In [607…

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error as mae
from sklearn.metrics import r2_score as r2
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
```

In [608…

```
student = pd.read_csv("/home/kasagg21/Downloads/archive (2)/StudentsPerformance.csv")
student.head()
```

Out[608…

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

In [609…

```
student.columns
```

Out[609…

```
Index(['gender', 'race/ethnicity', 'parental level of education', 'lunch',
       'test preparation course', 'math score', 'reading score',
       'writing score'],
      dtype='object')
```

In [610…

```
student.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 [611…

```
student.isna().sum()
```

Out[611…

```
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
```

In [612…

```
student.select_dtypes('object').nunique()
```

Out[612…

```
gender                         2
race/ethnicity                 5
parental level of education    6
lunch                          2
test preparation course        2
dtype: int64
```

In [613…

```
print("Categories in 'gender' variable: ",end=" ")
print(student['gender'].unique())
print("Categories in 'race/ethnicity' variable: ",end=" ")
print(student['race/ethnicity'].unique())
print("Categories in 'parental level of education' variable: ",end=" ")
print(student['parental level of education'].unique())
print("Categories in 'lunch' variable: ",end=" ")
print(student['lunch'].unique())
print("Categories in 'test preparation course' variable: ",end=" ")
print(student['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 college' "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 [614…

```
student.describe()
```

Out[614…

	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

In [615…
```
student['Total Score']=student['math score']+student['reading score']+student['writing score']
```

In [616…
```
def result(TS,MS,WS,RS ):
    if(TS>150 and MS>40 and WS>40 and RS>40):
        return 'P'
    else:
        return 'F'
```

In [617…
```
student['Pass/Fail']=student.apply(lambda x: result(x['Total Score'],x['math score'],x['writing score'],x['reading score']),axis = 1 )
```

In [618…
```
student.head()
```

Out[618…

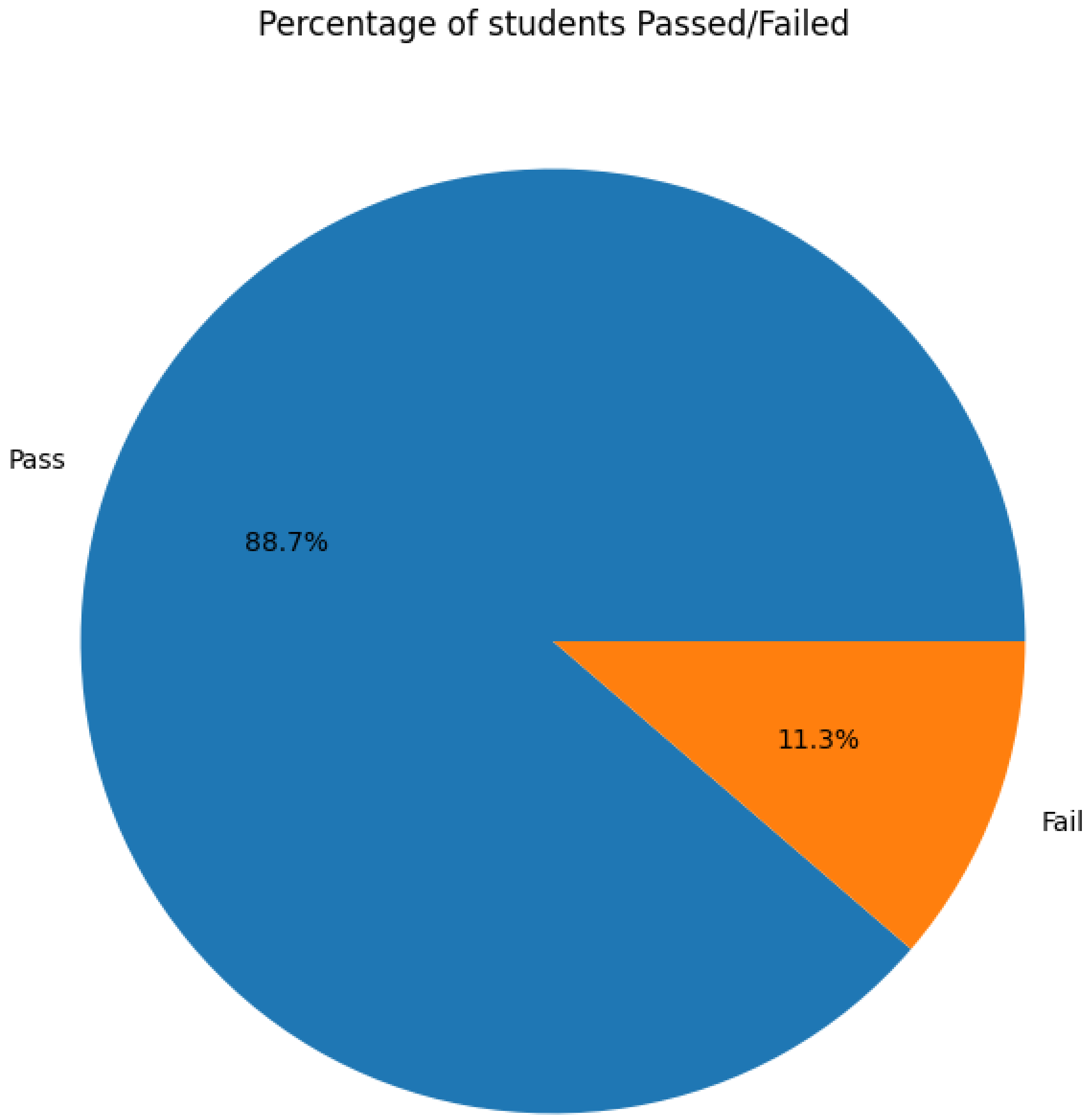
	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Total Score	Pass/Fail
0	female	group B	bachelor's degree	standard	none	72	72	74	218	P
1	female	group C	some college	standard	completed	69	90	88	247	P
2	female	group B	master's degree	standard	none	90	95	93	278	P
3	male	group A	associate's degree	free/reduced	none	47	57	44	148	F
4	male	group C	some college	standard	none	76	78	75	229	P

In [619…
```
student['Pass/Fail'].value_counts()
```

Out[619…
```
Pass/Fail
P    887
F    113
Name: count, dtype: int64
```

In [620…
```
plt.pie(student['Pass/Fail'].value_counts(),labels=['Pass','Fail'],autopct='%1.1f%%')
plt.title('Percentage of students Passed/Failed')
```

Out[620…
```
Text(0.5, 1.0, 'Percentage of students Passed/Failed')
```

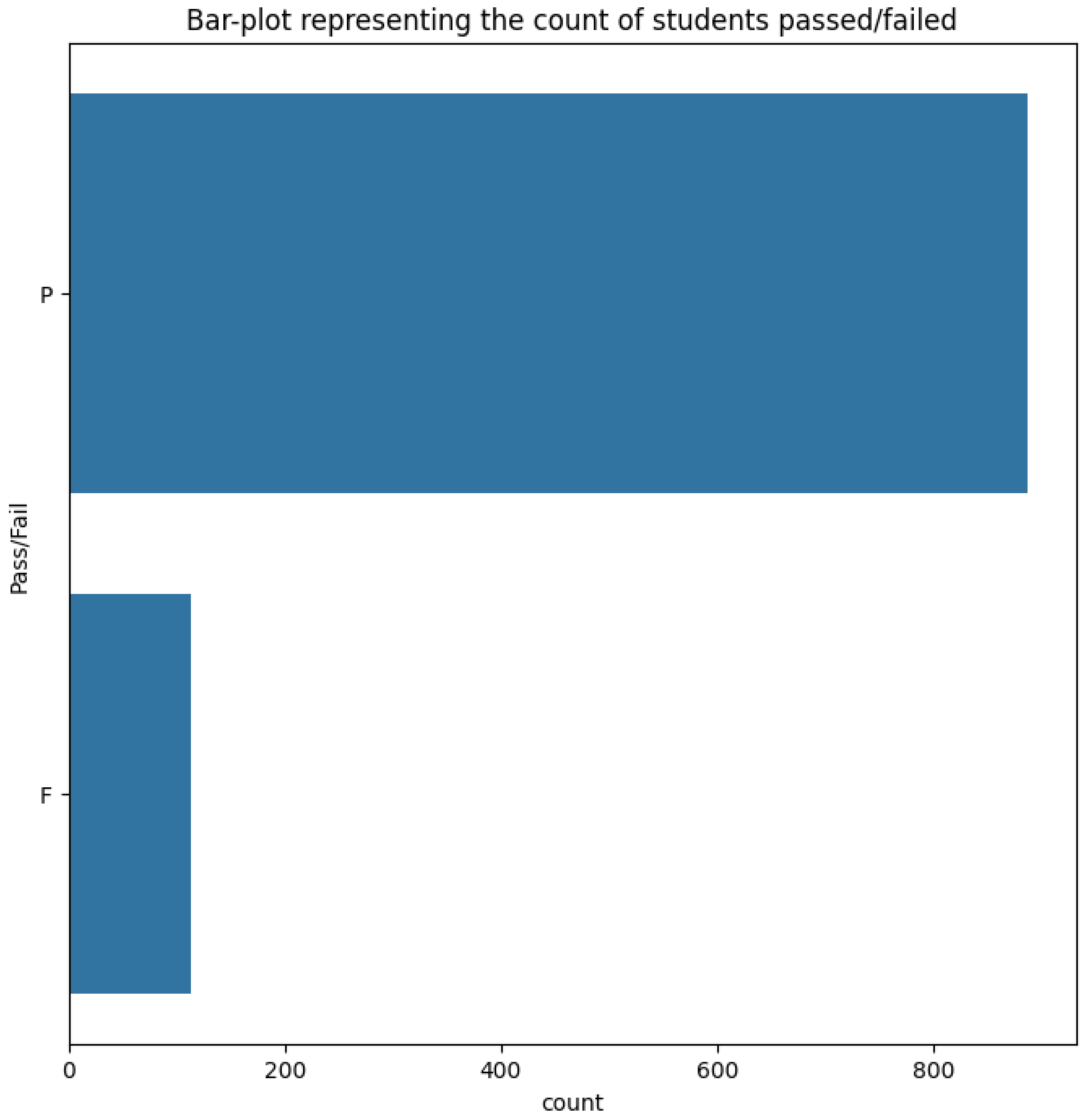


In [621…

```
sns.countplot(student['Pass/Fail'])
plt.title('Bar-plot representing the count of students passed/failed')
```

Out[621…

```
Text(0.5, 1.0, 'Bar-plot representing the count of students passed/failed')
```



In [622…
```
student['gender'].value_counts()
```

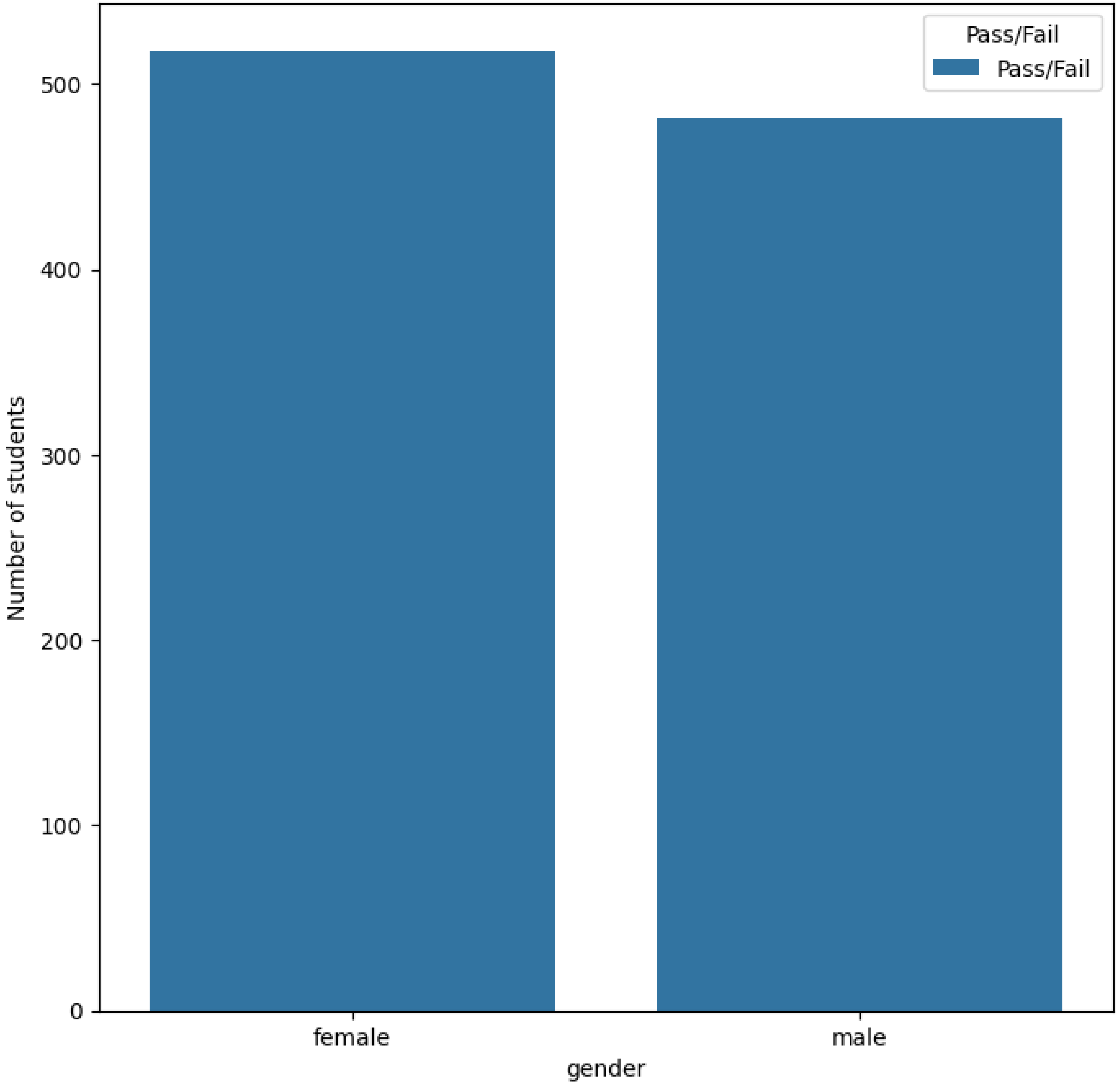
Out[622…
```
gender
female    518
male      482
Name: count, dtype: int64
```

In [623…
```
print("Percentage of female students passed: {0:.2f}%"
    .format((student[(student['gender']=='female') & (student['Pass/Fail']=='P')].shape[0]/student[student['gender']=='female'].shape[0])*100))

print("Percentage of male students passed: {0:.2f}%"
    .format((student[(student['gender']=='male') & (student['Pass/Fail']=='P')].shape[0]/student[student['gender']=='male'].shape[0])*100))
```

```
Percentage of female students passed: 90.73%
Percentage of male students passed: 86.51%
```

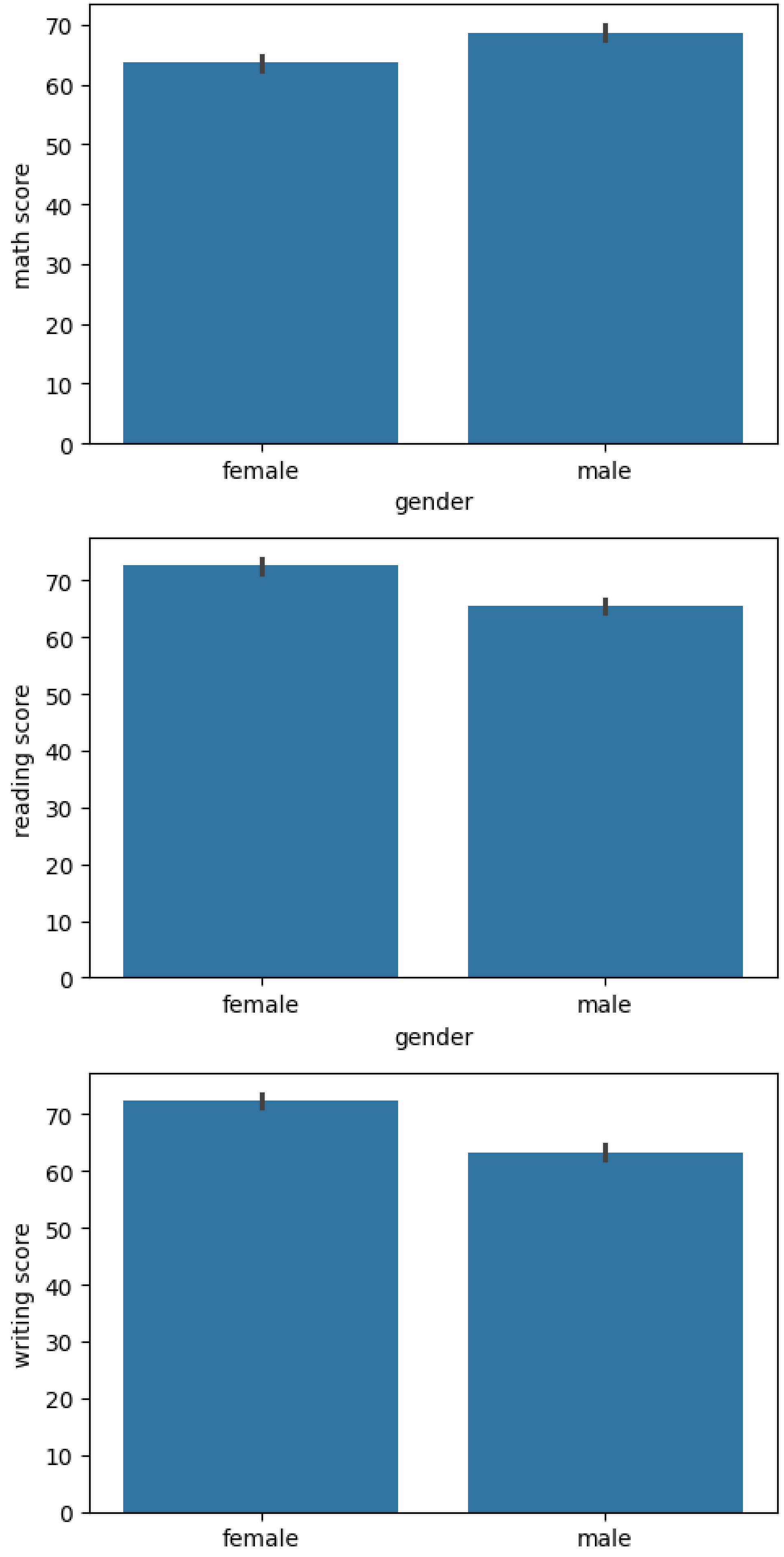
In [624…

```
import seaborn as sns
import matplotlib.pyplot as plt

melted_data = student.melt(id_vars="gender", value_vars="Pass/Fail", var_name="Pass/Fail")

sns.countplot(data=melted_data, x="gender", hue="Pass/Fail")
plt.ylabel("Number of students")
plt.show()
```



In [625…

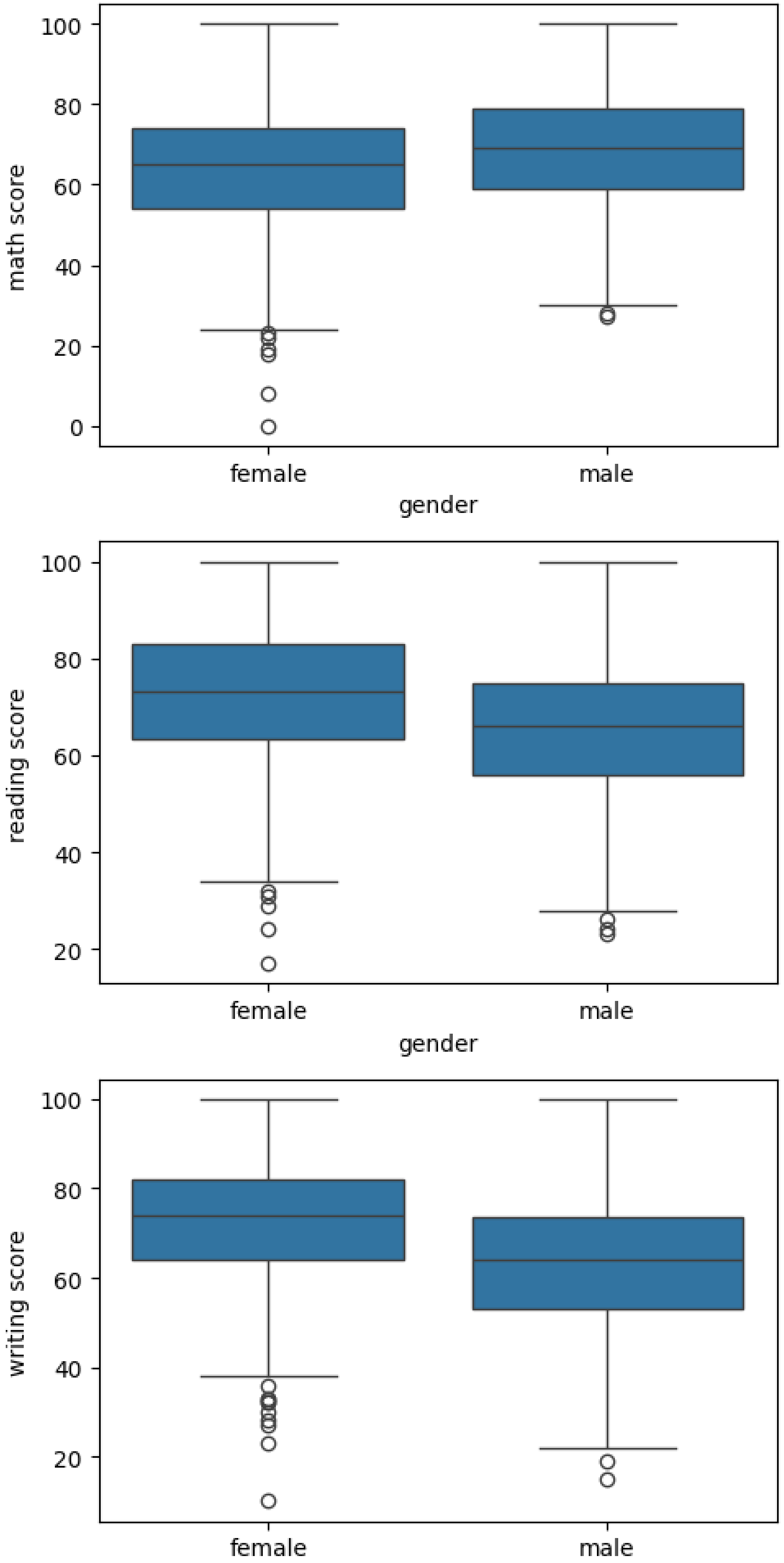
```
fig,ax = plt.subplots(3,1, figsize = (5,10))
sns.barplot(x=student['gender'],y=student['math score'], ax=ax[0], linewidth=2.5)
sns.barplot(x=student['gender'],y=student['reading score'], ax=ax[1],linewidth=2.5)
sns.barplot(x=student['gender'],y=student['writing score'], ax=ax[2],linewidth=2.5)
plt.tight_layout()
```


math score
0
10
20
30
40
50
60
70
female
male
gender
reading score
0
10
20
30
40
50
60
70
female
male
gender
writing score
0
10
20
30
40
50
60
70
female
male

gender

In [626…

```
fig,ax = plt.subplots(3,1, figsize = (5,10))
sns.boxplot(x=student['gender'],y=student['math score'],ax=ax[0])
sns.boxplot(x=student['gender'],y=student['reading score'],ax=ax[1])
sns.boxplot(x=student['gender'],y=student['writing score'],ax=ax[2])
plt.tight_layout()
```


math score
0
20
40
60
80
100
female
male
gender
reading score
20
40
60
80
100
female
male
gender
writing score
20
40
60
80
100
female
male

gender

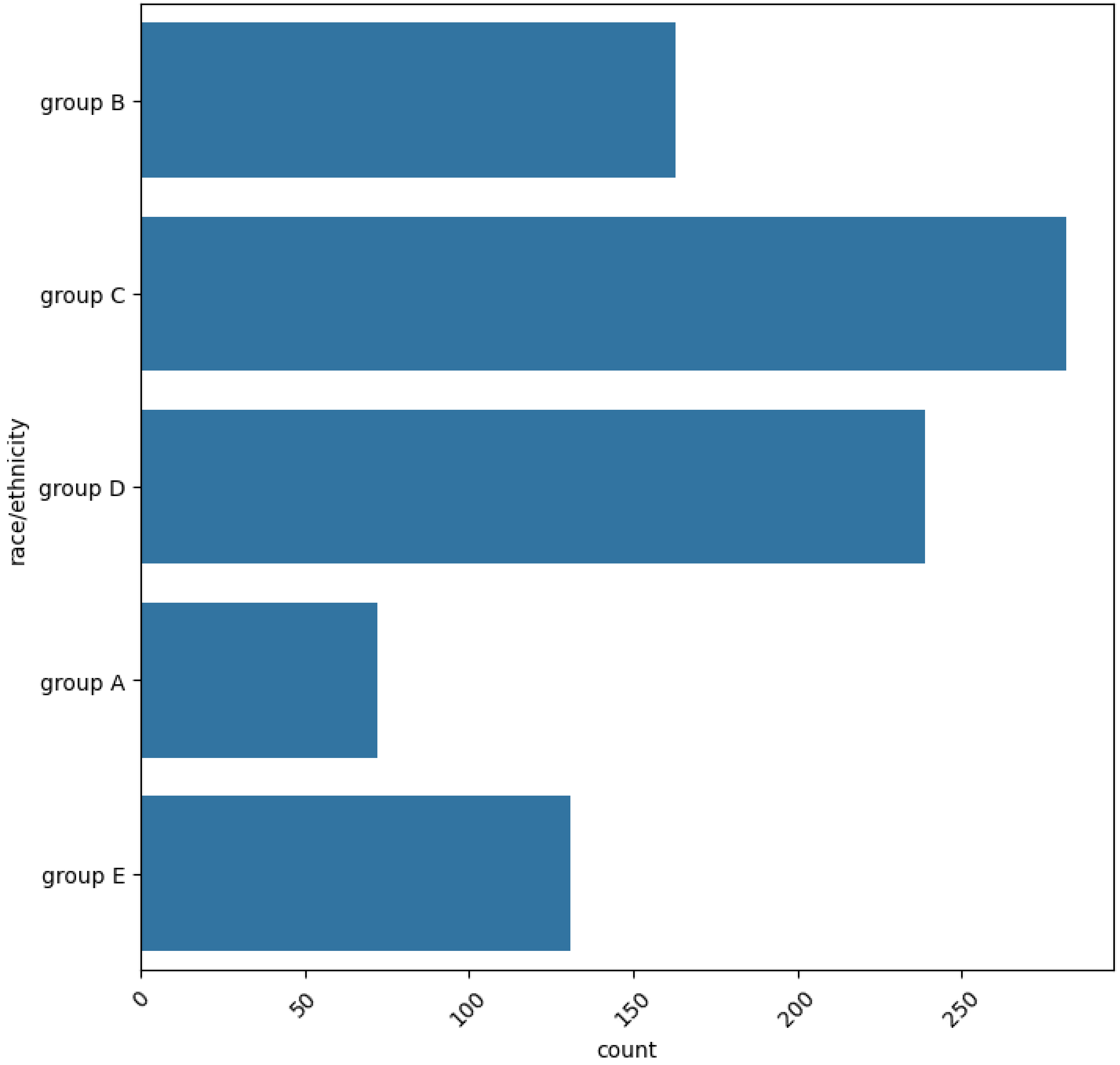
```
In [627… student['race/ethnicity'].value_counts()
```

```
Out[627… race/ethnicity
         group C    319
         group D    262
         group B    190
         group E    140
         group A     89
         Name: count, dtype: int64
```

```
In [628… print("The number of students passed across various race/ethnic group : ")
         print(student['race/ethnicity'].loc[student['Pass/Fail']=='P'].value_counts())
         sns.countplot(student['race/ethnicity'].loc[student['Pass/Fail']=='P'])
         plt.xticks(rotation = 45)
```

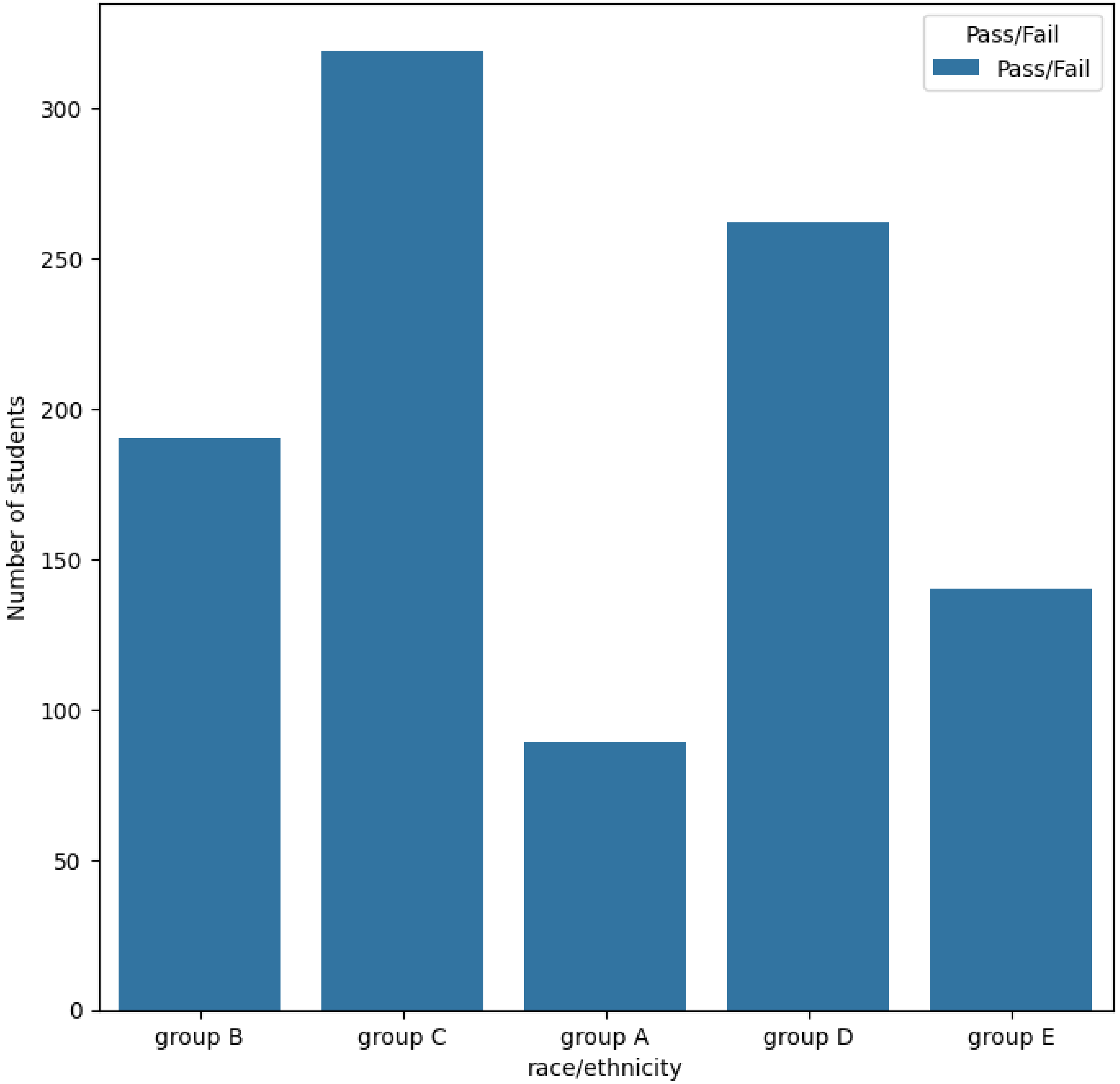
```
The number of students passed across various race/ethnic group :
race/ethnicity
group C    282
group D    239
group B    163
group E    131
group A     72
Name: count, dtype: int64
```

```
Out[628… (array([  0.,  50., 100., 150., 200., 250., 300.]),
          [Text(0.0, 0, '0'),
           Text(50.0, 0, '50'),
           Text(100.0, 0, '100'),
           Text(150.0, 0, '150'),
           Text(200.0, 0, '200'),
           Text(250.0, 0, '250'),
           Text(300.0, 0, '300')])
```



In [629…

```
import seaborn as sns
import matplotlib.pyplot as plt

melted_data = student.melt(id_vars="race/ethnicity", value_vars="Pass/Fail", var_name="Pass/Fail")

sns.countplot(data=melted_data, x="race/ethnicity", hue="Pass/Fail")
plt.ylabel("Number of students")
plt.show()
```



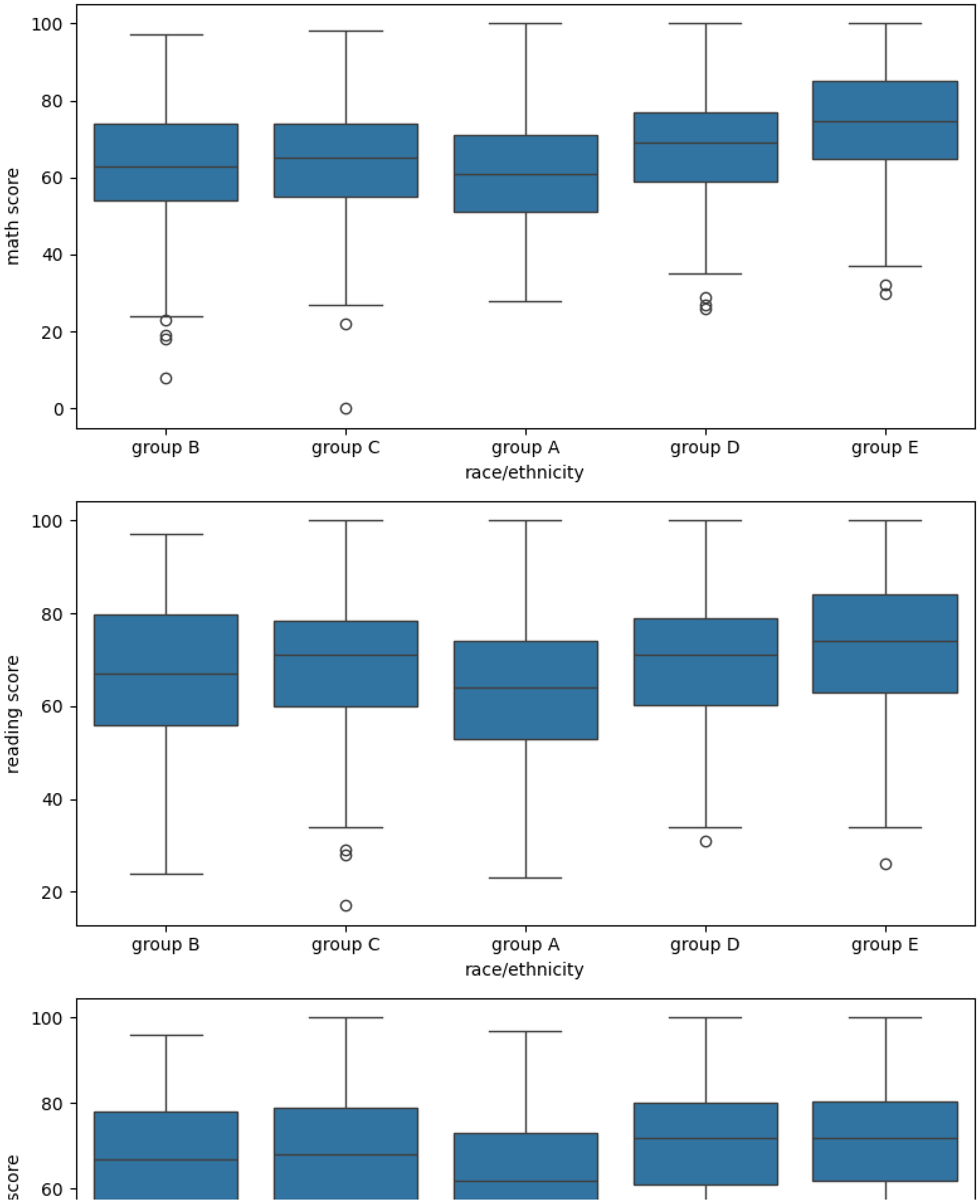
In [630…

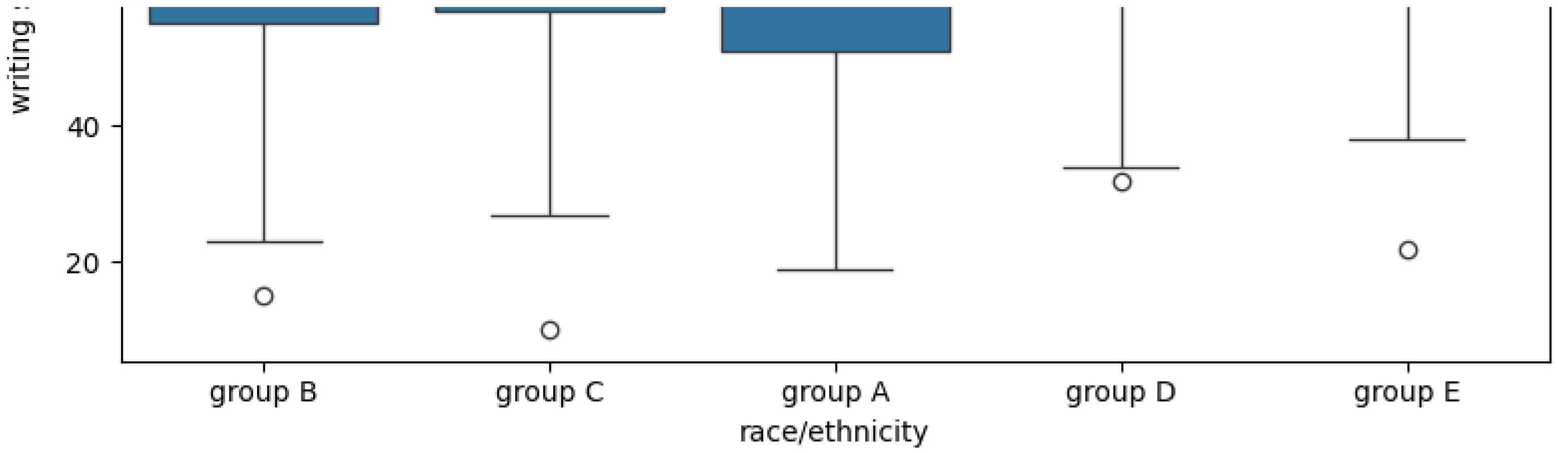
```
print("Percentage of students passed with the race/ethnicity  as 'group A': {0:.2f}%"
    .format((student[(student['race/ethnicity']=='group A') & (student['Pass/Fail']=='P')].shape[0]/student[student['race/ethnicity']=='group A'].shape[0])*100))

print("Percentage of students passed with the race/ethnicity  as 'group B': {0:.2f}%"
    .format((student[(student['race/ethnicity']=='group B') & (student['Pass/Fail']=='P')].shape[0]/student[student['race/ethnicity']=='group B'].shape[0])*100))

print("Percentage of students passed with the race/ethnicity  as 'group C': {0:.2f}%"
    .format((student[(student['race/ethnicity']=='group C') & (student['Pass/Fail']=='P')].shape[0]/student[student['race/ethnicity']=='group C'].shape[0])*100))

print("Percentage of students passed with the race/ethnicity  as 'group D': {0:.2f}%"
    .format((student[(student['race/ethnicity']=='group D') & (student['Pass/Fail']=='P')].shape[0]/student[student['race/ethnicity']=='group D'].shape[0])*100))

print("Percentage of students passed with the race/ethnicity  as 'group E': {0:.2f}%"
    .format((student[(student['race/ethnicity']=='group E') & (student['Pass/Fail']=='P')].shape[0]/student[student['race/ethnicity']=='group E'].shape[0])*100))
```

```
Percentage of students passed with the race/ethnicity  as 'group A': 80.90%
Percentage of students passed with the race/ethnicity  as 'group B': 85.79%
Percentage of students passed with the race/ethnicity  as 'group C': 88.40%
Percentage of students passed with the race/ethnicity  as 'group D': 91.22%
Percentage of students passed with the race/ethnicity  as 'group E': 93.57%
```

In [631…

```
fig, ax = plt.subplots(3,1, figsize=(8,12))
sns.boxplot(x=student['race/ethnicity'],y=student['math score'],ax=ax[0])
sns.boxplot(x=student['race/ethnicity'],y=student['reading score'],ax=ax[1])
sns.boxplot(x=student['race/ethnicity'],y=student['writing score'],ax=ax[2])
plt.tight_layout()
```


100
80
60
40
20
0
math score
group B
group C
group A
group D
group E
race/ethnicity
100
80
60
40
20
reading score
group B
group C
group A
group D
group E
race/ethnicity
100
80
60
score



In [632…

```
student['parental level of education'].value_counts()
```

Out[632…

```
parental level of education
some college          226
associate's degree    222
high school           196
some high school      179
bachelor's degree     118
master's degree        59
Name: count, dtype: int64
```

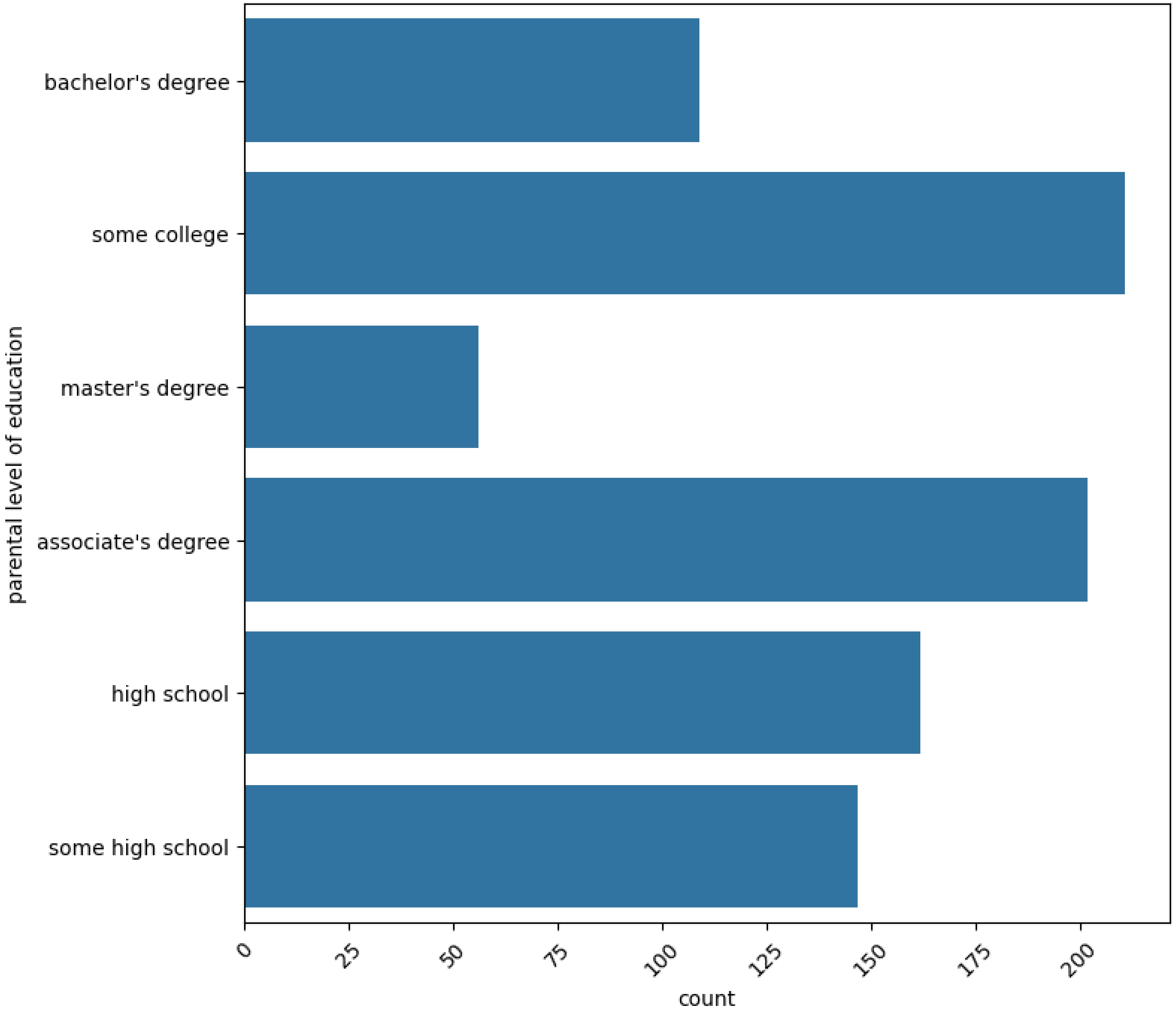
In [633…

```
#number of students passed across the parental levels of education
print("The number of students passed across the different parental levels of education: ")
print(student['parental level of education'].loc[student['Pass/Fail']=='P'].value_counts())
sns.countplot(student['parental level of education'].loc[student['Pass/Fail']=='P'])
plt.xticks(rotation = 45)
```

```
The number of students passed across the different parental levels of education:
parental level of education
some college          211
associate's degree    202
high school           162
some high school      147
bachelor's degree     109
master's degree        56
Name: count, dtype: int64
```

Out[633…

```
(array([  0.,  25.,  50.,  75., 100., 125., 150., 175., 200., 225.]),
 [Text(0.0, 0, '0'),
  Text(25.0, 0, '25'),
  Text(50.0, 0, '50'),
  Text(75.0, 0, '75'),
  Text(100.0, 0, '100'),
  Text(125.0, 0, '125'),
  Text(150.0, 0, '150'),
  Text(175.0, 0, '175'),
  Text(200.0, 0, '200'),
  Text(225.0, 0, '225')])
```



In [634…

```
print("Percentage of students passed with the parental level of education as 'some college': {0:.2f}%"
    .format((student[(student['parental level of education']=='some college') & (student['Pass/Fail']=='P')].shape[0]/student[student['parental level of education']=='some college'

print("Percentage of students passed with the parental level of education as 'associate's degree': {0:.2f}%"
    .format((student[(student['parental level of education']=="associate's degree") & (student['Pass/Fail']=='P')].shape[0]/student[student['parental level of education']=="associa

print("Percentage of students passed with the parental level of education as 'high school': {0:.2f}%"
    .format((student[(student['parental level of education']=="high school") & (student['Pass/Fail']=='P')].shape[0]/student[student['parental level of education']=="high school"].

print("Percentage of students passed with the parental level of education as 'some high school': {0:.2f}%"
    .format((student[(student['parental level of education']=="some high school") & (student['Pass/Fail']=='P')].shape[0]/student[student['parental level of education']=="some high

print("Percentage of students passed with the parental level of education as 'bachelor's degree': {0:.2f}%"
    .format((student[(student['parental level of education']=="bachelor's degree") & (student['Pass/Fail']=='P')].shape[0]/student[student['parental level of education']=="bachelor
print("Percentage of students passed with the parental level of education as 'master's degree': {0:.2f}%"
    .format((student[(student['parental level of education']=="master's degree") & (student['Pass/Fail']=='P')].shape[0]/student[student['parental level of education']=="master's d
```

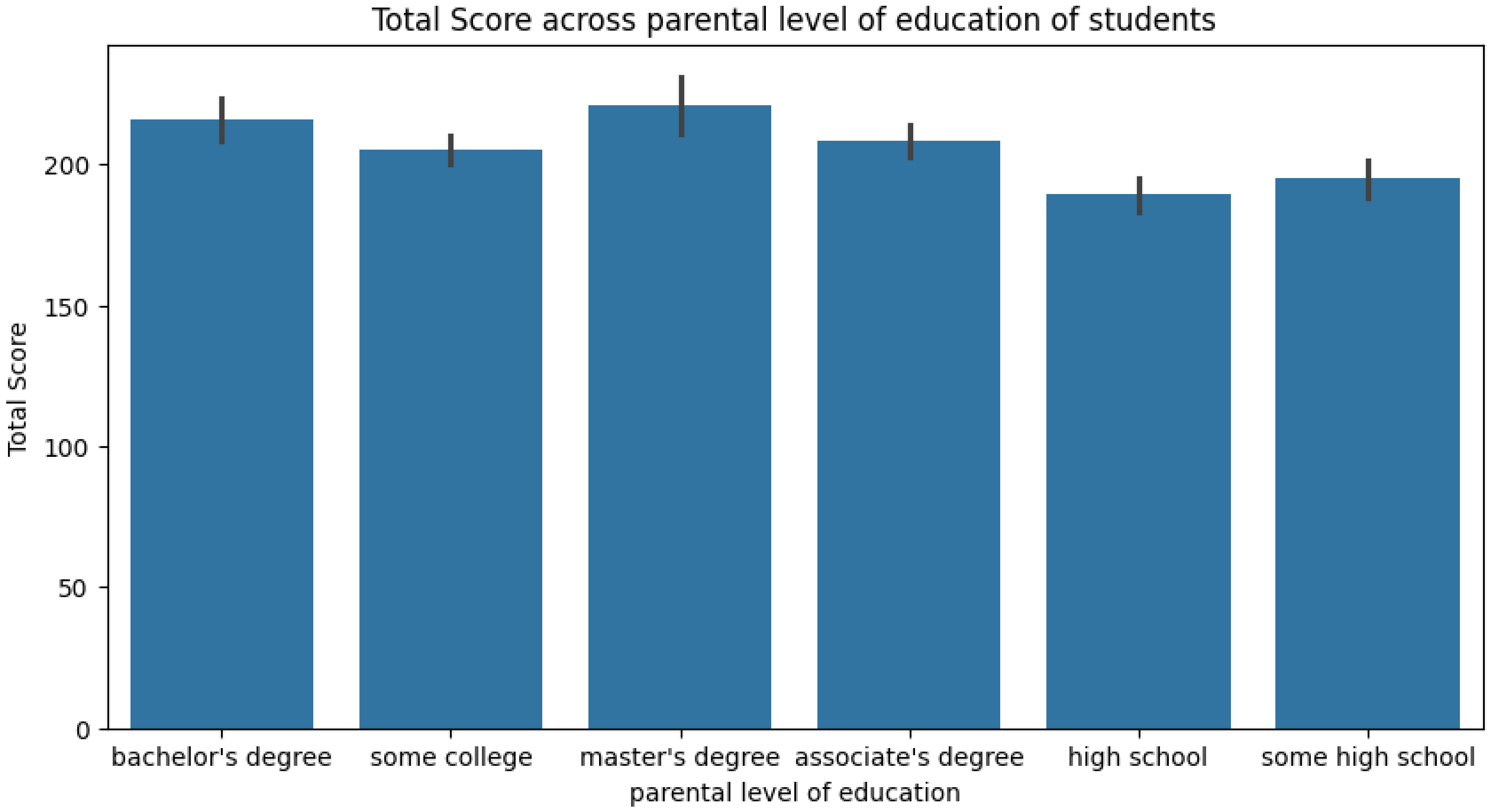
```
Percentage of students passed with the parental level of education as 'some college': 93.36%
Percentage of students passed with the parental level of education as 'associate's degree': 90.99%
Percentage of students passed with the parental level of education as 'high school': 82.65%
Percentage of students passed with the parental level of education as 'some high school': 82.12%
Percentage of students passed with the parental level of education as 'bachelor's degree': 92.37%
Percentage of students passed with the parental level of education as 'master's degree': 94.92%
```

In [635…

```
plt.figure(figsize=(10,5))
plt.title("Total Score across parental level of education of students")
sns.barplot(x=student['parental level of education'],y=student['Total Score'])
```

Out[635…

```
<Axes: title={'center': 'Total Score across parental level of education of students'}, xlabel='parental level of education', ylabel='Total Score'>
```



In [636…

```
student['lunch'].value_counts()
```

Out[636…

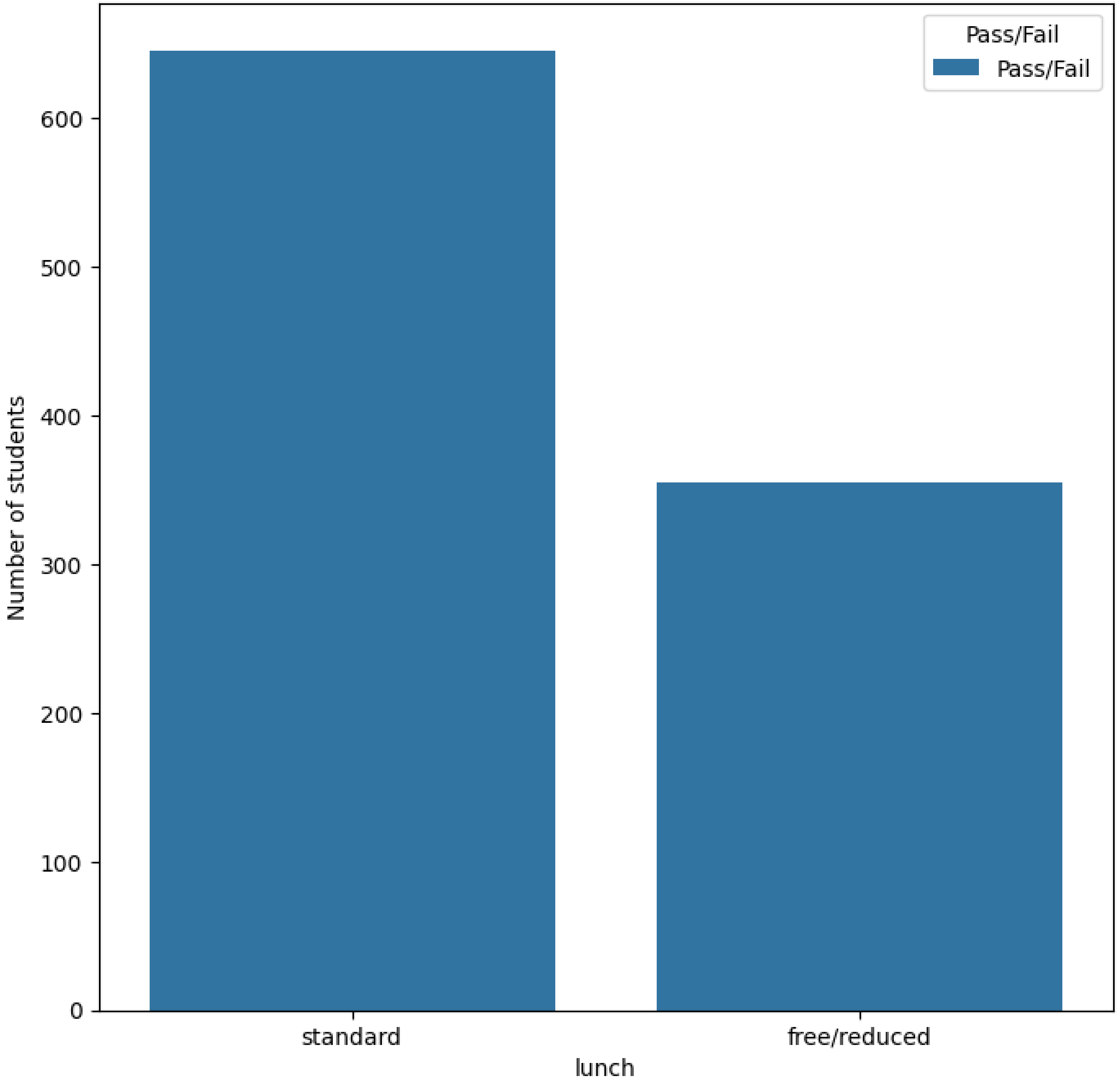
```
lunch
standard        645
free/reduced    355
Name: count, dtype: int64
```

In [637…

```
student['lunch'].loc[student['Pass/Fail']=='P'].value_counts()
```

```
Out[637… lunch
         standard        599
         free/reduced    288
         Name: count, dtype: int64
```

```
In [638… import seaborn as sns
         import matplotlib.pyplot as plt
         import pandas as pd

         melted_data = pd.melt(student, id_vars="lunch", value_vars="Pass/Fail", var_name="Pass/Fail")

         sns.countplot(data=melted_data, x="lunch", hue="Pass/Fail")
         plt.ylabel("Number of students")
         plt.show()
```



In [639…

```
print("Percentage of students passed with the lunch type as 'standard': {0:.2f}%"
    .format((student[(student['lunch']=='standard') & (student['Pass/Fail']=='P')].shape[0]/student[student['lunch']=='standard'].shape[0])*100))

print("Percentage of students passed with the lunch type as 'free/reduced': {0:.2f}%"
    .format((student[(student['lunch']=="free/reduced") & (student['Pass/Fail']=='P')].shape[0]/student[student['lunch']=="free/reduced"].shape[0])*100))
```

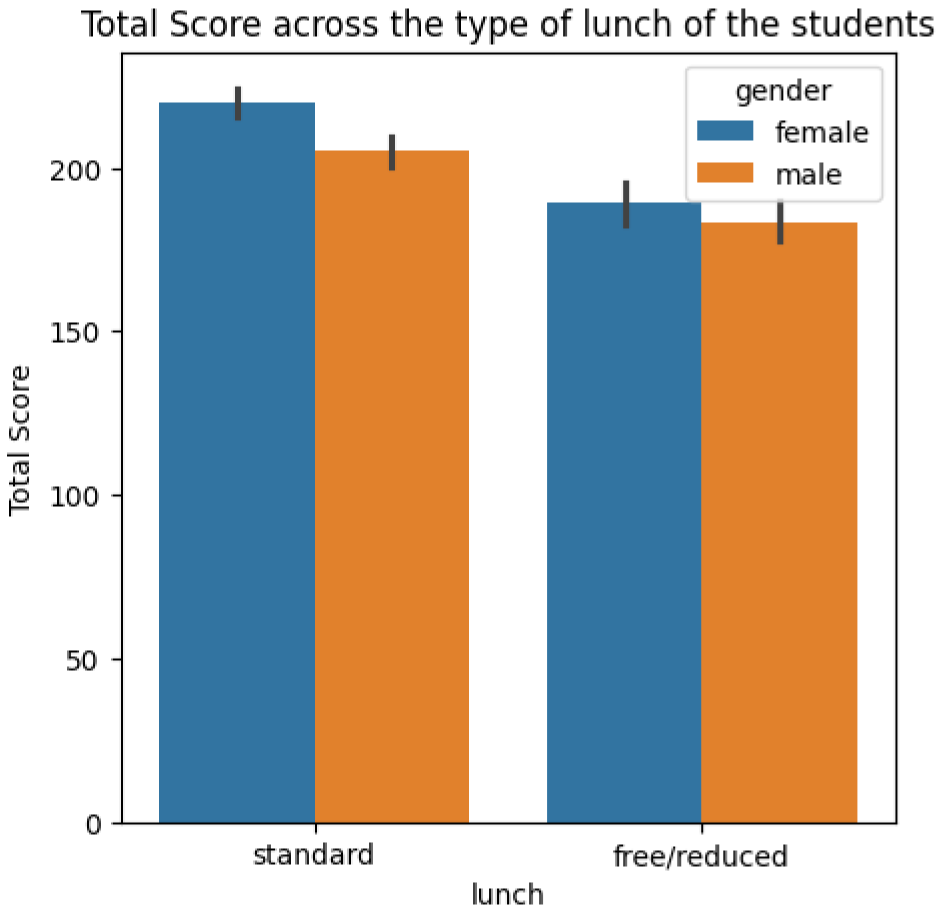
```
Percentage of students passed with the lunch type as 'standard': 92.87%
Percentage of students passed with the lunch type as 'free/reduced': 81.13%
```

In [640…

```
plt.figure(figsize=(5,5))
plt.title("Total Score across the type of lunch of the students")
sns.barplot(x=student['lunch'],y=student['Total Score'],hue=student['gender'])
```

Out[640…

```
<Axes: title={'center': 'Total Score across the type of lunch of the students'}, xlabel='lunch', ylabel='Total Score'>
```



In [641…

```
student['test preparation course'].value_counts()
```

Out[641…

```
test preparation course
none         642
completed    358
Name: count, dtype: int64
```

In [642…

```
print("The number of students passed across the status of completion of the test preparation course:")
print(student['test preparation course'].loc[student['Pass/Fail']=='P'].value_counts())
```

```
The number of students passed across the status of completion of the test preparation course:
test preparation course
none         550
completed    337
Name: count, dtype: int64
```

In [643…

```
print("Percentage of students passed with the test preparation course status as 'none': {0:.2f}%"
    .format((student[(student['test preparation course']=='none') & (student['Pass/Fail']=='P')].shape[0]/student[student['test preparation course']=='none'].shape[0])*100))

print("Percentage of students passed with the test preparation course status as 'completed': {0:.2f}%"
    .format((student[(student['test preparation course']=="completed") & (student['Pass/Fail']=='P')].shape[0]/student[student['test preparation course']=="completed"].shape[0])*10
```

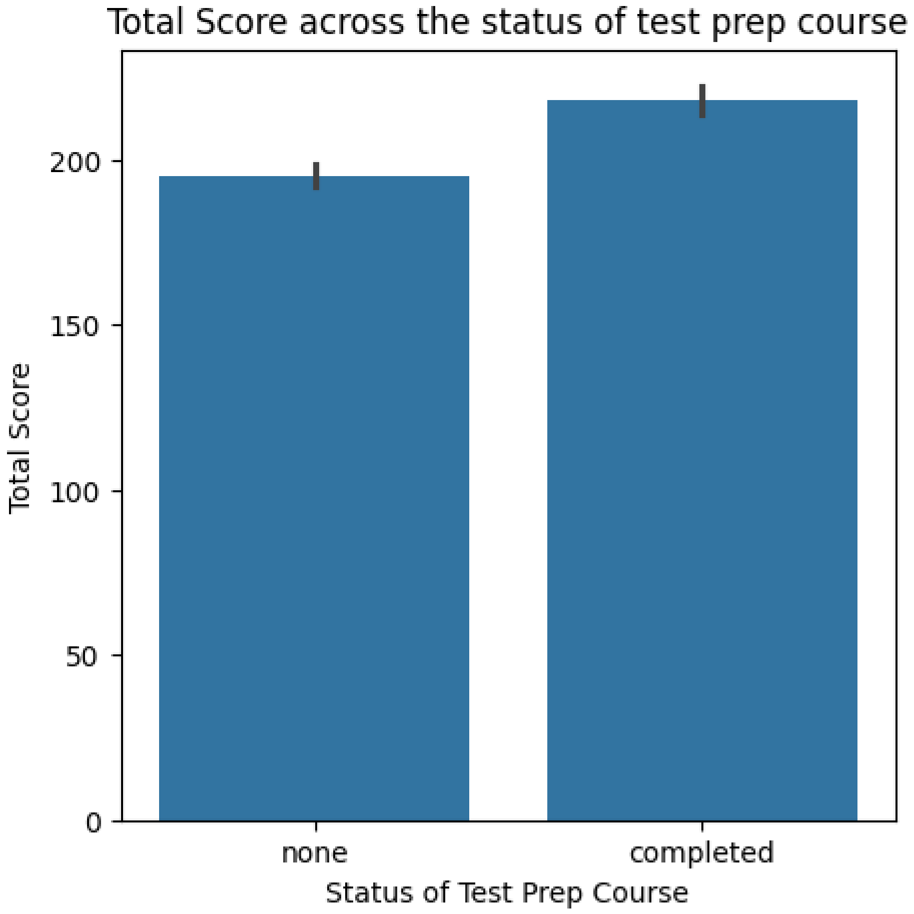
```
Percentage of students passed with the test preparation course status as 'none': 85.67%
Percentage of students passed with the test preparation course status as 'completed': 94.13%
```

In [644…

```
plt.figure(figsize=(5,5))
sns.barplot(x=student['test preparation course'],y=student['Total Score'])
plt.title("Total Score across the status of test prep course")
plt.xlabel('Status of Test Prep Course')
```

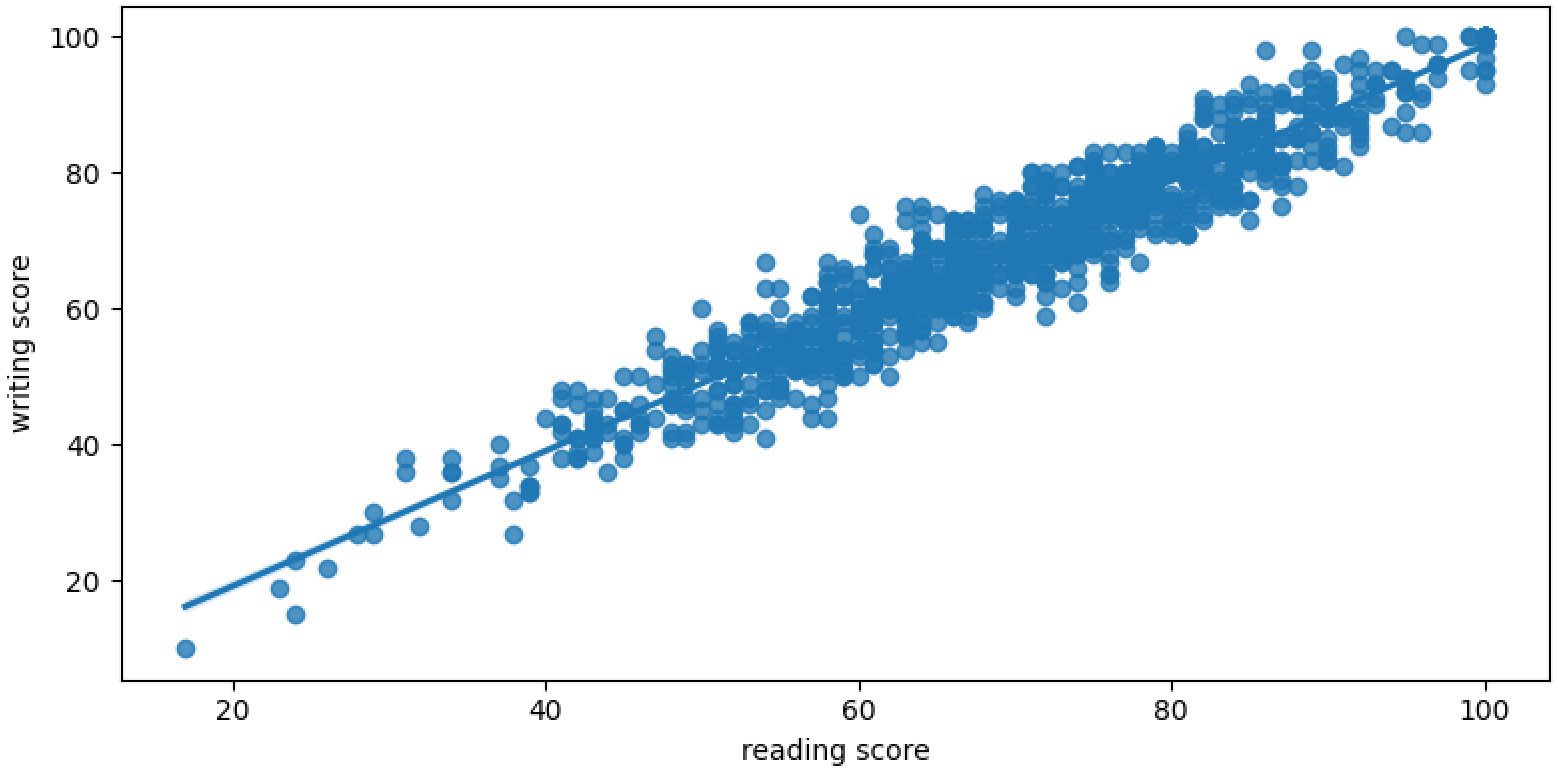
Out[644…

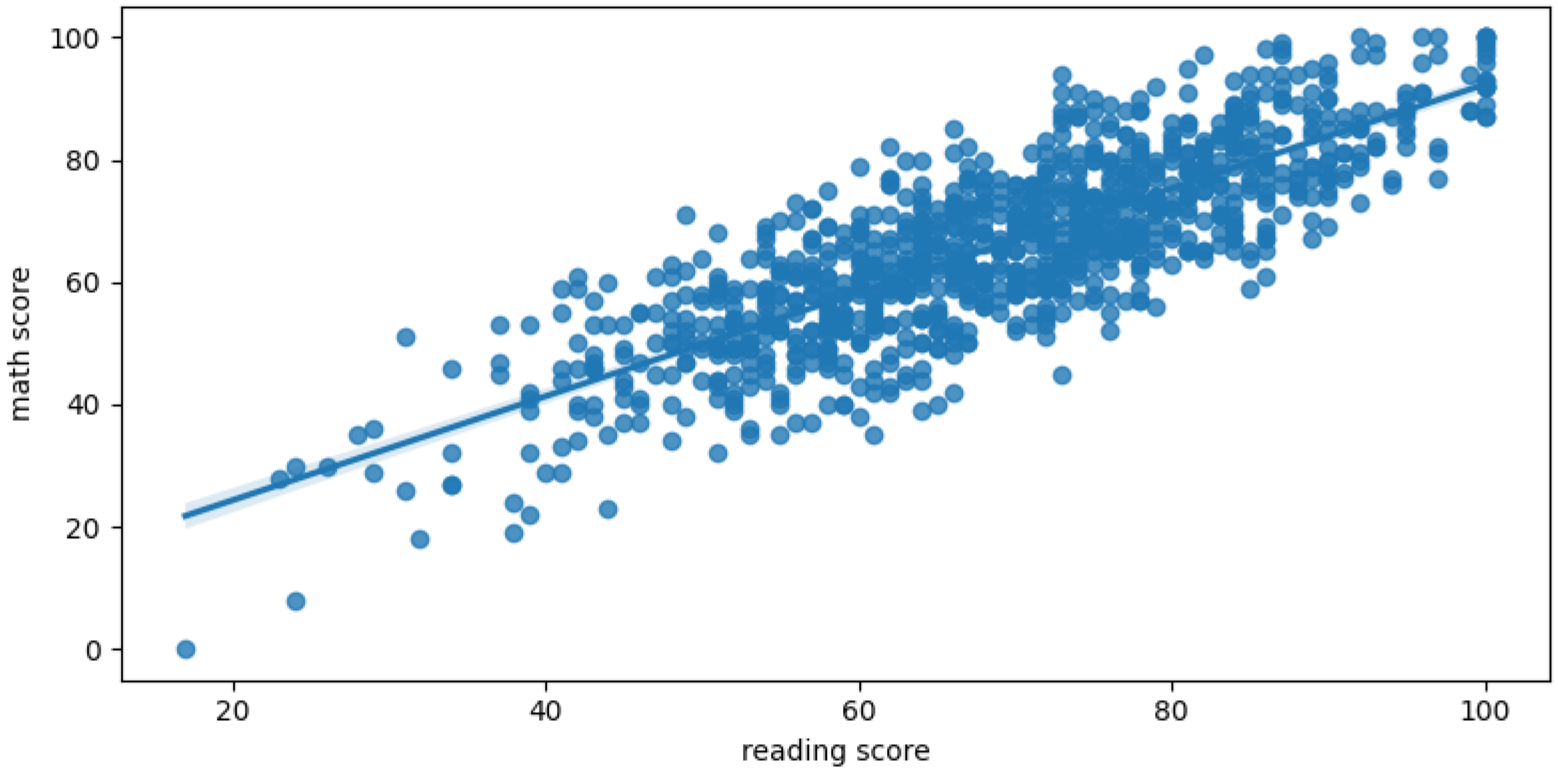
```
Text(0.5, 0, 'Status of Test Prep Course')
```

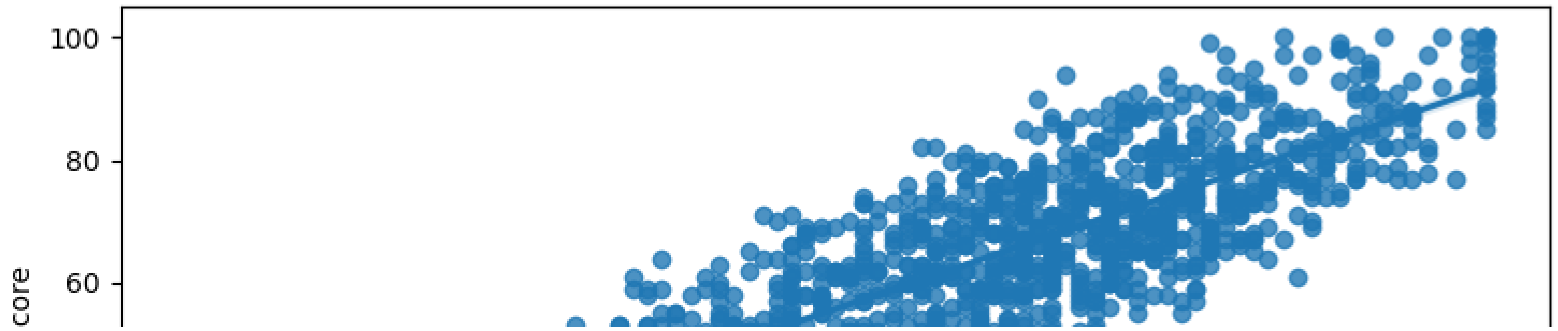


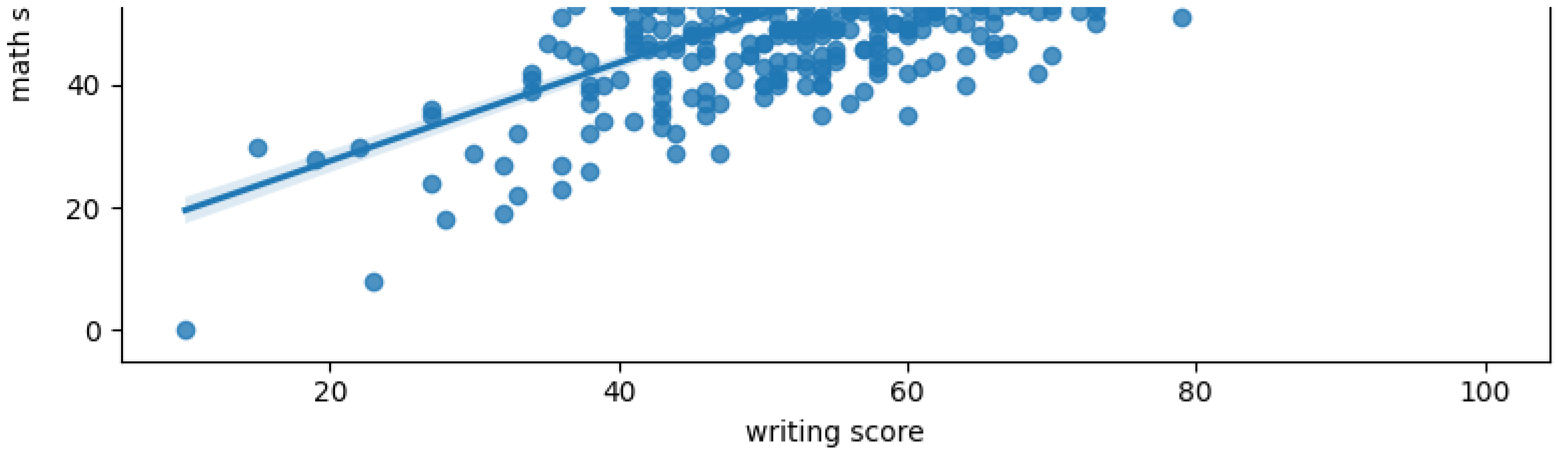
In [645…

```
fig, ax = plt.subplots(3,1, figsize=(8,12))
sns.regplot(x=student['reading score'],y=student['writing score'],ax = ax[0])
sns.regplot(x=student['reading score'],y=student['math score'],ax = ax[1])
sns.regplot(x=student['writing score'],y=student['math score'],ax=ax[2])
plt.tight_layout()
```


writing score
100
80
60
40
20
20
40
60
80
100
reading score

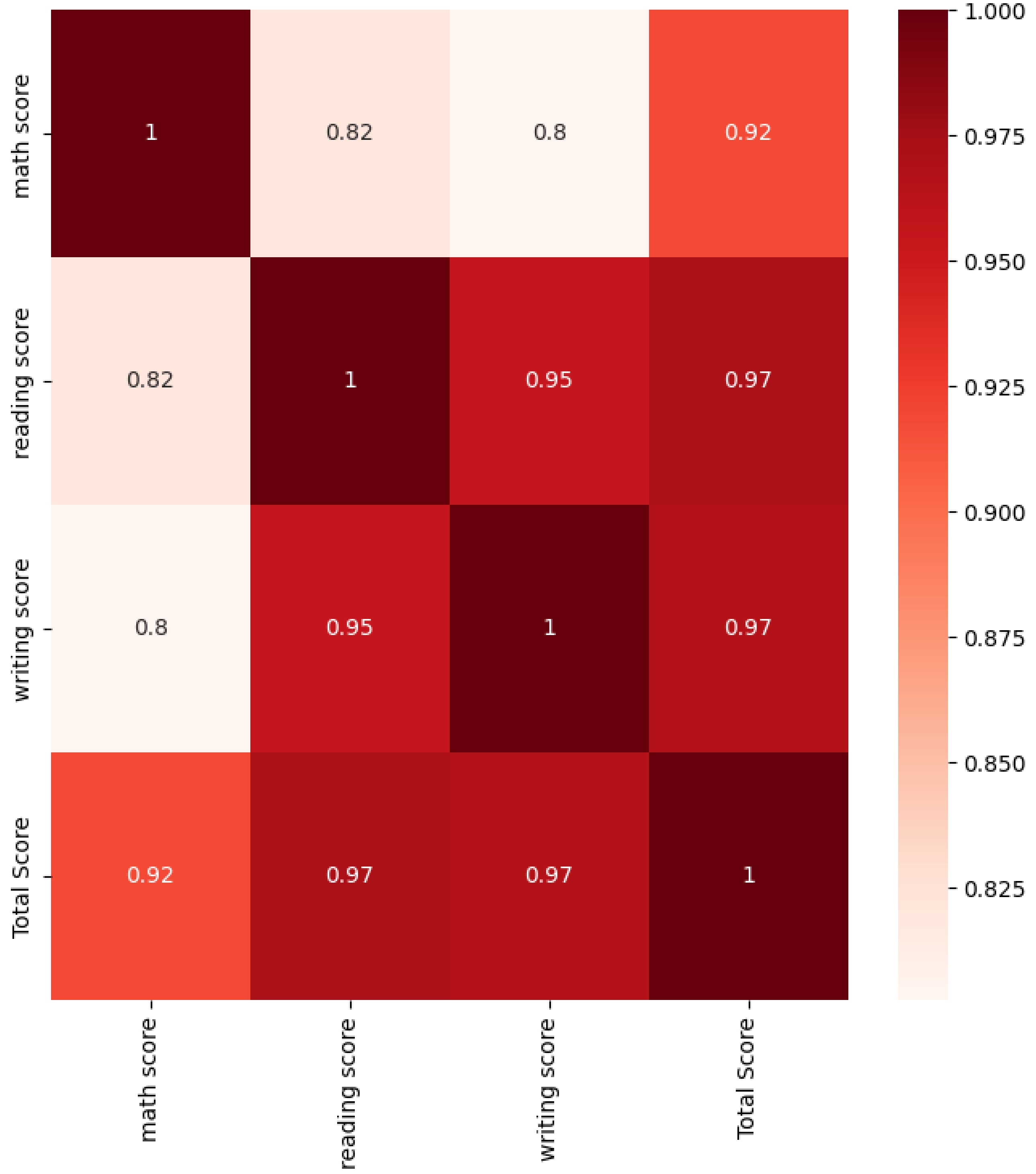

math score
100
80
60
40
20
0
20
40
60
80
100
reading score


core
100
80
60



In [646…

```
import seaborn as sns
import matplotlib.pyplot as plt

numeric_columns = student.select_dtypes(include='number')

correlation_matrix = numeric_columns.corr()

sns.heatmap(correlation_matrix, cmap="Reds", annot=True)
plt.xticks(rotation=90)
plt.show()
```



In [647…

```
student.head()
```

Out[647…

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Total Score	Pass/Fail
0	female	group B	bachelor's degree	standard	none	72	72	74	218	P
1	female	group C	some college	standard	completed	69	90	88	247	P
2	female	group B	master's degree	standard	none	90	95	93	278	P
3	male	group A	associate's degree	free/reduced	none	47	57	44	148	F
4	male	group C	some college	standard	none	76	78	75	229	P

In [648…

```
X=student[['gender','race/ethnicity','parental level of education','lunch','test preparation course']]
X.head()
```

Out[648…

	gender	race/ethnicity	parental level of education	lunch	test preparation course
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 [649…

```
X_category = student[['gender','race/ethnicity','parental level of education','lunch','test preparation course']]
```

In [650…

```
OH_encoder = OneHotEncoder(handle_unknown='ignore', sparse=False)
```

In [651…

```
X_OH = pd.DataFrame(OH_encoder.fit_transform(X_category))
X_OH.index = X.index
X_OH.head()
```

```
/home/kasagg21/.local/lib/python3.8/site-packages/sklearn/preprocessing/_encoders.py:975: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed i
n 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

Out[651…

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
1	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0
2	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0
3	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
4	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0

In [652…

```
y=student['Pass/Fail']
y.head()
```

Out[652…

```
0    P
1    P
2    P
3    F
4    P
Name: Pass/Fail, dtype: object
```

In [653…

```
lb=LabelEncoder()
y=lb.fit_transform(y)
```

In [654…

```
X_train, X_valid, y_train, y_valid = train_test_split(X_OH, y, train_size=0.8, test_size=0.2,random_state=0)
```

In [655…

```
model = RandomForestRegressor()
model.fit(X_train,y_train)
```

Out[655…

```
▾ RandomForestRegressor

RandomForestRegressor()
```

In [656…

```
preds=model.predict(X_valid)
```

In [657…

```
preds= np.where(preds<0.4,0,1)
```

In [658…

```
preds
```

Out[658…

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0])
```

In [659…

```
y_valid
```

Out[659…

```
array([1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1])
```

In [660…

```
mae(y_valid,preds)
```

Out[660…

```
0.14
```

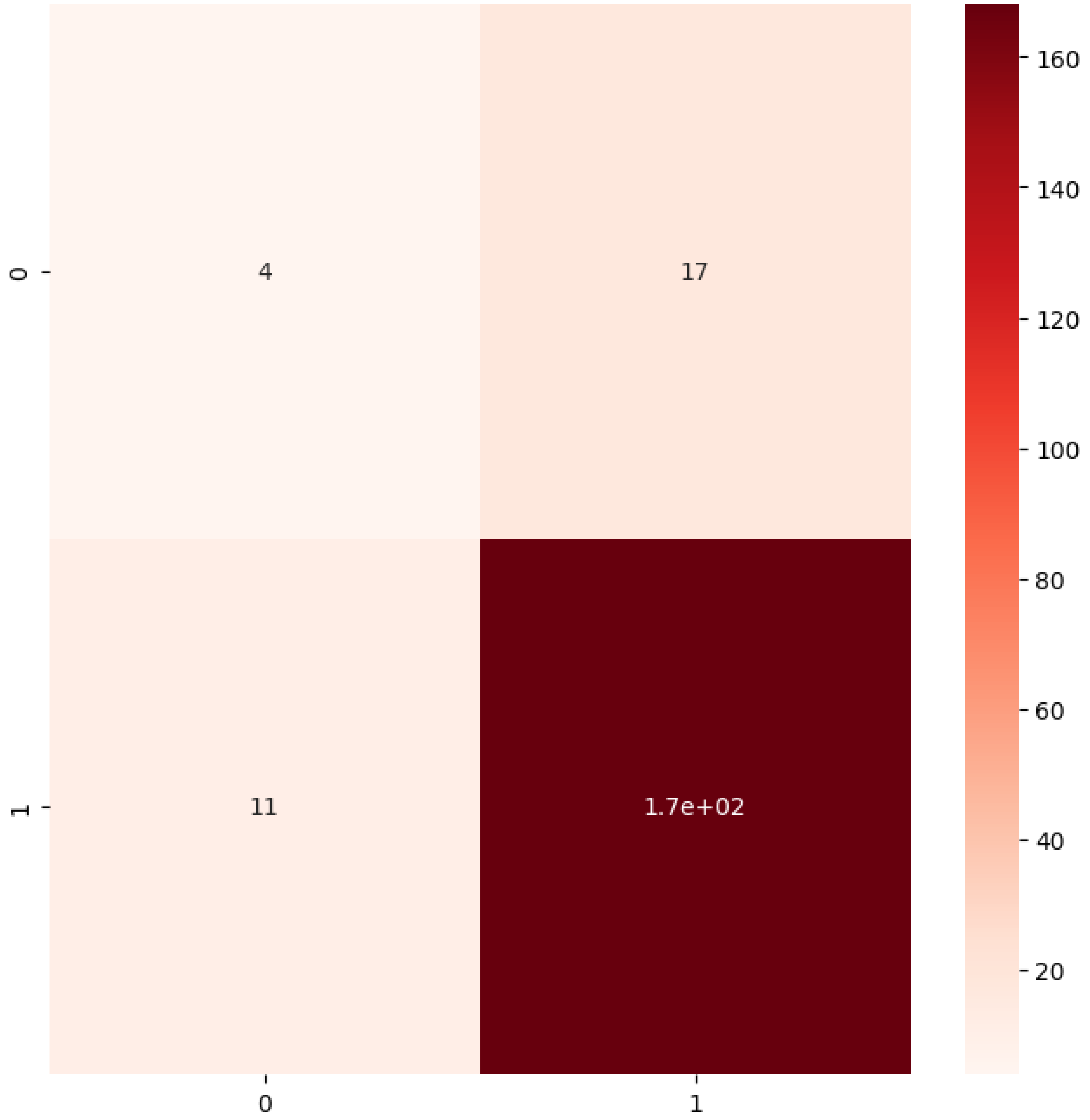
In [661…

```
scores = -1 * cross_val_score(model, X_OH, y,cv=5,scoring='neg_mean_absolute_error')
print("MAE scores:\n", scores)
```

```
MAE scores:
 [0.18972894 0.20502389 0.18242712 0.18319758 0.18461936]
```

In [662…

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_valid, preds)

plt.rcParams['figure.figsize'] = (8, 8)
sns.heatmap(cm, annot = True, cmap = 'Reds')
plt.show()


# Calculate precision, recall, and F1 score
precision = precision_score(y_valid, preds)
recall = recall_score(y_valid, preds)
f1 = f1_score(y_valid, preds)

print(f"Confusion Matrix:\n{cm}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```



```
Confusion Matrix:
[[  4  17]
 [ 11 168]]
Precision: 0.9081081081081082
Recall: 0.9385474860335196
F1 Score: 0.9230769230769231
```

In [663…

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_valid,preds)

print(f"Accuracy: {accuracy:.2%}")
```

```
Accuracy: 86.00%
```

In [676…

```
custom_data = {
    'gender': 'male',
    'race/ethnicity': 'group A',
    'parental level of education': 'some high school',
    'lunch': 'free/reduced',
    'test preparation course': 'none'
}
```

In [677…

```
custom_data_df = pd.DataFrame([custom_data])

custom_data_encoded = pd.DataFrame(OH_encoder.transform(custom_data_df))
custom_data_encoded.index = custom_data_df.index
```

In [678…

```
predictions = model.predict(custom_data_encoded)
```

In [679…

```
pass_fail_prediction = 1 if predictions[0] >= 0.66 else 0
```

In [680…

```
print("Pass/Fail Prediction:", pass_fail_prediction)
```

```
Pass/Fail Prediction: 0
```

In [669…

```
import joblib

#model = RandomForestRegressor()

#joblib.dump(model, '/home/kasagg21/Downloads/archive (2)/trained_model_v3.joblib')
```