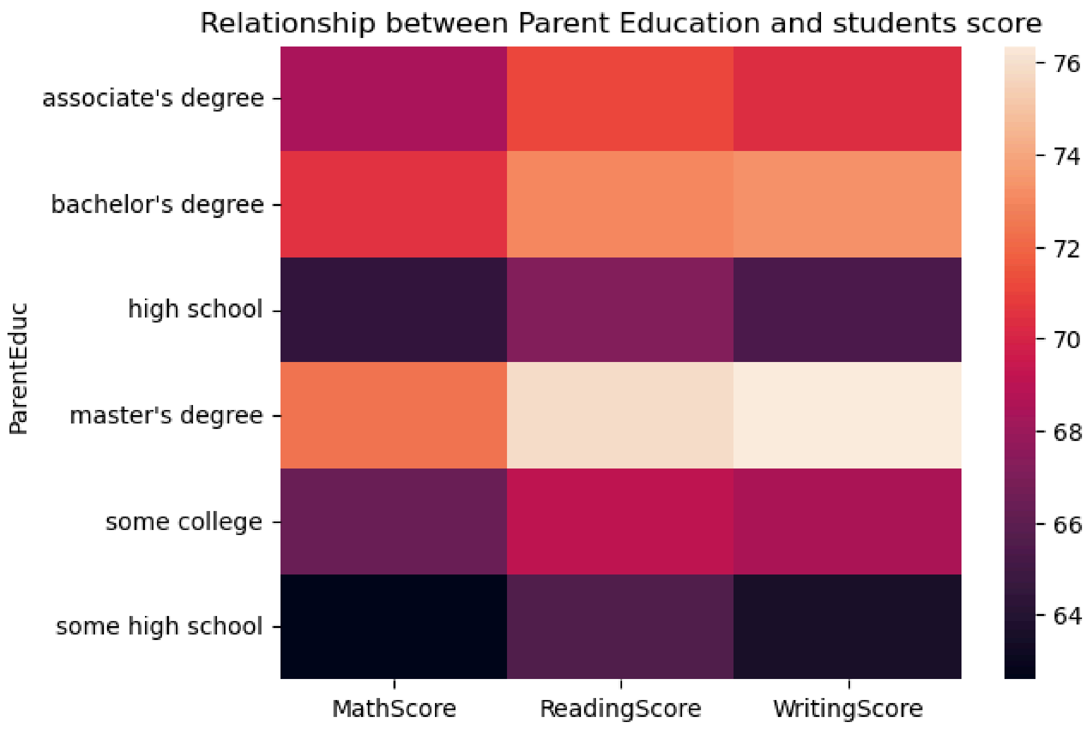
In [ ]:

```
#Now looking at the data, it's clear that students whose parents have bach

#Now we'll plot this effectively.
```

Heatmap

In [53]:

```
sns.heatmap(gb)   #We have a heat map within Seaborn >>> where we will pas

plt.title("Relationship between Parent Education and students score")
#This will be the title of the chart, which will tell us what the chart is

plt.show()
```



