In [1]:
```
# Part 1: Dataset Importing & Cleaning
```

In [2]:
```
# Step 1: Import Libraries and Load the Dataset
```

In [3]:
```
# Step 2: Check Basic Information About the Dataset
```

In [4]:
```
import pandas as pd
import numpy as np

# Load the dataset from the provided path
file_path =r"D:\project Analysis\student_habits_performance.csv"
df = pd.read_csv(file_path)

# Display a sample of 5 rows from the dataset
print("🔹 Sample Data:")
print(df.sample(5))  # This shows 5 random rows
```

```
🔹 Sample Data:
    student_id  age  gender  study_hours_per_day  social_media_hours  \
914      S1914   20  Female                  2.6                 1.6
153      S1153   19  Female                  4.3                 3.1
294      S1294   18    Male                  4.3                 3.3
125      S1125   24    Male                  1.5                 3.1
10       S1010   19  Female                  4.6                 3.7

     netflix_hours part_time_job  attendance_percentage  sleep_hours  \
914            1.3           Yes                   87.4          5.2
153            0.7            No                   77.9          6.5
294            0.0            No                  100.0          6.8
125            3.5            No                   95.5          7.4
10             0.8            No                   77.6          5.8

    diet_quality  exercise_frequency parental_education_level  \
914         Good                   6                 Bachelor
153         Good                   0                      NaN
294         Fair                   6              High School
125         Fair                   0              High School
10          Fair                   1                      NaN

    internet_quality  mental_health_rating extracurricular_participation  \
914             Poor                    10                           No
153          Average                     6                           No
294          Average                     9                           No
125          Average                     3                           No
10              Good                     3                           No

     exam_score
914        72.9
153        81.4
294        94.9
125        44.0
10         63.3
```

In [5]:
```
# Step 3: Check for Missing Values
```

In [6]:
```
print("🔹 Missing Values per Column:")
print(df.isnull().sum())
```

```
🔹 Missing Values per Column:
student_id                          0
age                                 0
gender                              0
study_hours_per_day                 0
social_media_hours                  0
netflix_hours                       0
part_time_job                       0
attendance_percentage               0
sleep_hours                         0
diet_quality                        0
exercise_frequency                  0
parental_education_level           91
internet_quality                    0
mental_health_rating                0
extracurricular_participation       0
exam_score                          0
dtype: int64
```

In [7]:
```
# Step 4: Handle Missing Values
```

In [8]:
```
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns
df[numerical_columns] = df[numerical_columns].fillna(df[numerical_columns].mean(
```

In [9]:
```
categorical_columns = df.select_dtypes(include=['object']).columns
for col in categorical_columns:
    df[col] = df[col].fillna(df[col].mode()[0])
```

In [10]:
```
# Step 5: Confirm Missing Values Are Handled
```

In [11]:
```
print(" 🔹 Missing Values After Cleaning:")
print(df.isnull().sum())
```

```
🔹 Missing Values After Cleaning:
student_id                         0
age                                0
gender                             0
study_hours_per_day                0
social_media_hours                 0
netflix_hours                      0
part_time_job                      0
attendance_percentage              0
sleep_hours                        0
diet_quality                       0
exercise_frequency                 0
parental_education_level           0
internet_quality                   0
mental_health_rating               0
extracurricular_participation      0
exam_score                         0
dtype: int64
```

In [12]:
```
# Step 6 : Save the Cleaned Dataset
```

In [13]:
```
df.to_csv("cleaned_student_data.csv", index=False)
print(" Cleaned dataset saved as 'cleaned_student_data.csv'")
```

```
Cleaned dataset saved as 'cleaned_student_data.csv'
```

In [14]:
```
# Part 2: Exploratory Data Analysis (EDA)
```

In [15]:
```
# Step 1: Import Libraries
```

In [16]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [17]:
```
# Step 2: Load Cleaned Dataset
df = pd.read_csv("cleaned_student_data.csv")
```

In [18]:
```
# Step 3: Summary Statistics
print("🔹 Descriptive Statistics:")
print(df.describe())
```

```
🔹 Descriptive Statistics:
              age  study_hours_per_day  social_media_hours  netflix_hours  \
count  1000.0000           1000.00000         1000.000000    1000.000000
mean     20.4980              3.55010            2.505500       1.819700
std       2.3081              1.46889            1.172422       1.075118
min      17.0000              0.00000            0.000000       0.000000
25%      18.7500              2.60000            1.700000       1.000000
50%      20.0000              3.50000            2.500000       1.800000
75%      23.0000              4.50000            3.300000       2.525000
max      24.0000              8.30000            7.200000       5.400000

       attendance_percentage  sleep_hours  exercise_frequency  \
count            1000.000000  1000.000000         1000.000000
mean               84.131700     6.470100            3.042000
std                 9.399246     1.226377            2.025423
min                56.000000     3.200000            0.000000
25%                78.000000     5.600000            1.000000
50%                84.400000     6.500000            3.000000
75%                91.025000     7.300000            5.000000
max               100.000000    10.000000            6.000000

       mental_health_rating   exam_score
count           1000.000000  1000.000000
mean               5.438000    69.601500
std                2.847501    16.888564
min                1.000000    18.400000
25%                3.000000    58.475000
50%                5.000000    70.500000
75%                8.000000    81.325000
max               10.000000   100.000000
```

In [19]:
```
import pandas as pd

numeric_cols = df.select_dtypes(include='number')

statistics = numeric_cols.agg(['count', 'mean', 'std', 'min', 'median', 'max', '

statistics = statistics.transpose().round(4)
statistics.reset_index(inplace=True)
statistics.rename(columns={'index': 'Feature'}, inplace=True)

statistics
```

Out[19]:

	Feature	count	mean	std	min	median	max	skew	kurt
0	age	1000.0	20.4980	2.3081	17.0	20.0	24.0	0.0084	-1.2190
1	study_hours_per_day	1000.0	3.5501	1.4689	0.0	3.5	8.3	0.0543	-0.0557
2	social_media_hours	1000.0	2.5055	1.1724	0.0	2.5	7.2	0.1198	-0.0941
3	netflix_hours	1000.0	1.8197	1.0751	0.0	1.8	5.4	0.2372	-0.4329
4	attendance_percentage	1000.0	84.1317	9.3992	56.0	84.4	100.0	-0.2378	-0.3907
5	sleep_hours	1000.0	6.4701	1.2264	3.2	6.5	10.0	0.0915	-0.2143
6	exercise_frequency	1000.0	3.0420	2.0254	0.0	3.0	6.0	-0.0319	-1.2765
7	mental_health_rating	1000.0	5.4380	2.8475	1.0	5.0	10.0	0.0378	-1.1886
8	exam_score	1000.0	69.6015	16.8886	18.4	70.5	100.0	-0.1564	-0.4199

In [20]:

```
import plotly.graph_objects as go

fig_stats = go.Figure(
    data=[go.Table(
        header=dict(values=list(statistics.columns),
                    fill_color='royalblue',
                    font=dict(color='white', size=14),
                    align='left'),
        cells=dict(values=[statistics[col] for col in statistics.columns],
                   fill_color='lavender',
                   align='left',
                   font=dict(size=12))
    )]
)

fig_stats.update_layout(title="Descriptive Statistics")
fig_stats.show()
```

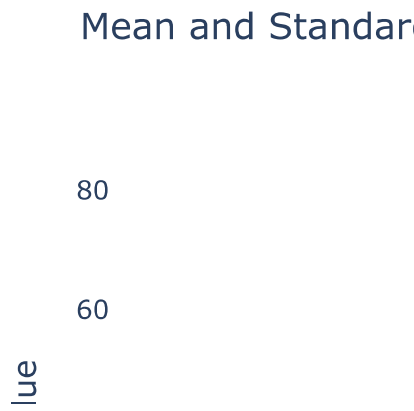
Descriptive Statistics

Feature	count	mean	std
age	1000	20.498	2.3081
study_hours_per_d	1000	3.5501	1.4689
social_media_hours	1000	2.5055	1.1724
netflix_hours	1000	1.8197	1.0751
attendance_percent	1000	84.1317	9.3992

In [21]:
```
# Select only numerical columns for statistical analysis
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Calculate mean and standard deviation
mean_values = df[numeric_columns].mean()
std_values = df[numeric_columns].std()

# Create a DataFrame for plotting
mean_std_df = pd.DataFrame({
    'Feature': numeric_columns,
    'mean': mean_values.values,
    'std': std_values.values
})

# Melt the DataFrame to long format for Plotly
mean_std_df_melted = mean_std_df.melt(id_vars='Feature',
                                      var_name='Statistic',
                                      value_name='Value')
```

In [22]:
```
import plotly.express as px

color_map = {'mean': '#1f77b4', 'std': '#ff7f0e'}

fig_mean_std = px.bar(
    mean_std_df_melted,
    x='Feature',
    y='Value',
```

```
    color='Statistic',
    color_discrete_map=color_map,
    barmode='group',
    title='Mean and Standard Deviation by Feature',
    labels={"Value": "Value", "Feature": "Feature", "Statistic": "Statistic"}
)
fig_mean_std.update_layout(
    xaxis_tickangle=-45,
    plot_bgcolor='white',
    title_font=dict(size=18),
    font=dict(size=13)
)
fig_mean_std.show()
```

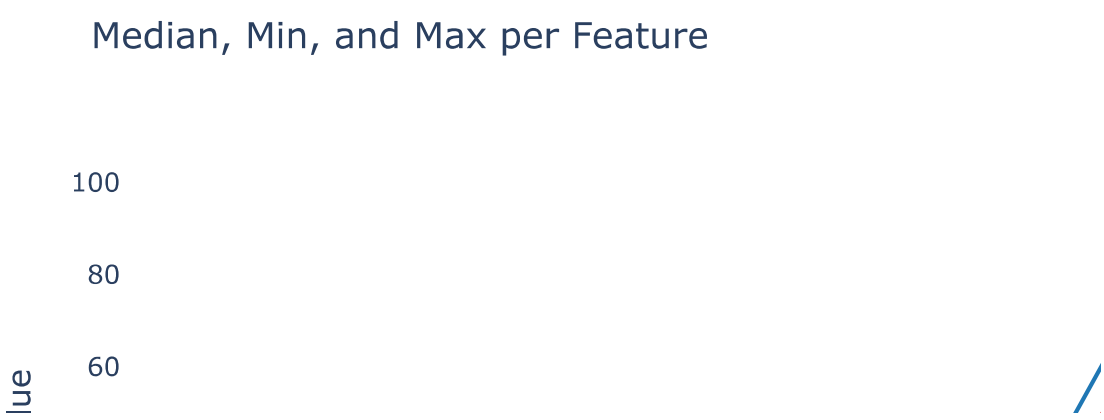


In [23]:
```
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns

median_values = df[numeric_columns].median()
min_values = df[numeric_columns].min()
max_values = df[numeric_columns].max()

median_min_max_df = pd.DataFrame({
    'Feature': numeric_columns,
    'median': median_values.values,
    'min': min_values.values,
    'max': max_values.values
})

median_min_max_melted = median_min_max_df.melt(
```

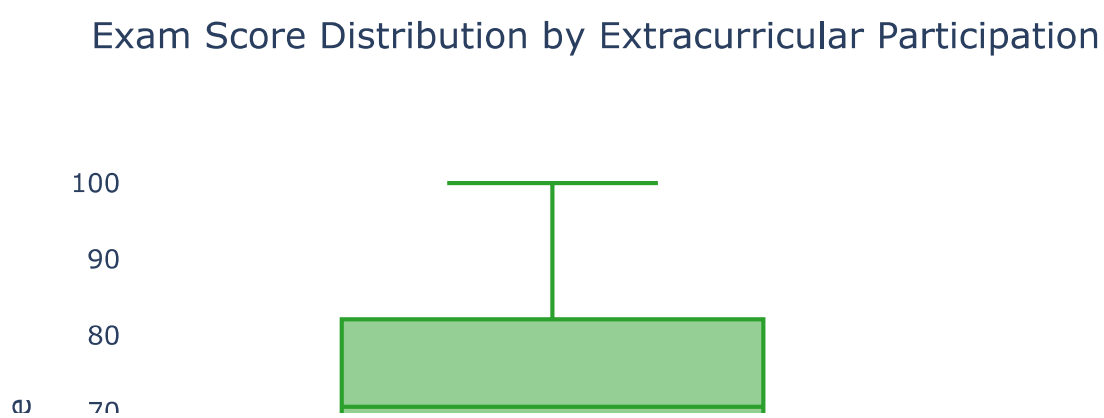
```
    id_vars='Feature',
    var_name='Statistic',
    value_name='Value'
)
```

In [24]:

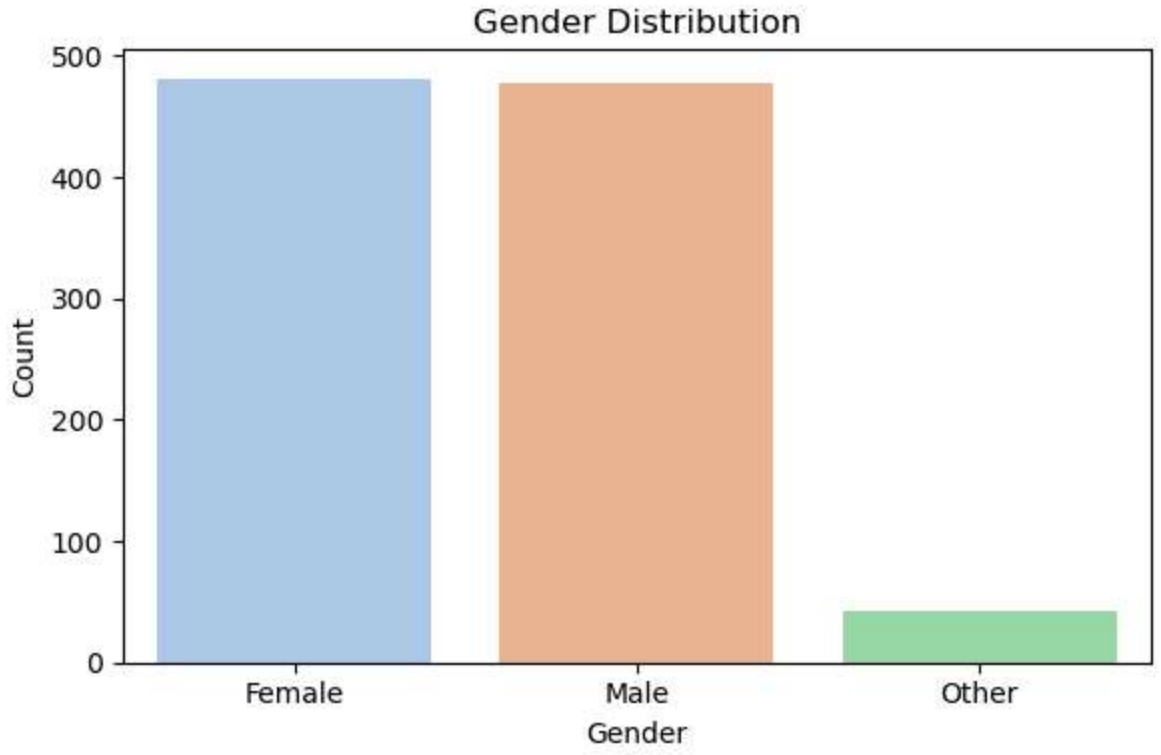
```
import plotly.express as px

color_map = {
    'min': '#2ca02c',
    'median': '#d62728',
    'max': '#1f77b4'
}

fig_median = px.line(
    median_min_max_melted,
    x='Feature',
    y='Value',
    color='Statistic',
    color_discrete_map=color_map,
    markers=True,
    title='Median, Min, and Max per Feature'
)

fig_median.update_layout(
    xaxis_tickangle=-45,
    plot_bgcolor='white',
    title_font=dict(size=18),
    font=dict(size=13)
)

fig_median.show()
```



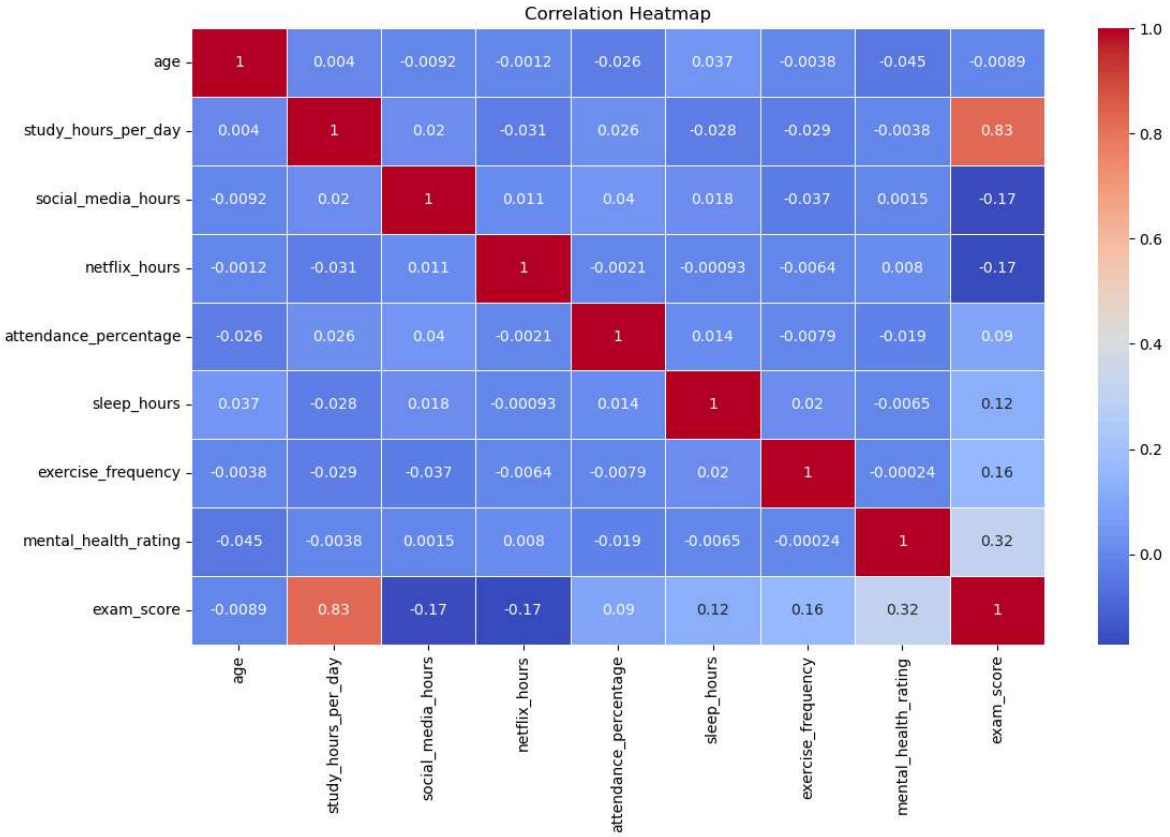
In [25]:
```
import plotly.express as px

fig_box = px.box(
    df,
    x='extracurricular_participation',
    y='exam_score',
    color='extracurricular_participation',
    title='Exam Score Distribution by Extracurricular Participation',
    labels={
        'extracurricular_participation': 'Participates in Extracurricular Activi
        'exam_score': 'Exam Score'
    },
    color_discrete_map={'Yes': '#2ca02c', 'No': '#d62728'}
)

fig_box.update_layout(
    plot_bgcolor='white',
    title_font=dict(size=18),
    font=dict(size=13)
)
fig_box.show()
```



In [26]:
```
plt.figure(figsize=(6, 4))
sns.countplot(x='gender', data=df, hue='gender', palette='pastel')
plt.title("Gender Distribution")
plt.xlabel("Gender")
plt.ylabel("Count")
plt.tight_layout()
plt.show()
```

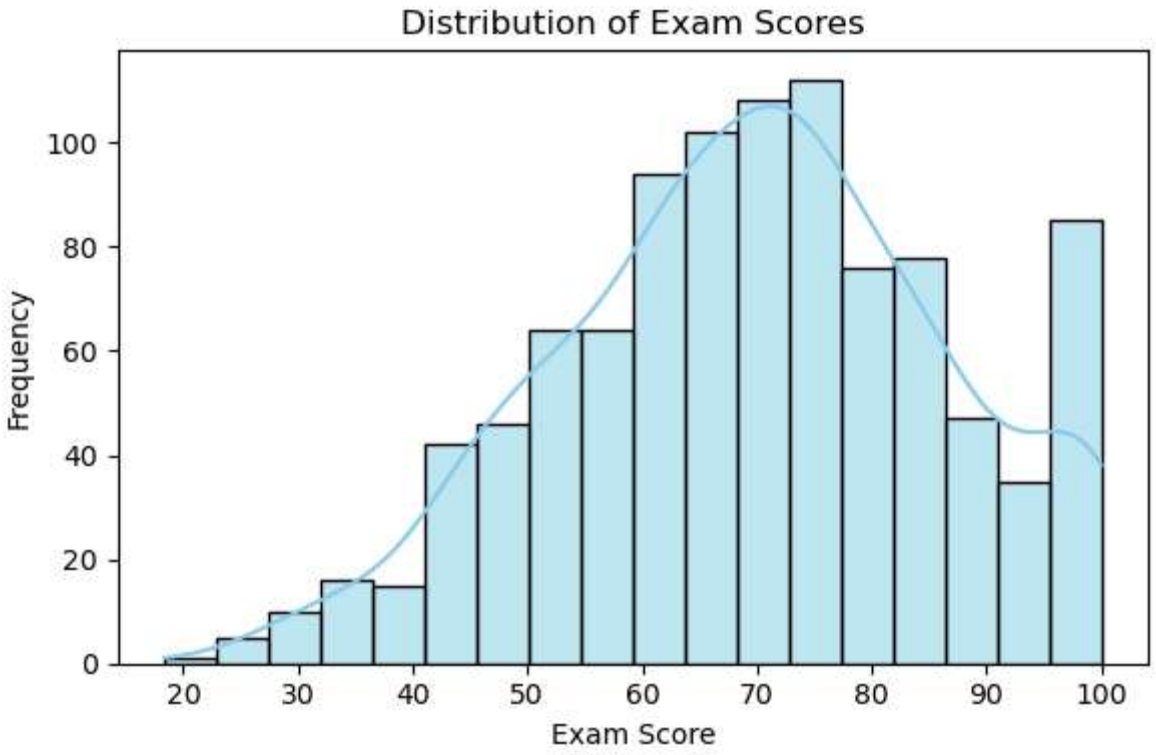


In [27]:
```
# Step 5: Correlation Heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm', linewidths=
plt.title("Correlation Heatmap")
plt.tight_layout()
plt.show()
```

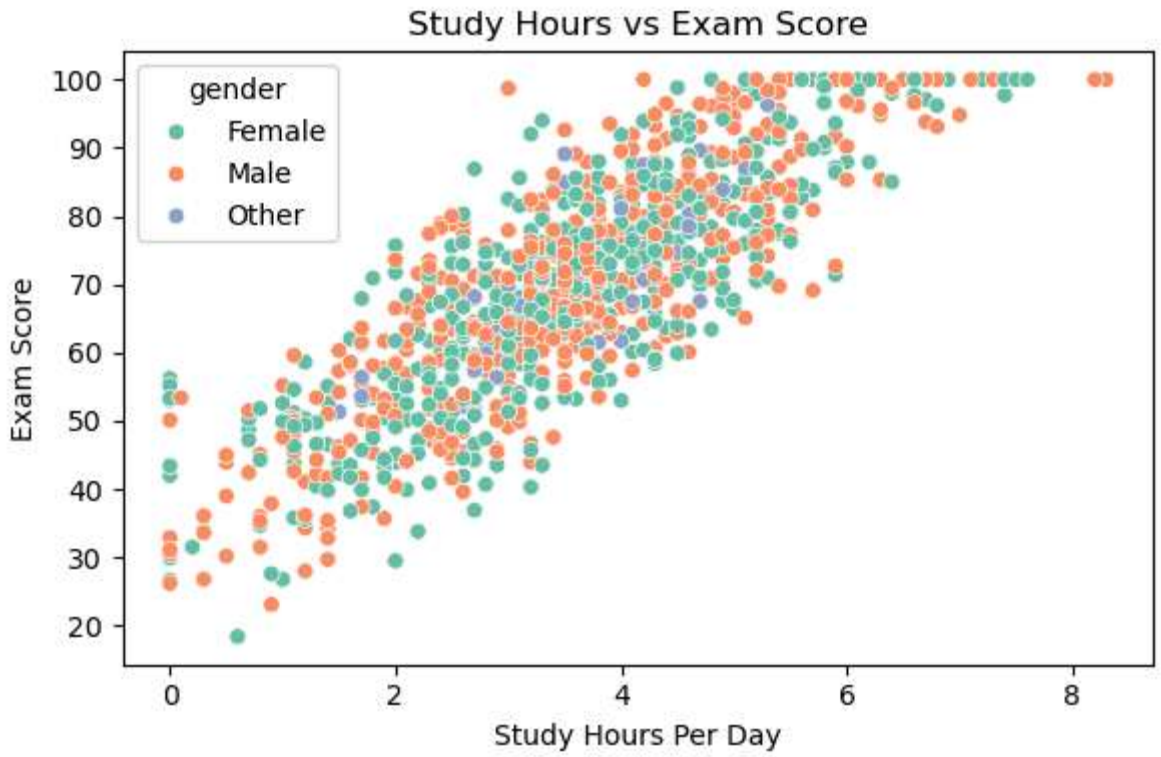


In [28]:
```
# Step 6: Histogram of Exam Scores
plt.figure(figsize=(6, 4))
sns.histplot(df['exam_score'], kde=True, color='skyblue')
```

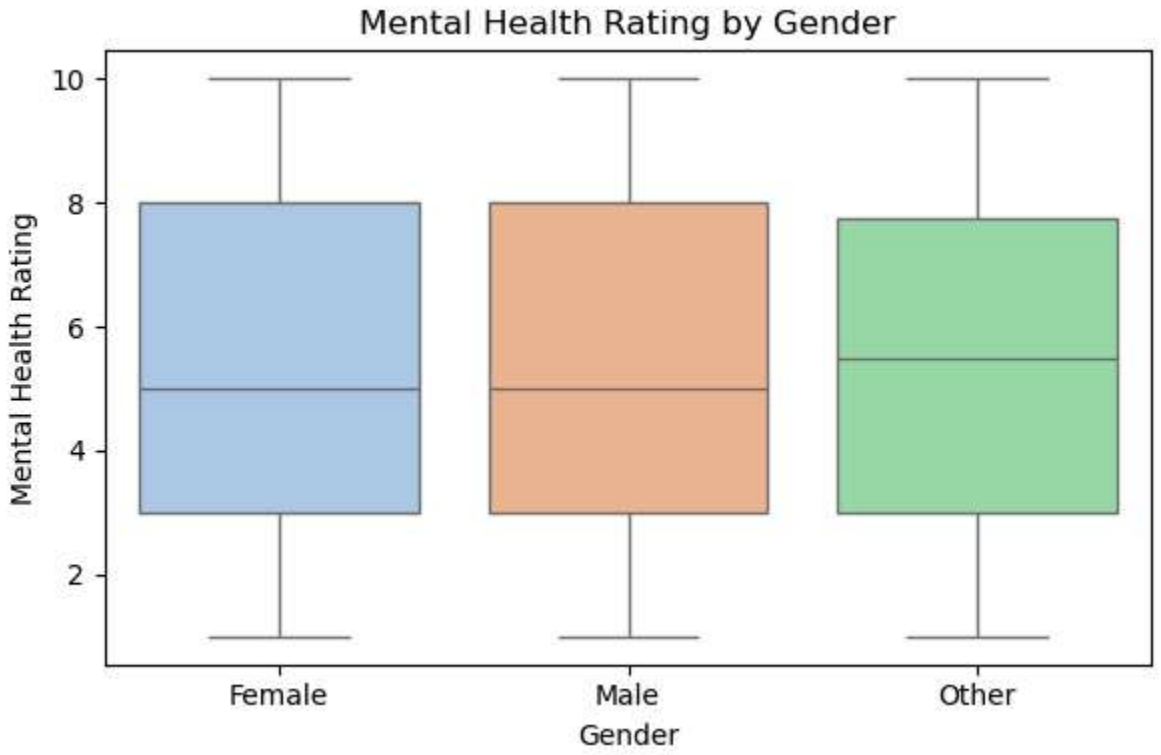
```
plt.title("Distribution of Exam Scores")
plt.xlabel("Exam Score")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```



In [29]:
```
# Step 7: Scatter Plot – Study Hours vs Exam Score
plt.figure(figsize=(6, 4))
sns.scatterplot(x='study_hours_per_day', y='exam_score', data=df, hue='gender',
plt.title("Study Hours vs Exam Score")
plt.xlabel("Study Hours Per Day")
plt.ylabel("Exam Score")
plt.tight_layout()
plt.show()
```



In [30]:
```
#  Step 8: Boxplot – Mental Health Rating by Gender
plt.figure(figsize=(6, 4))
sns.boxplot(data=df, x='gender', y='mental_health_rating', hue='gender', palette
plt.title("Mental Health Rating by Gender")
plt.xlabel("Gender")
plt.ylabel("Mental Health Rating")
plt.tight_layout()
plt.show()
```



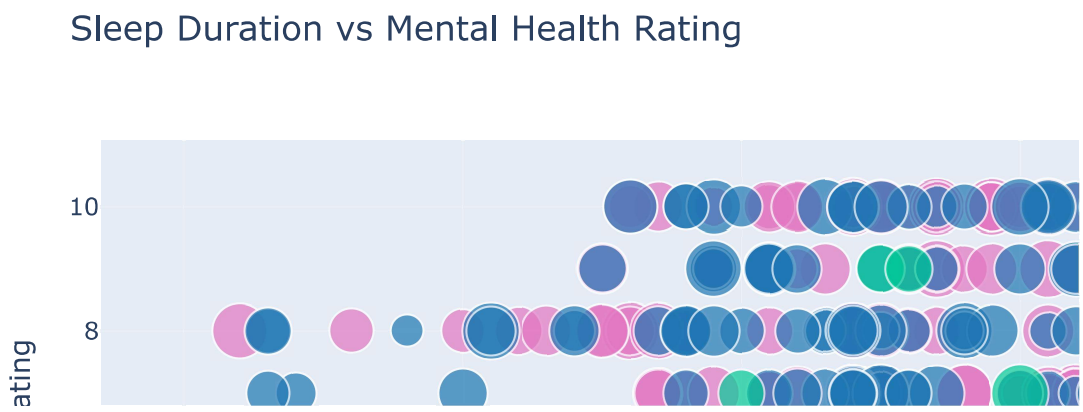
In [31]:
```
# Part 3: Main Visualization
# Visualization 1: Study Hours vs Exam Score
```

In [32]:
```
import plotly.express as px


custom_colors = {
    "Male": "#1f77b4",
    "Female": "#e377c2"
}

fig1 = px.scatter(
    df,
    x="study_hours_per_day",
    y="exam_score",
    color="gender",
    hover_data=["student_id", "sleep_hours", "mental_health_rating"],
    title="Relationship Between Study Hours and Exam Score",
    labels={"study_hours_per_day": "Study Hours Per Day", "exam_score": "Exam Sc
    color_discrete_map=custom_colors
)

fig1.update_traces(marker=dict(size=10, opacity=0.7))
fig1.show()
```

In [33]:
```
# Visualization 2: Sleep Hours vs Mental Health
```

In [34]:
```
import plotly.express as px

custom_colors = {
```

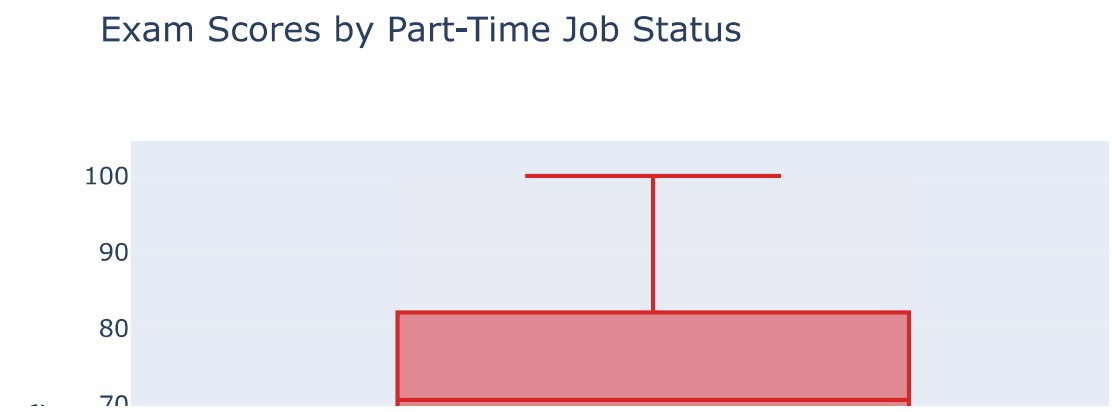
```
    "Male": "#1f77b4",
    "Female": "#e377c2"
}

fig2 = px.scatter(
    df,
    x="sleep_hours",
    y="mental_health_rating",
    size="exam_score",
    color="gender",
    hover_name="student_id",
    title="Sleep Duration vs Mental Health Rating",
    labels={
        "sleep_hours": "Sleep Hours",
        "mental_health_rating": "Mental Health Rating"
    },
    color_discrete_map=custom_colors
)

fig2.update_layout(showlegend=True)
fig2.show()
```



In [35]:
```
# Visualization 3: Exam Scores by Part-Time Job Status
```

In [36]:
```
import plotly.express as px


custom_colors = {
```

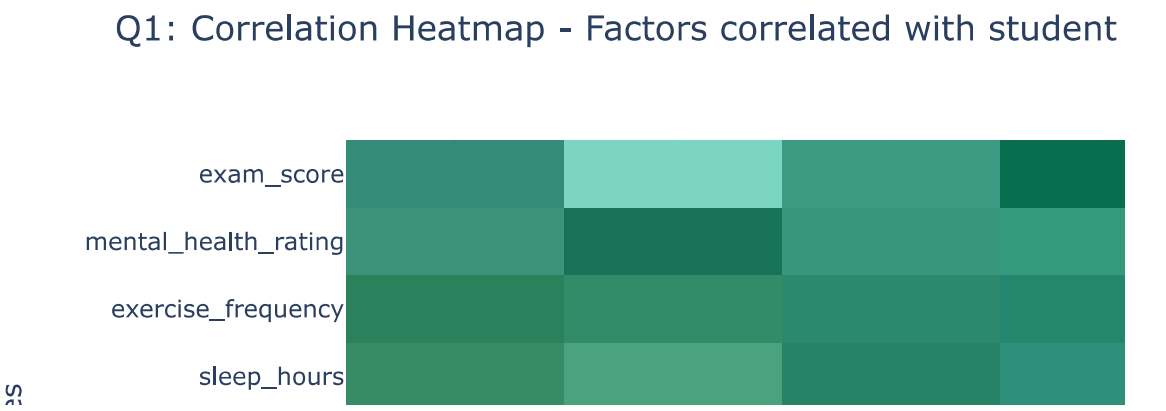
```
    "Yes": "#2ca02c",
    "No": "#d62728"
}

fig3 = px.box(
    df,
    x="part_time_job",
    y="exam_score",
    color="part_time_job",
    title="Exam Scores by Part-Time Job Status",
    labels={
        "part_time_job": "Has Part-Time Job",
        "exam_score": "Exam Score"
    },
    color_discrete_map=custom_colors
)

fig3.show()
```



In [38]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the cleaned student dataset
df = pd.read_csv("cleaned_student_data.csv")
```

In [39]:
```
# Q1: Which factors correlate most with student performance?
```

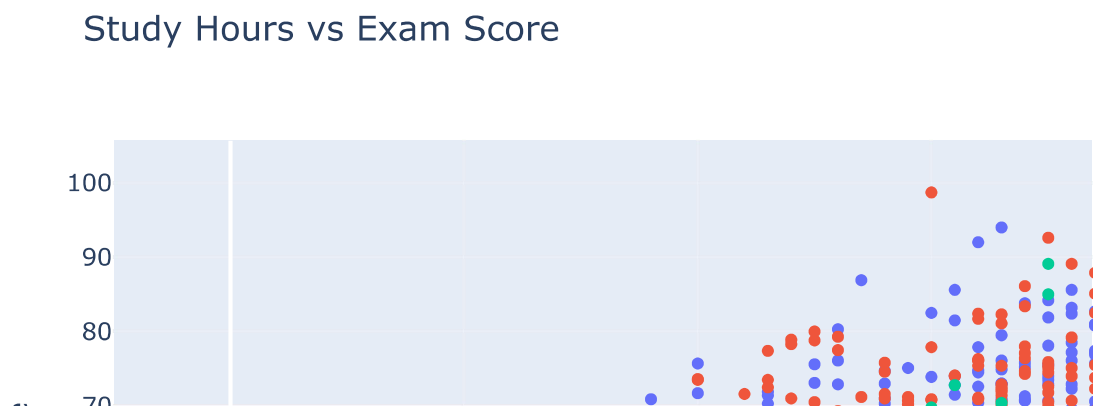
In [40]:
```
import plotly.graph_objects as go

corr_matrix = df.corr(numeric_only=True).round(2)

fig_q1 = go.Figure(data=go.Heatmap(
    z=corr_matrix.values,
    x=corr_matrix.columns,
    y=corr_matrix.index,
    colorscale='Viridis',
    zmin=-1, zmax=1,
    colorbar=dict(title='Correlation')
))
fig_q1.update_layout(
    title='Q1: Correlation Heatmap - Factors correlated with student performance
    xaxis_title='Features',
    yaxis_title='Features'
)
fig_q1.show()
```



In [41]:
```
# Q2: Is  Does study time affect exam scores?
```

In [42]:
```
fig = px.scatter(
    df,
    x='study_hours_per_day',
    y='exam_score',
    color='gender',
    title='Study Hours vs Exam Score',
```

```
    labels={'study_hours_per_day': 'Study Hours per Day', 'exam_score': 'Exam Sc
    hover_data=['student_id', 'age']
)
fig.show()
```

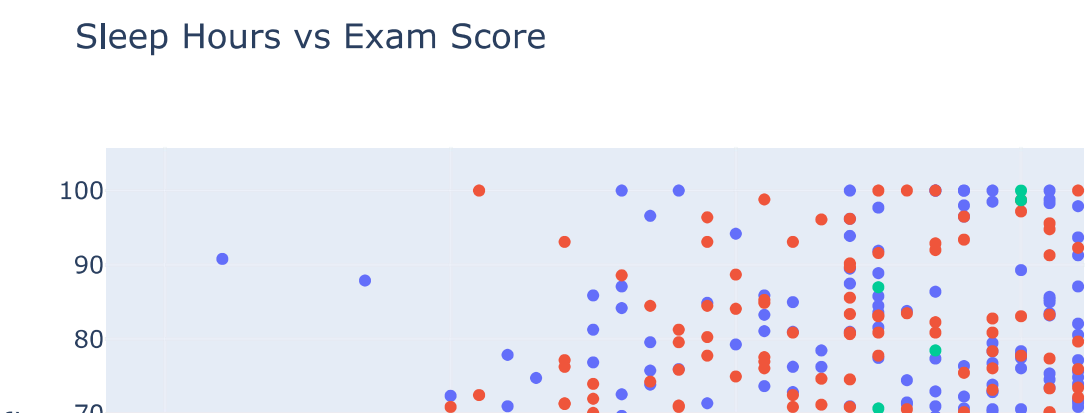


In [43]:
```
# Q3 : Do students who participate in extracurricular activities score higher?
```

In [44]:
```
fig = px.box(
    df,
    x='extracurricular_participation',
    y='exam_score',
    color='extracurricular_participation',
    title='Extracurricular Activities vs Exam Score'
)
fig.show()
```

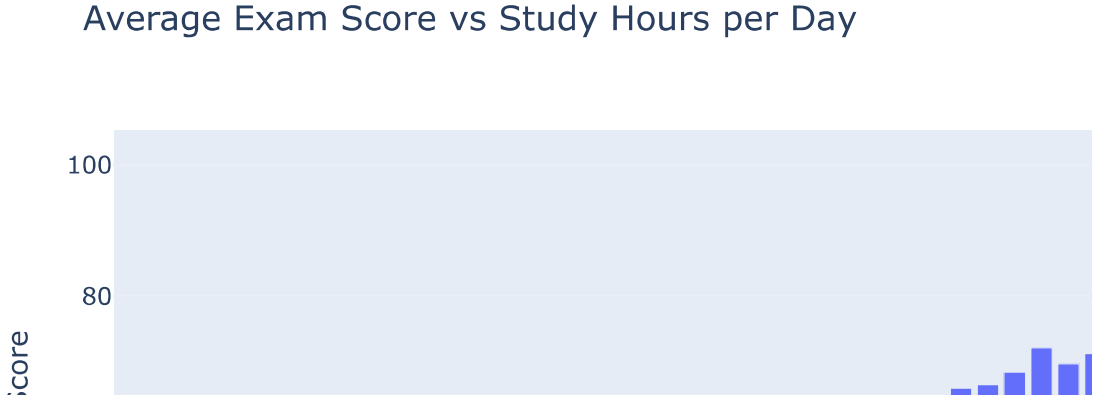
In [45]:
```
# Q4 Does amount of sleep impact exam performance?
```

In [46]:
```
fig = px.scatter(
    df,
    x='sleep_hours',
    y='exam_score',
    color='gender',
    title='Sleep Hours vs Exam Score',
    labels={'sleep_hours': 'Sleep Hours', 'exam_score': 'Exam Score'},
    hover_data=['student_id', 'diet_quality']
)
fig.show()
```



In [47]:
```
# Q5 :  Does study time affect exam scores?
```

In [48]:
```
# Calculate average exam score per study hours
study_score_avg = df.groupby('study_hours_per_day')['exam_score'].mean().reset_i

# Bar chart
fig = px.bar(
    study_score_avg,
    x='study_hours_per_day',
    y='exam_score',
    title='Average Exam Score vs Study Hours per Day',
    labels={'study_hours_per_day': 'Study Hours per Day', 'exam_score': 'Average
)
fig.show()
```

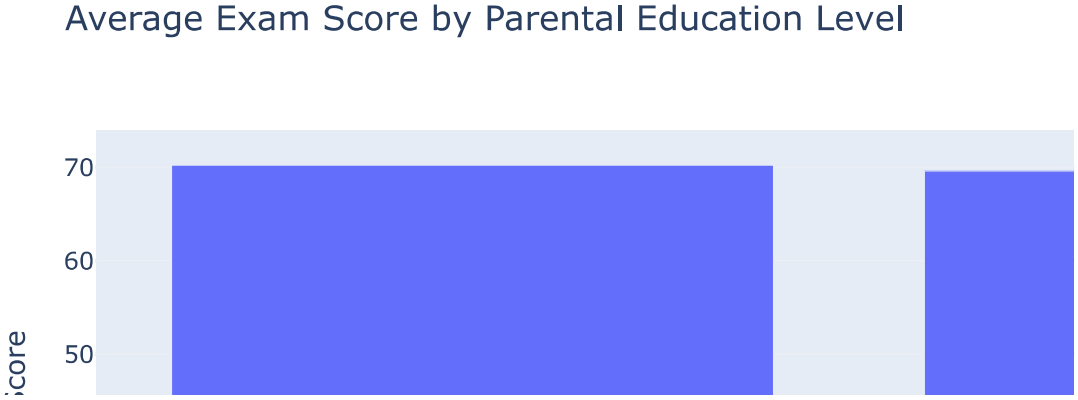


In [49]:
```
#Q6 : Is internet quality related to academic performance?
```

In [50]:
```
# Calculate average exam score by internet quality
internet_score_avg = df.groupby('internet_quality')['exam_score'].mean().reset_i

# Bar chart
fig = px.bar(
    internet_score_avg,
    x='internet_quality',
    y='exam_score',
    title='Average Exam Score by Internet Quality',
    labels={'internet_quality': 'Internet Quality', 'exam_score': 'Average Exam
)
fig.show()
```

In [51]:
```
# Q4: Does parental education level affect student exam scores?
```

In [52]:
```
# Calculate average exam score by parental education level
parental_education_score_avg = df.groupby('parental_education_level')['exam_scor

# Bar chart
fig = px.bar(
    parental_education_score_avg,
    x='parental_education_level',
    y='exam_score',
    title='Average Exam Score by Parental Education Level',
    labels={'exam_score': 'Average Exam Score'}
)
fig.show()
```


Average Exam Score by Parental Education Level
70
60
50
icore

In [ ]: