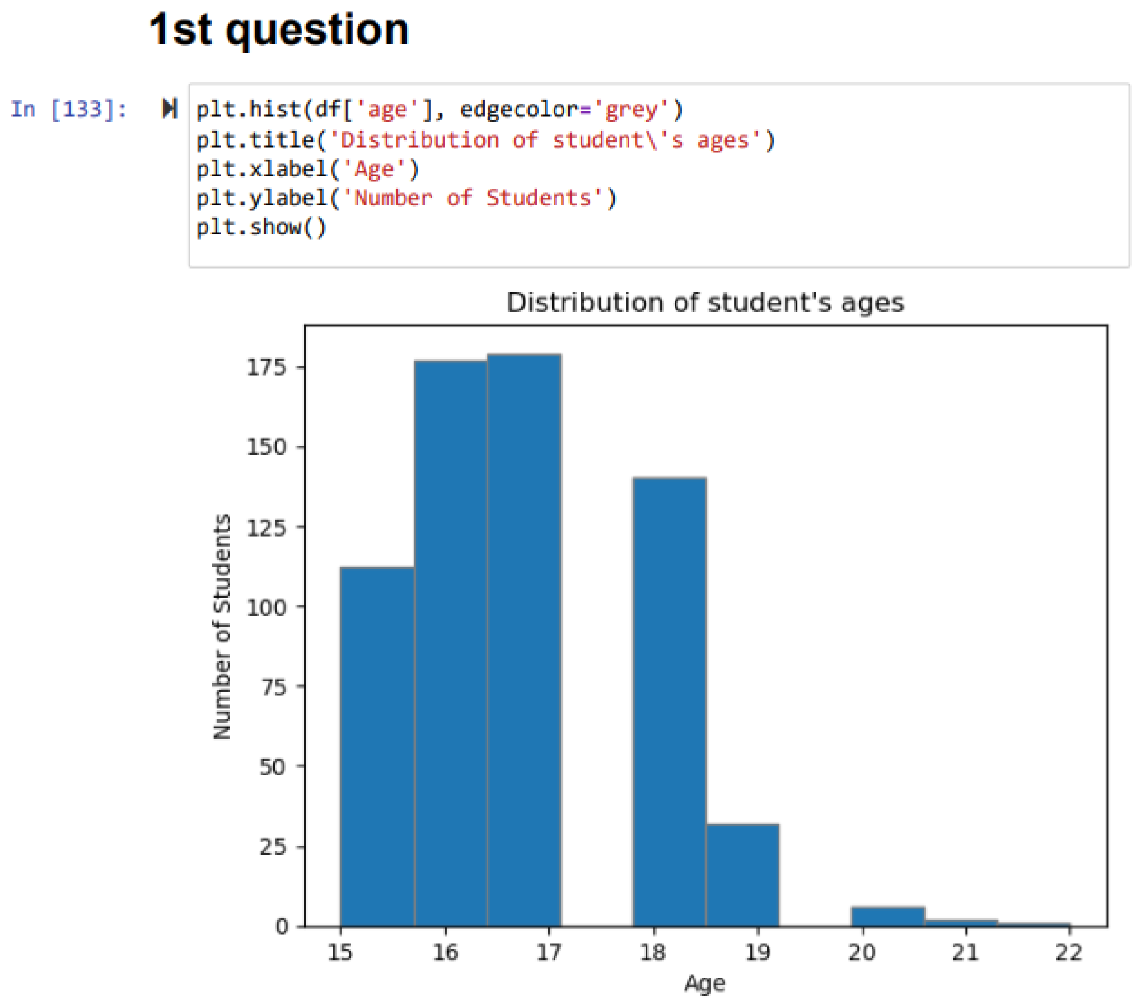
Garifulla Amir
Serek Azamat
Data Analysis
10 September 2023

Report for Data Analysis project

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

   Here we can see the distribution of students' ages by matplotlib.pyplot's hist function.

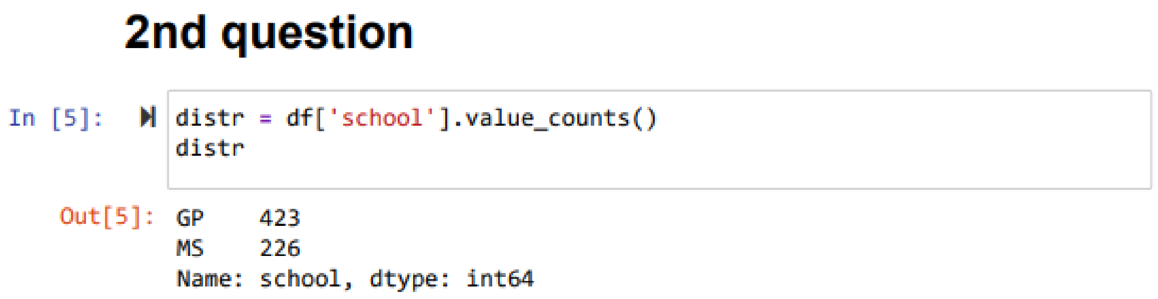


1st question

In [133]:
```
plt.hist(df['age'], edgecolor='grey')
plt.title('Distribution of student\'s ages')
plt.xlabel('Age')
plt.ylabel('Number of Students')
plt.show()
```

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

By value_counts function we can see the counts of unique values in the dataframe.



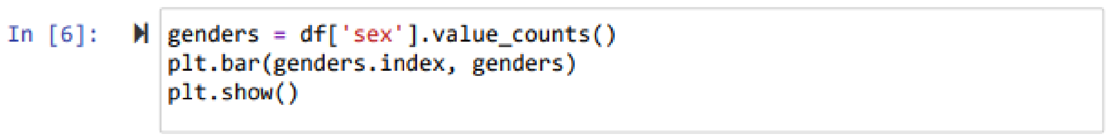
2nd question

```
In [5]:  distr = df['school'].value_counts()
         distr

Out[5]:  GP    423
         MS    226
         Name: school, dtype: int64
```

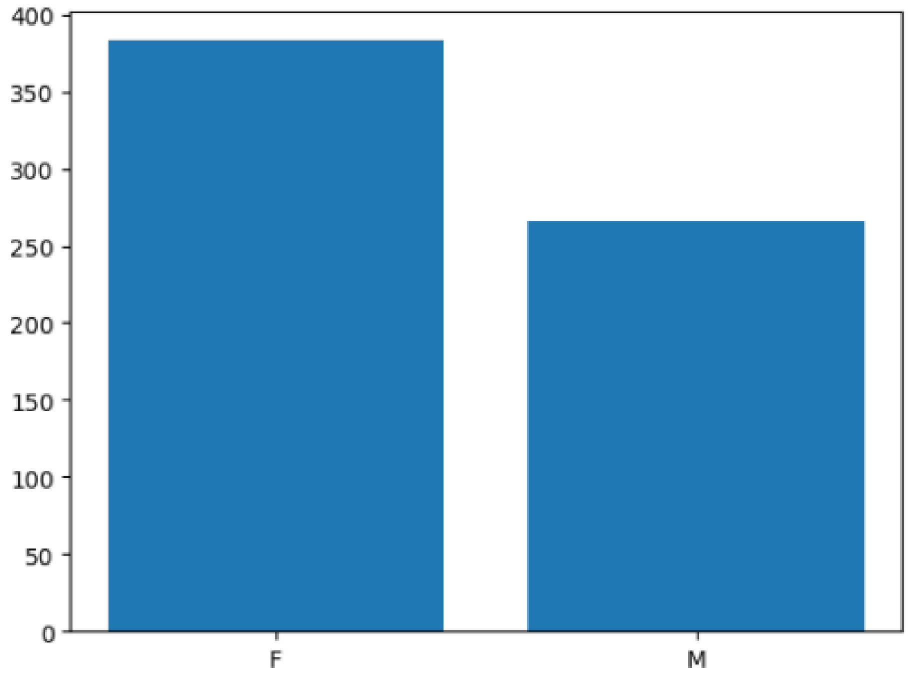
3) What is the gender distribution of students?

First of all we should take only the ‘sex’ column and apply the value_counts function. Then use plt’s bar function to draw a bar chart.

3rd question



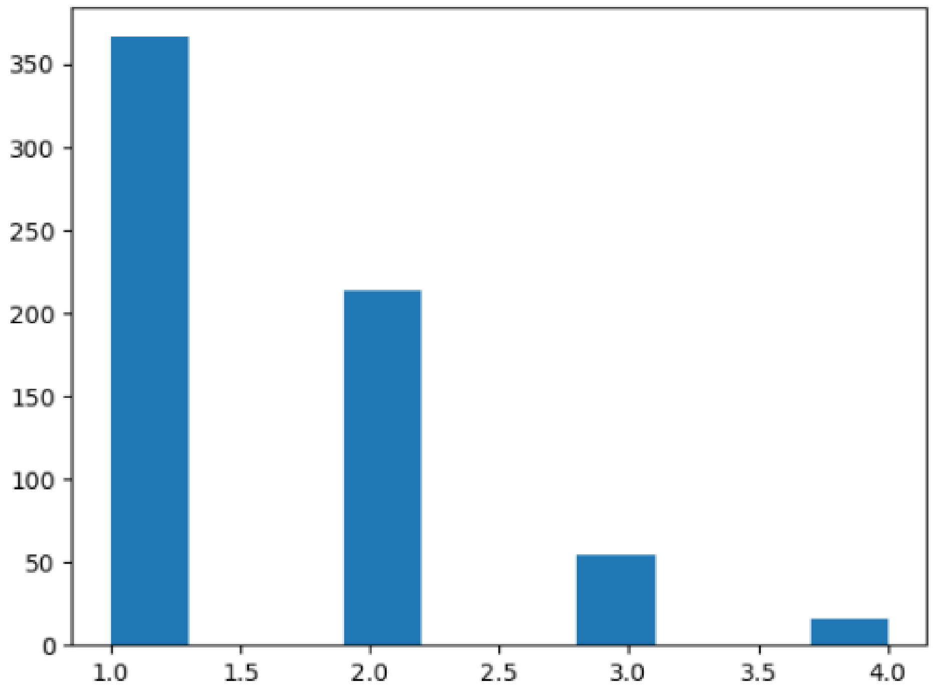
In [6]:
```python
genders = df['sex'].value_counts()
plt.bar(genders.index, genders)
plt.show()
```



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

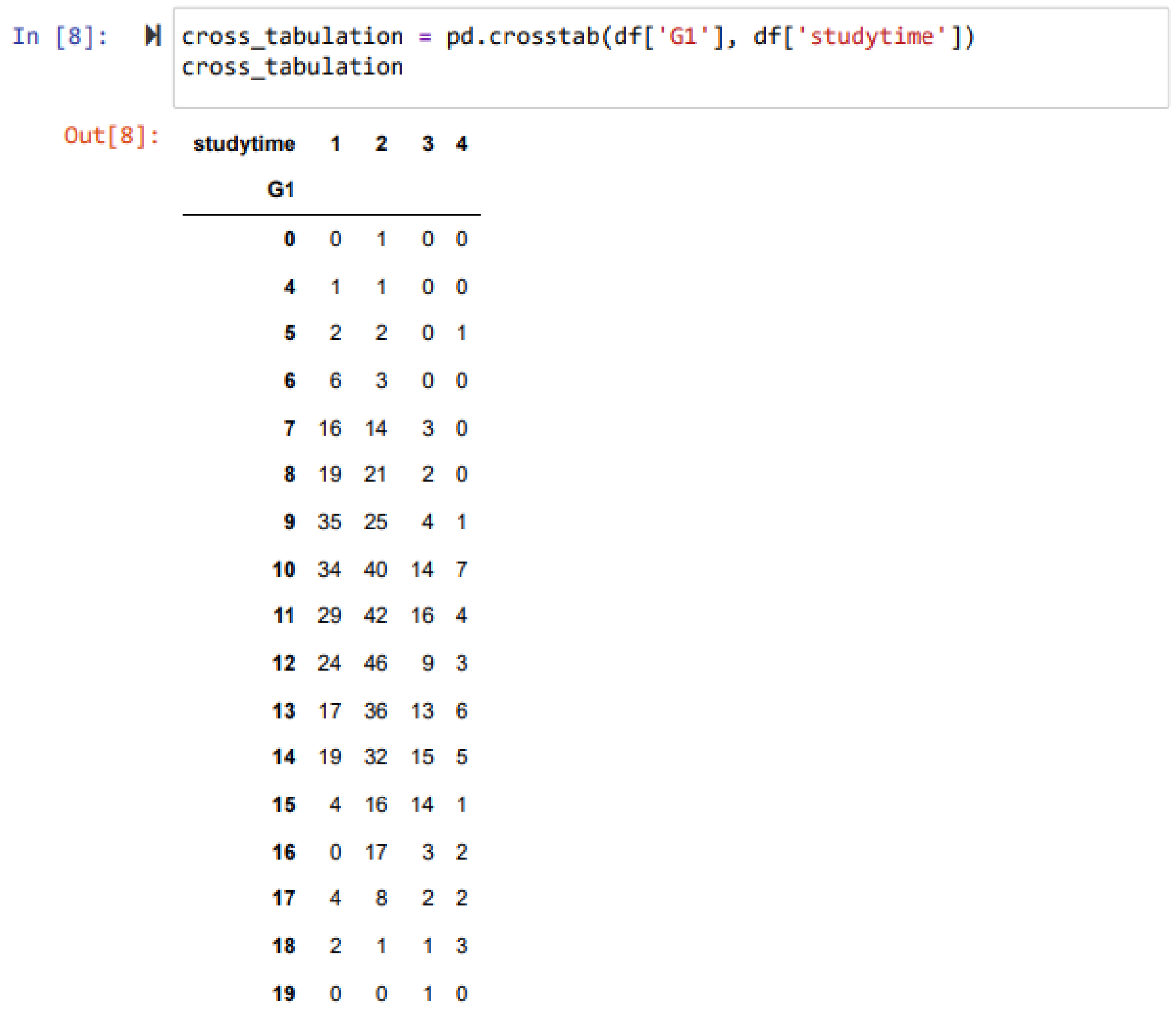
We take traveltime column and put it inside plt’s hist function. It will return a histogram where x axis is traveltime and y axis is counts of its appearance.

```
In [7]:  plt.hist(df['traveltime'])
          plt.show()
```



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

   We take from dataframe G1 and studytime columns and put it inside pandas' crosstab function. It will return crosstabulation of these two columns.



In [8]:
```python
cross_tabulation = pd.crosstab(df['G1'], df['studytime'])
cross_tabulation
```

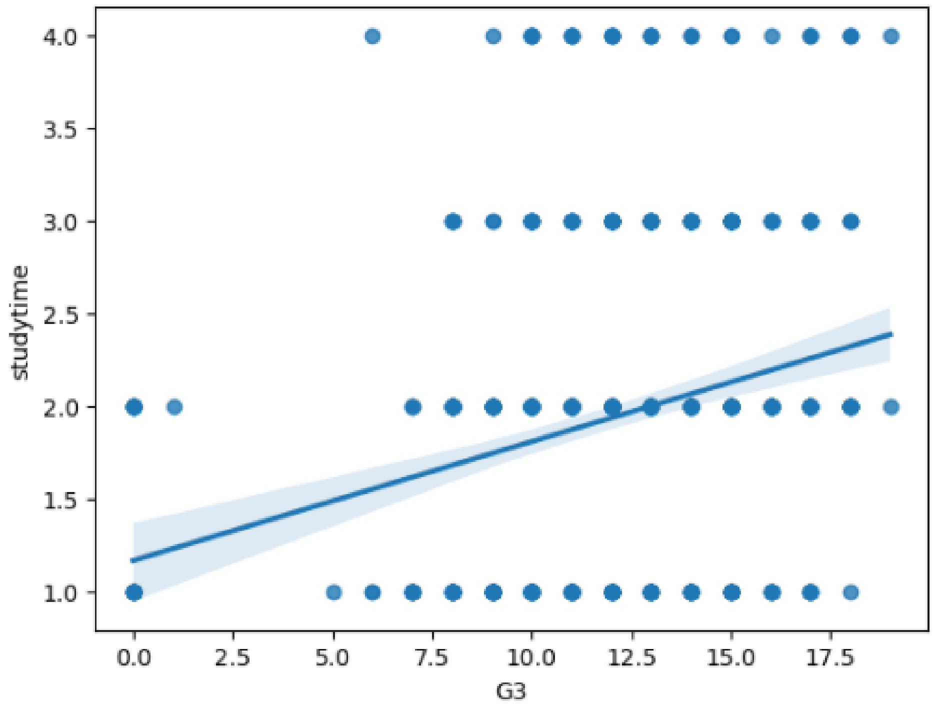
Out[8]:

studytime	1	2	3	4
G1				
0	0	1	0	0
4	1	1	0	0
5	2	2	0	1
6	6	3	0	0
7	16	14	3	0
8	19	21	2	0
9	35	25	4	1
10	34	40	14	7
11	29	42	16	4
12	24	46	9	3
13	17	36	13	6
14	19	32	15	5
15	4	16	14	1
16	0	17	3	2
17	4	8	2	2
18	2	1	1	3
19	0	0	1	0

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

   We take from dataframe G3 and studytime columns and put it inside seaborn’s regplot function. It will return a regression plot of these two columns where x axis is G3 and y axis is studytime. By which we can see relation of these variables.

