Exam Score Analysis

Prepared by

Yarthem Muivah

muivahyarthem11@gmail.com

Hardik Madnani

hardikmadnani02@gmail.com

Khushi Verma

cse200001036@iiti.ac.in

Nov 11, 2022

DA Major September Batch DS-09-MLB6

Introduction

This analysis's focus is on students' exam performance in a variety of subjects, including arithmetic, reading, and writing. It also discusses how each student performs after attending a test preparation course and determines whether there is a substantial difference in marks depending on the educational background of their parents.

Context

The dataset was obtain vai Kaggle.com.

This data collection contains exam results for students in Math, Reading, and Writing, group, gender, parental level of education and test preparation course.

Acknowledgements

We would like to thank Teachnook for giving us the opportunity to study and implement the concept in the creation of this project. We'd want to thank you again for this chance.

Content

Importing Libraries

In [1]:
```
import warnings
import numpy as np
from plotly.subplots import make_subplots
import plotly.express as px
import plotly.graph_objects as go
import chart_studio.plotly as py
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [2]:
```
dataset = pd.read_csv("exams.csv")
```

In [3]:
```
# copying data
df = dataset
```

In [4]:
```
df
```

Out[4]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	male	group A	high school	standard	completed	67	67	63
1	female	group D	some high school	free/reduced	none	40	59	55
2	male	group E	some college	free/reduced	none	59	60	50
3	male	group B	high school	standard	none	77	78	68
4	male	group E	associate's degree	standard	completed	78	73	68
...	...	...	...	...	...	...	...	...
995	male	group C	high school	standard	none	73	70	65
996	male	group D	associate's degree	free/reduced	completed	85	91	92
997	female	group C	some high school	free/reduced	none	32	35	41
998	female	group C	some college	standard	none	73	74	82
999	male	group A	some college	standard	completed	65	60	62

1000 rows × 8 columns

Information of dataset

In [5]:
```
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
```

Describing Dataset

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

Out[6]:

	math score	reading score	writing score
count	1000.000000	1000.000000	1000.000000
mean	66.396000	69.002000	67.738000
std	15.402871	14.737272	15.600985
min	13.000000	27.000000	23.000000
25%	56.000000	60.000000	58.000000
50%	66.500000	70.000000	68.000000
75%	77.000000	79.000000	79.000000
max	100.000000	100.000000	100.000000

Frequency distribution

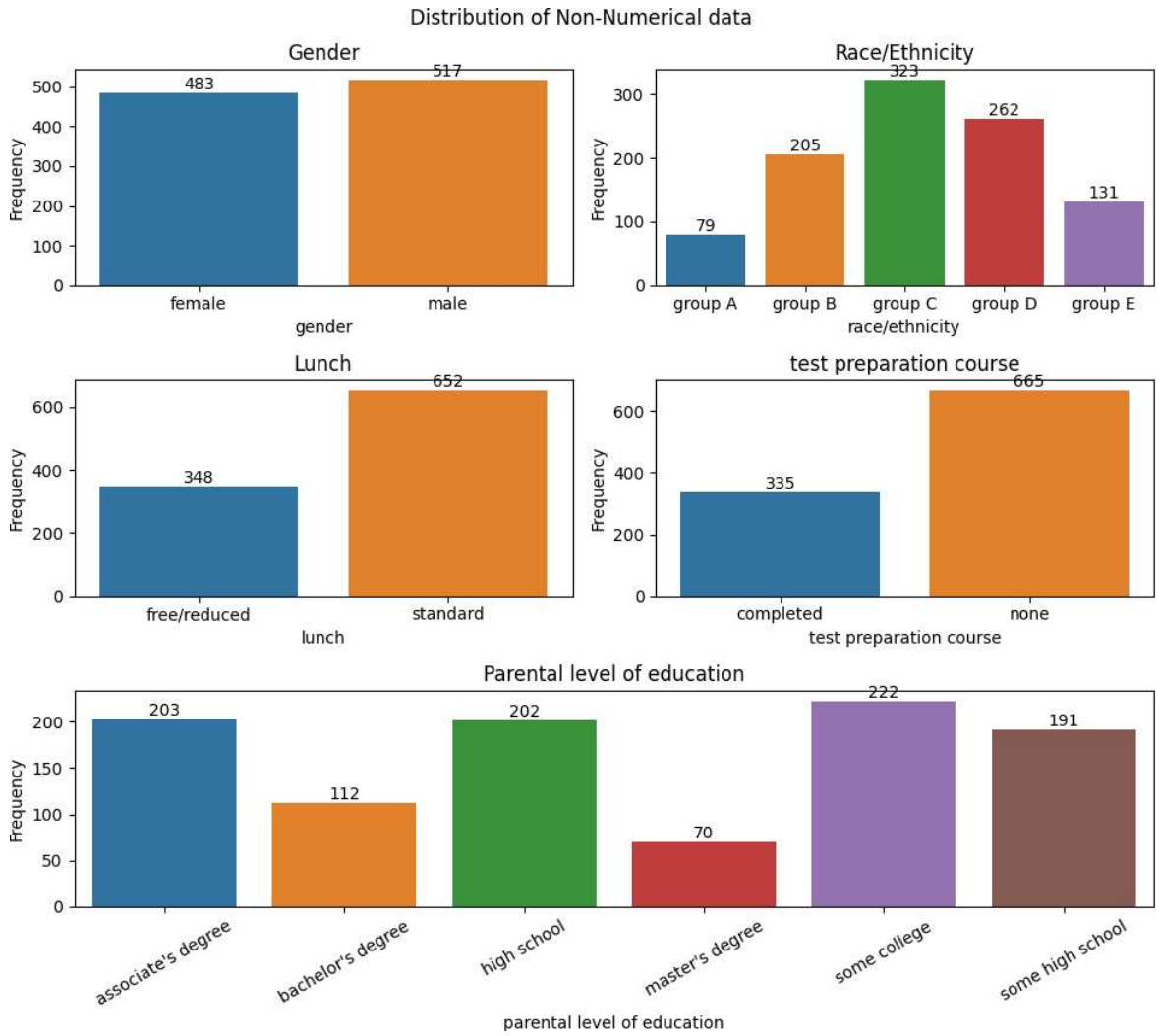
In [7]:
```
frequency = {}
```

In [8]:
```
for column in df.columns:
    frequency[column]  =  df.groupby(by=column)[column].count().to_frame()
    frequency[column].columns = ["Frequency"]
    frequency[column]  =  frequency[column].reset_index()
```

Non-numerical Distribution

In [9]:
```
nonNumericalFrequency = {k: frequency[k] for k in (
    'gender',
    'race/ethnicity',
    'parental level of education',
    'lunch',
    'test preparation course')
}

fig = plt.figure(figsize=(10, 9))
gs = fig.add_gridspec(3,2)
axes1 = fig.add_subplot(gs[0, 0])
axes2 = fig.add_subplot(gs[0, 1])
```

```
axes3 = fig.add_subplot(gs[1, 0])
axes4 = fig.add_subplot(gs[1, 1])
axes5 = fig.add_subplot(gs[2, :])

fig.suptitle("Distribution of Non-Numerical data")

sns.barplot(
    data=nonNumericalFrequency["gender"],
    x="gender",
    y='Frequency',
    ax=axes1
)
axes1.set_title("Gender")
axes1.bar_label(axes1.containers[0])

sns.barplot(
    data=nonNumericalFrequency["race/ethnicity"],
    x="race/ethnicity",
    y='Frequency',
    ax=axes2
)
axes2.set_title("Race/Ethnicity")
axes2.bar_label(axes2.containers[0])

sns.barplot(
    data=nonNumericalFrequency["lunch"],
    x='lunch',
    y='Frequency',
    ax=axes3
)
axes3.set_title("Lunch")
axes3.bar_label(axes3.containers[0])

sns.barplot(
    data=nonNumericalFrequency["test preparation course"],
    x='test preparation course',
    y='Frequency',
    ax=axes4
)
axes4.set_title("test preparation course")
axes4.bar_label(axes4.containers[0])

sns.barplot(
    data=nonNumericalFrequency["parental level of education"],
    x="parental level of education",
    y='Frequency',
    ax=axes5
)
axes5.set_title("Parental level of education")
axes5.bar_label(axes5.containers[0])
plt.xticks(rotation=30)

plt.tight_layout()
plt.show()
```

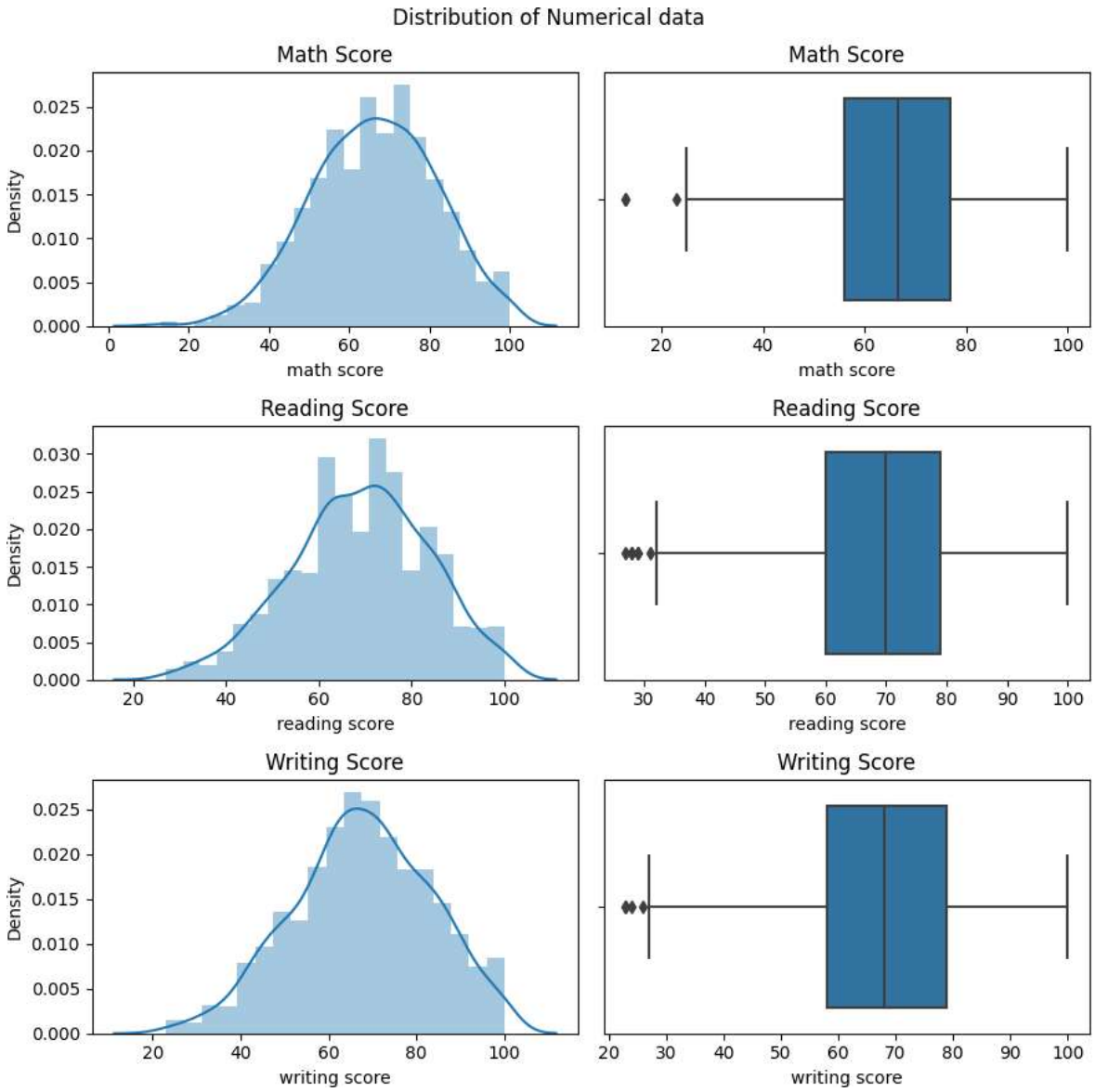


- Male students outnumber female students.
- The majority of the pupils are from Group C, followed by Group D and B.
- More than **50%** of the students dropped out of the course for exam preparation.
- **22.2%** of the parents of the pupils have a college degree, compared to **20.3%** who have an associate's degree, **20.2%** and **7%** with high school diploma and master degree respectively.

Numerical Distribution

In [10]:
```
fig = plt.figure(figsize=(9, 9))

gs = fig.add_gridspec(3,2)
axes1 = fig.add_subplot(gs[0, 0])
axes2 = fig.add_subplot(gs[0, 1])

axes3 = fig.add_subplot(gs[1, 0])
axes4 = fig.add_subplot(gs[1, 1])

axes5 = fig.add_subplot(gs[2, 0])
axes6 = fig.add_subplot(gs[2, 1])

fig.suptitle("Distribution of Numerical data")

# For Math score
sns.distplot(
```

```
    df['math score'],
    ax=axes1
)
axes1.set_title('Math Score')

sns.boxplot(
    df['math score'],
    ax=axes2
)
axes2.set_title('Math Score')

# For reading score
sns.distplot(
    df['reading score'],
    ax=axes3
)
axes3.set_title('Reading Score')

sns.boxplot(
    df['reading score'],
    ax=axes4
)
axes4.set_title('Reading Score')

# For Writing score
sns.distplot(
    df['writing score'],
    ax=axes5
)
axes5.set_title('Writing Score')

sns.boxplot(
    df['writing score'],
    ax=axes6
)
axes6.set_title('Writing Score')

plt.tight_layout()
plt.show()
```

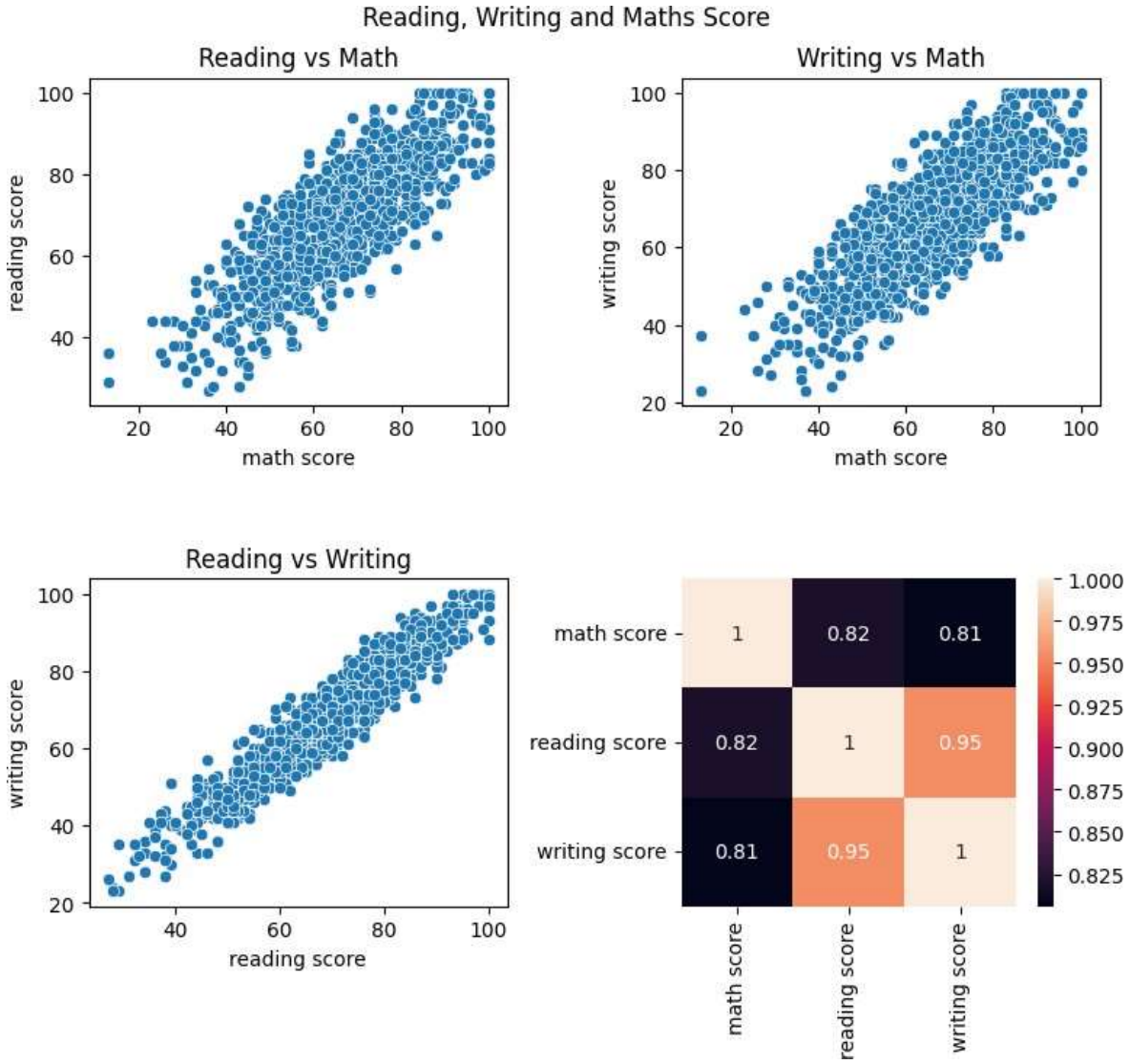


We can observe that the arithmetic, reading, and writing marks follow a normal distribution, with some outliers indicating that certain students did not do well.

Relationship

In [11]:
```
fig = plt.figure(figsize=(8, 10))

gs = fig.add_gridspec(3,2)
axes1 = fig.add_subplot(gs[0, 0])
axes2 = fig.add_subplot(gs[0, 1])
axes3 = fig.add_subplot(gs[1, 0])
axes4 = fig.add_subplot(gs[1, 1])

fig.suptitle("Reading, Writing and Maths Score")

sns.scatterplot(
    y=df["reading score"],
    x=df["math  score"],
    ax=axes1
)
axes1.set_title("Reading vs Math")
```

```
sns.scatterplot(
    y=df["writing  score"],
    x=df["math score"],
    ax=axes2
)
axes2.set_title("Writing vs Math")

sns.scatterplot(
    y=df["writing  score"],
    x=df["reading  score"],
    ax=axes3
)
axes3.set_title("Reading vs Writing")

readingWritingMath = df.loc[:, ["math score", "reading score", "writing score"]]
sns.heatmap(
    data=readingWritingMath.corr(),
    annot=True,
    ax=axes4
)

plt.tight_layout()
plt.show()
```



There is a significant association between all of the factors, thus reading and writing can also help students perform better in math exams.

Grading the marks score

In [12]:
```
def grade(marks):
    if marks >= 80:
        return "O"
    elif marks >= 75 and marks <= 79:
        return "A"
    elif marks >=70 and marks <= 74:
        return "B"
    elif marks >= 60 and marks <= 69:
        return "C"
    elif marks >= 50 and marks <= 59:
        return "D"
    elif marks >= 45 and marks <= 49:
        return "E"
    elif marks >= 40 and marks<= 44:
        return "P"
    else:
        return "F"

df["math grade"] = df["math score"].map(grade)
df["reading grade"] = df["reading score"].map(grade)
df["writing grade"] = df["writing score"].map(grade)
```

In [13]:
```
df[["writing grade", "reading grade", "math grade"]]
```

Out[13]:

	writing grade	reading grade	math grade
0	C	C	C
1	D	D	P
2	D	C	D
3	C	A	A
4	C	B	A
...	...	...	...
995	C	B	B
996	O	O	O
997	P	F	F
998	O	B	B
999	C	C	C

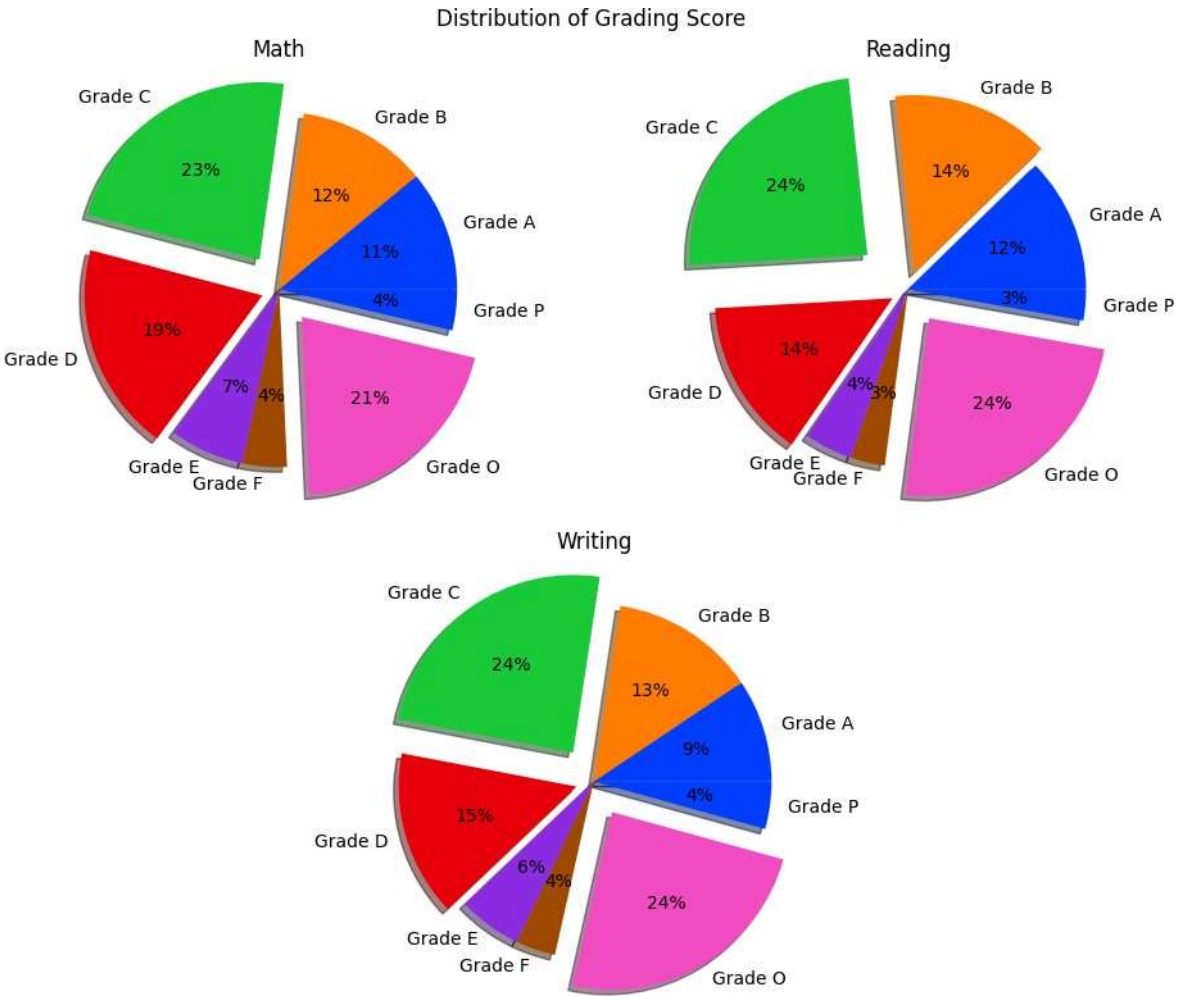
1000 rows × 3 columns

Distribution of Grading Score

In [14]:
```
frequency_grade = {}
```

In [15]:
```
for column in ["math grade", "reading grade", "writing grade"]:
    frequency_grade[column]  =  df.groupby(by=column)[column].count().to_frame()
    frequency_grade[column].columns = ["Frequency"]
    frequency_grade[column]  =  frequency_grade[column].reset_index()
```

In [16]:
```
fig = plt.figure(figsize=(10, 12))

gs = fig.add_gridspec(3,2)
axes1 = fig.add_subplot(gs[0, 0])
axes2 = fig.add_subplot(gs[0, 1])
axes3 = fig.add_subplot(gs[1, :])


fig.suptitle("Distribution of Grading Score")
colors = sns.color_palette('bright')[0:7]
labels = ["Grade A",
          "Grade B",
          "Grade C",
          "Grade D",
          "Grade E",
          "Grade F",
          "Grade O",
          "Grade P"]

axes1.pie(
    frequency_grade['math grade']["Frequency"],
    labels=labels,
    autopct='%.0f%%',
    colors=colors,
    explode=[0, 0, 0.2, 0.1, 0, 0, 0.2, 0],
    shadow=True
)
axes1.set_title('Math')

axes2.pie(
    frequency_grade['reading grade']["Frequency"],
    labels=labels,
    autopct='%.0f%%',
    colors=colors,
    explode=[0, 0.1, 0.3, 0.1, 0, 0, 0.2, 0],
    shadow=True
)
axes2.set_title('Reading')

# For reading score
axes3.pie(
    frequency_grade['writing grade']["Frequency"],
    labels=labels,
    autopct='%.0f%%',
    colors=colors,
    explode=[0, 0, 0.2, 0.1, 0, 0, 0.2, 0],
    shadow=True
)
axes3.set_title('Writing')

plt.tight_layout()
plt.show()
```



- In arithmetic, **21%** of students received an O, **23%** received a C, and **19%** received a D.
- In reading, **24%** of students received O and C grades, while **14%** received B and D grades.
- In writing, **4%** of students received grades of O and C, **15%** received grades of D, and **13%** received grades of B.

Grade performance of male and female in each subject

In [17]:
```
male = df[df["gender"] == "male"][["math grade", "reading grade", "writing grade"]
female = df[df["gender"] == "female"][["math grade", "reading grade", "writing gra

# this function is to find the grade frequency of each subject by gender
# subject_name = name of the subject
# subject_grades = grades score in subject by male and female
def grade_diff(subject_name, subject_grades, gender):
    for i in gender:
        if i == "Male":
            male_subject  =  subject_grades[0].groupby(by=subject_name)[subject_name
        else:
            female_subject  =  subject_grades[1].groupby(by=subject_name)[subject_na

    return {"Male" : male_subject, "Female": female_subject}
```

```
for_math = grade_diff("math grade", [male, female], ["Male", "Female"])
for_reading = grade_diff("reading grade", [male, female], ["Male", "Female"])
for_writing = grade_diff("writing grade", [male, female], ["Male", "Female"])
```

In [18]:

```
fig = make_subplots(
    rows=3, cols=1,
    subplot_titles=("Math",  "Reading",  "Writing"))

fig.add_trace(go.Bar(
    x=for_math["Male"].index,
    y=for_math["Male"]["math  grade"],
    name="Male"),
    row=1, col=1)

fig.add_trace(go.Bar(
    x=for_math["Female"].index,
    y=for_math["Female"]["math  grade"],
    name="Female"),
    row=1, col=1)

fig.add_trace(go.Bar(
    x=for_reading["Male"].index,
    y=for_reading["Male"]["reading grade"],
    name="Male"),
    row=2, col=1)

fig.add_trace(go.Bar(
    x=for_reading["Female"].index,
    y=for_reading["Female"]["reading grade"],
    name="Female"),
    row=2, col=1)

fig.add_trace(go.Bar(
    x=for_writing["Male"].index,
    y=for_writing["Male"]["writing grade"],
    name="Male"),
    row=3, col=1)

fig.add_trace(go.Bar(
    x=for_writing["Female"].index,
    y=for_writing["Female"]["writing grade"],
    name="Female"),
    row=3, col=1)

fig.update_layout(
    height=800, width=500,
    title_text="Grade Performace by Gender in Different Subject",
    coloraxis=dict(colorscale='RdBu'))

# Update xaxis properties
fig.update_xaxes(title_text="Grade",  row=1,  col=1)
fig.update_xaxes(title_text="Grade",  row=2,  col=1)
fig.update_xaxes(title_text="Grade",  row=3,  col=1)

# Update yaxis properties
fig.update_yaxes(title_text="Count",  row=1,  col=1)
fig.update_yaxes(title_text="Count",  row=2,  col=1)
fig.update_yaxes(title_text="Count",  row=3,  col=1)

fig.show()
```

- In math, **12.8%** of male students received an O mark, **12.3%** received a C grade, and only **1.1%** failed, compared to **2.9%** of female students who failed the math test, **7.8%** received an O grade, **12.8%** received a D grade, and **10.8%** received a C grade, suggesting that males perform better than females.
- In reading, **9.5%** of male students received an O mark, while **9.0%** failed, however **14.8%** of female students received an O grade and just **2.2%** failed the exam, demonstrating that females perform better than males.
- In reading, **8.2%** of male students received an O mark, while **2.7%** failed, however **16.0%** of female students received an O grade and only **9.0%** failed the exam,

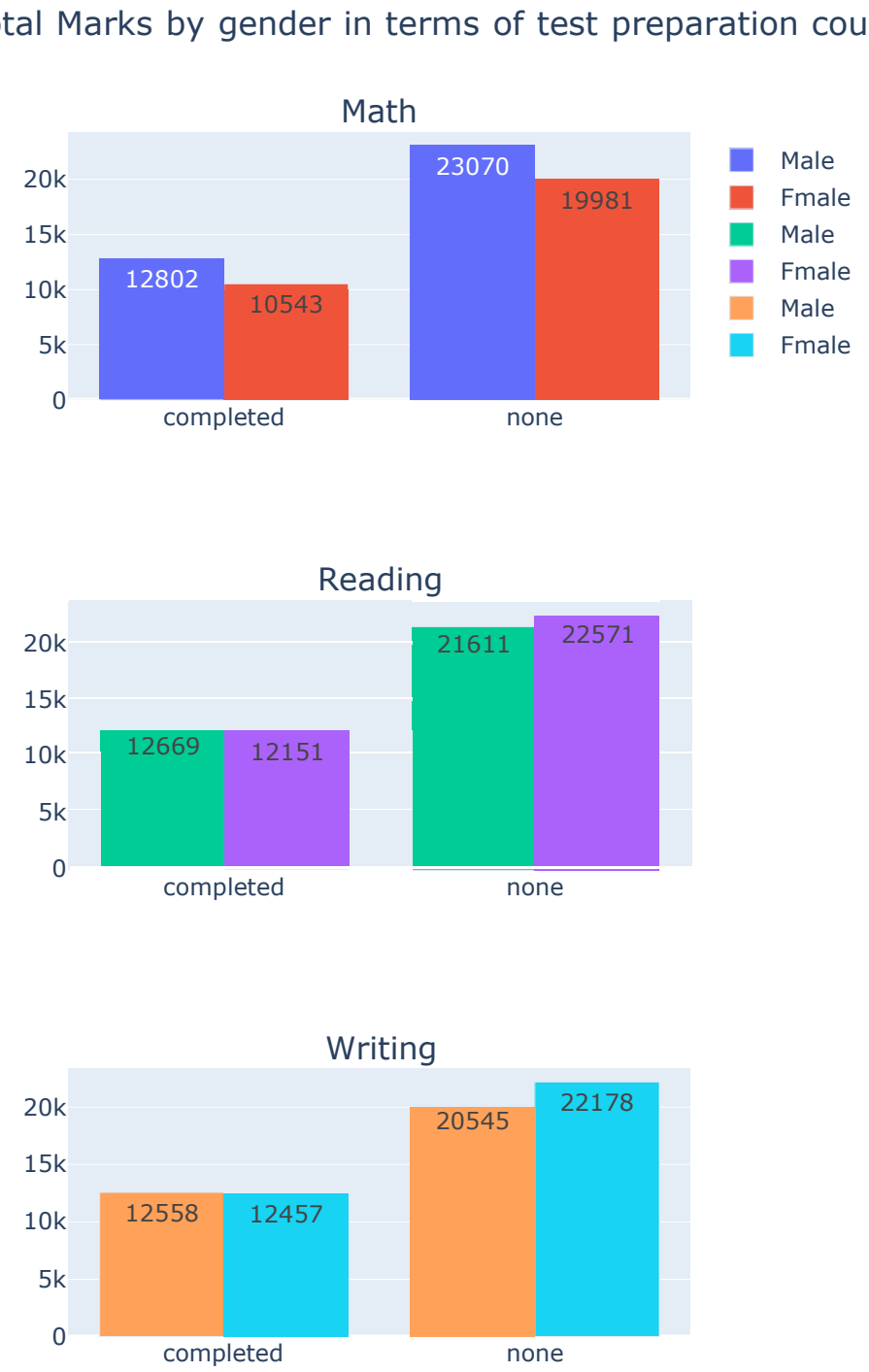
demonstrating that females perform better than male.

Marks score by gender in terms of test preparation course

In [19]:
```
total_score = df.groupby("test preparation course")["math score", "reading score",

test_pre_male = df[df["gender"] == "male"][
    ["test preparation course",
     "math score", "reading score",
     "writing score"
    ]
]

test_pre_female = df[df["gender"] == "female"][
    ["test preparation course",
     "math score", "reading score",
     "writing score"
    ]
]

forTestPreMale = test_pre_male.groupby("test preparation course")["math score",
                                                                  "reading score",
                                                                  "writing score"]
forTestPreFemale = test_pre_female.groupby("test preparation course")["math score"
                                                                      "reading sco
                                                                      "writing sco
```

In [20]:
```
fig = make_subplots(
    rows=3, cols=1,
    subplot_titles=("Math","Reading", "Writing")
)

fig.add_trace(go.Bar(
    x=forTestPreMale.index,
    y=forTestPreMale["math score"],
    text=forTestPreMale["math score"],
    textposition="auto",
    name="Male"),
    row=1, col=1)

fig.add_trace(go.Bar(
    x=forTestPreMale.index,
    y=forTestPreFemale["math score"],
    text=forTestPreFemale["math score"],
    textposition="auto",
    name="Fmale"),
    row=1, col=1)

fig.add_trace(go.Bar(
    x=forTestPreMale.index,
    y=forTestPreMale["reading score"],
    text=forTestPreMale["reading score"],
    textposition="auto",
    name="Male"),
    row=2, col=1)
```

```
fig.add_trace(go.Bar(
    x=forTestPreMale.index,
    y=forTestPreFemale["reading  score"],
    text=forTestPreFemale["reading score"],
    textposition="auto",
    name="Fmale"),
    row=2, col=1)

fig.add_trace(go.Bar(
    x=forTestPreMale.index,
    y=forTestPreMale["writing  score"],
    name="Male",
    text=forTestPreMale["writing score"],
    textposition="auto"),
    row=3, col=1)

fig.add_trace(go.Bar(
    x=forTestPreMale.index,
    y=forTestPreFemale["writing  score"],
    name="Fmale",
    text=forTestPreFemale["writing score"],
    textposition="auto"),
    row=3, col=1)

fig.update_layout(
    height=800, width=500,
    title_text="Total Marks by gender in terms of test preparation course",
    coloraxis=dict(colorscale='RdBu'))
```



How effective is the test preparation course?

In [21]:
```
# ANOVA test to compared the means

from scipy import stats

def  anova_test(subject_marks):
    completed_marks = df[df["test preparation course"] == "completed"][subject_mar
    none_complete_marks  =  df[df["test  preparation  course"]  ==  "none"][subject_mark
```

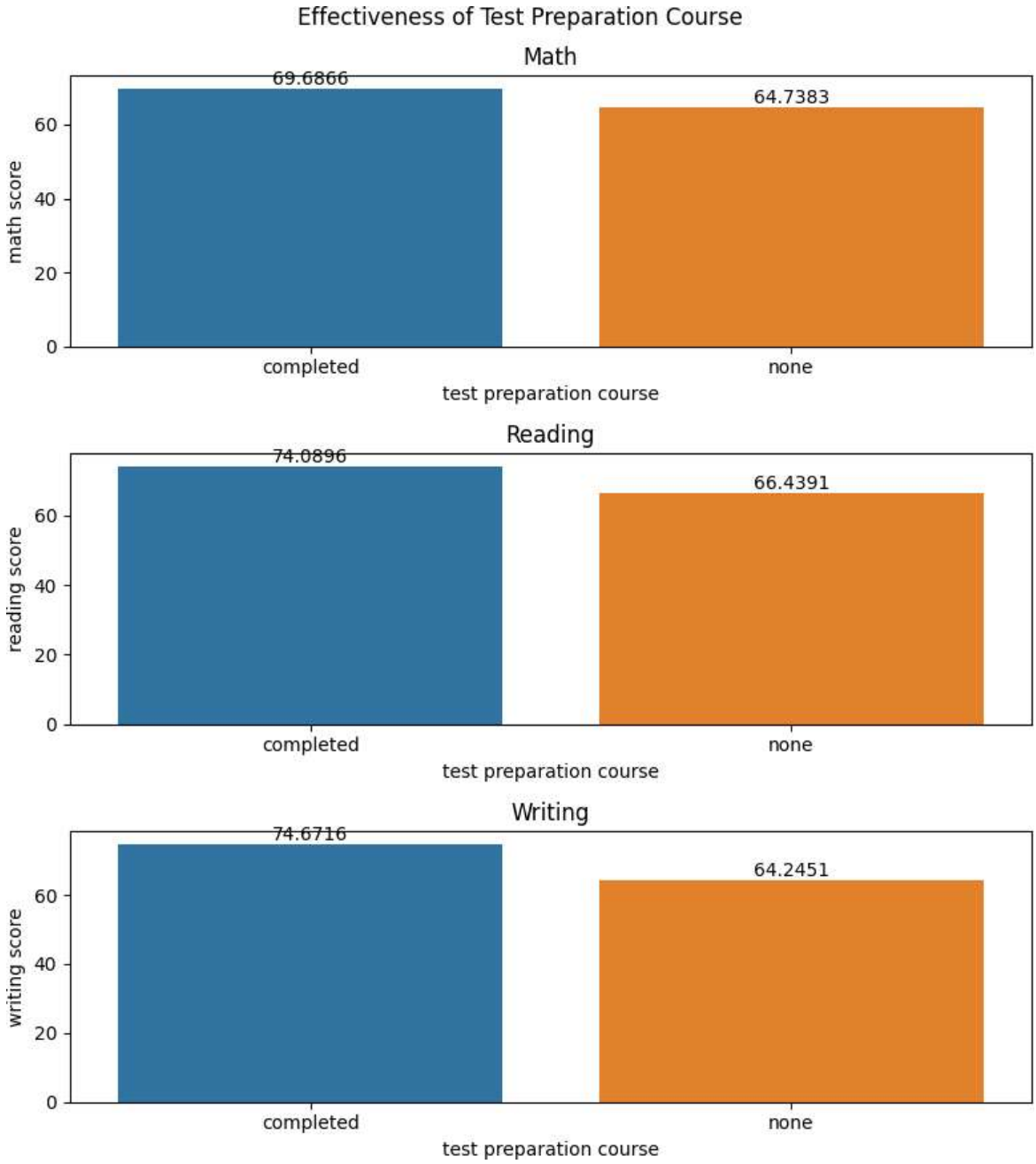
```
    F, p_value = stats.f_oneway(completed_marks, none_complete_marks)

    if p_value < .05:
        return f'There is a significant effect of test preparation course'
    else:
        return 'There is no significant difference between groups'
```

In [22]:
```
for i in ["math score", "writing score", "reading score"]:
    print(f"{anova_test([i])} for {i}")
```

```
There is a significant effect of test preparation course for math score
There is a significant effect of test preparation course for writing score
There is a significant effect of test preparation course for reading score
```

When we compared the means using the ANOVA test, we found a significant difference, indicating that the test preparation course was helpful in exam.

In [23]:
```
merge_data = df.groupby("test preparation course")["math score", "reading score",
x = merge_data["test preparation course"]

fig = plt.figure(figsize=(8, 9))

fig.suptitle("Effectiveness of Test Preparation Course")

gs = fig.add_gridspec(3,2)
axes1 = fig.add_subplot(gs[0, :])
axes2 = fig.add_subplot(gs[1, :])
axes3 = fig.add_subplot(gs[2, :])

sns.barplot(
    data=merge_data,
    x=x,
    y=merge_data["math score"],
    ax=axes1)
axes1.set_title("Math")
axes1.bar_label(axes1.containers[0])

sns.barplot(
    data=merge_data,
    x=x,
    y=merge_data["reading score"],
    ax=axes2)
axes2.set_title("Reading")
axes2.bar_label(axes2.containers[0])

sns.barplot(
    data=merge_data,
    x=x,
    y=merge_data["writing score"],
    ax=axes3)

axes3.set_title("Writing")
axes3.bar_label(axes3.containers[0])

plt.tight_layout()
plt.show()
```



Do the educational level of parent have an effect in scoring marks in exam?

In [24]:
```
def  anova_parent_test(parent_edu):
    F, p_value = stats.f_oneway(parent_edu[0], parent_edu[1], parent_edu[2],parent
    if p_value < .05:
        return f'There is a significant effect of parent level of education'
    else:
        return 'There is no significant difference between groups.'
```

In [25]:
```
edu_lvl = {}
parent_edu_lvl = df["parental level of education"].unique()
for i in ["math score", "reading score", "writing score"]:
    for j in parent_edu_lvl:
        edu_lvl[i] = [df[df["parental level of education"] == parent_edu_lvl[0]][i
                      df[df["parental level of education"] == parent_edu_lvl[1]][i
                      df[df["parental level of education"] == parent_edu_lvl[2]][i]
```

```
                    df[df["parental level of education"] == parent_edu_lvl[3]][i]
                    df[df["parental level of education"] == parent_edu_lvl[4]][i]
                    df[df["parental level of education"] == parent_edu_lvl[5]][i]
                    ]
```

In [26]:
```
for i in edu_lvl:
    print(f"{anova_parent_test(edu_lvl[i])} in {i}")
```

```
There is a significant effect of parent level of education in math score
There is a significant effect of parent level of education in reading score
There is a significant effect of parent level of education in writing score
```

We may say that the parent's degree of education influences the student's exam performance.

In [27]:
```
parentStudentScore = df.groupby(by='parental level of education')["math score", "r
parentStudentScore = parentStudentScore.round(2)
parentStudentScoreTotal = df.groupby(by='parental level of education')["math score
x = parentStudentScore.index

fig = make_subplots(
    rows=3, cols=1,
    subplot_titles=(
         "Mean Math Score",
         "Mean Reading Score",
         "Mean Writing Score")
)

fig.add_trace(go.Bar(
    x=x,
    y=parentStudentScore["math score"],
    text=parentStudentScore["math score"],
    textposition="auto",
    showlegend=False),
    row=1, col=1)

fig.add_trace(go.Bar(
    x=x,
    y=parentStudentScore["reading score"],
    text=parentStudentScore["reading score"],
    textposition="auto",
    showlegend=False),
    row=2, col=1)

fig.add_trace(go.Bar(
    x=x,
    y=parentStudentScore["writing score"],
    text=parentStudentScore["writing score"],
    textposition="auto",
    showlegend=False),
    row=3, col=1)

fig.update_layout(
    height=800, width=500,
    title_text="Effectiveness of Parent Level of Education in Exam",
    coloraxis=dict(colorscale='RdBu'))
```

In [28]:
```
# copying dataset for modeling
data = df
```

Data Preprocessing

In [29]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [30]:
```
data['gender']=le.fit_transform(data['gender'])
data['race/ethnicity']=le.fit_transform(data['race/ethnicity'])
```

```
data['parental level of education']=le.fit_transform(data['parental level of educa
data['lunch']=le.fit_transform(data['lunch'])
```

In [31]:
```
data['test  preparation  course']=le.fit_transform(data['test  preparation  course'])
```

In [32]:
```
data.head()
```

Out[32]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	math grade	readin grad
0	1	0	2	1	0	67	67	63	C	
1	0	3	5	0	1	40	59	55	P	D
2	1	4	4	0	1	59	60	50	D	
3	1	1	2	1	1	77	78	68	A	
4	1	4	0	1	0	78	73	68	A	

Creating a new dependent feature to predict the total scores using ML model

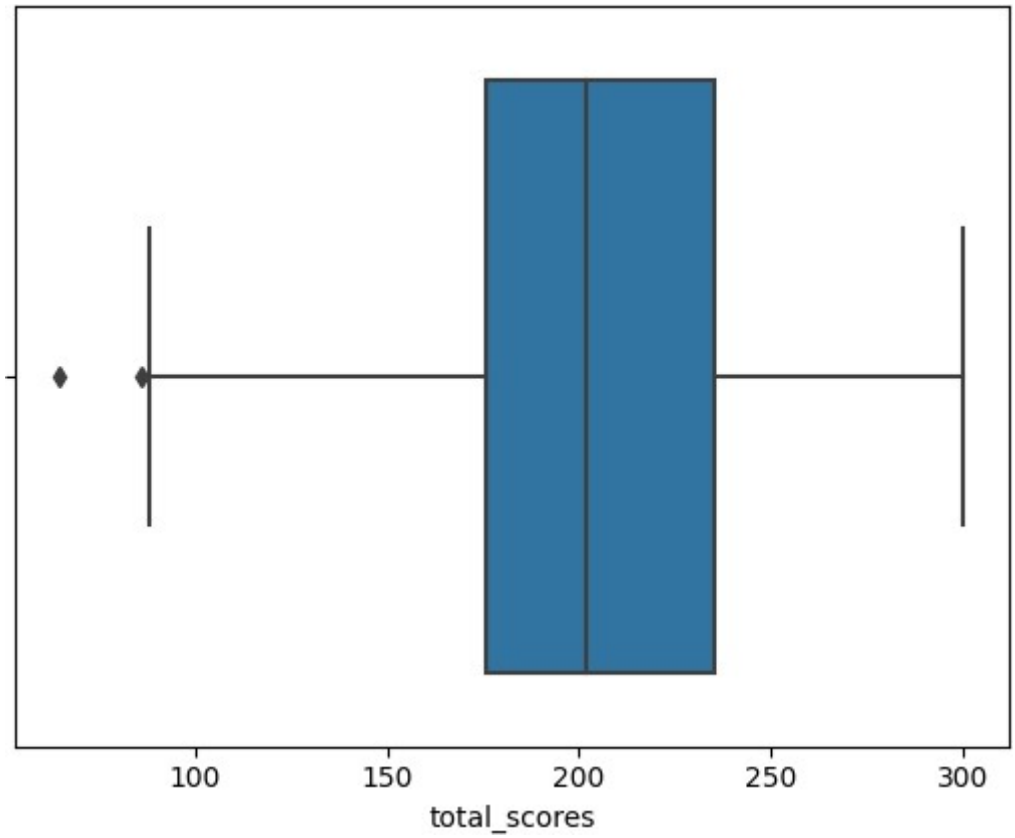
In [33]:
```
data['total_scores'] = data['math  score']+data['reading  score']+data['writing  scor
data
```

Out[33]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	math grade	read gr
0	1	0	2	1	0	67	67	63	C	
1	0	3	5	0	1	40	59	55	P	
2	1	4	4	0	1	59	60	50	D	
3	1	1	2	1	1	77	78	68	A	
4	1	4	0	1	0	78	73	68	A	
...	...	...	...	...	...	...	...	...	...	...
995	1	2	2	1	1	73	70	65	B	
996	1	3	0	0	0	85	91	92	O	
997	0	2	5	0	1	32	35	41	F	
998	0	2	4	1	1	73	74	82	B	
999	1	0	4	1	0	65	60	62	C	

1000 rows × 12 columns

In [34]:
```
sns.boxplot(data['total_scores'])
plt.show()
```



Splitting the dataset into independent and dependent variables

In [35]:
```
X=df[['gender','race/ethnicity','parental level of education','lunch','test prepar
y=df['total_scores']
```

In [36]:
```
X
```

Out[36]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	1	0	2	1	0	67	67	63
1	0	3	5	0	1	40	59	55
2	1	4	4	0	1	59	60	50
3	1	1	2	1	1	77	78	68
4	1	4	0	1	0	78	73	68
...	...	...	...	...	...	...	...	...
995	1	2	2	1	1	73	70	65
996	1	3	0	0	0	85	91	92
997	0	2	5	0	1	32	35	41
998	0	2	4	1	1	73	74	82
999	1	0	4	1	0	65	60	62

1000 rows × 8 columns

In [37]:
```
y
```

Out[37]:
```
0      197
1      154
2      169
3      223
4      219
      ...
995    208
996    268
997    108
998    229
999    187
Name: total_scores, Length: 1000, dtype: int64
```

Modelling

Splitting the dataset for training and testing the Model

In [38]:
```
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,random_state=0)
```

In [39]:
```
X_train
```

Out[39]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
105	0	3	1	1	1	70	78	82
68	1	1	2	1	1	79	64	60
479	1	2	0	0	1	62	43	45
399	0	0	4	0	0	53	70	69
434	1	2	0	1	1	67	63	66
...	...	...	...	...	...	...	...	...
835	0	1	5	1	1	76	81	79
192	1	4	5	0	1	68	62	61
629	1	2	2	1	1	72	59	54
559	1	4	2	0	1	56	48	36
684	1	2	0	0	1	69	77	70

700 rows × 8 columns

In [40]:
```
y_train
```

Out[40]:
```
105    230
68     203
479    150
399    192
434    196
      ...
835    236
192    191
629    185
559    140
684    216
Name: total_scores, Length: 700, dtype: int64
```

In [41]:
```
X_train.shape
```

Out[41]:
```
(700, 8)
```

In [42]:
```
X_test.shape
```

Out[42]:
```
(300, 8)
```

In [43]:
```
y_test.shape
```

Out[43]:
```
(300,)
```

Using the Decission Tree Regression

In [44]:
```
from sklearn.tree import DecisionTreeRegressor
dec_reg=DecisionTreeRegressor(max_depth=5)
```

In [45]:
```
dec_reg.fit(X_train,y_train)
```

Out[45]:
```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=5)
```

First 10 sample size of prediction score.

In [46]:
```
y_pred_dec_reg=dec_reg.predict(X_train)
y_pred_dec_reg[:30]
```

Out[46]:
```
array([234.13043478, 187.95652174, 147.38888889, 193.08695652,
       194.06521739, 210.95      , 231.6       , 223.86538462,
       205.5625    , 175.16666667, 101.53333333, 194.06521739,
       223.86538462, 242.8       , 223.3       , 139.93548387,
       193.08695652, 231.6       , 158.76923077, 179.18181818,
       260.46774194, 212.44      , 223.3       , 175.16666667,
       210.95      , 175.16666667, 204.70588235, 231.6       ,
       194.06521739, 245.8       ])
```

In [47]:
```
print("Score on Test Data : ",dec_reg.score(X_test,y_test))
print("Score on Training Data : ",dec_reg.score(X_train,y_train))
```

```
Score on Test Data :  0.9673170377927085
Score on Training Data :  0.9830698619940315
```

Find and Discussion

i) Males outperform females in math, whereas females outperform males in reading and writing.

ii) The were more male participants than female

iii) Students who have taken the exam preparation course outperform those who have not.

iv) Students' reading and writing abilities might help them do better in arithmetic

v) The degree of education of the parent has an impact on the student's ability to perform well.

Conclusion

The research goal was to understand how students' marks in each topic rely on the completion of a test preparation course and parent education, as well as to see the difference between male and female in each subject and assess the accuracy of the predicted values.