In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
#Generating new features from existing features is known as Feature Engineering
#Extracting helpful info from already existing data
```

In [3]:
```
Student_Result_Analysis=pd.read_csv(r"C:\Users\hemil\OneDrive\Desktop\Data Analyst\EDA PYTHON\Student Exam Score Data Analysis\Ex
Student_Result_Analysis.head()
```

Out[3]:

	Unnamed: 0	Gender	EthnicGroup	ParentEduc	LunchType	TestPrep	ParentMaritalStatus	PracticeSport	IsFirstChild	NrSiblings	TransportMeans	WklyStudyH
0	0	female	NaN	bachelor's degree	standard	none	married	regularly	yes	3.0	school_bus	
1	1	female	group C	some college	standard	NaN	married	sometimes	yes	0.0	NaN	5
2	2	female	group B	master's degree	standard	none	single	sometimes	yes	4.0	school_bus	
3	3	male	group A	associate's degree	free/reduced	none	married	never	no	1.0	NaN	5
4	4	male	group C	some college	standard	none	married	sometimes	yes	0.0	school_bus	5

In [4]:
```
Student_Result_Analysis.info()
```

```
<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 [5]:
```
Student_Result_Analysis.isna().sum()
```

Out[5]:
```
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
```

In [6]:
```
Student_Result_Analysis.describe()
```

Out[6]:

	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 [7]:
```
Student_Result_Analysis= Student_Result_Analysis.drop('Unnamed: 0',axis=1)
```

In [8]:
```
Student_Result_Analysis.head()
```

Out[8]:

	Gender	EthnicGroup	ParentEduc	LunchType	TestPrep	ParentMaritalStatus	PracticeSport	IsFirstChild	NrSiblings	TransportMeans	WklyStudyHours	MathS
0	female	NaN	bachelor's degree	standard	none	married	regularly	yes	3.0	school_bus	< 5	
1	female	group C	some college	standard	NaN	married	sometimes	yes	0.0	NaN	5 - 10	
2	female	group B	master's degree	standard	none	single	sometimes	yes	4.0	school_bus	< 5	
3	male	group A	associate's degree	free/reduced	none	married	never	no	1.0	NaN	5 - 10	
4	male	group C	some college	standard	none	married	sometimes	yes	0.0	school_bus	5 - 10	

In [9]:
```
Student_Result_Analysis['EthnicGroup'].unique()
```

Out[9]:
```
array([nan, 'group C', 'group B', 'group A', 'group D', 'group E'],
      dtype=object)
```

In [10]:
```
Student_Result_Analysis['EthnicGroup'] = Student_Result_Analysis['EthnicGroup'].replace({
    'group C': 'C',
    'group B': 'B',
    'group A': 'A',
    'group D': 'D',
    'group E': 'E'
})
```

In [11]:
```
Student_Result_Analysis['EthnicGroup'].unique()
```

Out[11]:
```
array([nan, 'C', 'B', 'A', 'D', 'E'], dtype=object)
```

In [12]:
```
Student_Result_Analysis['ParentEduc'].unique()
```

Out[12]:
```
array(["bachelor's degree", 'some college', "master's degree",
       "associate's degree", 'high school', 'some high school', nan],
      dtype=object)
```

In [13]:
```
Student_Result_Analysis['ParentEduc'] = Student_Result_Analysis['ParentEduc'].replace({
    "bachelor's degree": 'Bachelors',
    'some college': 'College',
    "master's degree": 'Masters',
     "associate's degree": 'Associate',
    "high school": 'High_school',
    "some high school":'High_school'
})
```

In [14]:
```
Student_Result_Analysis['ParentEduc'].unique()
```

Out[14]:
```
array(['Bachelors', 'College', 'Masters', 'Associate', 'High_school', nan],
      dtype=object)
```

In [15]:
```
Student_Result_Analysis['LunchType'].unique()
```

Out[15]:
```
array(['standard', 'free/reduced'], dtype=object)
```

In [16]:
```
Student_Result_Analysis['ParentMaritalStatus'].unique()
```

Out[16]:
```
array(['married', 'single', 'widowed', nan, 'divorced'], dtype=object)
```

