```
Start coding or generate with AI.
```

Objective: To load and explore the student performance dataset using Pandas.

Tasks:

1. Load the dataset using Pandas.
2. Display first and last five rows.
3. Find shape, columns, and data types.
4. Generate descriptive statistics.

```
import pandas as pd
df=pd.read_csv('Student_Performance.csv')
df
```

	student_id	age	gender	school_type	parent_education	study_hours	attendance_percentage	internet_access	travel_time
0	1	14	male	public	post graduate	3.1	84.3	yes	<15 min
1	2	18	female	public	graduate	3.7	87.8	yes	>60 min
2	3	17	female	private	post graduate	7.9	65.5	no	<15 min
3	4	16	other	public	high school	1.1	58.1	no	15-30 min
4	5	16	female	public	high school	1.3	61.0	yes	30-60 min
...	...	...	...	...	...	...	...	...	...
24995	12047	17	female	public	phd	1.8	55.2	yes	15-30 min
24996	1102	16	female	private	diploma	2.7	97.1	yes	<15 min
24997	4422	19	other	private	post graduate	1.0	63.0	yes	<15 min
24998	7858	14	male	private	diploma	1.0	69.4	yes	15-30 min
24999	11621	18	other	public	no formal	0.7	60.3	yes	30-60 min

5000 rows × 16 columns

Next steps: Generate code with df | New interactive sheet

```
Start coding or generate with AI.
```

```
df.head()
```

	student_id	age	gender	school_type	parent_education	study_hours	attendance_percentage	internet_access	travel_time
0	1	14	male	public	post graduate	3.1	84.3	yes	<15 min
1	2	18	female	public	graduate	3.7	87.8	yes	>60 min
2	3	17	female	private	post graduate	7.9	65.5	no	<15 min
3	4	16	other	public	high school	1.1	58.1	no	15-30 min
4	5	16	female	public	high school	1.3	61.0	yes	30-60 min

Next steps: Generate code with df | New interactive sheet

```
df.tail()
```

	student_id	age	gender	school_type	parent_education	study_hours	attendance_percentage	internet_access	travel_time
24995	12047	17	female	public	phd	1.8	55.2	yes	15-30 min
24996	1102	16	female	private	diploma	2.7	97.1	yes	<15 min
24997	4422	19	other	private	post graduate	1.0	63.0	yes	<15 min
24998	7858	14	male	private	diploma	1.0	69.4	yes	15-30 min
24999	11621	18	other	public	no formal	0.7	60.3	yes	30-60 min

```
print("DataFrame Shape:", df.shape)
print("\nDataFrame Columns:", df.columns.tolist())
print("\nDataFrame Info:")
df.info()
```

```
DataFrame Shape: (25000, 16)

DataFrame Columns: ['student_id', 'age', 'gender', 'school_type', 'parent_education', 'study_hours', 'attendance_percentage', 'i

DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   student_id            25000 non-null  int64
 1   age                   25000 non-null  int64
 2   gender                25000 non-null  object
 3   school_type           25000 non-null  object
 4   parent_education      25000 non-null  object
 5   study_hours           25000 non-null  float64
 6   attendance_percentage 25000 non-null  float64
 7   internet_access       25000 non-null  object
 8   travel_time           25000 non-null  object
 9   extra_activities      25000 non-null  object
 10  study_method          25000 non-null  object
 11  math_score            25000 non-null  float64
 12  science_score         25000 non-null  float64
 13  english_score         25000 non-null  float64
 14  overall_score         25000 non-null  float64
 15  final_grade           25000 non-null  object
dtypes: float64(6), int64(2), object(8)
memory usage: 3.1+ MB
```

```
df.describe()
```

	student_id	age	study_hours	attendance_percentage	math_score	science_score	english_score	overall_score
count	25000.00000	25000.000000	25000.000000	25000.000000	25000.000000	25000.000000	25000.000000	25000.000000
mean	7493.04380	16.482760	4.253224	75.084084	63.785944	63.745320	63.681948	64.006172
std	4323.56215	1.703895	2.167541	14.373171	20.875262	20.970529	20.792693	18.932025
min	1.00000	14.000000	0.500000	50.000000	0.000000	0.000000	0.000000	14.500000
25%	3743.75000	15.000000	2.400000	62.800000	48.300000	48.200000	48.300000	49.000000
50%	7461.50000	16.000000	4.300000	75.100000	64.100000	64.100000	64.200000	64.200000
75%	11252.00000	18.000000	6.100000	87.500000	80.000000	80.000000	80.000000	79.000000
max	15000.00000	19.000000	8.000000	100.000000	100.000000	100.000000	100.000000	100.000000

Objective: To practice selecting columns and filtering rows using Pandas.

Tasks:

1. Select score-related columns.
2. Filter students scoring above 70 in math.
3. Filter data based on gender.
4. Count number of students in each category.

```
score_columns = df[['math_score', 'science_score', 'english_score', 'overall_score']]
display(score_columns.head())
```

	math_score	science_score	english_score	overall_score
0	42.7	55.4	57.0	53.1
1	57.6	68.8	64.8	61.3
2	84.8	95.0	79.2	89.6
3	44.4	27.5	54.7	41.6
4	8.9	32.7	30.0	25.4

```
math_high_performers = df[df['math_score'] > 70]
display(math_high_performers.head())
```

	student_id	age	gender	school_type	parent_education	study_hours	attendance_percentage	internet_access	travel_time	ext
2	3	17	female	private	post graduate	7.9	65.5	no	<15 min	
9	10	14	female	public	diploma	6.8	62.4	yes	>60 min	
10	11	17	female	private	graduate	6.1	90.5	yes	15-30 min	
12	13	18	female	private	high school	6.8	58.2	yes	>60 min	
14	15	18	other	public	high school	4.9	85.3	yes	<15 min	

```
Start coding or generate with AI.
```

```
print("Unique genders:", df['gender'].unique())
```

```
Unique genders: ['male' 'female' 'other']
```

```
female_students = df[df['gender'] == 'female']
display(female_students.head())
```

	student_id	age	gender	school_type	parent_education	study_hours	attendance_percentage	internet_access	travel_time	extr
1	2	18	female	public	graduate	3.7	87.8	yes	>60 min	
2	3	17	female	private	post graduate	7.9	65.5	no	<15 min	
4	5	16	female	public	high school	1.3	61.0	yes	30-60 min	
6	7	14	female	private	post graduate	1.8	81.6	yes	30-60 min	
7	8	18	female	private	post graduate	5.6	59.4	yes	>60 min	

```
categorical_columns = ['gender', 'school_type', 'parent_education', 'internet_access', 'travel_time', 'extra_activities', 'study

for col in categorical_columns:
    print(f"\nCounts for {col.replace('_', ' ').title()}:")
    display(df[col].value_counts())
```

```
Counts for Gender:
```

gender	count
other	8463
female	8290
male	8247

dtype: int64

```
Counts for School Type:
```

school_type	count
private	12725
post graduate	4196
graduate	4127
no formal	4079
phd	4079

dtype: int64

```
Counts for Travel Time:
```

travel_time	count
15-30 min	6362
30-60 min	6345
>60 min	6166
<15 min	6127

dtype: int64

```
Counts for Extra Activities:
```

extra_activities	count
yes	12500

Objective: To perform numerical computations on student scores using NumPy.

Tasks:

1. Convert score columns into NumPy arrays.
2. Compute mean, median, standard deviation.
3. Find minimum and maximum scores.

```
import numpy as np

score_columns_list = ['math_score', 'science_score', 'english_score', 'overall_score']

# Create a dictionary to store NumPy arrays for each score column
score_arrays = {}
for col in score_columns_list:
    score_arrays[col] = df[col].to_numpy()
    print(f"Converted '{col}' to NumPy array. Type: {type(score_arrays[col])}")

# Display the first few elements of one of the arrays as an example
print(f"\nFirst 5 elements of 'math_score' NumPy array: {score_arrays['math_score'][:5]}")
```

```
Converted 'math_score' to NumPy array. Type: <class 'numpy.ndarray'>
Converted 'science_score' to NumPy array. Type: <class 'numpy.ndarray'>
Converted 'english_score' to NumPy array. Type: <class 'numpy.ndarray'>
Converted 'overall_score' to NumPy array. Type: <class 'numpy.ndarray'>

First 5 elements of 'math_score' NumPy array: [42.7 57.6 84.8 44.4  8.9]
```

```
print("\n--- Numerical Statistics for Score Columns ---")
for col_name, score_array in score_arrays.items():
    mean_score = np.mean(score_array)
    median_score = np.median(score_array)
    std_dev_score = np.std(score_array)
    print(f"\n{col_name.replace('_', ' ').title()} Scores:")
    print(f"  Mean: {mean_score:.2f}")
    print(f"  Median: {median_score:.2f}")
    print(f"  Standard Deviation: {std_dev_score:.2f}")
```

```
--- Numerical Statistics for Score Columns ---

Math Score Scores:
  Mean: 63.79
  Median: 64.10
  Standard Deviation: 20.87

Science Score Scores:
  Mean: 63.75
  Median: 64.10
  Standard Deviation: 20.97

English Score Scores:
  Mean: 63.68
  Median: 64.20
  Standard Deviation: 20.79

Overall Score Scores:
  Mean: 64.01
  Median: 64.20
  Standard Deviation: 18.93
```

```
Start coding or generate with AI.
```

```
print("\n--- Minimum and Maximum Scores ---")
for col_name, score_array in score_arrays.items():
    min_score = np.min(score_array)
```

```
    max_score = np.max(score_array)
    print(f"\n{col_name.replace('_', ' ').title()} Scores:")
    print(f"  Minimum: {min_score:.2f}")
    print(f"  Maximum: {max_score:.2f}")
```

```
notes          4165
--- Minimum and Maximum Scores ---
online videos  4139
Math Score Scores:
group study    4090
  Minimum: 0.00
  Maximum: 100.00
coaching       4026
Science Score Scores:
dtype: int64
  Minimum: 0.00
  Maximum: 100.00
Counts for Final Grade:
English Score Scores:
  Minimum: 0.00
  Maximum: 100.00
final_grade
-----------
Overall Score Scores:
  Minimum: 14.50
  Maximum: 100.00
```

final_grade	count
d	6311
c	6061
e	5672
f	2955
b	2696
a	1205

dtype: int64

Objective: To visualize student performance using Matplotlib.

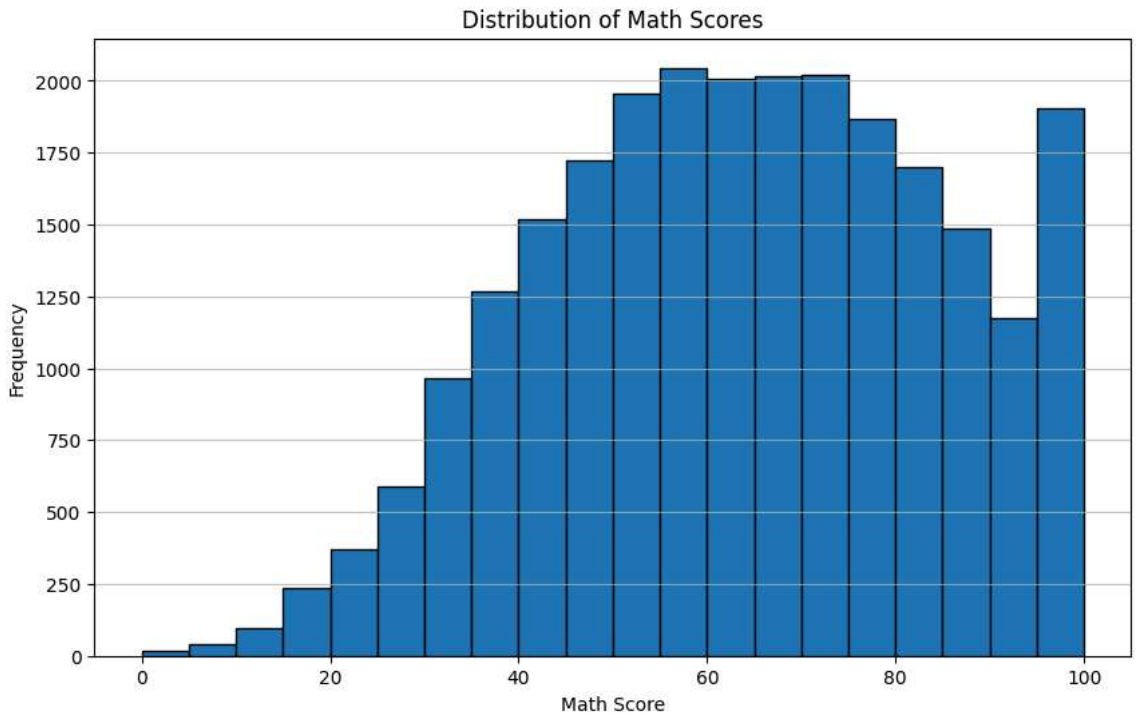
Tasks:

1. Plot histogram of math scores.
2. Plot bar chart of average scores.
3. Create scatter plot between math and writing scores.

```
Start coding or generate with AI.
```

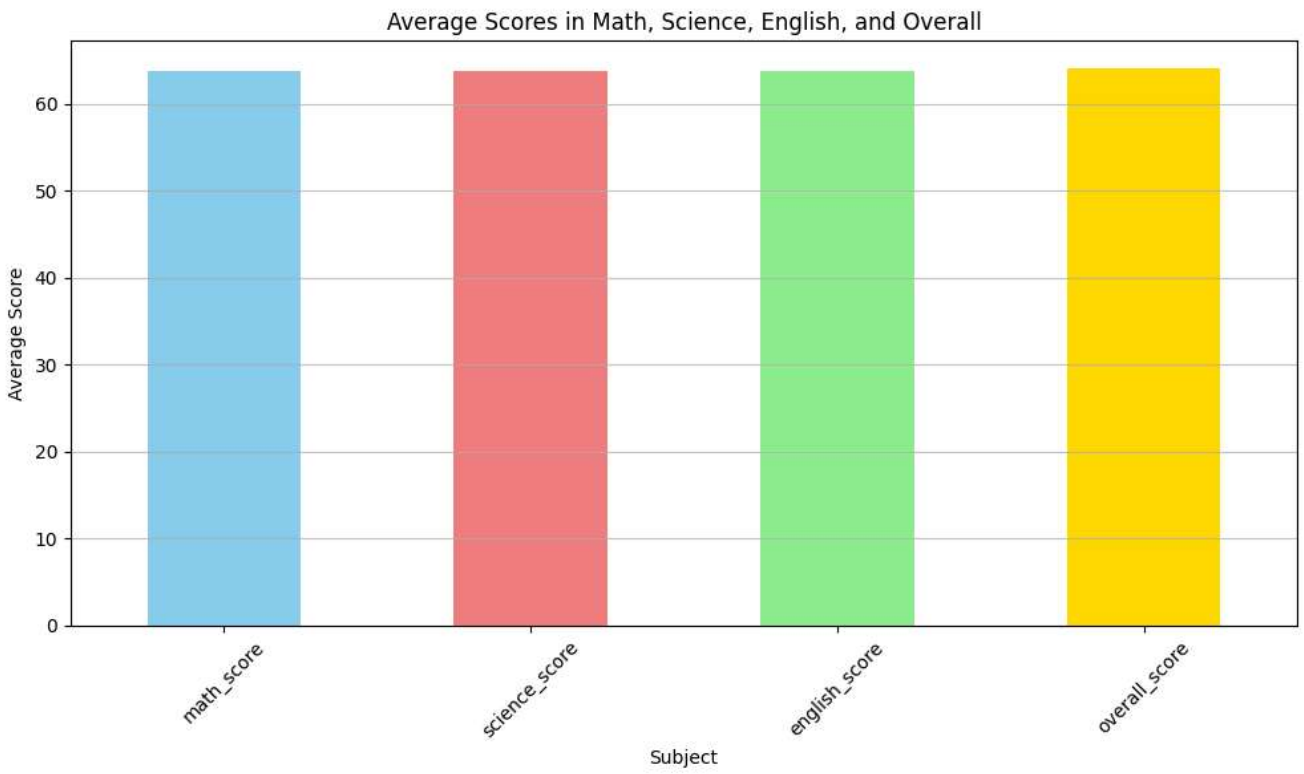
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.hist(df['math_score'], bins=20, edgecolor='black')
plt.title('Distribution of Math Scores')
plt.xlabel('Math Score')
plt.ylabel('Frequency')
plt.grid(axis='y', alpha=0.75)
plt.show()
```



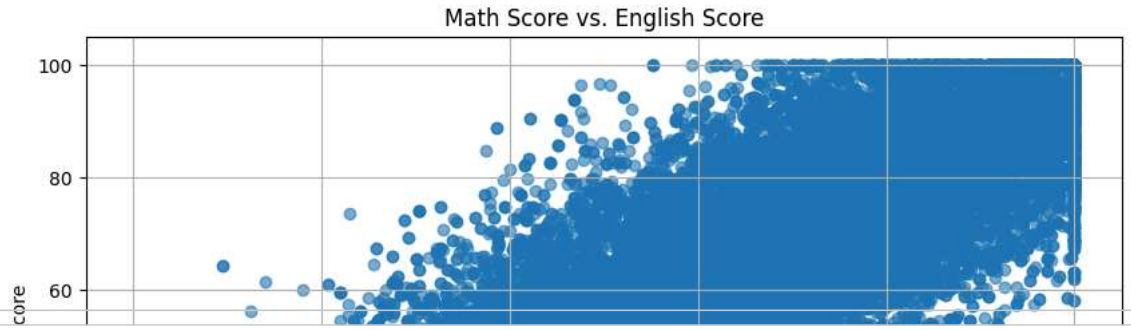
Reasoning: The next step is to create a bar chart showing the average scores for math, science, English, and overall, using the pre-calculated `average_scores` series. This will involve using `matplotlib.pyplot` to generate the bar chart and adding appropriate labels and a title for clarity.

```
plt.figure(figsize=(10, 6))
average_scores.plot(kind='bar', color=['skyblue', 'lightcoral', 'lightgreen', 'gold'])
plt.title('Average Scores in Math, Science, English, and Overall')
plt.xlabel('Subject')
plt.ylabel('Average Score')
plt.xticks(rotation=45)
plt.grid(axis='y', alpha=0.75)
plt.tight_layout()
plt.show()
```



Reasoning: To visualize the relationship between math and English scores, I will create a scatter plot. This involves using `matplotlib.pyplot` to generate the plot, setting appropriate labels for the axes, and adding a descriptive title.

```
plt.figure(figsize=(10, 6))
plt.scatter(df['math_score'], df['english_score'], alpha=0.6)
plt.title('Math Score vs. English Score')
plt.xlabel('Math Score')
plt.ylabel('English Score')
plt.grid(True)
plt.show()
```



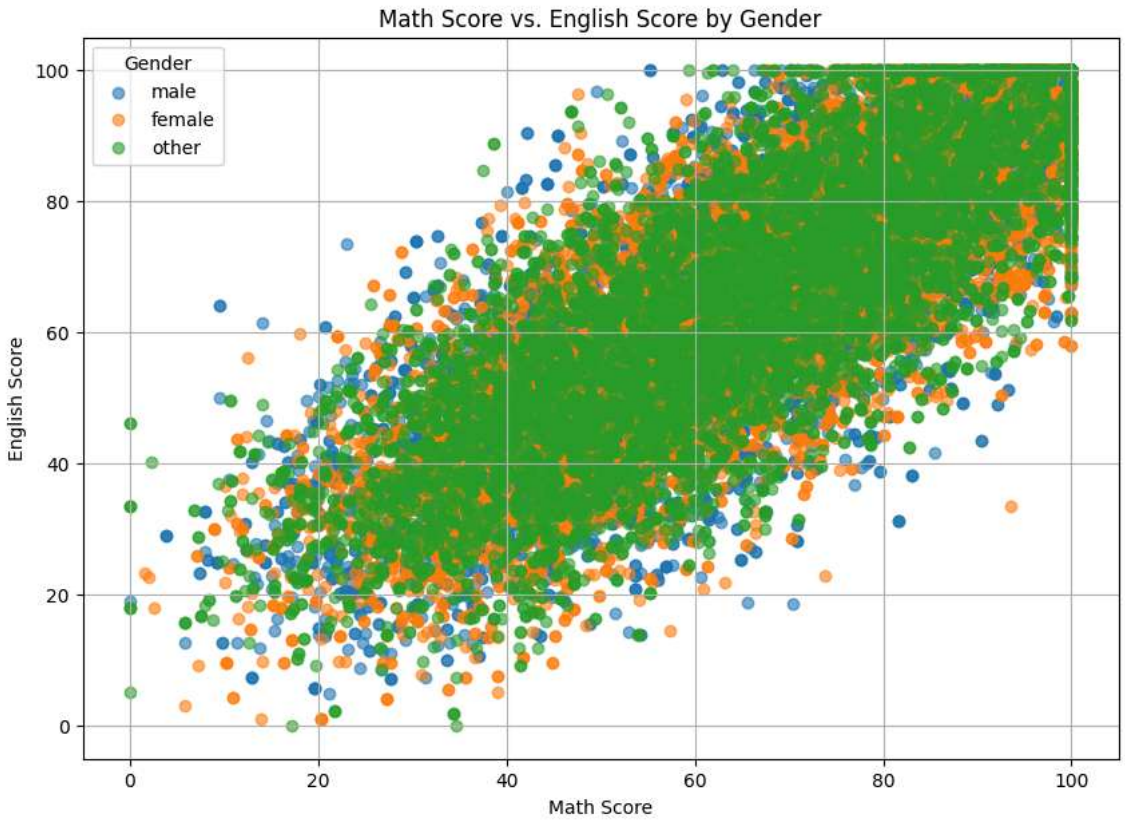

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 7))

unique_genders = df['gender'].unique()

for gender in unique_genders:
    gender_data = df[df['gender'] == gender]
    plt.scatter(gender_data['math_score'], gender_data['english_score'], label=gender, alpha=0.6)

plt.title('Math Score vs. English Score by Gender')
plt.xlabel('Math Score')
plt.ylabel('English Score')
plt.legend(title='Gender')
plt.grid(True)
plt.show()
```



Objective: To analyze student performance based on different groups.

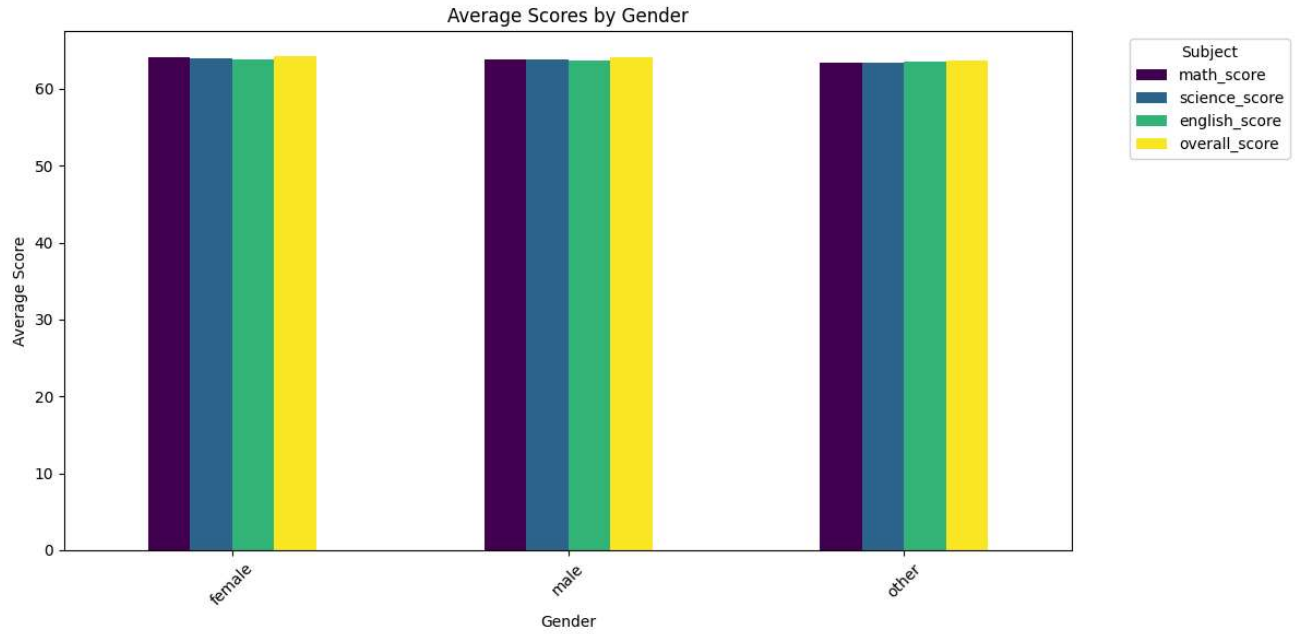
Tasks:

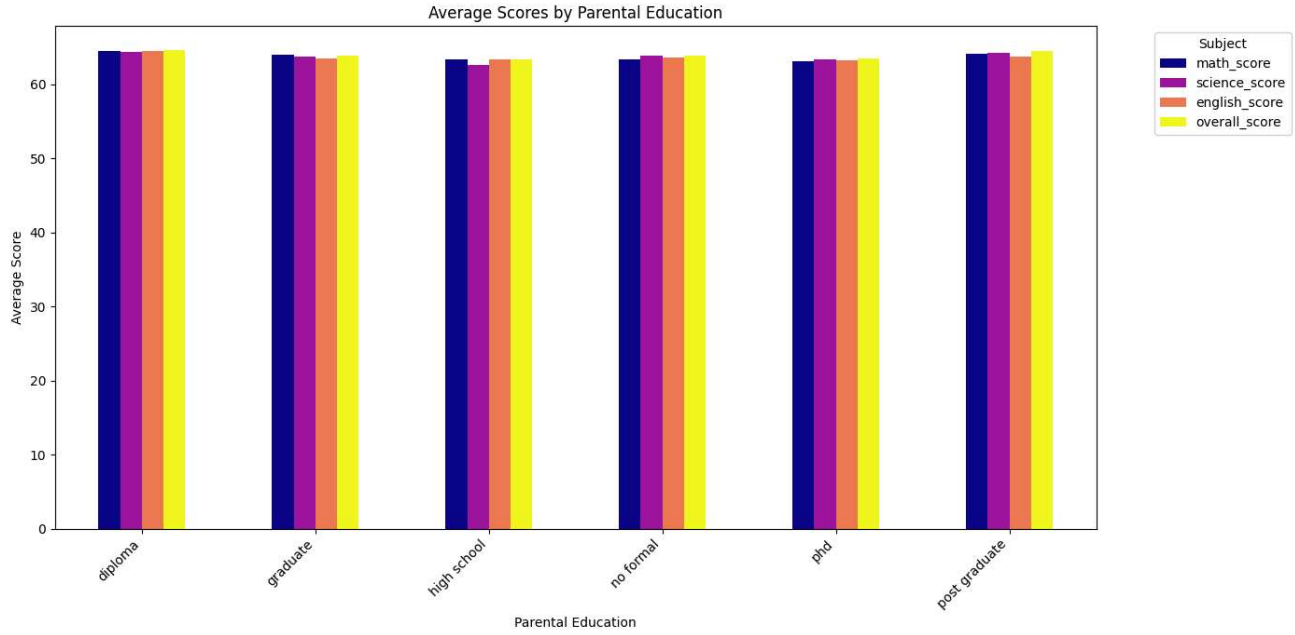
1. Group scores by gender.
2. Group scores by parental education.
3. Visualize group-wise average scores using bar charts.

```
import matplotlib.pyplot as plt
```

```
# Plot for Gender Grouped Scores
gender_grouped_scores.plot(kind='bar', figsize=(12, 6), colormap='viridis')
plt.title('Average Scores by Gender')
plt.xlabel('Gender')
plt.ylabel('Average Score')
plt.xticks(rotation=45)
plt.legend(title='Subject', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()

# Plot for Parental Education Grouped Scores
parent_education_grouped_scores.plot(kind='bar', figsize=(14, 7), colormap='plasma')
plt.title('Average Scores by Parental Education')
plt.xlabel('Parental Education')
plt.ylabel('Average Score')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Subject', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```






```
import matplotlib.pyplot as plt

# Plot for Gender Grouped Scores
gender_grouped_scores.plot(kind='bar', figsize=(12, 6), colormap='viridis')
plt.title('Average Scores by Gender')
plt.xlabel('Gender')
plt.ylabel('Average Score')
plt.xticks(rotation=45)
plt.legend(title='Subject', bbox_to_anchor=(1.05, 1), loc='upper left')
```