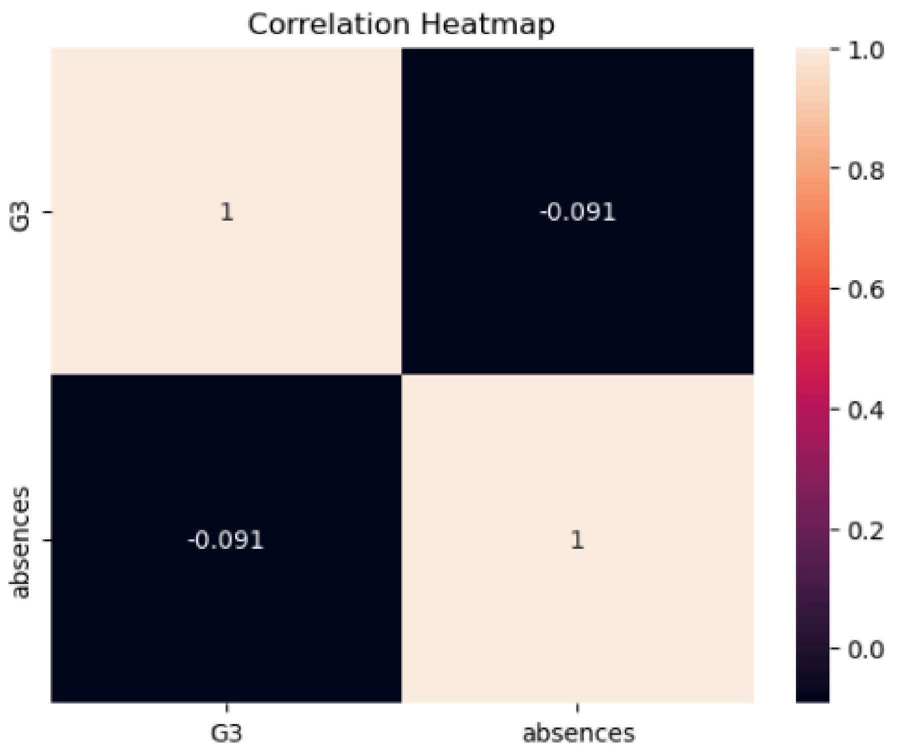
```
In [9]:  sns.regplot(x = df['G3'], y = df['studytime'])
          plt.show()
```



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

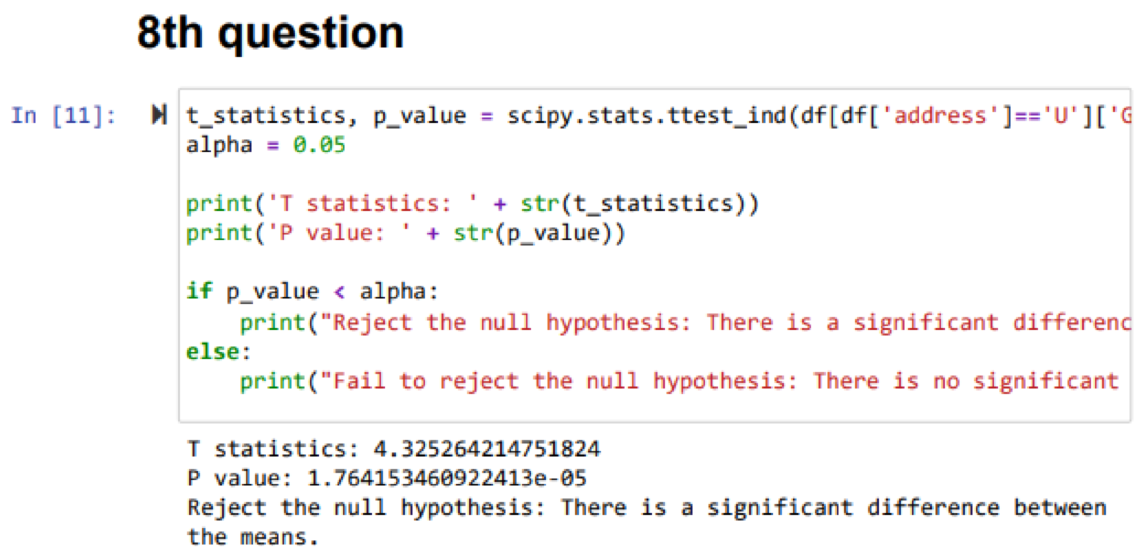
To find relation between variables we use corr function. It returns correlation matrix which we visualize by seaborn’s heatmap function.

In [10]:
```
correlation_matrix = df[['G3', 'absences']].corr()
sns.heatmap(correlation_matrix, annot=True)
plt.title('Correlation Heatmap')
plt.show()
```



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

We test the difference between urban and rural areas' grades by ttest_ind function from scipy.stats. We take first urban area's g3 and rural's g3 and put into ttest_ind function. It will return t_value and p_value. Then we compare p_value with our confidence level(alpha = 0.05). As the result there is no difference.



8th question

In [11]:

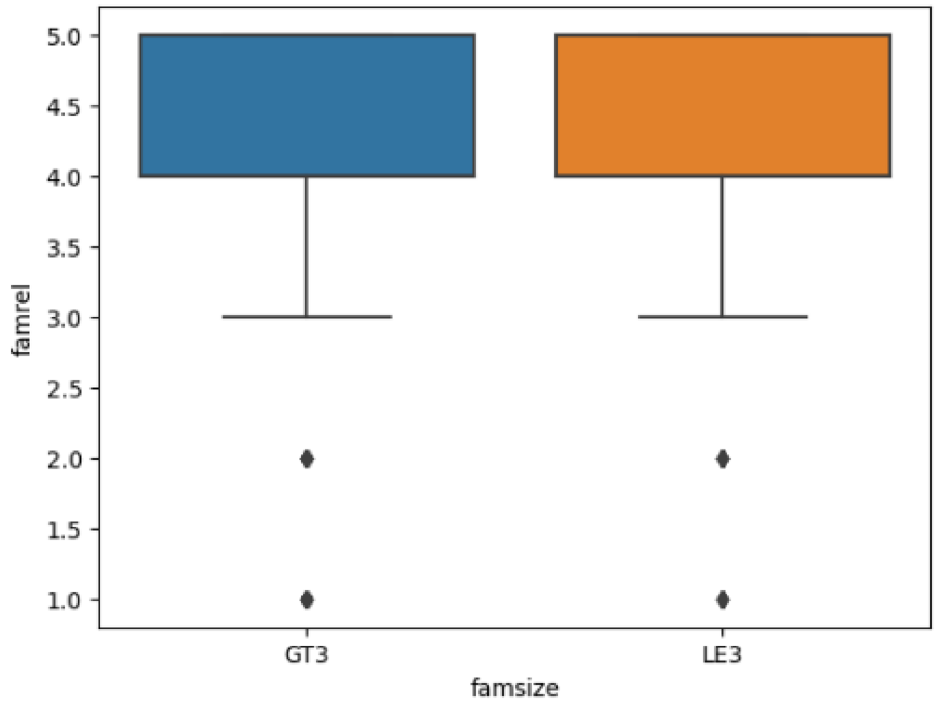
```
t_statistics, p_value = scipy.stats.ttest_ind(df[df['address']=='U']['G
alpha = 0.05

print('T statistics: ' + str(t_statistics))
print('P value: ' + str(p_value))

if p_value < alpha:
    print("Reject the null hypothesis: There is a significant differenc
else:
    print("Fail to reject the null hypothesis: There is no significant 
```

```
T statistics: 4.325264214751824
P value: 1.764153460922413e-05
Reject the null hypothesis: There is a significant difference between
the means.
```

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

We will use seaborn's boxplot function where x axis is famsize and y axis is famrel. As the result there is box plot by which it is clear that there is no relation between these variables.

```
In [12]:  sns.boxplot(x = 'famsize', y = 'famrel', data = df)
          plt.show()
```

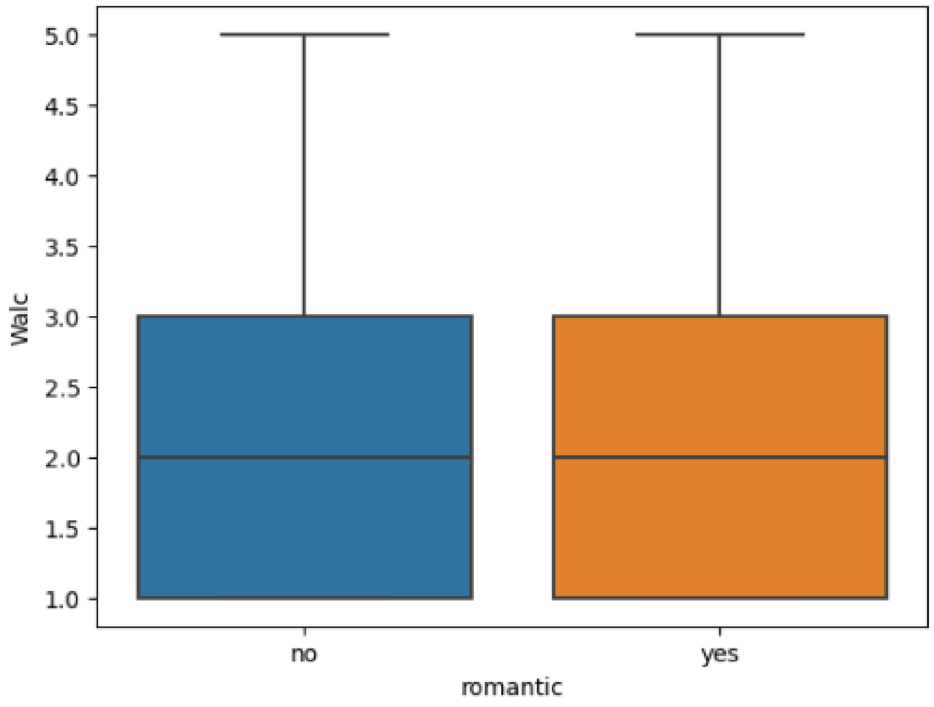


10) Does the presence of romantic relationships (romantic) affect students' alcohol consumption (Dalc and Walc)?

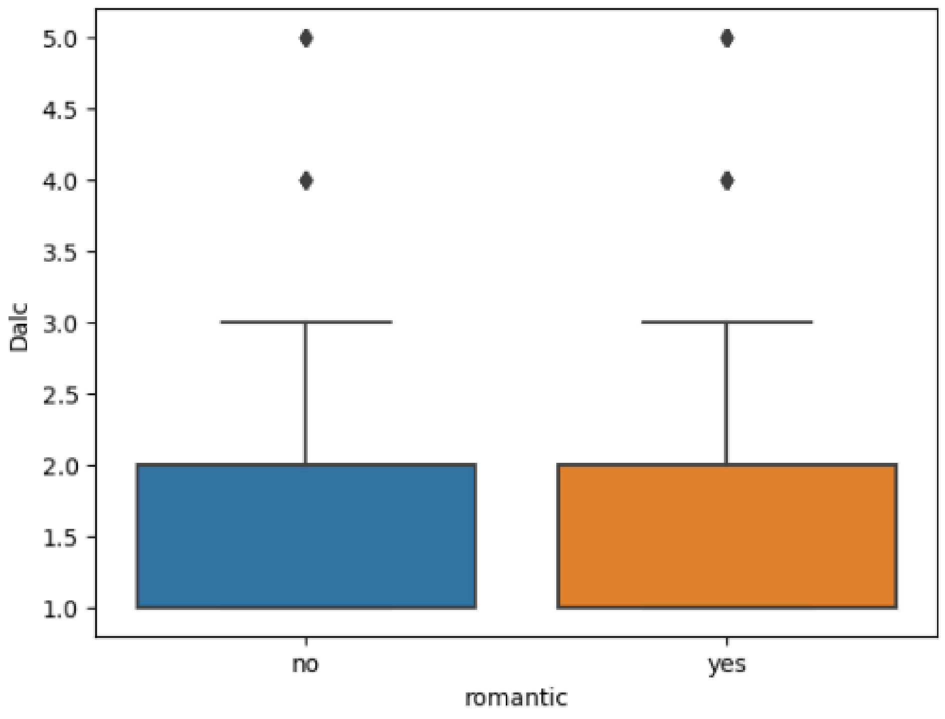
There we can use seaborn's boxplot function. As argument we put x axis as romantic and y axis as dalc and walc. As the result there are two box plots from which it is clear that romantic relationship does not affect student's alcohol consumption.

10th question

In [13]:
```
sns.boxplot(x = 'romantic', y = 'Walc', data = df)
plt.show()
```



```
In [14]:  sns.boxplot(x = 'romantic', y = 'Dalc', data = df)
          plt.show()
```



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

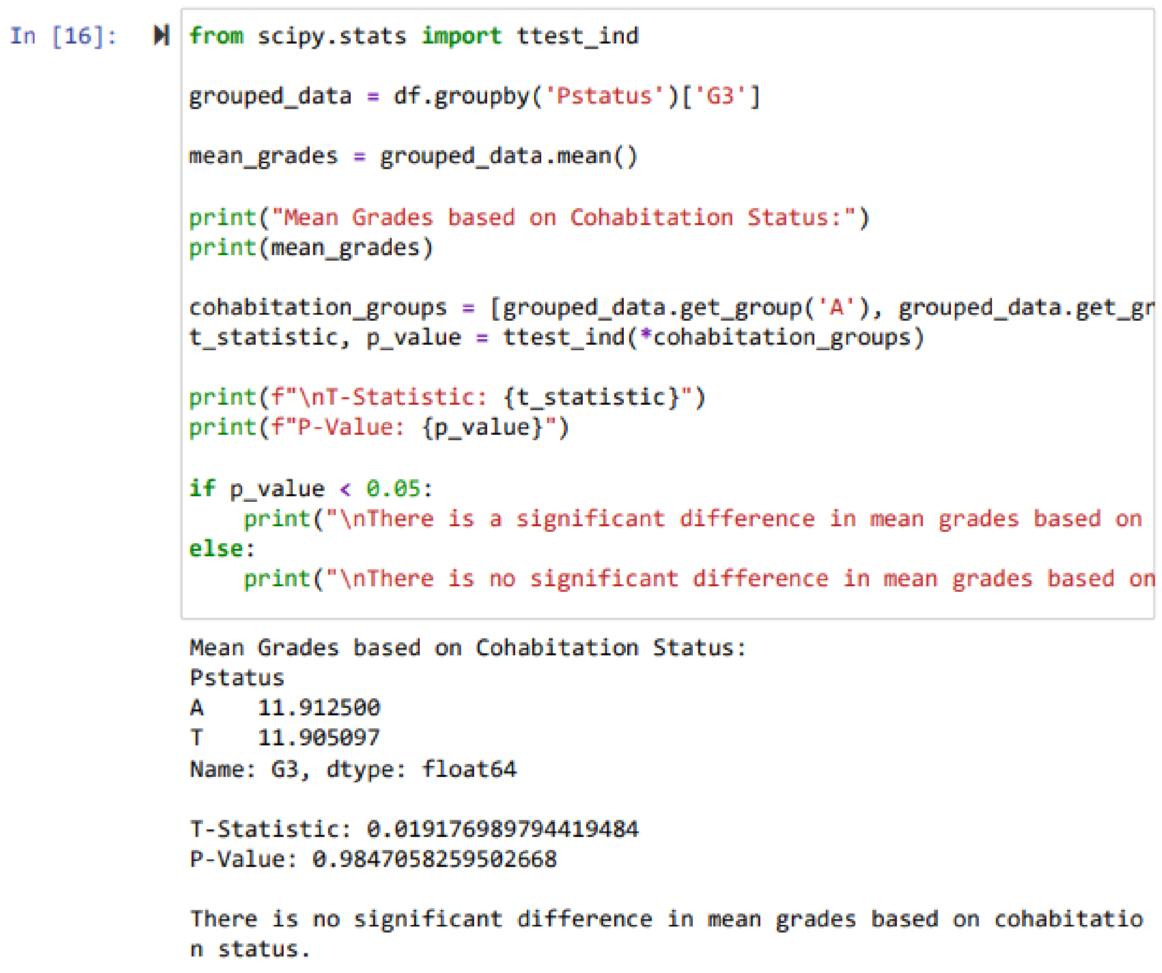
11th question

```
In [15]:  correlation_coefficient = df['Medu'].corr(df['Fedu'])

          print(f"The correlation coefficient between Medu and Fedu is: {correlat
```

```
The correlation coefficient between Medu and Fedu is: 0.64747660913649
51
```

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

First of all we group dataframe by pstatus column and get G3 columns from all groups. Secondly we use ttest_ind function to identify difference between groups in G3. As the result there is no difference.



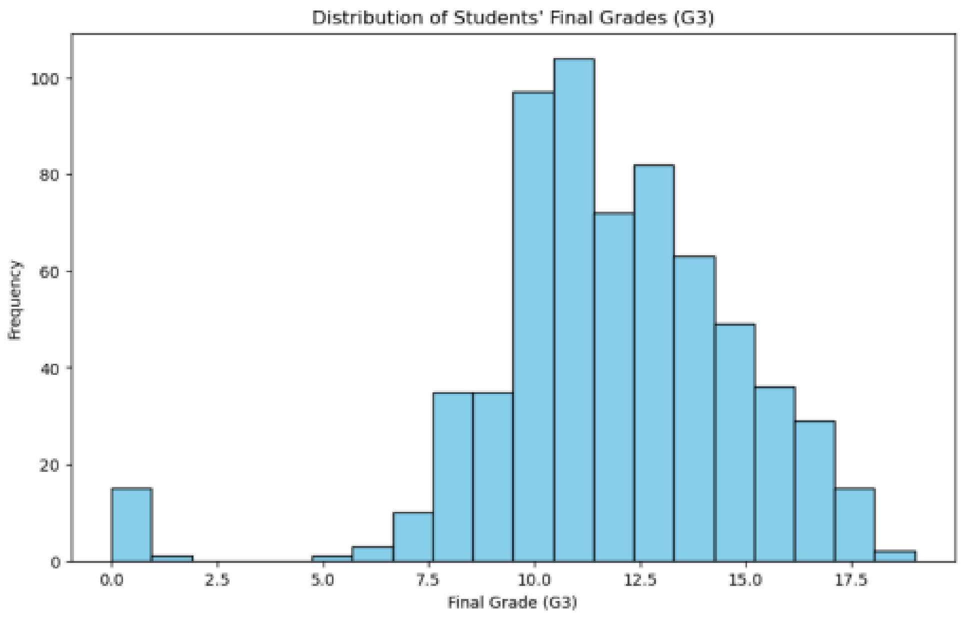
```
In [16]:  from scipy.stats import ttest_ind

          grouped_data = df.groupby('Pstatus')['G3']

          mean_grades = grouped_data.mean()

          print("Mean Grades based on Cohabitation Status:")
          print(mean_grades)

          cohabitation_groups = [grouped_data.get_group('A'), grouped_data.get_gr
          t_statistic, p_value = ttest_ind(*cohabitation_groups)

          print(f"\nT-Statistic: {t_statistic}")
          print(f"P-Value: {p_value}")

          if p_value < 0.05:
              print("\nThere is a significant difference in mean grades based on
          else:
              print("\nThere is no significant difference in mean grades based on
```

```
Mean Grades based on Cohabitation Status:
Pstatus
A    11.912500
T    11.905097
Name: G3, dtype: float64

T-Statistic: 0.019176989794419484
P-Value: 0.9847058259502668

There is no significant difference in mean grades based on cohabitatio
n status.
```

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

To create histogram we use pyplot’s hist function and take as an argument dataframe’s G3 column. As the result we have histogram which shows grade distribution of G3.

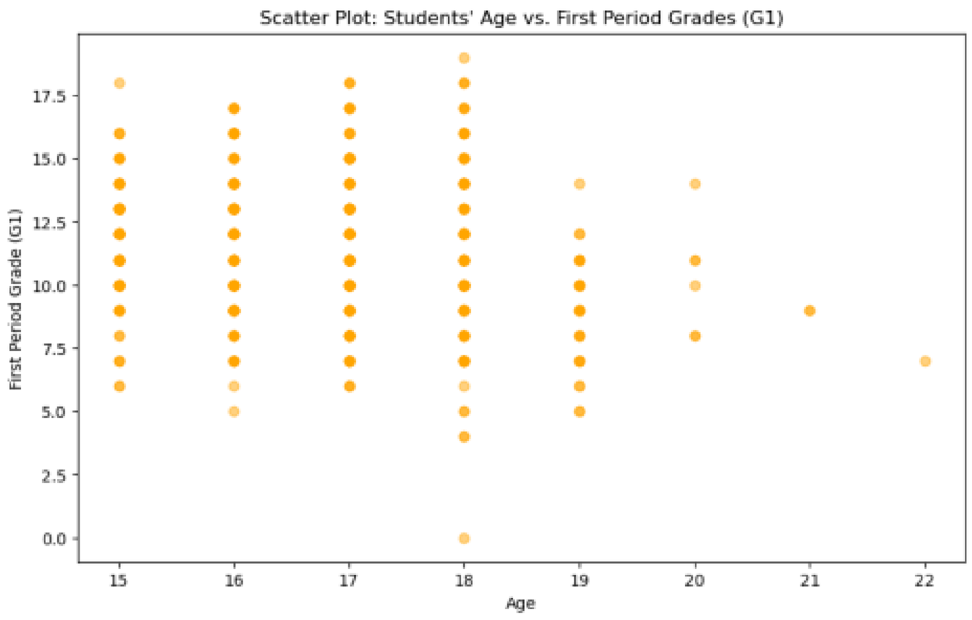
```
In [17]:  plt.figure(figsize=(10, 6))
          plt.hist(df['G3'], bins=20, color='skyblue', edgecolor='black')

          plt.title('Distribution of Students\' Final Grades (G3)')
          plt.xlabel('Final Grade (G3)')
          plt.ylabel('Frequency')

          plt.show()
```



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

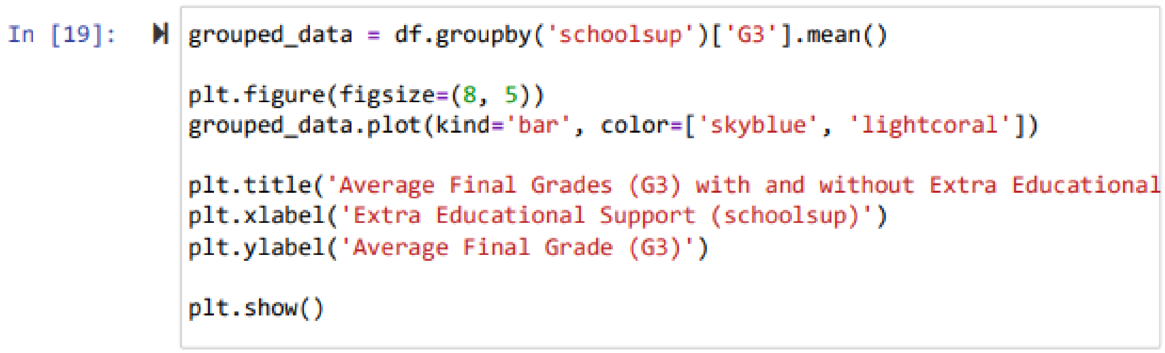
To create scatter plot we use pyplot’s scatter function and take as an arguments dataframe’s age and G1 columns. As the result there is scatter plot showing relation between student’s age and G1.

```
In [18]:  plt.figure(figsize=(10, 6))
          plt.scatter(df['age'], df['G1'], color='orange', alpha=0.5)

          plt.title('Scatter Plot: Students\' Age vs. First Period Grades (G1)')
          plt.xlabel('Age')
          plt.ylabel('First Period Grade (G1)')

          plt.show()
```

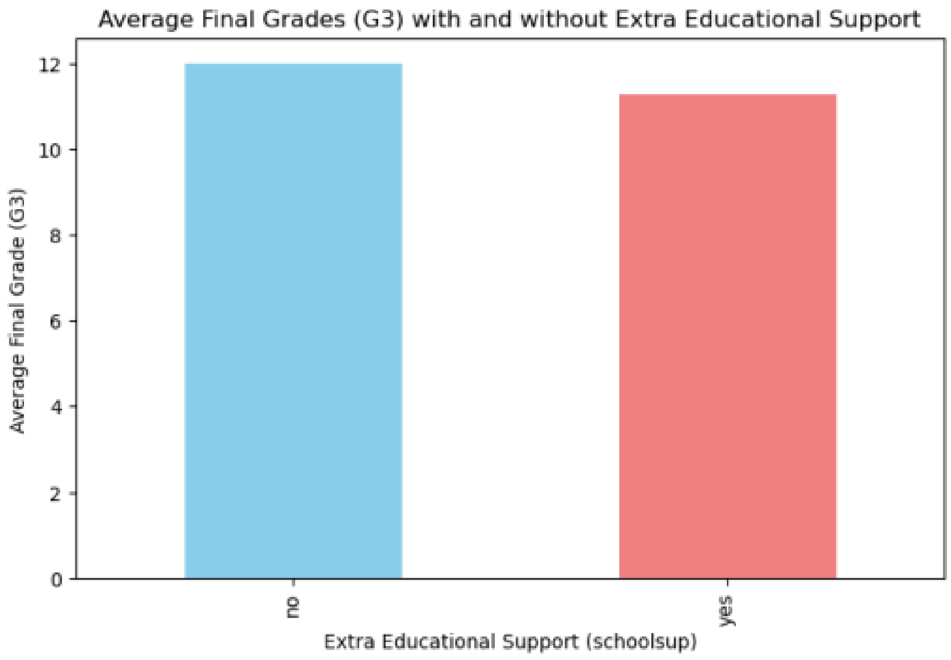


15) Create a bar chart to compare the average final grades (G3) of students with and without extra educational support (schoolsup).

To create a bar chart we will use dataframe's plot function. Dataframe to which we would use plot function should be grouped by schoolsup column and use mean function to G3 column. As a result there will be bar chart with average G3 and grouped with schoolsup.

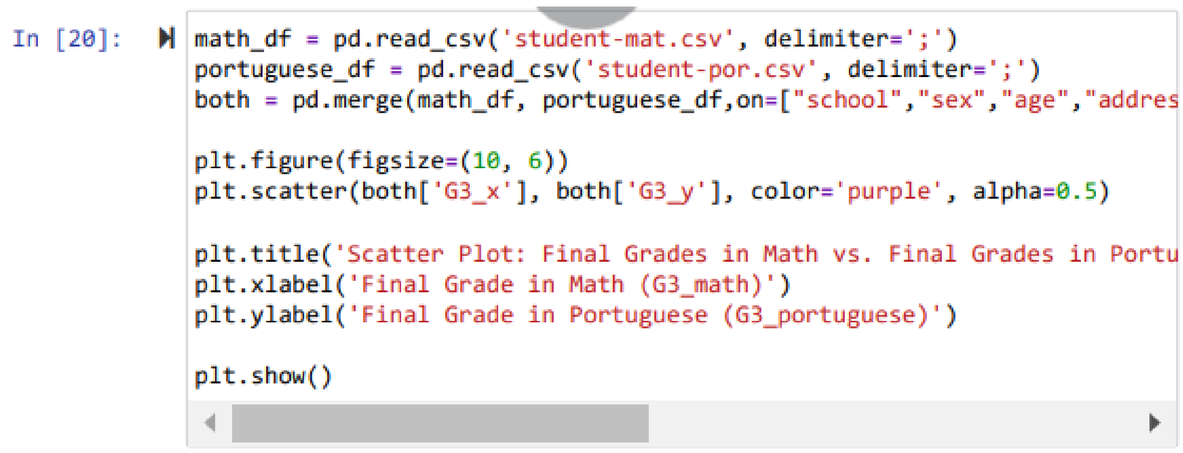


```
In [19]:    grouped_data = df.groupby('schoolsup')['G3'].mean()

            plt.figure(figsize=(8, 5))
            grouped_data.plot(kind='bar', color=['skyblue', 'lightcoral'])

            plt.title('Average Final Grades (G3) with and without Extra Educational
            plt.xlabel('Extra Educational Support (schoolsup)')
            plt.ylabel('Average Final Grade (G3)')

            plt.show()
```

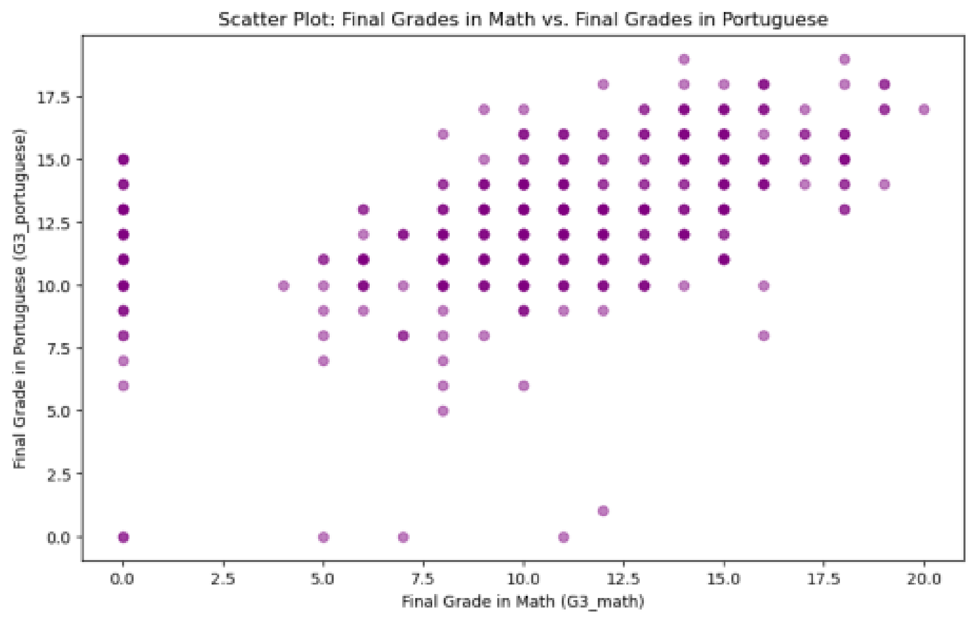


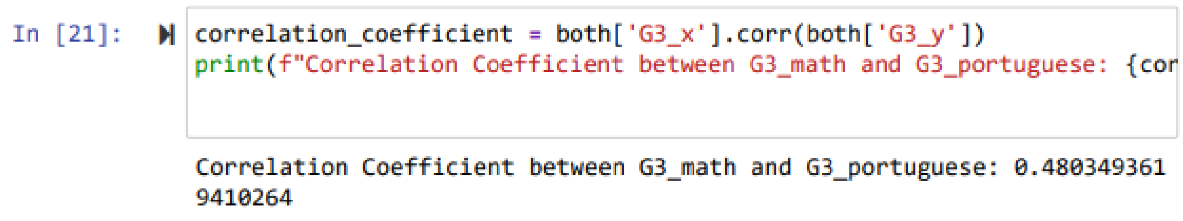
16) How do final grades (G3) in the math course compare to final grades in the Portuguese course for students who belong to both datasets?

First of all we should find students who belong to both datasets. We find it by unique columns that we can identify as primary key. Merging both datasets by its primary key we get these students. Next we will visualize data by scatter plot to see relation of G3 in math and G3 in portuguese. For this we can use plt's scatter function.



```
In [20]:  math_df = pd.read_csv('student-mat.csv', delimiter=';')
          portuguese_df = pd.read_csv('student-por.csv', delimiter=';')
          both = pd.merge(math_df, portuguese_df,on=["school","sex","age","addres

          plt.figure(figsize=(10, 6))
          plt.scatter(both['G3_x'], both['G3_y'], color='purple', alpha=0.5)

          plt.title('Scatter Plot: Final Grades in Math vs. Final Grades in Portu
          plt.xlabel('Final Grade in Math (G3_math)')
          plt.ylabel('Final Grade in Portuguese (G3_portuguese)')

          plt.show()
```





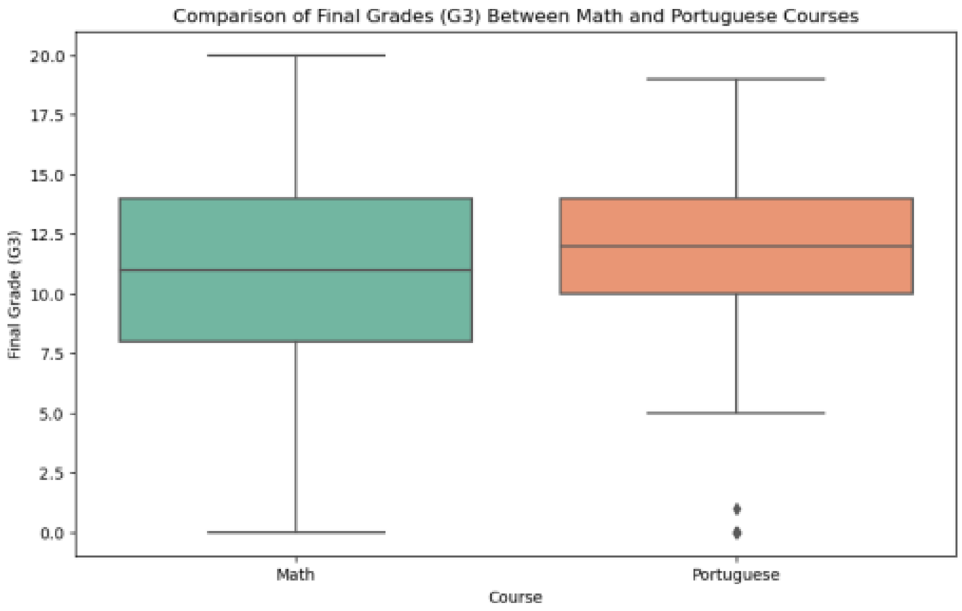
```
In [21]:  correlation_coefficient = both['G3_x'].corr(both['G3_y'])
          print(f"Correlation Coefficient between G3_math and G3_portuguese: {cor

          Correlation Coefficient between G3_math and G3_portuguese: 0.480349361
          9410264
```

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

To do side-by-side box plot we will add to math and portuguese datasets course column which will identify to which dataset data it belongs. And we will merge both datasets. Next to draw box plot we will use seaborn's boxplot function where x axis is course column and y axis is G3 column.

In [22]:

```
math_df['Course'] = 'Math'
portuguese_df['Course'] = 'Portuguese'

merged_df = pd.concat([math_df, portuguese_df])

plt.figure(figsize=(10, 6))
sns.boxplot(x='Course', y='G3', data=merged_df, palette='Set2')

plt.title('Comparison of Final Grades (G3) Between Math and Portuguese
plt.xlabel('Course')
plt.ylabel('Final Grade (G3)')

plt.show()
```

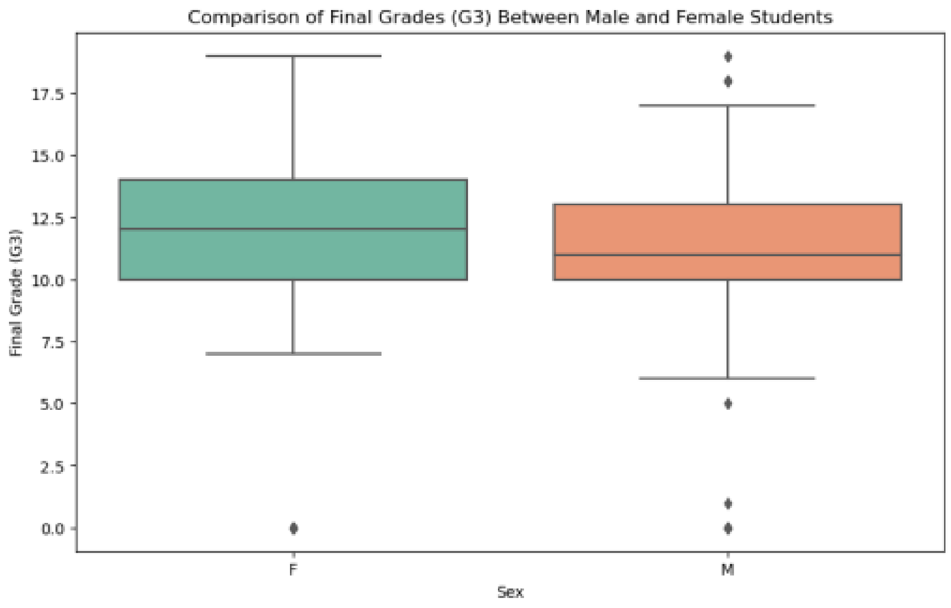


18) Is there a significant difference in the average final grades (G3) between male and female students? Conduct a two-sample t-test and visualize the results.

First we should take male and female students' grades separately. We can do so using filtering (comparing 'sex' column whether it is F or M). Next we will use ttest_ind function from scipy.stats to do two-sample t-test. As a result we see that there is significant difference. We will visualize data by boxplot function from seaborn library.

```
In [23]:  male_grades = math_df[math_df['sex'] == 'M']['G3']
          female_grades = math_df[math_df['sex'] == 'F']['G3']

          t_statistic, p_value = ttest_ind(male_grades, female_grades)

          print(f"T-Statistic: {t_statistic}")
          print(f"P-Value: {p_value}")

          if p_value < 0.05:
              print("\nThere is a significant difference in average final grades
          else:
              print("\nThere is no significant difference in average final grades

          plt.figure(figsize=(10, 6))
          sns.boxplot(x='sex', y='G3', data=df, palette='Set2')

          plt.title('Comparison of Final Grades (G3) Between Male and Female Stud
          plt.xlabel('Sex')
          plt.ylabel('Final Grade (G3)')

          plt.show()
```

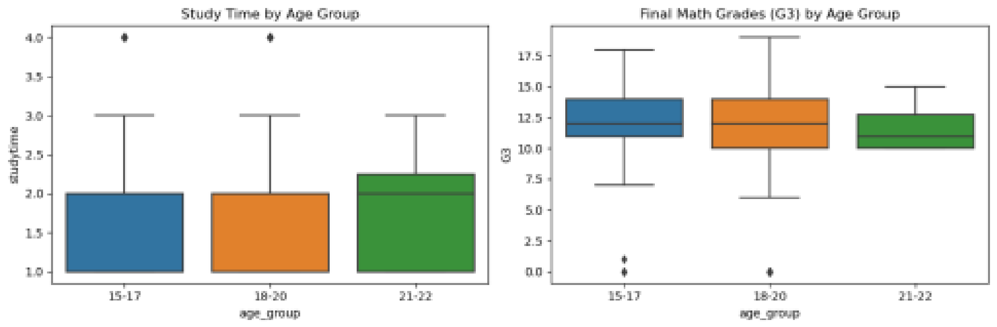
```
T-Statistic: 2.061992815503971
P-Value: 0.039865332341527636

There is a significant difference in average final grades between male
and female students.
```



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)?

To create new variable that categorizes students by age we will use pandas' cut function. First we will identify bins which will be range of age. We create age_bins variable with 15, 17, 20, 22 elements such that it will categorize it by 15-17, 18-20, 21-22. Then we visualize results by seaborn's boxplot function.

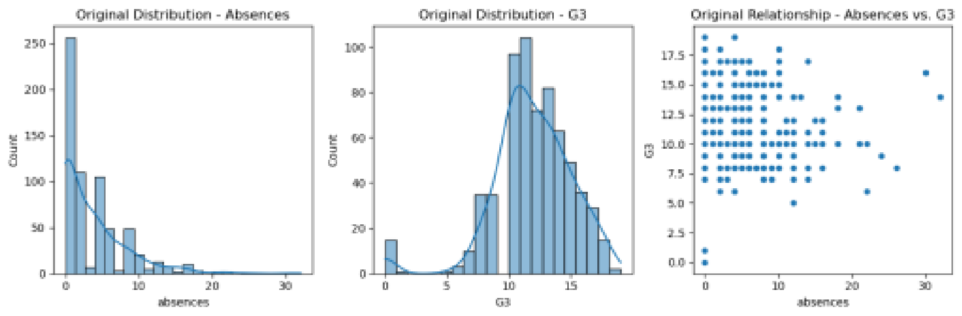
```
In [36]:  age_bins = [15, 17, 20, 22]
          age_labels = ['15-17', '18-20', '21-22']

          df['age_group'] = pd.cut(df['age'], bins=age_bins, labels=age_labels, r

          plt.figure(figsize=(12, 4))

          plt.subplot(1, 2, 1)
          sns.boxplot(x='age_group', y='studytime', data=df)
          plt.title('Study Time by Age Group')

          plt.subplot(1, 2, 2)
          sns.boxplot(x='age_group', y='G3', data=df)
          plt.title('Final Math Grades (G3) by Age Group')

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



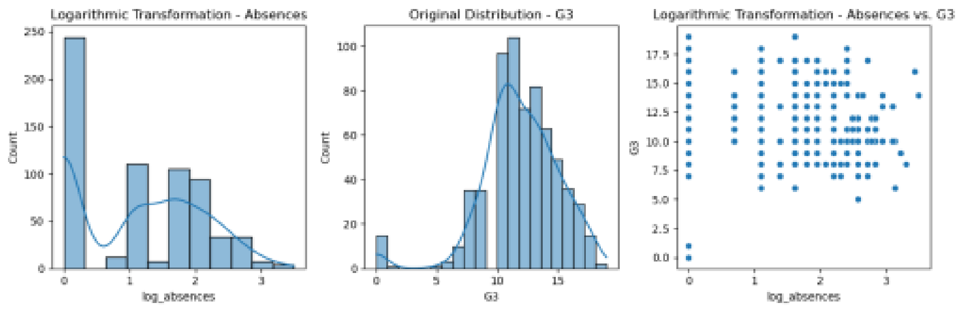
20) Apply a mathematical transformation, such as logarithm or square root, to the number of school absences (absences). How does this transformation impact the distribution of absences and its relationship with final math grades (G3)?

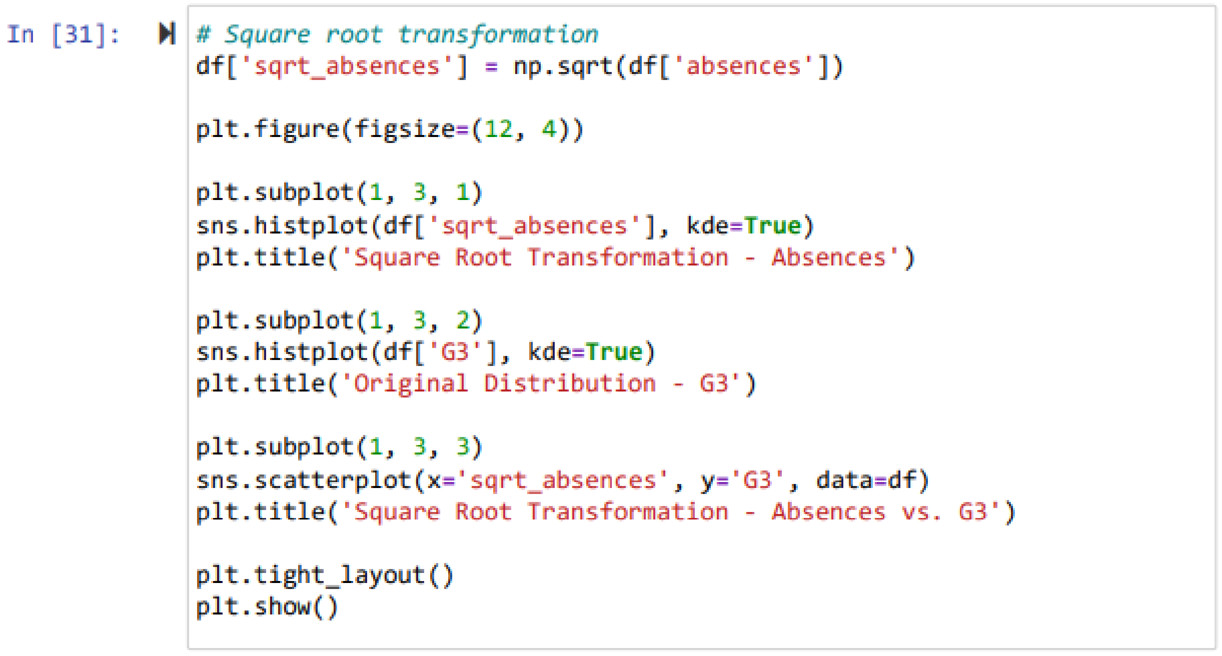
First we show original distribution of absences, G3 and its relationship by seaborn's histplot and scatterplot functions. To apply logarithm we will use numpy's log1p function and for square root sqrt function. Next we will visualize the updates absences by same scheme.

```
In [25]:  plt.figure(figsize=(12, 4))

          plt.subplot(1, 3, 1)
          sns.histplot(df['absences'], kde=True)
          plt.title('Original Distribution - Absences')

          plt.subplot(1, 3, 2)
          sns.histplot(df['G3'], kde=True)
          plt.title('Original Distribution - G3')

          plt.subplot(1, 3, 3)
          sns.scatterplot(x='absences', y='G3', data=df)
          plt.title('Original Relationship - Absences vs. G3')

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

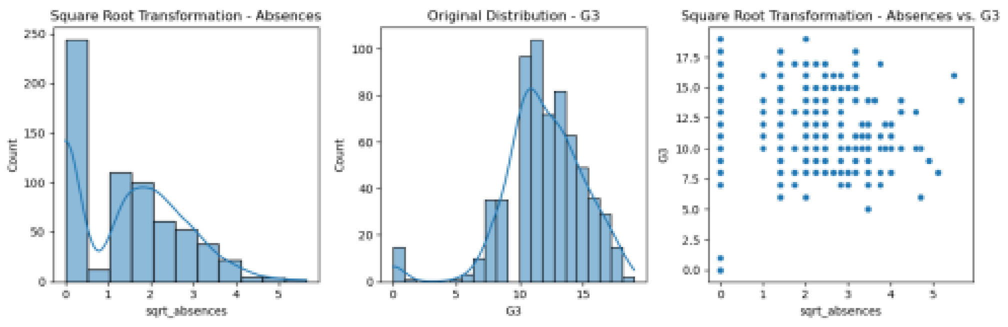


```
In [29]:  df['log_absences'] = np.log1p(df['absences'])

          plt.figure(figsize=(12, 4))

          plt.subplot(1, 3, 1)
          sns.histplot(df['log_absences'], kde=True)
          plt.title('Logarithmic Transformation - Absences')

          plt.subplot(1, 3, 2)
          sns.histplot(df['G3'], kde=True)
          plt.title('Original Distribution - G3')

          plt.subplot(1, 3, 3)
          sns.scatterplot(x='log_absences', y='G3', data=df)
          plt.title('Logarithmic Transformation - Absences vs. G3')

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





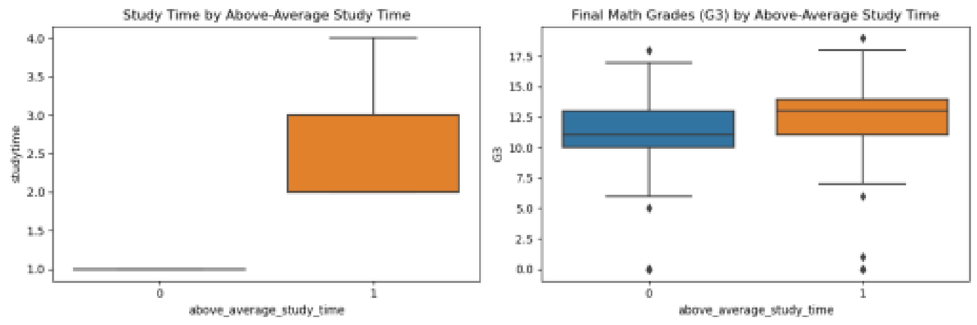
```
In [31]:  # Square root transformation
          df['sqrt_absences'] = np.sqrt(df['absences'])

          plt.figure(figsize=(12, 4))

          plt.subplot(1, 3, 1)
          sns.histplot(df['sqrt_absences'], kde=True)
          plt.title('Square Root Transformation - Absences')

          plt.subplot(1, 3, 2)
          sns.histplot(df['G3'], kde=True)
          plt.title('Original Distribution - G3')

          plt.subplot(1, 3, 3)
          sns.scatterplot(x='sqrt_absences', y='G3', data=df)
          plt.title('Square Root Transformation - Absences vs. G3')

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



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)?

First we identify average weekly study time by mean function. Next we create new column above_average_study_time by comparing average to each value. Next we create box plot by seaborn's boxplot function in which x axis is above_average_study_time and y axis is G3. As result we can see that students who studied above average has higher final grade than others.

```
In [44]:  average_study_time = df['studytime'].mean()

          df['above_average_study_time'] = (df['studytime'] > average_study_time)

          plt.figure(figsize=(12, 4))

          plt.subplot(1, 2, 1)
          sns.boxplot(x='above_average_study_time', y='studytime', data=df)
          plt.title('Study Time by Above-Average Study Time')

          plt.subplot(1, 2, 2)
          sns.boxplot(x='above_average_study_time', y='G3', data=df)
          plt.title('Final Math Grades (G3) by Above-Average Study Time')

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

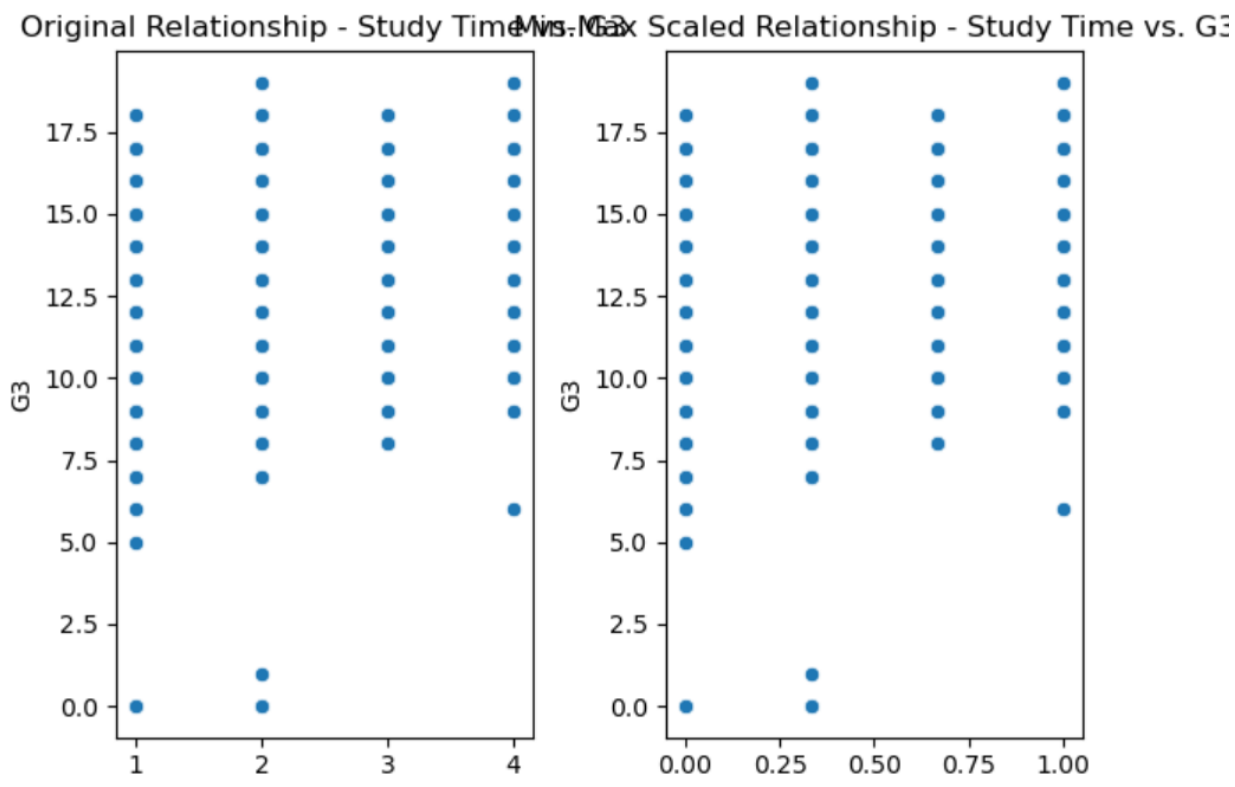


22) Apply feature scaling (e.g., Min-Max scaling or standardization) to numeric variables like age, absences, and study time. How does this scaling affect the relationships between these variables and math grades (G3)?

We use MinMaxScaler and StandardScaler from sklearn.preprocessing to transform all data to one scale(usually from 0 to 1). It helps to correctly compare data and visualize it correctly.

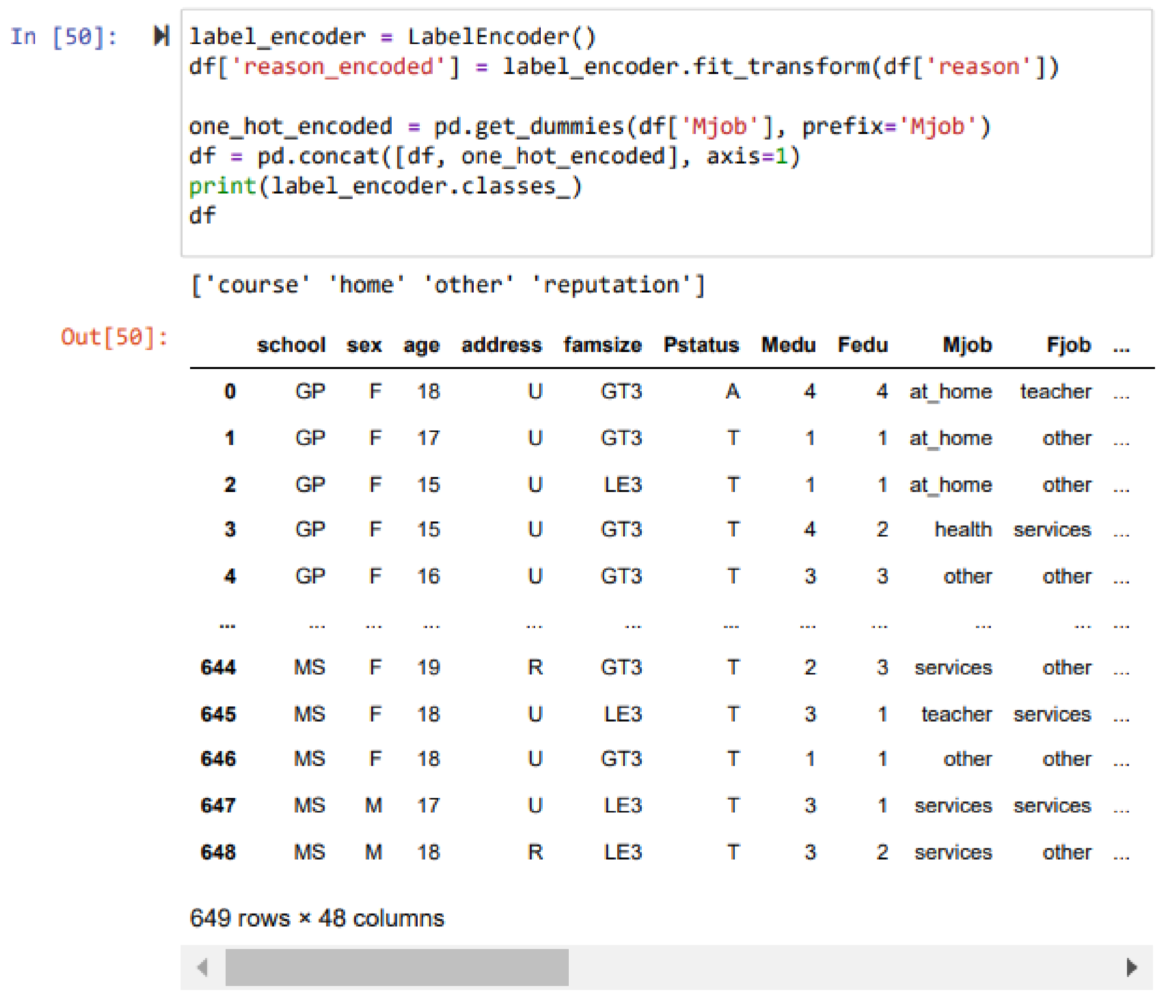
22nd question

```
In [61]:  numeric_vars = ['age', 'absences', 'studytime']

          minmax_scaler = MinMaxScaler()
          data_minmax_scaled = pd.DataFrame(minmax_scaler.fit_transform(df[numeric_vars]), columns=numeric_vars)
          data_minmax_scaled['G3'] = df['G3']

          standard_scaler = StandardScaler()
          data_standard_scaled = pd.DataFrame(standard_scaler.fit_transform(df[numeric_vars]), columns=numeric_vars)
          data_standard_scaled['G3'] = df['G3']


          plt.subplot(1, 2, 1)
          sns.scatterplot(x='studytime', y='G3', data=df)
          plt.title('Original Relationship - Study Time vs. G3')

          plt.subplot(1, 2, 2)
          sns.scatterplot(x='studytime', y='G3', data=data_minmax_scaled)
          plt.title('Min-Max Scaled Relationship - Study Time vs. G3')

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



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?

First we import LabelEncoder from sklearn.preprocessing. By label encoder we transform reason column into numeric format such that 0, 1, 2, 3 etc. Next one-hot encoding we do by pandas' get_dummies function which creates for every category new column with 0 or 1 value.



In [50]:
```
label_encoder = LabelEncoder()
df['reason_encoded'] = label_encoder.fit_transform(df['reason'])

one_hot_encoded = pd.get_dummies(df['Mjob'], prefix='Mjob')
df = pd.concat([df, one_hot_encoded], axis=1)
print(label_encoder.classes_)
df
```

```
['course' 'home' 'other' 'reputation']
```

Out[50]:

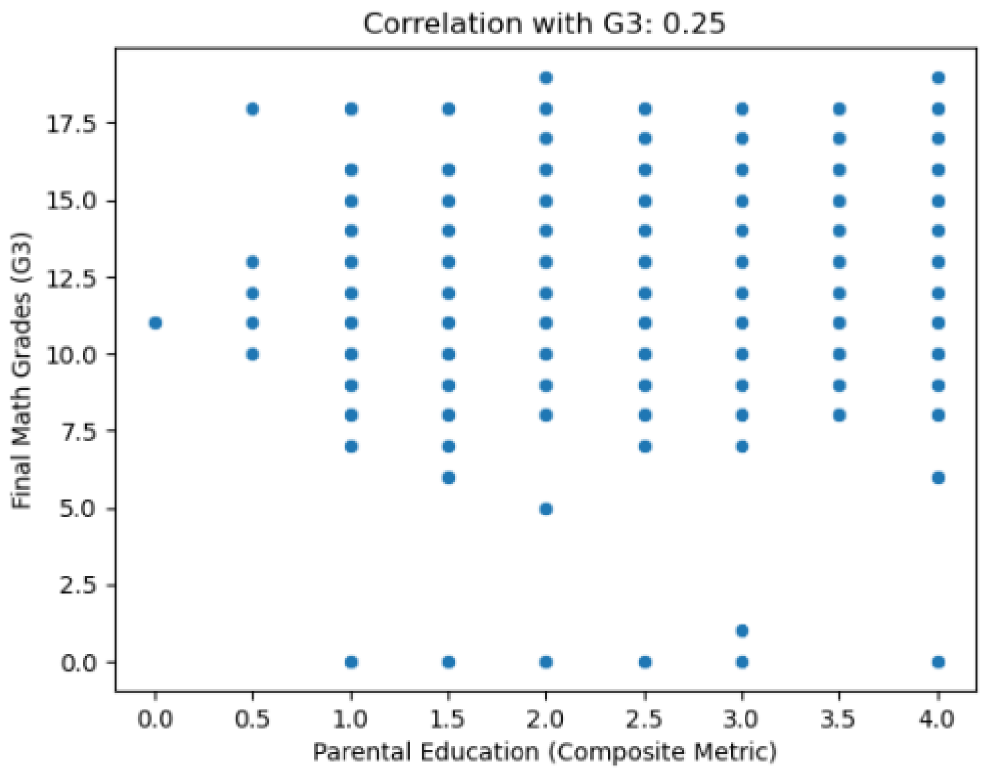
	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...
1	GP	F	17	U	GT3	T	1	1	at_home	other	...
2	GP	F	15	U	LE3	T	1	1	at_home	other	...
3	GP	F	15	U	GT3	T	4	2	health	services	...
4	GP	F	16	U	GT3	T	3	3	other	other	...
...	...	...	...	...	...	...	...	...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

649 rows × 48 columns

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)?

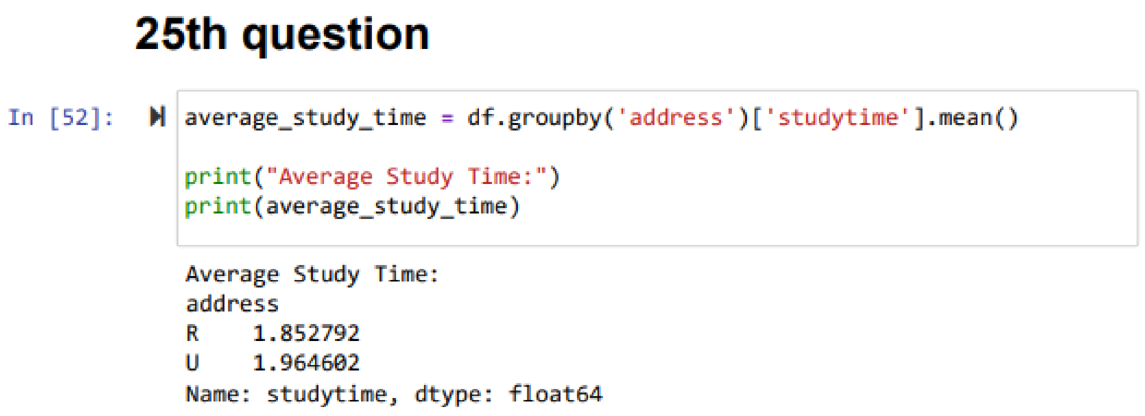
First we sum up medu and fedu then we divide it by 2. It is our Parental_Eductation column. Next find correlation matrix by corr function and take as argument Parental_Education and G3 columns. In the next step we visualize results by scatterplot.

```
In [51]:  df['Parental_Education'] = (df['Medu'] + df['Fedu']) / 2

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

          sns.scatterplot(x='Parental_Education', y='G3', data=df)
          plt.title(f'Correlation with G3: {correlation:.2f}')
          plt.xlabel('Parental Education (Composite Metric)')
          plt.ylabel('Final Math Grades (G3)')
          plt.show()
```



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?

First we should group dataframe by address column and apply mean function to studytime column for each group. By this way we find average weekly study time.



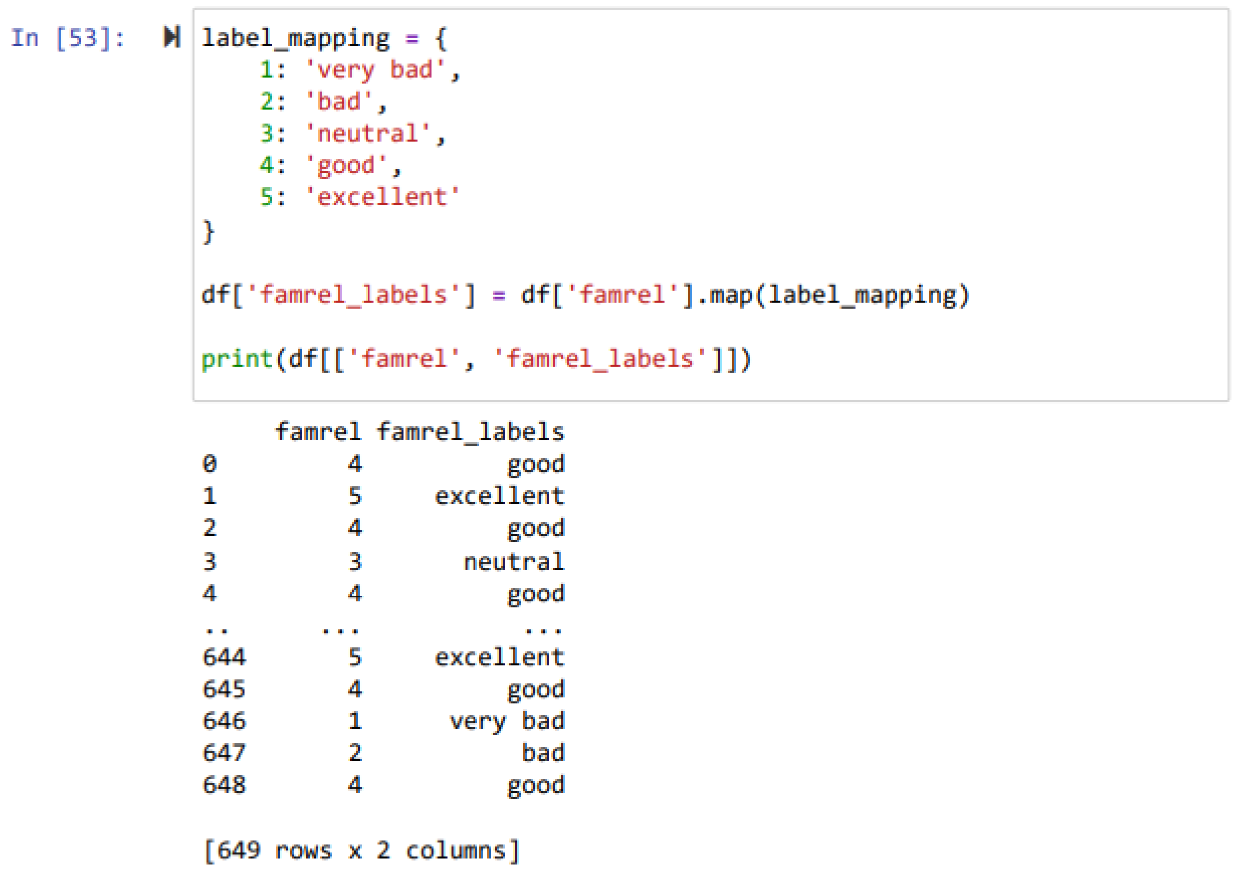
25th question

```
In [52]:  average_study_time = df.groupby('address')['studytime'].mean()

          print("Average Study Time:")
          print(average_study_time)
```

```
Average Study Time:
address
R    1.852792
U    1.964602
Name: studytime, dtype: float64
```

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?

To assign labels for numerical values we will use mapping. So first we create dict for each numeric value its corresponding category. Then with map function we apply these categories to famrel column creating new column (famrel_labels).

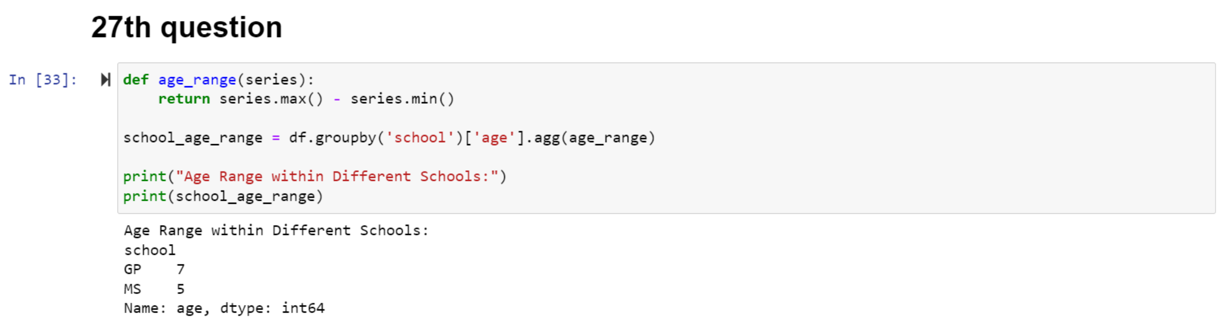


```
In [53]:  label_mapping = {
              1: 'very bad',
              2: 'bad',
              3: 'neutral',
              4: 'good',
              5: 'excellent'
          }

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

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

```
     famrel famrel_labels
0         4          good
1         5     excellent
2         4          good
3         3       neutral
4         4          good
..      ...           ...
644       5     excellent
645       4          good
646       1      very bad
647       2           bad
648       4          good

[649 rows x 2 columns]
```

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?

To create custom aggregation function we create function which returns some result using aggregation function. There we created age_range function that return difference between max and min values. Then we apply that function by agg function to grouped by school age column of the dataset.



27th question

```
In [33]:  def age_range(series):
              return series.max() - series.min()

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

          print("Age Range within Different Schools:")
          print(school_age_range)
```

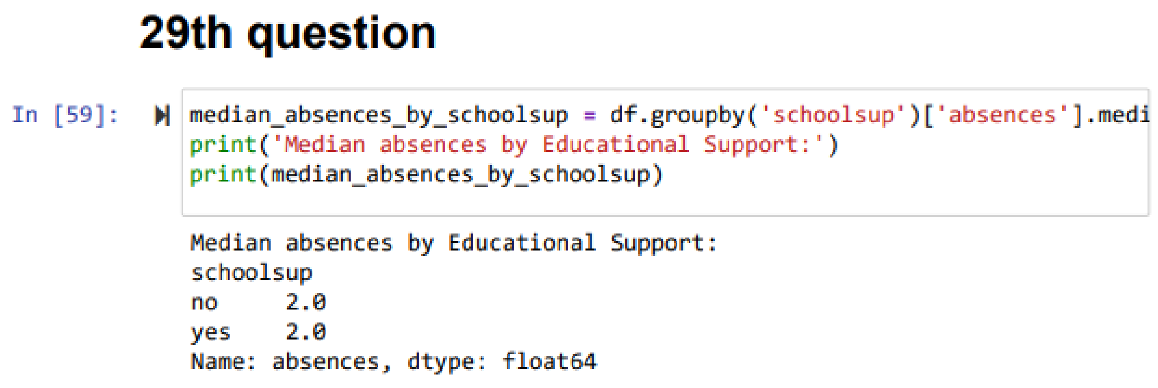
```
Age Range within Different Schools:
school
GP    7
MS    5
Name: age, dtype: int64
```

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 to do it because of date-related functions

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

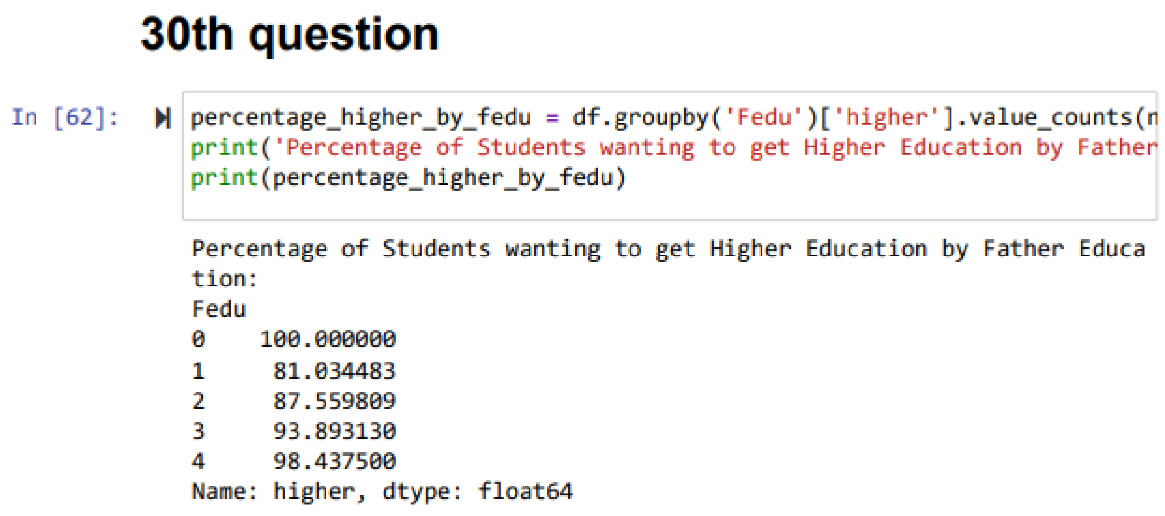
First we group dataset by schoolsup function and apply median function to absences column. From result we can see that both with school support and without have same (2.0) median number.



29th question

```
In [59]:  median_absences_by_schoolsup = df.groupby('schoolsup')['absences'].medi
          print('Median absences by Educational Support:')
          print(median_absences_by_schoolsup)

          Median absences by Educational Support:
          schoolsup
          no      2.0
          yes     2.0
          Name: absences, dtype: float64
```

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

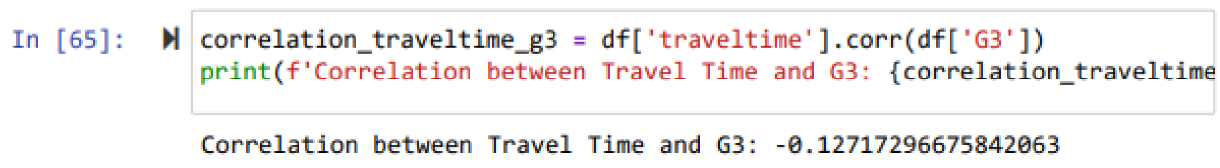


30th question

```
In [62]:  percentage_higher_by_fedu = df.groupby('Fedu')['higher'].value_counts(n
          print('Percentage of Students wanting to get Higher Education by Father
          print(percentage_higher_by_fedu)

          Percentage of Students wanting to get Higher Education by Father Educa
          tion:
          Fedu
          0    100.000000
          1     81.034483
          2     87.559809
          3     93.893130
          4     98.437500
          Name: higher, dtype: float64
```

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

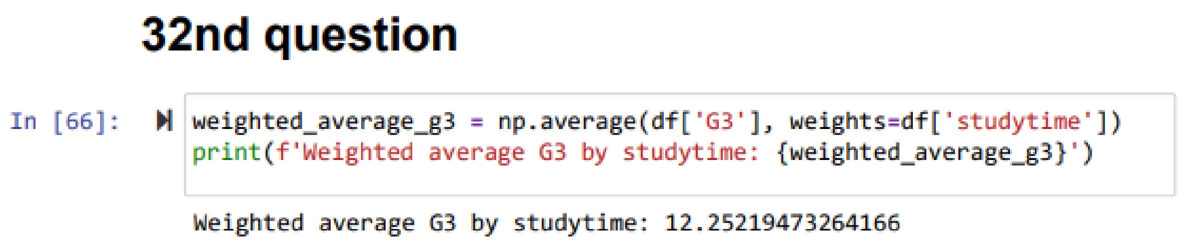
We find correlation by corr function to traveltime and G3 columns. Correlation is equal to -0.127.



```
In [65]:  correlation_traveltime_g3 = df['traveltime'].corr(df['G3'])
          print(f'Correlation between Travel Time and G3: {correlation_traveltime

          Correlation between Travel Time and G3: -0.12717296675842063
```

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

To find weighted average of G3 using studytime we use numpy's average function taking as arguments G3 column and weights as studytime..

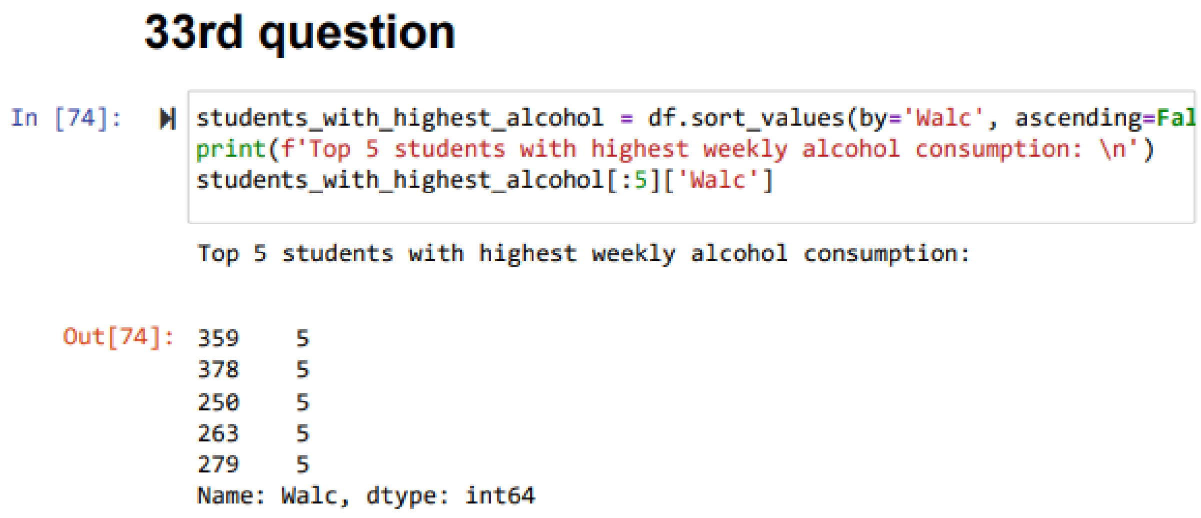


32nd question

```
In [66]:  weighted_average_g3 = np.average(df['G3'], weights=df['studytime'])
          print(f'Weighted average G3 by studytime: {weighted_average_g3}')

          Weighted average G3 by studytime: 12.25219473264166
```

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

To find student with the highest walc we should sort values by sort_values function. As a result we can see that highest walc is 5.

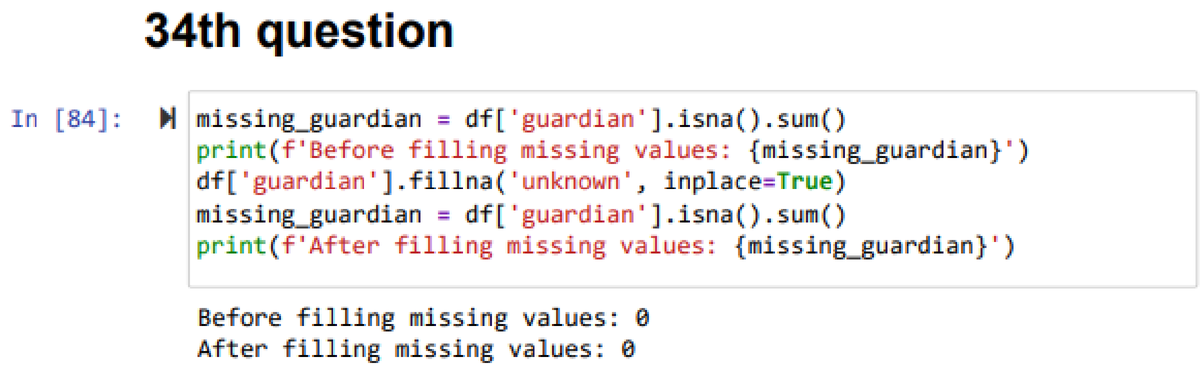


33rd question

```
In [74]:  students_with_highest_alcohol = df.sort_values(by='Walc', ascending=Fal
          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[74]:  359    5
          378    5
          250    5
          263    5
          279    5
          Name: Walc, dtype: int64
```

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

To replace missing values we use fillna function taking as argument ‘unknown’ to replace missing values.

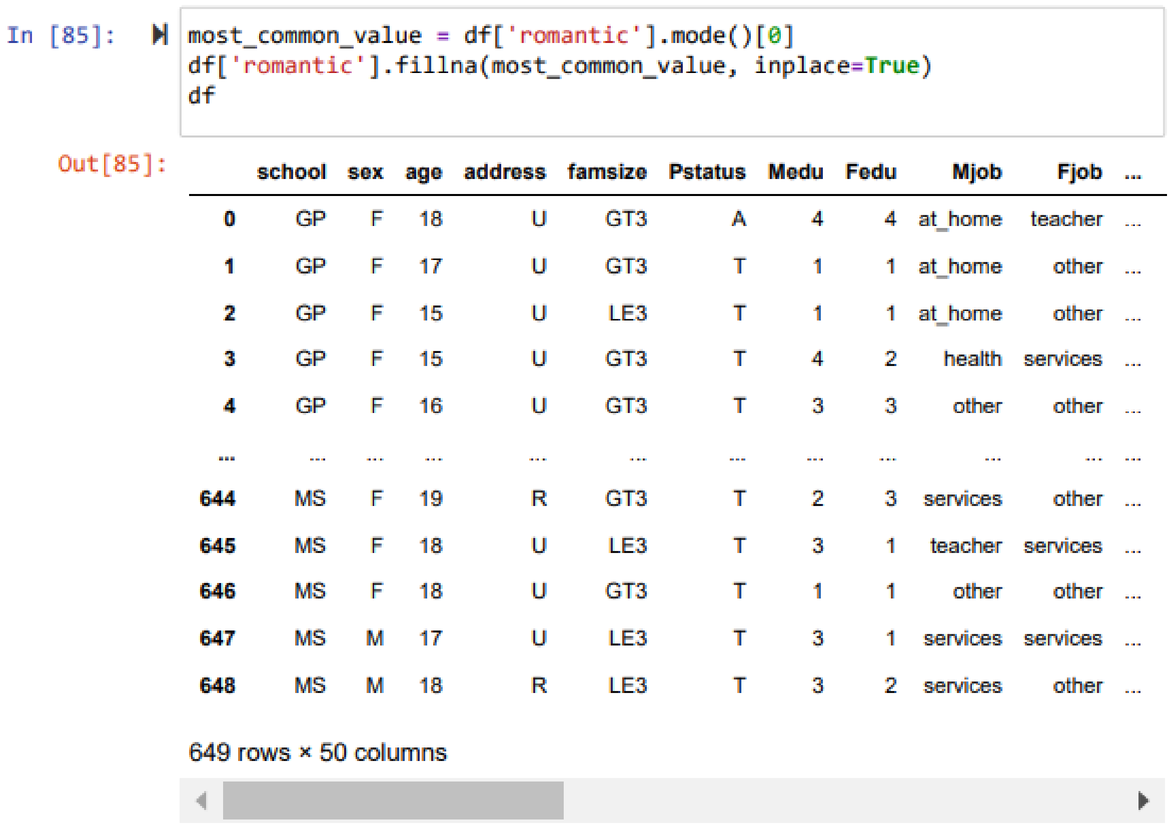


34th question

```
In [84]:  missing_guardian = df['guardian'].isna().sum()
          print(f'Before filling missing values: {missing_guardian}')
          df['guardian'].fillna('unknown', inplace=True)
          missing_guardian = df['guardian'].isna().sum()
          print(f'After filling missing values: {missing_guardian}')

          Before filling missing values: 0
          After filling missing values: 0
```

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

To fill missing values in the dataframe we use fillna function taking as argument mode of romantic column.



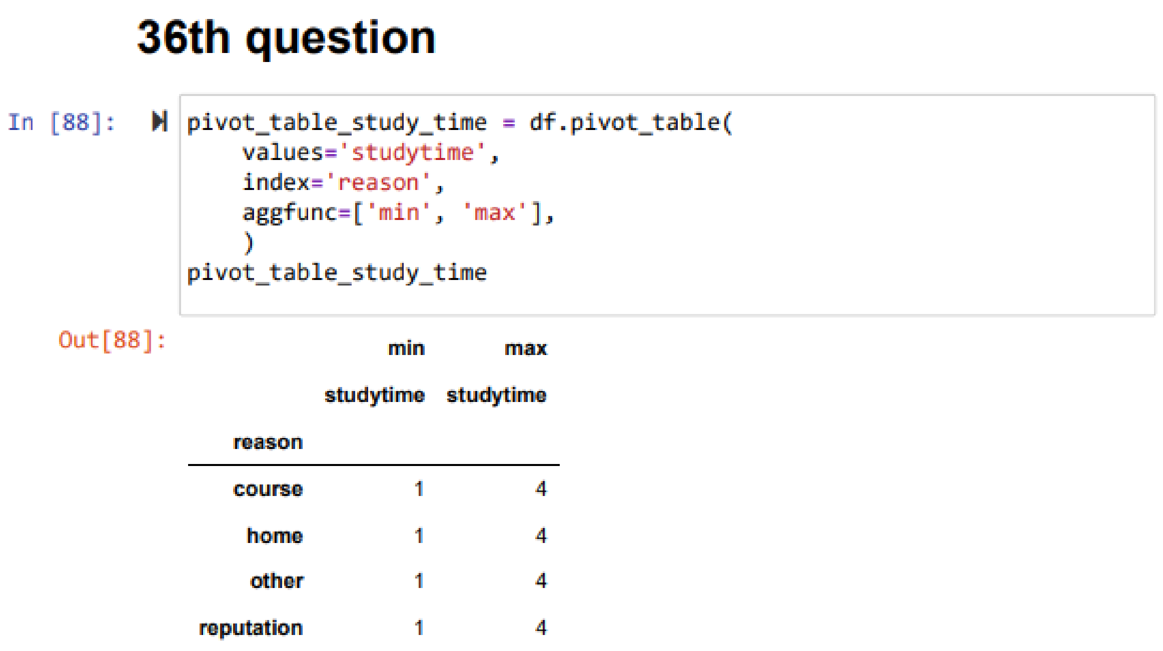
```
In [85]:  most_common_value = df['romantic'].mode()[0]
          df['romantic'].fillna(most_common_value, inplace=True)
          df

Out[85]:
```

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

649 rows × 50 columns

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

To create pivot table we use dataframe's pivot_table function taking in arguments studytime as values, reason as index and min, max as aggfunc. Min and max means aggregation functions min and max respectively.



36th question

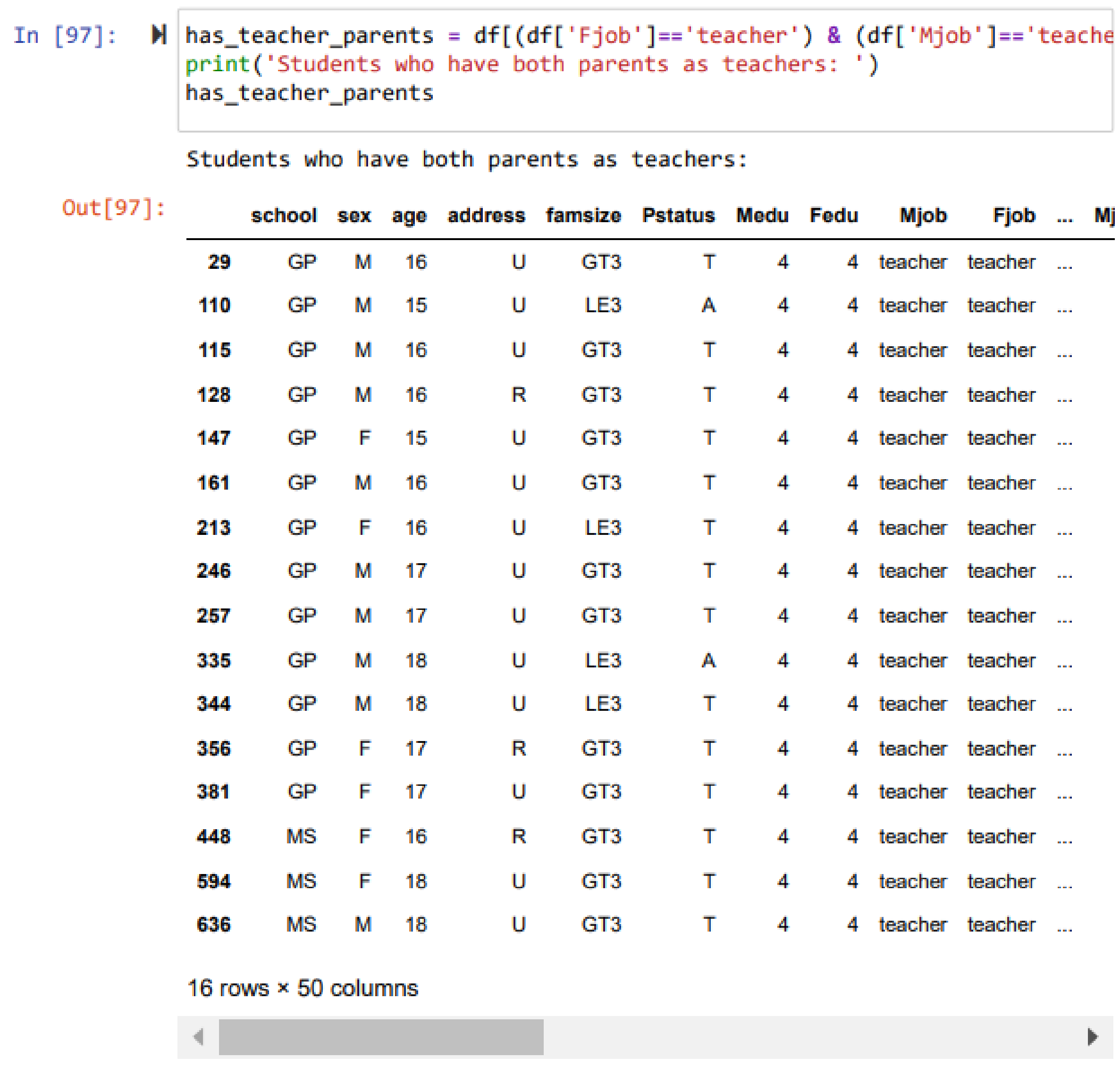
In [88]:
```
pivot_table_study_time = df.pivot_table(
    values='studytime',
    index='reason',
    aggfunc=['min', 'max'],
    )
pivot_table_study_time
```

Out[88]:

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

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

To check if a student has ‘teacher’ as both mother’s and father’s job we use filtering. We compare whether fjob is equal to teacher and if mjob is equal to teacher. As a result there is only 16 such student.



```
In [97]:  has_teacher_parents = df[(df['Fjob']=='teacher') & (df['Mjob']=='teache
          print('Students who have both parents as teachers: ')
          has_teacher_parents
```

```
Students who have both parents as teachers:
```

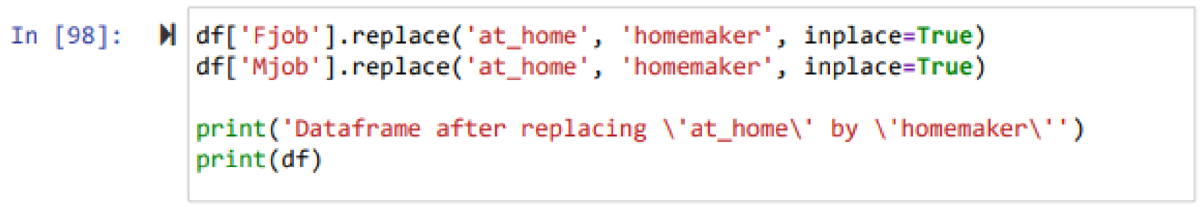
Out[97]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	Mj
29	GP	M	16	U	GT3	T	4	4	teacher	teacher	...	
110	GP	M	15	U	LE3	A	4	4	teacher	teacher	...	
115	GP	M	16	U	GT3	T	4	4	teacher	teacher	...	
128	GP	M	16	R	GT3	T	4	4	teacher	teacher	...	
147	GP	F	15	U	GT3	T	4	4	teacher	teacher	...	
161	GP	M	16	U	GT3	T	4	4	teacher	teacher	...	
213	GP	F	16	U	LE3	T	4	4	teacher	teacher	...	
246	GP	M	17	U	GT3	T	4	4	teacher	teacher	...	
257	GP	M	17	U	GT3	T	4	4	teacher	teacher	...	
335	GP	M	18	U	LE3	A	4	4	teacher	teacher	...	
344	GP	M	18	U	LE3	T	4	4	teacher	teacher	...	
356	GP	F	17	R	GT3	T	4	4	teacher	teacher	...	
381	GP	F	17	U	GT3	T	4	4	teacher	teacher	...	
448	MS	F	16	R	GT3	T	4	4	teacher	teacher	...	
594	MS	F	18	U	GT3	T	4	4	teacher	teacher	...	
636	MS	M	18	U	GT3	T	4	4	teacher	teacher	...	

16 rows × 50 columns

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

To replace 'at_home' in the 'Mjob' and 'Fjob' columns with 'homemaker' we use replace function taking as the first argument ‘at_home’ and the second argument ‘homemaker’.



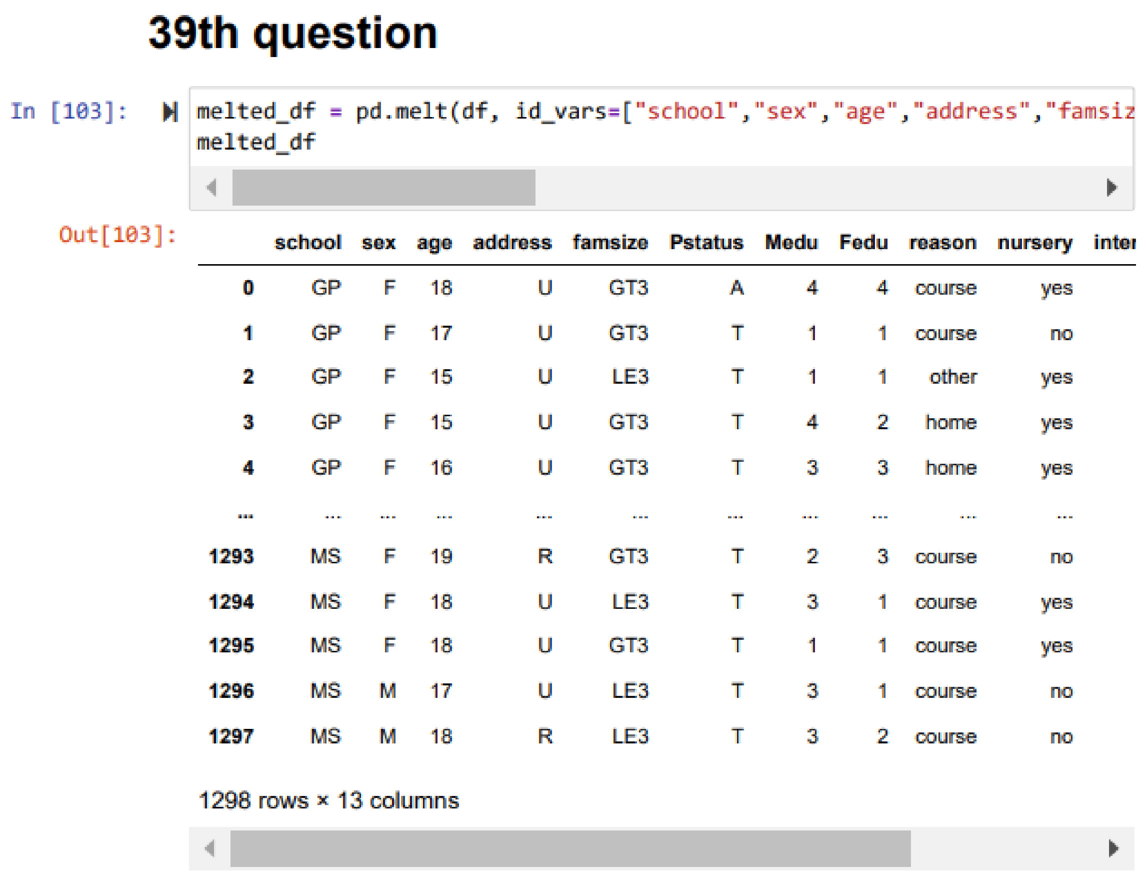
```
In [98]:  df['Fjob'].replace('at_home', 'homemaker', inplace=True)
          df['Mjob'].replace('at_home', 'homemaker', inplace=True)

          print('Dataframe after replacing \'at_home\' by \'homemaker\'')
          print(df)
```

```
Dataframe after replacing 'at_home' by 'homemaker'
    school sex  age address famsize Pstatus  Medu  Fedu       Mjob
Fjob  \
0       GP   F   18       U     GT3       A     4     4  homemaker   t
eacher
1       GP   F   17       U     GT3       T     1     1  homemaker
other
2       GP   F   15       U     LE3       T     1     1  homemaker
other
3       GP   F   15       U     GT3       T     4     2     health  se
rvices
4       GP   F   16       U     GT3       T     3     3      other
other
..     ...  ..  ...     ...     ...     ...   ...   ...        ...
...
644     MS   F   19       R     GT3       T     2     3   services
other
645     MS   F   18       U     LE3       T     3     1    teacher  se
rvices
646     MS   F   18       U     GT3       T     1     1      other
other
647     MS   M   17       U     LE3       T     3     1   services  se
rvices
648     MS   M   18       R     LE3       T     3     2   services
other

     ... Mjob_other Mjob_services Mjob_teacher  Mjob_at_home  Mjob_hea
lth  \
0    ...          0             0            0             1
0
1    ...          0             0            0             1
0
2    ...          0             0            0             1
0
3    ...          0             0            0             0
1
4    ...          1             0            0             0
0
..   ...        ...           ...          ...           ...
...
644  ...          0             1            0             0
0
645  ...          0             0            1             0
0
646  ...          1             0            0             0
0
647  ...          0             1            0             0
0
648  ...          0             1            0             0
0

    Mjob_other Mjob_services Mjob_teacher Parental_Education famrel_la
bels
0            0             0            0                4.0
good
1            0             0            0                1.0     excel
lent
2            0             0            0                1.0
good
```

```
4            1            0            0               3.0
good
..          ...          ...          ...              ...
...
644          0            1            0               2.5     excel
lent
645          0            0            1               2.0
good
646          1            0            0               1.0      very
bad
647          0            1            0               2.0
bad
648          0            1            0               2.5
good

[649 rows x 50 columns]
```

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

To melt the dataset to convert the 'Mjob' and 'Fjob' columns into a single column 'ParentJob' we use pandas' melt function taking as first argument dataframe and as id_vars we take primary key columns.



39th question

In [103]:
```
melted_df = pd.melt(df, id_vars=["school","sex","age","address","famsiz
melted_df
```

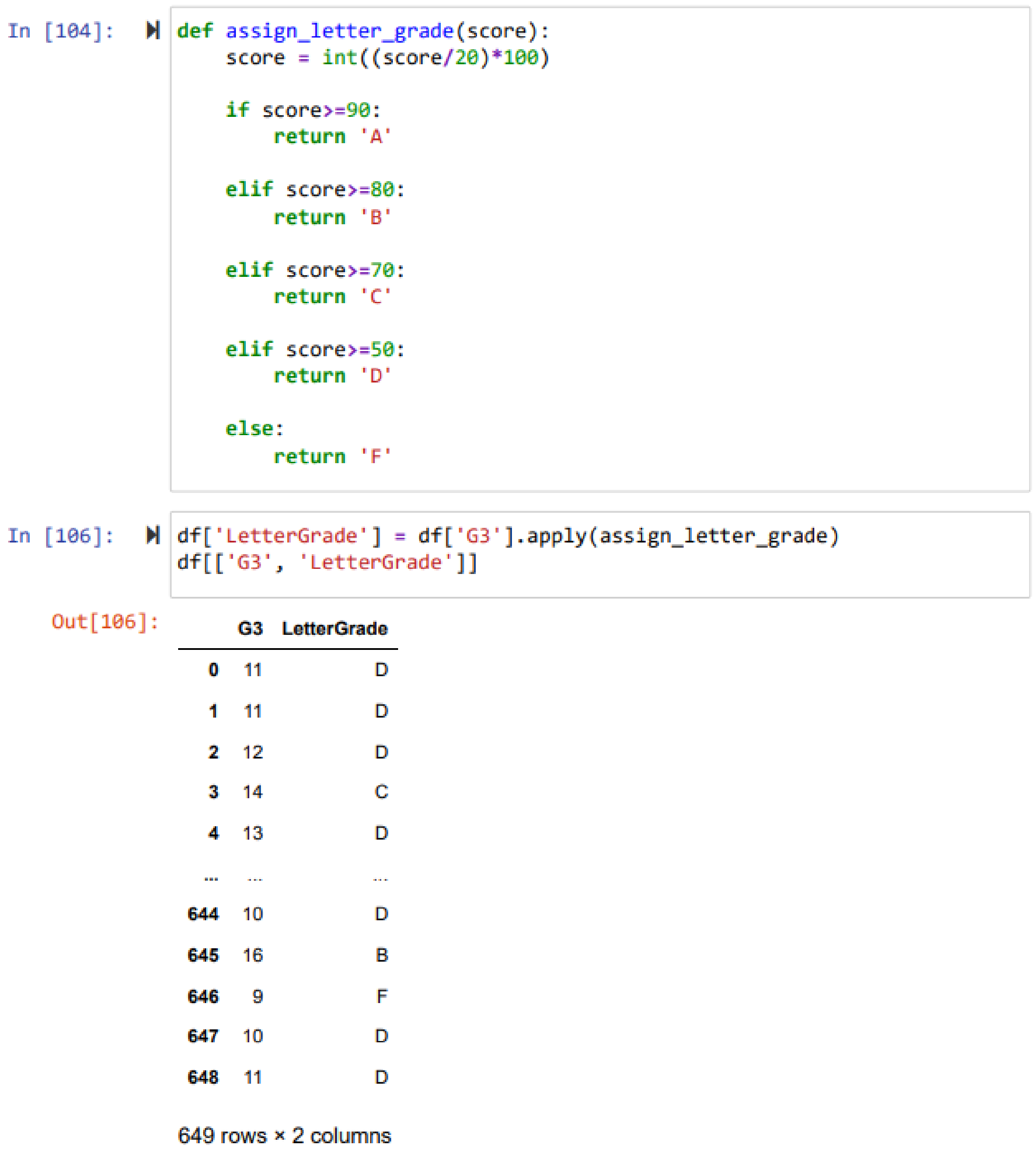
Out[103]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	reason	nursery	inter
0	GP	F	18	U	GT3	A	4	4	course	yes	
1	GP	F	17	U	GT3	T	1	1	course	no	
2	GP	F	15	U	LE3	T	1	1	other	yes	
3	GP	F	15	U	GT3	T	4	2	home	yes	
4	GP	F	16	U	GT3	T	3	3	home	yes	
...	...	...	...	...	...	...	...	...	...	...	
1293	MS	F	19	R	GT3	T	2	3	course	no	
1294	MS	F	18	U	LE3	T	3	1	course	yes	
1295	MS	F	18	U	GT3	T	1	1	course	yes	
1296	MS	M	17	U	LE3	T	3	1	course	no	
1297	MS	M	18	R	LE3	T	3	2	course	no	

1298 rows × 13 columns

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.

First we create function assign_letter_grade which takes G3 and finds its percentage out of 100. Then by if-else statements we return A, B, C, D or F letter. Then we apply function by dataframe's apply function creating new LetterGrade column.



In [104]:
```python
def assign_letter_grade(score):
    score = int((score/20)*100)

    if score>=90:
        return 'A'

    elif score>=80:
        return 'B'

    elif score>=70:
        return 'C'

    elif score>=50:
        return 'D'

    else:
        return 'F'
```

In [106]:
```python
df['LetterGrade'] = df['G3'].apply(assign_letter_grade)
df[['G3', 'LetterGrade']]
```

Out[106]:

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

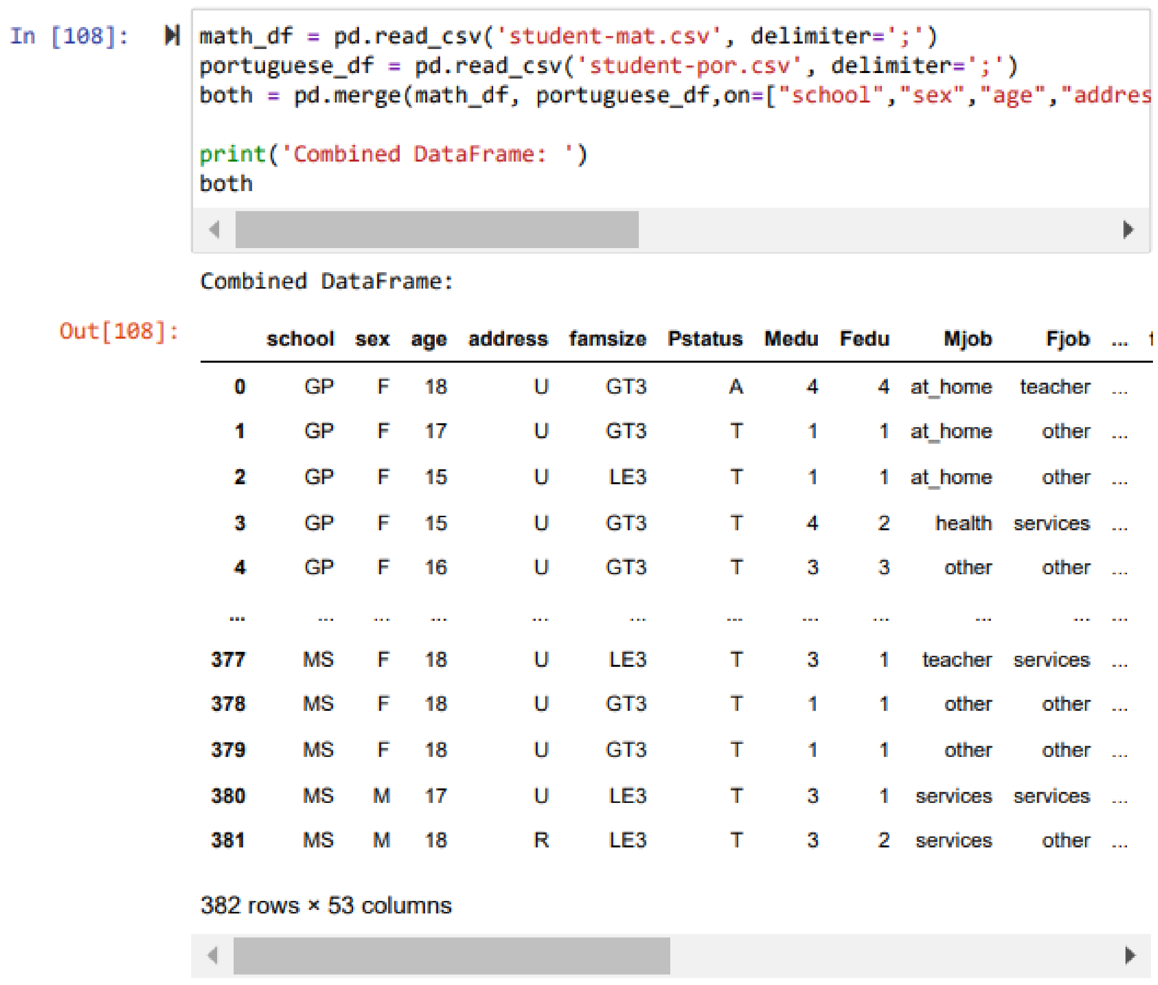
649 rows × 2 columns

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

No time series in the datasets

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

To find students who appear in both datasets we use pandas' merge function taking as argument on 'on' primary key columns. As the result there are 382 such students.



```
In [108]:  math_df = pd.read_csv('student-mat.csv', delimiter=';')
           portuguese_df = pd.read_csv('student-por.csv', delimiter=';')
           both = pd.merge(math_df, portuguese_df,on=["school","sex","age","addres

           print('Combined DataFrame: ')
           both
```

```
Combined DataFrame:
```

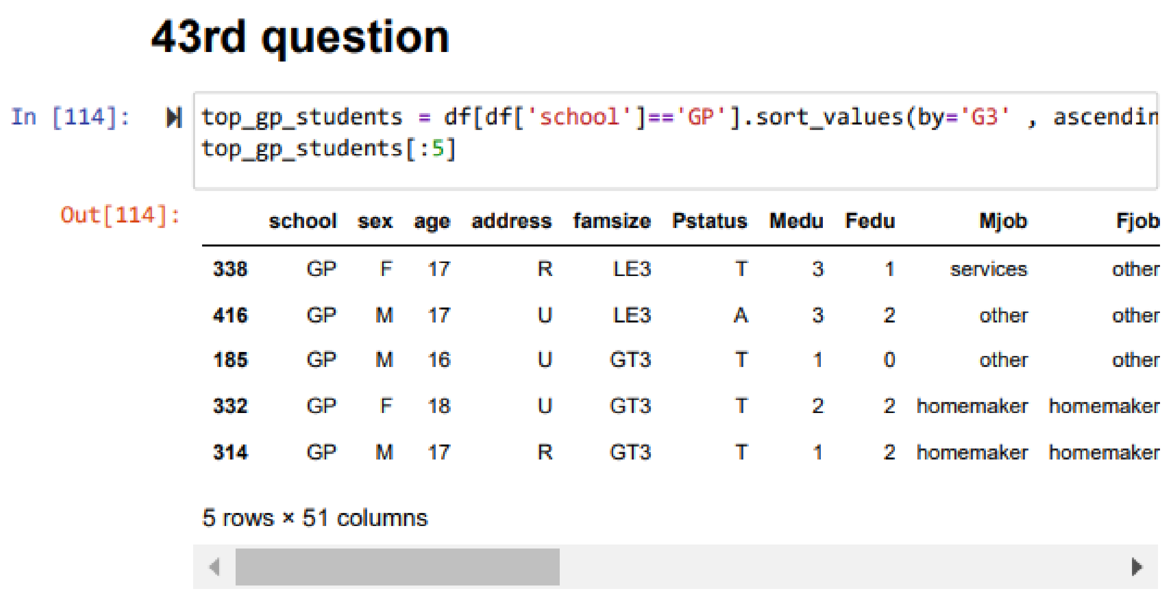
Out[108]:

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

382 rows × 53 columns

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

To find top 5 students with the highest final grades (G3) in the 'GP' school we first use filtering by school column and sort values by sort_values function by G3 column.



43rd question

```
In [114]:  top_gp_students = df[df['school']=='GP'].sort_values(by='G3' , ascendin
           top_gp_students[:5]
```

Out[114]:

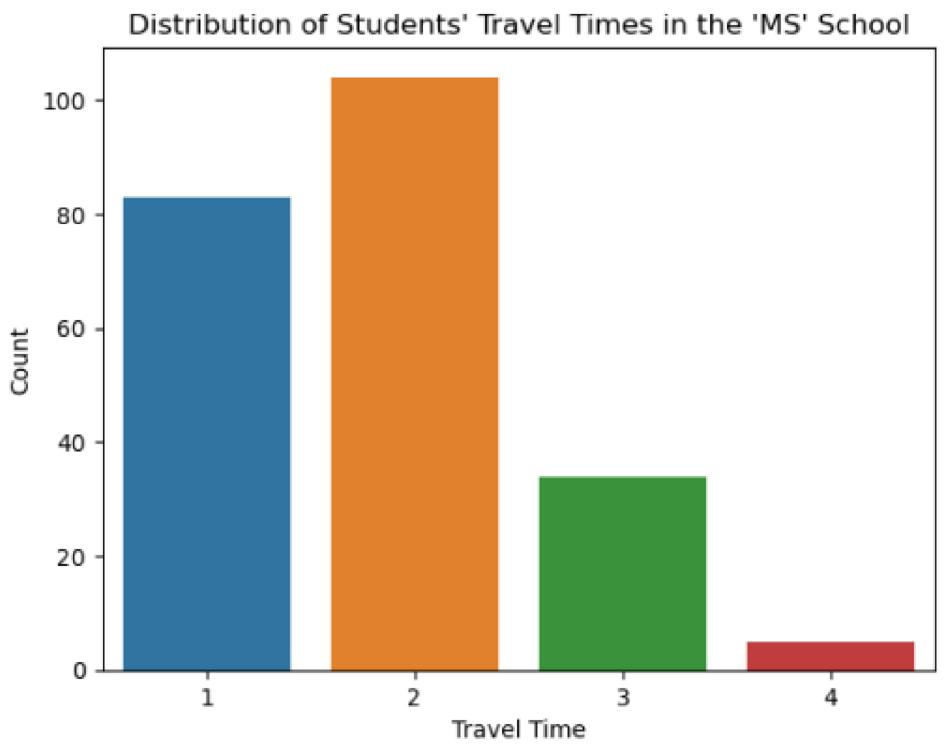
	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob
338	GP	F	17	R	LE3	T	3	1	services	other
416	GP	M	17	U	LE3	A	3	2	other	other
185	GP	M	16	U	GT3	T	1	0	other	other
332	GP	F	18	U	GT3	T	2	2	homemaker	homemaker
314	GP	M	17	R	GT3	T	1	2	homemaker	homemaker

5 rows × 51 columns

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

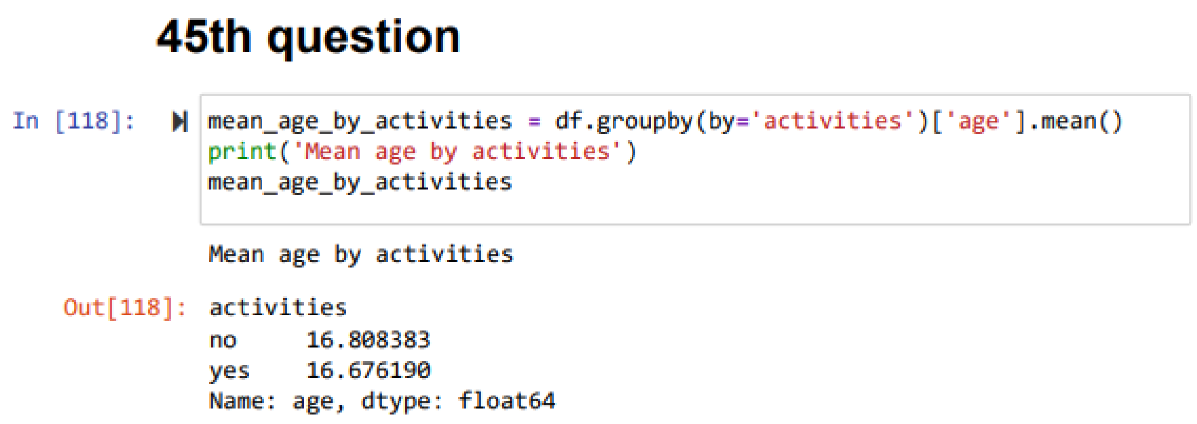
To create a bar chart showing the distribution of students' travel times (traveltime) in the 'MS' school we will use seaborn’s countplot function. It is just similar to bar chart. First we should filter data by school is equal to MS and apply it to countplot function. As a result there is bar chart showing distribution of students' travel times (traveltime) in the 'MS' school.

```
In [117]:  sns.countplot(x='traveltime', data=df[df['school']=='MS'])
           plt.title("Distribution of Students' Travel Times in the 'MS' School")
           plt.xlabel('Travel Time')
           plt.ylabel('Count')
           plt.show()
```



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

To compute the mean age of students who have extra-curricular activities (activities) and those who don't first we should group dataframe by activities column and apply mean function to age column. As a result they have barely same means(16.8, 16.6).

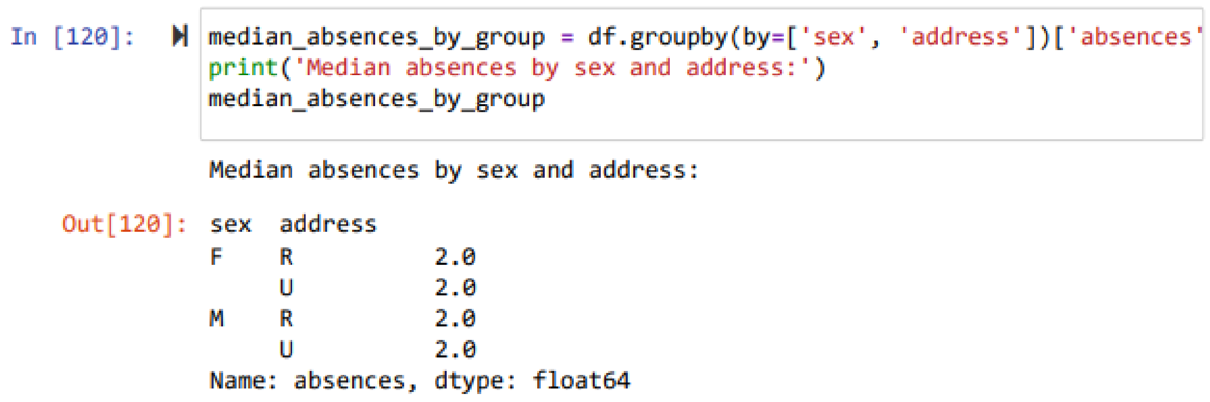


45th question

```
In [118]:  mean_age_by_activities = df.groupby(by='activities')['age'].mean()
           print('Mean age by activities')
           mean_age_by_activities

           Mean age by activities

Out[118]:  activities
           no      16.808383
           yes     16.676190
           Name: age, dtype: float64
```

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

First we should group the data by dataframe's groupby function taking by as 'sex', 'address' columns. Then we apply to grouped data's 'absences' column median function. As a result all group's medians are 2.0.



```
In [120]:  median_absences_by_group = df.groupby(by=['sex', 'address'])['absences'
           print('Median absences by sex and address:')
           median_absences_by_group

           Median absences by sex and address:

Out[120]:  sex  address
           F    R          2.0
                U          2.0
           M    R          2.0
                U          2.0
           Name: absences, dtype: float64
```

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

To Calculate the percentage of students who receive extra educational support (schoolsup) in the 'GP' school we should firstly filter dataframe by school column and apply to schoolsup column value_counts function with argument normalize=True in order to show percentage distribution.

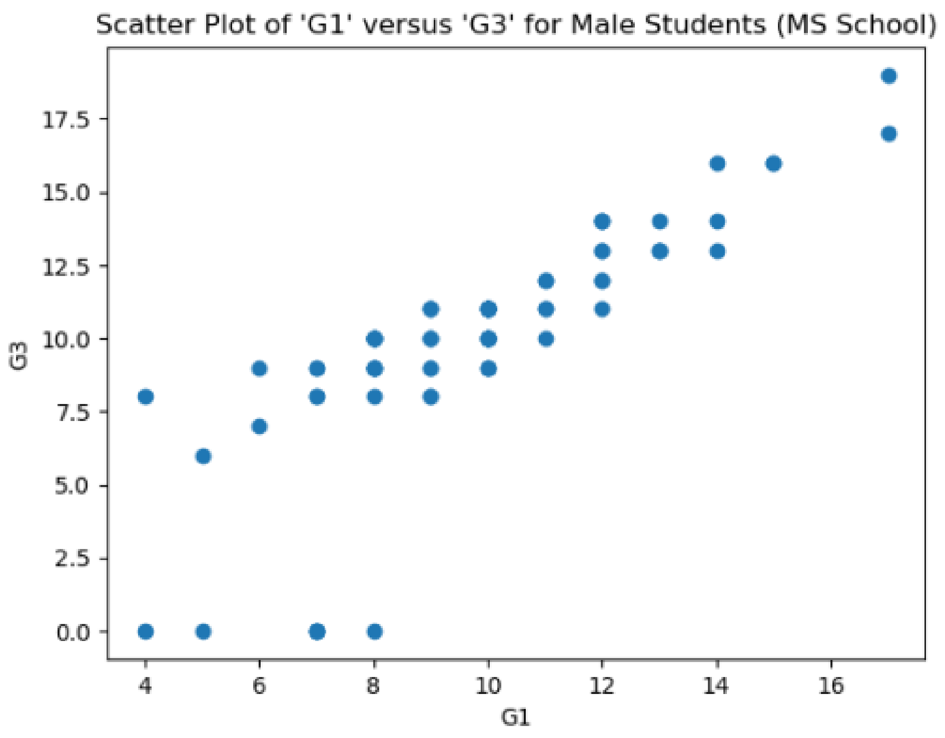
47th question

```
In [121]:  percentage_schoolsup_gp = df[df['school']=='GP']['schoolsup'].value_cou
            print('Percentage of school support in gp school: ')
            percentage_schoolsup_gp

Out[121]:  no     0.867612
           yes    0.132388
           Name: schoolsup, dtype: float64
```

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

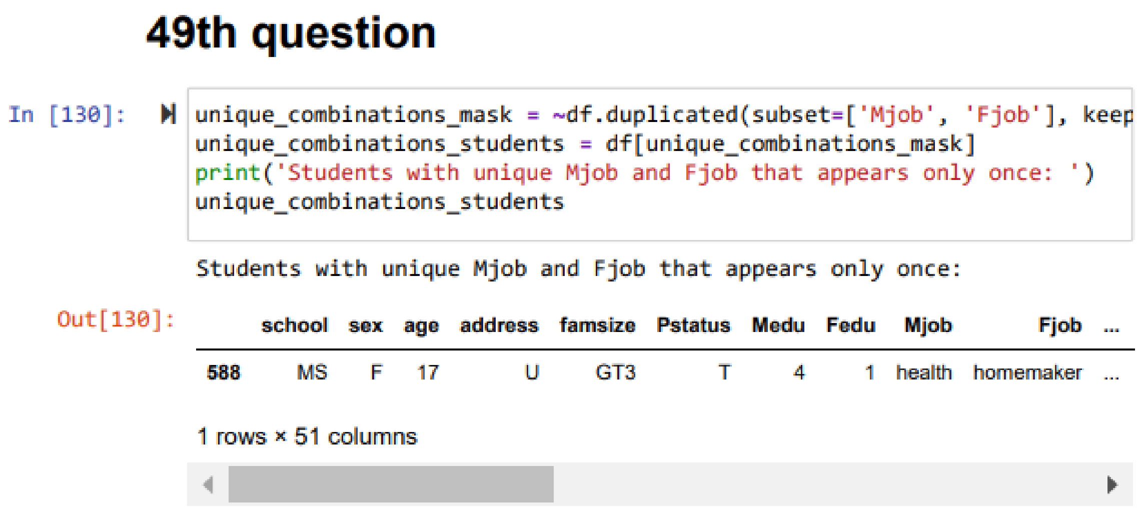
To create a scatter plot of 'G1' versus 'G3' for male students from the 'MS' school first we filter dataframe by ‘sex’ and ‘school’ columns and use pyplot’s scatter function taking as x axis G1 and as y axis G3. As a result there is scatter plot showing 'G1' versus 'G3' for male students from the 'MS' school.

```
In [127]:  male_ms_students_data = df[(df['sex']=='M') & (df['school']=='MS')]
           plt.scatter(male_ms_students_data['G1'], male_ms_students_data['G3'])
           plt.title("Scatter Plot of 'G1' versus 'G3' for Male Students (MS Schoo
           plt.xlabel('G1')
           plt.ylabel('G3')
           plt.show()
```



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

To identify students with a unique combination of 'Mjob' and 'Fjob' that appears only once in the dataset we use dataframe's duplicated function. It shows ids of all duplicate subsets. As subset we take mjob and fjob. We conversely need not duplicated so we apply negation (~). Next we get student with this id. This student has a unique combination of 'Mjob' and 'Fjob'.



49th question

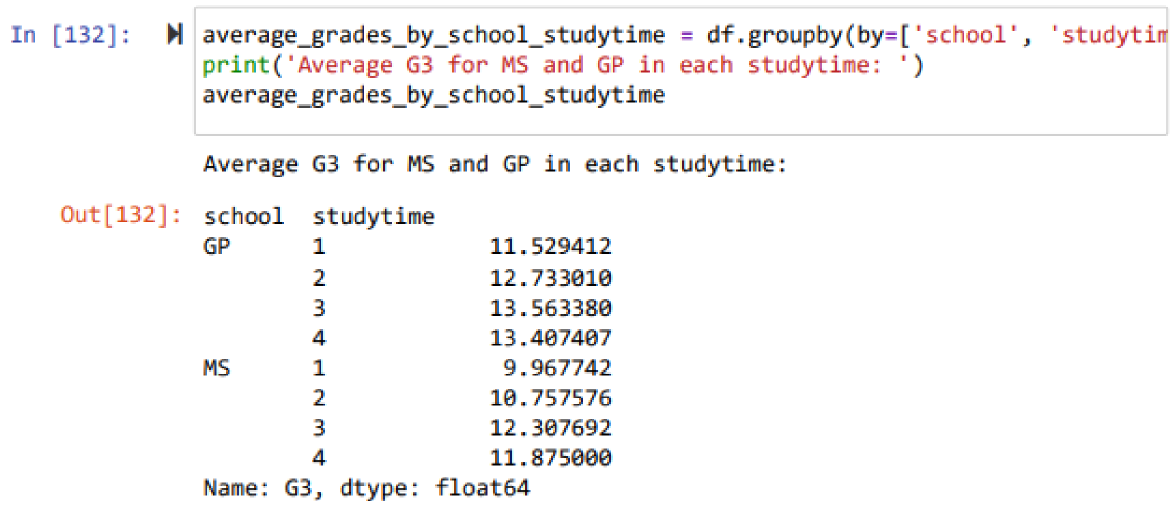
```
In [130]:  unique_combinations_mask = ~df.duplicated(subset=['Mjob', 'Fjob'], keep
           unique_combinations_students = df[unique_combinations_mask]
           print('Students with unique Mjob and Fjob that appears only once: ')
           unique_combinations_students

           Students with unique Mjob and Fjob that appears only once:
Out[130]:
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...
588	MS	F	17	U	GT3	T	4	1	health	homemaker	...

1 rows × 51 columns

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

First we should use groupby function taking by as school and studytime columns. Next we apply mean function to G3. As a result we get the average final grade (G3) for students from 'GP' and 'MS' schools in each 'studytime' category.



```
In [132]:  average_grades_by_school_studytime = df.groupby(by=['school', 'studytim
           print('Average G3 for MS and GP in each studytime: ')
           average_grades_by_school_studytime

           Average G3 for MS and GP in each studytime:
Out[132]:  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
```