In [17]:
```
Student_Result_Analysis['ParentMaritalStatus'] = Student_Result_Analysis['ParentMaritalStatus'].replace({
    "married": 'Married',
    'single': 'divorced/widowed',
    "widowed": 'divorced/widowed',
    "divorced": 'divorced/widowed'
})
```

In [18]:
```
Student_Result_Analysis['ParentMaritalStatus'].unique()
```

Out[18]:
```
array(['Married', 'divorced/widowed', nan], dtype=object)
```

In [19]:
```
Student_Result_Analysis['TestPrep'].unique()
```

Out[19]:
```
array(['none', nan, 'completed'], dtype=object)
```

In [20]:
```
Student_Result_Analysis['TestPrep'] = Student_Result_Analysis['TestPrep'].replace({
    "completed": 'Complete',
    'none': 'Incomplete',
     })
Student_Result_Analysis['TestPrep'].fillna('Incomplete', inplace=True)
```

In [21]:
```
Student_Result_Analysis['TestPrep'].unique()
```

Out[21]:
```
array(['Incomplete', 'Complete'], dtype=object)
```

In [22]:
```
Student_Result_Analysis['PracticeSport'].unique()
```

Out[22]:
```
array(['regularly', 'sometimes', 'never', nan], dtype=object)
```

In [23]:
```
Student_Result_Analysis['PracticeSport'] = Student_Result_Analysis['PracticeSport'].replace({
    "regularly": 'Frequently',
    'sometimes':'Sometimes',
    'never':'Never'
     })
```

In [24]:
```
Student_Result_Analysis['PracticeSport'].unique()
```

Out[24]:
```
array(['Frequently', 'Sometimes', 'Never', nan], dtype=object)
```

In [25]:
```
Student_Result_Analysis['IsFirstChild'].unique()
```

Out[25]:
```
array(['yes', 'no', nan], dtype=object)
```

In [26]:
```
Student_Result_Analysis['NrSiblings'].unique()
```

Out[26]:
```
array([ 3.,  0.,  4.,  1., nan,  2.,  5.,  7.,  6.])
```

In [27]:
```
Student_Result_Analysis['TransportMeans'].unique()
```

Out[27]:
```
array(['school_bus', nan, 'private'], dtype=object)
```

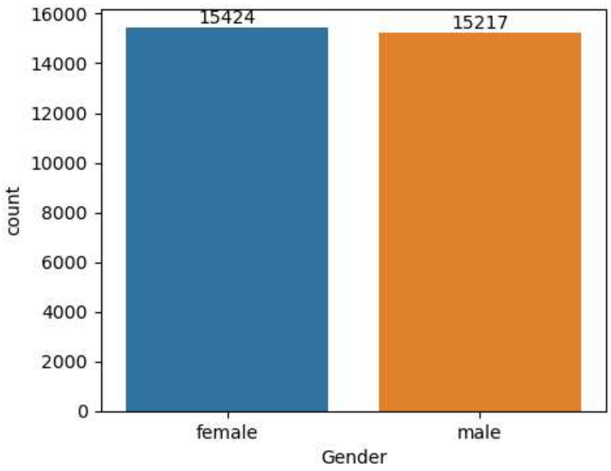
In [28]:
```
Student_Result_Analysis['WklyStudyHours'].unique()
```

Out[28]:
```
array(['< 5', '5 - 10', '> 10', nan], dtype=object)
```

In [29]:
```
#Their are many null values in the dataset, you can Impute some data also.
#Some columns like EthnicGroup might have NAN values,as it is completely possible that a guy/girl does not belong to specific eth
#But some columns like weekly hours studied,colud not be empty as it range from 0 to Infinity.
#Here I ahve not doen any tpye of Imputation
```

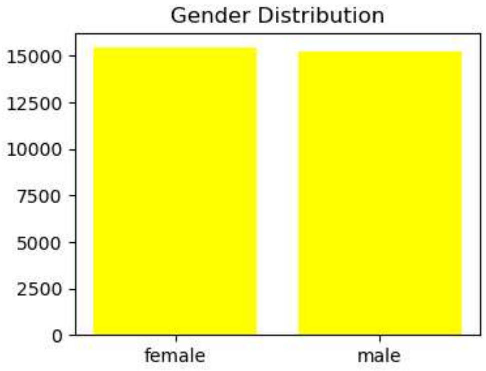
In [30]:
```
#gender distribution
'''Bar Container: In a bar plot
, each set of bars (e.g., the bars for different categories or series) is held within a bar container.
The container allows you to apply operations to all the bars at once
, like adding labels or changing their appearance.'''
plt.figure(figsize=(5,4))
ax=sns.countplot(data=Student_Result_Analysis,x='Gender')
ax.bar_label(ax.containers[0]) #Adding labels
plt.show
```

Out[30]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



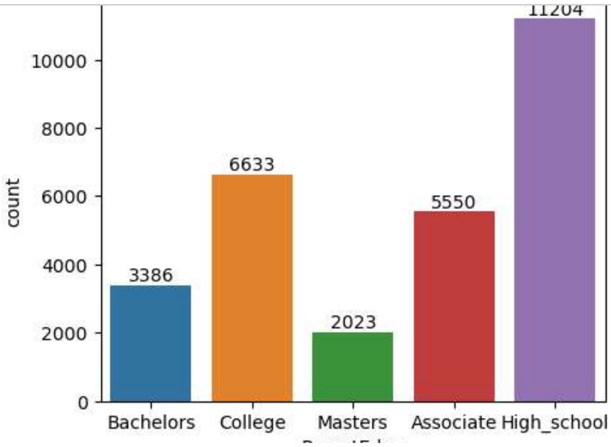
In [31]:
```
# Count the values in the 'Gender' column
gender_counts = Student_Result_Analysis['Gender'].value_counts()

# Create a bar plot
plt.figure(figsize=(4, 3))
ax=plt.bar(gender_counts.index, gender_counts.values, color='yellow')

plt.title('Gender Distribution')
```

Out[31]:
```
Text(0.5, 1.0, 'Gender Distribution')
```



In [32]:

```
plt.figure(figsize=(5,4))
ax=sns.countplot(data=Student_Result_Analysis,x='ParentEduc')
ax.bar_label(ax.containers[0]) #Adding labels
plt.show
```



In [33]:

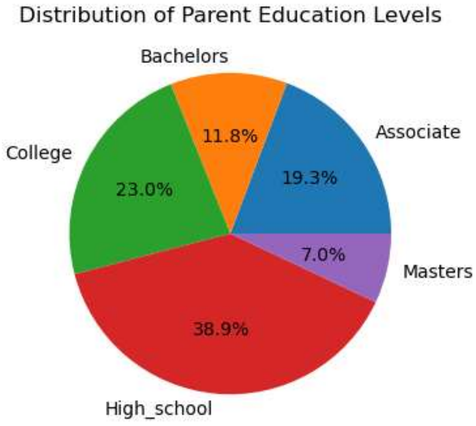
```
ParentEduc_Count=Student_Result_Analysis.groupby('ParentEduc').size()
ParentEduc_Count
```

Out[33]:

```
ParentEduc
Associate       5550
Bachelors       3386
College         6633
High_school    11204
Masters         2023
dtype: int64
```

In [34]:

```
# Group by 'ParentEduc' and get the counts (returns a Series)
ParentEduc_Count = Student_Result_Analysis.groupby('ParentEduc').size()

# Create the pie chart
plt.figure(figsize=(4, 4))
plt.pie(ParentEduc_Count, labels=ParentEduc_Count.index, autopct='%1.1f%%')


# Add a title
plt.title('Distribution of Parent Education Levels')

# Show the pie chart
plt.show()
```



In [35]:

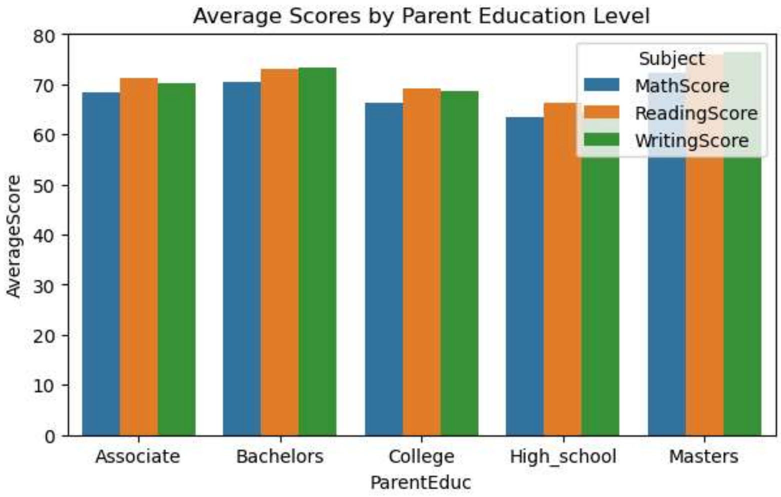
```
ParentEduc_Count=Student_Result_Analysis.groupby('ParentEduc').agg({"MathScore":'mean',"ReadingScore":'mean',"WritingScore":'mean
ParentEduc_Count
```

Out[35]:

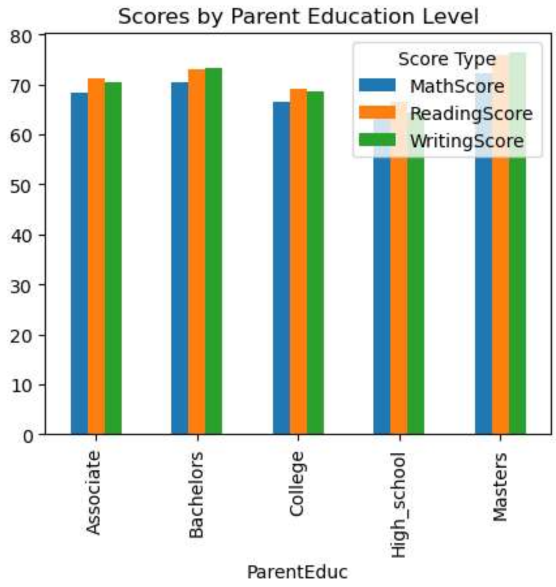
ParentEduc	MathScore	ReadingScore	WritingScore
Associate	68.365586	71.124324	70.299099
Bachelors	70.466627	73.062020	73.331069
College	66.390472	69.179708	68.501432
High_school	63.523920	66.375312	64.540343
Masters	72.336134	75.832921	76.356896

In [36]:

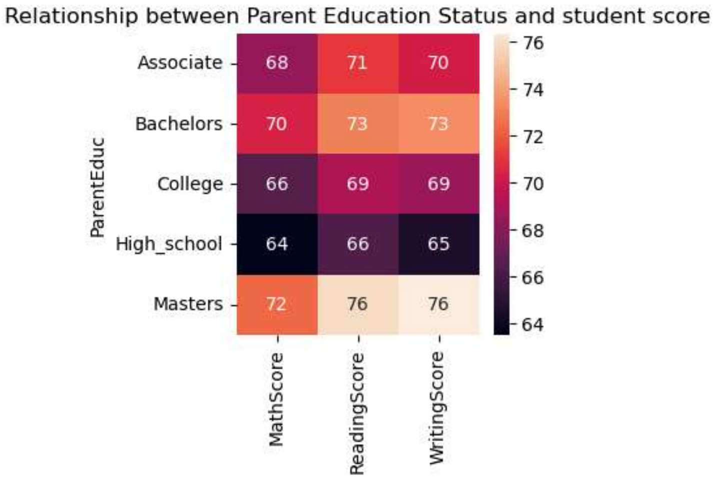
```

plt.figure(figsize=(7, 4))

# Plot each score type with a separate call to sns.barplot
sns.barplot(data=ParentEduc_Count.reset_index().melt(id_vars='ParentEduc', var_name='Subject', value_name='AverageScore'),
            x='ParentEduc', y='AverageScore', hue='Subject')

plt.title('Average Scores by Parent Education Level')
plt.legend(title='Subject')
plt.show()
```



In [37]:
```
# Create a figure and axes
fig, ax = plt.subplots(1, 1, figsize=(5, 4))

# Plot the data
ParentEduc_Count.plot(kind='bar', ax=ax)

# Customize the plot
ax.set_title('Scores by Parent Education Level')
ax.legend(title='Score Type')

# Display the plot
plt.show()
```



In [38]:
```
plt.figure(figsize=(3, 3))
plt.title("Relationship between Parent Education Status and student score")
sns.heatmap(ParentEduc_Count,annot=True)
plt.show()
```

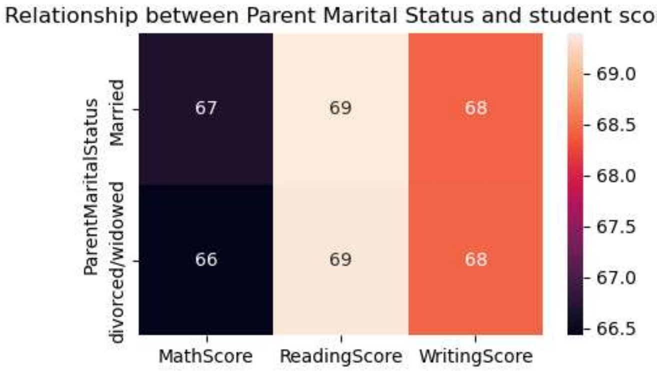


In [39]:
```
Parent_Marital_Status=Student_Result_Analysis.groupby('ParentMaritalStatus').agg({"MathScore":'mean',"ReadingScore":'mean',"Writi
Parent_Marital_Status
```

Out[39]:

	MathScore	ReadingScore	WritingScore
ParentMaritalStatus			
Married	66.657326	69.389575	68.420981
divorced/widowed	66.427144	69.374633	68.436424

In [40]:
```
plt.figure(figsize=(5, 3))
plt.title("Relationship between Parent Marital Status and student score")
sns.heatmap(Parent_Marital_Status,annot=True)
plt.show()
```

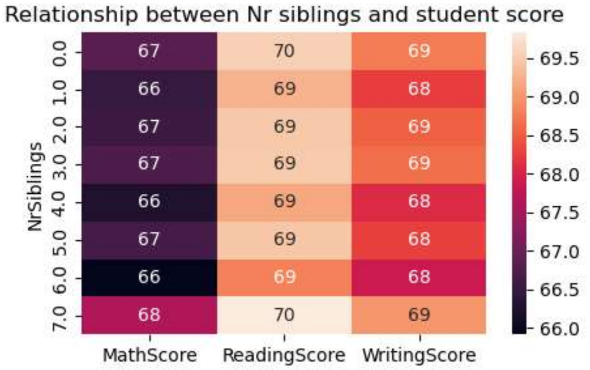


In [41]:
```
Nr_Siblings=Student_Result_Analysis.groupby('NrSiblings').agg({"MathScore":'mean',"ReadingScore":'mean',"WritingScore":'mean'})
Nr_Siblings
```

Out[41]:

NrSiblings	MathScore	ReadingScore	WritingScore
0.0	66.819449	69.547812	68.746515
1.0	66.473896	69.259097	68.245345
2.0	66.554934	69.472018	68.522533
3.0	66.719092	69.488159	68.650498
4.0	66.245495	69.144169	68.073444
5.0	66.630303	69.453788	68.282576
6.0	65.917219	68.801325	67.860927
7.0	67.615120	69.828179	68.986254

In [42]:
```
plt.figure(figsize=(5, 3))
plt.title("Relationship between Nr siblings and student score")
sns.heatmap(Nr_Siblings,annot=True)
plt.show()
```

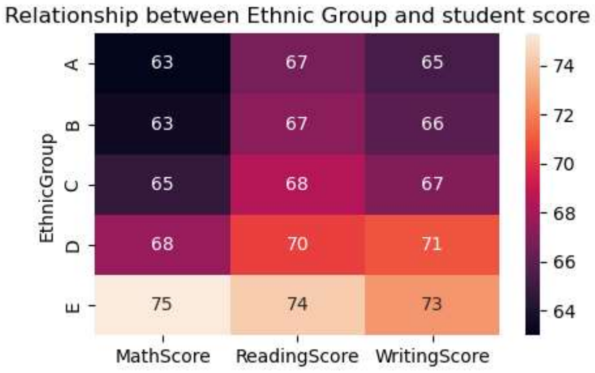


In [43]:
```
Ethnic_Group=Student_Result_Analysis.groupby('EthnicGroup').agg({"MathScore":'mean',"ReadingScore":'mean',"WritingScore":'mean'})
Ethnic_Group
```

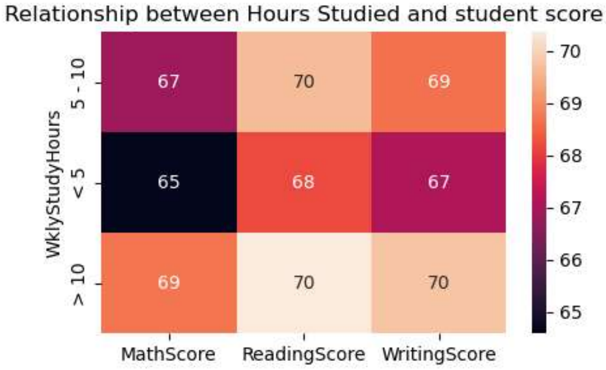
Out[43]:

EthnicGroup	MathScore	ReadingScore	WritingScore
A	62.991888	66.787742	65.251915
B	63.490216	67.320460	65.895125
C	64.695723	68.438233	66.999240
D	67.666400	70.382247	70.890844
E	75.298936	74.251423	72.677060

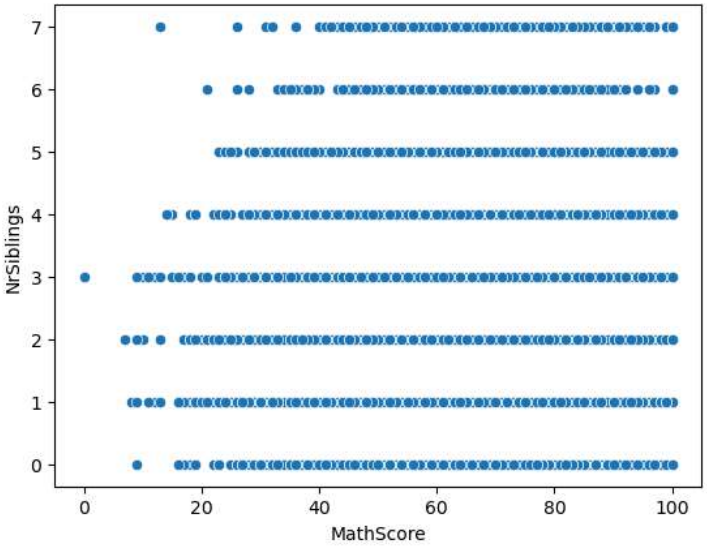
In [44]:
```
plt.figure(figsize=(5, 3))
plt.title("Relationship between Ethnic Group and student score")
sns.heatmap(Ethnic_Group,annot=True)
plt.show()
```



In [45]:
```
Wkly_Study_Hours=Student_Result_Analysis.groupby('WklyStudyHours').agg({"MathScore":'mean',"ReadingScore":'mean',"WritingScore":'
Wkly_Study_Hours
plt.figure(figsize=(5, 3))
plt.title("Relationship between Hours Studied and student score")
sns.heatmap(Wkly_Study_Hours,annot=True)
plt.show()
```

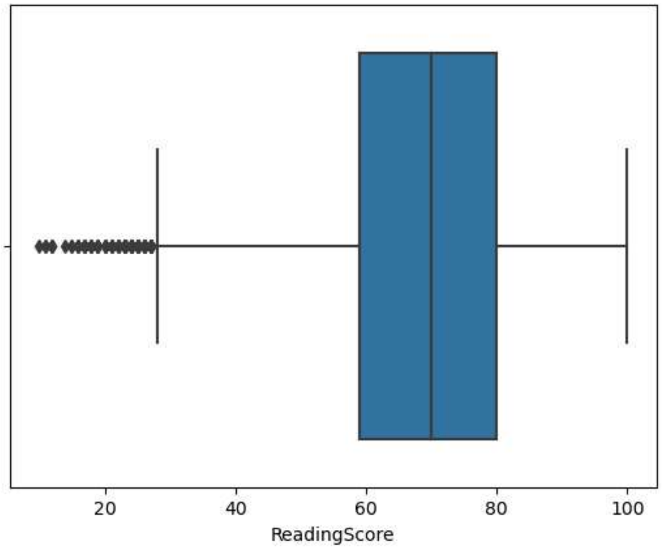


In [46]:
```
#we are using scatter plot for detecting outliers from the data of maths score
sns.scatterplot(data=Student_Result_Analysis,x='MathScore',y='NrSiblings')
plt.show()
```



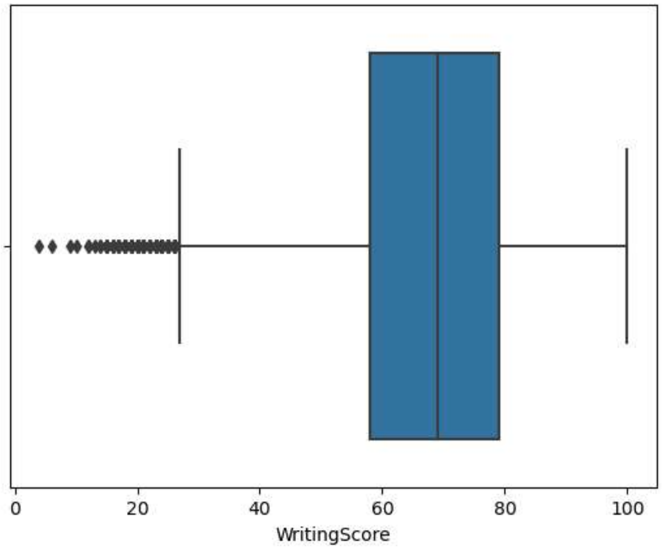
In [47]:

```
#we are using box plot for detecting outliers for reading score
sns.boxplot(data=Student_Result_Analysis,x='ReadingScore')
plt.show()
```



In [48]:

```
#we are using box plot for detecting outliers for Writing score
sns.boxplot(data=Student_Result_Analysis,x='WritingScore')
plt.show()
```



In [49]:

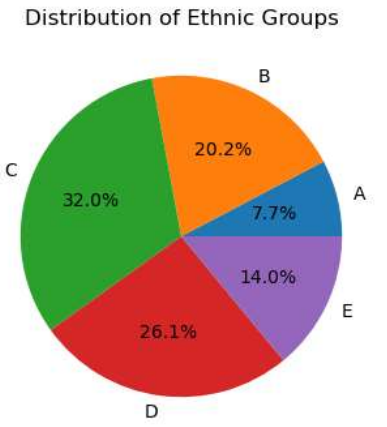
```
Ethnic_Group=Student_Result_Analysis.groupby('EthnicGroup').size()
Ethnic_Group
```

Out[49]:

```
EthnicGroup
A    2219
B    5826
C    9212
D    7503
E    4041
dtype: int64
```

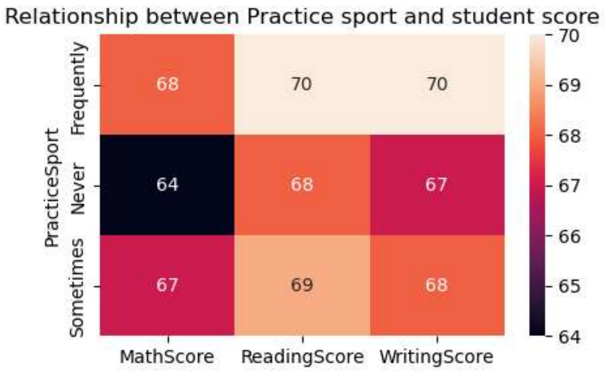
In [50]:

```
# Create the pie chart
plt.figure(figsize=(4, 4))
plt.pie(Ethnic_Group, labels=Ethnic_Group.index, autopct='%1.1f%%')


# Add a title
plt.title('Distribution of Ethnic Groups')

# Show the pie chart
plt.show()
```



In [51]:

```
Practice_Sport=Student_Result_Analysis.groupby('PracticeSport').agg({"MathScore":'median',"ReadingScore":'median',"WritingScore":
Practice_Sport
plt.figure(figsize=(5, 3))
plt.title("Relationship between Practice sport and student score")
sns.heatmap(Practice_Sport,annot=True)
plt.show()
```



In [ ]: