

Importing libraries

```
In [2]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Connecting Dataset

```
In [3]: df = pd.read_csv("student_score_dataset.csv")
```

Data Exploration

```
In [77]: num_rows, num_cols = df.shape

print("Number of rows:", num_rows)
print("Number of columns:", num_cols)
```

Number of rows: 30641

Number of columns: 16

```
In [22]: df.head()
```

Out[22]:

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

In [16]: df.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 [17]: `df.describe()`

Out[17]:

	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 [8]: df.isnull().sum()
```

```
Out[8]: 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 [9]: df.nunique()
```

```
Out[9]: Unnamed: 0    1000
        Gender      2
        EthnicGroup  5
        ParentEduc  6
        LunchType   2
        TestPrep    2
        ParentMaritalStatus  4
        PracticeSport  3
        IsFirstChild  2
        NrSiblings   8
        TransportMeans  2
        WklyStudyHours  3
        MathScore    95
        ReadingScore  90
        WritingScore  93
        dtype: int64
```

Drop Unnamed column

```
In [18]: 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

Add a New column Total marks

```
In [49]: df["Totalmarks"] = (df["MathScore"] + df["ReadingScore"] + df["WritingScore"]) / 3
df["Totalmarks"] = df["Totalmarks"].round(2)

df.head(2)
```

Out[49]:

	Gender	EthnicGroup	ParentEduc	LunchType	TestPrep	ParentMaritalStatus	PracticeSport	IsFirstChild	NrSiblings	TransportationMode
0	female	NaN	bachelor's degree	standard	none	married	regularly	yes	3.0	school bus
1	female	group C	some college	standard	NaN	married	sometimes	yes	0.0	parent drop off

Add new conditional column


```
In [51]: # Function to categorize students
def categorize_marks(Totalmarks):
    if Totalmarks > 85:
        return 'Brilliant'
    elif Totalmarks >= 65:
        return 'Above Average'
    elif Totalmarks >= 50:
        return 'Average'
    else:
        return 'Below Average'

# Create a new column based on the condition
df['Student_Category'] = df['Totalmarks'].apply(categorize_marks)

# Display the DataFrame
df.head(5)
```

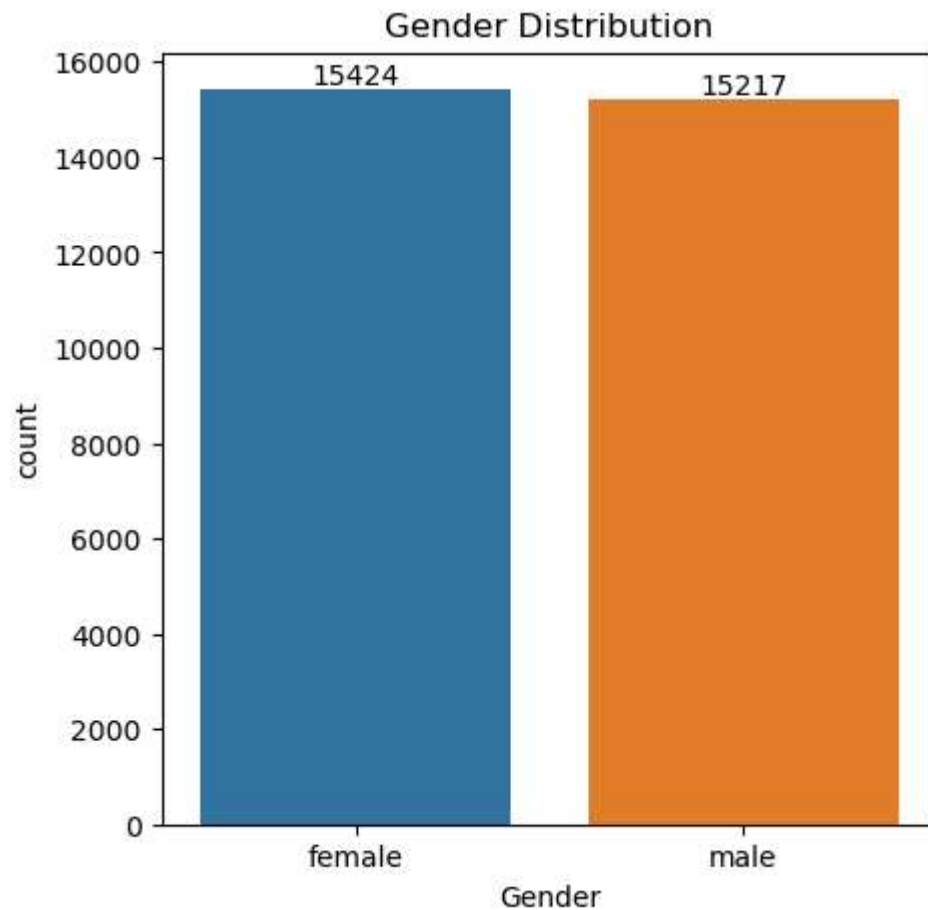
Out[51]:

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



Gender Distribution

```
In [4]: plt.figure(figsize = (5,5))
ax = sns.countplot(data = df , x = "Gender")
ax.bar_label(ax.containers[0])
plt.title("Gender Distribution")
plt.show()
```



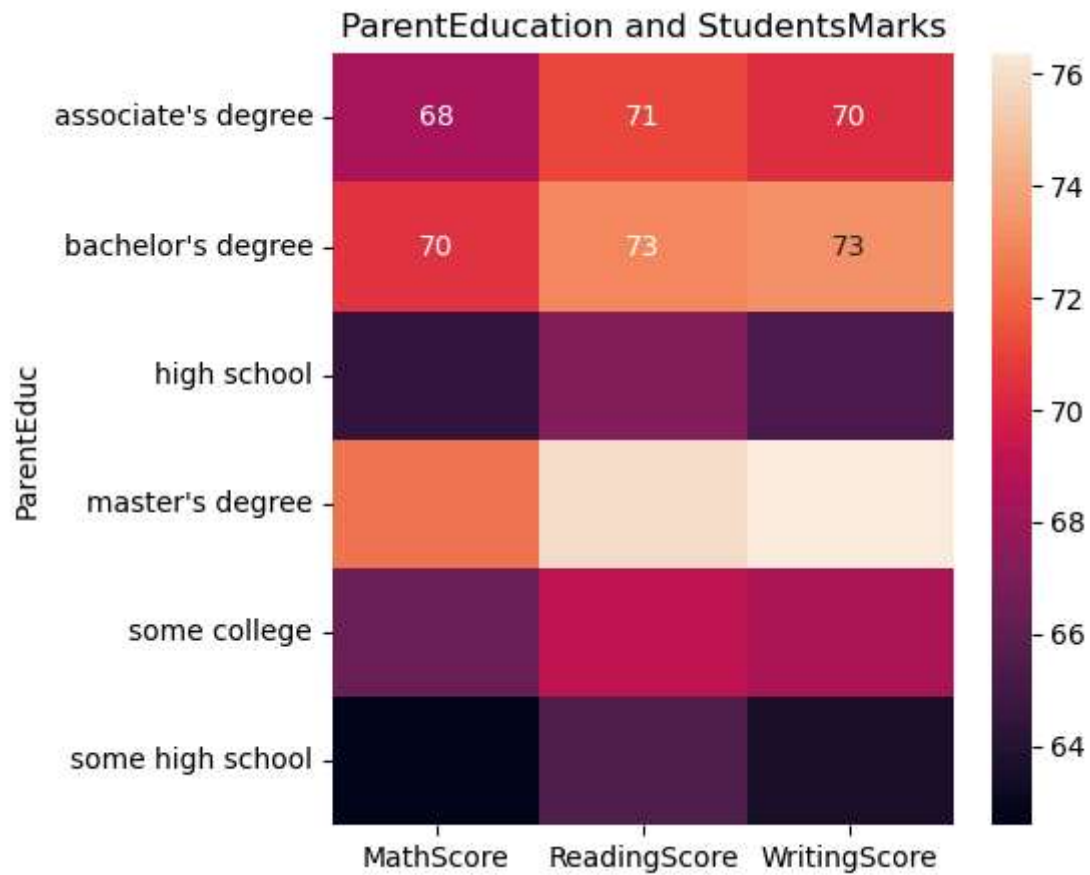
```
In [75]: #From the above chart i have analyzed that  
#The no. of females in the data is more than the no. of males
```

Finding the relation between Parent-education and the scores of their children

```
In [5]: gb = df.groupby("ParentEduc").agg({"MathScore" : "mean" , "ReadingScore" : "mean" , "WritingScore" : "mean"})  
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 [6]: plt.figure(figsize=(5,5)) # Adjust the figure size as needed
sns.heatmap(gb, annot=True)
plt.title('ParentEducation and StudentsMarks')
plt.show()
```



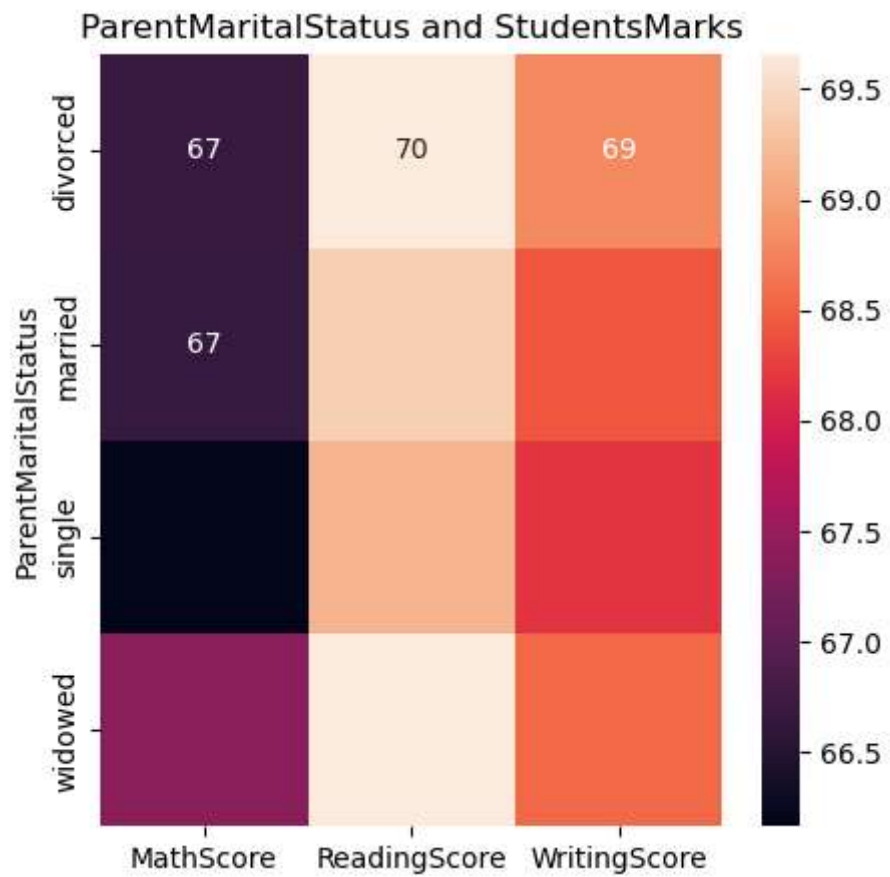
In [74]: *#from the above chart i have concluded that the education level of parents have a good imact on the students marks*

Finding the relation between Parent Marital Status and the scores of their children

```
In [7]: gb1 = df.groupby("ParentMaritalStatus").agg({"MathScore" : "mean" , "ReadingScore" : "mean" , "WritingScore" : "mean"}  
print(gb1)
```

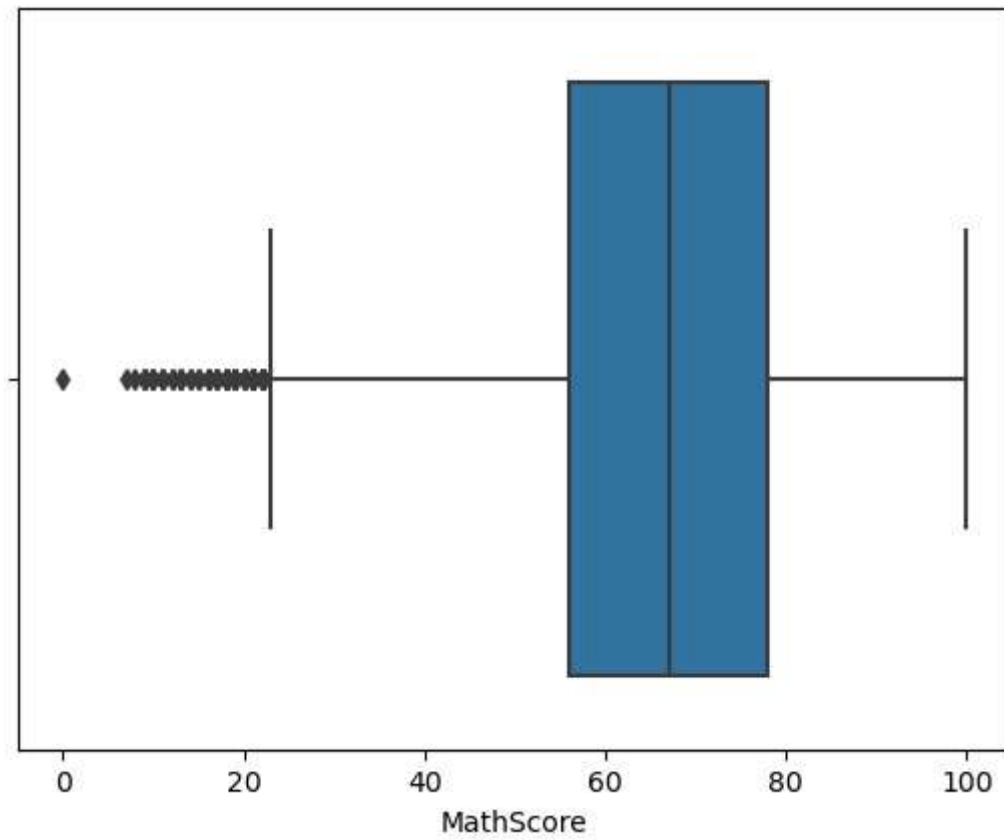
	MathScore	ReadingScore	WritingScore
ParentMaritalStatus			
divorced	66.691197	69.655011	68.799146
married	66.657326	69.389575	68.420981
single	66.165704	69.157250	68.174440
widowed	67.368866	69.651438	68.563452

```
In [8]: plt.figure(figsize=(5,5)) # Adjust the figure size as needed  
sns.heatmap(gb1, annot=True)  
plt.title('ParentMaritalStatus and StudentsMarks')  
plt.show()
```

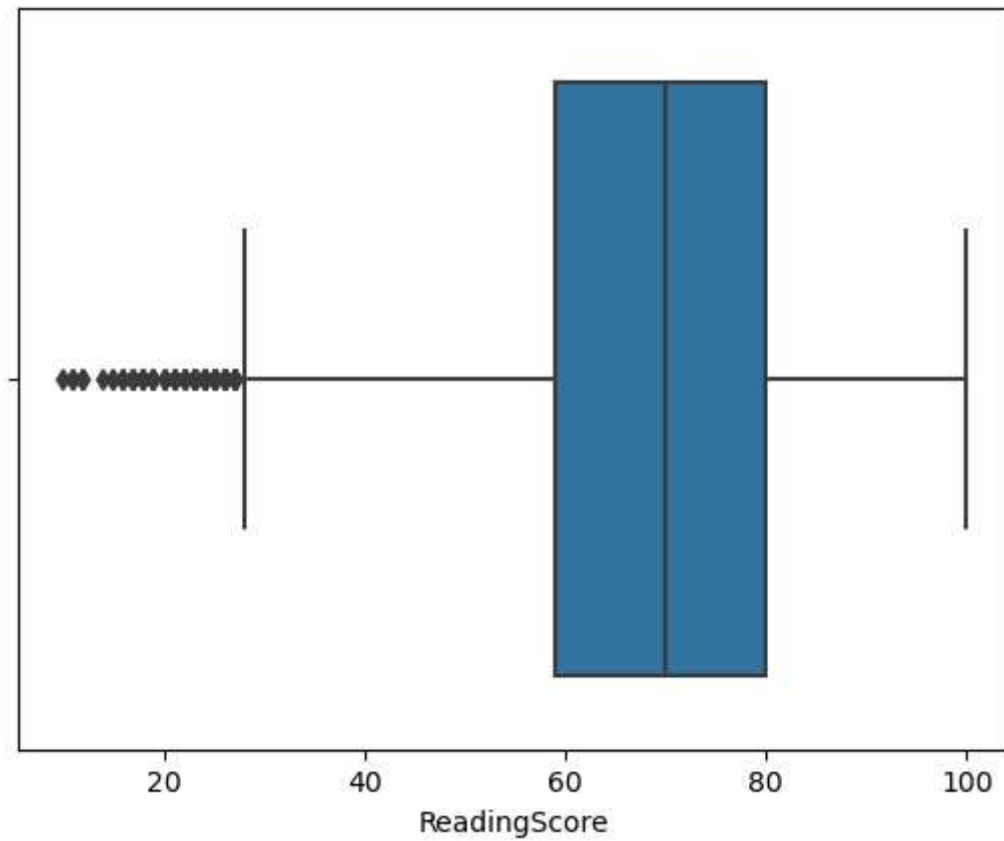


In [78]: *#From the above chart i have concluded that there is no/negligible impact of parrent marital status on students marks*

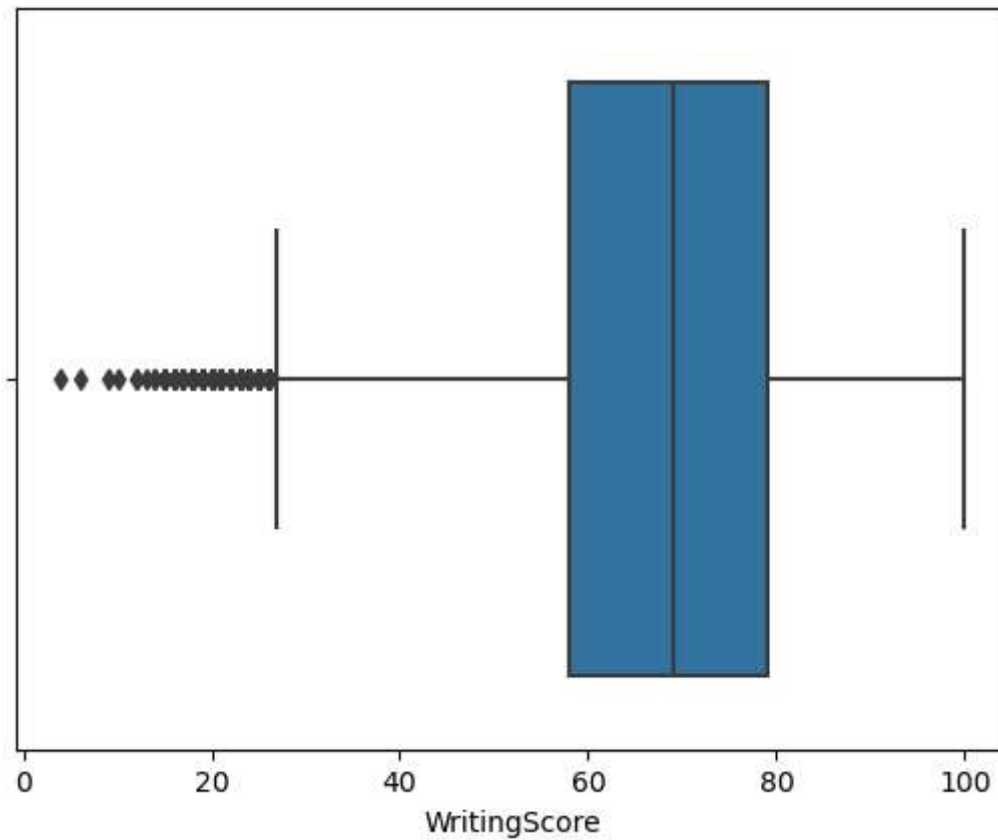
```
In [12]: sns.boxplot(data = df , x = "MathScore")  
plt.show()
```



```
In [13]: sns.boxplot(data = df , x = "ReadingScore")  
plt.show()
```



```
In [14]: sns.boxplot(data = df , x = "WritingScore")  
plt.show()
```



Distribution of Ethnic Group

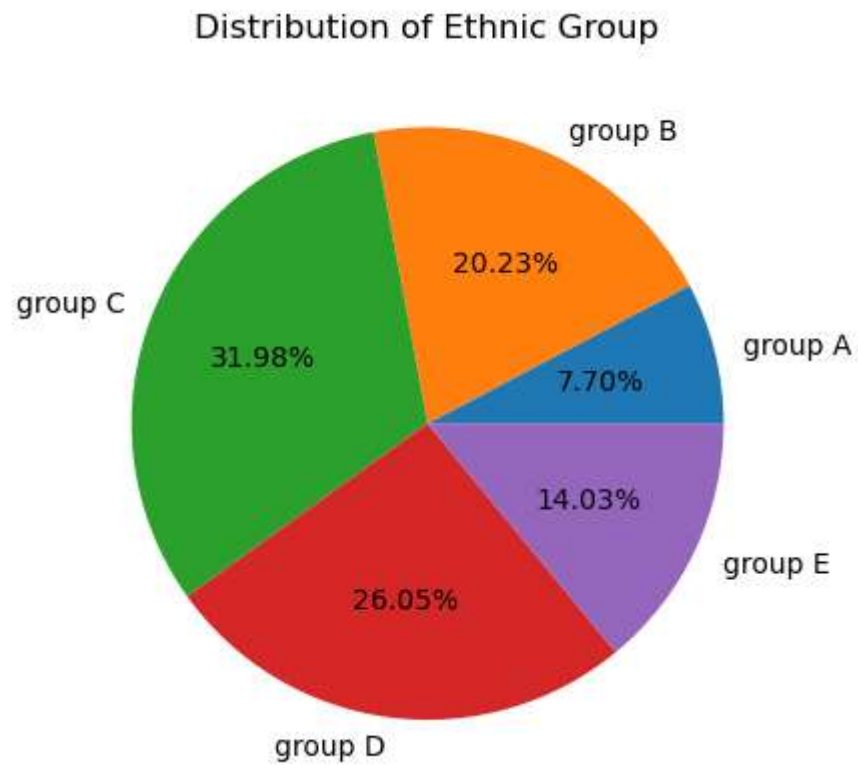
```
In [15]: print(df["EthnicGroup"].unique())
```

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

```
In [33]: 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 = ["group A", "group B", "group C", "group D", "group E"]
mlist = [groupA["EthnicGroup"], groupB["EthnicGroup"], groupC["EthnicGroup"], groupD["EthnicGroup"], groupE["EthnicGroup"]]
```

```
plt.pie(mlist, labels = 1, autopct = "%1.2f%%")  
plt.title("Distribution of Ethnic Group")  
plt.show()
```



```
In [34]: print(df["NrSiblings"].unique())
```

```
[ 3.  0.  4.  1. nan  2.  5.  7.  6.]
```

```
In [65]: df.head()
```

Out[65]:

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

Distribution of Student category

```
In [70]: print(df["Student_Category"].unique())
```

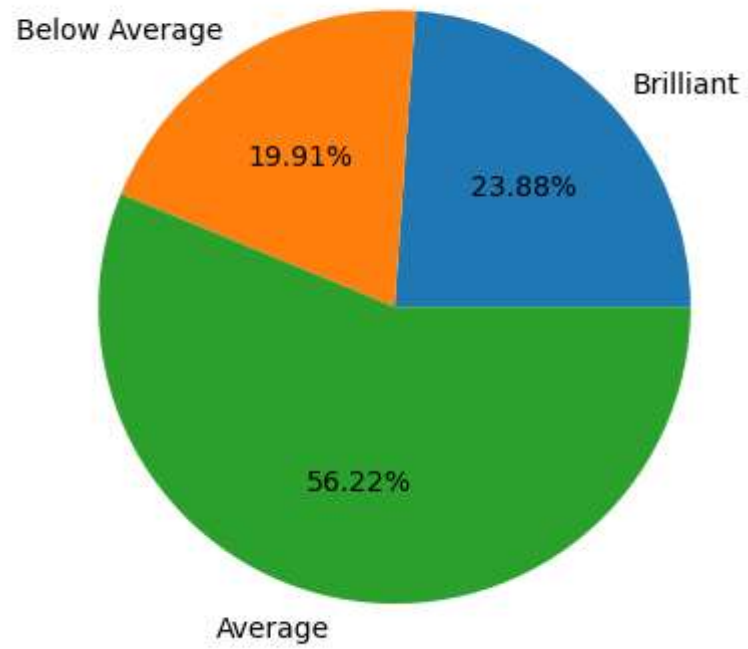
```
['Above Average' 'Brilliant' 'Below Average' 'Average']
```

```
In [75]: #aboveAverage = df.loc[(df["Student_Category"] == "Student_Category").count()
brilliant = df.loc[(df["Student_Category"] == "Brilliant").count()
belowAverage = df.loc[(df["Student_Category"] == "Below Average").count()
average = df.loc[(df["Student_Category"] == "Average").count()

c = ["Brilliant", "Below Average", "Average"]
mlist = [brilliant["Student_Category"], belowAverage["Student_Category"], average["Student_Category"]]

plt.pie(mlist, labels = c, autopct = "%1.2f%%")
plt.title("Distribution of student category")
plt.show()
```


Distribution of student category



In []: