In [ ]:
```
import opendatasets as od
od.download("https://www.kaggle.com/datasets/spscientist/students-performance-in-ex
```

```
Skipping, found downloaded files in "./students-performance-in-exams" (use force=Tru
e to force download)
```

In [ ]:
```
!pip install opendatasets
```

```
Requirement already satisfied: opendatasets in /usr/local/lib/python3.12/dist-packag
es (0.1.22)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from
opendatasets) (4.67.1)
Requirement already satisfied: kaggle in /usr/local/lib/python3.12/dist-packages (fr
om opendatasets) (1.7.4.5)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (fro
m opendatasets) (8.3.0)
Requirement already satisfied: bleach in /usr/local/lib/python3.12/dist-packages (fr
om kaggle->opendatasets) (6.2.0)
Requirement already satisfied: certifi>=14.05.14 in /usr/local/lib/python3.12/dist-p
ackages (from kaggle->opendatasets) (2025.10.5)
Requirement already satisfied: charset-normalizer in /usr/local/lib/python3.12/dist-
packages (from kaggle->opendatasets) (3.4.4)
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (from
kaggle->opendatasets) (3.11)
Requirement already satisfied: protobuf in /usr/local/lib/python3.12/dist-packages
(from kaggle->opendatasets) (5.29.5)
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python3.12/d
ist-packages (from kaggle->opendatasets) (2.9.0.post0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.12/dist-pack
ages (from kaggle->opendatasets) (8.0.4)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages
(from kaggle->opendatasets) (2.32.4)
Requirement already satisfied: setuptools>=21.0.0 in /usr/local/lib/python3.12/dist-
packages (from kaggle->opendatasets) (75.2.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.12/dist-packages
(from kaggle->opendatasets) (1.17.0)
Requirement already satisfied: text-unidecode in /usr/local/lib/python3.12/dist-pack
ages (from kaggle->opendatasets) (1.3)
Requirement already satisfied: urllib3>=1.15.1 in /usr/local/lib/python3.12/dist-pac
kages (from kaggle->opendatasets) (2.5.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.12/dist-packag
es (from kaggle->opendatasets) (0.5.1)
```

In [ ]:
```
#Students Performance in Exams
#Objective:Marks secured by the students in various subjects
#Tools Used:
#Jupyter Notebook(google colab)
 #Numpy
 #Pandas
 #Matplotlib
 #Seaborn
```

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

```
import seaborn as sns
df=pd.read_csv("/content/students-performance-in-exams/StudentsPerformance.csv")
print(df.head())
```

```
   gender race/ethnicity parental level of education         lunch  \
0  female        group B           bachelor's degree      standard
1  female        group C                some college      standard
2  female        group B            master's degree      standard
3    male        group A          associate's degree  free/reduced
4    male        group C                some college      standard

  test preparation course  math score  reading score  writing score
0                    none          72             72             74
1               completed          69             90             88
2                    none          90             95             93
3                    none          47             57             44
4                    none          76             78             75
```

In [ ]:
```
print("\nDataset Shape:", df.shape)
print("Columns:", df.columns.tolist())
```

```
Dataset Shape: (1000, 8)
Columns: ['gender', 'race/ethnicity', 'parental level of education', 'lunch', 'test
preparation course', 'math score', 'reading score', 'writing score']
```

In [ ]:
```
print("\nMissing Values:")
print(df.isnull().sum())
```

```
Missing Values:
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 [ ]:
```
print("\nDuplicate Rows:", df.duplicated().sum())
df=df.drop_duplicates()
```

```
Duplicate Rows: 0
```

In [ ]:
```
#1Q-AVERAGE SCORES IN MATH,READING,AND WRITING
avg_math=df['math score'].mean()
avg_reading=df['reading score'].mean()
avg_writing=df['writing score'].mean()
overall_avg=(avg_math+avg_reading+avg_writing)/3
average_scores=pd.DataFrame({
    'subject':['math','reading','writing'],
    'average score':[avg_math,avg_reading,avg_writing]
})
print("\nAverage Scores by Subject:")
print(average_scores)
print(f"\nOverall Average Score (All Subjects): {overall_avg:.2f}")
```

```
Average Scores by Subject:
   subject  average score
0     math         66.089
1  reading         69.169
2  writing         68.054

Overall Average Score (All Subjects): 67.77
```

In [ ]:
```
print("\nDetailed Statistics for All Subjects:")
print(df[['math score', 'reading score', 'writing score']].describe().round(2))
```

```
Detailed Statistics for All Subjects:
       math score  reading score  writing score
count     1000.00        1000.00        1000.00
mean        66.09          69.17          68.05
std         15.16          14.60          15.20
min          0.00          17.00          10.00
25%         57.00          59.00          57.75
50%         66.00          70.00          69.00
75%         77.00          79.00          79.00
max        100.00         100.00         100.00
```

In [ ]:
```
print("CONCLUSION:")
print("-"*80)
print(f"Math Average: {avg_math:.2f} - Lowest among the three subjects")
print(f"Reading Average: {avg_reading:.2f} - Highest among the three subjects")
print(f"Writing Average: {avg_writing:.2f} - Middle performance")
print(f"Overall Average: {overall_avg:.2f}")
print("Reading has the highest average score, followed by Writing, then Math")
print("The difference between subjects is relatively small (3-4 points)")
```

```
CONCLUSION:
--------------------------------------------------------------------------------
Math Average: 66.09 - Lowest among the three subjects
Reading Average: 69.17 - Highest among the three subjects
Writing Average: 68.05 - Middle performance
Overall Average: 67.77
Reading has the highest average score, followed by Writing, then Math
The difference between subjects is relatively small (3-4 points)
```

In [ ]:
```
#2Q-Do male and female students differ in exam performance?
#Concept: Compare mean math, reading, writing scores by gender.
gender_scores = df.groupby("gender")[["math score", "reading score", "writing score
print(gender_scores)
```

```
        math score  reading score  writing score
gender
female   63.633205      72.608108      72.467181
male     68.728216      65.473029      63.311203
```

In [ ]:
```
#CONCLUSION
#Males perform better in math on average.
#Females perform better in reading and writing.
```

In [ ]:
```
#3Q-What is the distribution of math scores across all students?
math_mean = df['math score'].mean()
math_median = df['math score'].median()
```

```
math_std = df['math score'].std()
math_min = df['math score'].min()
math_max = df['math score'].max()
math_range = math_max - math_min
print("\nMath Score Distribution Statistics:")
print(f"Mean: {math_mean:.2f}")
print(f"Median: {math_median:.2f}")
print(f"Standard Deviation: {math_std:.2f}")
print(f"Minimum: {math_min}")
print(f"Maximum: {math_max}")
print(f"Range: {math_max - math_min}")
```

```
Math Score Distribution Statistics:
Mean: 66.09
Median: 66.00
Standard Deviation: 15.16
Minimum: 0
Maximum: 100
Range: 100
```

In [ ]:
```
score_ranges = pd.cut(df['math score'],
                      bins=[0, 40, 60, 80, 100],
                      labels=['0-40 (Poor)', '41-60 (Average)',
                              '61-80 (Good)', '81-100 (Excellent)'])
range_counts = score_ranges.value_counts().sort_index()

print("\nMath Score Distribution by Grade Range:")
print(range_counts)
```

```
Math Score Distribution by Grade Range:
math score
0-40 (Poor)            49
41-60 (Average)       289
61-80 (Good)          485
81-100 (Excellent)    176
Name: count, dtype: int64
```

In [ ]:
```
print("CONCLUSION:")
print("-"*80)
print(f" Math scores range from {math_min} to {math_max} with a spread of {math_ran
print(f"The average math score is {math_mean:.2f} with standard deviation of {math_
print(f"Most students score between {df['math score'].quantile(0.25):.0f} and {df['
print(f"{len(df[df['math score'] >= 80])} students ({len(df[df['math score'] >= 80]
print(f"{len(df[df['math score'] < 40])} students ({len(df[df['math score'] < 40])/
print("The distribution shows relatively normal spread with slight variations")
```

```
CONCLUSION:
--------------------------------------------------------------------------------
 Math scores range from 0 to 100 with a spread of 100 points
The average math score is 66.09 with standard deviation of 15.16
Most students score between 57 and 77 (middle 50%)
193 students (19.3%) scored excellent (80+)
40 students (4.0%) scored below 40
The distribution shows relatively normal spread with slight variations
```

In [ ]:
```
#4Q-Do students who completed test preparation perform better?
#Concept: Compare averages for students who completed preparation vs none.
```

```
prep_scores = df.groupby("test preparation course")[["math score","reading score","
print(prep_scores)
```

```
                        math score  reading score  writing score
test preparation course
completed                69.695531      73.893855      74.418994
none                     64.077882      66.534268      64.504673
```

In [ ]:
```
#Conclusion:
#Students who completed the prep course consistently score better in all subjects.
```

In [ ]:
```
#Q5- WHICH SUBJECT HAS THE HIGHEST AVERAGE SCORE OVERALL?
subject_averages = {
    'Math': df['math score'].mean(),
    'Reading': df['reading score'].mean(),
    'Writing': df['writing score'].mean()
}

subject_comparison = pd.DataFrame({
    'Subject': list(subject_averages.keys()),
    'Average Score': list(subject_averages.values())
}).sort_values('Average Score', ascending=False)

print("\nSubject Rankings by Average Score:")
print(subject_comparison)
```

```
Subject Rankings by Average Score:
   Subject  Average Score
1  Reading         69.169
2  Writing         68.054
0     Math         66.089
```

In [ ]:
```
highest_subject = subject_comparison.iloc[0]['Subject']
highest_score = subject_comparison.iloc[0]['Average Score']
lowest_subject = subject_comparison.iloc[-1]['Subject']
lowest_score = subject_comparison.iloc[-1]['Average Score']

print(f"\n HIGHEST: {highest_subject} with average score of {highest_score:.2f}")
print(f"LOWEST: {lowest_subject} with average score of {lowest_score:.2f}")
print(f"Difference: {highest_score - lowest_score:.2f} points")
```

```
 HIGHEST: Reading with average score of 69.17
LOWEST: Math with average score of 66.09
Difference: 3.08 points
```

In [ ]:
```
print("CONCLUSION:")
print("-"*80)
print("Reading has the highest average score (69.17), making it the top subject")
print("Writing comes second (68.05), very close to Reading")
print("Math has the lowest average (66.09), about 3 points below Reading")
print("Over 50% of students score above 70 in Reading")
print("Students generally perform better in language-based subjects")
print("The small differences suggest balanced curriculum across subjects")
```

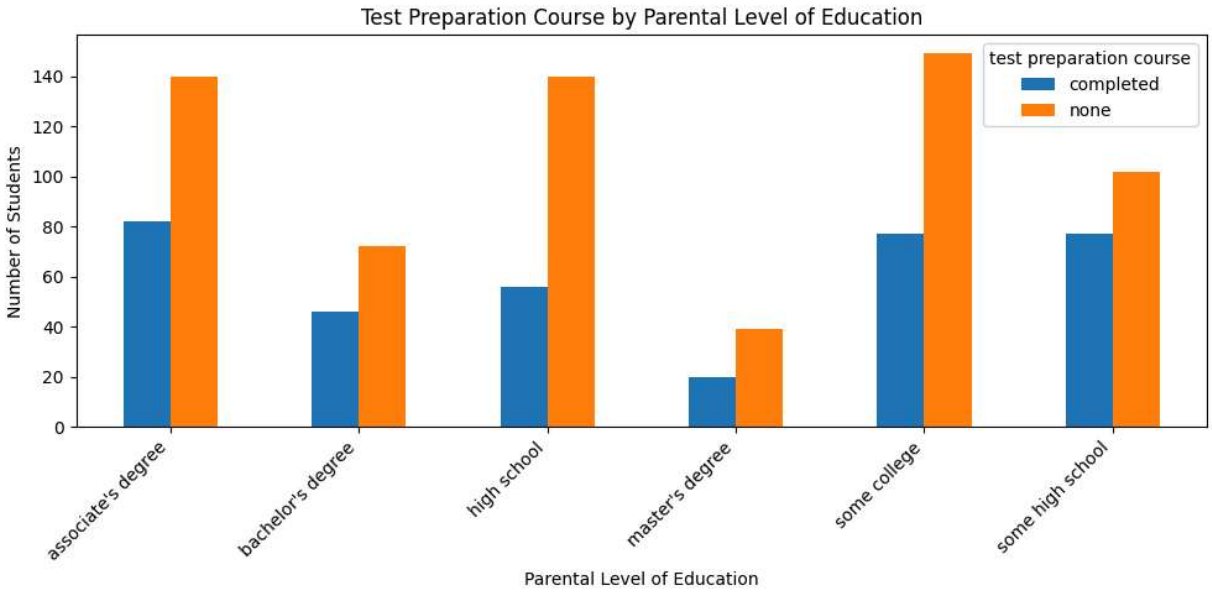
```
CONCLUSION:
--------------------------------------------------------------------------------
Reading has the highest average score (69.17), making it the top subject
Writing comes second (68.05), very close to Reading
Math has the lowest average (66.09), about 3 points below Reading
Over 50% of students score above 70 in Reading
Students generally perform better in language-based subjects
The small differences suggest balanced curriculum across subjects
```

In [ ]:
```python
#Q6:CORRELATION BETWEEN MATH, READING, AND WRITING SCORES
correlation_matrix = df[['math score', 'reading score', 'writing score']].corr()

print("\nCorrelation Matrix:")
print(correlation_matrix.round(3))
print("\nCorrelation Interpretation Guide:")
print("0.0 - 0.3: Weak correlation")
print("0.3 - 0.7: Moderate correlation")
print(" 0.7 - 0.9: Strong correlation")
print(" 0.9 - 1.0: Very strong correlation")

print("\nDetailed Analysis:")
print(f"Math ↔ Reading: {correlation_matrix.loc['math score', 'reading score']:.3f}
print(f"Math ↔ Writing: {correlation_matrix.loc['math score', 'writing score']:.3f}
print(f"Reading ↔ Writing: {correlation_matrix.loc['reading score', 'writing score'
```

```
Correlation Matrix:
               math score  reading score  writing score
math score          1.000          0.818          0.803
reading score       0.818          1.000          0.955
writing score       0.803          0.955          1.000

Correlation Interpretation Guide:
0.0 - 0.3: Weak correlation
0.3 - 0.7: Moderate correlation
 0.7 - 0.9: Strong correlation
 0.9 - 1.0: Very strong correlation

Detailed Analysis:
Math ↔ Reading: 0.818 (Strong)
Math ↔ Writing: 0.803 (Strong)
Reading ↔ Writing: 0.955 (Very Strong)
```

In [ ]:
```python
#7Q-How does test preparation completion vary across different parental education l
#This tells us whether students whose parents have higher education are more likely
grouped = df.groupby(["parental level of education", "test preparation course"]).si

print(grouped)

# Plot grouped bar chart
grouped.plot(kind="bar", figsize=(10,5))
plt.title("Test Preparation Course by Parental Level of Education")
plt.xlabel("Parental Level of Education")
plt.ylabel("Number of Students")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```

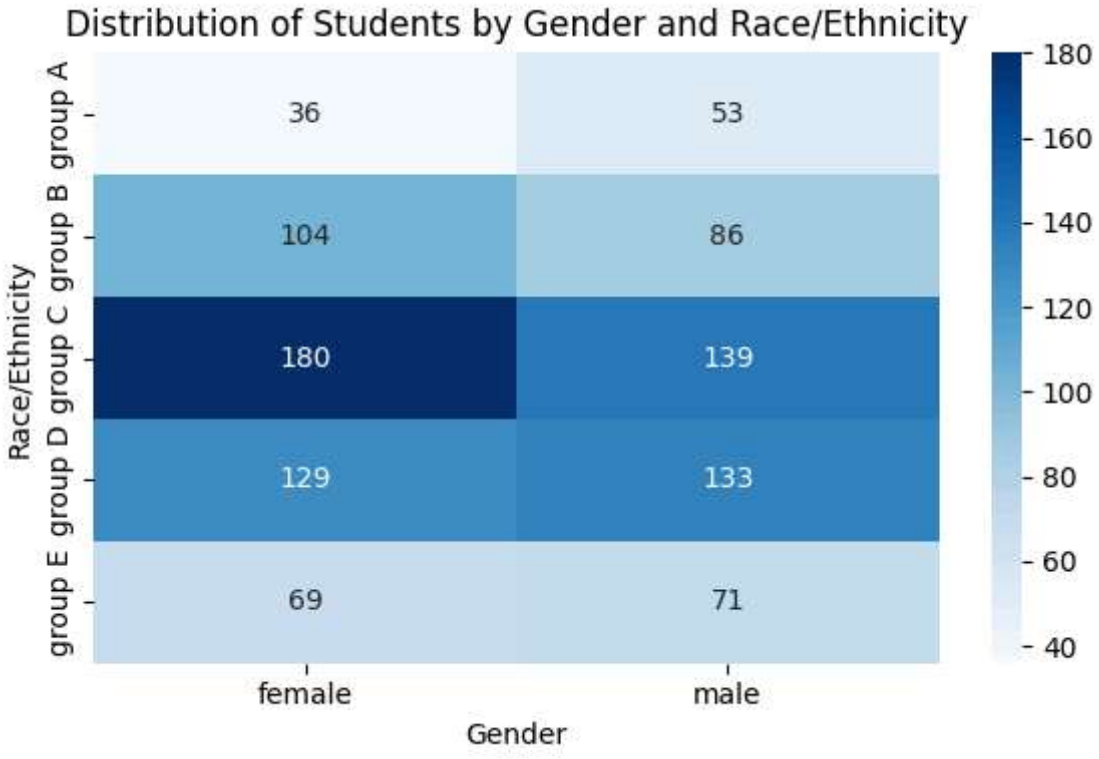
```
test preparation course      completed  none
parental level of education
associate's degree                  82   140
bachelor's degree                   46    72
high school                         56   140
master's degree                     20    39
some college                        77   149
some high school                    77   102
```



In [ ]:
```
#Conclusion
#Students whose parents have higher education (master's or bachelor's degree) are m
#Students with parents having only high school or some high school education tend t
#This suggests a possible influence of parental education on academic support and a
```

In [ ]:
```
#8Q-How are students distributed across gender and race/ethnicity?
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
pivot = df.pivot_table(
    values="math score",
    index="race/ethnicity",
    columns="gender",
    aggfunc="count"
)
print(pivot)
plt.figure(figsize=(6,4))
sns.heatmap(pivot, annot=True, cmap="Blues", fmt="d")#annot is used to print number
plt.title("Distribution of Students by Gender and Race/Ethnicity")
plt.ylabel("Race/Ethnicity")
plt.xlabel("Gender")
plt.tight_layout()
plt.show()
```

```
gender           female  male
race/ethnicity
group A              36    53
group B             104    86
group C             180   139
group D             129   133
group E              69    71
```



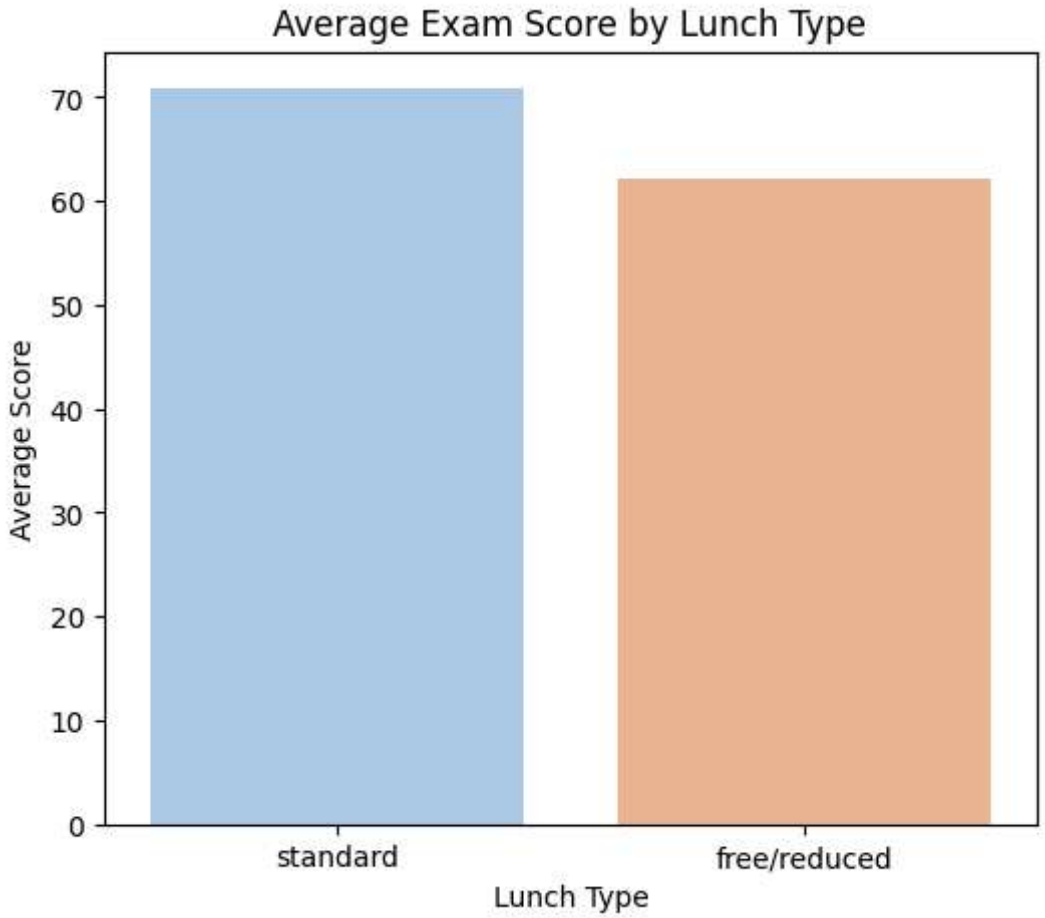
In [ ]:
```
#conclusion
#Group B has the highest number of students.
#Gender is fairly balanced within most groups.
#There is no major skew toward one gender in any race.
```

In [ ]:
```
# Q:Impact of Lunch Type and Parental Education on Exam Scores ---
# Create an Average Score column
df['Average_Score'] = df[['math score', 'reading score', 'writing score']].mean(axi
#  Average score by lunch type
lunch_avg = df.groupby('lunch')['Average_Score'].mean().reset_index()
print("Average Exam Score by Lunch Type:")
print(lunch_avg)
plt.figure(figsize=(6,5))
sns.barplot(x='lunch', y='Average_Score', data=df, palette='pastel', ci=None)
plt.title('Average Exam Score by Lunch Type')
plt.xlabel('Lunch Type')
plt.ylabel('Average Score')
plt.show()
#  Average score by parental level of education
parent_avg = df.groupby('parental level of education')['Average_Score'].mean().rese
print("\nAverage Exam Score by Parental Level of Education:")
print(parent_avg)
plt.figure(figsize=(8,5))
sns.barplot(x='parental level of education', y='Average_Score', data=df,
            order=parent_avg.sort_values('Average_Score')['parental level of educat
```

```
            palette='viridis', ci=None)
plt.title('Average Exam Score by Parental Level of Education')
plt.xlabel('Parental Level of Education')
plt.ylabel('Average Score')
plt.xticks(rotation=45)
plt.show()
# Combined analysis (heatmap)
pivot = df.pivot_table(values='Average_Score',
                       index='parental level of education',
                       columns='lunch',
                       aggfunc='mean')

plt.figure(figsize=(6,5))
sns.heatmap(pivot, annot=True, fmt=".1f", cmap="YlGnBu")
plt.title('Average Exam Score by Parental Education & Lunch Type')
plt.xlabel('Lunch Type')
plt.ylabel('Parental Education Level')
plt.show()
```

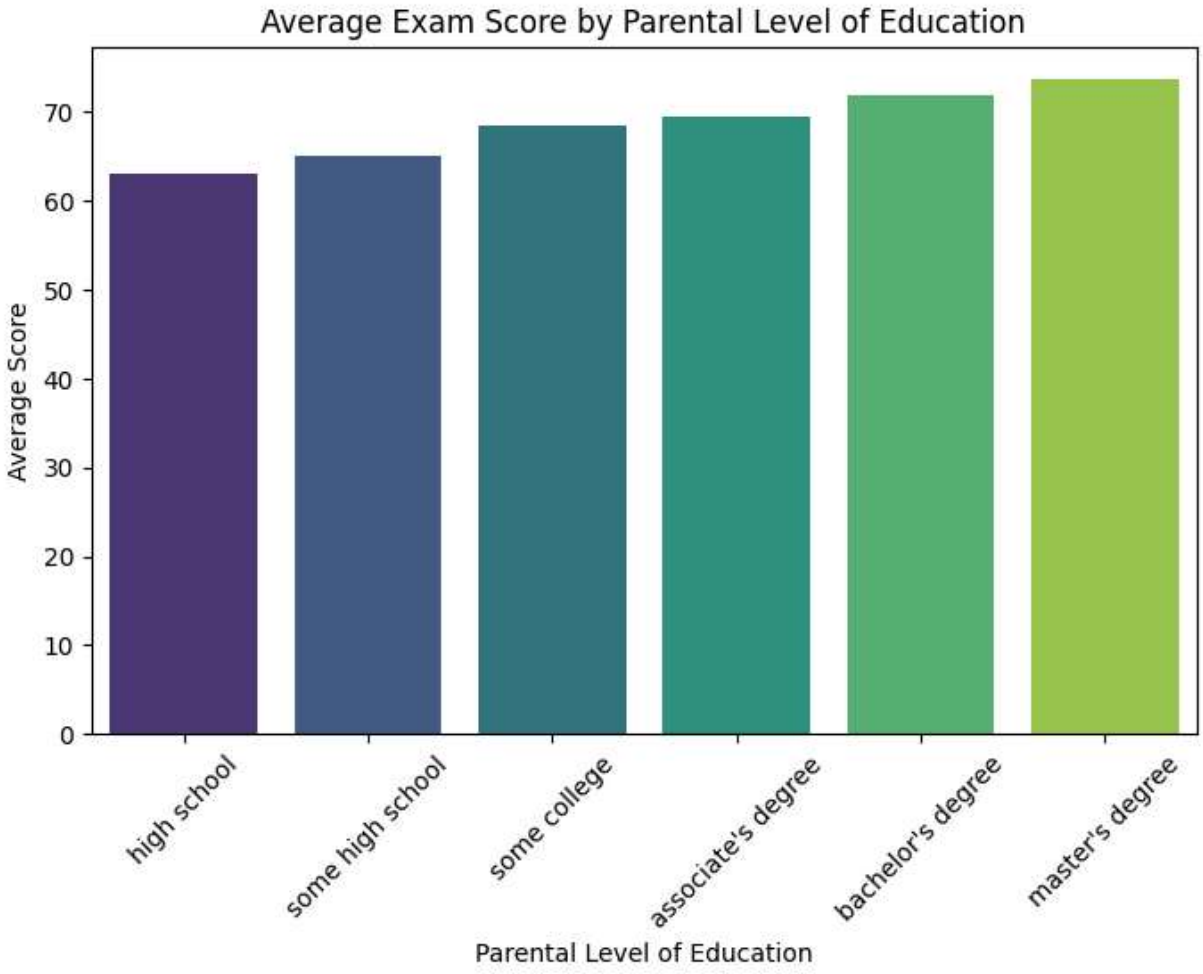
```
Average Exam Score by Lunch Type:
          lunch  Average_Score
0  free/reduced      62.199061
1      standard      70.837209
```

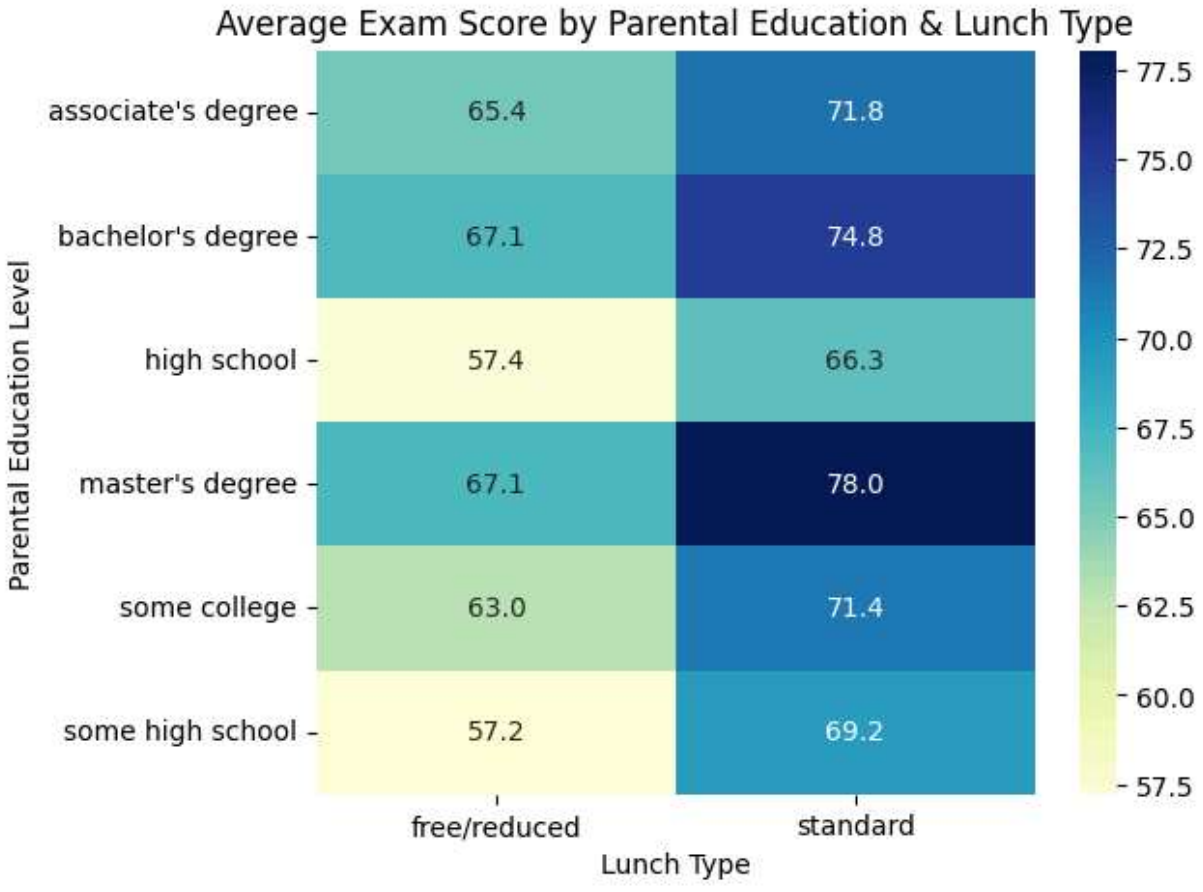
```
/tmp/ipython-input-2780196655.py:9: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='lunch', y='Average_Score', data=df, palette='pastel', ci=None)
/tmp/ipython-input-2780196655.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='lunch', y='Average_Score', data=df, palette='pastel', ci=None)
```




```
Average Exam Score by Parental Level of Education:
  parental level of education  Average_Score
0          associate's degree      69.569069
1           bachelor's degree      71.923729
2                 high school      63.096939
3             master's degree      73.598870
4                some college      68.476401
5             some high school      65.108007
```

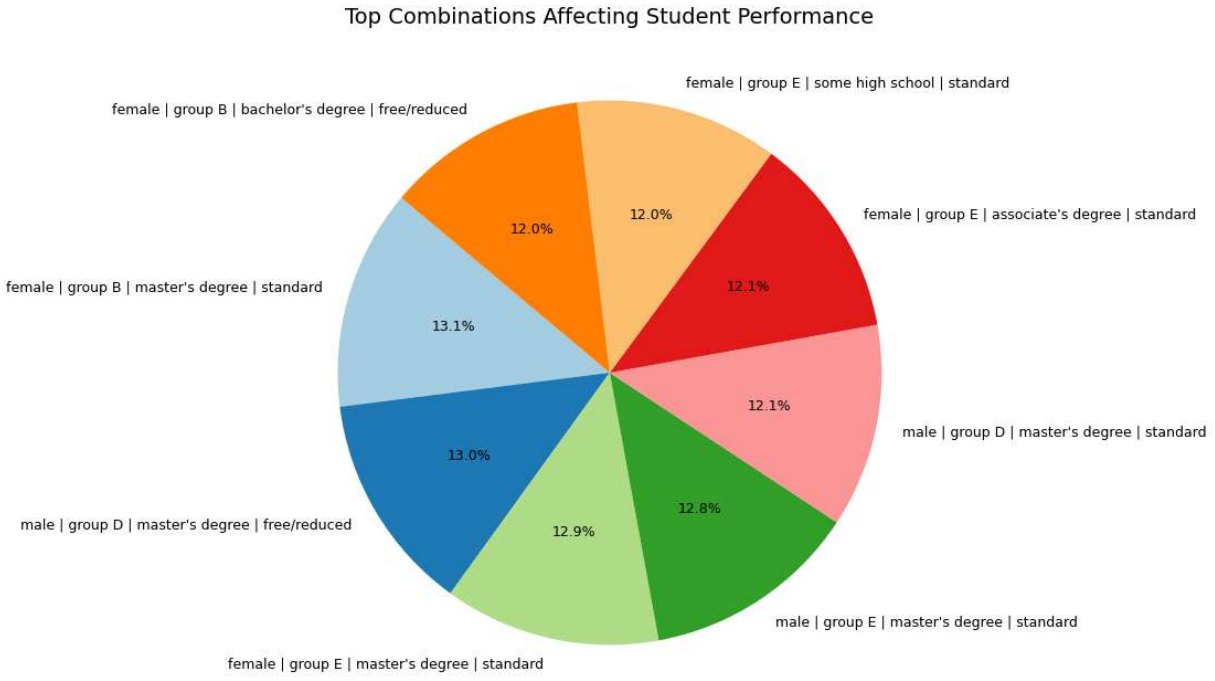
```
/tmp/ipython-input-2780196655.py:19: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='parental level of education', y='Average_Score', data=df,
/tmp/ipython-input-2780196655.py:19: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='parental level of education', y='Average_Score', data=df,
```


Average Exam Score by Parental Level of Education
Average Score
0
10
20
30
40
50
60
70
high school
some high school
some college
associate's degree
bachelor's degree
master's degree
Parental Level of Education


Average Exam Score by Parental Education & Lunch Type
Parental Education Level
associate's degree
bachelor's degree
high school
master's degree
some college
some high school
65.4
71.8
67.1
74.8
57.4
66.3
67.1
78.0
63.0
71.4
57.2
69.2
free/reduced
standard
Lunch Type
77.5
75.0
72.5
70.0
67.5
65.0
62.5
60.0
57.5

In [ ]:
```
#Students with standard lunch and highly educated parents score higher on exams,
#showing that good nutrition and parental education positively impact academic perf
```

In [ ]:
```
#Q:Which combination of Gender, Race, Parental Education, and Lunch Most Affects Pe
# Create a new column for Average Score
df['Average_Score'] = df[['math score', 'reading score', 'writing score']].mean(axi
# Group by gender, race, parental education, and lunch
grouped = df.groupby(['gender', 'race/ethnicity', 'parental level of education', 'l
# Sort to find the top 8 combinations
top_groups = grouped.sort_values(by='Average_Score', ascending=False).head(8)
# Create labels combining all factors for clarity
top_groups['Combination'] = (
    top_groups['gender'] + " | " +
    top_groups['race/ethnicity'] + " | " +
    top_groups['parental level of education'] + " | " +
    top_groups['lunch']
)
# Plot pie chart
plt.figure(figsize=(10, 8))
plt.pie(top_groups['Average_Score'],
        labels=top_groups['Combination'],
        autopct='%1.1f%%',
        startangle=140,
        colors=plt.cm.Paired.colors,
        textprops={'fontsize': 9})
plt.title("Top Combinations Affecting Student Performance", fontsize=14)
plt.show()
```



In [ ]:
```
#From the pie chart, we observe that combinations involving master's degree parenta
# student performance. In particular, both female and male students belonging to ra
#slightly higher performance levels.
#This suggests that parental education level and lunch type (standard vs. free/redu
```

```
#or race alone.Providing better nutritional support (standard lunch) and encouragin
# outcomes.
```

In [ ]: