Assignment 3

1. Initilisation and Intial Checks

In [1]:

```
#Step 1: Import the Pandas and NumPy Library

import pandas as pd
import numpy as np
# Import the MatPlotLib and the Seaborn library
import matplotlib.pyplot as plt
import seaborn as sns

print("Libraries imported successfully!")
```

```
Libraries imported successfully!
```

In [2]:

```
#Step 2: Mount Google Drive to give Collab access to the dataset

from google.colab import drive

try:
    drive.mount('/content/drive')
    print("Google Drive mounted successfully!")

except Exception as e:
    print(f"An error occurred during mounting: {e}") #Easier to understand the error
```

```
Mounted at /content/drive
Google Drive mounted successfully!
```

In [3]:

```
# Step 3: Read the Kaggle dataset (csv) using read_csv function in Pandas

dataset_path = '/content/drive/MyDrive/Python
Class/Students_Grading_Dataset.csv' #Path from drive set

try:
       df = pd.read_csv(dataset_path, header=0) #Reading the CSV File
       print("CSV file loaded successfully!")

except Exception as e:
       print(f"An unexpected error occurred: {e}") #Easier to understand
the error
```

```
CSV file loaded successfully!
```

In [4]:

```
# Step 4: Prove it works by displaying top 5
try:
    print("Top 5 rows:\n")
    print(df.head())

    print("\nBottom 5 rows:\n")
    print(df.tail())

except Exception as e:
    print(f"An error occurred while displaying the data: {e}") #Easier to
understand the error
```

```
Top 5 rows:

  Student_ID First_Name Last_Name                    Email  Gender
Age  \
0      S1000       Omar  Williams  student0@university.com  Female
22
1      S1001      Maria     Brown  student1@university.com    Male
18
2      S1002      Ahmed     Jones  student2@university.com    Male
24
3      S1003       Omar  Williams  student3@university.com  Female
24
4      S1004       John     Smith  student4@university.com  Female
23

    Department  Attendance (%)  Midterm_Score  Final_Score  ...  \
0  Engineering           52.29          55.03        57.82  ...
1  Engineering           97.27          97.23        45.80  ...
2     Business           57.19          67.05        93.68  ...
3  Mathematics           95.15          47.79        80.63  ...
4           CS           54.18          46.59        78.89  ...

   Projects_Score  Total_Score  Grade  Study_Hours_per_Week  \
0           85.90        56.09      F                   6.2
1           55.65        50.64      A                  19.0
2           73.79        70.30      D                  20.7
3           92.12        61.63      A                  24.8
4           68.42        66.13      F                  15.4

   Extracurricular_Activities Internet_Access_at_Home
Parent_Education_Level  \
0                          No                     Yes
High School
1                          No                     Yes
NaN
2                          No                     Yes
Master's
3                         Yes                     Yes
High School
4                         Yes                     Yes
High School
```

```
  Family_Income_Level Stress_Level (1-10) Sleep_Hours_per_Night
0              Medium                   5                   4.7
1              Medium                   4                   9.0
2                 Low                   6                   6.2
3                High                   3                   6.7
4                High                   2                   7.1

[5 rows x 23 columns]

Bottom 5 rows:

     Student_ID First_Name Last_Name                       Email
Gender  Age  \
4995      S5995      Ahmed     Jones  student4995@university.com
Male   19
4996      S5996       Emma     Brown  student4996@university.com
Male   19
4997      S5997       John     Brown  student4997@university.com
Female   24
4998      S5998       Sara     Davis  student4998@university.com
Male   23
4999      S5999      Maria     Brown  student4999@university.com
Female   21

       Department  Attendance (%)  Midterm_Score  Final_Score  ...  \
4995     Business             NaN          82.15        60.33  ...
4996     Business           65.11          86.31        49.80  ...
4997           CS           87.54          63.55        64.21  ...
4998           CS           92.56          79.79        94.28  ...
4999  Engineering           83.92          83.24        53.47  ...

      Projects_Score  Total_Score  Grade  Study_Hours_per_Week  \
4995           58.42        85.21      D                  25.5
4996           60.87        95.96      C                   5.0
4997           82.65        54.25      A                  24.8
4998           94.29        55.84      A                  16.1
4999           69.25        77.86      F                  29.2

      Extracurricular_Activities Internet_Access_at_Home  \
4995                          No                     Yes
4996                          No                     Yes
4997                         Yes                      No
4998                         Yes                     Yes
4999                          No                     Yes

      Parent_Education_Level Family_Income_Level Stress_Level (1-10)
\
4995             High School                 Low                  10
4996                     NaN              Medium                   4
4997             High School              Medium                   4
4998              Bachelor's                 Low                   1
4999                     PhD                 Low                   2
```

```
     Sleep_Hours_per_Night
4995                   8.3
4996                   4.0
4997                   6.3
4998                   8.4
4999                   6.1

[5 rows x 23 columns]
```

2. Initial Data Exploration

In [5]:

```
# Step 5: Find out how many students are in the dataset and how many
variables are present in the dataset

try:

  print("The row and column count details about the dataset are as
follows:", df.shape)

  print("\nThe number of student records available are:", df.shape[0])
#prints first part of the shape function output array

  print("The number of variables are:", df.shape[1]) #prints second part of
the shape function output array

except Exception as e:
    print(f"An error occurred while displaying the data: {e}") #Easier to
understand the error
```

```
The row and column count details about the dataset are as follows:
(5000, 23)

The number of student records available are: 5000
The number of variables are: 23
```

In [6]:

```
#Step 6: Understanding the columns

try:
  print(df.columns.values)

except Exception as e:
    print(f"An error occurred while displaying the data: {e}") #Easier to
understand the error
```

```
['Student_ID' 'First_Name' 'Last_Name' 'Email' 'Gender' 'Age'
'Department'
 'Attendance (%)' 'Midterm_Score' 'Final_Score' 'Assignments_Avg'
 'Quizzes_Avg' 'Participation_Score' 'Projects_Score' 'Total_Score'
 'Grade' 'Study_Hours_per_Week' 'Extracurricular_Activities'
 'Internet_Access_at_Home' 'Parent_Education_Level'
'Family_Income_Level'
 'Stress_Level (1-10)' 'Sleep_Hours_per_Night']
```

In [7]:

```
# Step 7: Rename variables to more relevant and readable names

try:
    new_column_names = {
        'Student_ID': 'Student ID',
        'First_Name': 'First Name',
        'Last_Name': 'Last Name',
        'Midterm_Score': 'Midterm Score',
        'Final_Score': 'Final Score',
        'Assignments_Avg': 'Assignments Score (Averaged)',
        'Quizzes_Avg': 'Quizzes Score (Averaged)',
        'Participation_Score':'Participation Score',
        'Projects_Score':'Projects Score',
        'Total_Score': 'Total Score',
        'Study_Hours_per_Week': 'Study Hours per Week',
        'Extracurricular_Activities': 'Extracurricular Activities',
        'Internet_Access_at_Home': 'Internet Access at Home',
        'Parent_Education_Level': 'Parent Education Level',
        'Family_Income_Level': 'Family Income Level',
        'Stress_Level (1-10)': 'Stress Level (1-10)',
        'Sleep_Hours_per_Night': 'Sleep Hours per Night'
    }
    df = df.rename(columns=new_column_names)
    print("Column names updated successfully!")
    print("The updated Column names are:\n", df.columns.values) #Verify the
changes
except Exception as e:
    print(f"An error occurred while renaming columns: {e}")
```

```
Column names updated successfully!
The updated Column names are:
 ['Student ID' 'First Name' 'Last Name' 'Email' 'Gender' 'Age'
'Department'
 'Attendance (%)' 'Midterm Score' 'Final Score'
 'Assignments Score (Averaged)' 'Quizzes Score (Averaged)'
 'Participation Score' 'Projects Score' 'Total Score' 'Grade'
 'Study Hours per Week' 'Extracurricular Activities'
 'Internet Access at Home' 'Parent Education Level' 'Family Income
Level'
 'Stress Level (1-10)' 'Sleep Hours per Night']
```

In [8]:

```
# Step 8: Create a Dictionary called Variable Details to explain more about
each of the variables

variable_details = {
    'Student ID': 'Unique identifier for each student.',
    'First Name': "Student's first name.",
    'Last Name': "Student's last name.",
    'Email': 'Contact email (can be anonymized).',
    'Gender': 'Male, Female, Other.',
    'Age': 'The age of the student.',
    'Department': "Student's department (e.g., CS, Engineering,
Business).",
    'Attendance (%)': 'Attendance percentage (0-100%).',
    'Midterm Score': 'Midterm exam score (out of 100).',
    'Final Score': 'Final exam score (out of 100).',
    'Assignments Score (Averaged)': 'Average score of all assignments (out
of 100).',
    'Quizzes Score (Averaged)': 'Average quiz scores (out of 100).',
    'Participation Score': 'Score based on class participation (0-10).',
    'Projects Score': 'Project evaluation score (out of 100).',
    'Total Score': 'Weighted sum of all grades.',
    'Grade': 'Letter grade (A, B, C, D, F).',
    'Study Hours per Week': 'Average study hours per week.',
    'Extracurricular Activities': 'Whether the student participates in
extracurriculars (Yes/No).',
    'Internet Access at Home': 'Does the student have access to the
internet at home? (Yes/No).',
    'Parent Education Level': 'Highest education level of parents (None,
High School, Bachelor\'s, Master\'s, PhD).',
    'Family Income Level': 'Low, Medium, High.',
    'Stress Level (1-10)': 'Self-reported stress level (1: Low, 10:
High).',
    'Sleep Hours per Night': 'Average hours of sleep per night.'
} #Creating a dictionary to store the explanation of all the variables
print("Variable Details Dictionary-\n")

# Display the variable details in a user-friendly format
for column, description in variable_details.items(): #.items() function
allows us to read the key and value inividually
    print(f"{column}: {description}")

#https://discuss.python.org/t/printing-the-elements-of-a-dictionary-that-
is-contained-in-a-list-how/13568
```

```
Variable Details Dictionary-

Student ID: Unique identifier for each student.
First Name: Student's first name.
Last Name: Student's last name.
Email: Contact email (can be anonymized).
Gender: Male, Female, Other.
Age: The age of the student.
Department: Student's department (e.g., CS, Engineering, Business).
```

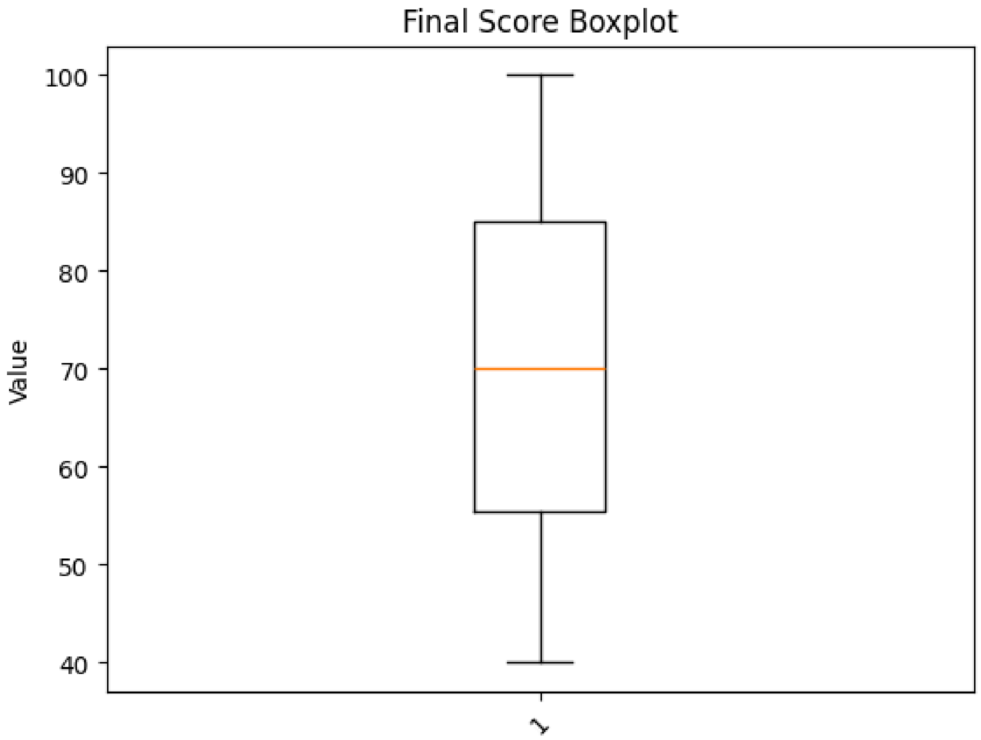
```
Attendance (%): Attendance percentage (0-100%).
Midterm Score: Midterm exam score (out of 100).
Final Score: Final exam score (out of 100).
Assignments Score (Averaged): Average score of all assignments (out
of 100).
Quizzes Score (Averaged): Average quiz scores (out of 100).
Participation Score: Score based on class participation (0-10).
Projects Score: Project evaluation score (out of 100).
Total Score: Weighted sum of all grades.
Grade: Letter grade (A, B, C, D, F).
Study Hours per Week: Average study hours per week.
Extracurricular Activities: Whether the student participates in
extracurriculars (Yes/No).
Internet Access at Home: Does the student have access to the internet
at home? (Yes/No).
Parent Education Level: Highest education level of parents (None,
High School, Bachelor's, Master's, PhD).
Family Income Level: Low, Medium, High.
Stress Level (1-10): Self-reported stress level (1: Low, 10: High).
Sleep Hours per Night: Average hours of sleep per night.
```

In [9]:

```
# Step 9: Identify the quality issues in the dataset to provide a
comprehensive overview of its integrity and completeness.
try:
    print("Columns with Data missing:")
    print(df.isnull().sum())
    print("\nNumber of duplicates:", df.duplicated().sum())
    print("\nCheck for Columns with incorrect data types:")
    print(df.dtypes)
    print("\nCheck for outliers:")
    plt.boxplot(df['Final Score'])
    plt.title('Final Score Boxplot')
    plt.ylabel('Value')
    plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
    plt.show()
    print("\nSimiliarly.....Checking for Outliers in other numeric
fields:\n")
    for col in df.select_dtypes(include=["number"]).columns:
      Q1 = np.percentile(df[col], 25)
      Q3 = np.percentile(df[col], 75)
      IQR = Q3 - Q1
      lower_bound = Q1 - 1.5 * IQR
      upper_bound = Q3 + 1.5 * IQR

      # Identify outliers
      outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]

      # Print the number of outliers for the current column
      print(f"Number of outliers in '{col}': {len(outliers)}")

except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

```
Columns with Data missing:
Student ID                       0
First Name                       0
Last Name                        0
Email                            0
Gender                           0
Age                              0
Department                       0
Attendance (%)                 515
Midterm Score                    0
Final Score                      0
Assignments Score (Averaged)   503
Quizzes Score (Averaged)         0
Participation Score              0
Projects Score                   0
Total Score                      0
Grade                            0
Study Hours per Week             0
Extracurricular Activities       0
Internet Access at Home          0
Parent Education Level        1800
```

```
Family Income Level                0
Stress Level (1-10)                0
Sleep Hours per Night              0
dtype: int64

Number of duplicates: 757

Check for Columns with incorrect data types:
Student ID                       object
First Name                       object
Last Name                        object
Email                            object
Gender                           object
Age                               int64
Department                       object
Attendance (%)                  float64
Midterm Score                   float64
Final Score                     float64
Assignments Score (Averaged)    float64
Quizzes Score (Averaged)        float64
Participation Score             float64
Projects Score                  float64
Total Score                     float64
Grade                            object
Study Hours per Week            float64
Extracurricular Activities       object
Internet Access at Home          object
Parent Education Level           object
Family Income Level              object
Stress Level (1-10)               int64
Sleep Hours per Night           float64
dtype: object

Check for outliers:
```



```
Similiarly.....Checking for Outliers in other numeric fields:

Number of outliers in 'Age': 0
Number of outliers in 'Attendance (%)': 0
Number of outliers in 'Midterm Score': 0
Number of outliers in 'Final Score': 0
Number of outliers in 'Assignments Score (Averaged)': 0
Number of outliers in 'Quizzes Score (Averaged)': 0
Number of outliers in 'Participation Score': 0
Number of outliers in 'Projects Score': 0
Number of outliers in 'Total Score': 0
Number of outliers in 'Study Hours per Week': 0
Number of outliers in 'Stress Level (1-10)': 0
Number of outliers in 'Sleep Hours per Night': 0
```

In [10]:

```
#Step 9.1: Analyze plan of action to handle missing values
try:
  missing_cols = ['Attendance (%)', 'Assignments Score (Averaged)', 'Parent
Education Level']
  for col in missing_cols:
      # Calculate missing values within each group and then sum
      missing_counts = df.groupby('Department')[col].apply(lambda x:
x.isnull().sum())

      # Calculate total values per department
      total_counts = df.groupby('Department')[col].size()

      # Calculate percentage of missing values
      percentage_missing = (missing_counts / total_counts) * 100

      # Round percentage to two decimal places
      percentage_missing = percentage_missing.round(2)

      print(f"Missing values in {col} by department:\n{missing_counts}\n")
      print(f"Percentage of missing values in {col} by
department:\n{percentage_missing}\n")

except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

```
Missing values in Attendance (%) by department:
Department
Business       111
CS             194
Engineering    152
Mathematics     58
Name: Attendance (%), dtype: int64

Percentage of missing values in Attendance (%) by department:
Department
Business       10.69
CS              9.76
Engineering    10.15
Mathematics    12.16
Name: Attendance (%), dtype: float64

Missing values in Assignments Score (Averaged) by department:
Department
Business       123
CS             203
Engineering    125
Mathematics     52
Name: Assignments Score (Averaged), dtype: int64

Percentage of missing values in Assignments Score (Averaged) by
department:
Department
Business       11.85
```

```
CS             10.21
Engineering     8.35
Mathematics    10.90
Name: Assignments Score (Averaged), dtype: float64

Missing values in Parent Education Level by department:
Department
Business       382
CS             704
Engineering    533
Mathematics    181
Name: Parent Education Level, dtype: int64

Percentage of missing values in Parent Education Level by department:
Department
Business       36.80
CS             35.41
Engineering    35.60
Mathematics    37.95
Name: Parent Education Level, dtype: float64
```

In [11]:

```
# Step 9.1.1: Calculate the overall percentage of missing values in 'Parent
Education Level'

missing_parent_education = df['Parent Education Level'].isnull().sum()
total_parent_education = len(df['Parent Education Level'])
percentage_missing_parent_education = (missing_parent_education /
total_parent_education) * 100

print(f"Percentage of missing values in 'Parent Education Level':
{percentage_missing_parent_education:.2f}%")

# Group by 'Grade' and calculate missing values in 'Parent Education Level'
missing_by_grade = df.groupby('Grade')['Parent Education
Level'].apply(lambda x: x.isnull().sum())

# Calculate total students in each grade
total_by_grade = df.groupby('Grade')['Parent Education Level'].count()

# Calculate the percentage of missing values in 'Parent Education Level'
for each grade
percentage_missing_by_grade = (missing_by_grade / total_by_grade) * 100

print("\nPercentage of missing 'Parent Education Level' by Grade:")
percentage_missing_by_grade
```

```
Percentage of missing values in 'Parent Education Level': 36.00%

Percentage of missing 'Parent Education Level' by Grade:
```

Out[11]:

	Parent Education Level
Grade	
A	57.009346
B	55.694228
C	56.804734
D	52.816901
F	58.733205

dtype: float64

In [12]:

```
#Step 9.2: Apply imputation techniques
try:
  for col in ['Attendance (%)', 'Assignments Score (Averaged)','Parent
Education Level']:
    if df[col].dtype == 'object':
        mode_val = df[col].mode()[0]  # Calculate mode
        df[col].fillna(mode_val, inplace=True)  # Impute missing values
with mode
    else:
        mean_val = df[col].mean()  # Calculate mean
        df[col].fillna(mean_val, inplace=True)  # Impute missing values
with mean



  print("Columns with Data missing:")

  print(df.isnull().sum())

except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

```
Columns with Data missing:
Student ID                      0
First Name                      0
Last Name                       0
Email                           0
Gender                          0
Age                             0
Department                      0
Attendance (%)                  0
Midterm Score                   0
Final Score                     0
Assignments Score (Averaged)    0
Quizzes Score (Averaged)        0
Participation Score             0
Projects Score                  0
Total Score                     0
Grade                           0
Study Hours per Week            0
Extracurricular Activities      0
Internet Access at Home         0
Parent Education Level          0
Family Income Level             0
Stress Level (1-10)             0
Sleep Hours per Night           0
dtype: int64
```

```
<ipython-input-12-06102eaa7a15>:9: FutureWarning: A value is
trying to be set on a copy of a DataFrame or Series through
chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
```

```
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df[col].fillna(mean_val, inplace=True)  # Impute missing values
with mean
<ipython-input-12-06102eaa7a15>:6: FutureWarning: A value is
trying to be set on a copy of a DataFrame or Series through
chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will
never work because the intermediate object on which we are setting
values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on
the original object.


  df[col].fillna(mode_val, inplace=True)  # Impute missing values
with mode
```

In [13]:

```
#step 9.3: Drop the duplicate records

try:
  print("Number of duplicates:", df.duplicated().sum())
  df.drop_duplicates(inplace=True)
  print("Number of duplicates after dropping:", df.duplicated().sum())
  print("Duplicate records removed successfully!")
  print("Number of rows after dropping duplicates:", df.shape[0])
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

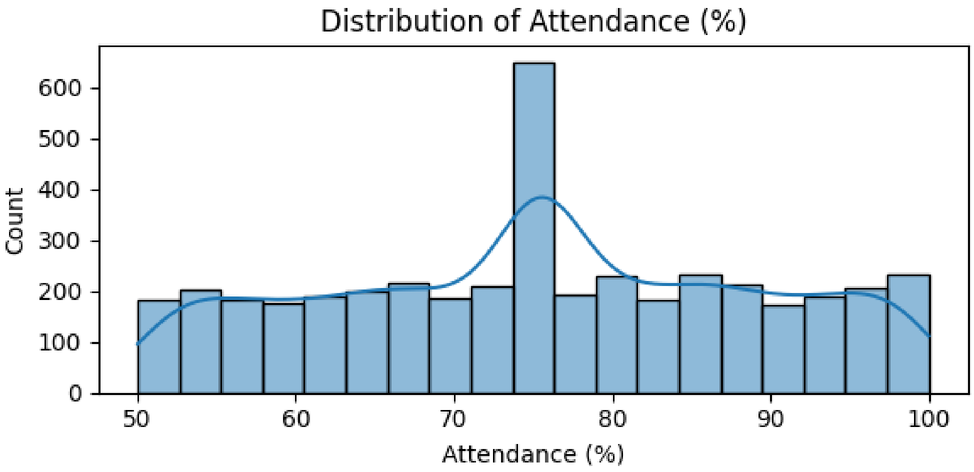
```
Number of duplicates: 757
Number of duplicates after dropping: 0
Duplicate records removed successfully!
Number of rows after dropping duplicates: 4243
```

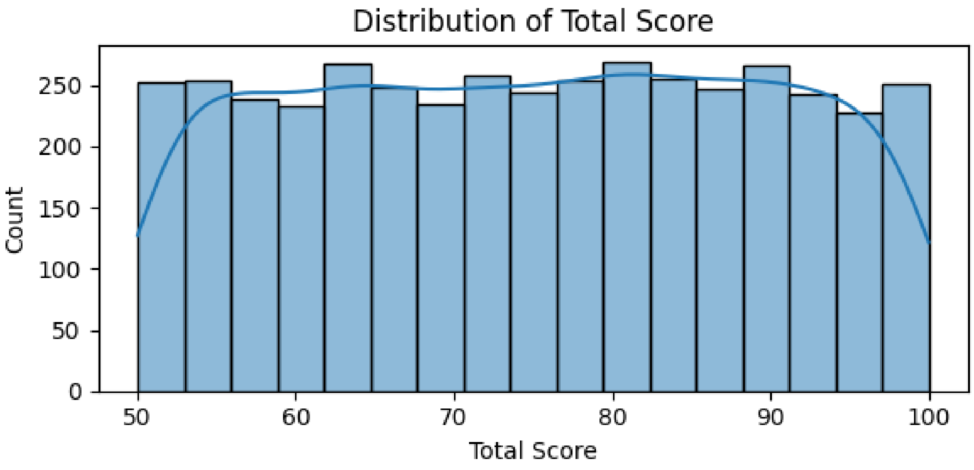
Visualisation and Analysis

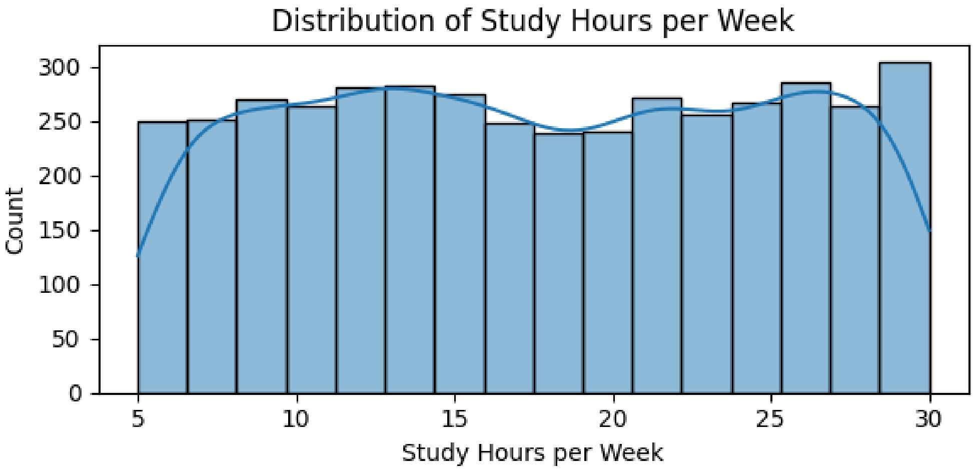
Q.1) What is the distribution of each numerical metric (e.g., attendance, scores, study hours)?

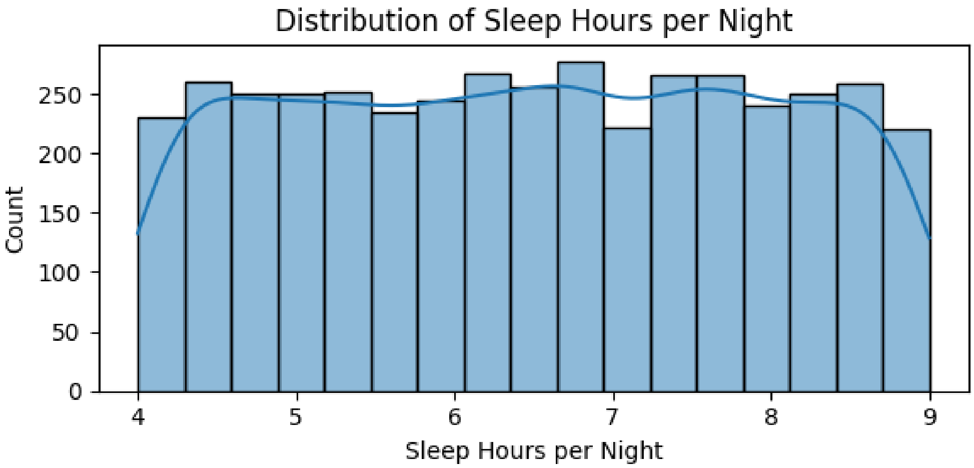
In [14]:

```
#What is the distribution of each numerical metric (e.g., attendance,
scores, study hours)?

# List of columns to visualize
numeric_cols = ['Attendance (%)',
                'Total Score','Study Hours per Week','Sleep Hours per
Night']

# Loop through each column
for col in numeric_cols:
    plt.figure(figsize=(6,3)) # Create a new figure for the plot
    sns.histplot(df[col], kde=True)  # Create a histogram with a density
curve
    plt.title(f'Distribution of {col}') # Add a title to the plot
    plt.tight_layout()  # Adjust the plot layout
    plt.show() # Show the plot
```





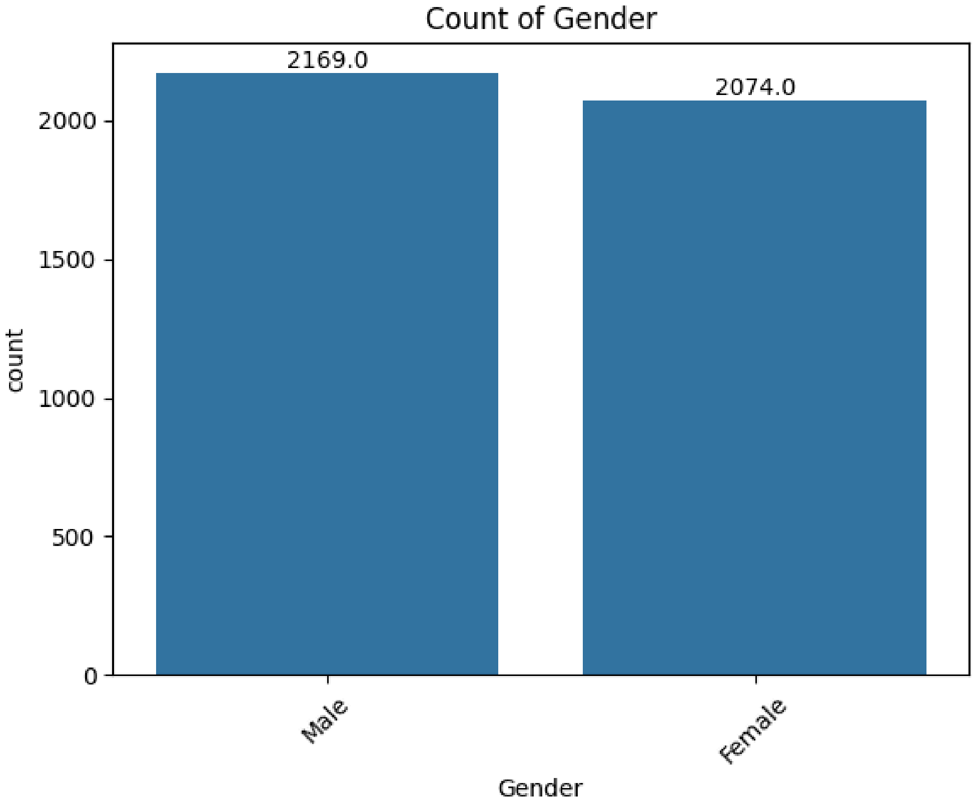


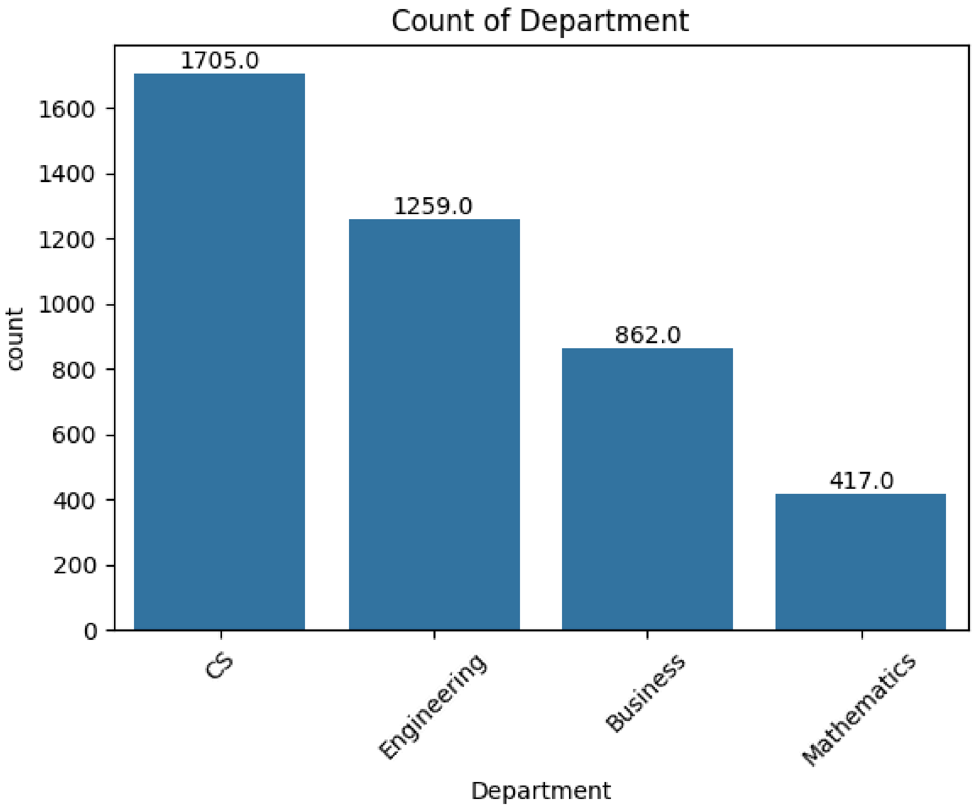


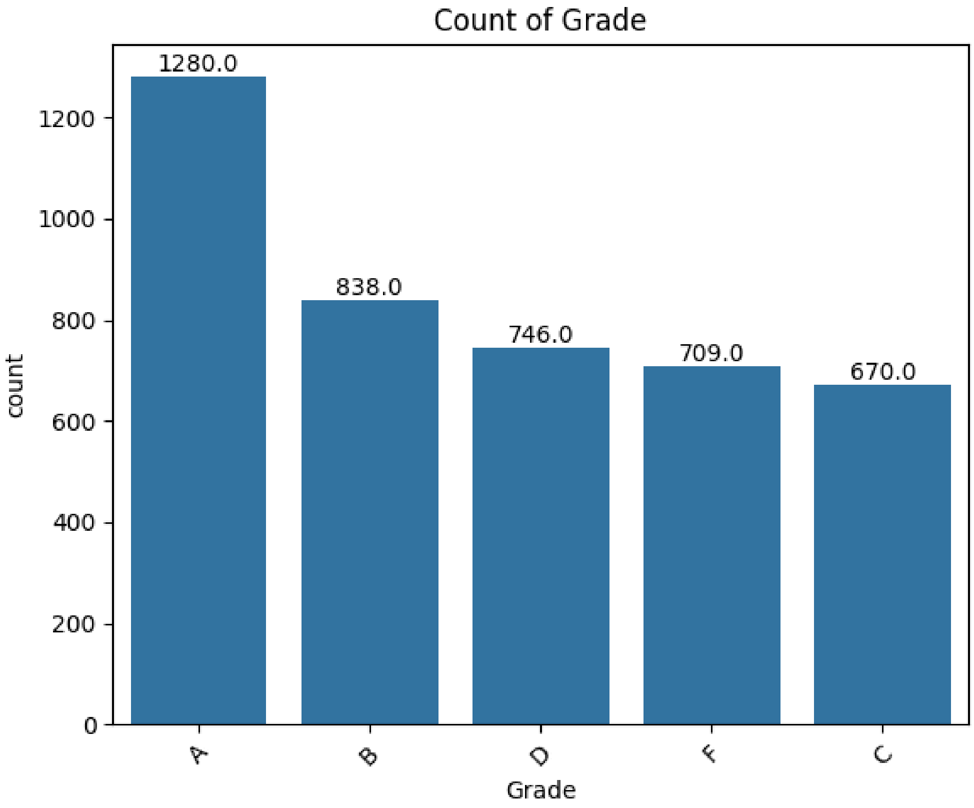
Q.2) How are students distributed across key categories (e.g., gender, department, grade)?

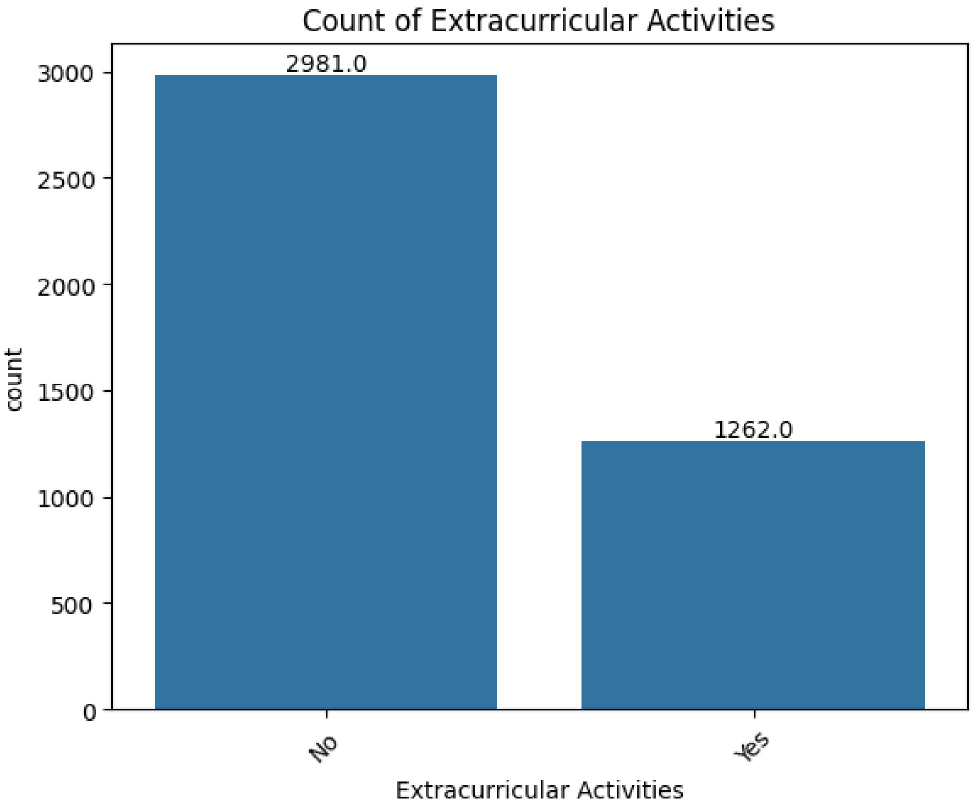
In [15]:

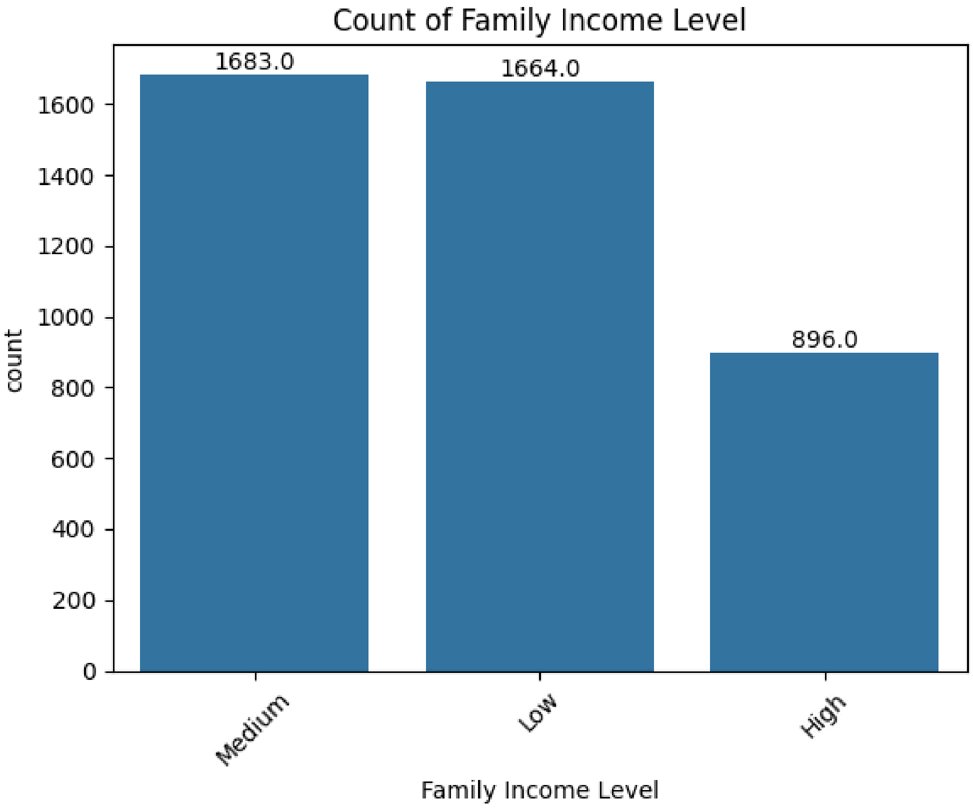
```
cat_cols = ['Gender','Department','Grade',
            'Extracurricular Activities','Family Income Level']

for col in cat_cols:
    plt.figure(figsize=(6,5))
    ax = sns.countplot(data=df, x=col,
                  order=df[col].value_counts().index)
    plt.title(f'Count of {col}')
    plt.xticks(rotation=45)
    plt.tight_layout()

    # Add value counts on top of bars
    for p in ax.patches:
        ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2.,
p.get_height()),
                    ha='center', va='center', xytext=(0, 5),
textcoords='offset points')

#https://dataplotplus.com/how-to-annotate-bars-with-values-on-pandas-bar-
plots/

    plt.show()
```




Count of Department
count
Department
1705.0
1259.0
862.0
417.0
0
200
400
600
800
1000
1200
1400
1600
CS
Engineering
Business
Mathematics


Count of Grade
count
Grade
1280.0
838.0
746.0
709.0
670.0
0
200
400
600
800
1000
1200
A
B
D
F
C

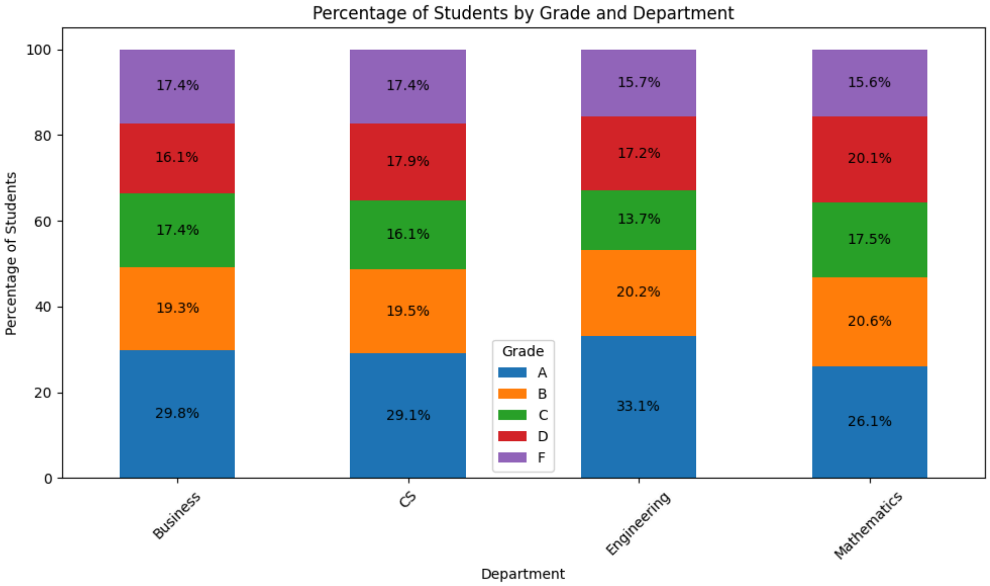




Q.3) Which department has the highest share of A grades, and which has the largest proportion of D grades?

In [16]:

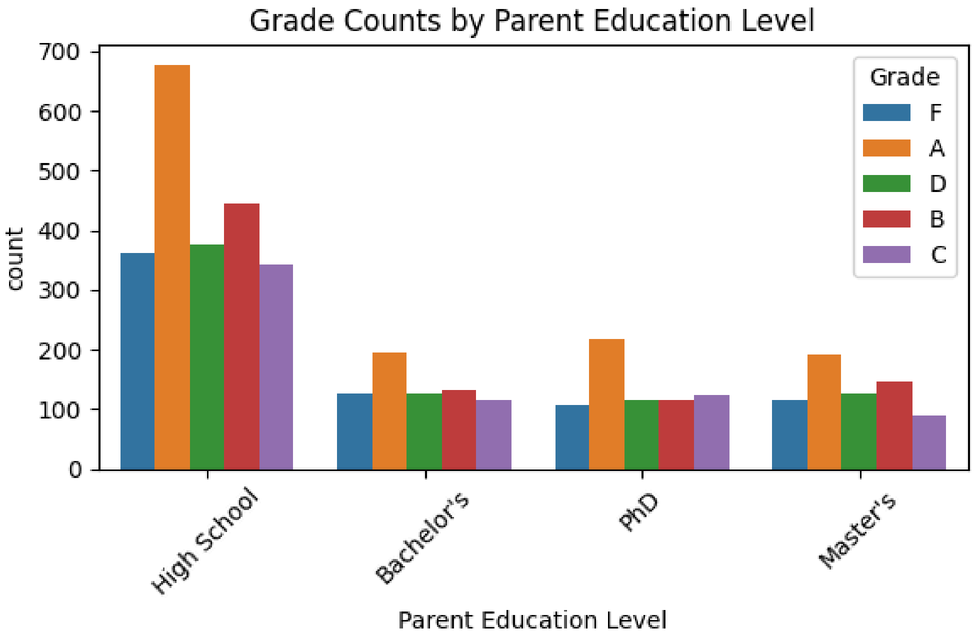
```
# Group data by 'Department' and 'Grade', then count 'Student ID' to get
student counts per group
grade_counts = df.groupby(['Department', 'Grade'])['Student
ID'].count().unstack(fill_value=0)
# unstack() pivots the 'Grade' level to columns, filling missing values
with 0

# Calculate the total number of students per department (summing across
grades)
department_totals = grade_counts.sum(axis=1) # axis=1 sums horizontally
(across columns)

# Calculate the percentage of students with each grade in each department
grade_percentages = grade_counts.div(department_totals, axis=0) * 100 #
Divides each grade count by department total, then multiplies by 100

# Create the stacked bar chart using the percentage data
ax = grade_percentages.plot(kind='bar', stacked=True, figsize=(10, 6)) #
kind='bar' specifies a bar chart, stacked=True makes it stacked
plt.title('Percentage of Students by Grade and Department')
plt.xlabel('Department')
plt.ylabel('Percentage of Students')
plt.xticks(rotation=45)
plt.legend(title='Grade') # Add a legend with the title 'Grade'

# Annotate the bars with percentages
for p in ax.patches: # Loop through each bar (patch) in the chart
    width, height = p.get_width(), p.get_height() # Get the width and
height of the bar
    x, y = p.get_xy() # Get the x and y coordinates of the bar's bottom
left corner
    if height > 0:  # Only annotate if the bar has a height greater than 0
        ax.text(x + width / 2, y + height / 2, f'{height:.1f}%',
ha='center', va='center') # Add the percentage text to the center of the
bar

#https://dataplotplus.com/how-to-annotate-bars-with-values-on-pandas-bar-
plots/

plt.tight_layout() # Adjust layout to prevent overlapping elements
plt.show() # Display the chart
```



Q.4) How does parental education level relate to student's final letter grades?

In [17]:

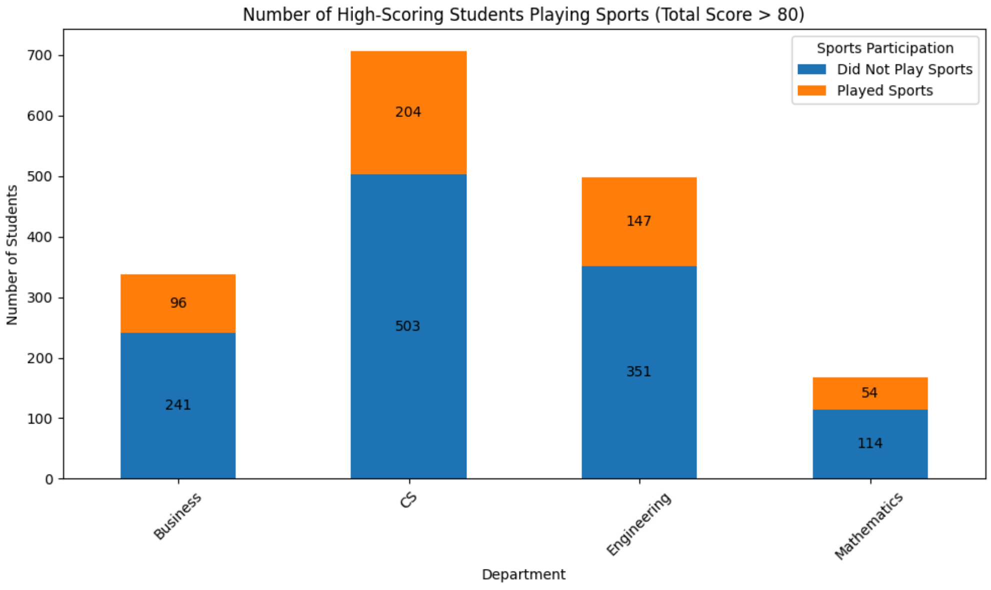
```
plt.figure(figsize=(6,4))
sns.countplot(
    data=df,
    x='Parent Education Level',
    hue='Grade',
    order=df['Parent Education Level'].value_counts().index # Order
categories by frequency
)
plt.title('Grade Counts by Parent Education Level')
plt.xticks(rotation=45)

#values have not been displayed because they are overlapping

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



Q.5) Which department has the highest number of high-scoring students who participate in sports?

In [18]:

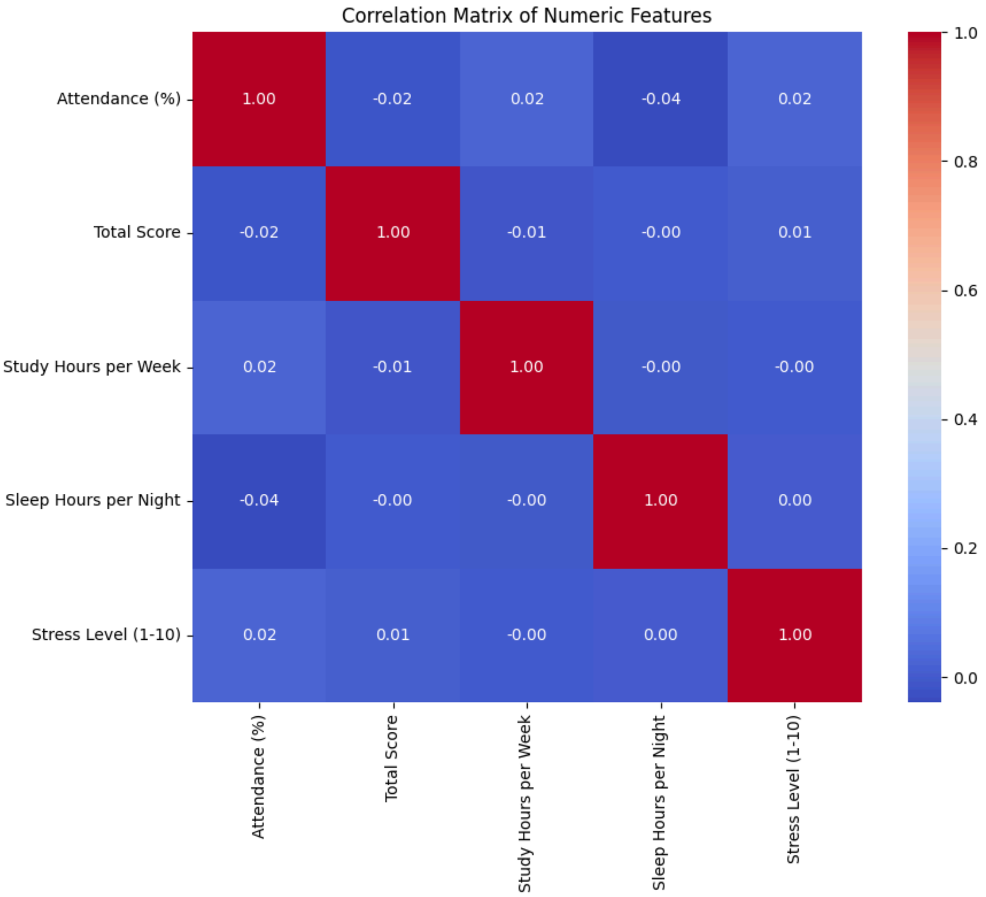
```
# Filter students with Total Score > 80
high_scorers = df[df['Total Score'] > 80]

# Group by department and Extracurricular Activities, then count
sports_summary = high_scorers.groupby(['Department', 'Extracurricular
Activities'])['Student ID'].count().unstack(fill_value=0)

# Rename columns for clarity
sports_summary = sports_summary.rename(columns={
    'No': 'Did Not Play Sports',
    'Yes': 'Played Sports'
})

# Create the stacked bar chart
ax = sports_summary.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Number of High-Scoring Students Playing Sports (Total Score >
80)')
plt.xlabel('Department')
plt.ylabel('Number of Students')
plt.xticks(rotation=45)
plt.legend(title='Sports Participation', loc='upper right')


# Display values on the chart
for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    x, y = p.get_xy()
    ax.text(x + width / 2, y + height / 2, int(height), ha='center',
va='center')

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



Q.5)What linear relationships exist among the numeric attributes?

In [19]:

```
plt.figure(figsize=(10,8))
corr = df[numeric_cols + ['Stress Level (1-10)']].corr()
sns.heatmap(
    corr, annot=True, fmt=".2f",
    cmap='coolwarm', square=True
)
plt.title('Correlation Matrix of Numeric Features')
plt.tight_layout()
plt.show()
```



In [26]:

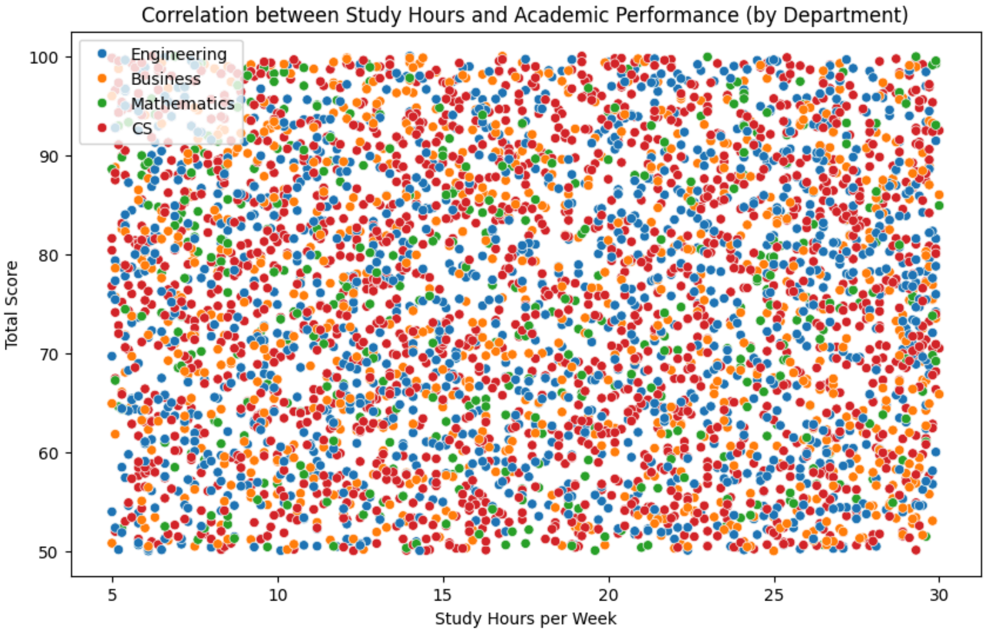
```
import statsmodels.api as sm

# Define independent and dependent variables
X = df[['Midterm Score', 'Final Score', 'Assignments Score (Averaged)',
        'Quizzes Score (Averaged)', 'Participation Score', 'Projects
Score',
        'Study Hours per Week', 'Stress Level (1-10)', 'Sleep Hours per
Night',
        'Attendance (%)']]  # Add all your desired numeric columns here

y = df['Total Score']

# Add a constant to the independent variables (intercept)
X = sm.add_constant(X)

# Fit the OLS model
model = sm.OLS(y, X).fit()

# Print the regression results
print(model.summary())

#https://www.geeksforgeeks.org/ordinary-least-squares-ols-using-
statsmodels/
```

```
                            OLS Regression Results
=====================================================================
=========
Dep. Variable:            Total Score   R-squared:
0.002
Model:                            OLS   Adj. R-squared:
-0.000
Method:                 Least Squares   F-statistic:
0.9321
Date:                Mon, 19 May 2025   Prob (F-statistic):
0.502
Time:                        00:52:50   Log-Likelihood:
-17325.
No. Observations:                4243   AIC:
3.467e+04
Df Residuals:                    4232   BIC:
3.474e+04
Df Model:                          10
Covariance Type:            nonrobust
=====================================================================
===========================
                                   coef    std err          t
P>|t|      [0.025      0.975]
---------------------------------------------------------------------
---------------------------
const                           76.8013      2.999     25.613
0.000      70.923      82.680
Midterm Score                   -0.0018      0.013     -0.137
0.891      -0.027       0.024
```

```
Final Score                      0.0032      0.013      0.246
0.806      -0.022       0.028
Assignments Score (Averaged)     0.0048      0.016      0.299
0.765      -0.027       0.037
Quizzes Score (Averaged)         0.0241      0.015      1.585
0.113      -0.006       0.054
Participation Score             -0.1257      0.076     -1.643
0.100      -0.276       0.024
Projects Score                  -0.0242      0.015     -1.581
0.114      -0.054       0.006
Study Hours per Week            -0.0203      0.030     -0.670
0.503      -0.080       0.039
Stress Level (1-10)              0.0265      0.077      0.343
0.731      -0.125       0.178
Sleep Hours per Night           -0.0269      0.152     -0.177
0.860      -0.325       0.271
Attendance (%)                  -0.0165      0.016     -1.016
0.309      -0.048       0.015
=====================================================================
=========
Omnibus:                     3143.530   Durbin-Watson:
1.995
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
248.232
Skew:                          -0.013   Prob(JB):
1.25e-54
Kurtosis:                       1.815   Cond. No.
2.47e+03
=====================================================================
=========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors
is correctly specified.
[2] The condition number is large, 2.47e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

In [21]:

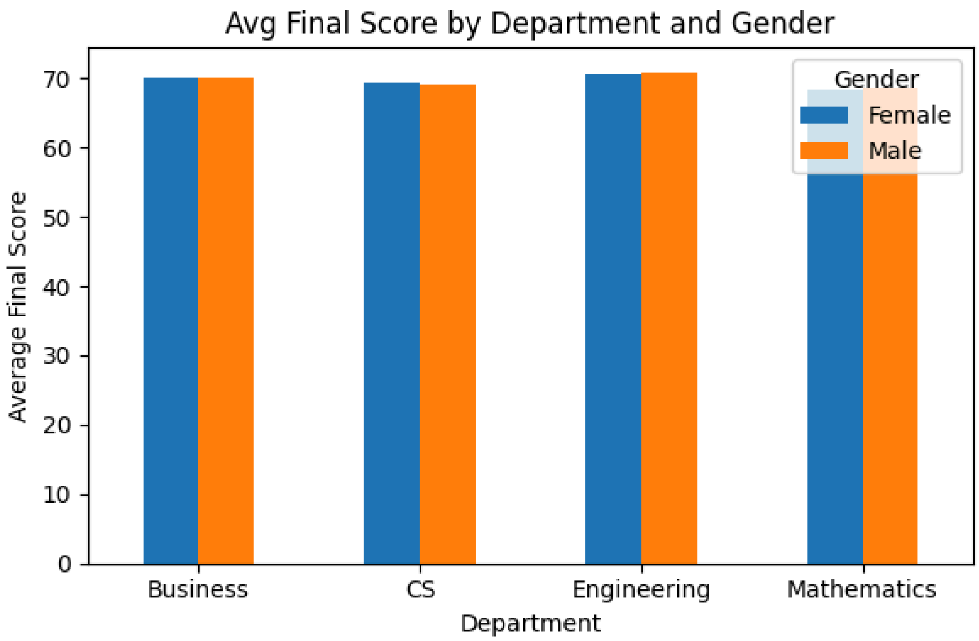
```
department_avg_scores = df.groupby('Department')['Total Score'].mean()

# Create the scatter plot with department as hue and averaged total score
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Study Hours per Week', y='Total Score',
hue='Department')
plt.title('Correlation between Study Hours and Academic Performance (by
Department)')
plt.xlabel('Study Hours per Week')
plt.ylabel('Total Score')

plt.legend()
plt.show()
```



Q.6) Do average final exam scores differ across departments and between genders?

In [22]:

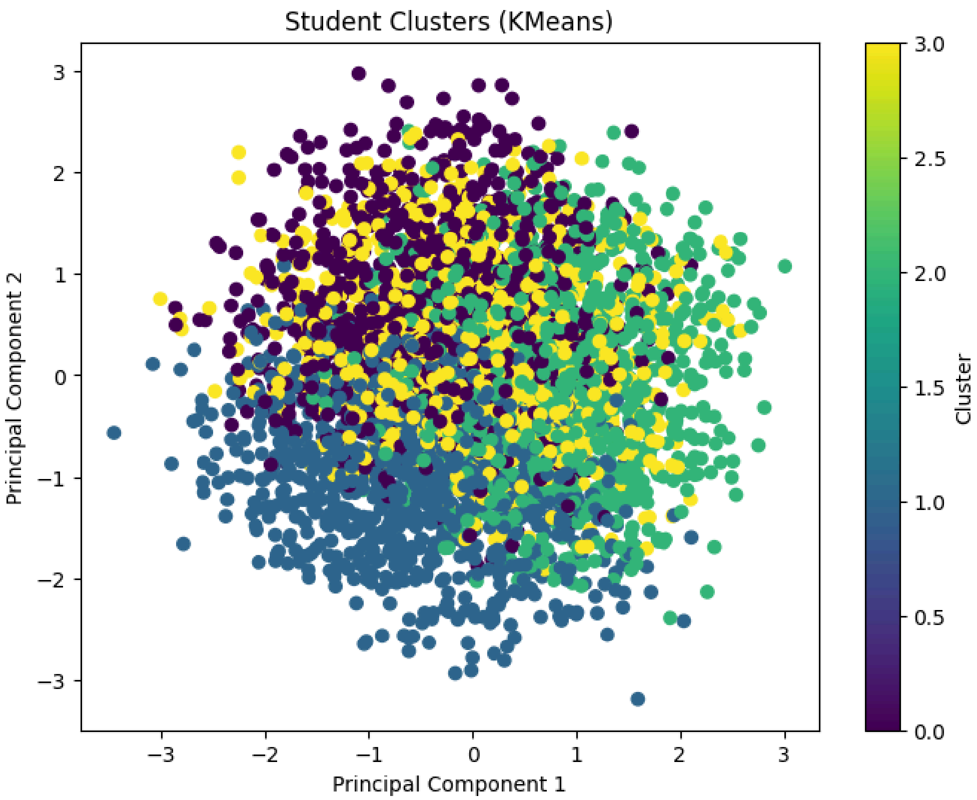
```
grouped = df.groupby(['Department','Gender'])['Final
Score'].mean().unstack()
grouped.plot(kind='bar', figsize=(6,4))
plt.title('Avg Final Score by Department and Gender')
plt.ylabel('Average Final Score')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



Q.7) What distinct study and performance patterns can we identify among students based on their academic scores, study hours, and stress levels?

In [23]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans

# Select relevant features for clustering
features_for_clustering = ['Midterm Score', 'Final Score', 'Assignments Score (Averaged)', 'Quizzes Score (Averaged)', 'Study Hours per Week', 'Stress Level (1-10)']
X_cluster = df[features_for_clustering]

# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_cluster)

# Apply PCA to reduce dimensionality (optional, but can improve clustering)
pca = PCA(n_components=2)  # Reduce to 2 principal components for visualization
X_pca = pca.fit_transform(X_scaled)

# Perform KMeans clustering
kmeans = KMeans(n_clusters=4, random_state=42) # Experiment with different numbers of clusters
df['Cluster'] = kmeans.fit_predict(X_scaled) # Use scaled data for clustering

# Visualize clusters (using PCA-reduced data for 2D visualization)
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=df['Cluster'], cmap='viridis')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Student Clusters (KMeans)')
plt.colorbar(label='Cluster')
plt.show()

# Analyze cluster characteristics
for cluster in range(4):  # Assuming 4 clusters
  print(f"Cluster {cluster}:")
  print(df[df['Cluster'] == cluster][features_for_clustering].describe())

#https://365datascience.com/tutorials/python-tutorials/pca-k-means/
#https://pulsedatahub.com/blog/k-means-clustering
```



```
Cluster 0:
       Midterm Score  Final Score  Assignments Score (Averaged)  \
count    1059.000000  1059.000000                   1059.000000   
mean       74.554721    82.324797                     75.865203   
std        14.824129    10.884766                     13.563318   
min        40.010000    49.790000                     50.080000   
25%        63.765000    74.645000                     65.545000   
50%        76.330000    83.210000                     74.828117   
75%        86.155000    91.070000                     86.390000   
max        99.910000    99.980000                     99.960000   

       Quizzes Score (Averaged)  Study Hours per Week  Stress Level 
(1-10)  
count               1059.000000           1059.000000          
1059.000000  
mean                  71.218593             10.763362          
5.556185  
std                   13.908271              3.629310          
2.824984  
min                   50.030000              5.000000          
1.000000  
25%                   59.390000              7.600000          
3.000000  
50%                   70.100000             10.400000          
6.000000  
75%                   81.790000             13.400000          
8.000000  
max                   99.900000             21.100000         
10.000000  
Cluster 1:
       Midterm Score  Final Score  Assignments Score (Averaged)  \
count    1058.000000  1058.000000                   1058.000000
mean       52.817807    57.252836                     74.227787
std         8.512407    10.724053                     13.751985
min        40.000000    40.090000                     50.130000
25%        45.380000    48.285000                     62.817500
50%        52.075000    56.675000                     74.828117
75%        59.117500    64.472500                     85.102500
max        76.980000    86.790000                     99.980000

       Quizzes Score (Averaged)  Study Hours per Week  Stress Level
(1-10)
count               1058.000000           1058.000000
1058.000000
mean                  75.112543             16.443195
4.987713
std                   14.496912              6.662057
2.831910
min                   50.030000              5.000000
1.000000
25%                   62.595000             11.200000
2.000000
50%                   74.940000             15.800000
5.000000
75%                   87.882500             21.600000
7.000000
max                   99.960000             30.000000
10.000000
Cluster 2:
       Midterm Score  Final Score  Assignments Score (Averaged)  \
count    1064.000000  1064.000000                   1064.000000
mean       69.218882    84.760254                     74.614870
std        15.918761     9.577428                     13.490179
min        40.020000    56.820000                     50.000000
25%        56.607500    78.210000                     64.002500
50%        67.980000    85.755000                     74.828117
75%        82.115000    92.390000                     84.745000
max        99.880000    99.980000                     99.780000

       Quizzes Score (Averaged)  Study Hours per Week  Stress Level
(1-10)
count               1064.000000           1064.000000
1064.000000
mean                  77.387284             24.202256
5.535714
std                   14.446141              3.852310
2.857981
min                   50.190000             13.400000
1.000000
25%                   65.560000             21.300000
3.000000
50%                   78.415000             24.700000
6.000000
```

```
75%                   90.130000             27.400000
8.000000
max                   99.940000             30.000000
10.000000
Cluster 3:
       Midterm Score  Final Score  Assignments Score (Averaged)  \
count    1062.000000  1062.000000                   1062.000000
mean       84.840377    54.583004                     74.637426
std         9.587196     9.023724                     13.728215
min        61.530000    40.000000                     50.010000
25%        77.452500    47.107500                     62.910000
50%        85.580000    53.810000                     74.828117
75%        93.227500    60.782500                     85.842500
max        99.970000    81.440000                     99.920000

       Quizzes Score (Averaged)  Study Hours per Week  Stress Level
(1-10)
count               1062.000000           1062.000000
1062.000000
mean                  75.923399             19.177589
5.794727
std                   14.476284              6.773679
2.865711
min                   50.160000              5.000000
1.000000
25%                   63.620000             13.900000
3.000000
50%                   76.630000             19.500000
6.000000
75%                   88.385000             25.000000
8.000000
max                   99.960000             29.900000
10.000000
```