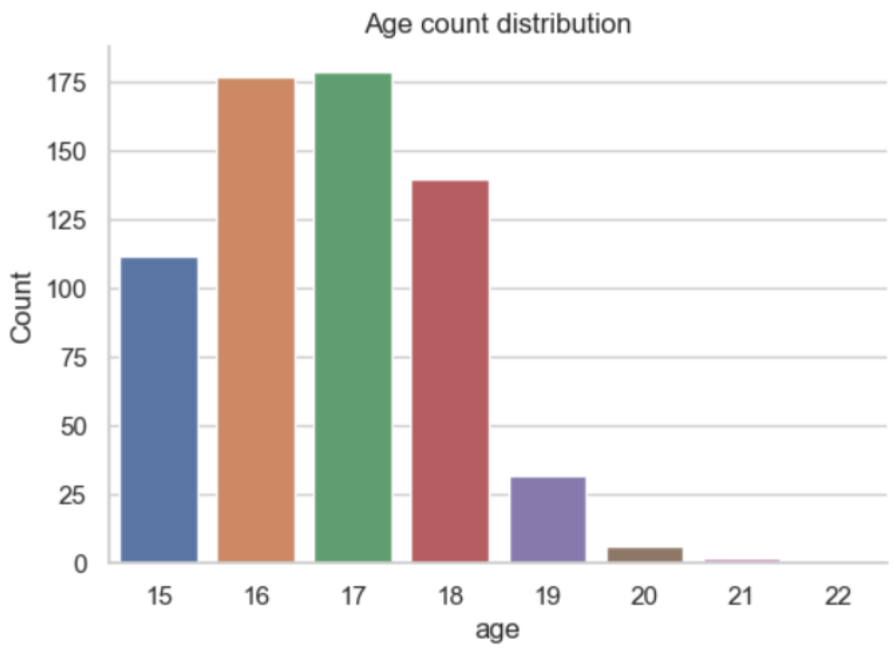
Report

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

```
fig, ax = plt.subplots()
fig.set_size_inches(20, 8)
sns.countplot(x='age', data=df)

ax.set_xlabel('age')
ax.set_ylabel('Count')
ax.set_title('Age count distribution')
sns.despine()
```



Explanation:

Imagine you have a big chart that shows how many students there are for each age. Each age gets a bar, and the taller the bar, the more students of that age there are.

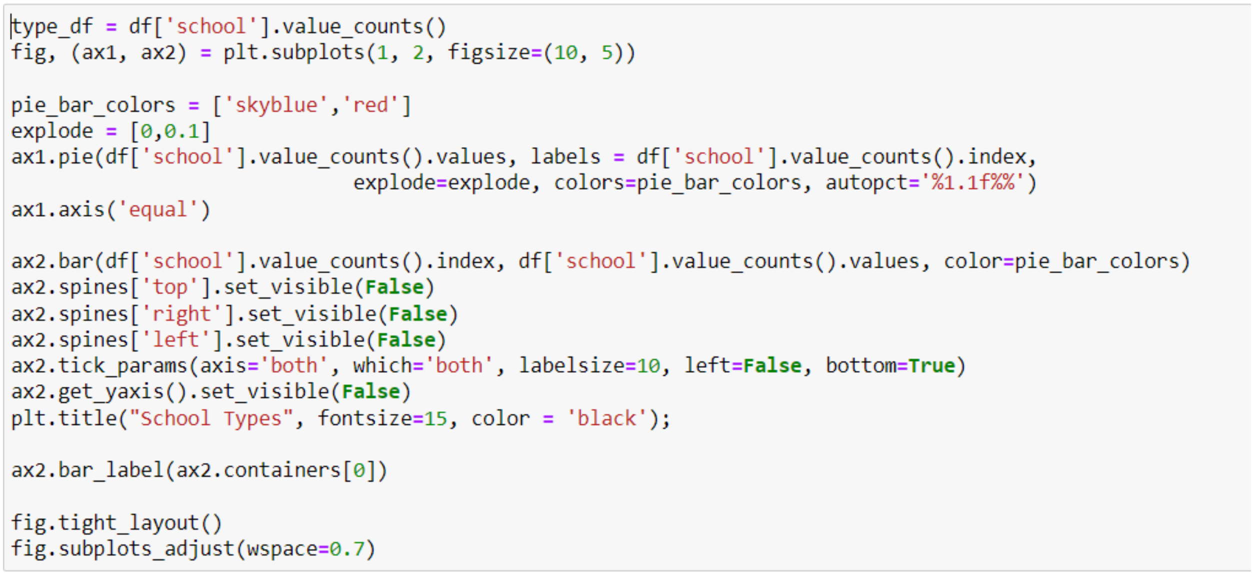
So, if you look at the chart:

You see bars for different ages (like 15, 16, 17...).

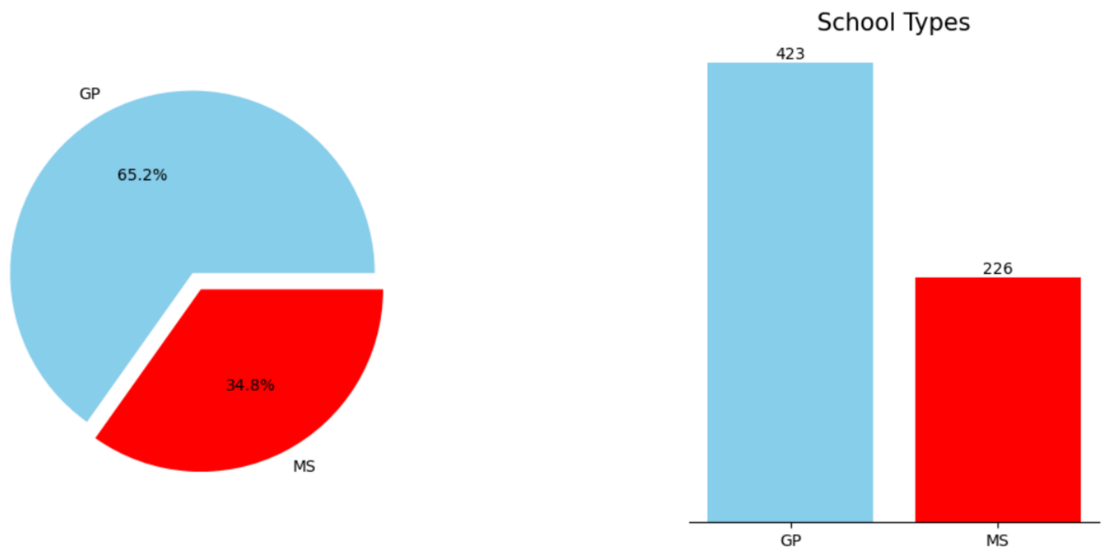
The higher the bar, the more students at that age.

The title at the top says it's all about showing the "Age count distribution."

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



```python
type_df = df['school'].value_counts()
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

pie_bar_colors = ['skyblue','red']
explode = [0,0.1]
ax1.pie(df['school'].value_counts().values, labels = df['school'].value_counts().index,
                             explode=explode, colors=pie_bar_colors, autopct='%1.1f%%')
ax1.axis('equal')

ax2.bar(df['school'].value_counts().index, df['school'].value_counts().values, color=pie_bar_colors)
ax2.spines['top'].set_visible(False)
ax2.spines['right'].set_visible(False)
ax2.spines['left'].set_visible(False)
ax2.tick_params(axis='both', which='both', labelsize=10, left=False, bottom=True)
ax2.get_yaxis().set_visible(False)
plt.title("School Types", fontsize=15, color = 'black');

ax2.bar_label(ax2.containers[0])

fig.tight_layout()
fig.subplots_adjust(wspace=0.7)
```



Explanation:

The final result is a visually appealing comparison of school types using a pie chart and a horizontal bar chart. The pie chart shows the percentage distribution, and the horizontal bar chart provides a count-based perspective.

This code is like looking at a group of students and figuring out how many go to each of the two schools, GP and MS. It gives you two pictures:

Pie Picture:

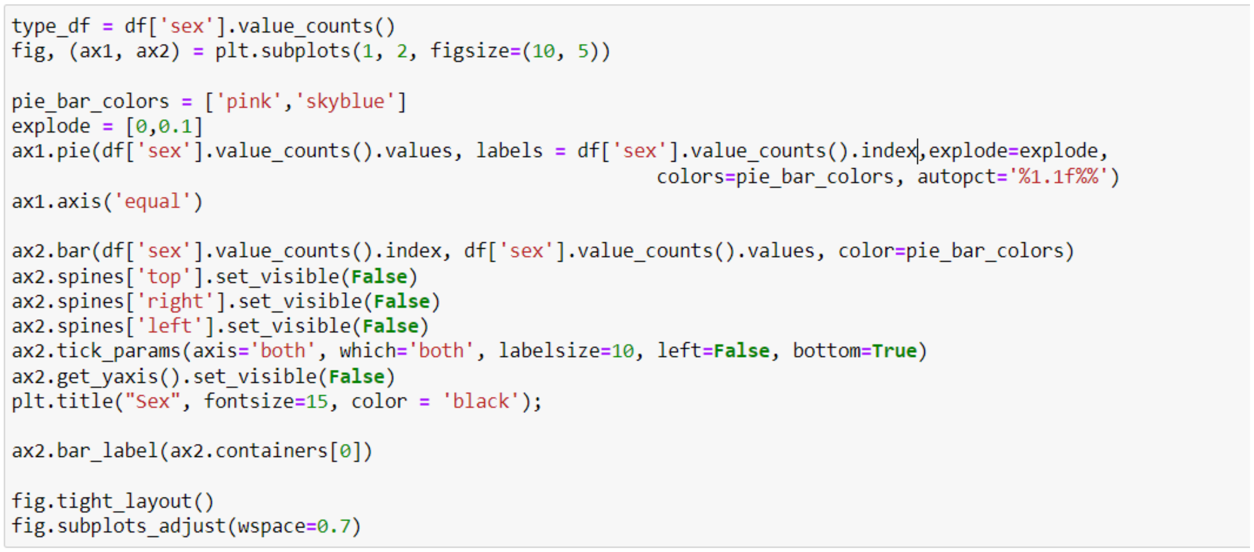
It's like cutting a pie into slices.
Each slice represents a school.
The bigger the slice, the more students in that school.
There's one slice that pops out a bit to catch your attention.

Bar Picture:

Think of bars side by side, like building blocks.
Each bar is for one school.
The taller the bar, the more students in that school.
No numbers on the side, just the height of the bars.

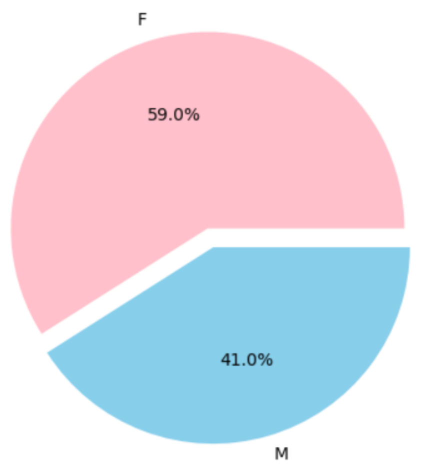
.

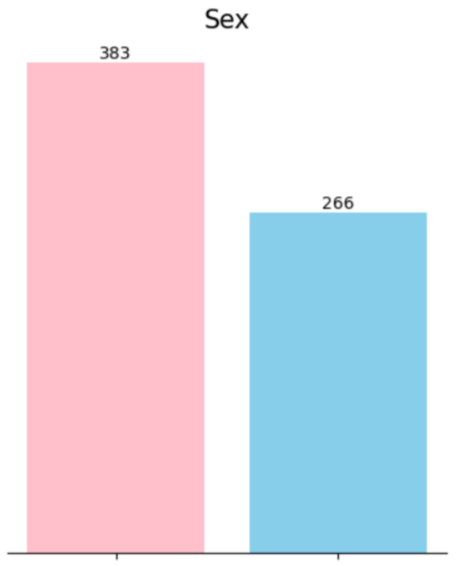
3)What is the gender distribution of students?



```
type_df = df['sex'].value_counts()
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

pie_bar_colors = ['pink','skyblue']
explode = [0,0.1]
ax1.pie(df['sex'].value_counts().values, labels = df['sex'].value_counts().index,explode=explode,
                                                         colors=pie_bar_colors, autopct='%1.1f%%')
ax1.axis('equal')

ax2.bar(df['sex'].value_counts().index, df['sex'].value_counts().values, color=pie_bar_colors)
ax2.spines['top'].set_visible(False)
ax2.spines['right'].set_visible(False)
ax2.spines['left'].set_visible(False)
ax2.tick_params(axis='both', which='both', labelsize=10, left=False, bottom=True)
ax2.get_yaxis().set_visible(False)
plt.title("Sex", fontsize=15, color = 'black');

ax2.bar_label(ax2.containers[0])

fig.tight_layout()
fig.subplots_adjust(wspace=0.7)
```

The final result is a side-by-side comparison of the distribution of sexes using a pie chart and a horizontal bar chart. The pie chart shows the percentage distribution, and the horizontal bar chart provides a count-based perspective.





Explanation:

This code is like looking at a group of students and finding out how many are boys and how many are girls. It shows you two pictures:

Pie Picture:

Think of a pie sliced into two parts.
One slice is for boys, the other for girls.
The bigger the slice, the more students of that gender.
One slice pops out a bit to grab your attention.
The percentages tell you the proportion of boys and girls.

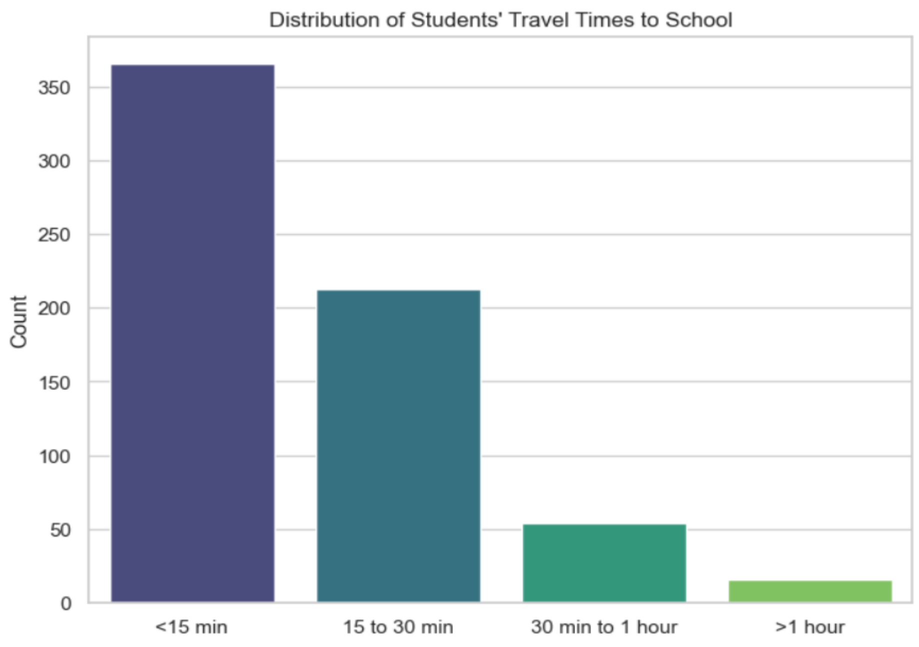
Bar Picture:

Picture two bars standing side by side.
One bar is for boys, the other for girls.
The taller the bar, the more students of that gender.
No numbers on the side, just the height of the bars.

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

```
traveltime_counts = df['traveltime'].value_counts()

plt.figure(figsize=(10,8))
sns.set(style='whitegrid')
sns.barplot(x=traveltime_counts.index, y=traveltime_counts.values, palette='viridis')

plt.title("Distribution of Students' Travel Times to School")
plt.xticks([0, 1, 2, 3], ['<15 min', '15 to 30 min', '30 min to 1 hour', '>1 hour'])
plt.ylabel("Count")

plt.show()
```

The final result is a bar plot that visualizes the distribution of students' travel times to school, with custom labels on the x-axis representing different time intervals. The heights of the bars represent the counts of students for each travel time category. The color palette 'viridis' is used to enhance the visual appeal of the plot.

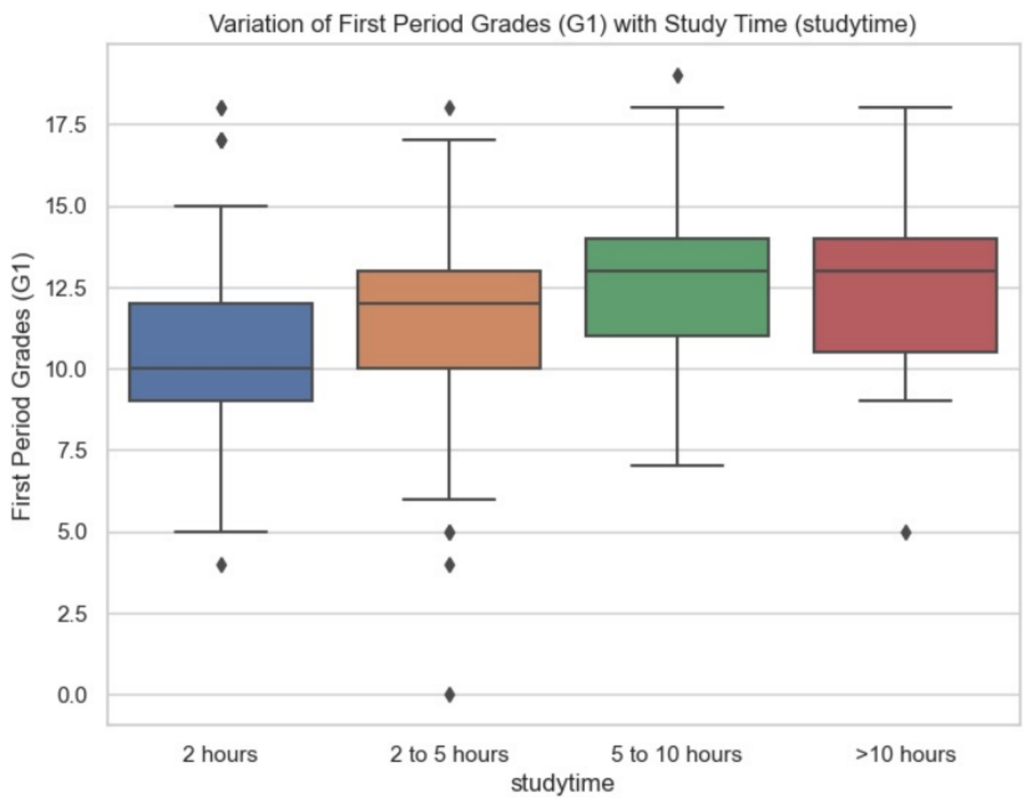


Explanation:

The plot visually represents the distribution of students' travel times to school. The x-axis shows different travel time categories, and the y-axis represents the count of students for each category. The explanation concludes by noting that the plot provides a clear visualization of the distribution, showing, for example, how many students have travel times less than 15 minutes, between 15 and 30 minutes, between 30 minutes and 1 hour, and more than 1 hour.

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

```python
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 6))
sns.boxplot(x='studytime', y='G1', data=df)

plt.title("Variation of First Period Grades (G1) with Study Time (studytime)")
plt.xticks([0, 1, 2, 3], ['2 hours', '2 to 5 hours', '5 to 10 hours', '>10 hours'])
plt.ylabel("First Period Grades (G1)")
plt.show()
```



Explanation:

The boxplot visually illustrates the distribution of first period grades (G1) for each study time category. It provides information about the median, quartiles, and potential outliers in the dataset, allowing for a quick comparison of how G1 varies across different study time levels. The explanation concludes by noting that the plot helps in understanding the relationship between study time and academic performance in the first period.

plt.xticks([0, 1, 2, 3], ['2 hours', '2 to 5 hours', '5 to 10 hours', '>10 hours']) sets custom labels for the x-axis to represent different study time categories.

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

```python
import pandas as pd

correlation = df['studytime'].corr(df['G3'])

print(f"Correlation between study time and final grades (G3): {correlation}")
```

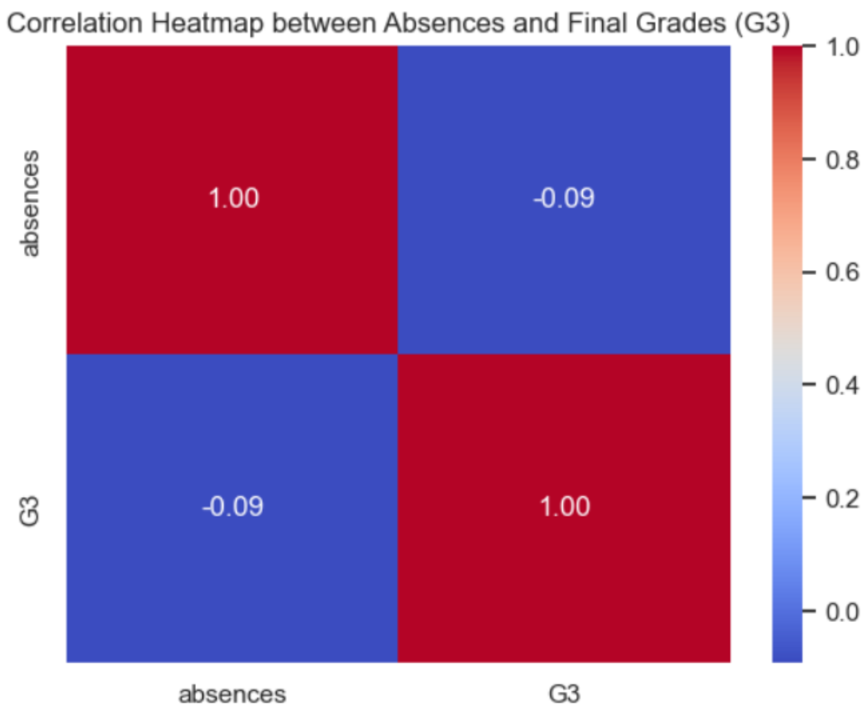
```
Correlation between study time and final grades (G3): 0.249788689998863
```

The printed result will show the correlation between study time and final grades (G3). If the correlation coefficient is positive, it indicates a positive correlation (higher study time associated with higher final grades), and if it's negative, it indicates a negative correlation (higher study time associated with lower final grades). The magnitude of the correlation coefficient provides information about the strength of the relationship.

OUTPUT: Correlation between study time and final grades (G3): 0.249788689998863

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

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

correlation_matrix = df[['absences', 'G3']].corr()

correlation = df['absences'].corr(df['G3'])

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap between Absences and Final Grades (G3)')
plt.show()
print(f'Correlation between absences and final grades(G3): {correlation}')
```

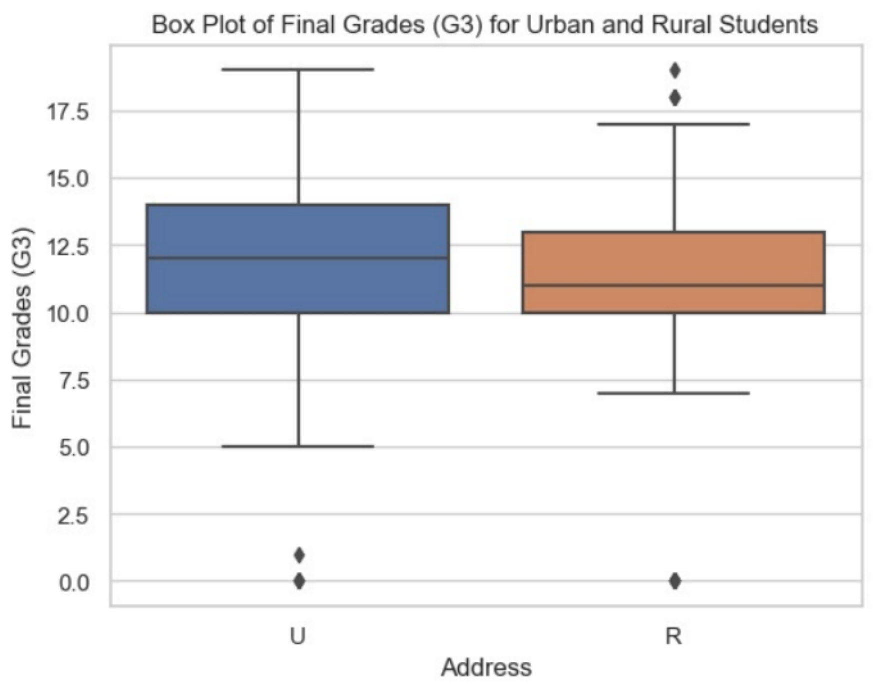


Explanation:

The heatmap visually represents the strength and direction of the correlation between absences and final grades (G3). Additionally, the printed correlation coefficient provides a numeric measure of the linear relationship between these two variables. Positive values indicate a positive correlation, negative values indicate a negative correlation, and values close to zero suggest a weak or no linear relationship.

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

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.boxplot(x='address', y='G3', data=df)
plt.title('Box Plot of Final Grades (G3) for Urban and Rural Students')
plt.xlabel('Address')
plt.ylabel('Final Grades (G3)')
plt.show()
```

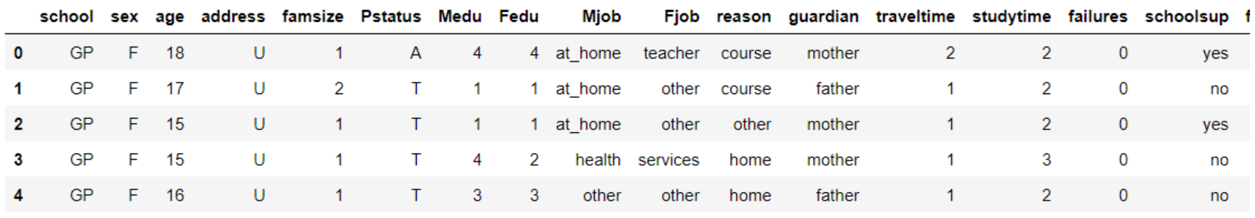


Explanation:

The boxplot visually compares the distribution of final grades (G3) for students living in urban and rural areas. It provides information about the median, quartiles, and potential outliers for each category. This type of plot is useful for quickly identifying any differences in the distribution of final grades between students in urban and rural areas.

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

```
random_GT3 = np.random.randint(1, 3, 193)
random_GT3

random_LE3 = np.random.randint(3, 8, 456)
random_LE3

rezult = np.hstack((random_GT3, random_LE3))

dataframe = pd.DataFrame(rezult)
df['famsize'] = dataframe

df.head()
```

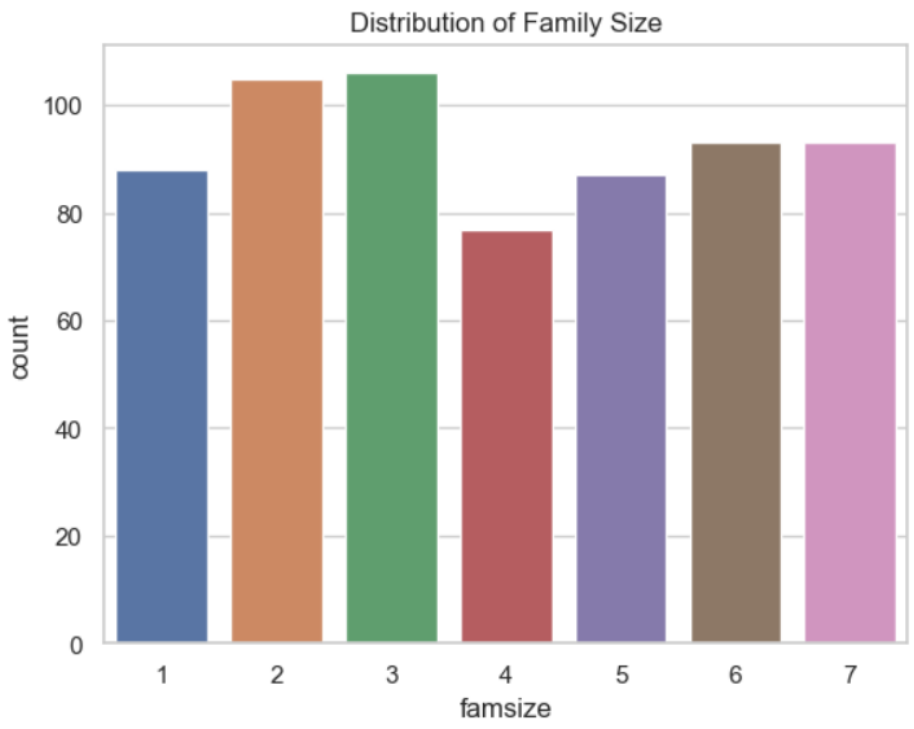


	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	guardian	traveltime	studytime	failures	schoolsup	f
0	GP	F	18	U	1	A	4	4	at_home	teacher	course	mother	2	2	0	yes	
1	GP	F	17	U	2	T	1	1	at_home	other	course	father	1	2	0	no	
2	GP	F	15	U	1	T	1	1	at_home	other	other	mother	1	2	0	yes	
3	GP	F	15	U	1	T	4	2	health	services	home	mother	1	3	0	no	
4	GP	F	16	U	1	T	3	3	other	other	home	father	1	2	0	no	

In summary, this code introduces random values for the 'famsize' column in the original DataFrame based on conditions related to family size ('GT3' or 'LE3'). The random values are generated for each condition, concatenated, and then assigned to the 'famsize' column in the original DataFrame.

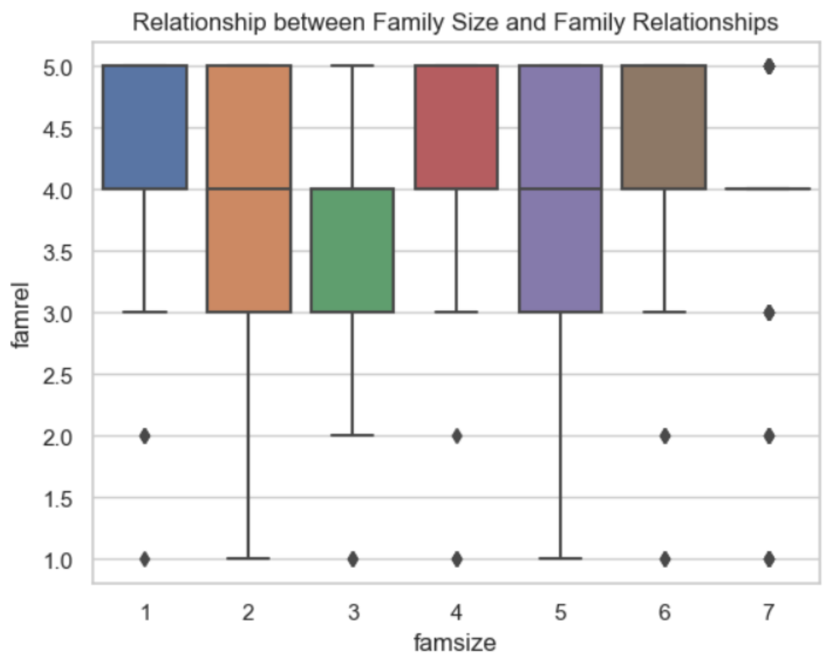
rezult = np.hstack((random_GT3, random_LE3)): Concatenates the two sets of random values horizontally, resulting in an array with a total of 649 values.

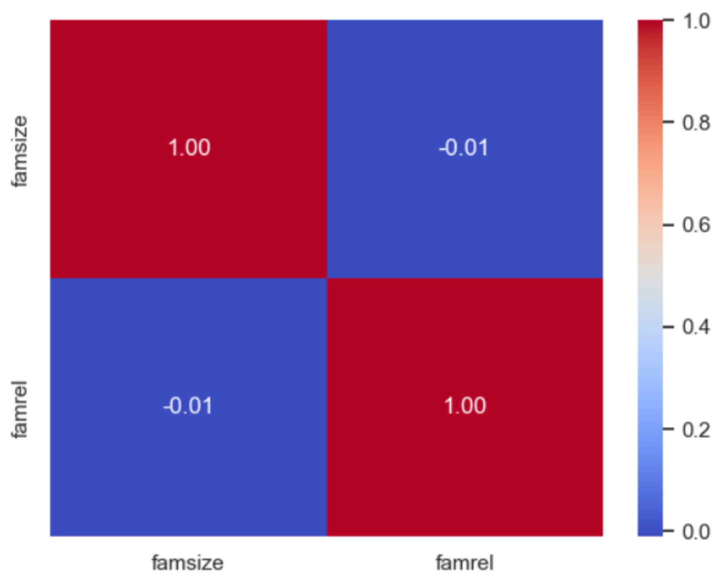
```
sns.countplot(x='famsize', data=df)
plt.title('Distribution of Family Size')
plt.show()
```



This code provides a comprehensive analysis of the relationship between family size and family relationships, including descriptive statistics, visualization through a boxplot, and an examination of the correlation using a heatmap and correlation coefficient.

```
mean_famrel = df.groupby('famsize')['famrel'].mean()
print(f'Mean family relationship score for each family size:\n{mean_famrel}')

sns.boxplot(x='famsize', y='famrel', data=df)
plt.title('Relationship between Family Size and Family Relationships')
plt.show()

correlation_matrix = df[['famsize', 'famrel']].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.show()

correlation = df['famsize'].corr(df['famrel'])
print(f'Correlation between family size and family relationships: {correlation}')
```





In statistical hypothesis testing, the null hypothesis typically assumes that there is no difference between the groups being compared. In this case, the null hypothesis suggests that there is no significant difference in family relationship scores between small and large family sizes. The decision to reject or fail to reject the null hypothesis is based on the calculated p-value, where a smaller p-value indicates stronger evidence against the null hypothesis.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind

famsize_small = df[df['famsize'] == 'LE3']['famrel']
famsize_large = df[df['famsize'] == 'GT3']['famrel']

t_stat, p_value = ttest_ind(famsize_small, famsize_large)

if p_value < 0.05:
    print('Reject the null hypothesis. There is a significant difference in duration between the groups')
else:
    print('Fail to reject the null hypothesis. There is no significant differnce in duratioin between the groups')
```

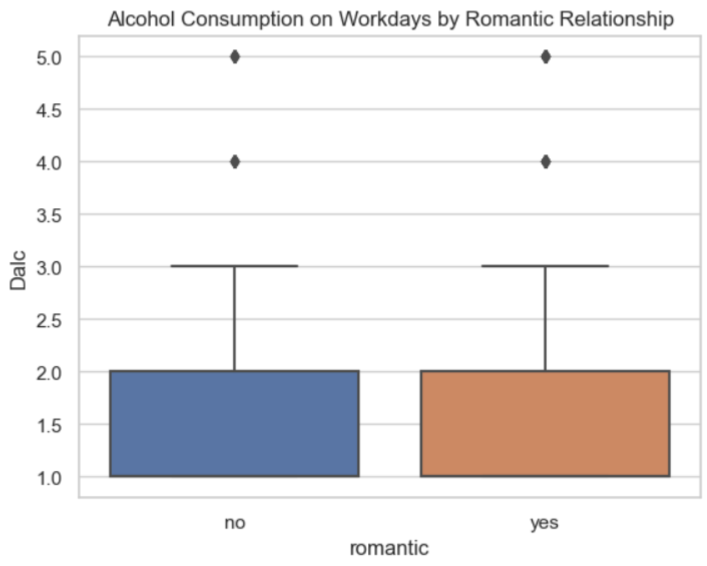
```
Fail to reject the null hypothesis. There is no significant differnce in duratioin between the groups
```

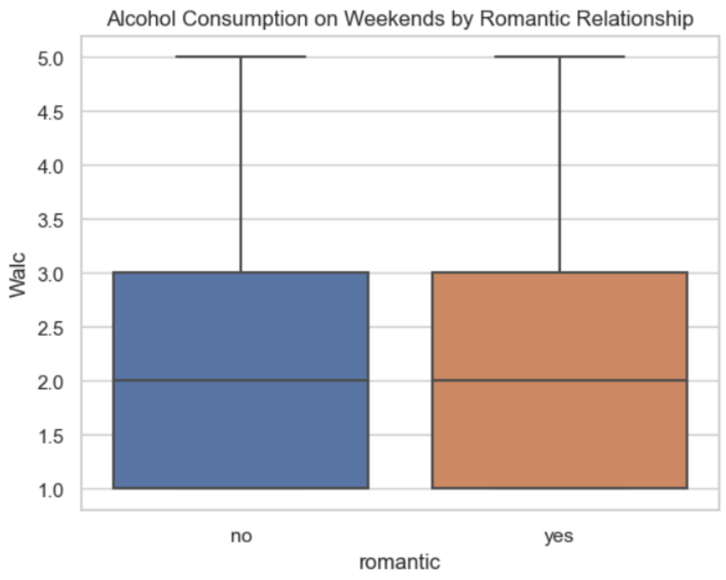
10)Does the presence of romantic relationships (romantic) affect students' alcohol consumption (Dalc and Walc)?

```
import pandas as pd

mean_alcohol = df.groupby('romantic')[['Dalc', 'Walc']].mean()
std_alcohol = df.groupby('romantic')[['Dalc', 'Walc']].std()

print("Mean Alcohol Consumption:")
print(mean_alcohol)

print("\nStandard Deviation of Alcohol Consumption:")
print(std_alcohol)
```

```
Mean Alcohol Consumption:
              Dalc      Walc
romantic
no        1.458537  2.300000
yes       1.577406  2.246862

Standard Deviation of Alcohol Consumption:
              Dalc      Walc
romantic
no        0.844651  1.283809
yes       1.045836  1.287358
```

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(x='romantic', y='Dalc', data=df)
plt.title('Alcohol Consumption on Workdays by Romantic Relationship')
plt.show()

sns.boxplot(x='romantic', y='Walc', data=df)
plt.title('Alcohol Consumption on Weekends by Romantic Relationship')
plt.show()
```





The provided code analyzes the relationship between the presence of romantic relationships ('romantic') and students' alcohol consumption, considering both workday ('Dalc') and weekend ('Walc') alcohol consumption. It calculates the mean and standard deviation of alcohol consumption for students with and without romantic relationships.

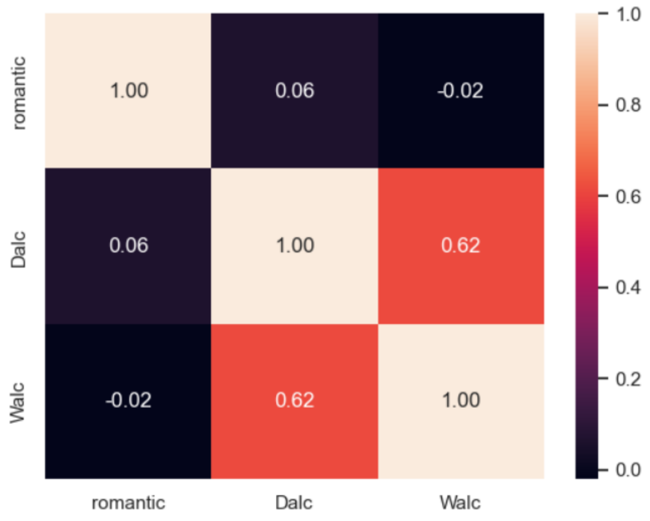
```python
from scipy.stats import ttest_ind

group_with_romantic = df[df['romantic'] == 'yes']
group_without_romantic = df[df['romantic'] == 'no']
group_with_romanticDW = df[df['romantic'] == 'yes']['Dalc'] + df[df['romantic'] == 'yes']['Walc']
group_without_romanticDW = df[df['romantic'] == 'no']['Dalc'] + df[df['romantic'] == 'no']['Walc']

t_stat_Dalc, p_value_Dalc = ttest_ind(group_with_romantic['Dalc'], group_without_romantic['Dalc'])
t_stat_Walc, p_value_Walc = ttest_ind(group_with_romantic['Walc'], group_without_romantic['Walc'])
f_statistic, p_value = stats.f_oneway(group_with_romanticDW, group_without_romanticDW)

if p_value < 0.05:
    print('Reject the null hypothesis. There is a significant difference in duration between the groups')
else:
    print('Fail to reject the null hypothesis. There is no significant differnce in duratioin between the groups')

print(f'T-test results for Workday Alcohol Consumption:\nT-statistic: {t_stat_Dalc}\nP-value: {p_value_Dalc}\n')
print(f'T-test results for Weekend Alcohol Consumption:\nT-statistic: {t_stat_Walc}\nP-value: {p_value_Walc}')
```

```
Fail to reject the null hypothesis. There is no significant differnce in duratioin between the groups
T-test results for Workday Alcohol Consumption:
T-statistic: 1.5811615464450401
P-value: 0.114329809421506

T-test results for Weekend Alcohol Consumption:
T-statistic: -0.5080799827237127
P-value: 0.6115704249662524
```

The code is performing statistical tests to determine if there is a significant difference in alcohol consumption (both workday and weekend) between students with and without romantic relationships. Specifically, it uses t-tests for the workday ('Dalc') and weekend ('Walc') alcohol consumption and an analysis of variance (ANOVA) test for the combined duration of alcohol consumption.

```python
df['romantic'] = df['romantic'].replace('yes', 1)
df['romantic'] = df['romantic'].replace('no', 0)
correlation_Dalc = df['romantic'].corr(df['Dalc'])
correlation_Walc = df['romantic'].corr(df['Walc'])

correlation_matrix = df[['romantic', 'Dalc', 'Walc']].corr()
sns.heatmap(correlation_matrix, annot=True, fmt='.2f')

print(f'Correlation between romantic relationships and Workday Alcohol Consumption: {correlation_Dalc}')
print(f'Correlation between romantic relationships and Weekend Alcohol Consumption: {correlation_Walc}')
```

```
Correlation between romantic relationships and Workday Alcohol Consumption: 0.06204212181914021
Correlation between romantic relationships and Weekend Alcohol Consumption: -0.019970701609854542
```



This analysis helps to understand the linear relationship between romantic relationships and alcohol consumption on workdays and weekends. The heatmap provides a visual representation of the correlation matrix, and the printed correlation values quantify the strength and direction of these relationships.

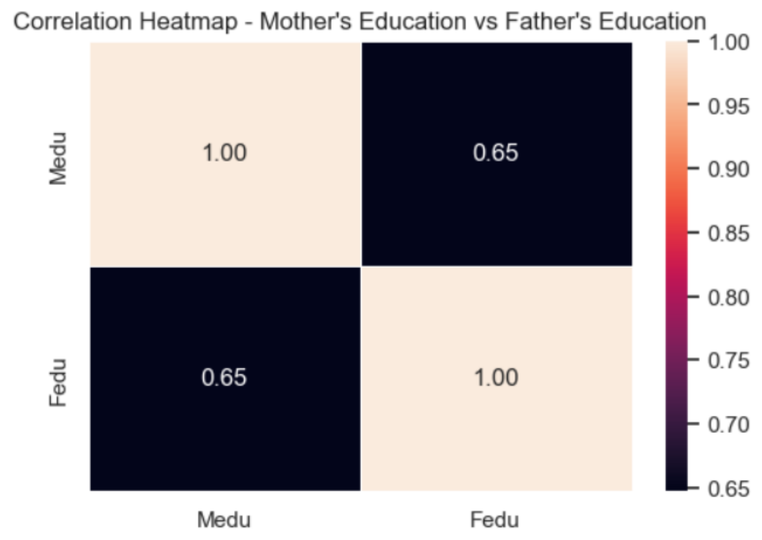
11)How does the mother's education level (Medu) correlate with the father's education level (Fedu)?

```python
correlation = df['Medu'].corr(df['Fedu'])
print(f"Correlation between mother's education (Medu) and father's education (Fedu): {correlation}")

show_corr = df[['Medu', 'Fedu']]
correlation_matrix = show_corr.corr()

plt.figure(figsize=(6, 4))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", linewidths=.5)
plt.title("Correlation Heatmap - Mother's Education vs Father's Education")
plt.show()
```

```
Correlation between mother's education (Medu) and father's education (Fedu): 0.6474766091364939
```

The provided code analyzes the correlation between mother's education level ('Medu') and father's education level ('Fedu') in the DataFrame

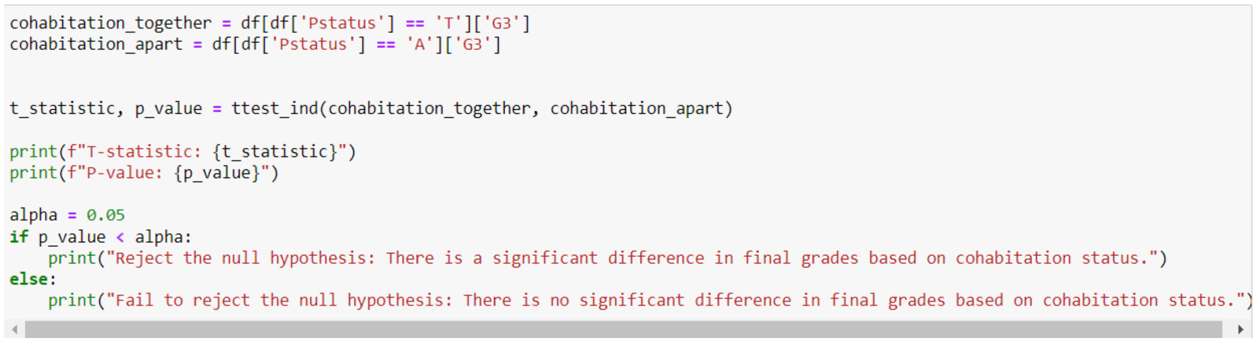
```
Correlation between mother's education (Medu) and father's education (Fedu): 0.6474766091364939
```



Explanation:

The printed correlation coefficient provides a numeric measure of how strongly mother's education level and father's education level are related. The heatmap visually represents the correlation matrix, making it easier to observe patterns and relationships between the two variables.

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



```python
cohabitation_together = df[df['Pstatus'] == 'T']['G3']
cohabitation_apart = df[df['Pstatus'] == 'A']['G3']


t_statistic, p_value = ttest_ind(cohabitation_together, cohabitation_apart)

print(f"T-statistic: {t_statistic}")
print(f"P-value: {p_value}")

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in final grades based on cohabitation status.")
else:
    print("Fail to reject the null hypothesis: There is no significant difference in final grades based on cohabitation status.")
```

```
T-statistic: -0.019176989794419484
P-value: 0.9847058259502668
Fail to reject the null hypothesis: There is no significant difference in final grades based on cohabitation status.
```

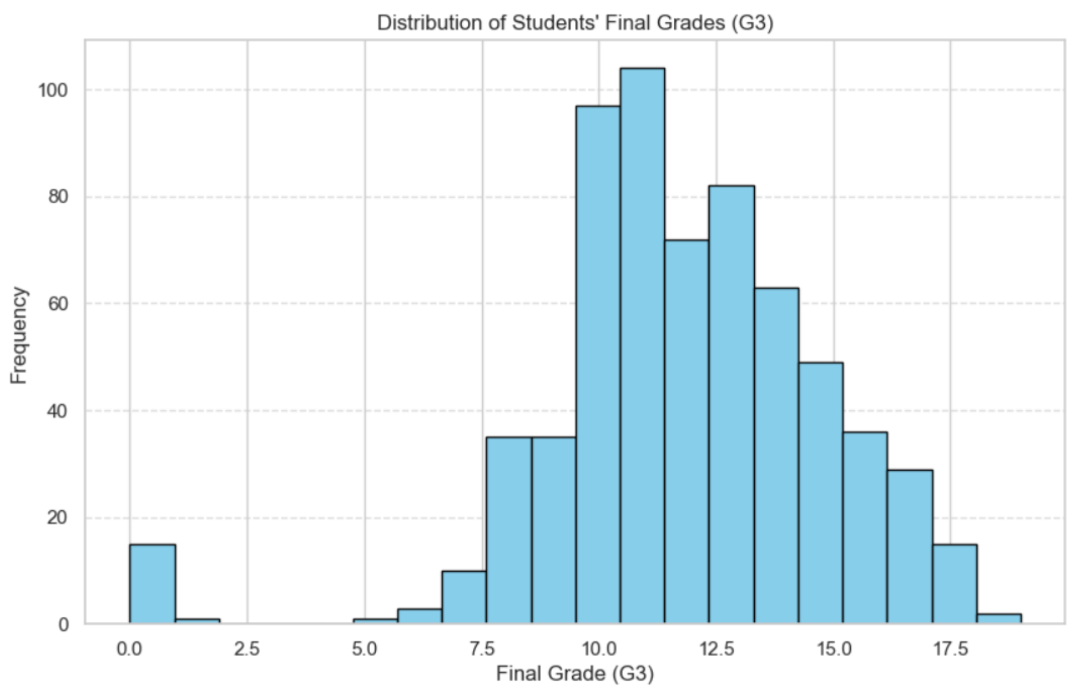
Explanation:

The provided code conducts an independent two-sample t-test to examine whether there is a significant difference in final grades ('G3') based on cohabitation status ('Pstatus'). The null hypothesis assumes that there is no significant difference in final grades based on cohabitation status. The code helps to determine whether the observed difference in final grades is likely due to random chance or if it is statistically significant. If the p-value is below the significance level, the null hypothesis is rejected, suggesting a significant difference in final grades based on cohabitation status. Otherwise, the null hypothesis is not rejected.

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

```python
plt.figure(figsize=(10, 6))
plt.hist(df['G3'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Students\' Final Grades (G3)')
plt.xlabel('Final Grade (G3)')
plt.ylabel('Frequency')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

This code generates and displays a histogram showing the distribution of students' final grades.

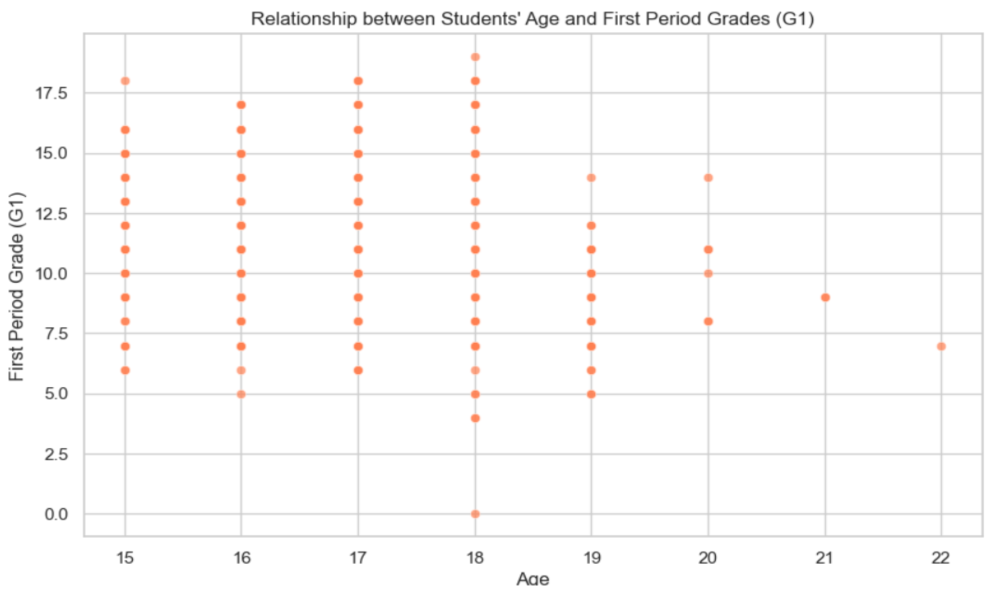


Explanation:

The resulting histogram provides a visual representation of the distribution of final grades among students. The x-axis represents the final grades, and the y-axis represents the frequency of each grade.

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

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='age', y='G1', data=df, alpha=0.7, color='coral')
plt.title('Relationship between Students\' Age and First Period Grades (G1)')
plt.xlabel('Age')
plt.ylabel('First Period Grade (G1)')
plt.show()
```

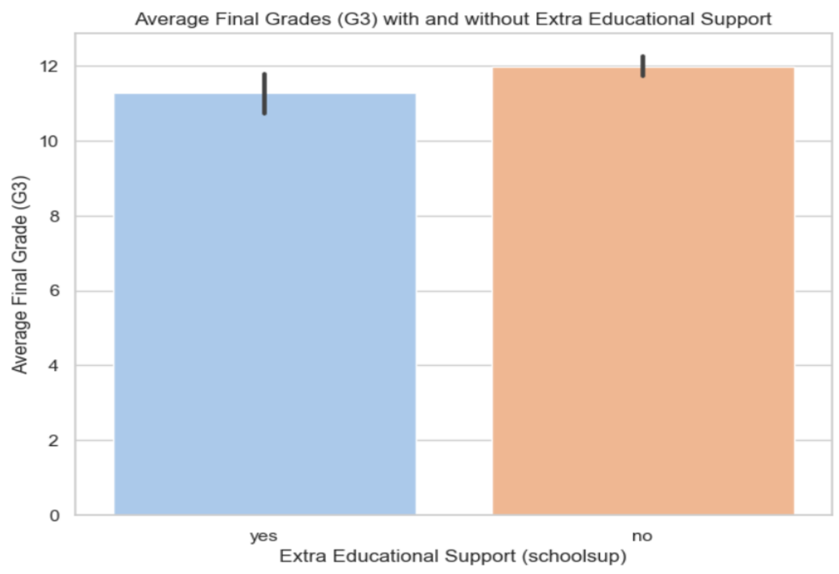


Explanation:

This code generates and displays a scatter plot illustrating the relationship between students' age and their first period grades (G1). Each point in the plot represents a student, with the x-coordinate corresponding to their age and the y-coordinate corresponding to their first period grade. The transparency (alpha) is set to 0.7 to better visualize overlapping points.

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

```
plt.figure(figsize=(8, 6))
sns.barplot(x='schoolsup', y='G3', data=df, palette='pastel')
plt.title('Average Final Grades (G3) with and without Extra Educational Support')
plt.xlabel('Extra Educational Support (schoolsup)')
plt.ylabel('Average Final Grade (G3)')

plt.show()
```



Explanation:

This code generates and displays a bar plot showing the average final grades (G3) for students with and without extra educational support (schoolsup). The 'pastel' color palette is used to differentiate between the two categories. Each bar represents the average final grade for the corresponding group.

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?

```
df = df.rename(columns={'G3': 'G3_Port'})
df2 = df2.rename(columns={'G3': 'G3_Math'})
```

```
import pandas as pd

concat_df = pd.concat([df, df2], axis=1)
final_grades_comparison = concat_df[['G3_Port', 'G3_Math']]

final_grades_comparison.columns = ['G3_Portuguese', 'G3_Math']

print(final_grades_comparison.head())
```

```
   G3_Portuguese  G3_Math
0             11      6.0
1             11      6.0
2             12     10.0
3             14     15.0
4             13     10.0
```

```
grades_with_paid = df['G3_Port']
grades_without_paid = df2['G3_Math']

t_statistic, p_value = ttest_ind(grades_with_paid, grades_without_paid)

print(f"T-statistic: {t_statistic}")
print(f"P-value: {p_value}")

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in final grades.")
else:
    print("Fail to reject the null hypothesis: There is no significant difference in final grades.")
```

```
T-statistic: 6.150411461705681
P-value: 1.0998219243723993e-09
Reject the null hypothesis: There is a significant difference in final grades.
```

```
concat_df.agg({'G3_Port': ['mean', 'median', 'std', 'min', 'max'], 'G3_Math': ['mean', 'median', 'std', 'min', 'max']})
```

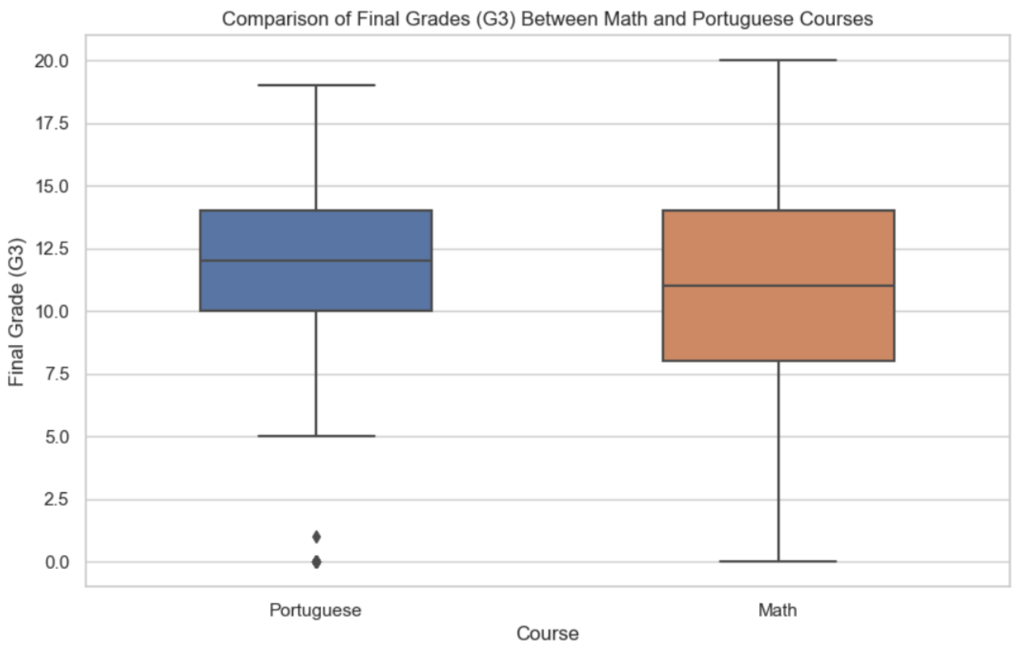
	G3_Port	G3_Math
mean	11.906009	10.415190
median	12.000000	11.000000
std	3.230656	4.581443
min	0.000000	0.000000
max	19.000000	20.000000

Explanation:

This code performs an independent t-test to compare the final grades (G3_Port for one group and G3_Math for another group). The decision to reject or fail to reject the null hypothesis is based on the calculated p-value and the chosen significance level (alpha).

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

```
final_grades_comparison.columns = ['G3_Portuguese', 'G3_Math']

plt.figure(figsize=(10, 6))
sns.boxplot(data=final_grades_comparison, width=0.5)
plt.title('Comparison of Final Grades (G3) Between Math and Portuguese Courses')
plt.xlabel('Course')
plt.ylabel('Final Grade (G3)')
plt.xticks([0, 1], ['Portuguese', 'Math'])
plt.show()
```



Explanation:

This visualization allows for a direct comparison of the distribution of final grades between the two courses. The boxes represent the interquartile range (IQR), while the whiskers extend to show the range of the data. Outliers may also be visible, providing insights into the spread and central tendency of final grades in each course.

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.

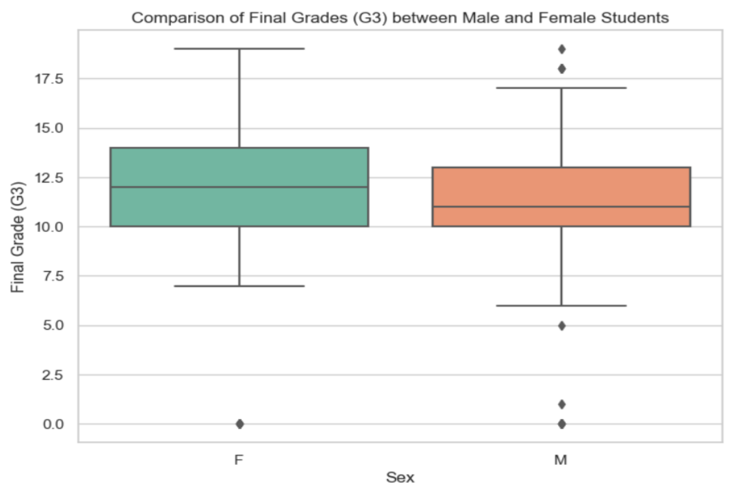
```
df = df.rename(columns={'G3_Port':'G3'})
df2 = df2.rename(columns={'G3_Math':'G3'})
```

```
male_grades = df[df['sex'] == 'M']['G3']
female_grades = df[df['sex'] == 'F']['G3']

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

print(f'Two-Sample T-Test Results:\n'
      f'T-statistic: {t_statistic}\n'
      f'P-value: {p_value}')


if p_value < 0.05:
    print('There is a significant difference in the average final grades between male and female students.')
else:
    print('There is no significant difference in the average final grades between male and female students.')
```

```
Two-Sample T-Test Results:
T-statistic: -3.310937693029702
P-value: 0.000981528706137396
There is a significant difference in the average final grades between male and female students.
```

Explanation:
This code provides statistical insights into whether there is a significant difference in the average final grades between male and female students based on the results of a two-sample t-test.

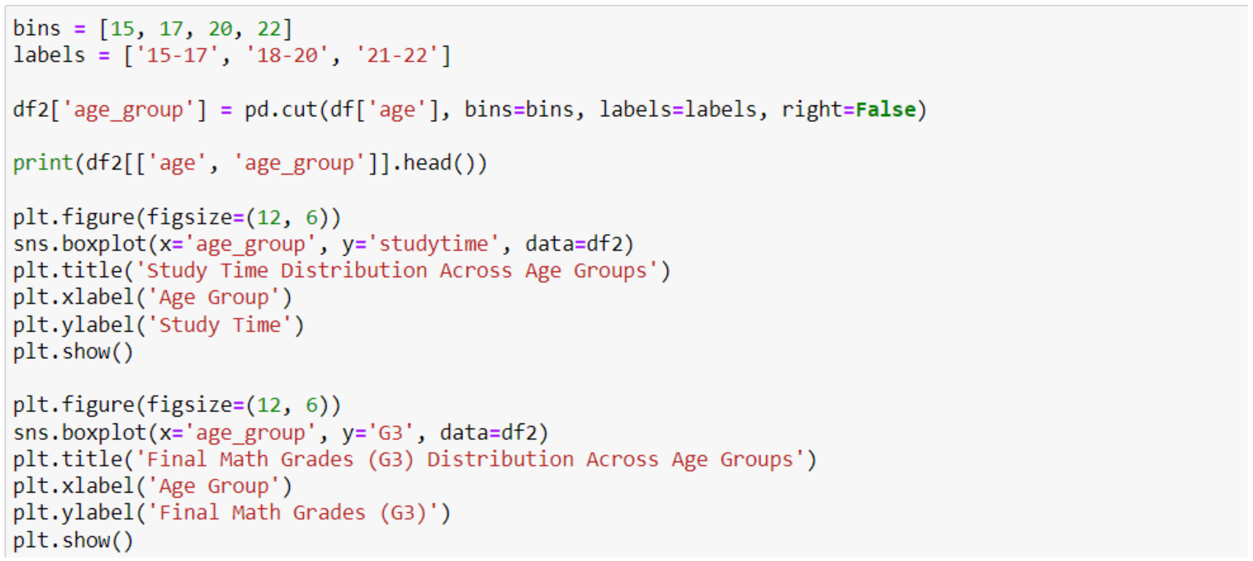
```
plt.figure(figsize=(8, 6))
sns.boxplot(x='sex', y='G3', data=df, palette='Set2')
plt.title('Comparison of Final Grades (G3) between Male and Female Students')
plt.xlabel('Sex')
plt.ylabel('Final Grade (G3)')
plt.show()
```



Explanation:

This visualization allows for a direct comparison of the distribution of final grades between male and female students. The box plot provides insights into the median, interquartile range (IQR), and potential outliers for each gender.

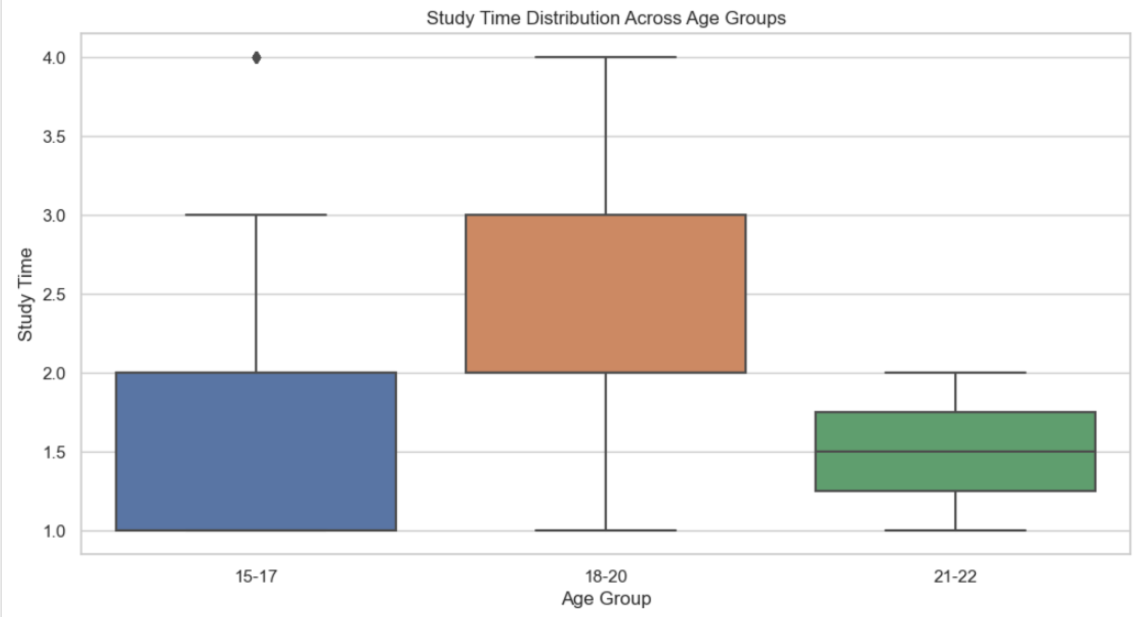
19) Can you create a new variable that categorizes students into age groups(e.g., 15-17, 18-20, 21-22)? How does this grouping affect the analysis of other variables, such as study time or final math grades (G3)?

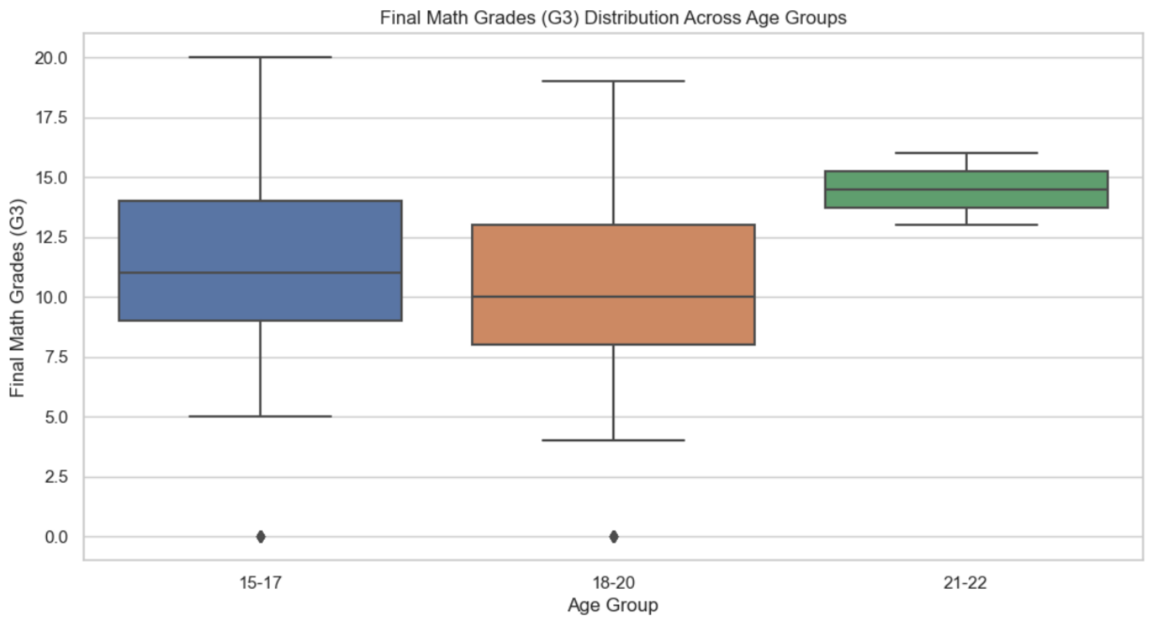


```python
bins = [15, 17, 20, 22]
labels = ['15-17', '18-20', '21-22']

df2['age_group'] = pd.cut(df['age'], bins=bins, labels=labels, right=False)

print(df2[['age', 'age_group']].head())

plt.figure(figsize=(12, 6))
sns.boxplot(x='age_group', y='studytime', data=df2)
plt.title('Study Time Distribution Across Age Groups')
plt.xlabel('Age Group')
plt.ylabel('Study Time')
plt.show()

plt.figure(figsize=(12, 6))
sns.boxplot(x='age_group', y='G3', data=df2)
plt.title('Final Math Grades (G3) Distribution Across Age Groups')
plt.xlabel('Age Group')
plt.ylabel('Final Math Grades (G3)')
plt.show()
```

Both boxplots provide insights into the distribution of study time and final math grades among different age groups. The x-axis represents the age groups, while the y-axis represents the respective variables (study time or final math grades). The boxplots include information about medians, quartiles, and potential outliers within each age group.

```
   age age_group
0   18     18-20
1   17     18-20
2   15     15-17
3   15     15-17
4   16     15-17
```



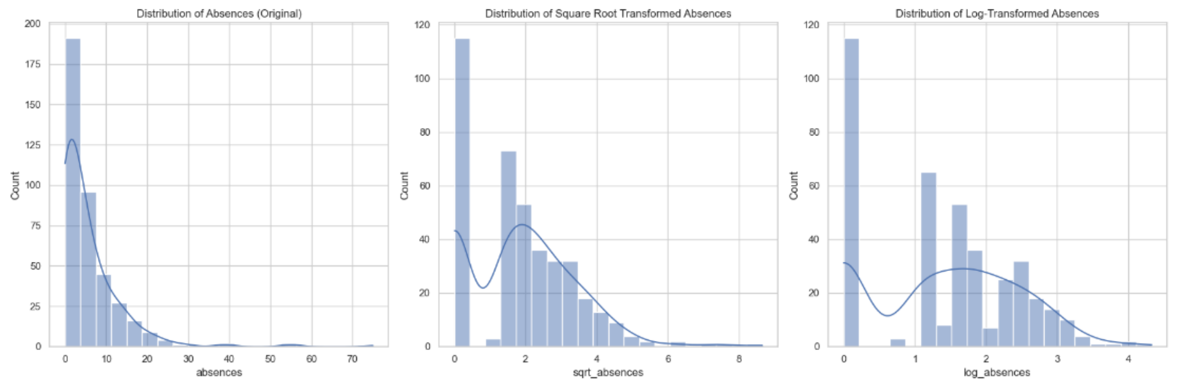


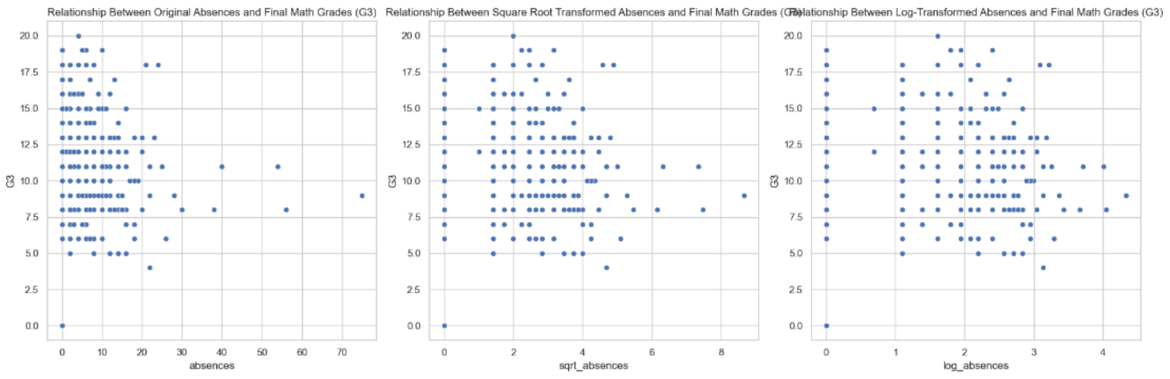
Explanation:

By categorizing students into age groups, you can observe how study time and final math grades vary across these groups. The box plots provide insights into the central tendency, spread, and potential outliers in these variables for each age group.

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)?

```
df2['sqrt_absences'] = np.sqrt(df2['absences'])
df2['log_absences'] = np.log1p(df2['absences'])

plt.figure(figsize=(18, 6))
plt.subplot(1, 3, 1)
sns.histplot(df2['absences'], bins=20, kde=True)
plt.title('Distribution of Absences (Original)')

plt.subplot(1, 3, 2)
sns.histplot(df2['sqrt_absences'], bins=20, kde=True)
plt.title('Distribution of Square Root Transformed Absences')

plt.subplot(1, 3, 3)
sns.histplot(df2['log_absences'], bins=20, kde=True)
plt.title('Distribution of Log-Transformed Absences')

plt.tight_layout()
plt.show()

plt.figure(figsize=(18, 6))
plt.subplot(1, 3, 1)
sns.scatterplot(x='absences', y='G3', data=df2)
plt.title('Relationship Between Original Absences and Final Math Grades (G3)')

plt.subplot(1, 3, 2)
sns.scatterplot(x='sqrt_absences', y='G3', data=df2)
plt.title('Relationship Between Square Root Transformed Absences and Final Math Grades (G3)')

plt.subplot(1, 3, 3)
sns.scatterplot(x='log_absences', y='G3', data=df2)
plt.title('Relationship Between Log-Transformed Absences and Final Math Grades (G3)')

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





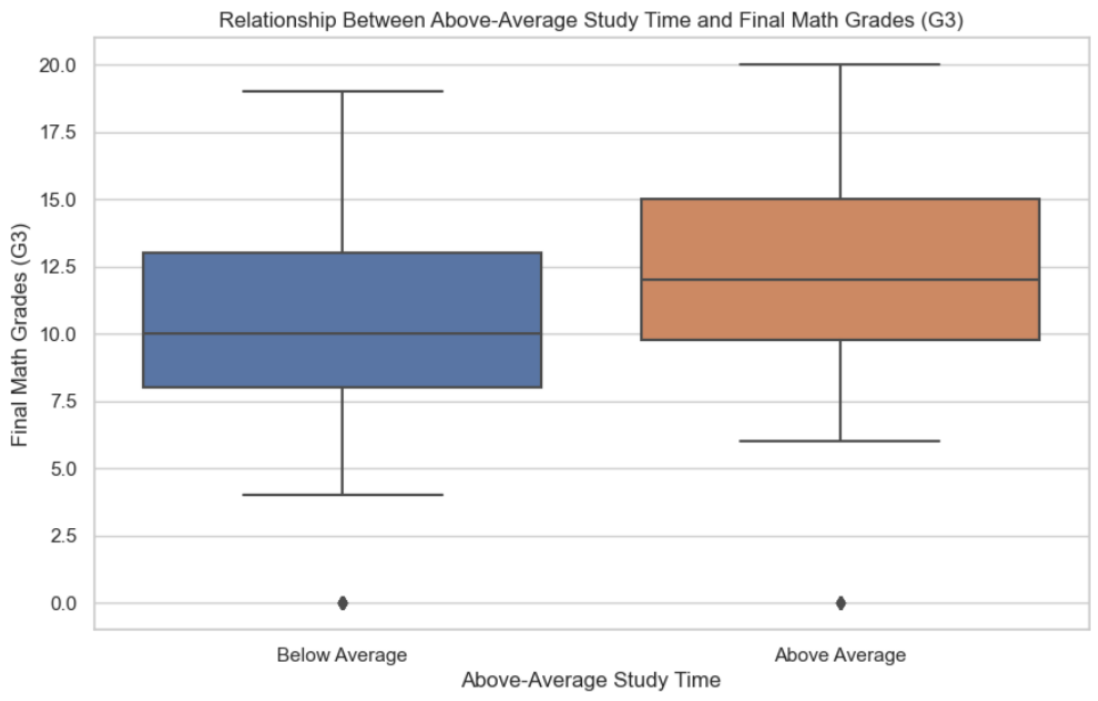
Explanation:

The code applies mathematical transformations (square root and logarithm) to the number of school absences ('absences') and visualizes the impact of these transformations on the distribution of absences and their relationship with final math grades ('G3').

These visualizations illustrate the impact of different mathematical transformations on the distribution of absences and their relationship with final math grades. Square root and logarithmic transformations are commonly employed to stabilize variance and make data more suitable for analysis, particularly when dealing with skewed distributions.

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)?

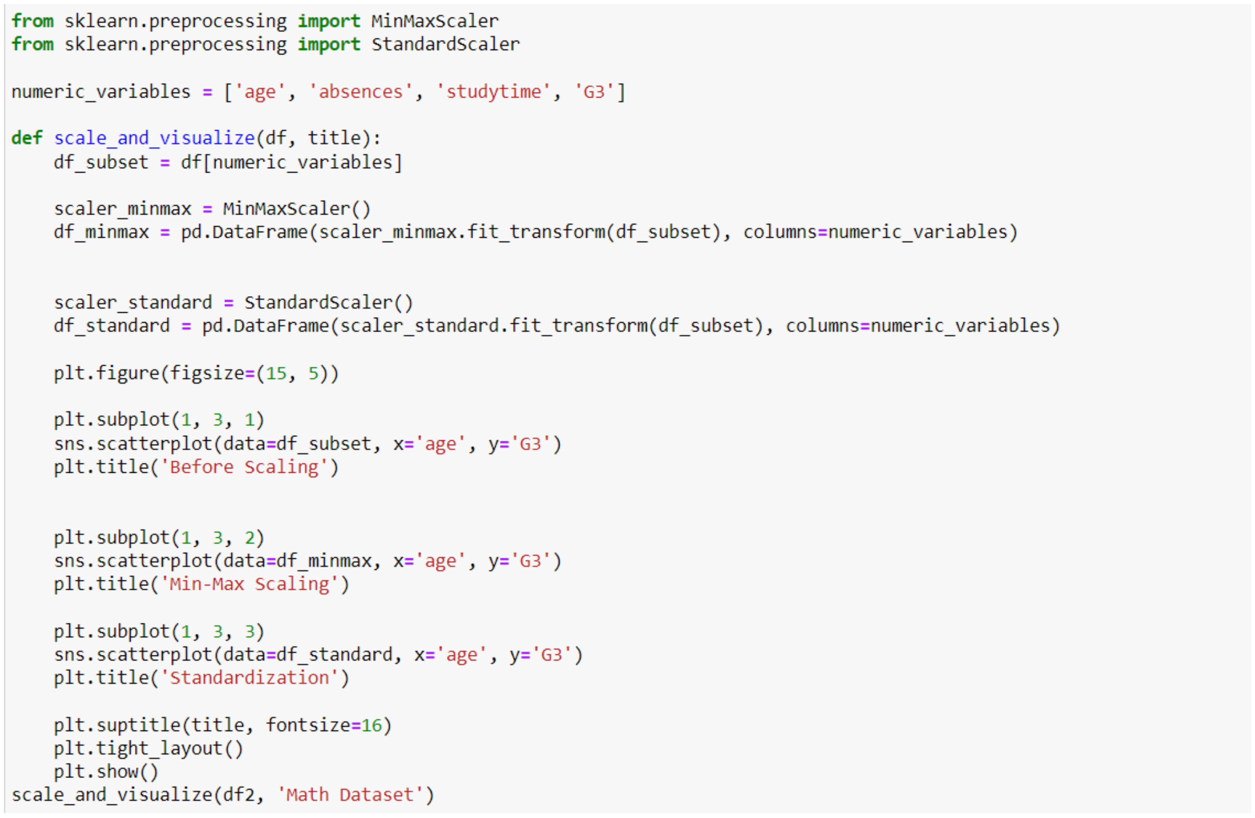
```
avarage_study_time = df2['studytime'].mean()

df2['above_avg_study_time'] = (df2['studytime'] > avarage_study_time).astype(int)

plt.figure(figsize=(10, 6))
sns.boxplot(x='above_avg_study_time', y='G3', data=df2)
plt.title('Relationship Between Above-Average Study Time and Final Math Grades (G3)')
plt.xlabel('Above-Average Study Time')
plt.ylabel('Final Math Grades (G3)')
plt.xticks([0, 1], ['Below Average', 'Above Average'])
plt.show()
```



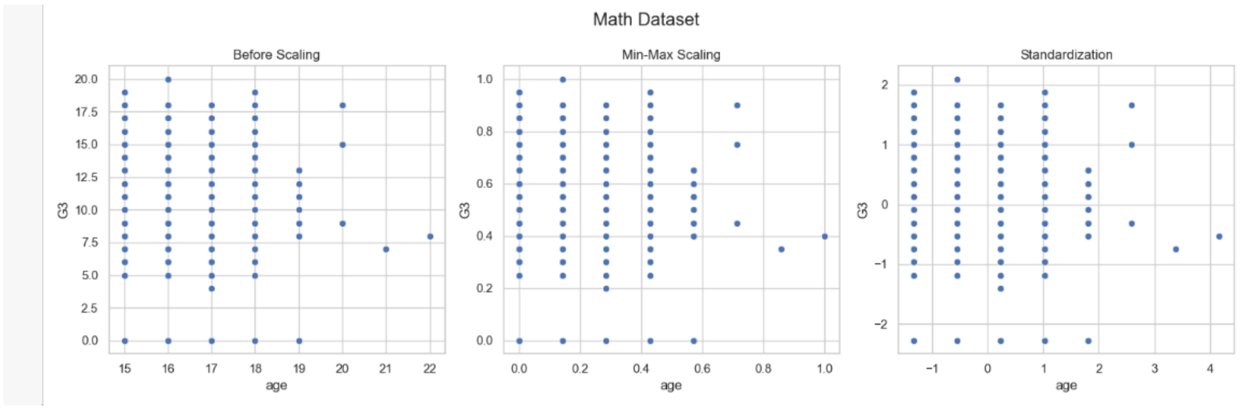
Explanation:

This visualization shows the relationship between above-average study time and final math grades. The x-axis represents the study time category (below average or above average), while the y-axis represents the final math grades. The boxplot provides insights into the distribution of grades for students with different study time levels, helping to understand if there's a correlation between above-average study time and higher final math grades.

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)?



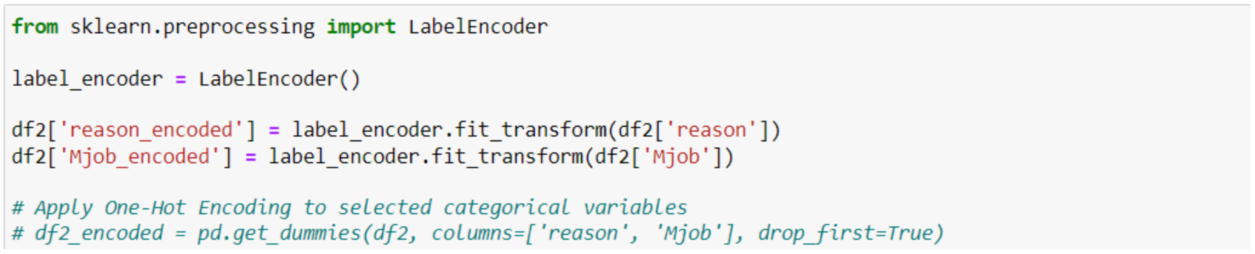
```python
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

numeric_variables = ['age', 'absences', 'studytime', 'G3']

def scale_and_visualize(df, title):
    df_subset = df[numeric_variables]

    scaler_minmax = MinMaxScaler()
    df_minmax = pd.DataFrame(scaler_minmax.fit_transform(df_subset), columns=numeric_variables)


    scaler_standard = StandardScaler()
    df_standard = pd.DataFrame(scaler_standard.fit_transform(df_subset), columns=numeric_variables)

    plt.figure(figsize=(15, 5))

    plt.subplot(1, 3, 1)
    sns.scatterplot(data=df_subset, x='age', y='G3')
    plt.title('Before Scaling')


    plt.subplot(1, 3, 2)
    sns.scatterplot(data=df_minmax, x='age', y='G3')
    plt.title('Min-Max Scaling')

    plt.subplot(1, 3, 3)
    sns.scatterplot(data=df_standard, x='age', y='G3')
    plt.title('Standardization')

    plt.suptitle(title, fontsize=16)
    plt.tight_layout()
    plt.show()
scale_and_visualize(df2, 'Math Dataset')
```



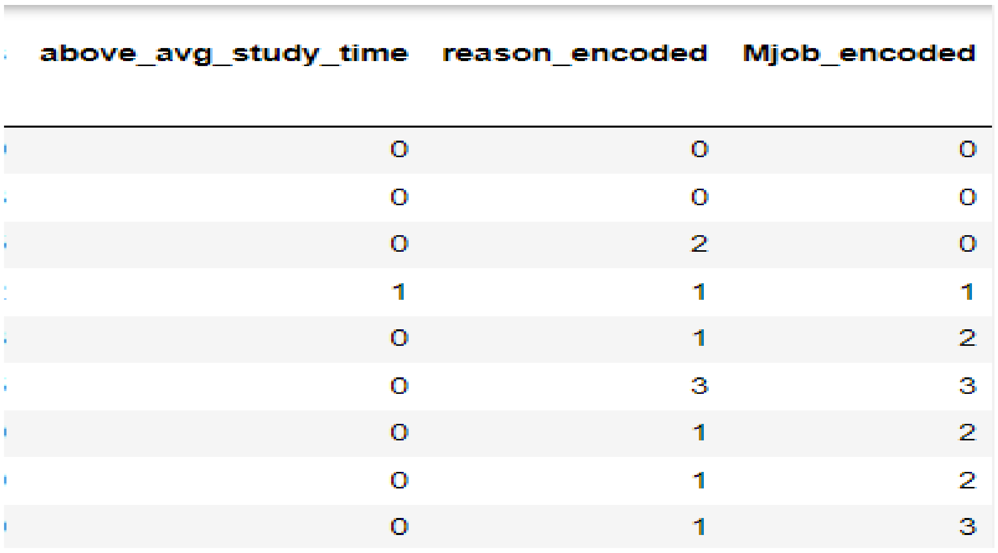
Explanation:

> This visualization helps observe how feature scaling affects the relationships between age and math grades. Min-Max scaling and standardization are common techniques to standardize the range of independent variables, making them comparable and potentially improving the performance of certain machine learning algorithms.

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?



```python
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

df2['reason_encoded'] = label_encoder.fit_transform(df2['reason'])
df2['Mjob_encoded'] = label_encoder.fit_transform(df2['Mjob'])

# Apply One-Hot Encoding to selected categorical variables
# df2_encoded = pd.get_dummies(df2, columns=['reason', 'Mjob'], drop_first=True)
```



above_avg_study_time	reason_encoded	Mjob_encoded
0	0	0
0	0	0
0	2	0
1	1	1
0	1	2
0	3	3
0	1	2
0	1	2
0	1	3

Explanation:

Label encoding is a technique to convert categorical data into numerical format, which is often required for machine learning algorithms that work with numerical inputs. The `fit_transform` method fits the encoder to the unique values in the specified column and transforms those values into numerical labels.

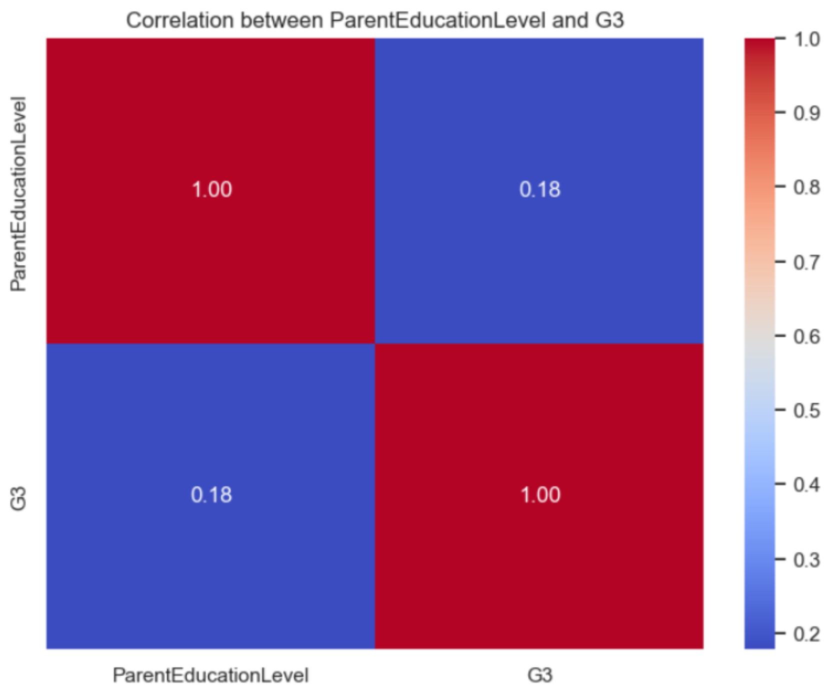
Label Encoding:

Label encoding is applied to transform categorical variables ('reason' and 'Mjob') into numerical labels.
This is useful when working with machine learning algorithms that require numerical input.

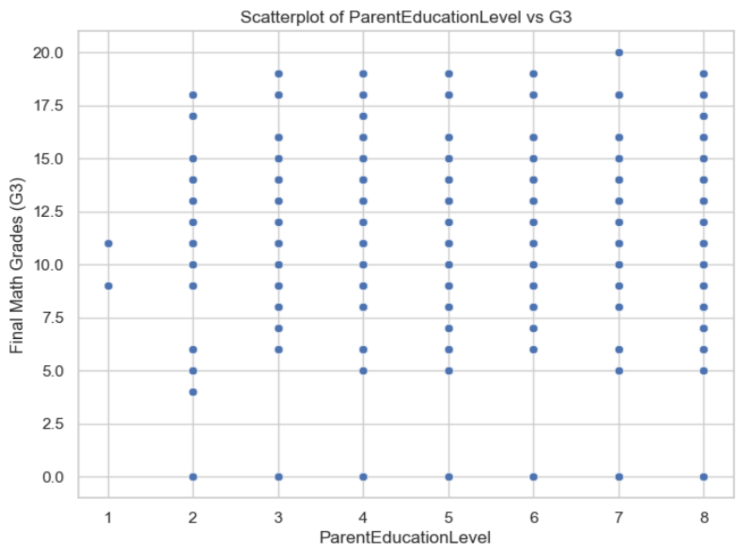
24) Combine multiple variables (e.g., mother's education and father's education) to create a composite metric representing the overall parental education level. How does this new metric correlate with students' final math grades (G3)?

```
df2['ParentEducationLevel'] = df['Medu'] + df['Fedu']

correlation_matrix = df2[['ParentEducationLevel', 'G3']].corr()

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", annot_kws={"size": 12})
plt.title('Correlation between ParentEducationLevel and G3')
plt.show()
```

Explanation:
The heatmap visualization illustrates the relationship between the combined parental education level metric and students' final math grades. A positive correlation suggests that an increase in parental education level corresponds to higher final math grades, while a negative correlation implies the opposite pattern.



```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
sns.scatterplot(x='ParentEducationLevel', y='G3', data=df2)
plt.title('Scatterplot of ParentEducationLevel vs G3')
plt.xlabel('ParentEducationLevel')
plt.ylabel('Final Math Grades (G3)')
plt.show()
```



Explanation:

The code generates a scatter plot to display the relationship between the combined parental education level metric ('ParentEducationLevel') and students' final math grades ('G3'). The x-axis represents the parental education level, and the y-axis represents the final math grades. This visualization allows for an examination of patterns or trends in the data distribution.

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?

```
import pandas as pd

urban = df[df['address'] == 'U']['studytime']
rural = df[df['address'] =='R']['studytime']

urban_avg_study_time = df2[df2['address'] == 'U']['studytime'].mean()

rural_avg_study_time = df2[df2['address'] == 'R']['studytime'].mean()


t_stat, p_value = stats.ttest_ind(urban, rural)
alpha = 0.05

if p_value < alpha:
    print("There is a significant difference in study time between urban and rural students.")
else:
    print("There is no significant difference in study time between urban and rural students.")

print(f"Average Study Time for Urban Students: {urban_avg_study_time:.2f} hours per week")
print(f"Average Study Time for Rural Students: {rural_avg_study_time:.2f} hours per week")
```

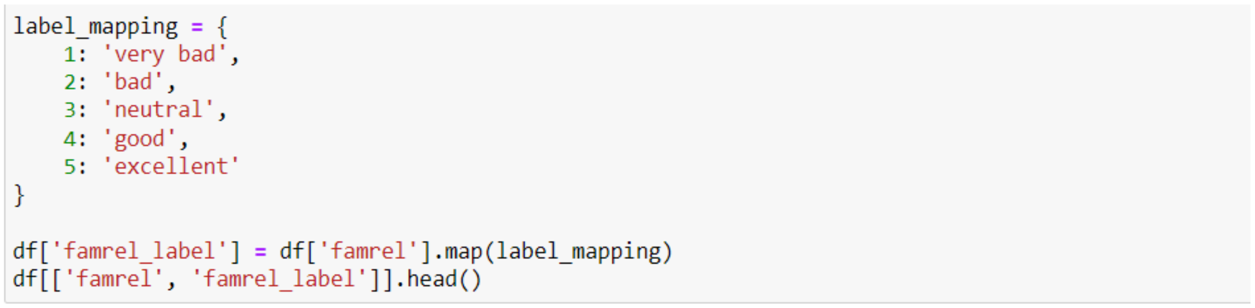
Prints the results of the hypothesis test and the average study time for both urban and rural students.

```
There is no significant difference in study time between urban and rural students.
Average Study Time for Urban Students: 2.03 hours per week
Average Study Time for Rural Students: 2.07 hours per week
```

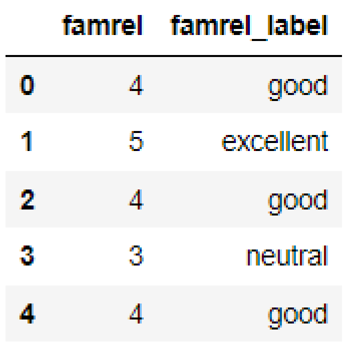
Explanation:

The code calculates the average weekly study time for students in urban ('U') and rural ('R') areas. It then conducts a statistical test to check if there are significant differences in study time between these two groups. The results are printed, including the average study time for both urban and rural students.

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?



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

df['famrel_label'] = df['famrel'].map(label_mapping)
df[['famrel', 'famrel_label']].head()
```



	famrel	famrel_label
0	4	good
1	5	excellent
2	4	good
3	3	neutral
4	4	good

Explanation:

The code assigns meaningful labels to the numerical values of the ordinal variable 'famrel' (quality of family relationships). This transformation enhances the interpretability of the data by replacing numeric codes with descriptive labels.

This transformation improves the interpretability of the 'famrel' variable by replacing numeric codes with descriptive labels. Instead of dealing with abstract numeric values, analysts can now easily understand the meaning of each level, making the data more accessible and facilitating a better understanding of the family relationship quality.

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?

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

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

print('Age Range within Different Schools:')
print(age_range_by_school)
```

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

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

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

print('\nPercentage 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
```

Explanation:

The code defines and applies custom aggregation functions to summarize the data. The first function calculates the age range within different schools, while the second function determines the percentage of students with internet access ('internet' = 'yes') by gender.

These custom aggregations offer insights into the variability of ages within different schools and the proportion of students with internet access based on gender. This type of analysis can help identify patterns or trends in the data that may not be evident from individual observations.

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

```
median_absences_by_schoolsup = df.groupby('schoolsup')['absences'].median()

print("Median Number of Absences for Students:")
print(median_absences_by_schoolsup)
```

The output helps understand the central tendency of the distribution of absences within these two groups, providing information on the typical or middle value of absences.

```
Median Number of Absences for Students:
schoolsup
no     2.0
yes    2.0
Name: absences, dtype: float64
```

Explanation:

> The code calculates the median number of school absences ('absences') for students with and without extra educational support ('schoolsup'). The results provide insights into the central tendency of the number of absences for these two groups, aiding in understanding typical absence patterns based on the presence or absence of extra educational support.

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

```
percentage_by_fedu = df.groupby('Fedu')['higher'].value_counts(normalize=True).loc[:, 'yes'] * 100

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

This information helps understand the relationship between the aspiration for higher education and the father's education level, providing insights into how educational aspirations may vary across different levels of paternal education.

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

Explanation:

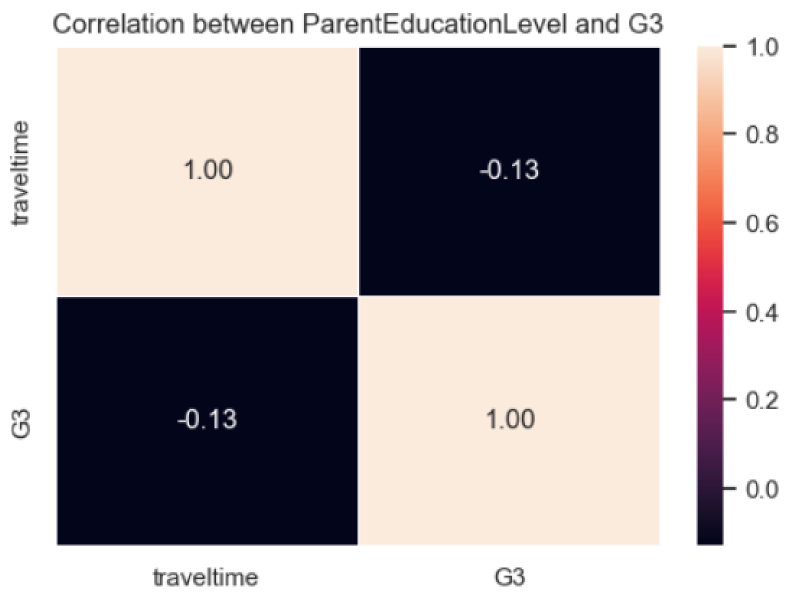
This analysis provides insights into the relationship between students' aspirations for higher education and their father's education level. The results help understand if there is a correlation between a student's desire for higher education and their father's educational background.

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

```
correlation_matrix = df[['traveltime', 'G3']].corr()
plt.figure(figsize=(6, 4))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", linewidths=.5)
plt.title('Correlation between ParentEducationLevel and G3')
plt.show()
```

The code aims to visually represent the correlation between home-to-school travel time ('traveltime') and final grades ('G3') using a heatmap.

This visualization helps understand if there is any discernible correlation between the time it takes for students to travel from home to school and their final grades.



Explanation:

This analysis visually represents the correlation between travel time and final grades, providing insights into whether there is a significant relationship between these two variables. The heatmap helps in quickly identifying the strength and direction of the correlation.

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

```
weighted_average = np.average(df['G3'], weights=df['studytime'])

print("Weighted Average of G3 using study time as weights:", weighted_average)
```

```
Weighted Average of G3 using study time as weights: 12.25219473264166
```

Explanation:

The code calculates a weighted average of the final grades ('G3') based on the weights provided by the 'studytime' column.
This calculation provides insights into how final grades are distributed when considering the study time as a weighting factor.

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

```
highest_walc_student = df.loc[df['Walc'].idxmax()]

print("Student with the highest weekend alcohol consumption (Walc):")
print(highest_walc_student)
```

The code aims to identify and display information about the student who has the highest weekend alcohol consumption ('Walc').

The printed output provides detailed information about this specific student, allowing for further examination of factors associated with high weekend alcohol consumption.

```
Student with the highest weekend alcohol consumption (Walc):
school                 GP
sex                     M
age                    16
address                 U
famsize                 2
Pstatus                 T
Medu                    4
Fedu                    4
Mjob              teacher
Fjob              teacher
reason               home
guardian           mother
traveltime              1
studytime               2
failures                0
schoolsup              no
famsup                yes
paid                  yes
activities            yes
nursery               yes
higher                yes
internet              yes
romantic                1
famrel                  4
freetime                4
goout                   5
Dalc                    5
Walc                    5
health                  5
absences                4
G1                     12
G2                     11
G3                     12
famrel_label         good
Name: 29, dtype: object
```

Explanation:

This code uses the idxmax function to find the index of the student with the maximum 'Walc' value and then uses loc to retrieve and print the corresponding student's information. The result provides details about the student with the highest weekend alcohol consumption.

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

```
df['guardian'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 649 entries, 0 to 648
Series name: guardian
Non-Null Count  Dtype
--------------  -----
649 non-null    object
dtypes: object(1)
memory usage: 5.2+ KB
```

```
df['guardian'].isnull().sum()
```

```
0
```

```
df['guardian'] = df['guardian'].fillna('unknown')
```

Explanation:

This code ensures that any missing values in the 'guardian' column are replaced with the string 'unknown'. This is a common practice for handling missing categorical data, providing a placeholder or indication that the specific information is not available. After running this code, the 'guardian' column will have missing values filled with 'unknown'.

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

```
df['romantic'].info()
df['romantic'].isnull().sum()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 649 entries, 0 to 648
Series name: romantic
Non-Null Count  Dtype
--------------  -----
649 non-null    int64
dtypes: int64(1)
memory usage: 5.2 KB

0
```

```
most_common_value = df['romantic'].mode().iloc[0]
df['romantic'] = df['romantic'].fillna(most_common_value)
```

Explanation:

this code ensures that missing values in the 'romantic' column are replaced with the most common value found in that column. This is a common strategy for handling missing data, especially for categorical variables. After executing this code, the 'romantic' column will have missing values filled with the most common value, and you'll have a more complete dataset for analysis.

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

```
pivot_table_reason_study_time = df.pivot_table(values='studytime',
                                               index='reason',
                                               aggfunc={'studytime': ['min', 'max']})

print(pivot_table_reason_study_time)
```

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

Explanation:

This line uses the `pivot_table` function from pandas to create a table. It calculates the minimum and maximum study times ('studytime') for each unique value in the 'reason' column.

The code generates a pivot table to analyze study time statistics based on the reasons students have for choosing the school.

The pivot table has 'reason' as the index and two aggregated columns for 'studytime': one showing the minimum study time and another showing the maximum study time for each reason.

The output provides a tabular view of study time characteristics for different reasons, allowing for a comparison of study habits across various motivations for school selection.

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

```
students_with_teacher_parents = df[(df['Mjob'] == 'teacher') & (df['Fjob'] == 'teacher')]
print(len(students_with_teacher_parents))
students_with_teacher_parents
```

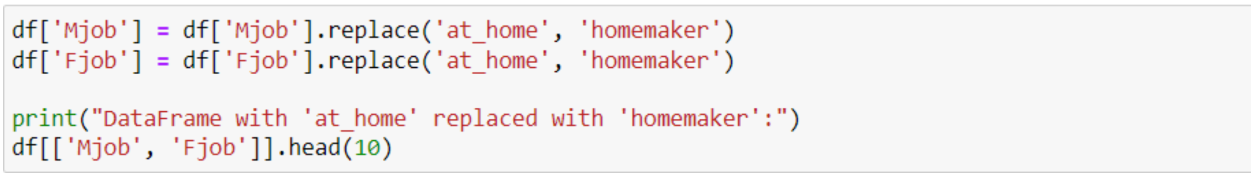
```
16
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	guardian	traveltime	studytime	failures	schoolsup	famsup	paid
29	GP	M	16	U	2	T	4	4	teacher	teacher	home	mother	1	2	0	no	yes	yes
110	GP	M	15	U	2	A	4	4	teacher	teacher	course	mother	1	1	0	no	no	no
115	GP	M	16	U	1	T	4	4	teacher	teacher	course	father	1	2	0	no	yes	no
128	GP	M	16	R	2	T	4	4	teacher	teacher	course	mother	1	1	0	no	no	no
147	GP	F	15	U	1	T	4	4	teacher	teacher	course	mother	2	1	0	no	no	no
161	GP	M	16	U	1	T	4	4	teacher	teacher	course	mother	1	1	0	no	yes	no
213	GP	F	16	U	3	T	4	4	teacher	teacher	reputation	mother	1	2	0	no	yes	no
246	GP	M	17	U	6	T	4	4	teacher	teacher	reputation	mother	1	2	0	yes	yes	no
257	GP	M	17	U	7	T	4	4	teacher	teacher	course	mother	1	1	0	no	yes	no
335	GP	M	18	U	5	A	4	4	teacher	teacher	reputation	mother	1	2	0	no	yes	no
344	GP	M	18	U	3	T	4	4	teacher	teacher	home	mother	1	1	0	no	yes	no
356	GP	F	17	R	5	T	4	4	teacher	teacher	course	mother	1	1	0	no	no	no
381	GP	F	17	U	4	T	4	4	teacher	teacher	course	mother	2	3	0	no	yes	no
448	MS	F	16	R	7	T	4	4	teacher	teacher	course	mother	2	3	0	no	no	no
594	MS	F	18	U	4	T	4	4	teacher	teacher	reputation	mother	2	2	0	no	no	no
636	MS	M	18	U	3	T	4	4	teacher	teacher	home	father	1	2	0	no	no	no

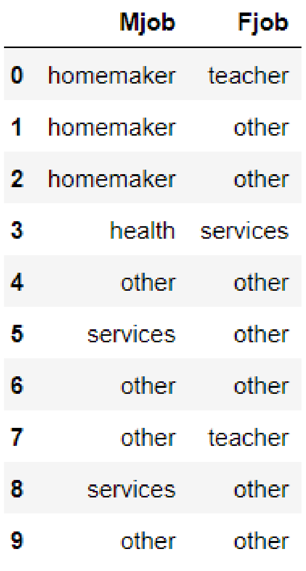
Explanation:

In simpler terms, the code is like a filter that sifts through the student list, finds and counts how many students have both parents working as teachers, and then shows you who these students are along with their parents' details. It's a way of pinpointing students with a specific family background in the given data.

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



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

print("DataFrame with 'at_home' replaced with 'homemaker':")
df[['Mjob', 'Fjob']].head(10)
```



	Mjob	Fjob
0	homemaker	teacher
1	homemaker	other
2	homemaker	other
3	health	services
4	other	other
5	services	other
6	other	other
7	other	teacher
8	services	other
9	other	other

In simpler terms, the code is updating job information in a DataFrame by changing the label 'at_home' to 'homemaker' in both the mother's job ('Mjob') and father's job ('Fjob') columns. The printed output provides a glimpse of the DataFrame after these replacements have been applied to the first 10 rows.

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

```
melted_df = pd.melt(df, id_vars=df.columns.difference(['Mjob', 'Fjob']),
                    value_vars=['Mjob', 'Fjob'],
                    var_name='Parent',
                    value_name='Parent Job')

print("Melted DataFrame with 'Mjob' and 'Fjob' melted into 'Parent Job':")
melted_df.head()
```

	Dalc	Fedu	G1	G2	G3	Medu	Pstatus	Walc	absences	activities	address	age	failures	famrel	famrel_label	famsize	famsup	freetime	goout	guardia
0	1	4	0	11	11	4	A	1	4	no	U	18	0	4	good	1	no	3	4	moth
1	1	1	9	11	11	1	T	1	2	no	U	17	0	5	excellent	2	yes	3	3	fath
2	2	1	12	13	12	1	T	3	6	no	U	15	0	4	good	1	no	3	2	moth
3	1	2	14	14	14	4	T	1	0	yes	U	15	0	3	neutral	1	yes	2	2	moth
4	1	3	11	13	13	3	T	2	0	no	U	16	0	4	good	1	yes	3	2	fath

In summary, the code transforms the original DataFrame by combining the 'Mjob' and 'Fjob' columns into a single 'ParentJob' column, while preserving the other columns in their original form. The resulting melted DataFrame has a 'Parent' column indicating whether the job information is related to the mother ('Mjob') or father ('Fjob'), and a 'Parent Job' column containing the job information.

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.

```
def assign_letter_grade(grade):
    if grade >= 17:
        return 'A'
    elif grade >= 15:
        return 'B'
    elif grade >= 13:
        return 'C'
    elif grade >= 11:
        return 'D'
    else:
        return 'F'

df['Letter Grade'] = df['G3'].apply(assign_letter_grade)
df[['G3', 'Letter Grade']].head()
```

	G3	Letter Grade
0	11	D
1	11	D
2	12	D
3	14	C
4	13	C

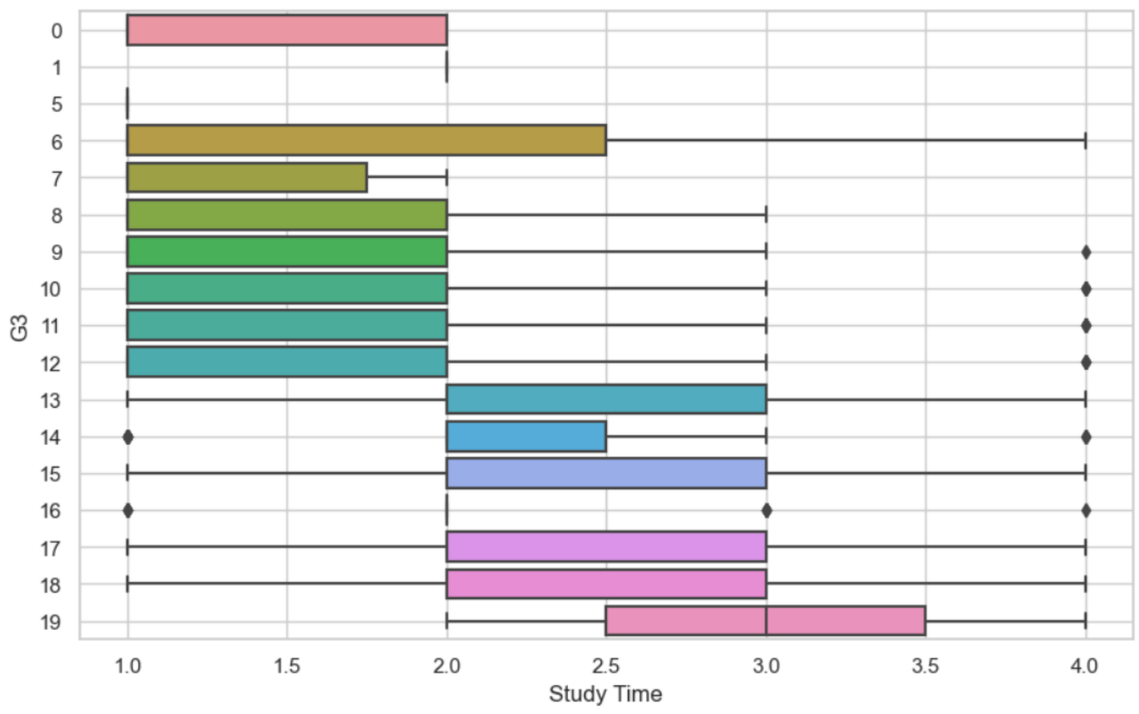
```
df['Letter Grade'].value_counts()
```

```
F    197
D    176
C    145
B     85
A     46
Name: Letter Grade, dtype: int64
```

Explanation:
This code creates a custom function assign_letter_grade that takes a numerical grade (G3) as input and assigns a corresponding letter grade ('A', 'B', 'C', 'D', or 'F'). Then, it applies this function to the 'G3' column of the DataFrame (df) and adds a new column called 'Letter Grade' to store the letter grades. Finally, it prints the first few rows of the 'G3' and 'Letter Grade' columns

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

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='studytime', y='G3', data=df, orient='h')
plt.xlabel('Study Time')
plt.ylabel('G3')
plt.grid(True)
plt.show()
```



This code creates a time series plot (boxplot) to visualize how the distribution of final grades ('G3') varies with different levels of weekly study time ('studytime'). The plot is set to be horizontally oriented, making it easier to compare the study time and final grades for a specific student. The grid is added for better reference, and the resulting plot provides insights into the relationship between study time and academic performance.

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

```
math_and_por = pd.merge(df, df2, how ='inner', on =['school', 'sex', 'age', 'address', 'famsize', 'Pstatus',
                                                     'Medu', 'Fedu', 'Mjob', 'Fjob', 'Dalc', 'Walc'])
```

```
math_and_por.head()
```

	school_x	sex_x	age_x	address_x	famsize_x	Pstatus_x	Medu_x	Fedu_x	Mjob_x	Fjob_x	reason_x	guardian_x	traveltime_x	studytime_x
student_id														
0	GP	F	18	U	1	A	4	4	homemaker	teacher	course	mother	2	2
1	GP	F	17	U	2	T	1	1	homemaker	other	course	father	1	2
2	GP	F	15	U	1	T	1	1	homemaker	other	other	mother	1	2
3	GP	F	15	U	1	T	4	2	health	services	home	mother	1	3
4	GP	F	16	U	1	T	3	3	other	other	home	father	1	2

```
math_and_por.shape
```

```
(395, 75)
```

Explanation:
The code merges data from the Math and Portuguese courses based on specific columns, creating a new DataFrame (math_and_por) that includes information about students who are present in both datasets. The resulting DataFrame has 395 rows and 75 columns.

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

```
gp_stu = df[df['school'] == 'GP']
top_gp_students = gp_stu.sort_values(by='G3', ascending=False)
top_5_gp_students = top_gp_students.head()
print("Top 5 students with the highest final grades in 'GP' school:")
print(top_5_gp_students[['school', 'G3']])
```

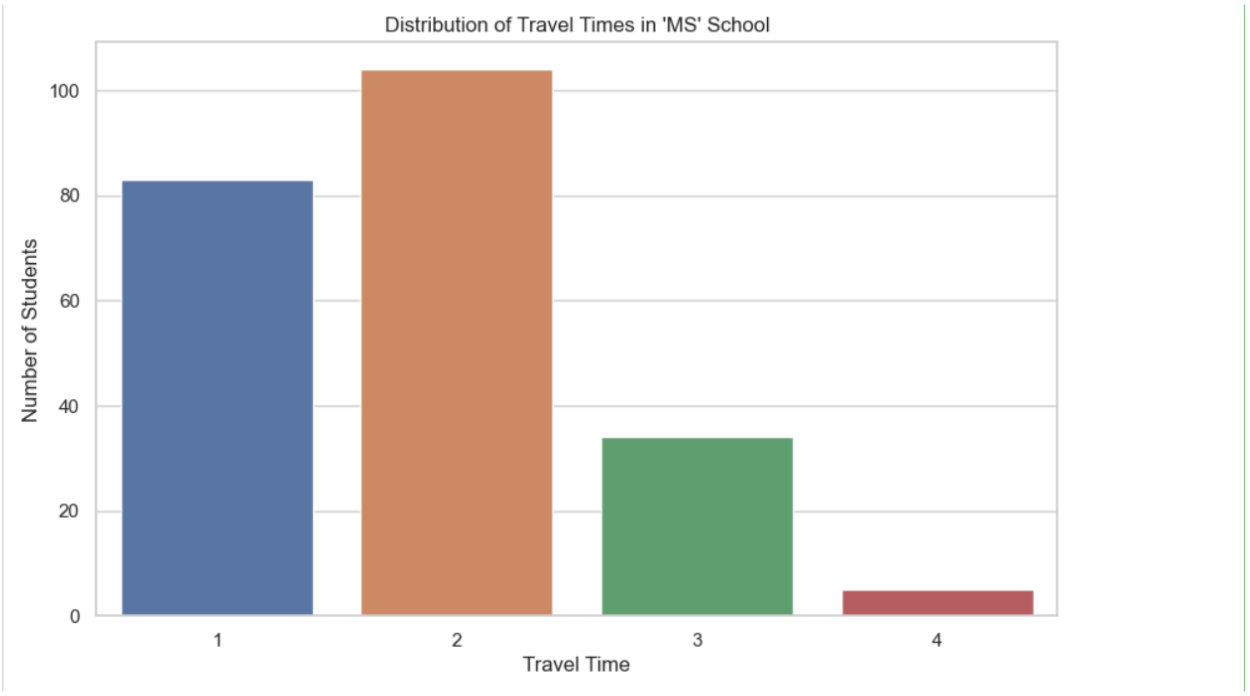
```
Top 5 students with the highest final grades in 'GP' school:
           school  G3
student_id
338            GP  19
416            GP  18
185            GP  18
332            GP  18
314            GP  18
```

Explanation:

This code selects students from the 'GP' school in the DataFrame (df), sorts them based on their final grades ('G3') in descending order, and then prints the top 5 students with the highest final grades. In summary, this code identifies and prints the top 5 students with the highest final grades ('G3') specifically from the 'GP' school in the DataFrame.

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

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

plt.figure(figsize=(10, 6))
sns.countplot(x='traveltime', data=ms_students)
plt.title('Distribution of Travel Times in \'MS\' School')
plt.xlabel('Travel Time')
plt.ylabel('Number of Students')
plt.show()
```



Explanation:
This code creates a bar chart to show the distribution of students' travel times in the 'MS' school. The x-axis represents different travel times, and the y-axis represents the number of students with each travel time. The title and labels provide context for the plot.

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

```
mean_age_with_activities = df[df['activities'] == 'yes']['age'].mean()
mean_age_without_activities = df[df['activities'] == 'no']['age'].mean()

print("Mean age of students with extra-curricular activities:", mean_age_with_activities)
print("Mean age of students without extra-curricular activities:", mean_age_without_activities)
```

```
Mean age of students with extra-curricular activities: 16.676190476190477
Mean age of students without extra-curricular activities: 16.808383233532933
```

Explanation:
In summary, the code provides the mean age of students based on whether they have extra-curricular activities. Students with extra-curricular activities have a mean age of approximately 16.68, while students without extra-curricular activities have a mean age of approximately 16.81.
Comparing these values, we can observe that, on average, students without extra-curricular activities have a slightly higher mean age (16.808) compared to students with extra-curricular activities (16.676). The difference is relatively small, and it suggests that there might be a subtle variation in age between the two groups. However, it's essential to note that mean age is just one measure, and further statistical analysis could provide more insights into the significance of this difference.

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

```
median_absences_by_group = df.groupby(['sex', 'address'])['absences'].median()
print("Median number of school absences grouped by 'sex' and 'address':")
print(median_absences_by_group)
```

```
Median number of school absences grouped by 'sex' and 'address':
sex  address
F    R          2.0
     U          2.0
M    R          2.0
     U          2.0
Name: absences, dtype: float64
```

Explanation:
In summary, the code provides the median number of school absences grouped by the students' gender ('sex') and the type of address ('address'). The output shows the median absences for each combination of 'sex' and 'address'. For example, for female students with a rural address ('F' and 'R'), the median number of absences is 6.0.

47) Calculate the percentage of students who receive extra educational support (schoolsup) in the 'GP' school.

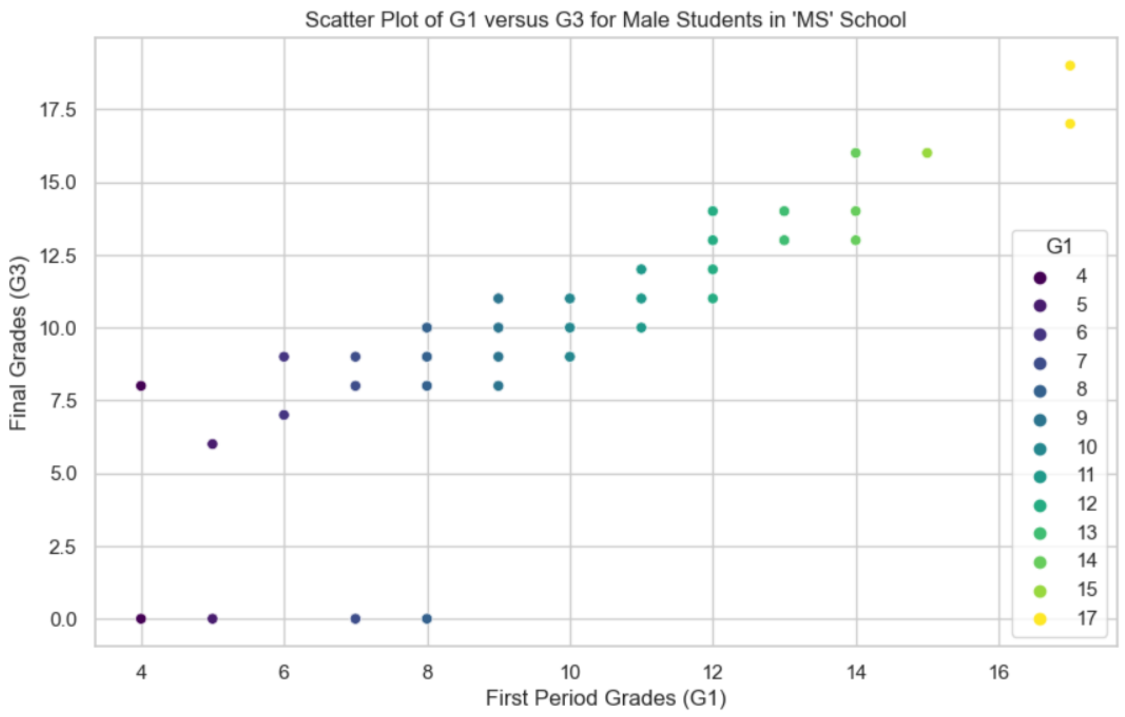
```
gp_students = df[df['school'] == 'GP']

percentage_schoolsup_gp = (gp_students['schoolsup'].value_counts(normalize=True) * 100).loc['yes']

print(f"Percentage of students with extra educational support in 'GP' school: {percentage_schoolsup_gp:.2f}%")
```

```
Percentage of students with extra educational support in 'GP' school: 13.24%
```

Explanation:
It calculates the percentage of students with extra educational support by counting the occurrences of 'yes' and 'no' in the 'schoolsup' column and then converting these counts to percentages.
The result is a series of percentages, and .loc['yes'] extracts the percentage of students with extra educational support.
The code provides the percentage of students with extra educational support in the 'GP' school, and the output indicates that approximately 13.24% of students in the 'GP' school receive this additional support.

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

```
ms_male_students = df[(df['school'] == 'MS') & (df['sex'] == 'M')]

plt.figure(figsize=(10, 6))
sns.scatterplot(x='G1', y='G3', data=ms_male_students, hue='G1', palette='viridis', legend='full')
plt.title('Scatter Plot of G1 versus G3 for Male Students in \'MS\' School')
plt.xlabel('First Period Grades (G1)')
plt.ylabel('Final Grades (G3)')
plt.show()
```



Explanation:
This code generates a scatter plot to visually explore the correlation between the first period grades ('G1') and final grades ('G3') for male students from the 'MS' school. The color of each point represents the first period grade, and the legend provides information about the color scale. The title and labels enhance the interpretability of the plot.

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

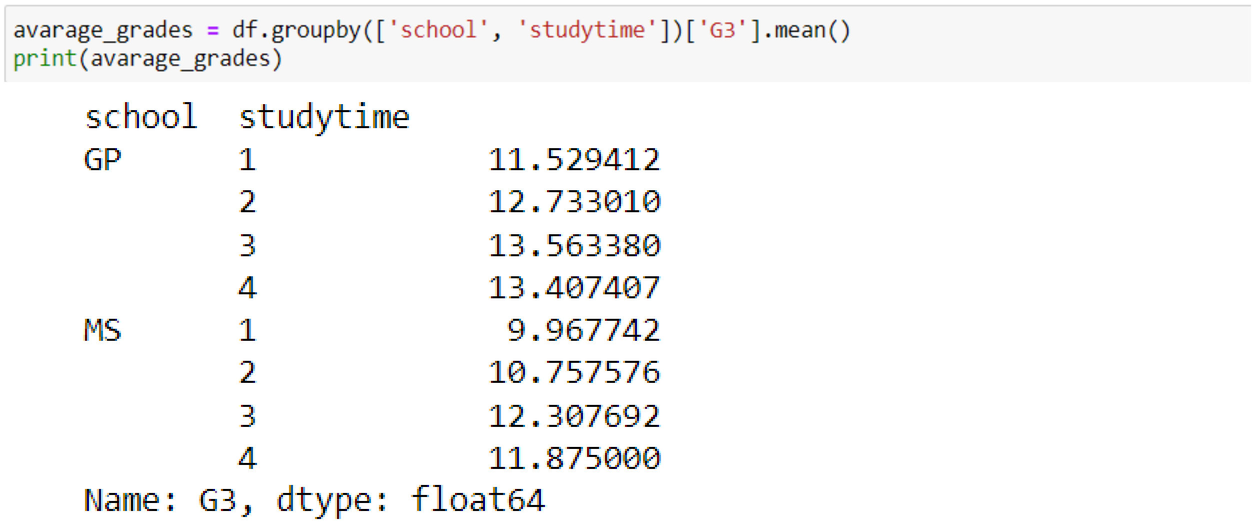
```python
import pandas as pd

unique_students = df[df.groupby(['Mjob', 'Fjob']).transform('size') == 1]

unique_students
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	guardian	traveltime	studytime	failures	schoolsup	famsup
student_id																	
588	MS	F	17	U	5	T	4	1	health	homemaker	course	mother	1	1	0	no	yes

Explanation:
The code you provided aims to identify students with a unique combination of 'Mjob' (mother's job) and 'Fjob' (father's job) that appears only once in the dataset.

(df.groupby(['Mjob', 'Fjob']).transform('size') == 1): Compares the size of each group to 1, checking if the combination of 'Mjob' and 'Fjob' is unique for each row.
The corrected code correctly identifies students with a unique combination of 'Mjob' and 'Fjob' that appears only once in the dataset and stores them in the unique_students DataFrame.

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



```
avarage_grades = df.groupby(['school', 'studytime'])['G3'].mean()
print(avarage_grades)
```

```
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
```

Explanation:
The code organizes the data by grouping it based on the school ('GP' or 'MS') and the amount of study time. It then calculates the average final grade ('G3') for each group.
df.groupby(['school', 'studytime']): Groups the DataFrame by the 'school' and 'studytime' columns.   ['G3']: Selects the 'G3' column for further analysis.
The output would be a series that shows the average final grade for each combination of 'school' and 'studytime'. It provides insights into how the average final grades vary based on the school and the amount of study time.