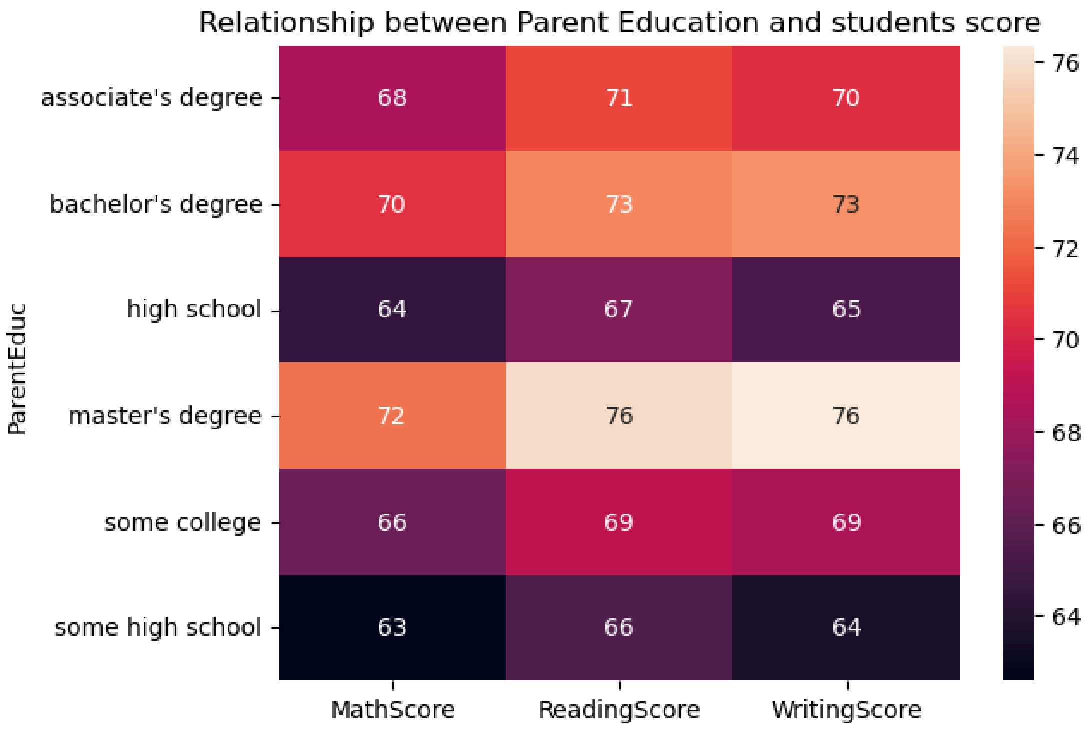
In [ ]:

```
#Now from the above map we are understanding that for lower values there i
#A color scale is given on the side so we are understanding this thing by
```

In [52]:

```
#Now only colors are showing, we also want to show values, so we will...
```

```
sns.heatmap(gb,annot=True)   #In Seaborn, we have a heatmap >>> in which w

plt.title("Relationship between Parent Education and students score")
#This will add a title to the chart, which will tell us what the chart is

plt.show()



#As soon as we run this, the values started showing to us.
```

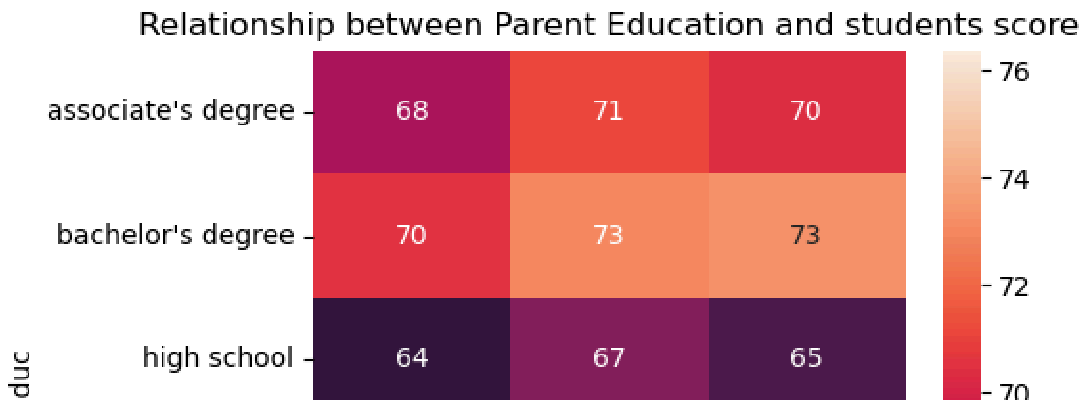


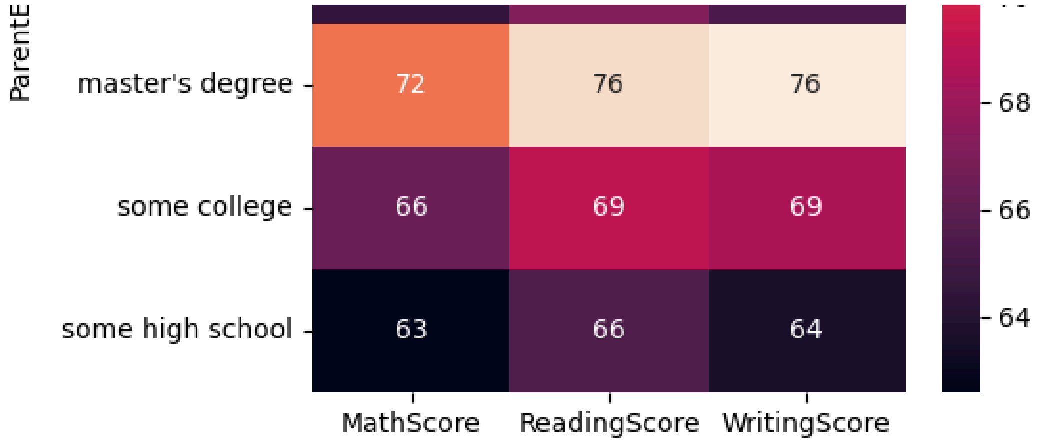
In [ ]:

```
#Now, if we want, we can also change the size of this in the same method a
```

In [50]:

```
plt.figure(figsize=(5,5))

plt.title("Relationship between Parent Education and students score")
#This will add a title to the top of the chart, indicating what the chart


sns.heatmap(gb,annot=True)   #We have a heatmap within Seaborn >>> we will
plt.show()
```





In [ ]:

```
#From the above chart, we have concluded that parents' education has a pos
```

parents maritial status on students score

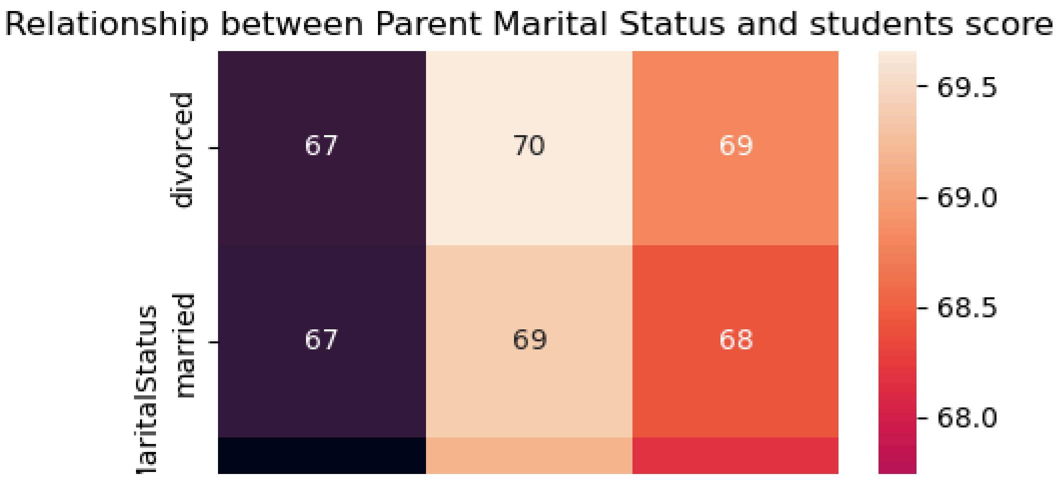
In [36]:

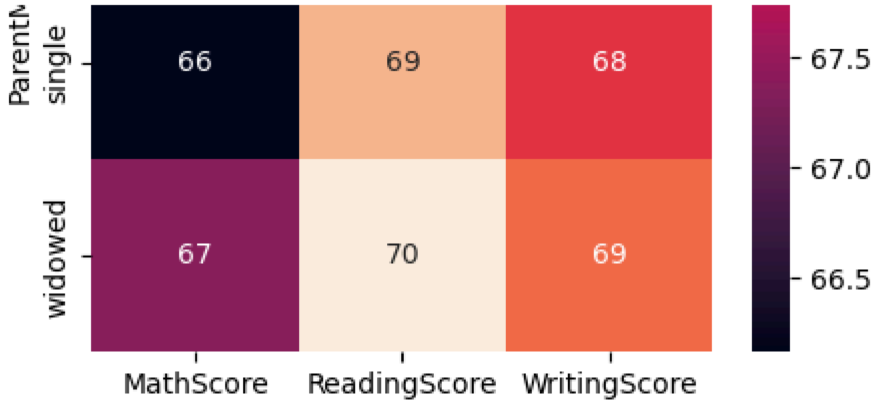
```
gb1=df.groupby("ParentMaritalStatus").agg({"MathScore" :'mean',"ReadingSco
print(gb1)
```

```
                     MathScore  ReadingScore  WritingScore
ParentMaritalStatus                                       
divorced             66.691197     69.655011     68.799146
married              66.657326     69.389575     68.420981
single               66.165704     69.157250     68.174440
widowed              67.368866     69.651438     68.563452
```

In [49]:

```
# If you want to create a chart for this, we can use seaborn to create a he

plt.figure(figsize=(5,5))

plt.title("Relationship between Parent Marital Status and students score")
#This will be the title of the chart, which will tell us what the chart is

sns.heatmap(gb1,annot=True)   #We have a heat map in seaborn >>> where we w
plt.show()
```





In [ ]:

```
#The parents' marital status is not having a significant impact on student
```

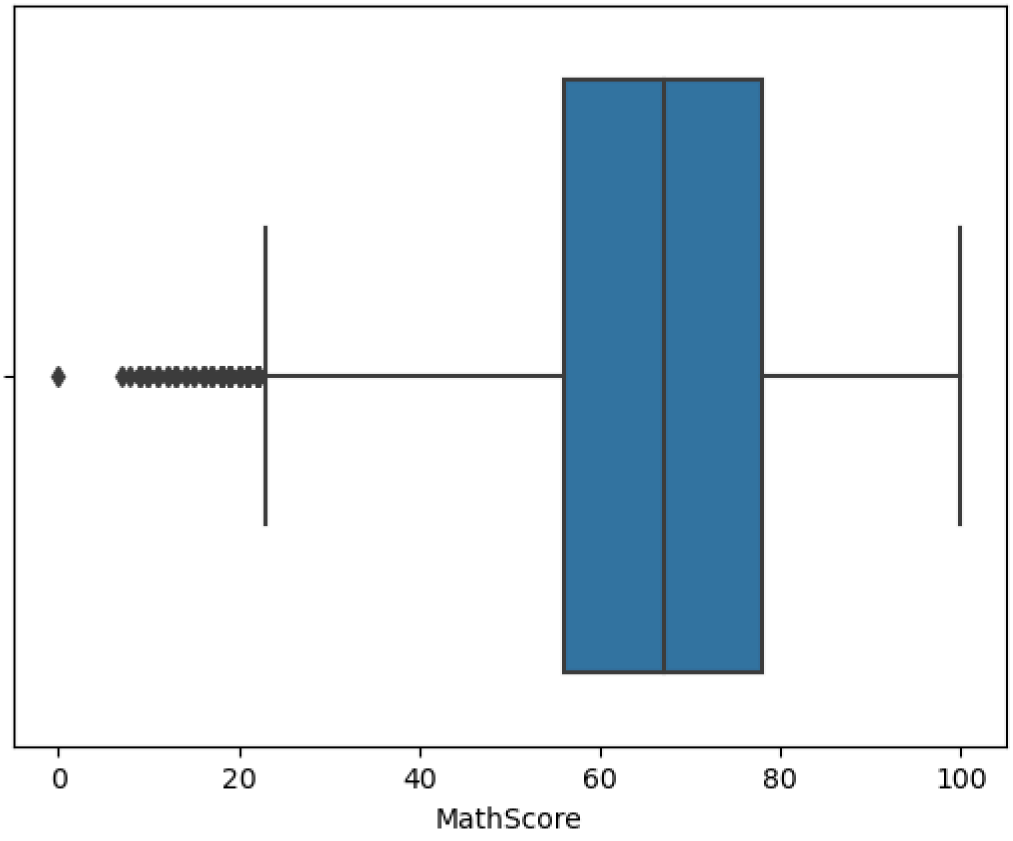
In [ ]:

```
#So from the above chart, we have analyzed that there is no/negligible imp
```

Box Plot

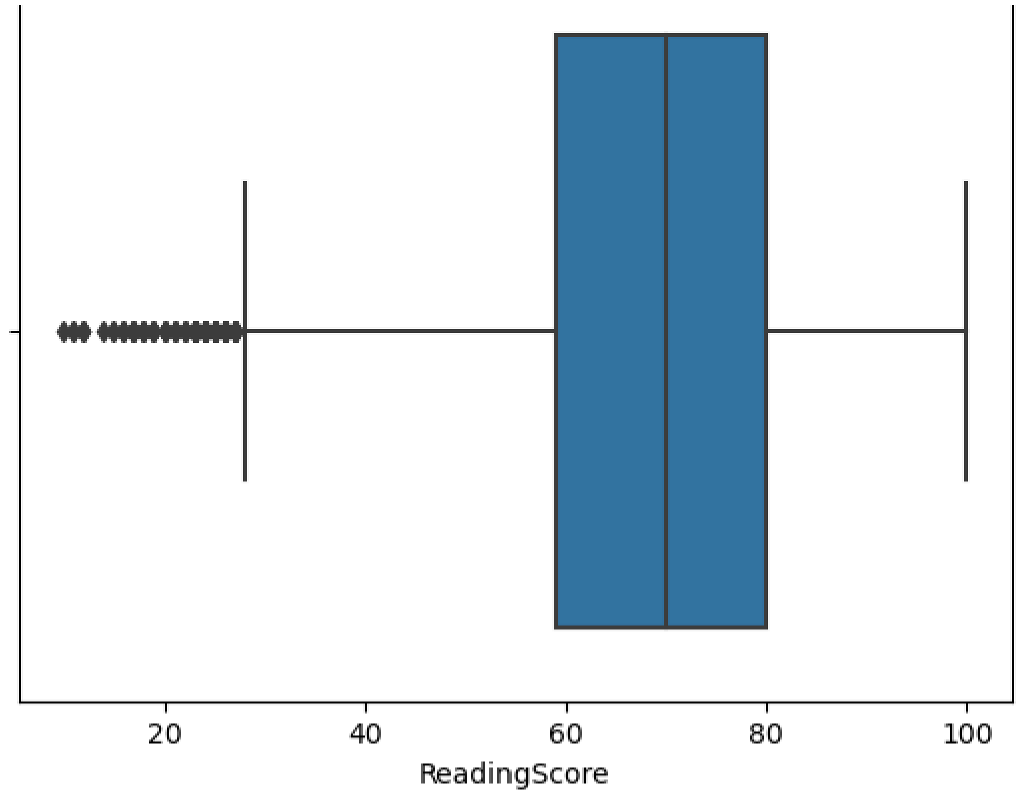
In [56]:

```
sns.boxplot(data=df, x="MathScore")
plt.show()
```



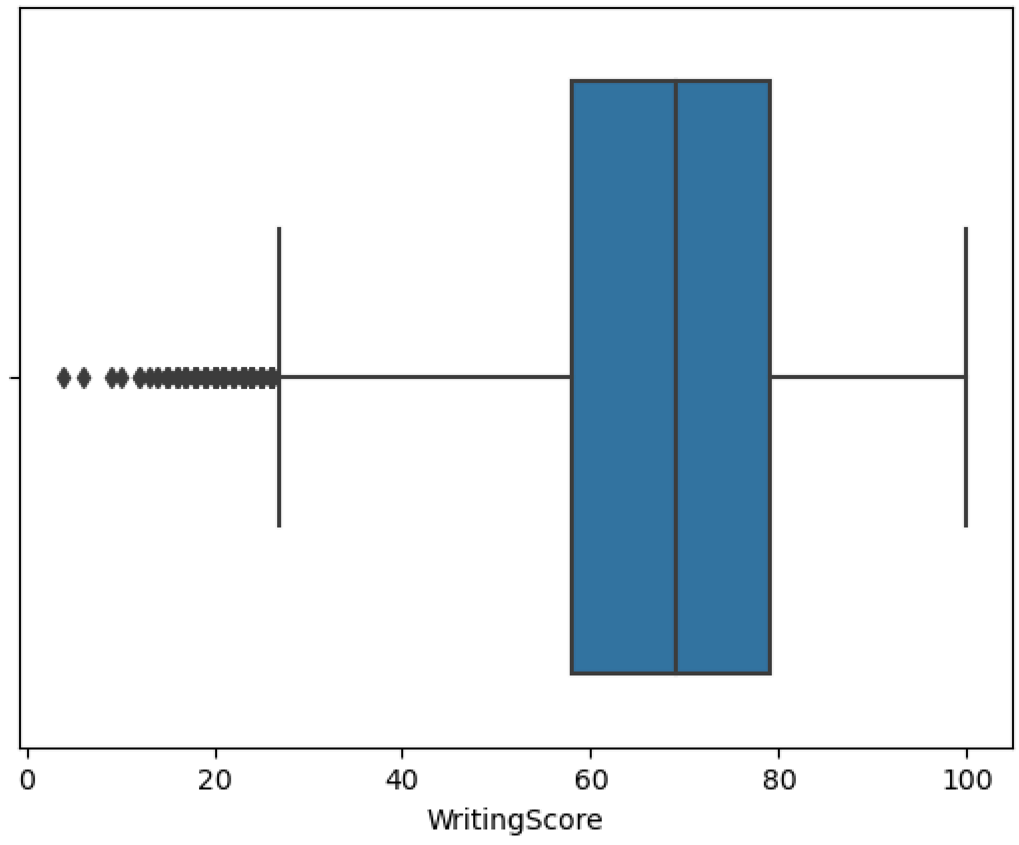
In [57]:

```
sns.boxplot(data=df, x="ReadingScore")
plt.show()
```



In [58]:

```
sns.boxplot(data=df, x="WritingScore")
plt.show()
```



In [30]:

```
print(df["EthnicGroup"].unique())
```

```
[nan 'group C' 'group B' 'group A' 'group D' 'group E']
```

Distribution of Ethnic Group

In [ ]:
```
# We will use a pie chart for finding out percentages.

# We will use the .loc function of pandas.
```

In [8]:
```
groupA=df.loc[(df["EthnicGroup"]=="group A")].count()
print(groupA)

#We only have the values or details that are present within Group "A".
```

```
Unnamed: 0             2219
Gender                 2219
EthnicGroup            2219
ParentEduc             2078
LunchType              2219
TestPrep               2081
ParentMaritalStatus    2121
PracticeSport          2167
IsFirstChild           2168
NrSiblings             2096
TransportMeans         1999
WklyStudyHours         2146
MathScore              2219
ReadingScore           2219
WritingScore           2219
dtype: int64
```

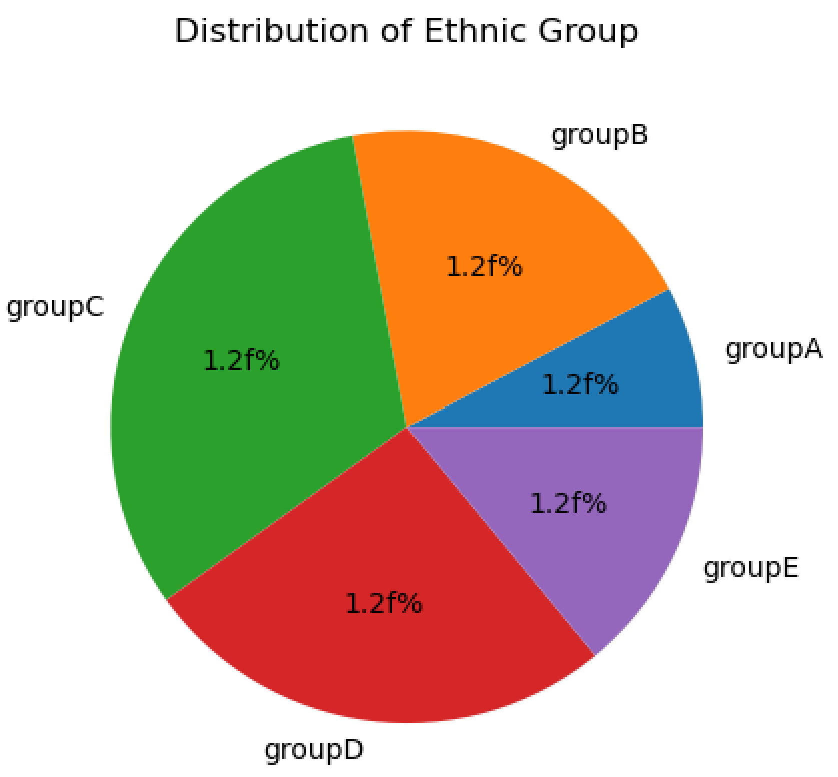
In [18]:
```
#We can check this very thing by removing the parentheses after the count.
groupA=df.loc[(df["EthnicGroup"]=="group A")].count
print(groupA)
```

```
<bound method DataFrame.count of        Unnamed: 0  Gender EthnicGroup
ParentEduc     LunchType  \
3               3    male     group A  associate's degree  free/reduced
13             13    male     group A        some college      standard
14             14  female     group A     master's degree      standard
25             25    male     group A     master's degree  free/reduced
56             61    male     group A    some high school  free/reduced
...           ...     ...         ...                 ...           ...
30603         281    male     group A         high school      standard
30621         638  female     group A   bachelor's degree      standard
30622         640    male     group A  associate's degree  free/reduced
30627         730  female     group A         high school      standard
30634         785    male     group A  associate's degree  free/reduced

        TestPrep ParentMaritalStatus PracticeSport IsFirstChild  NrSiblings
\
3           none             married         never           no         1.0
13     completed              single     sometimes          yes         1.0
14          none            divorced     sometimes          yes         2.0
25          none             married     regularly          yes         1.0
56          none             married     sometimes          yes         NaN
...          ...                 ...           ...          ...         ...
30603       none              single     regularly           no         2.0
30621       none              single     regularly           no         2.0
30622  completed            divorced     regularly           no         3.0
30627  completed             married         never           no         NaN
30634  completed                 NaN     sometimes           no         2.0

      TransportMeans WklyStudyHours  MathScore  ReadingScore  WritingScore
3                NaN         5 - 10         45            56            42
```

```
13           private          > 10        80         73         71
14           private           < 5        48         53         58
25        school_bus        5 - 10        75         76         76
56        school_bus        5 - 10        39         39         34
...              ...           ...       ...        ...        ...
30603     school_bus        5 - 10        71         63         65
30621        private        5 - 10        66         80         78
30622        private        5 - 10        53         53         53
30627     school_bus          > 10        58         77         82
30634     school_bus        5 - 10        65         60         60

[2219 rows x 15 columns]>
```

In [33]:

```
#Now we can also calculate for the remaining groups.
groupA=df.loc[(df["EthnicGroup"]=="group A")].count()
groupB=df.loc[(df["EthnicGroup"]=="group B")].count()
groupC=df.loc[(df["EthnicGroup"]=="group C")].count()
groupD=df.loc[(df["EthnicGroup"]=="group D")].count()
groupE=df.loc[(df["EthnicGroup"]=="group E")].count()







l = ["groupA","groupB","groupC","groupD","groupE"]

mlist=(groupA["EthnicGroup"],groupB["EthnicGroup"],groupC["EthnicGroup"],g
plt.pie(mlist,labels=l , autopct="1.2f%%")               #DataHasBeenPlott
                                                        #ShowPercentages #



plt.title("Distribution of Ethnic Group")   # We used it to put in the tit
plt.show()


print(mlist)    #In the end, we also printed and checked the values.
```



```
(2219, 5826, 9212, 7503, 4041)
```

In [32]:

```
ax=sns.countplot(data=df,x="EthnicGroup")
ax.bar_label(ax.containers[0])
```

Out[32]:

```
[Text(0, 0, '9212'),
 Text(0, 0, '5826'),
 Text(0, 0, '2219'),
 Text(0, 0, '7503'),
 Text(0, 0, '4041')]
```

9212
8000
7503