In [1]:

```python
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import numpy as np
```

In [2]:

```python
df_mat = pd.read_csv('student-mat.csv', delimiter=';')
df_mat
```

Out[2]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
390	MS	M	20	U	LE3	A	2	2	services	services	...	5	5	4
391	MS	M	17	U	LE3	T	3	1	services	services	...	2	4	5
392	MS	M	21	R	GT3	T	1	1	other	other	...	5	5	3
393	MS	M	18	R	LE3	T	3	2	services	other	...	4	4	1
394	MS	M	19	U	LE3	T	1	1	other	at_home	...	3	2	3

395 rows × 33 columns

In [3]:

```python
df_mat.columns.tolist()
```

Out[3]:

```
['school',
 'sex',
 'age',
 'address',
 'famsize',
 'Pstatus',
 'Medu',
 'Fedu',
 'Mjob',
 'Fjob',
 'reason',
 'guardian',
 'traveltime',
 'studytime',
 'failures',
 'schoolsup',
 'famsup',
 'paid',
 'activities',
 'nursery',
 'higher',
 'internet',
 'romantic',
 'famrel',
 'freetime',
 'goout',
 'Dalc',
 'Walc',
 'health',
 'absences',
 'G1',
 'G2',
 'G3']
```

In [4]:

```
df_por = pd.read_csv('student-por.csv', delimiter=';')
df_por
```

Out[4]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
644	MS	F	19	R	GT3	T	2	3	services	other	...	5	4	2
645	MS	F	18	U	LE3	T	3	1	teacher	services	...	4	3	4
646	MS	F	18	U	GT3	T	1	1	other	other	...	1	1	1
647	MS	M	17	U	LE3	T	3	1	services	services	...	2	4	5
648	MS	M	18	R	LE3	T	3	2	services	other	...	4	4	1

649 rows × 33 columns

In [5]:

```
df_por.columns.tolist()
```

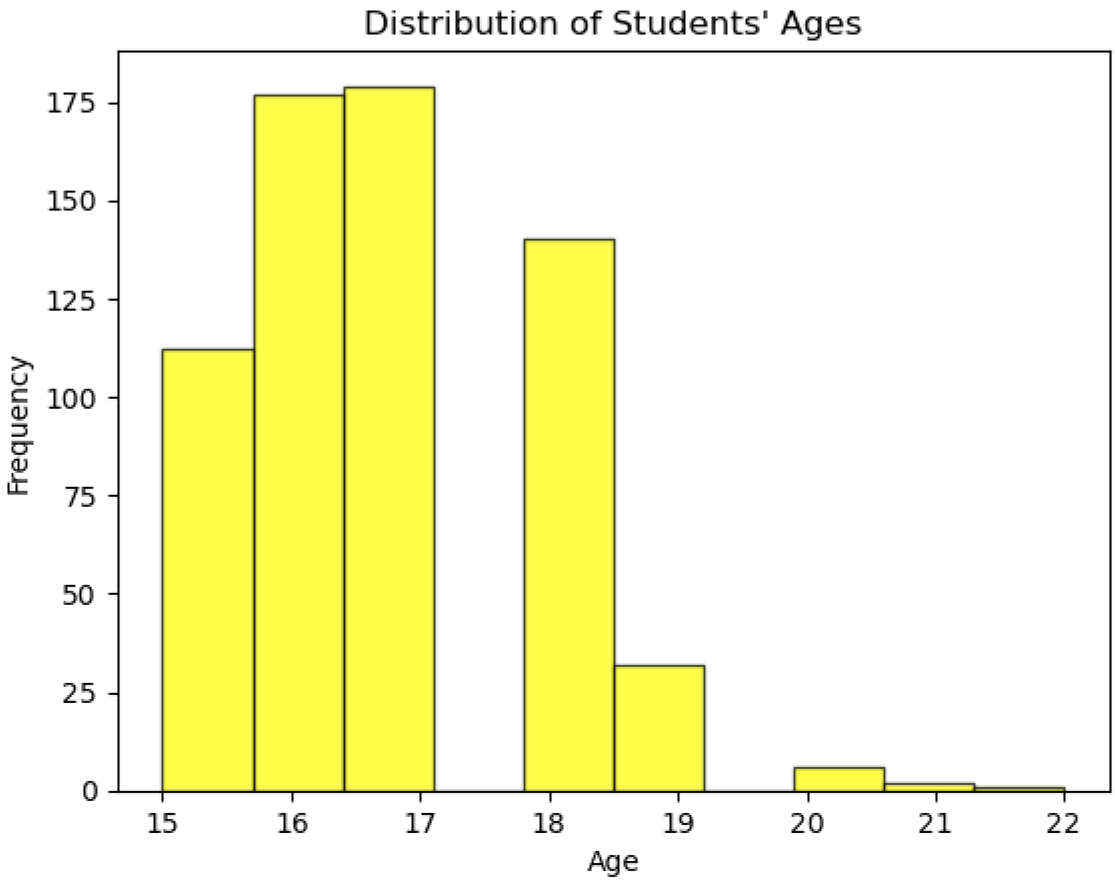
Out[5]:

```
['school',
 'sex',
 'age',
 'address',
 'famsize',
 'Pstatus',
 'Medu',
 'Fedu',
 'Mjob',
 'Fjob',
 'reason',
 'guardian',
 'traveltime',
 'studytime',
 'failures',
 'schoolsup',
 'famsup',
 'paid',
 'activities',
 'nursery',
 'higher',
 'internet',
 'romantic',
 'famrel',
 'freetime',
 'goout',
 'Dalc',
 'Walc',
 'health',
 'absences',
 'G1',
 'G2',
 'G3']
```

Task 1: What is the distribution of students' ages in the dataset?

In [6]:

```
plt.hist(df_por['age'], bins=10, alpha=0.7, color='yellow', edgecolor='black')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Distribution of Students\' Ages')
plt.show()
```



Task 2: How many students belong to each school (GP or MS)?

In [7]:

```
df2 = df_por['school'].value_counts()
df2
```

Out[7]:

```
GP    423
MS    226
Name: school, dtype: int64
```

Task 3: What is the gender distribution of students?

In [8]:

```
gen_dist = df_por['sex'].value_counts()
gen_dist
```

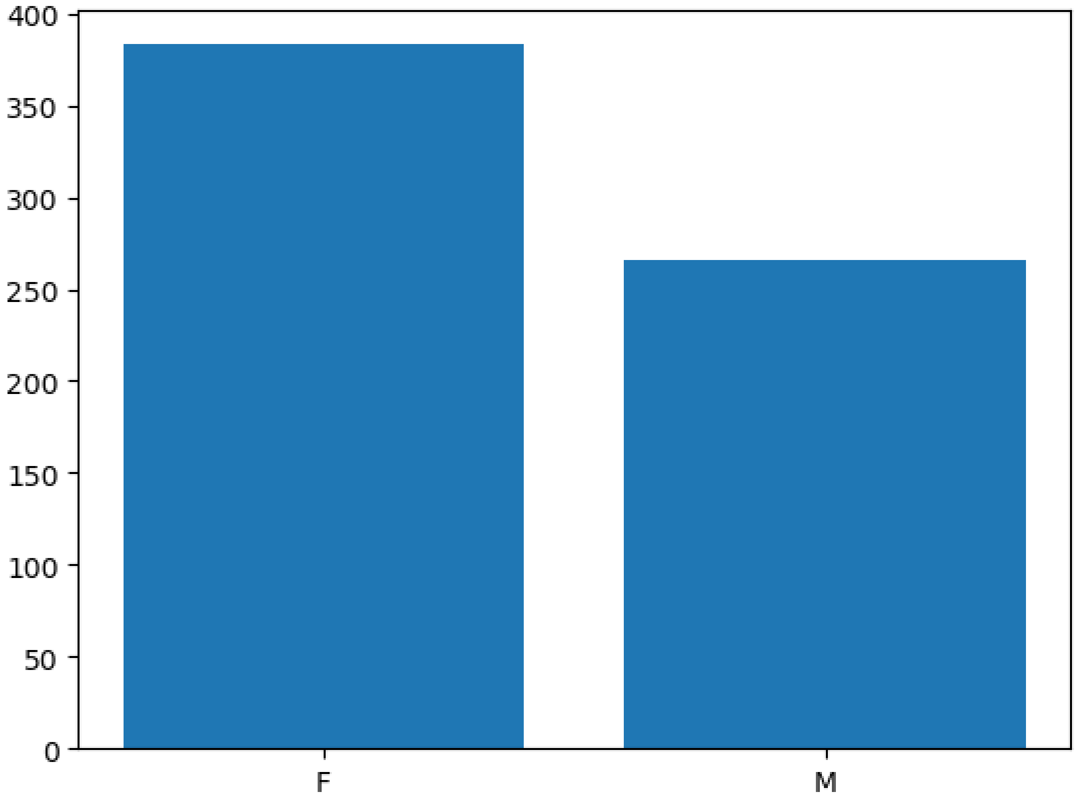
Out[8]:

```
F    383
M    266
```

```
Name: sex, dtype: int64
```

In [9]:

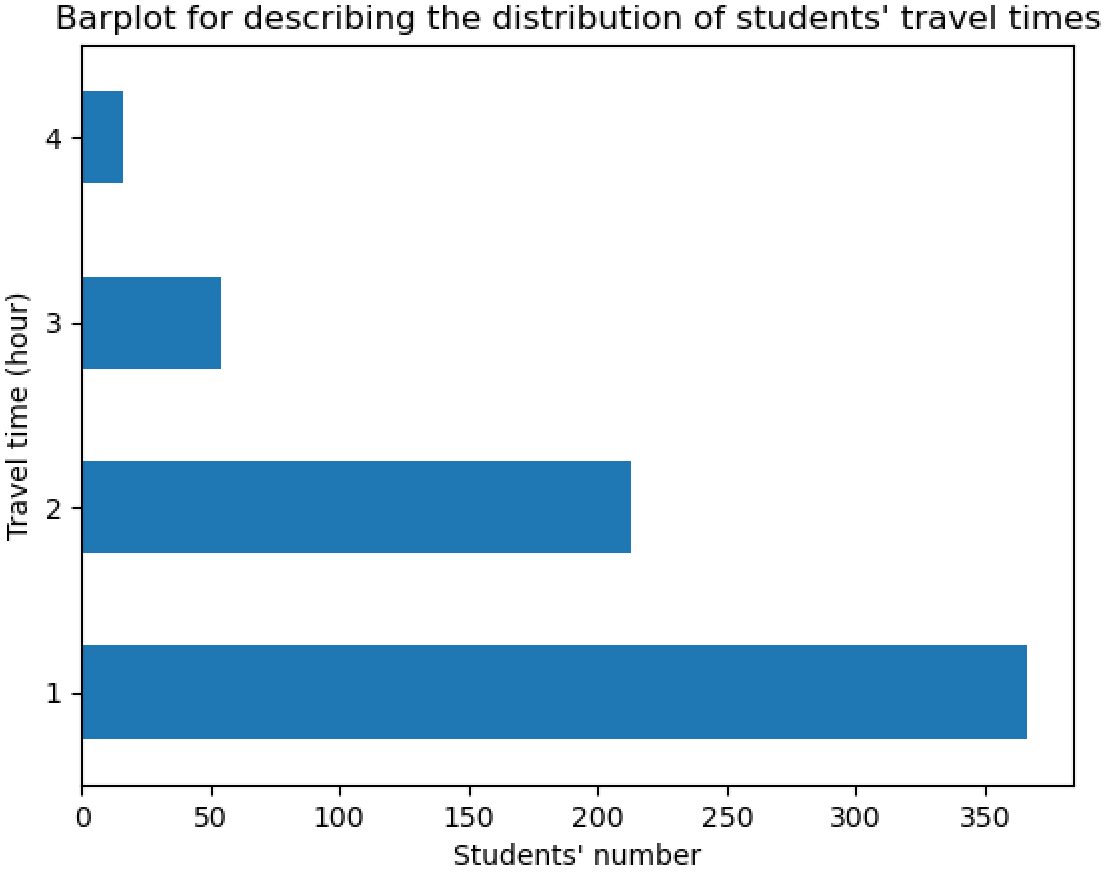
```
df4 = df_por['sex'].value_counts()
plt.bar(df4.index, df4)
plt.show()
```



Task 4: What is the distribution of students' travel times to school?

In [10]:

```
df_por['traveltime'].value_counts().plot.barh()
plt.xlabel('Students\' number')
plt.ylabel('Travel time (hour)')
plt.title('Barplot for describing the distribution of students\' travel times')
plt.show()
```



Task 5: How do the first period grades (G1) vary with study time (studytime)?

In [11]:

```
vary = pd.crosstab(index = df_por['G1'], columns = df_por['studytime'], margins = True)
vary
```

Out[11]:

studytime	**1**	**2**	**3**	**4**	**All**
G1					
0	0	1	0	0	1
4	1	1	0	0	2
5	2	2	0	1	5
6	6	3	0	0	9
7	16	14	3	0	33
8	19	21	2	0	42
9	35	25	4	1	65
10	34	40	14	7	95
11	29	42	16	4	91
12	24	46	9	3	82
13	17	36	13	6	72
14	19	32	15	5	71
15	4	16	14	1	35
16	0	17	3	2	22

17	4	8	2	2	16
18	2	1	1	3	7
19	0	0	1	0	1
All	212	305	97	35	649

Task 6: Is there a correlation between students' weekly study time (studytime) and their final grades (G3)?

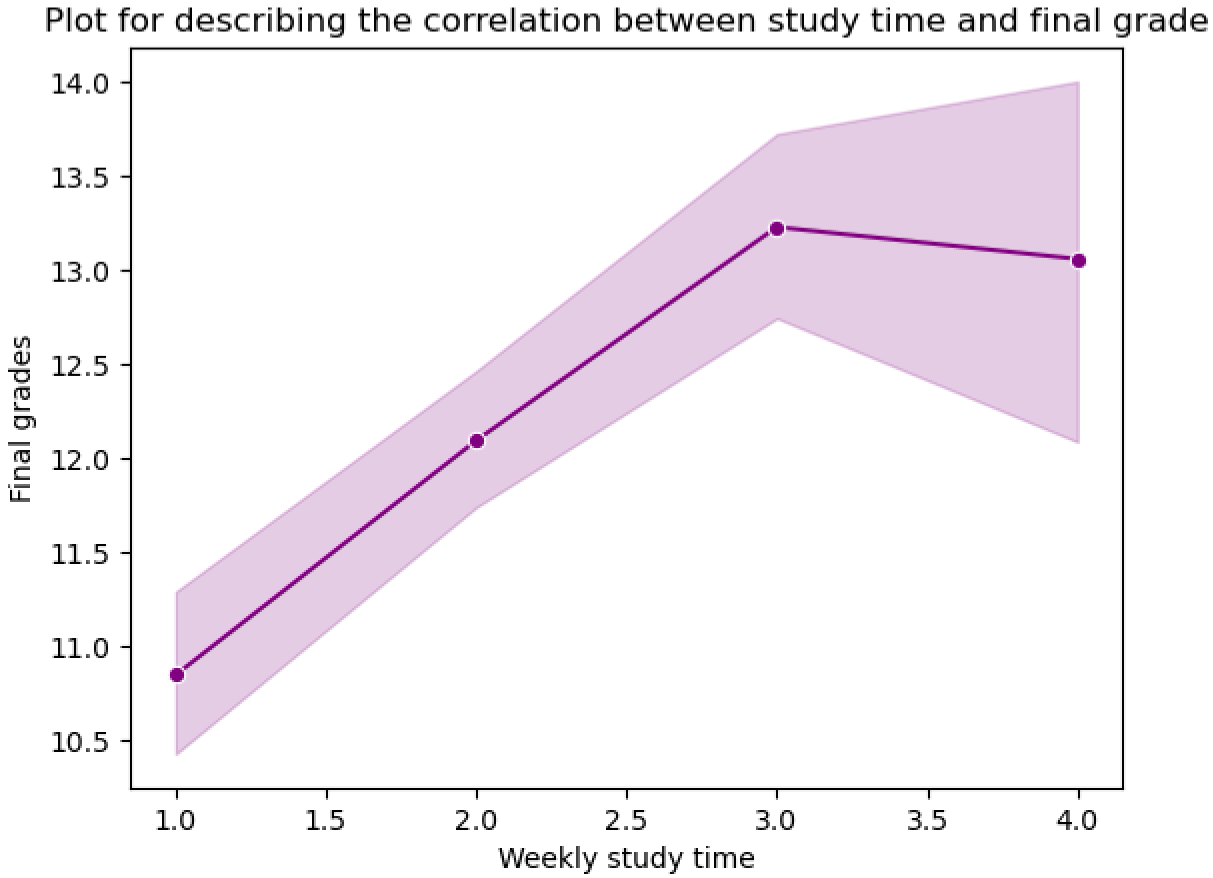
In [12]:

```
correlation_absences_G3 = df_por['studytime'].corr(df_por['G3'])
print(f"Correlation between study time and final grades (G3): {correlation_absences_G3:.
```

```
Correlation between study time and final grades (G3): 0.25
```

In [13]:

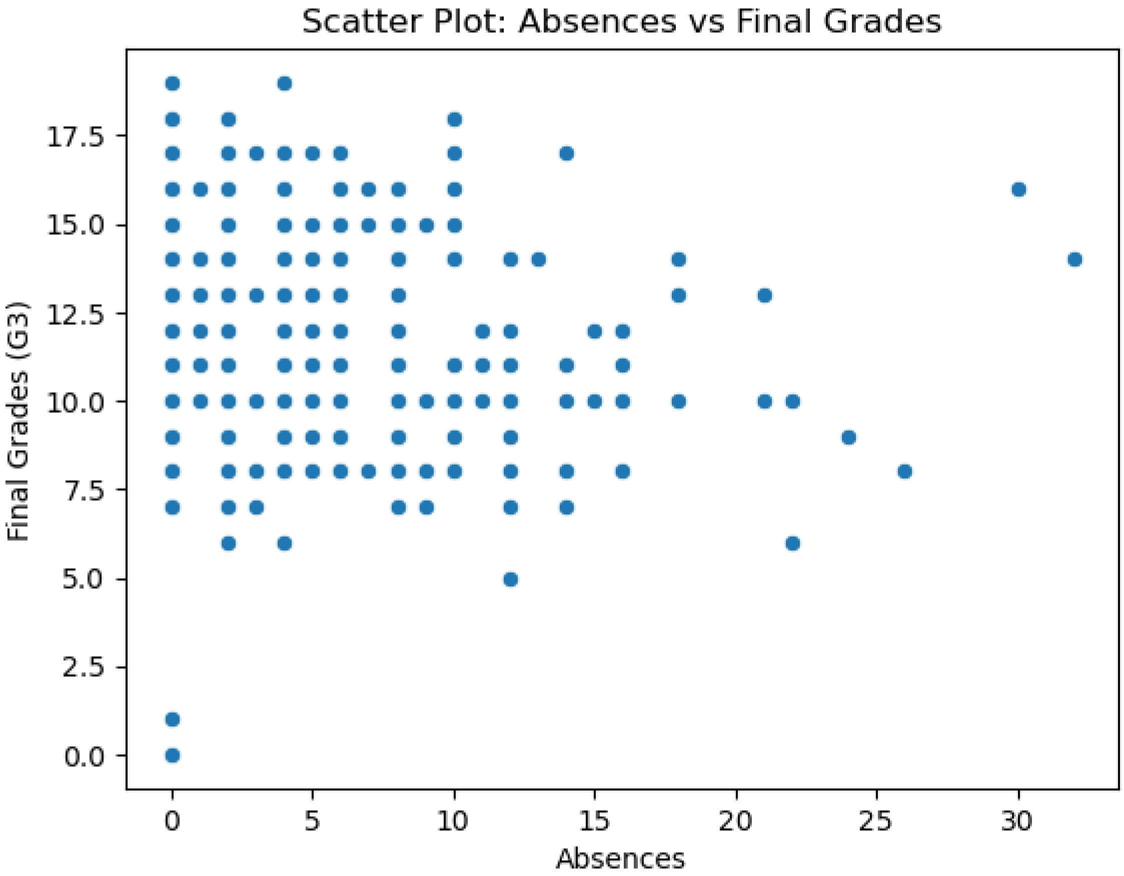
```
sns.lineplot(data=df_por, x='studytime', y='G3', marker='o', color = 'purple')
plt.title('Plot for describing the correlation between study time and final grade')
plt.xlabel('Weekly study time')
plt.ylabel('Final grades')
plt.show()
```



Task 7: How do students' absences (absences) relate to their final grades (G3)?

In [14]:

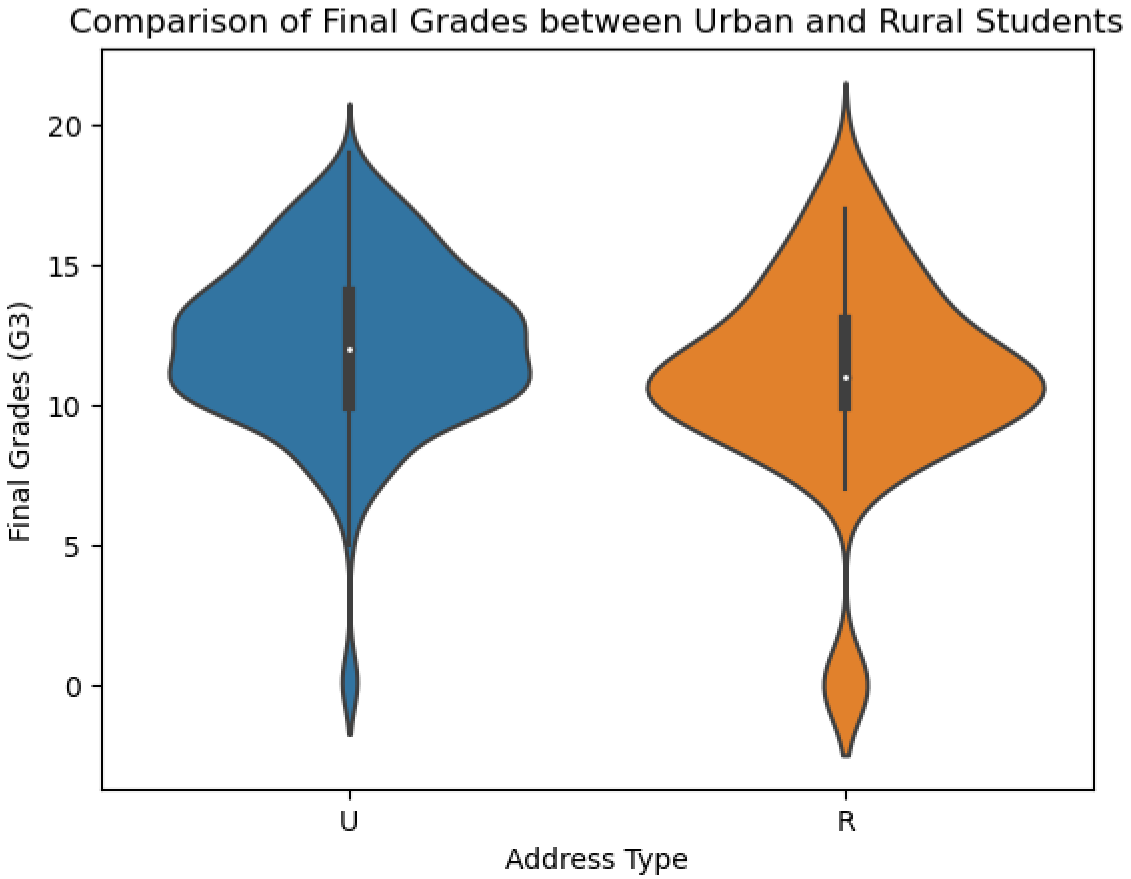
```
sns.scatterplot(data=df_por, x='absences', y='G3')
plt.title('Scatter Plot: Absences vs Final Grades')
plt.xlabel('Absences')
plt.ylabel('Final Grades (G3)')
plt.show()
```



Task 8: Are there differences in final grades (G3) between students living in urban (U) and rural (R) areas?

In [15]:

```
sns.violinplot(data=df_por, x='address', y='G3')
plt.title('Comparison of Final Grades between Urban and Rural Students')
plt.xlabel('Address Type')
plt.ylabel('Final Grades (G3)')
plt.show()
```



Task 9: What is the relationship between family size (famsize) and the quality of family relationships (famrel)?

In [16]:

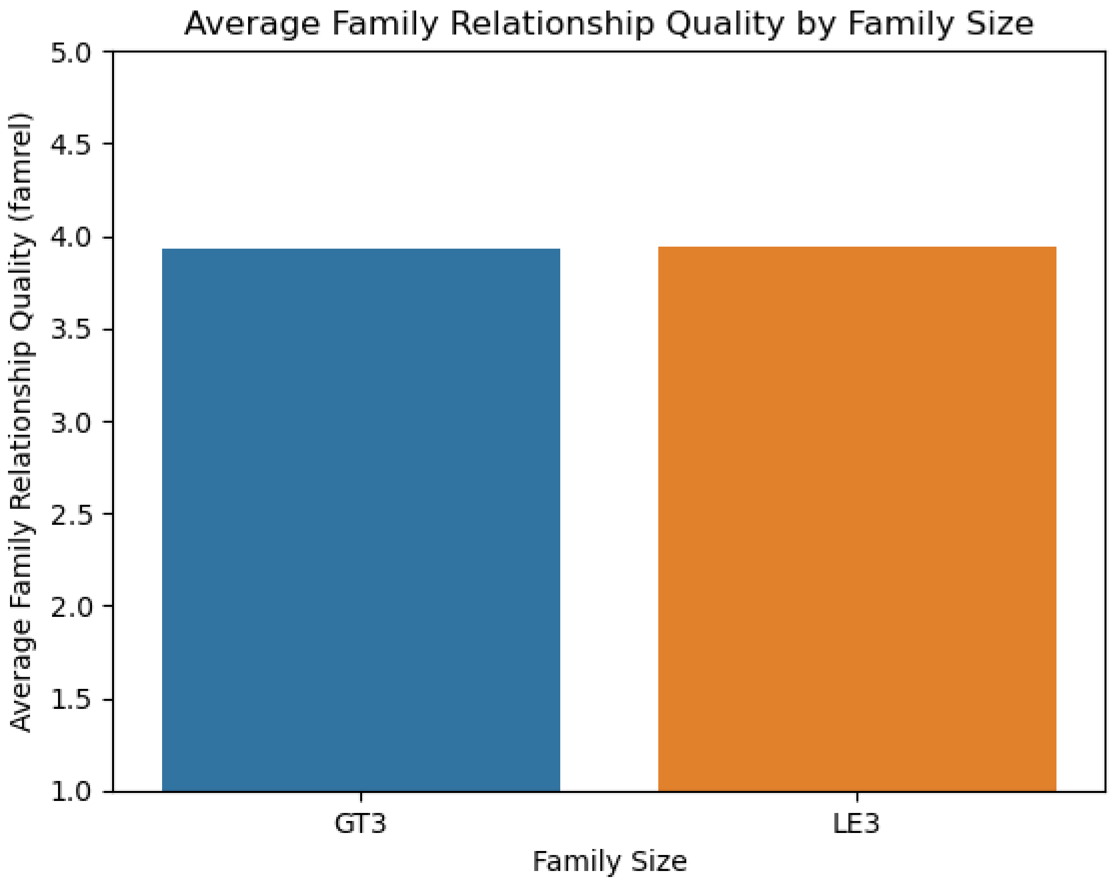
```
ct3 = pd.crosstab(df_por['famsize'], df_por['famrel'], margins = True)
ct3
```

Out[16]:

famrel famsize	1	2	3	4	5	All
GT3	15	18	77	222	125	457
LE3	7	11	24	95	55	192
All	22	29	101	317	180	649

In [17]:

```
avg_famrel = df_por.groupby('famsize')['famrel'].mean().reset_index()

sns.barplot(data=avg_famrel, x='famsize', y='famrel')
plt.title('Average Family Relationship Quality by Family Size')
plt.xlabel('Family Size')
plt.ylabel('Average Family Relationship Quality (famrel)')
plt.ylim(1, 5)
plt.show()
```



Task 10: Does the presence of romantic relationships (romantic) affect students' alcohol consumption (Dalc and Walc)?

In [18]:

```
df_por['romantic'].isnull().sum()
```

Out[18]:

```
0
```

In [19]:

```
df_por['romantic']
```

Out[19]:

```
0       no
1       no
2       no
3      yes
4       no
      ... 
644     no
645     no
646     no
647     no
648     no
Name: romantic, Length: 649, dtype: object
```

In [20]:

```
df_por['romantic'] = df_por['romantic'].map({"yes":1,"no":0})
df_por['romantic']
```

Out[20]:

```
0      0
1      0
2      0
3      1
4      0
      ..
644    0
```

```
645    0
646    0
647    0
648    0
Name: romantic, Length: 649, dtype: int64
```

In [21]:

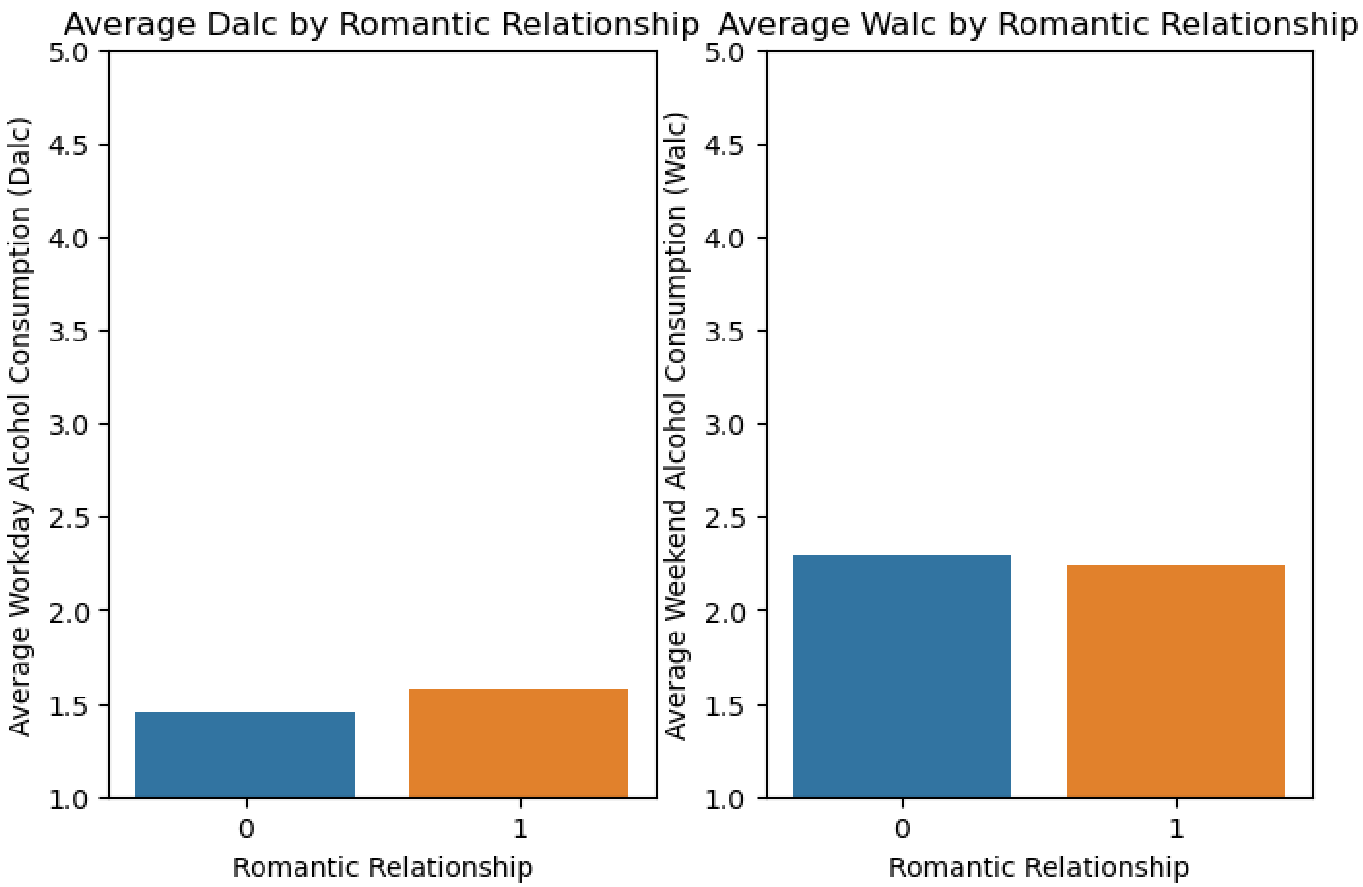
```
vary1 = pd.crosstab(index=[df_por['Dalc'], df_por['Walc']], columns=df_por['romantic'],
vary1
```

Out[21]:

	romantic	**0**	**1**	**All**
Dalc	**Walc**			
1	**1**	152	89	241
	2	76	37	113
	3	39	25	64
	4	18	10	28
	5	4	1	5
2	**1**	1	2	3
	2	18	16	34
	3	27	16	43
	4	23	11	34
	5	6	1	7
3	**1**	0	1	1
	2	1	0	1
	3	7	2	9
	4	16	4	20
	5	8	4	12
4	**1**	0	1	1
	2	0	1	1
	3	2	2	4
	4	3	2	5
	5	2	4	6
5	**1**	0	1	1
	2	0	1	1
	5	7	8	15
All		410	239	649

In [22]:

```
avg_alcohol = df_por.groupby('romantic')[['Dalc', 'Walc']].mean().reset_index()

plt.figure(figsize=(8, 5))

plt.subplot(1, 2, 1)
sns.barplot(data=avg_alcohol, x='romantic', y='Dalc')
plt.title('Average Dalc by Romantic Relationship')
plt.xlabel('Romantic Relationship')
plt.ylabel('Average Workday Alcohol Consumption (Dalc)')
plt.ylim(1, 5)

plt.subplot(1, 2, 2)

sns.barplot(data=avg_alcohol, x='romantic', y='Walc')
plt.title('Average Walc by Romantic Relationship')
plt.xlabel('Romantic Relationship')
plt.ylabel('Average Weekend Alcohol Consumption (Walc)')

plt.ylim(1, 5)
plt.show()
```



Task11: How does the mother's education level (Medu) correlate with the father's education level (Fedu)?

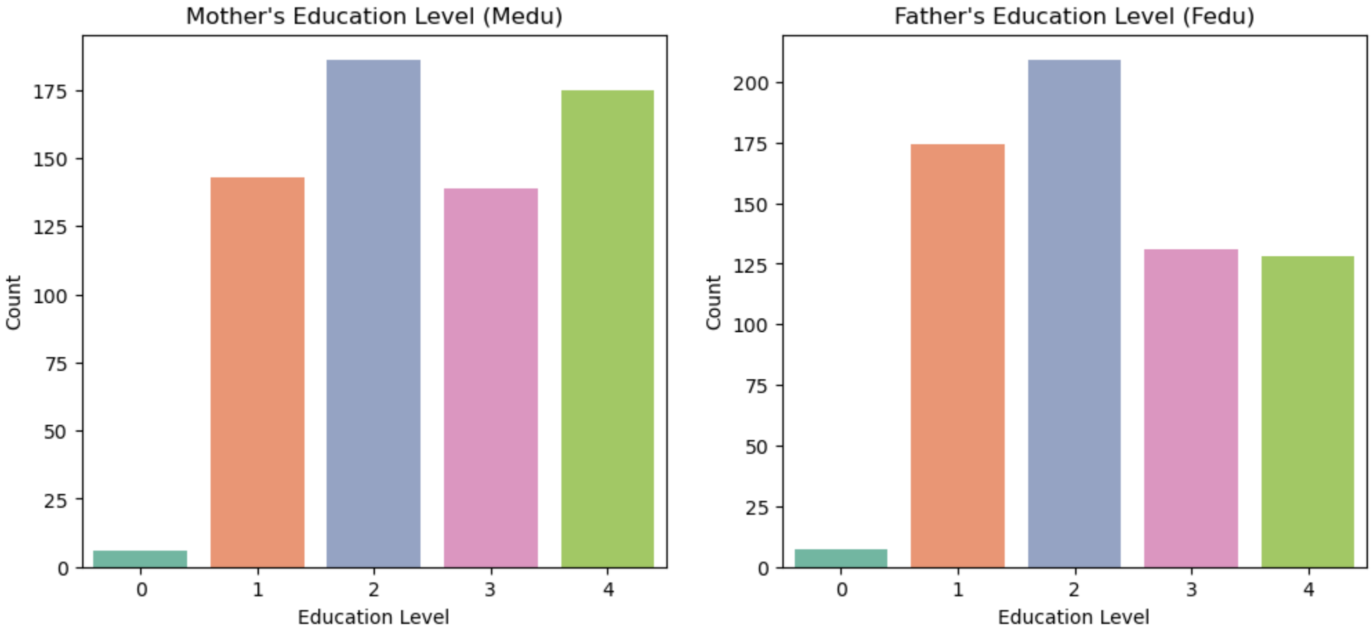
In [23]:

```
corr = df_por['Medu'].corr(df_por['Fedu'])
print(f"Correlation between Medu and Fedu: {corr:.2f}")
```

```
Correlation between Medu and Fedu: 0.65
```

In [24]:

```
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
sns.countplot(data=df_por, x='Medu', palette='Set2')
plt.title("Mother's Education Level (Medu)")
plt.xlabel("Education Level")
plt.ylabel("Count")

plt.subplot(1, 2, 2)
sns.countplot(data=df_por, x='Fedu', palette='Set2')
plt.title("Father's Education Level (Fedu)")
plt.xlabel("Education Level")
plt.ylabel("Count")
plt.show()
```

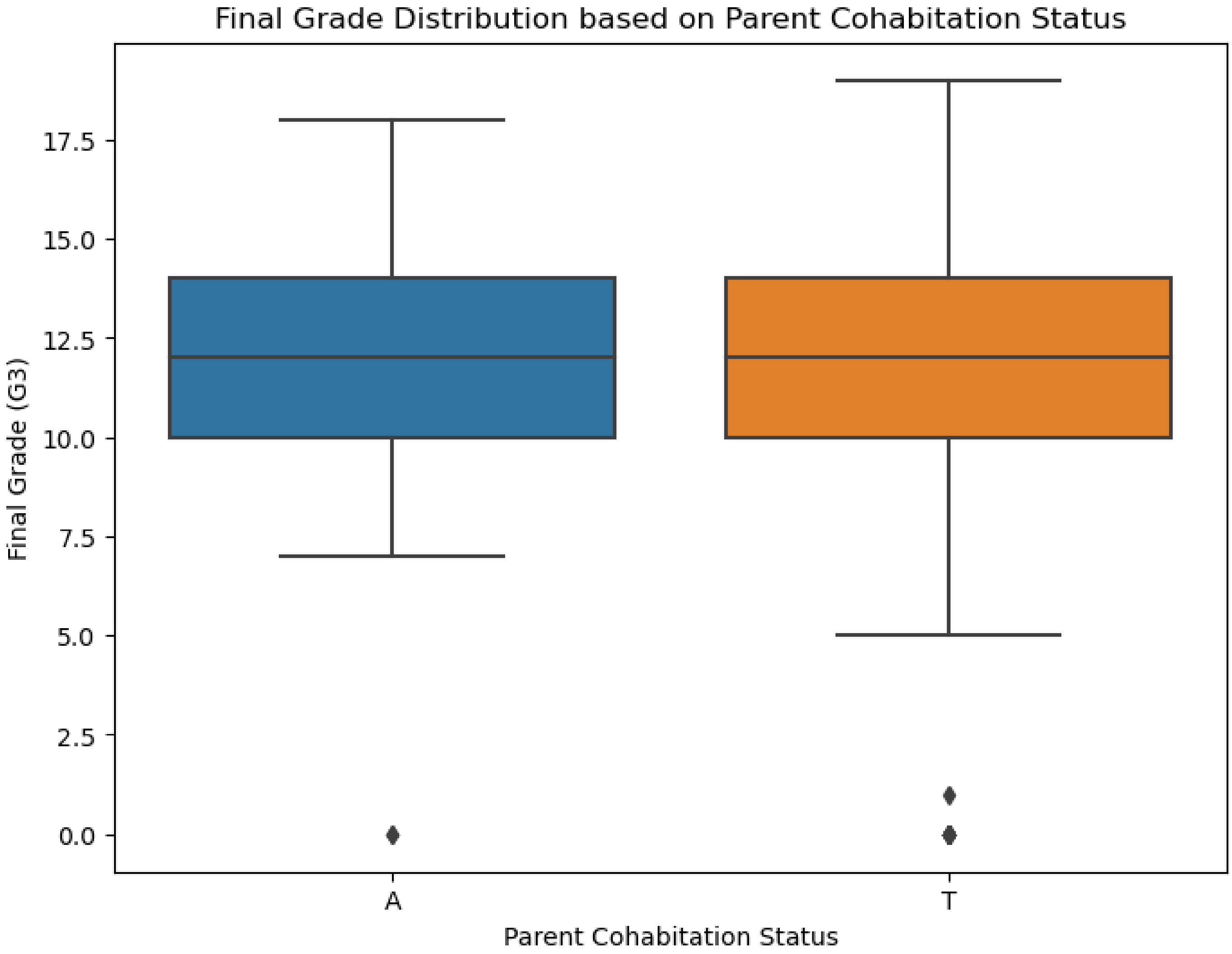


Task 12: Are there differences in students' final grades (G3) based on their parents' cohabitation status (Pstatus)?

In [25]:

```
grouped_data = df_por.groupby('Pstatus')['G3'].describe()
print(grouped_data)

plt.figure(figsize=(8, 6))
sns.boxplot(x='Pstatus', y='G3', data=df_por)
plt.xlabel('Parent Cohabitation Status')
plt.ylabel('Final Grade (G3)')
plt.title('Final Grade Distribution based on Parent Cohabitation Status')
plt.show()
```

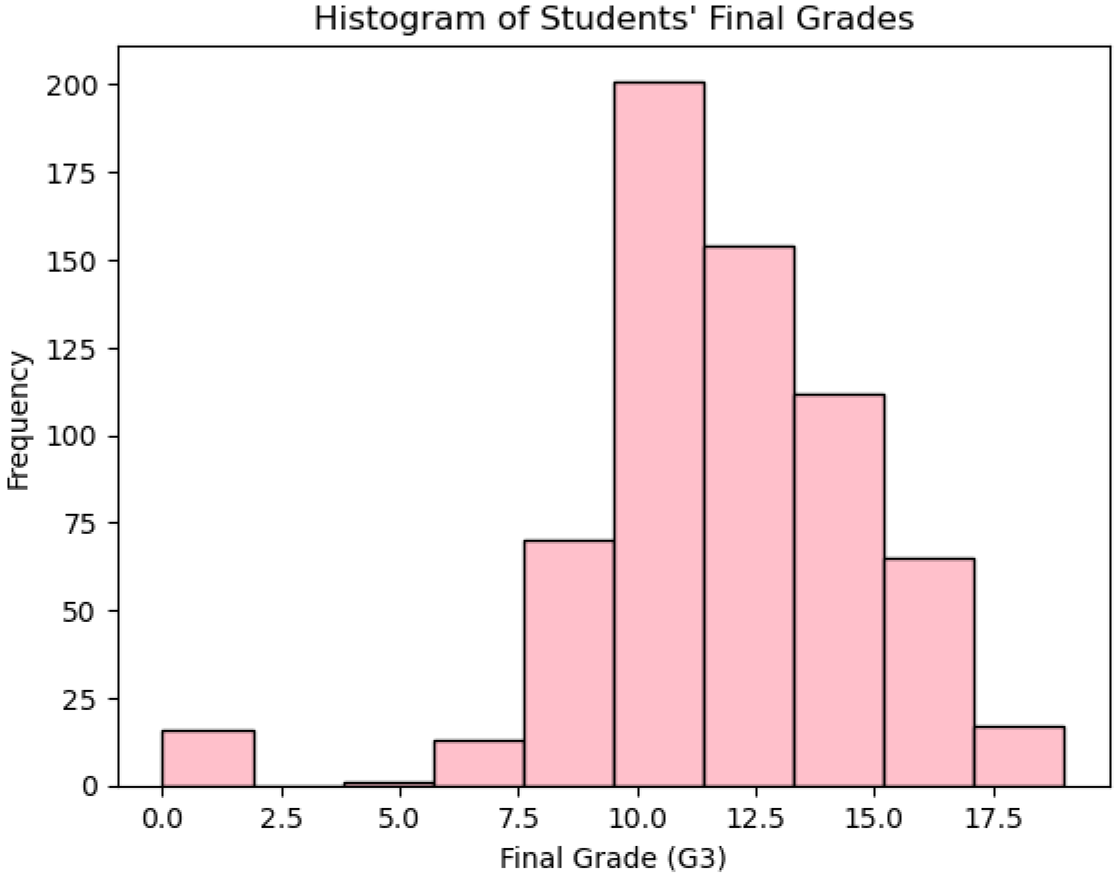
```
         count       mean       std  min   25%   50%   75%   max
Pstatus
A         80.0  11.912500  3.222523  0.0  10.0  12.0  14.0  18.0
T        569.0  11.905097  3.234626  0.0  10.0  12.0  14.0  19.0
```



Task 13: Create a histogram of students' final grades (G3) to visualize the grade distribution.

In [26]:

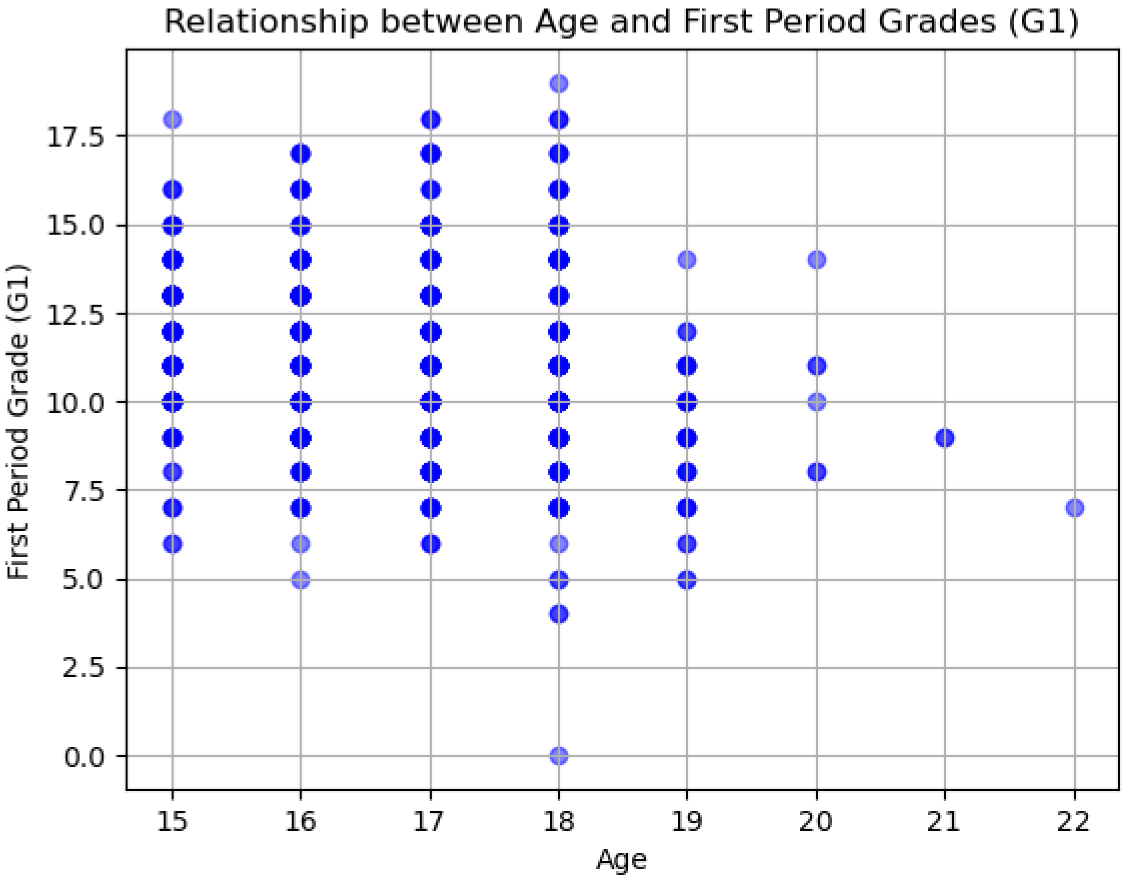
```
plt.hist(df_por['G3'], bins=10, color='pink', edgecolor='black')
plt.xlabel('Final Grade (G3)')
plt.ylabel('Frequency')
plt.title('Histogram of Students\' Final Grades')
plt.show()
```



Task 14: Generate a scatter plot to show the relationship between students' age and their first period grades (G1).

In [27]:

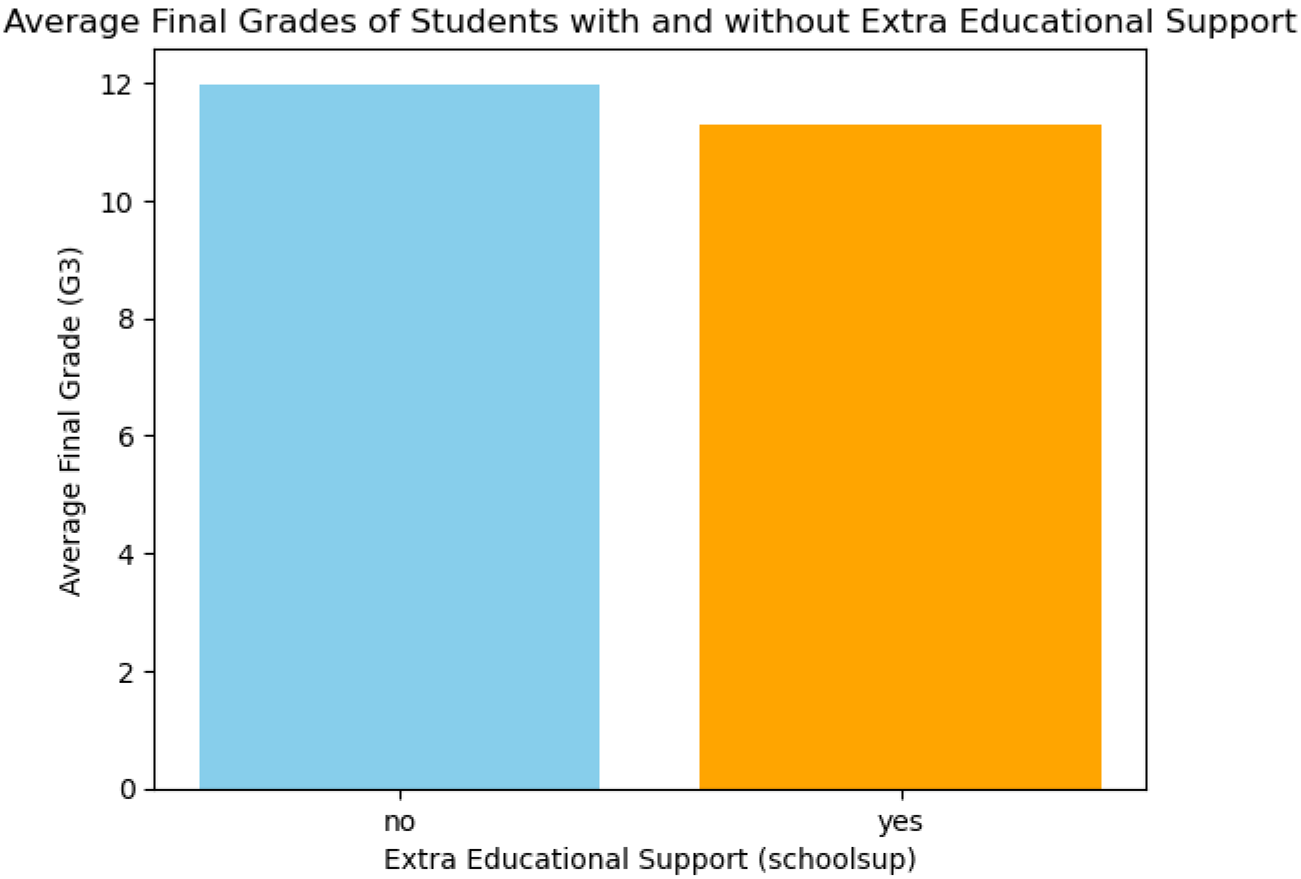
```python
plt.scatter(df_por['age'], df_por['G1'], color='blue', alpha=0.5)
plt.xlabel('Age')
plt.ylabel('First Period Grade (G1)')
plt.title('Relationship between Age and First Period Grades (G1)')
plt.grid(True)
plt.show()
```



Task 15: Create a bar chart to compare the average final grades (G3) of students with and without extra educational support (schoolsup).

In [33]:

```
avg_G3 = df_por.groupby('schoolsup')['G3'].mean().reset_index()
plt.bar(avg_G3['schoolsup'], avg_G3['G3'], color=['skyblue', 'orange'])
plt.xlabel('Extra Educational Support (schoolsup)')
plt.ylabel('Average Final Grade (G3)')
plt.title('Average Final Grades of Students with and without Extra Educational Support')
plt.show()
```



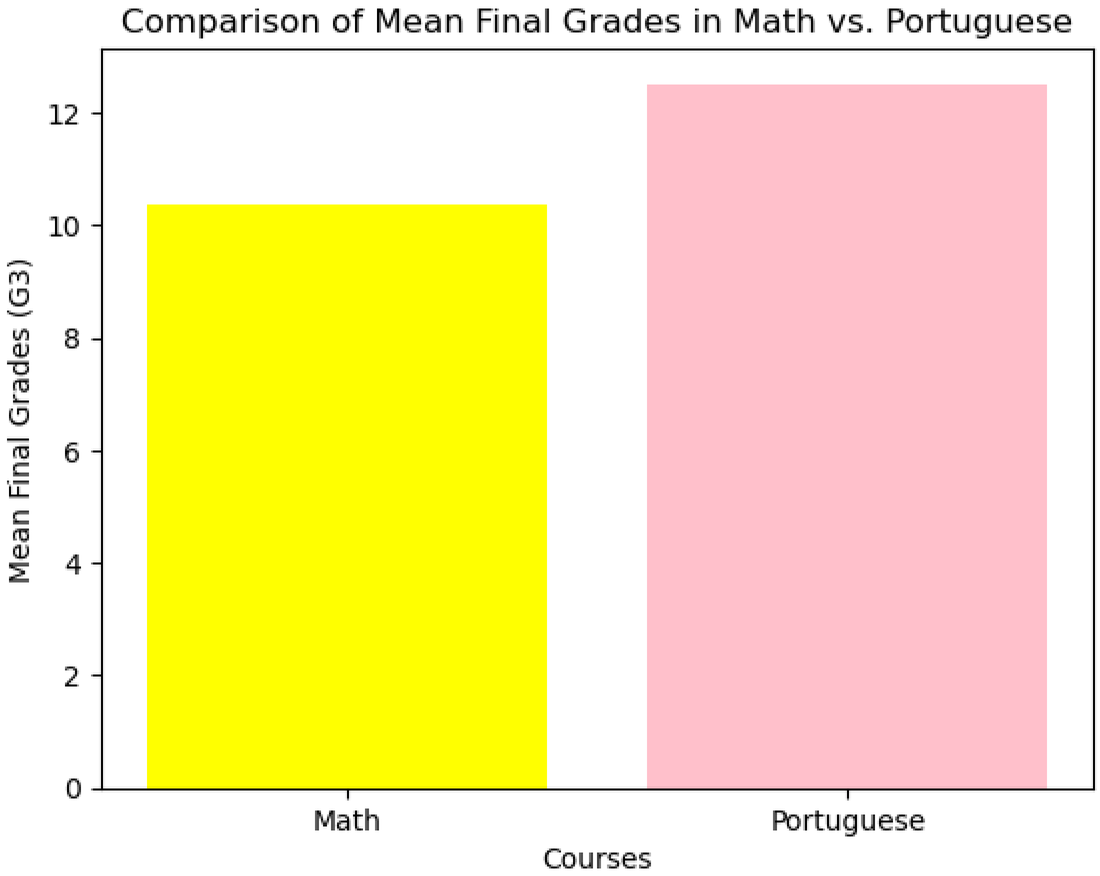
Task 16: How do final grades (G3) in the math course compare to final grades in the Portuguese course for students who belong to both datasets?

In [34]:

```
both = pd.merge(df_mat, df_por,on=["school","sex","age","address","famsize","Pstatus","M
```

In [35]:

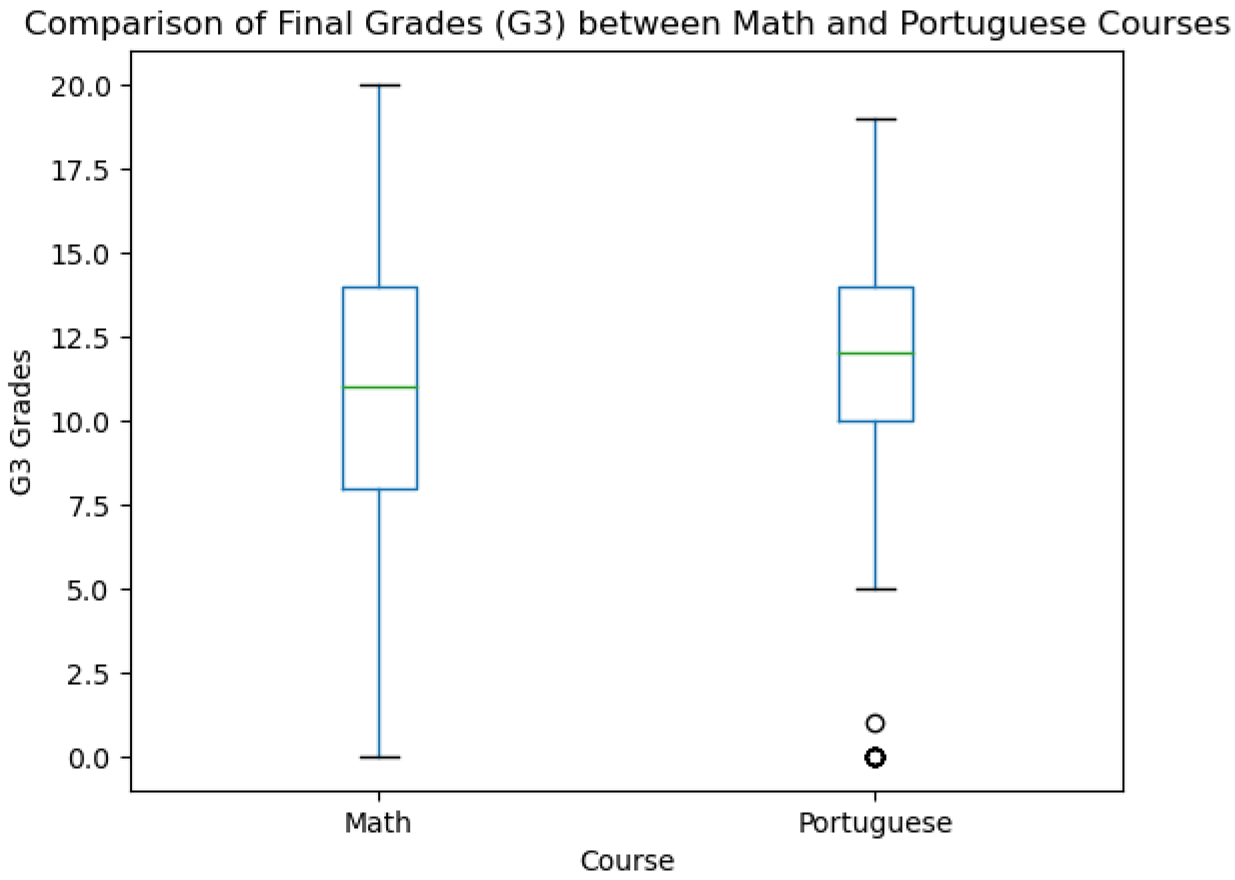
```
math_mean = both['G3_math'].mean()
port_mean = both['G3_port'].mean()

plt.bar(['Math', 'Portuguese'], [math_mean, port_mean], color=['yellow', 'pink'])
plt.xlabel('Courses')
plt.ylabel('Mean Final Grades (G3)')
plt.title('Comparison of Mean Final Grades in Math vs. Portuguese')
plt.show()
```



Task 17: Create a side-by-side box plot to compare the distribution of final grades (G3) between the math and Portuguese courses.

In [38]:

```
math_grades = df_mat['G3']
port_grades = df_por['G3']

data = pd.DataFrame({'Math': math_grades, 'Portuguese': port_grades})

data.boxplot(column=['Math', 'Portuguese'])
plt.title('Comparison of Final Grades (G3) between Math and Portuguese Courses')
plt.ylabel('G3 Grades')
plt.xlabel('Course')
plt.grid(False)
plt.show()
```

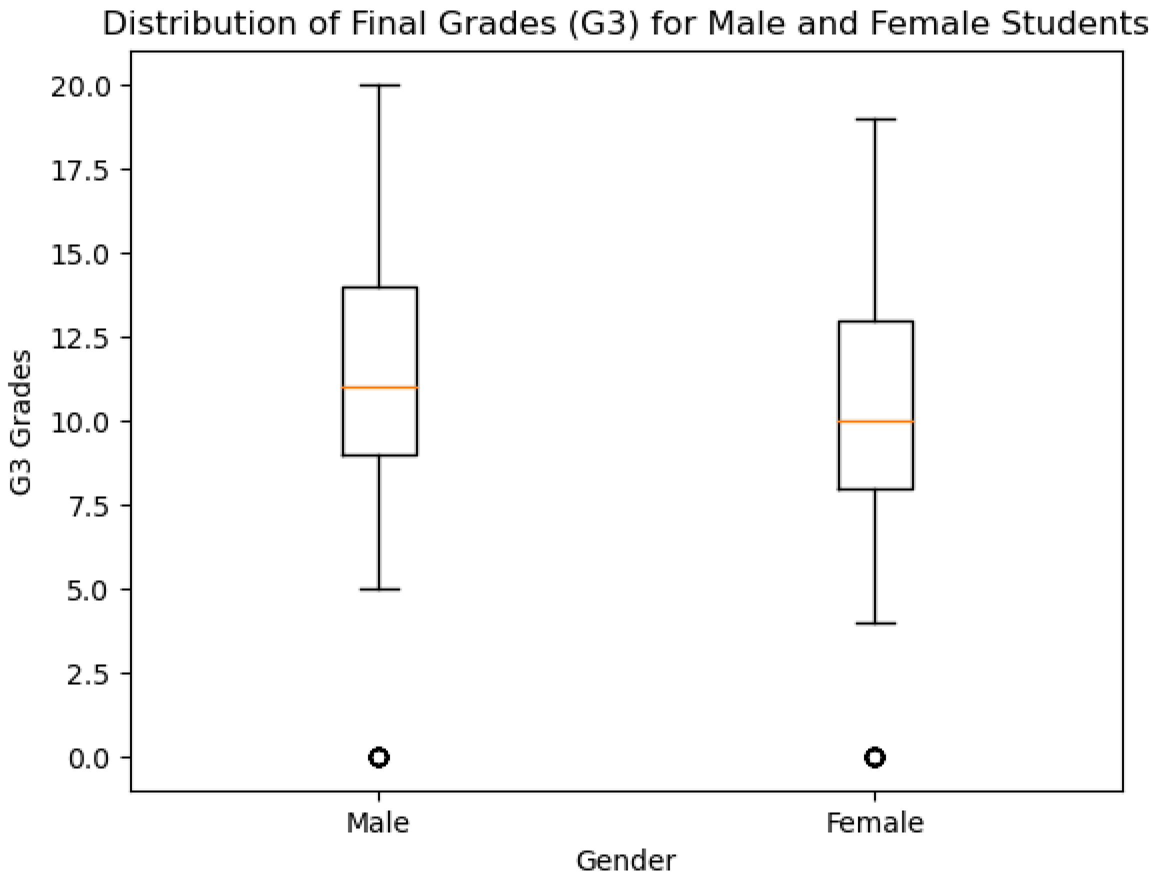


Task 18: Is there a significant difference in the average final grades (G3) between male and female students? Conduct a two-sample t-test and visualize the results.

In [39]:

```
male_grades = both[both['sex'] == 'M']['G3_math']
female_grades = both[both['sex'] == 'F']['G3_math']

t_stat, p_value = stats.ttest_ind(male_grades, female_grades)

alpha = 0.05
if p_value < alpha:
    print("There is a significant difference between the average final grades of male an
else:
    print("There is no significant difference between the average final grades of male a

plt.boxplot([male_grades, female_grades], labels=['Male', 'Female'])
plt.title('Distribution of Final Grades (G3) for Male and Female Students')
plt.xlabel('Gender')
plt.ylabel('G3 Grades')
plt.grid(False)
plt.show()
```

```
There is a significant difference between the average final grades of male and female st
udents.
```



Task 19: Can you create a new variable that categorizes students into age groups (e.g., 15-17, 18-20, 21-22)? How does this grouping affect the analysis of other variables, such as study time or final math grades (G3)?

In [41]:

```
df_mat['age_group'] = pd.cut(df_mat['age'], bins=[15, 17, 20, 22], labels=['15-17', '18-

study_time_by_age = df_mat.groupby('age_group')['studytime'].mean()

math_grades_by_age = df_mat.groupby('age_group')['G3'].mean()

print("Average study time by age group:")
print(study_time_by_age)

print("\nAverage final math grades (G3) by age group:")
print(math_grades_by_age)
```

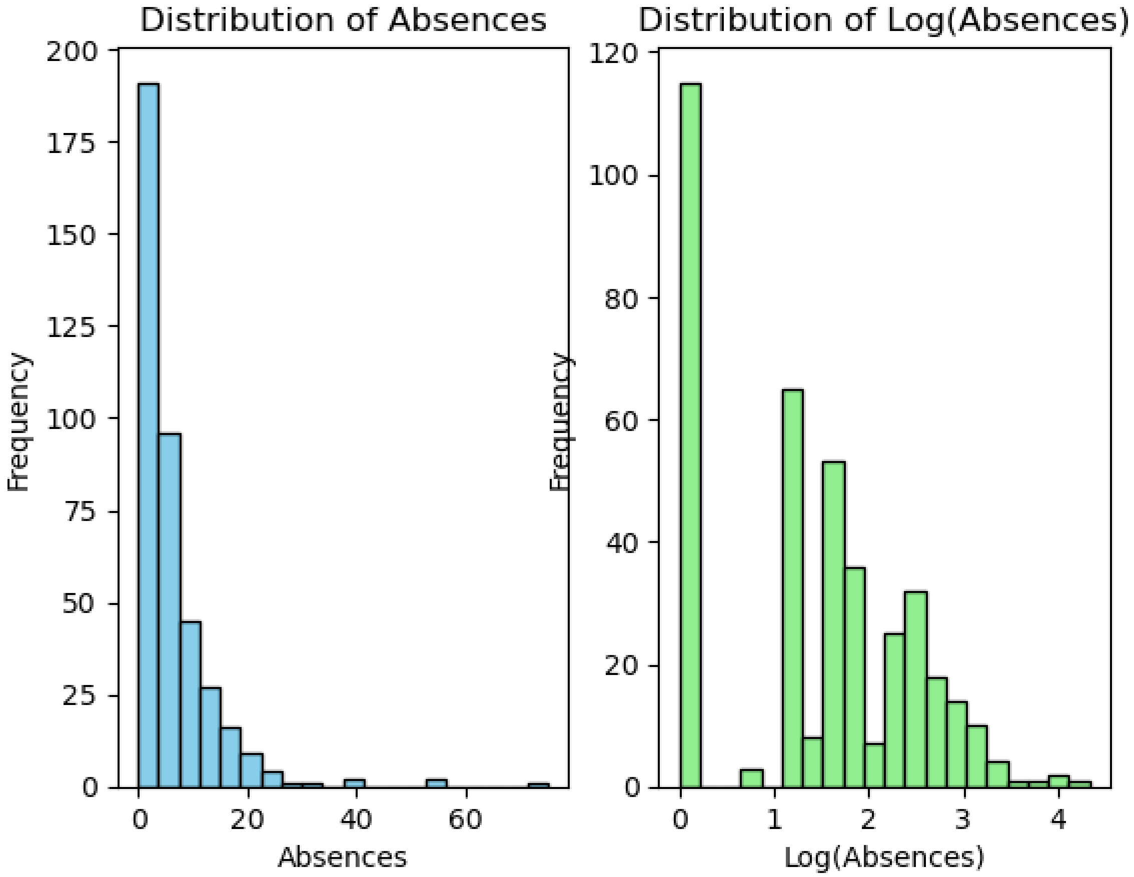
```
Average study time by age group: age_group
15-17    1.945545
18-20    2.128440
21-22    1.000000
Name: studytime, dtype: float64

Average final math grades (G3) by age group:
age_group
15-17    10.663366
18-20     9.376147
21-22     7.500000
Name: G3, dtype: float64
```

Task 20: Apply a mathematical transformation, such as logarithm or square root, to the number of school absences

(absences). How does this transformation impact the distribution of absences and its relationship with final math grades (G3)?

In [47]:

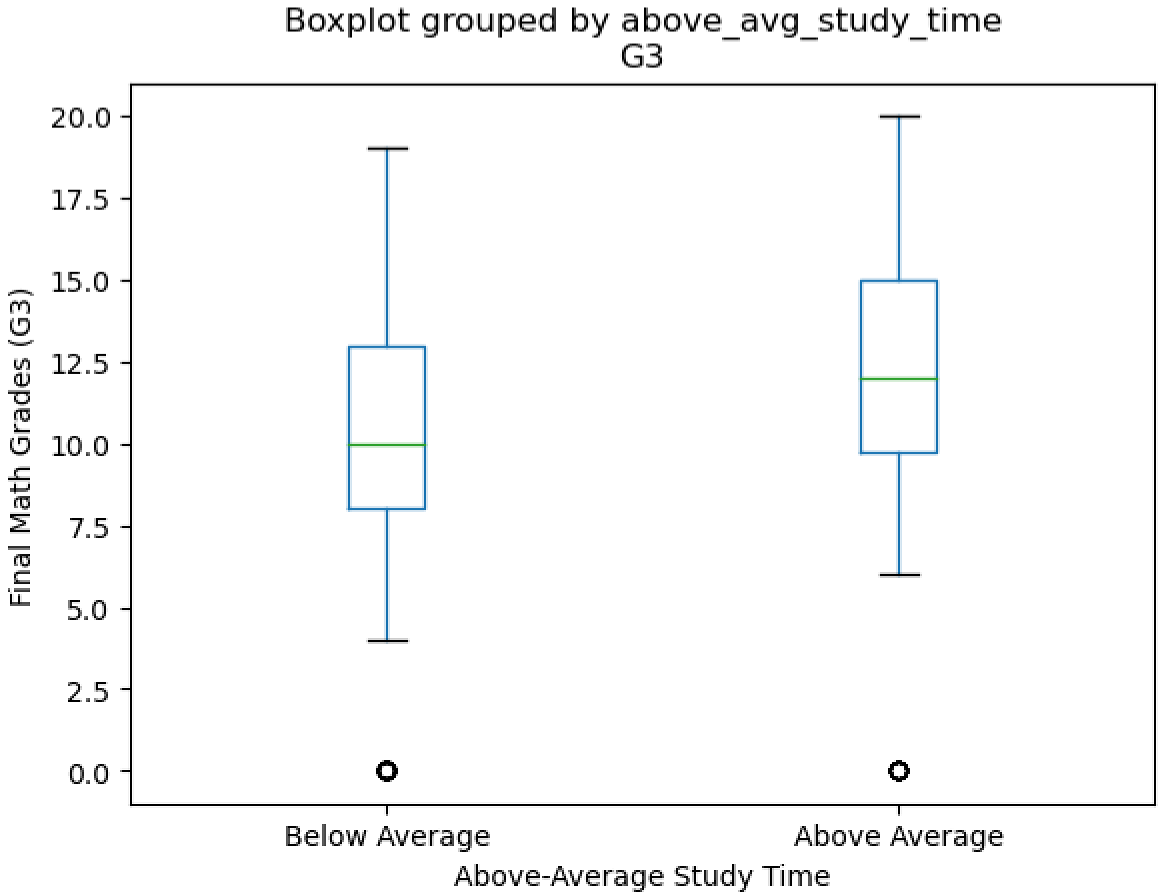
```
df_mat['log_absences'] = np.log(df_mat['absences'] + 1)

plt.subplot(1, 2, 1)
plt.hist(df_mat['absences'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Absences')
plt.xlabel('Absences')
plt.ylabel('Frequency')

plt.subplot(1, 2, 2)
plt.hist(df_mat['log_absences'], bins=20, color='lightgreen', edgecolor='black')
plt.title('Distribution of Log(Absences)')
plt.xlabel('Log(Absences)')
plt.ylabel('Frequency')

plt.show()
```



Task 21: Create a new binary variable that indicates whether a student has above-average weekly study time (studytime). How does this modified variable relate to the final math grades (G3)?

In [48]:

```
average_study_time = df_mat['studytime'].mean()
df_mat['above_avg_study_time'] = df_mat['studytime'] > average_study_time
df_mat['above_avg_study_time'] = df_mat['above_avg_study_time'].astype(int)

df_mat.boxplot(column='G3', by='above_avg_study_time', grid=False)
plt.xlabel('Above-Average Study Time')
plt.ylabel('Final Math Grades (G3)')
plt.xticks([1, 2], ['Below Average', 'Above Average'])
plt.show()
```

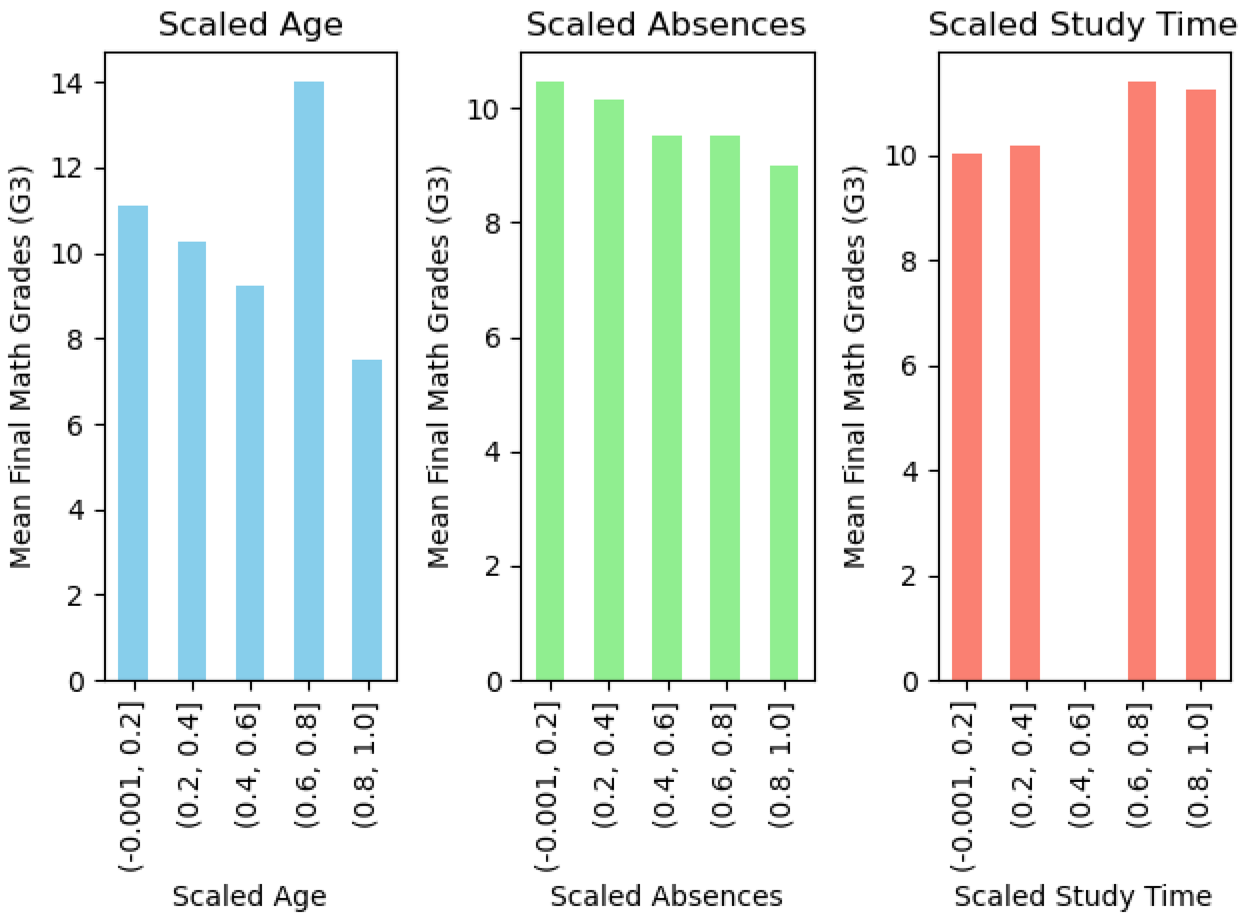


Task 22: Apply feature scaling (e.g., Min-Max scaling or standardization) to numeric variables like age, absences, and study time. How does this scaling affect the relationships between these variables and math grades (G3)?

In [53]:

```
from sklearn.preprocessing import MinMaxScaler
numeric_cols = ['age', 'absences', 'studytime']

scaler = MinMaxScaler()

df_mat[numeric_cols] = scaler.fit_transform(df_mat[numeric_cols])

age_grouped = df_mat.groupby(pd.cut(df_mat['age'], bins=5)).mean(numeric_only=True)['G3'
absences_grouped = df_mat.groupby(pd.cut(df_mat['absences'], bins=5)).mean(numeric_only=
studytime_grouped = df_mat.groupby(pd.cut(df_mat['studytime'], bins=5)).mean(numeric_onl

plt.subplot(1, 3, 1)
age_grouped.plot(kind='bar', color='skyblue')
plt.title('Scaled Age')
plt.xlabel('Scaled Age')
plt.ylabel('Mean Final Math Grades (G3)')

plt.subplot(1, 3, 2)
absences_grouped.plot(kind='bar', color='lightgreen')
plt.title('Scaled Absences')
plt.xlabel('Scaled Absences')
plt.ylabel('Mean Final Math Grades (G3)')

plt.subplot(1, 3, 3)
```

```
studytime_grouped.plot(kind='bar', color='salmon')
plt.title('Scaled Study Time')
plt.xlabel('Scaled Study Time')
plt.ylabel('Mean Final Math Grades (G3)')
plt.tight_layout()
plt.show()
```



Task 23: Convert the categorical variables (e.g., "reason" and "Mjob") into numeric format using label encoding or one-hot encoding. How does this transformation make the data suitable for analysis, and what insights can you gain?

In [50]:

```
sample_df = pd.DataFrame(df_mat)

reason_dummies = pd.get_dummies(sample_df['reason'], prefix='reason')
Mjob_dummies = pd.get_dummies(sample_df['Mjob'], prefix='Mjob')
sample_df = pd.concat([sample_df, reason_dummies, Mjob_dummies], axis=1)
sample_df.drop(['reason', 'Mjob'], axis=1, inplace=True)

sample_df
```

Out[50]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Fjob	guardian	...	above_avg_study_time
0	GP	F	0.428571	U	GT3	A	4	4	teacher	mother	...	0
1	GP	F	0.285714	U	GT3	T	1	1	other	father	...	0
2	GP	F	0.000000	U	LE3	T	1	1	other	mother	...	0
3	GP	F	0.000000	U	GT3	T	4	2	services	mother	...	1
4	GP	F	0.142857	U	GT3	T	3	3	other	father	...	0

...	...	...	...	...	...	...	...	...	...	...	...	..
390	MS	M	0.714286	U	LE3	A	2	2	services	other	...	(
391	MS	M	0.285714	U	LE3	T	3	1	services	mother	...	(
392	MS	M	0.857143	R	GT3	T	1	1	other	other	...	(
393	MS	M	0.428571	R	LE3	T	3	2	other	mother	...	(
394	MS	M	0.571429	U	LE3	T	1	1	at_home	father	...	(

395 rows × 43 columns

In [51]:

```
from sklearn.preprocessing import LabelEncoder
sample_df = pd.DataFrame(df_mat)
label_encoder = LabelEncoder()

sample_df['reason_encoded'] = label_encoder.fit_transform(sample_df['reason'])
sample_df['Mjob_encoded'] = label_encoder.fit_transform(sample_df['Mjob'])

print(label_encoder.classes_)

sample_df['reason_encoded'].head(20)
```

Out[51]:

```
['at_home' 'health' 'other' 'services' 'teacher']
0      0
1      0
2      2
3      1
4      1
5      3
6      1
7      1
8      1
9      1
10     3
11     3
12     0
13     0
14     1
15     1
16     3
17     3
18     0
19     1
Name: reason_encoded, dtype: int64
```

Task 24: Combine multiple variables (e.g., mother's education and father's education) to create a composite metric representing the overall parental education level. How does this new metric correlate with students' final math grades (G3)?

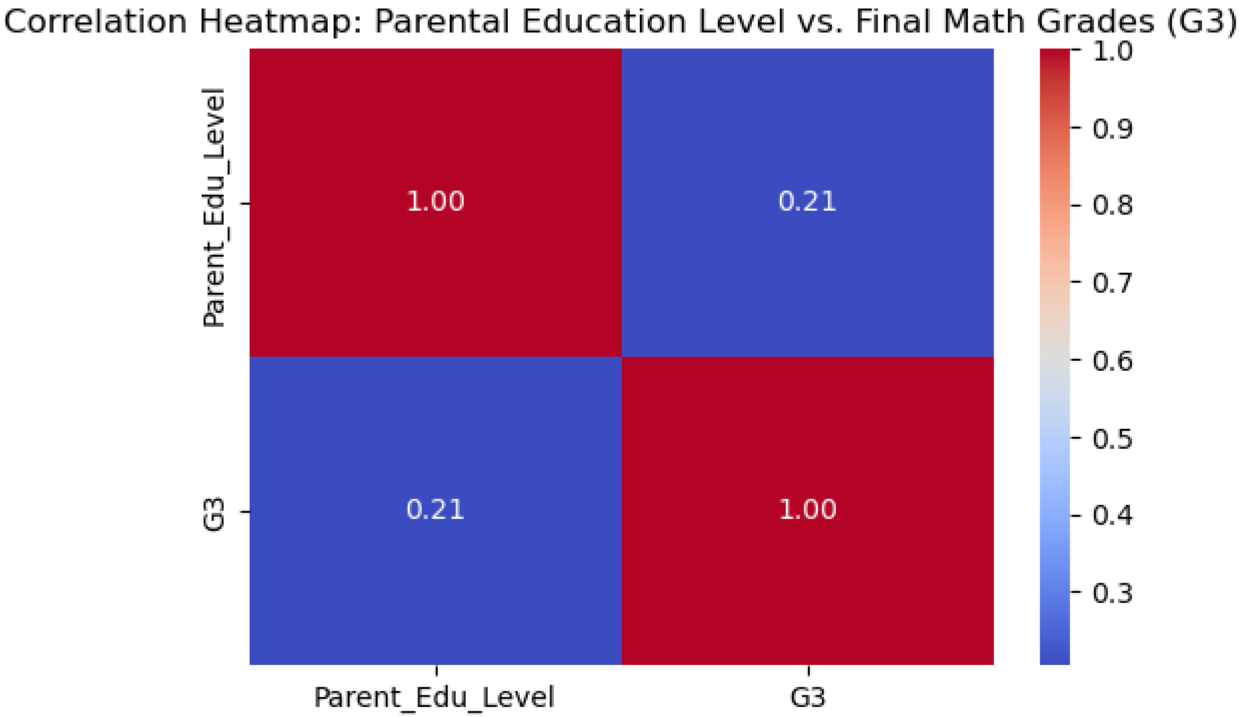
In [52]:

```
df_mat['Parent_Edu_Level'] = (df_mat['Medu'] + df_mat['Fedu']) / 2

correlation = df_mat['Parent_Edu_Level'].corr(df_mat['G3'])

print(f"Correlation between Parental Education Level and G3: {correlation}")
```

```
Correlation between Parental Education Level and G3: 0.2052244341145388
```

In [53]:

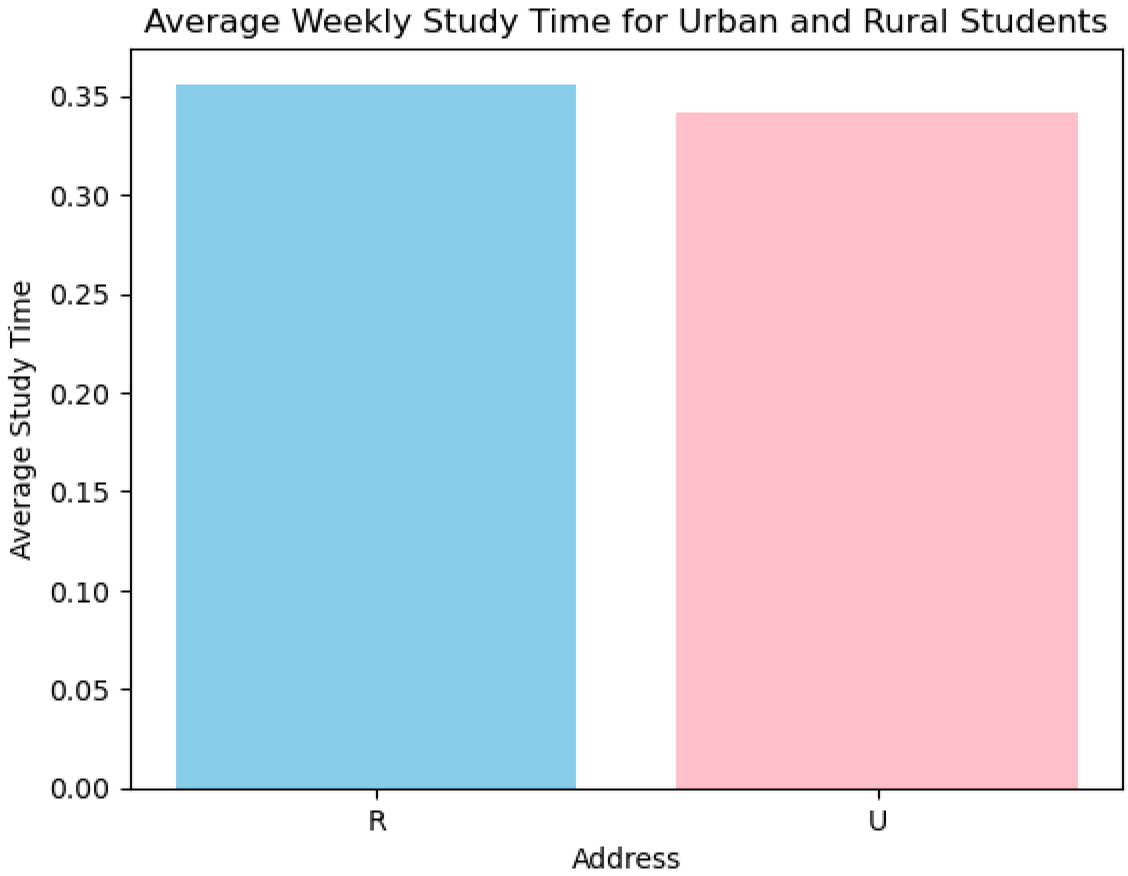
```
correlation_matrix = df_mat[['Parent_Edu_Level', 'G3']].corr()
plt.figure(figsize=(6, 4))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap: Parental Education Level vs. Final Math Grades (G3)')
plt.show()
```



Task 25: Calculate the average weekly study time for students from urban (address = 'U') and rural (address = 'R') areas. Are there differences in study time between these two groups?

In [54]:

```
average_study_time = df_mat.groupby('address')['studytime'].mean()

plt.bar(average_study_time.index, average_study_time.values, color=['skyblue', 'pink'])
plt.xlabel('Address')
plt.ylabel('Average Study Time')
plt.title('Average Weekly Study Time for Urban and Rural Students')
plt.show()

difference = average_study_time['U'] - average_study_time['R']
print(f"Difference in average study time between urban and rural students: {difference:.
```



```
Difference in average study time between urban and rural students: -0.01 hours
```

Task 26: For ordinal variables like the quality of family relationships (famrel), assign meaningful labels to the numerical values (e.g., 'very bad,' 'bad,' 'neutral,' 'good,' 'excellent'). How does this transformation make the data more interpretable?

In [52]:

```
label_mapping = {
    1: 'very bad',
    2: 'bad',
    3: 'neutral',
    4: 'good',
    5: 'excellent'
}

df_mat['famrel_labels'] = df_mat['famrel'].map(label_mapping)

print(df_mat[['famrel', 'famrel_labels']])
```

```
     famrel famrel_labels
0         4          good
1         5     excellent
2         4          good
3         3       neutral
4         4          good
..      ...           ...
390       5     excellent
391       2           bad
392       5     excellent
393       4          good
394       3       neutral

[395 rows x 2 columns]
```

Task 27: Apply custom aggregation functions to summarize the data, such as calculating the range of ages within different schools or determining the percentage of students with Internet access (internet = 'yes') by gender. What insights do these custom aggregations provide?

In [56]:

```python
def age_range(series):
    return series.max() - series.min()

age_range_by_school = df_por.groupby('school')['age'].agg(age_range)

print("Range of Ages within Different Schools:")
print(age_range_by_school)
```

```
Range of Ages within Different Schools:
school
GP    7
MS    5
Name: age, dtype: int64
```

In [57]:

```python
def percentage_yes(series):
    return (series[series == 'yes'].count() / len(series)) * 100

percentage_internet_by_gender = df_por.groupby('sex')['internet'].agg(percentage_yes)

print("Percentage of Students with Internet Access by Gender:")
print(percentage_internet_by_gender)
```

```
Percentage of Students with Internet Access by Gender:
sex
F    74.412533
M    80.075188
Name: internet, dtype: float64
```

Task 28: If relevant, consider applying date-related functions to variables, such as determining the day of the week for which students have the most absences. How does this transformation reveal patterns related to attendance?

no need

Task 29: Calculate the median number of school absences (absences) for students with and without extra educational support (schoolsup).

In [60]:

```python
med = df_por.groupby('schoolsup')['absences'].median()

print("Median Number of School Absences for Students with and without Extra Educational
print(med)
```

```
Median Number of School Absences for Students with and without Extra Educational Suppor
t:
schoolsup
no     2.0
yes    2.0
Name: absences, dtype: float64
```

Task 30: Calculate the percentage of students who want to take higher education (higher) for each level of father's education (Fedu).

In [61]:

```
percentage_higher_by_Fedu = df_por.groupby('Fedu')['higher'].apply(lambda x: (x == 'yes'

print("Percentage of Students Wanting Higher Education by Father's Education Level (Fedu
print(percentage_higher_by_Fedu)
```

```
Percentage of Students Wanting Higher Education by Father's Education Level (Fedu):
Fedu
0    100.000000
1     81.034483
2     87.559809
3     93.893130
4     98.437500
Name: higher, dtype: float64
```

Task 31: Calculate the correlation between travel time (traveltime) and final grades (G3).

In [62]:

```
correlation_traveltime_G3 = df_por['traveltime'].corr(df_por['G3'])

print(f"Correlation between Travel Time and Final Grades (G3): {correlation_traveltime_G
```

```
Correlation between Travel Time and Final Grades (G3): -0.12717296675842063
```

Task 32: Calculate the weighted average of final grades (G3) using study time (studytime) as weights.

In [63]:

```
weighted_average = (df_por['G3'] * df_por['studytime']).sum() / df_por['studytime'].sum(

print(f"Weighted Average of Final Grades (G3) using Study Time as Weights: {weighted_ave
```

```
Weighted Average of Final Grades (G3) using Study Time as Weights: 12.25
```

Task 33: Find the student with the highest weekend alcohol consumption (Walc).

In [66]:

```
students_with_highest_alcohol = df_por.sort_values(by='Walc', ascending=False)
print(f'Top 5 students with highest weekly alcohol consumption: \n')
students_with_highest_alcohol[:5]['Walc']
```

```
Top 5 students with highest weekly alcohol consumption:
```

Out[66]:

```
359    5
378    5
250    5
263    5
279    5
Name: Walc, dtype: int64
```

Task 34: Replace missing values in the 'guardian' column with 'unknown'.

In [85]:

```
not_filled = df_por['guardian'].isnull().sum()
print(f"Before: {not_filled}")
df_por['guardian'].fillna('unknown')
filled = df_por['guardian'].isna().sum()
print(f"After: {filled}")
```

```
Before: 0
After: 0
```

Task 35: Fill missing values in the 'romantic' column with the most common value.

In [42]:

```
pop = df_por['romantic'].mode()
df_por['romantic'].fillna(pop, inplace=True)
df_por['romantic'].isnull().sum()
```

Out[42]:

```
0
```

Task 36: Create a pivot table to find the maximum and minimum study times for each 'reason' for choosing the school.

In [93]:

```
pivot_table = pd.pivot_table(df_por, values='studytime', index='reason', aggfunc={'study
pivot_table
```

Out[93]:

	max	min
reason		
course	4	1
home	4	1
other	4	1
reputation	4	1

Task 37: Check if any student has 'teacher' as both mother's and father's job.

In [94]:

```
teacher = df_por[(df_por['Mjob'] == 'teacher') & (df_por['Fjob'] == 'teacher')].any()

if teacher.any():
    print("There are students with both parents as teachers.")
else:
    print("No student has both parents as teachers.")
```

```
There are students with both parents as teachers.
```

In [95]:

```
count = df_por[(df_por['Mjob'] == 'teacher') & (df_por['Fjob'] == 'teacher')].shape[0]

print(f"The number of students with both parents as teachers is: {count}")
```

```
The number of students with both parents as teachers is: 16
```

Task 38: Replace 'at_home' in the 'Mjob' and 'Fjob' columns with 'homemaker'.

In [46]:

```
df_por['Mjob'] = df_por['Mjob'].replace('at_home', 'homemaker')
df_por['Fjob'] = df_por['Fjob'].replace('at_home', 'homemaker')

print(f"Mjob: \n{df_por['Mjob'].head(5)}")
print(f"Fjob: \n{df_por['Fjob'].head(5)}")

count_m = df_por['Mjob'].value_counts()['homemaker']
count_f = df_por['Fjob'].value_counts()['homemaker']

print(f"\nCount of 'homemaker' in Mother's Job (Mjob): {count_m}")
print(f"\nCount of 'homemaker' in Father's Job (Fjob): {count_f}")
```

```
Mjob:
0    homemaker
1    homemaker
2    homemaker
3       health
4        other
Name: Mjob, dtype: object
Fjob:
0     teacher
1       other
2       other
3    services
4       other
Name: Fjob, dtype: object

Count of 'homemaker' in Mother's Job (Mjob): 135

Count of 'homemaker' in Father's Job (Fjob): 42
```

Task 39: Melt the dataset to convert the 'Mjob' and 'Fjob' columns into a single column 'ParentJob' while preserving other columns

In [106…

```
copy = df_por.copy()
melt = pd.melt(copy, id_vars=copy.columns.difference(['Mjob', 'Fjob']), value_vars=['Mjo
melt
```

Out[106]:

	Dalc	Fedu	G1	G2	G3	Medu	Pstatus	Walc	absences	activities	...	paid	reason	romantic	school
0	1	4	0	11	11	4	A	1	4	no	...	no	course	0	GP
1	1	1	9	11	11	1	T	1	2	no	...	no	course	0	GP
2	2	1	12	13	12	1	T	3	6	no	...	no	other	0	GP
3	1	2	14	14	14	4	T	1	0	yes	...	no	home	1	GP
4	1	3	11	13	13	3	T	2	0	no	...	no	home	0	GP
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1293	1	3	10	11	10	2	T	2	4	yes	...	no	course	0	MS
1294	1	1	15	15	16	3	T	1	4	no	...	no	course	0	MS
1295	1	1	11	12	9	1	T	1	6	yes	...	no	course	0	MS
1296	3	1	10	10	10	3	T	4	6	no	...	no	course	0	MS
1297	3	2	10	11	11	3	T	4	4	no	...	no	course	0	MS

1298 rows × 33 columns

Task 40: Create a custom function that assigns a letter grade (A, B, C, D, or F) based on the final grade (G3) and apply it to a new column.

In [108…

```
def assign_letter_grade(score):
    if score >= 16:
        return 'A'
    elif score >= 14:
        return 'B'
    elif score >= 12:
        return 'C'
    elif score >= 10:
        return 'D'
    else:
        return 'F'

df_por['LetterGrade'] = df_por['G3'].apply(assign_letter_grade)
df_por[['G3', 'LetterGrade']]
```

Out[108]:

	G3	LetterGrade
0	11	D
1	11	D
2	12	C
3	14	B
4	13	C
...	...	...
644	10	D
645	16	A
646	9	F
647	10	D
648	11	D

649 rows × 2 columns

Task 41: Create a time series plot showing the trend in weekly study time (studytime) over time for a specific student.

no need

Task 42: Create a new DataFrame that combines data from the Math and Portuguese courses for students who appear in both datasets.

In [117…

```
df_mat = pd.read_csv('student-mat.csv', delimiter=';')
df_por = pd.read_csv('student-por.csv', delimiter=';')
```

```
df_mat['age'] = df_mat['age'].astype(int)
df_mat['Medu'] = df_mat['Medu'].astype(int)
df_por['age'] = df_por['age'].astype(int)
df_por['Medu'] = df_por['Medu'].astype(int)

common_columns = ["school", "sex", "age", "address", "famsize", "Pstatus", "Medu", "Fedu
merged_df = pd.merge(df_mat, df_por, on=common_columns)

print('Merged dataframe:')
merged_df
```

```
Merged dataframe:
```

Out[117]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel_y	freetime_y	goo
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
377	MS	F	18	U	LE3	T	3	1	teacher	services	...	4	3	
378	MS	F	18	U	GT3	T	1	1	other	other	...	3	4	
379	MS	F	18	U	GT3	T	1	1	other	other	...	1	1	
380	MS	M	17	U	LE3	T	3	1	services	services	...	2	4	
381	MS	M	18	R	LE3	T	3	2	services	other	...	4	4	

382 rows × 53 columns

Task 43: Calculate and list the top 5 students with the highest final grades (G3) in the 'GP' school.

In [124...

```
both = pd.merge(df_mat, df_por,on=["school","sex","age","address","famsize","Pstatus","M
```

In [126...

```
gp = both[both['school'] == 'GP']
top_gp = gp.nlargest(5, 'G3_math')
print("Top 5 students with the highest final grades in 'GP' school:")
print(top_gp[['school', 'G3_math']])
```

```
Top 5 students with the highest final grades in 'GP' school:
    school  G3_math
47      GP       20
8       GP       19
116     GP       19
119     GP       19
292     GP       19
```

Task 44: Create a bar chart showing the distribution of students' travel times (traveltime) in the 'MS' school.

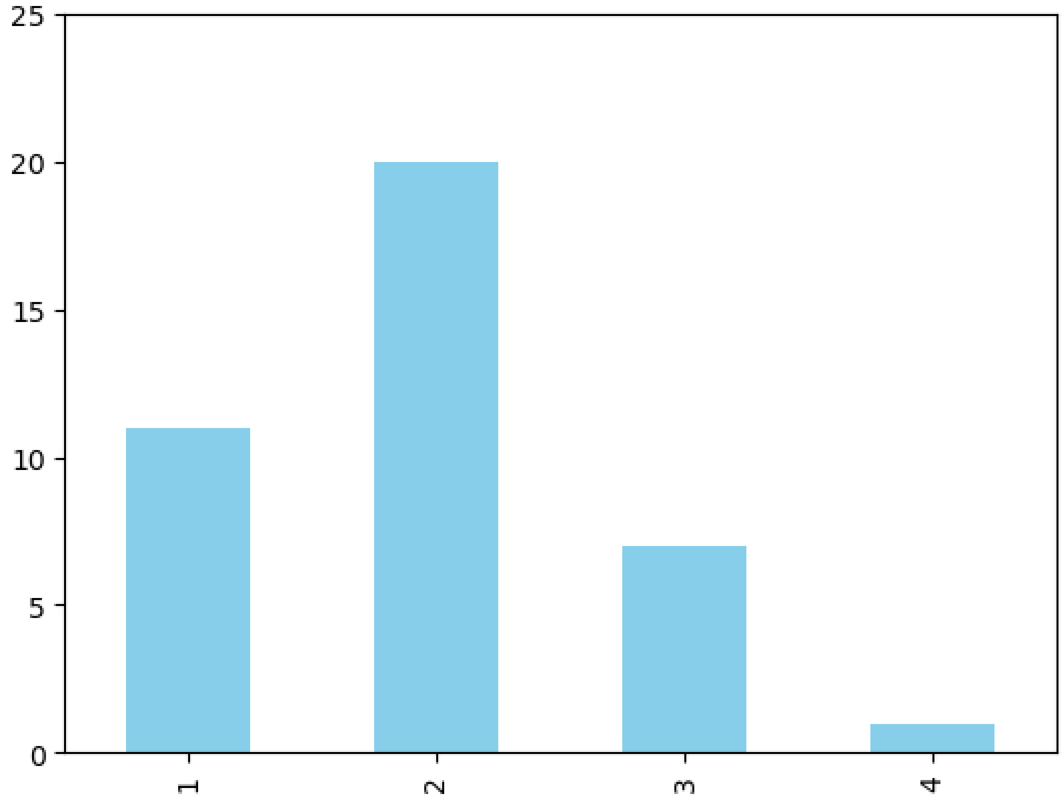
In [141...

```
both = pd.merge(df_mat, df_por,on=["school", "traveltime", "sex","age","address","famsiz
```

In [139...

```
ms_students = both[both['school'] == 'MS']
```

```
count = ms_students['traveltime'].value_counts().sort_index()
count.plot(kind='bar', color='skyblue')
plt.ylim(0, 25)
plt.show()
```



Task 45: Compute the mean age of students who have extra-curricular activities (activities) and those who don't.

In [146…

```
both = pd.merge(df_mat, df_por,on=["school", "activities" , "absences" ,"traveltime", "s
```

In [143…

```
activities = both[both['activities'] == 'yes']['age'].mean()
no_activities = both[both['activities'] == 'no']['age'].mean()
print(f"Mean age of students with activities: {activities:.2f}")
print(f"Mean age of students without activities: {no_activities:.2f}")
```

```
Mean age of students with activities: 16.49
Mean age of students without activities: 16.67
```

Task 46: Group the data by 'sex' and 'address,' and find the median number of school absences for each group.

In [150…

```
absmed = both.groupby(['sex', 'address'])['absences'].median()
absmed
```

Out[150]:

```
sex  address
F    R          2.0
     U          0.0
M    R          2.5
     U          0.0
Name: absences, dtype: float64
```

Task 47: Calculate the percentage of students who receive

extra educational support (schoolsup) in the 'GP' school.

In [156…

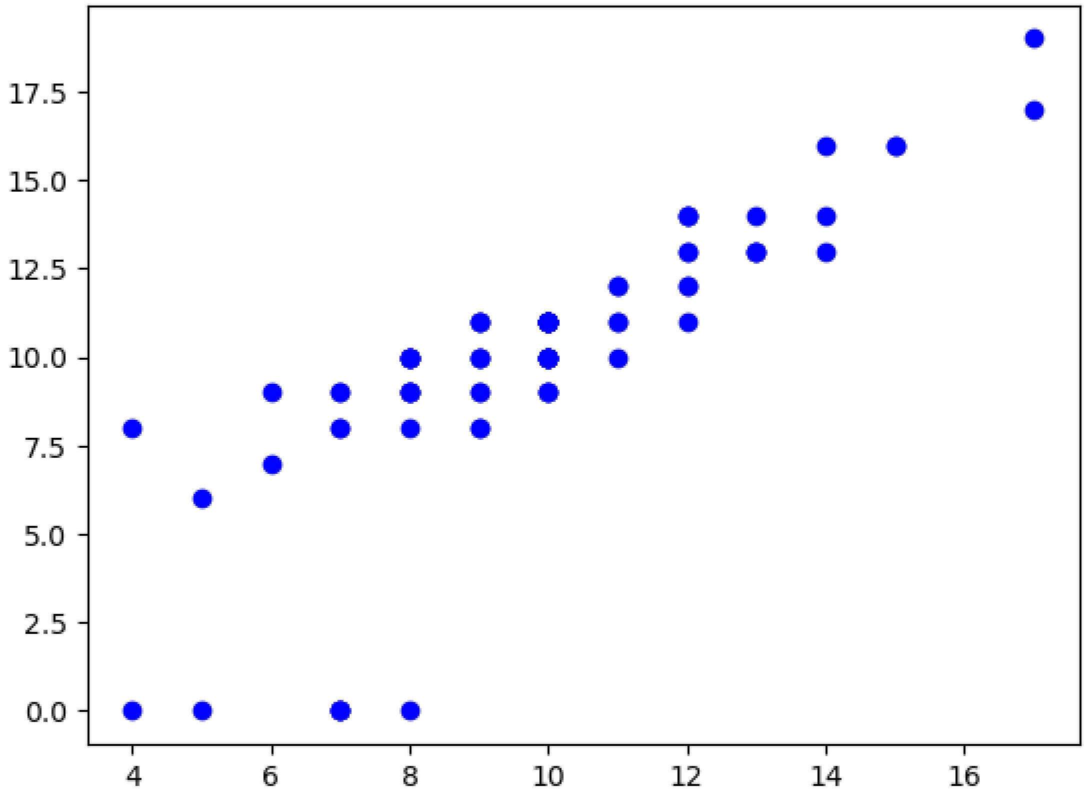
```
gp = df_por[df_por['school'] == 'GP']
total = gp.shape[0]
yes = gp[gp['schoolsup'] == 'yes'].shape[0]
no = gp[gp['schoolsup'] == 'no'].shape[0]
per = (yes / total) * 100
per = (no / total) * 100

print(f"yes: {per:.2f}%")
print(f"no: {per:.2f}%")
```

```
yes: 86.76%
no: 86.76%
```

Task 48: Create a scatter plot of 'G1' versus 'G3' for male students from the 'MS' school.

In [157…

```
ms_male = df_por[(df_por['school'] == 'MS') & (df_por['sex'] == 'M')]

plt.scatter(ms_male['G1'], ms_male['G3'], color='blue')
plt.show()
```



Task 49: Identify students with a unique combination of 'Mjob' and 'Fjob' that appears only once in the dataset.

In [159…

```
uni = df_por.groupby(['Mjob', 'Fjob']).size()
once = uni[uni == 1].reset_index()
stud = df_por[(df_por['Mjob'].isin(once['Mjob'])) & (df_por['Fjob'].isin(once['Fjob']))]
stud[['Mjob', 'Fjob']]
```

Out[159]:

	Mjob	Fjob
588	health	at_home

Task 50: Calculate the average final grade (G3) for students from 'GP' and 'MS' schools in each 'studytime' category.

In [161…

```
avg = df_por.groupby(['school', 'studytime'])['G3'].mean()
avg
```

Out[161]:

```
school  studytime
GP      1            11.529412
        2            12.733010
        3            13.563380
        4            13.407407
MS      1             9.967742
        2            10.757576
        3            12.307692
        4            11.875000
Name: G3, dtype: float64
```

In [ ]: