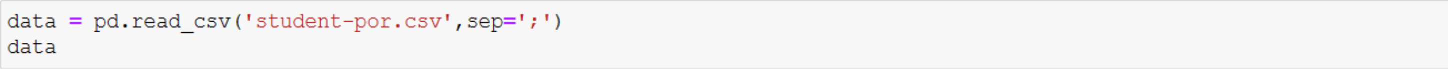
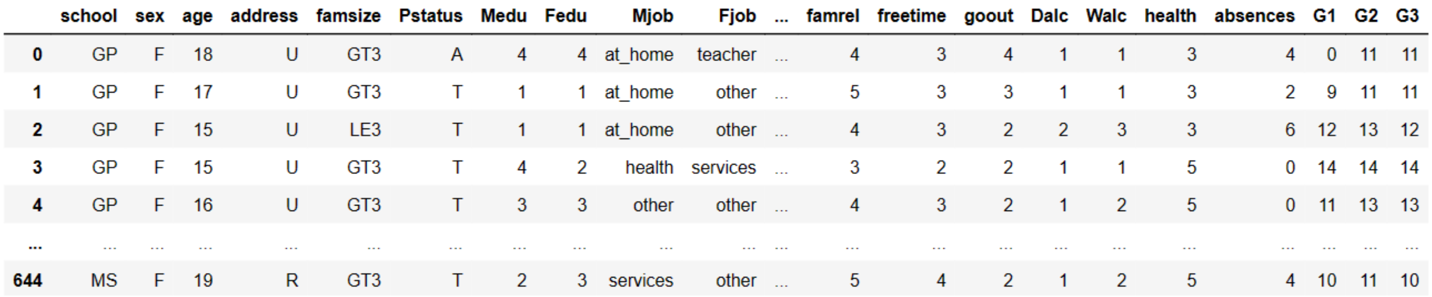
Nurai Abugalieva 210103331 /final report

Data : Reading csv file, using sep = ‘;’ to separate values



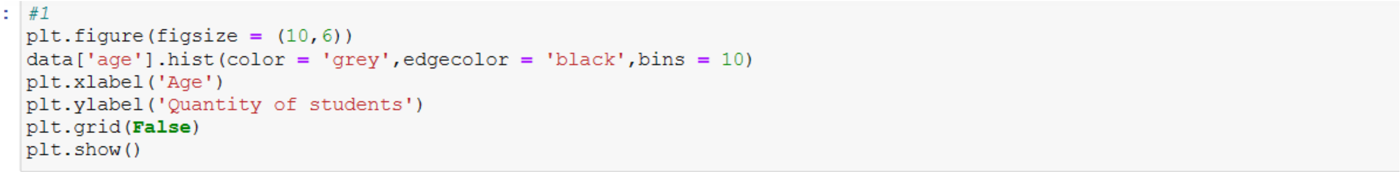
```
data = pd.read_csv('student-por.csv',sep=';')
data
```



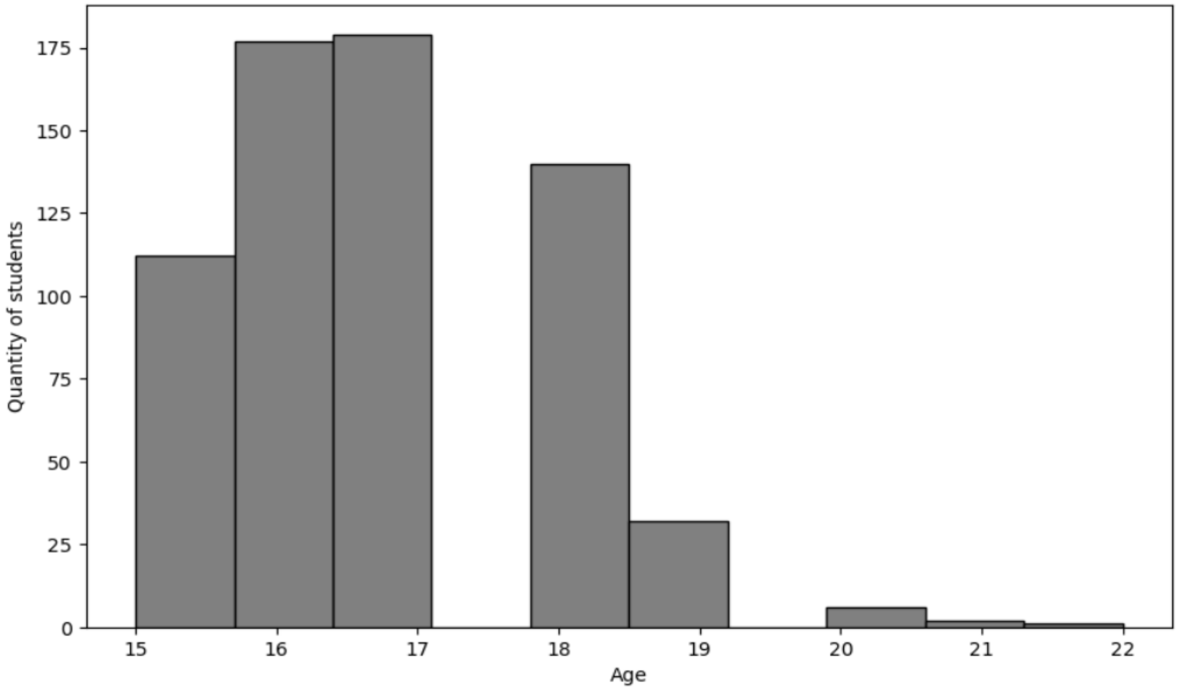
	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	4	0	11	11
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	2	9	11	11
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	6	12	13	12
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	0	14	14	14
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	0	11	13	13
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
644	MS	F	19	R	GT3	T	2	3	services	other	...	5	4	2	1	2	5	4	10	11	10

1)Question 1

This code draws histogram ,defines size,color,edgecolor,removes lines(grid),at the bottom writes ‘Age’,on the left side ‘Quantity of students’(label).The result returns histogram to show the distribution of ages of students:

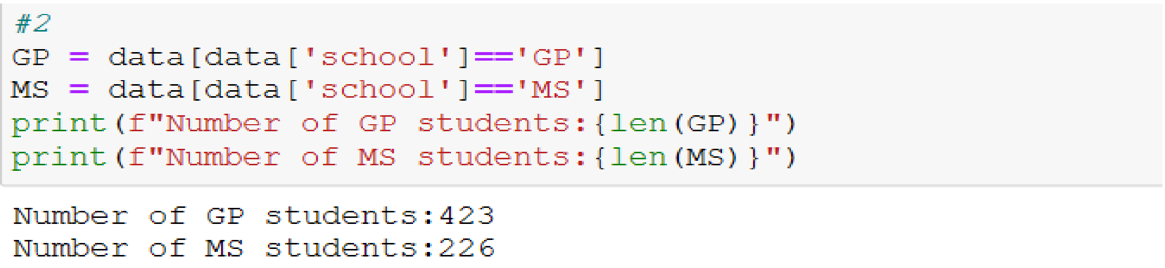


```
: #1
  plt.figure(figsize = (10,6))
  data['age'].hist(color = 'grey',edgecolor = 'black',bins = 10)
  plt.xlabel('Age')
  plt.ylabel('Quantity of students')
  plt.grid(False)
  plt.show()
```



2)Question 2

This code creates two new dataframes ,by checking data’s school values,then counts the number of these dataframes by len.The result returns number of students which belongs to these groups:

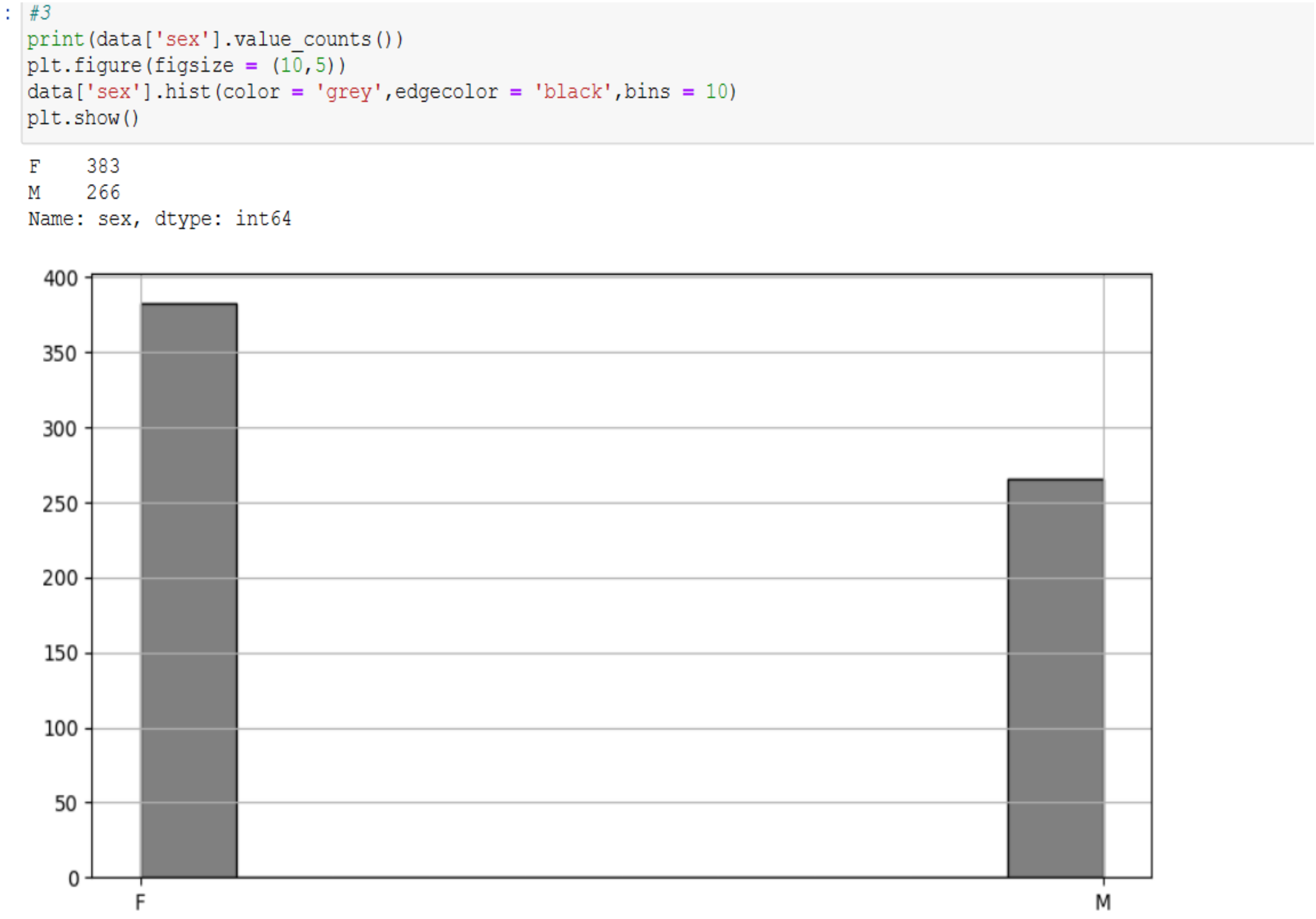


```
#2
GP = data[data['school']=='GP']
MS = data[data['school']=='MS']
print(f"Number of GP students:{len(GP)}")
print(f"Number of MS students:{len(MS)}")
```

```
Number of GP students:423
Number of MS students:226
```

3)Question 3

This code counts the number of gender by group ,draws histogram,by defining size, color, edgecolor.The result returns number of female and male students and histogram to show gender distribution

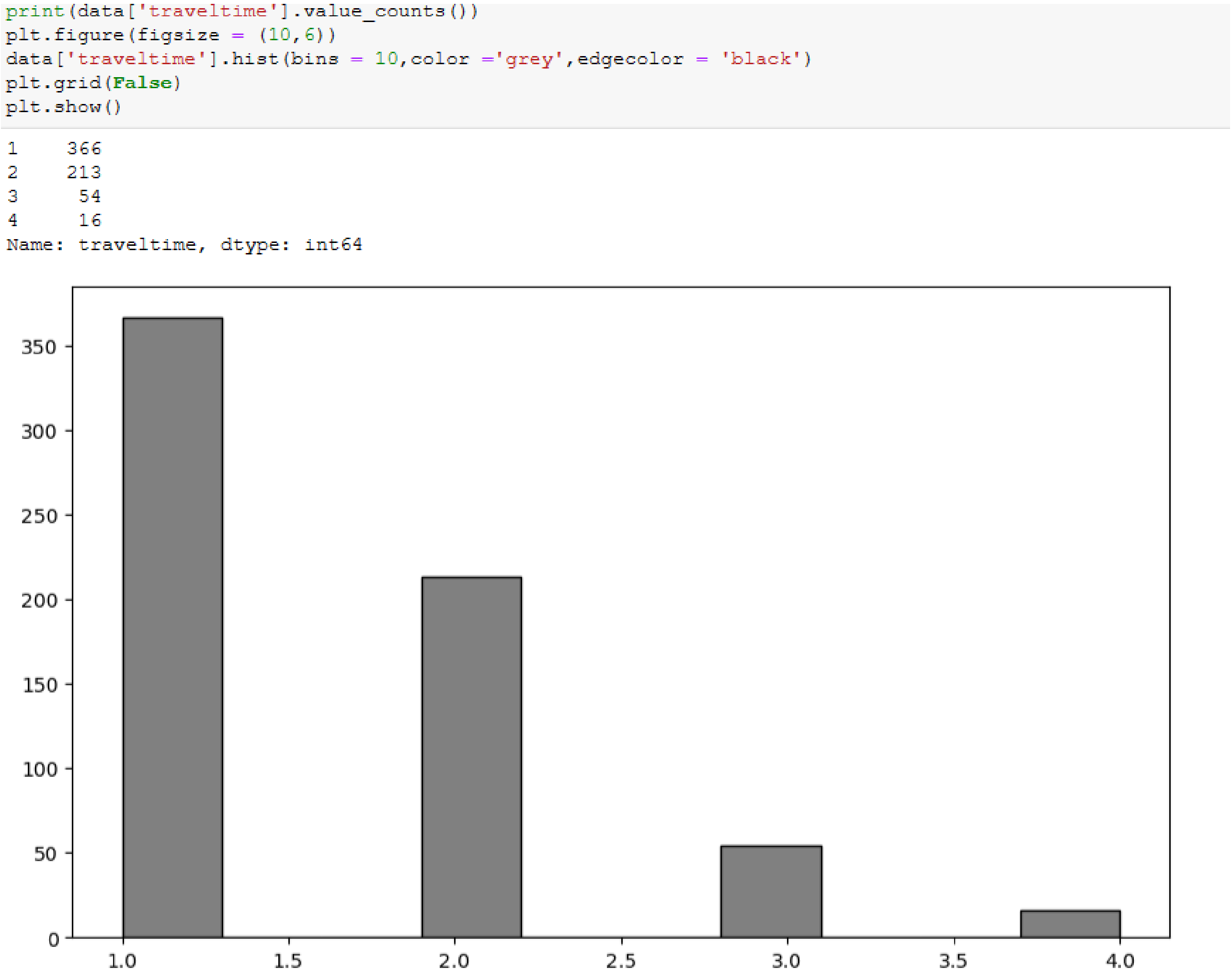


```
#3
print(data['sex'].value_counts())
plt.figure(figsize = (10,5))
data['sex'].hist(color = 'grey',edgecolor = 'black',bins = 10)
plt.show()
```

```
F    383
M    266
Name: sex, dtype: int64
```

4)Question 4

This code counts the number of traveltimes by group,draws histogram by defining size,color,edgecolor,deletes line(grid).The result returns count of each traveltime’s group,histogram to show distribution of student’s traveltime

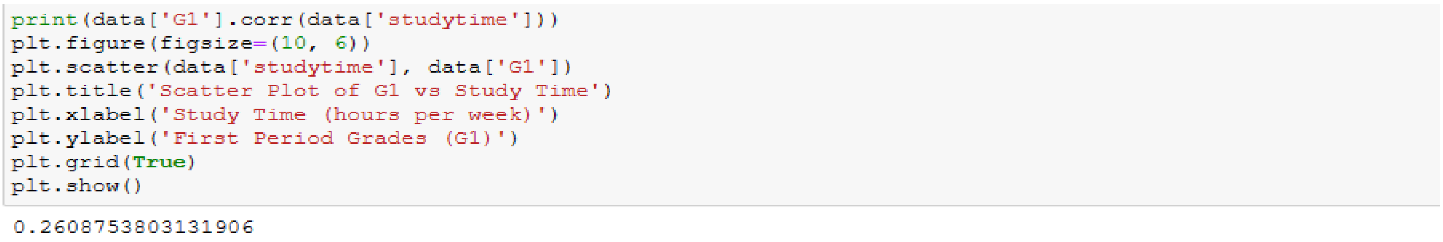


```
print(data['traveltime'].value_counts())
plt.figure(figsize = (10,6))
data['traveltime'].hist(bins = 10,color ='grey',edgecolor = 'black')
plt.grid(False)
plt.show()
```

```
1    366
2    213
3     54
4     16
Name: traveltime, dtype: int64
```

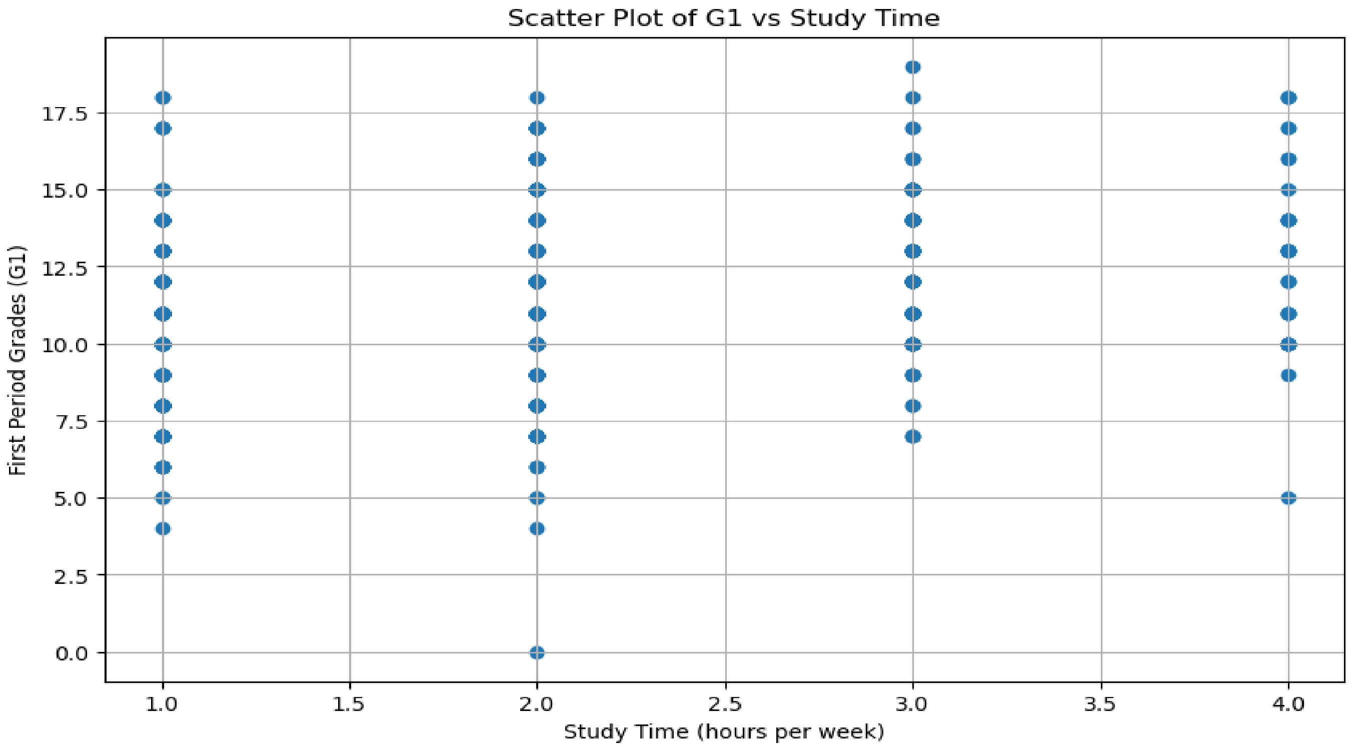
5)Question 5

This code calculates correlation coefficient of studytime and first period grade.Then draws scatter plot ,by giving title,labels,shows lines(grid) to better visualize.The result returns calculated coefficient,which is weakly positive correlation:if studytime increases G1 also increases. Also returns scatter plot to show how first period grade (G1) vary with studytime.



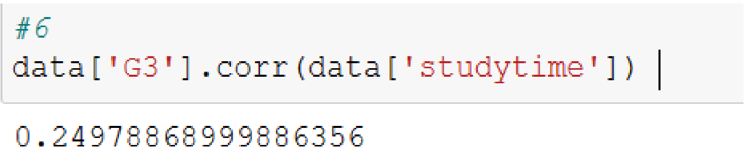
```
print(data['G1'].corr(data['studytime']))
plt.figure(figsize=(10, 6))
plt.scatter(data['studytime'], data['G1'])
plt.title('Scatter Plot of G1 vs Study Time')
plt.xlabel('Study Time (hours per week)')
plt.ylabel('First Period Grades (G1)')
plt.grid(True)
plt.show()
```

```
0.2608753803131906
```



6)Question 6

This code calculates correlation coefficient of studytime and final grade(G3).The result returns calculated coefficient ,which shows weakly positive correlation:if studytime increases G3 also increases.

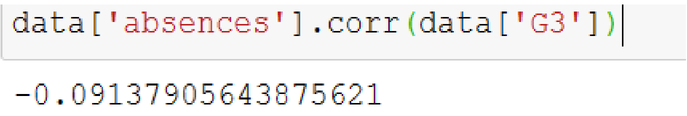


```
#6
data['G3'].corr(data['studytime'])
```

```
0.24978868999886356
```

7)Question 7

This code calculates correlation coefficient of absences and final grade(G3).The result returns calculated coefficient ,which shows negative correlation:if absence increases,final grade will lower.

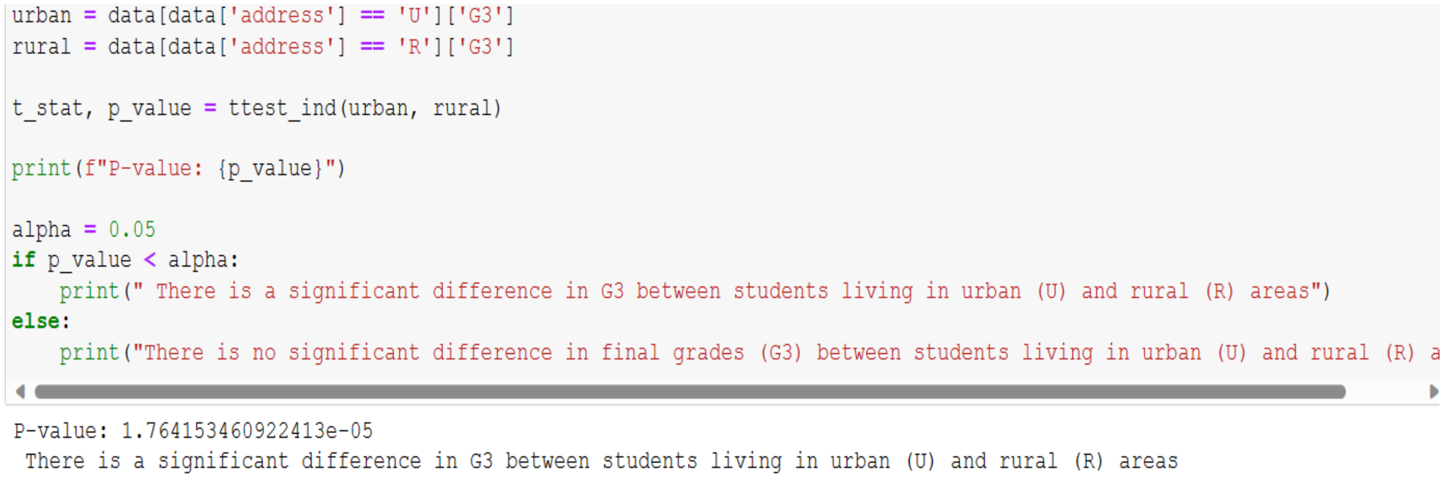


```
data['absences'].corr(data['G3'])
```

```
-0.09137905643875621
```

8)Question 8

This code divides data by address,gets their final grade value.Then does t-test to check difference,by calculating p value and comparing with alpha .The result prints Pvalue and answer:there is significant difference or not.

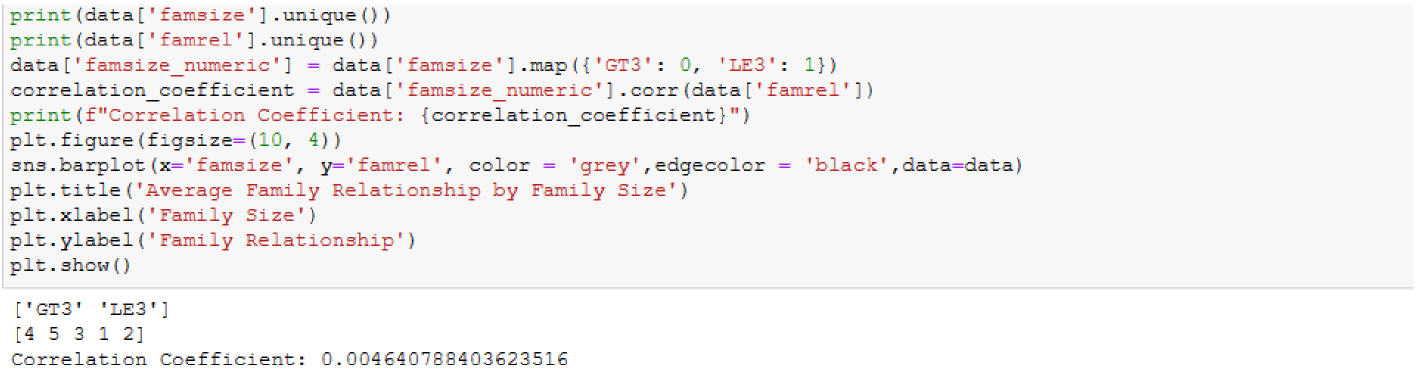


```
urban = data[data['address'] == 'U']['G3']
rural = data[data['address'] == 'R']['G3']

t_stat, p_value = ttest_ind(urban, rural)

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

alpha = 0.05
if p_value < alpha:
    print(" There is a significant difference in G3 between students living in urban (U) and rural (R) areas")
else:
    print("There is no significant difference in final grades (G3) between students living in urban (U) and rural (R) a
```

```
P-value: 1.764153460922413e-05
 There is a significant difference in G3 between students living in urban (U) and rural (R) areas
```

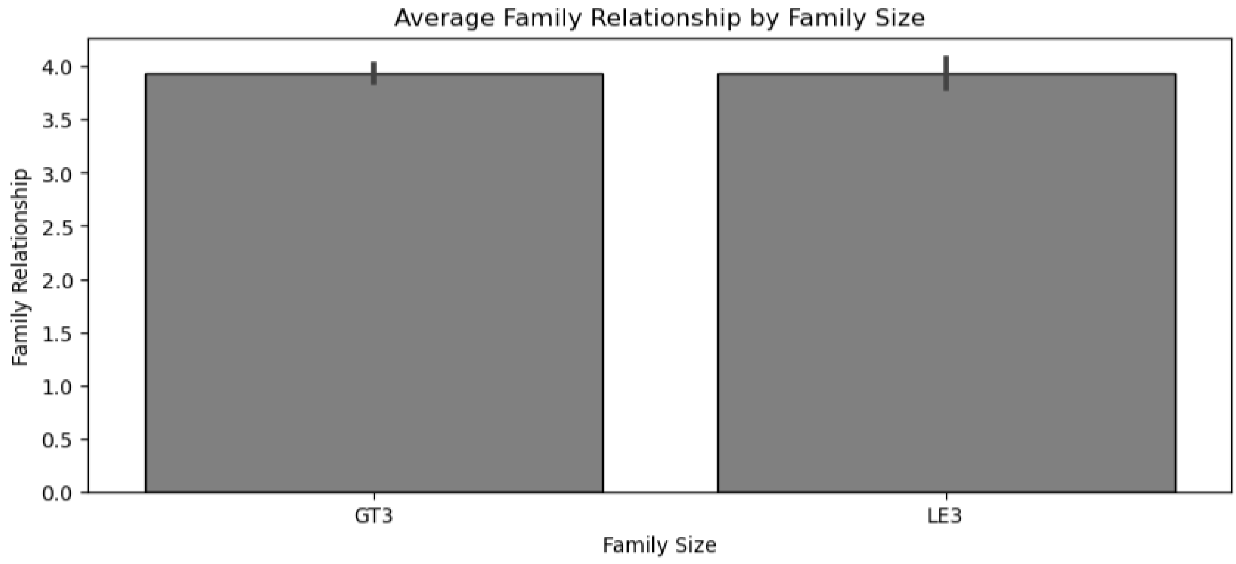
9)Question 9

 This code gets unique values of famsize and famrel,changes famsize values to numeric,calculates correlation coefficient,draws bar .The result returns unique values of columns,coefficient:which is near 0(very weak),it means famsize do not affect famrel as well,returns barplot to show relationship of famrel and famsize



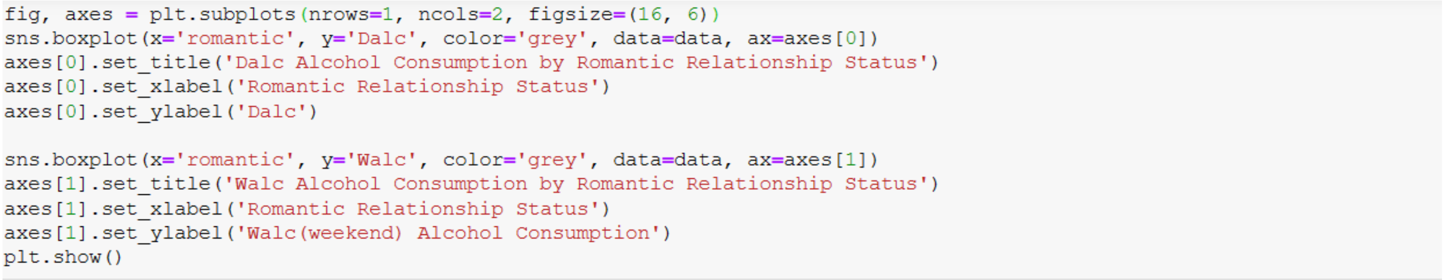
```
print(data['famsize'].unique())
print(data['famrel'].unique())
data['famsize_numeric'] = data['famsize'].map({'GT3': 0, 'LE3': 1})
correlation_coefficient = data['famsize_numeric'].corr(data['famrel'])
print(f"Correlation Coefficient: {correlation_coefficient}")
plt.figure(figsize=(10, 4))
sns.barplot(x='famsize', y='famrel', color = 'grey',edgecolor = 'black',data=data)
plt.title('Average Family Relationship by Family Size')
plt.xlabel('Family Size')
plt.ylabel('Family Relationship')
plt.show()
```

```
['GT3' 'LE3']
[4 5 3 1 2]
Correlation Coefficient: 0.004640788403623516
```

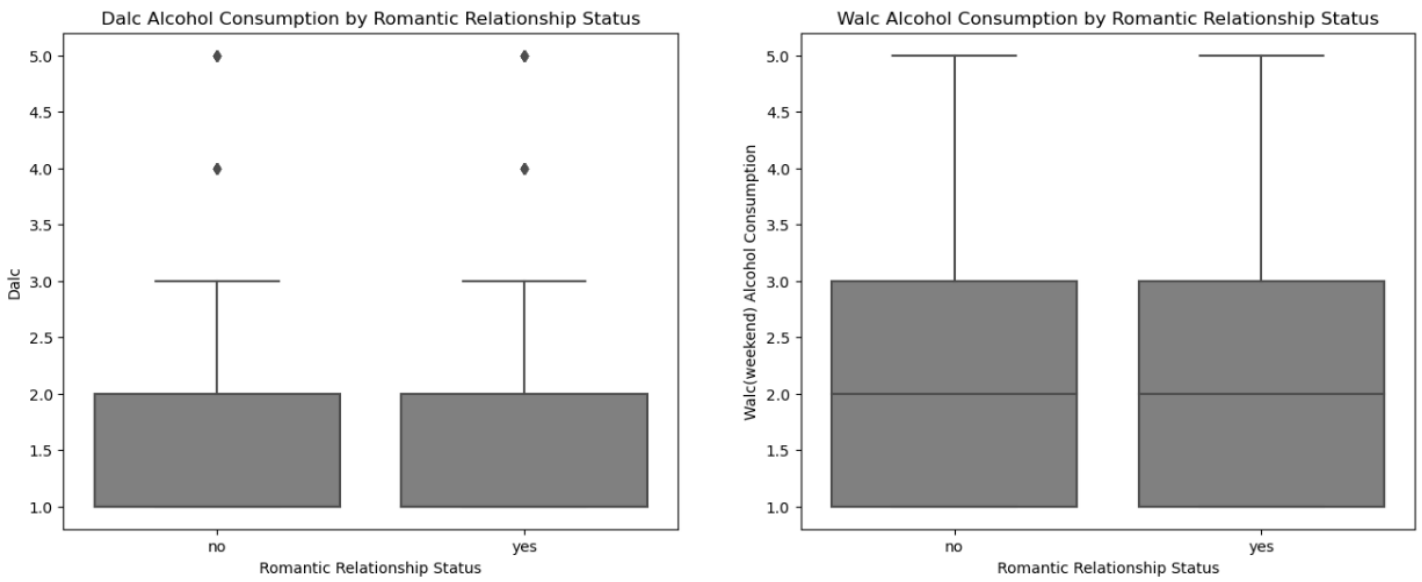


10)Question 10

This code draws 2 boxplots that determines being romantic affects to alcoholism or not.One for Weekday,one for Weekend,gives title and labels,color.The result returns 2 boxplots,which there is no big difference between romantic and not romantic students who drink alcohol,but in the weekend alcohol statistics increases.

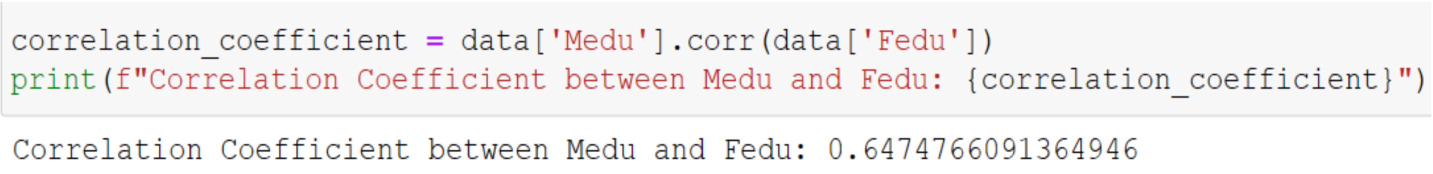


```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(16, 6))
sns.boxplot(x='romantic', y='Dalc', color='grey', data=data, ax=axes[0])
axes[0].set_title('Dalc Alcohol Consumption by Romantic Relationship Status')
axes[0].set_xlabel('Romantic Relationship Status')
axes[0].set_ylabel('Dalc')

sns.boxplot(x='romantic', y='Walc', color='grey', data=data, ax=axes[1])
axes[1].set_title('Walc Alcohol Consumption by Romantic Relationship Status')
axes[1].set_xlabel('Romantic Relationship Status')
axes[1].set_ylabel('Walc(weekend) Alcohol Consumption')
plt.show()
```



11)Question 11

This code calculates correlation coeficient of Mother’s and Fother’s education level.The result returns coefficient: if mother’s education level is above,father’s education level is approximately same

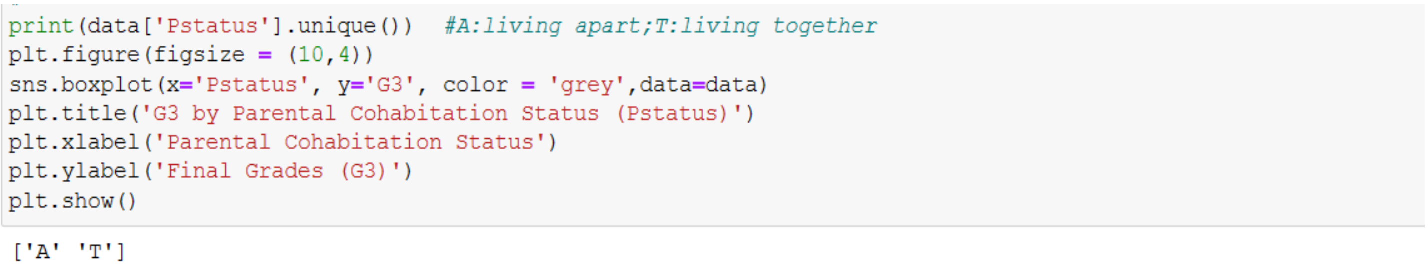


```
correlation_coefficient = data['Medu'].corr(data['Fedu'])
print(f"Correlation Coefficient between Medu and Fedu: {correlation_coefficient}")
```

```
Correlation Coefficient between Medu and Fedu: 0.6474766091364946
```

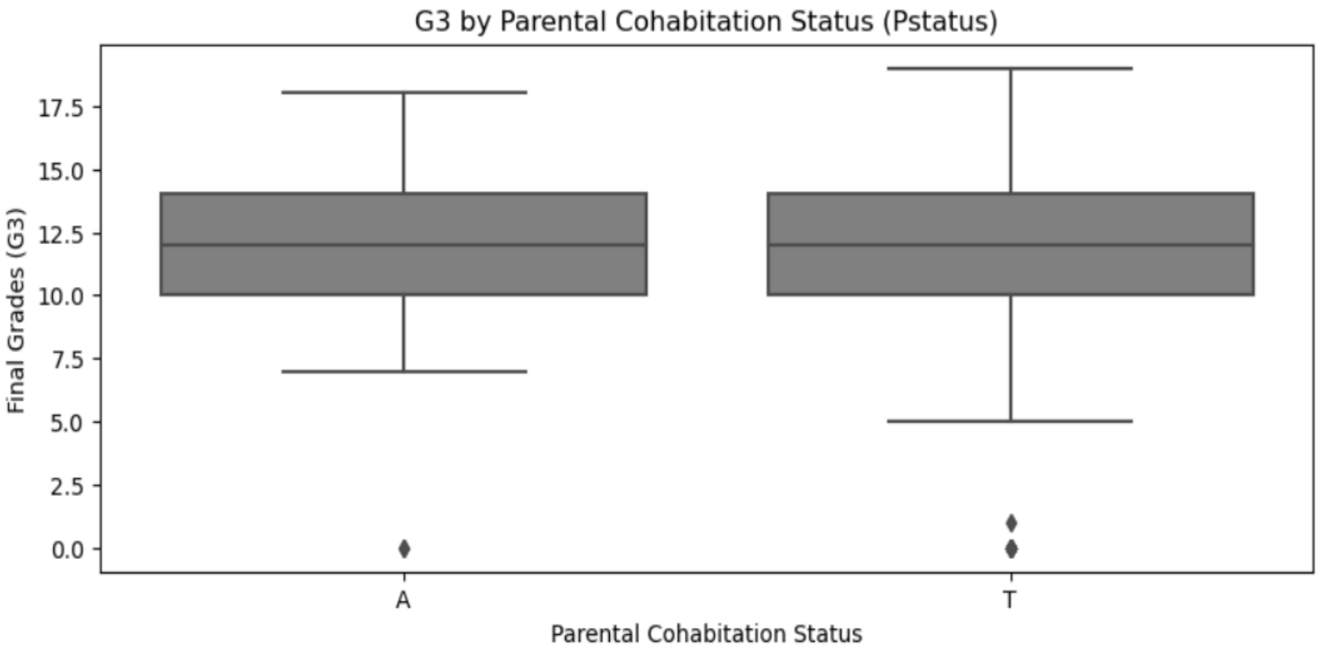
12)Question 12

This code draws boxplot by category of Parent status relation with Final grades,determines size,color,title and labels.The result returns unique value of Parent status and boxplot: students that live together with parents have maximum grades than students that living apart,but their grades are approximately same



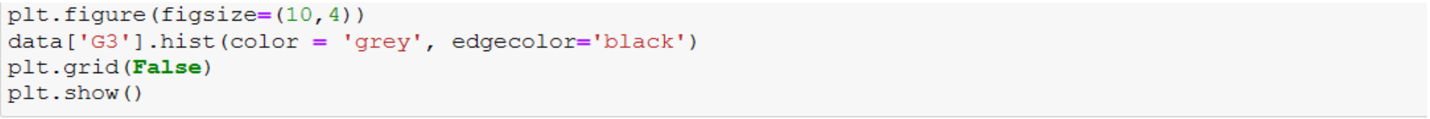
```
print(data['Pstatus'].unique())  #A:living apart;T:living together
plt.figure(figsize = (10,4))
sns.boxplot(x='Pstatus', y='G3', color = 'grey',data=data)
plt.title('G3 by Parental Cohabitation Status (Pstatus)')
plt.xlabel('Parental Cohabitation Status')
plt.ylabel('Final Grades (G3)')
plt.show()
```

```
['A' 'T']
```

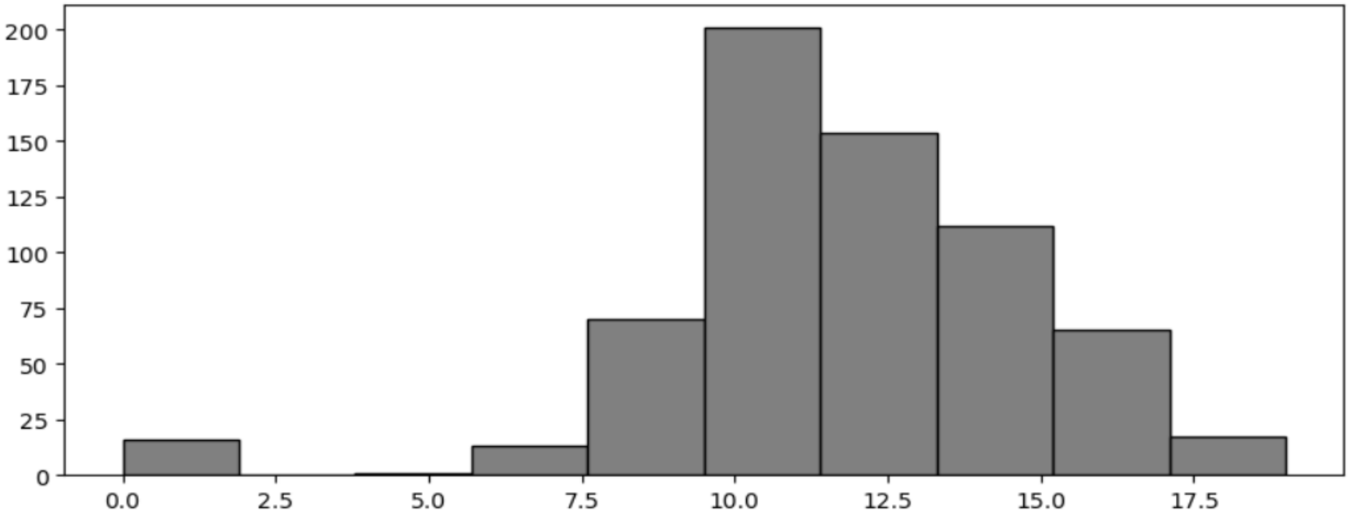


13)Question 13

The code draws histogram of Final grades by giving size,color,edgecolor,without lines(grid).The result returns histogram that shows the distribution of grade

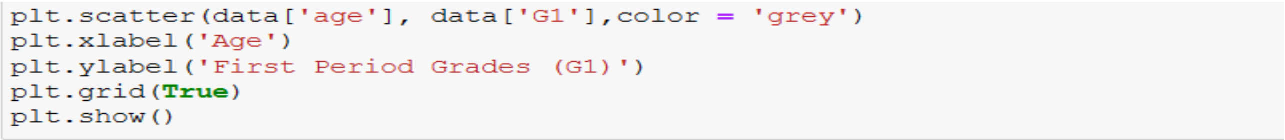


```
plt.figure(figsize=(10,4))
data['G3'].hist(color = 'grey', edgecolor='black')
plt.grid(False)
plt.show()
```

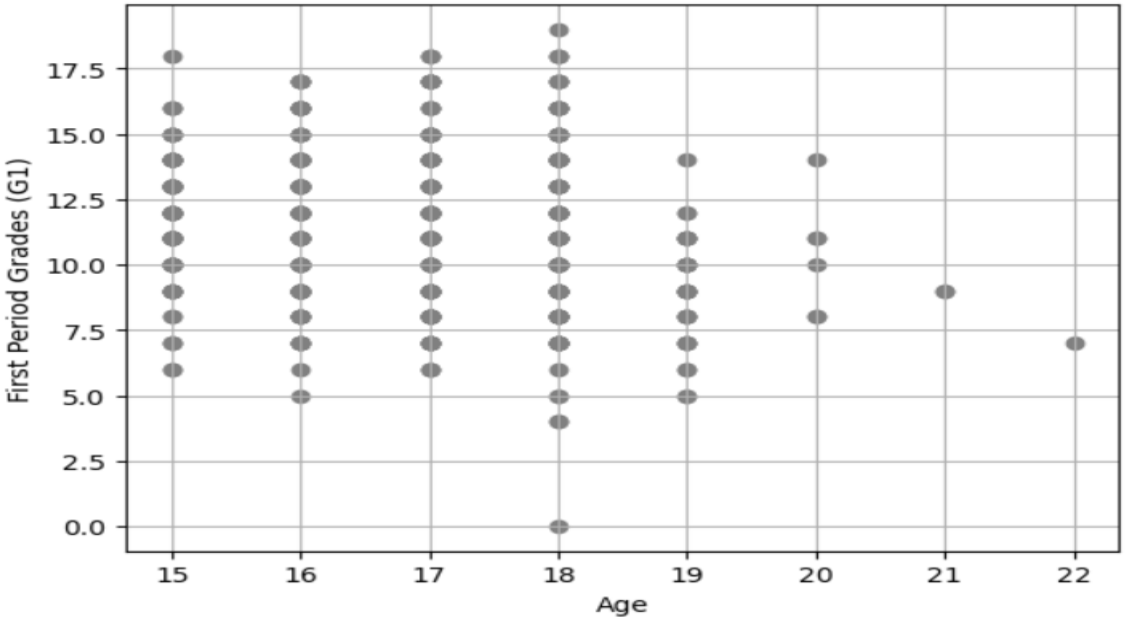


14)Question 14

This code draws scatter plot that shows relationship of G1 and age by giving color,label,grid.The result returns Scatter plot

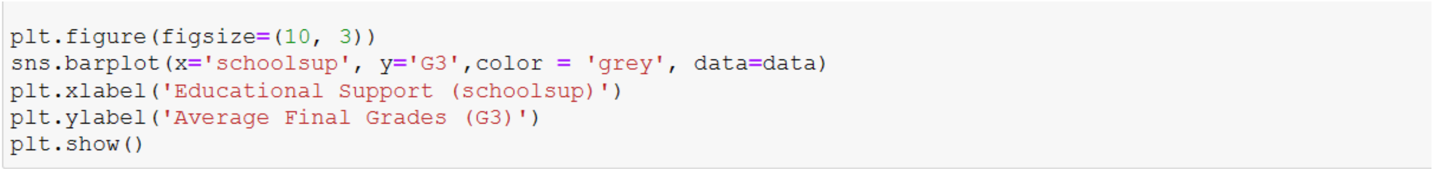


```
plt.scatter(data['age'], data['G1'],color = 'grey')
plt.xlabel('Age')
plt.ylabel('First Period Grades (G1)')
plt.grid(True)
plt.show()
```

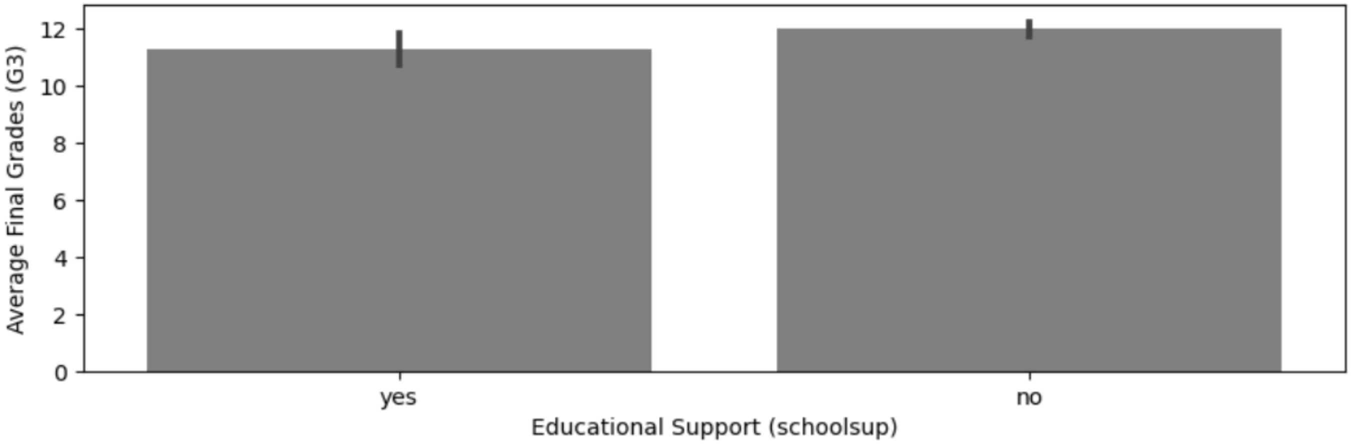


15)Question 15

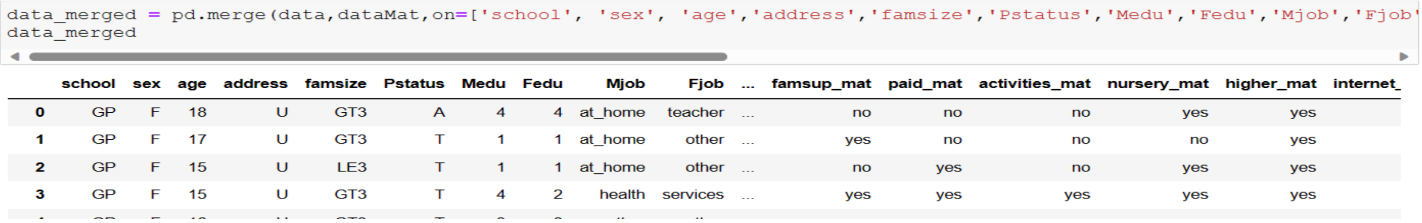
The code draws bar chart ,by calculating average final grades of categories of schoolsup.The result returns bar chart,that shows students without schoolsup have greater final grades than sudents with schoolsup.



```
plt.figure(figsize=(10, 3))
sns.barplot(x='schoolsup', y='G3',color = 'grey', data=data)
plt.xlabel('Educational Support (schoolsup)')
plt.ylabel('Average Final Grades (G3)')
plt.show()
```



Combination of two datasets to answer to the next questions:

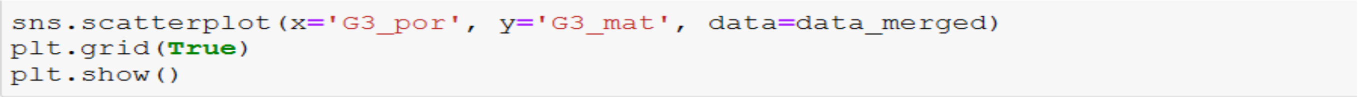


```
data_merged = pd.merge(data,dataMat,on=['school', 'sex', 'age','address','famsize','Pstatus','Medu','Fedu','Mjob','Fjob'
data_merged
```

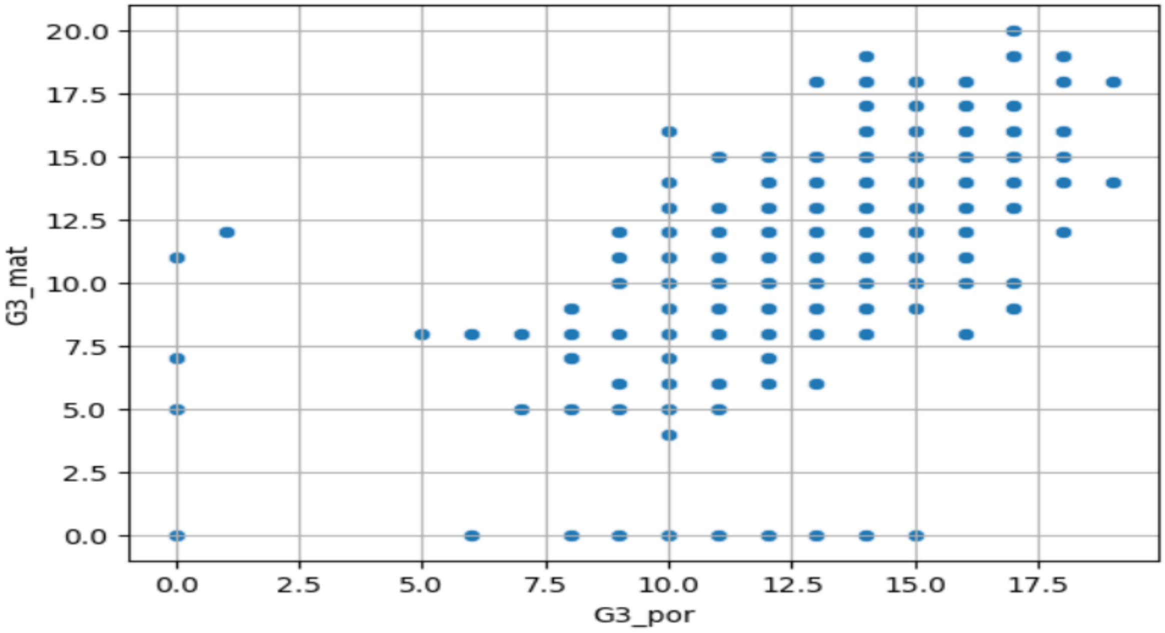
	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famsup_mat	paid_mat	activities_mat	nursery_mat	higher_mat	internet_
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	no	no	no	yes	yes	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	yes	no	no	no	yes	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	no	yes	no	yes	yes	
3	GP	F	15	U	GT3	T	4	2	health	services	...	yes	yes	yes	yes	yes	

16)Question 16

This code draws scatter plot of grades of math and Portuguese courses of students,The result retirns scatter plot,which shows the comparison of two grades

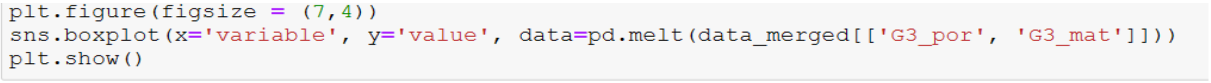


```
sns.scatterplot(x='G3_por', y='G3_mat', data=data_merged)
plt.grid(True)
plt.show()
```

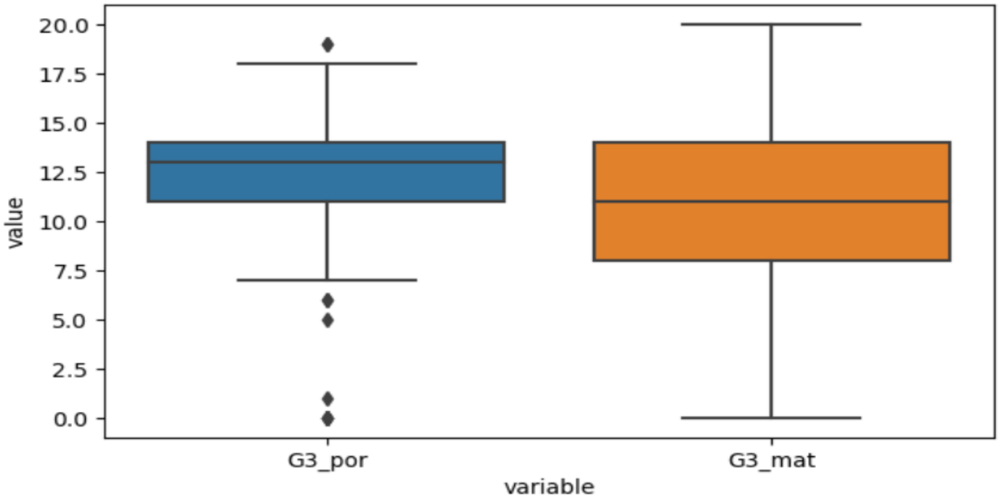


17)Question 17

This code draws box plot for comparing final grades in Portuguese and Math courses ,reshapes the ‘data_merged’ to better visualization.The result displays the boxplot,where we can see that G3 of Portuguese course have outliners,mean is greater than G3 of Math course

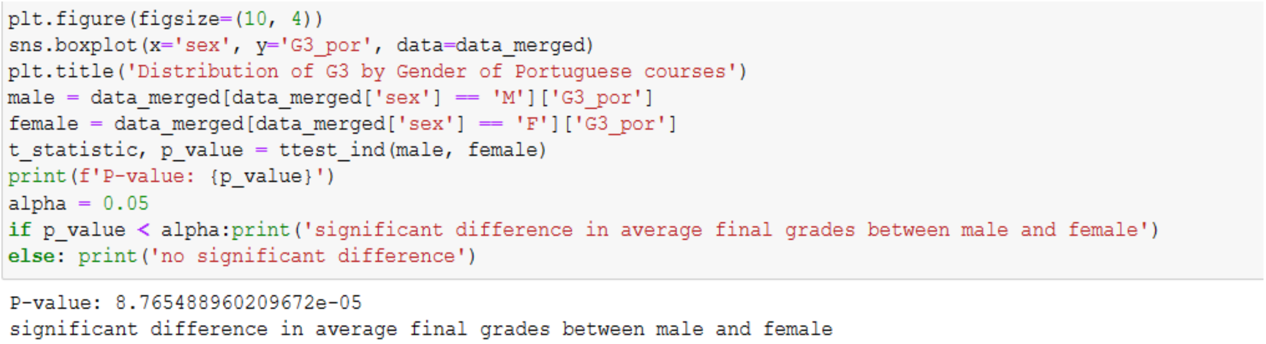


```
plt.figure(figsize = (7,4))
sns.boxplot(x='variable', y='value', data=pd.melt(data_merged[['G3_por', 'G3_mat']]))
plt.show()
```



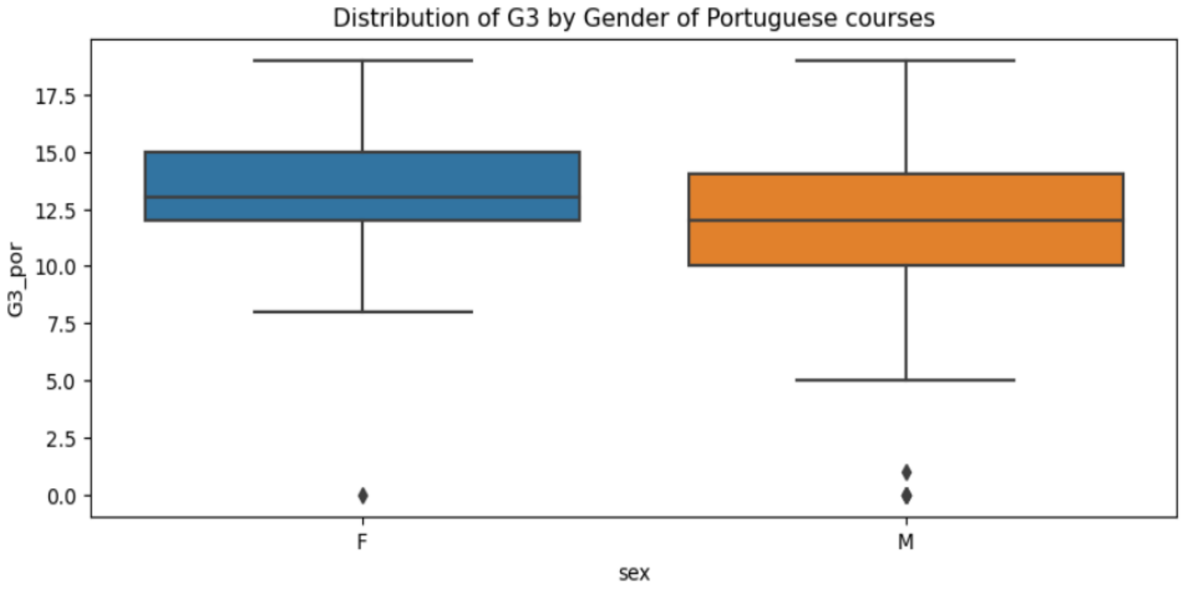
18)Question 18

This code draws boxplot to compare final grades by gender,then divides into two groups ,male and female by taking gender with final grades.Then does two-sample t test,compares p value with alpha.The result returns boxplot and p value and shows there is difference or not.



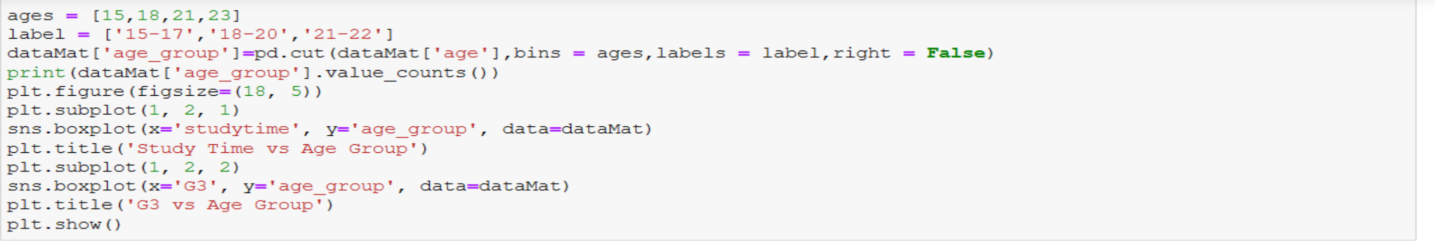
```
plt.figure(figsize=(10, 4))
sns.boxplot(x='sex', y='G3_por', data=data_merged)
plt.title('Distribution of G3 by Gender of Portuguese courses')
male = data_merged[data_merged['sex'] == 'M']['G3_por']
female = data_merged[data_merged['sex'] == 'F']['G3_por']
t_statistic, p_value = ttest_ind(male, female)
print(f'P-value: {p_value}')
alpha = 0.05
if p_value < alpha:print('significant difference in average final grades between male and female')
else: print('no significant difference')
```

```
P-value: 8.765488960209672e-05
significant difference in average final grades between male and female
```



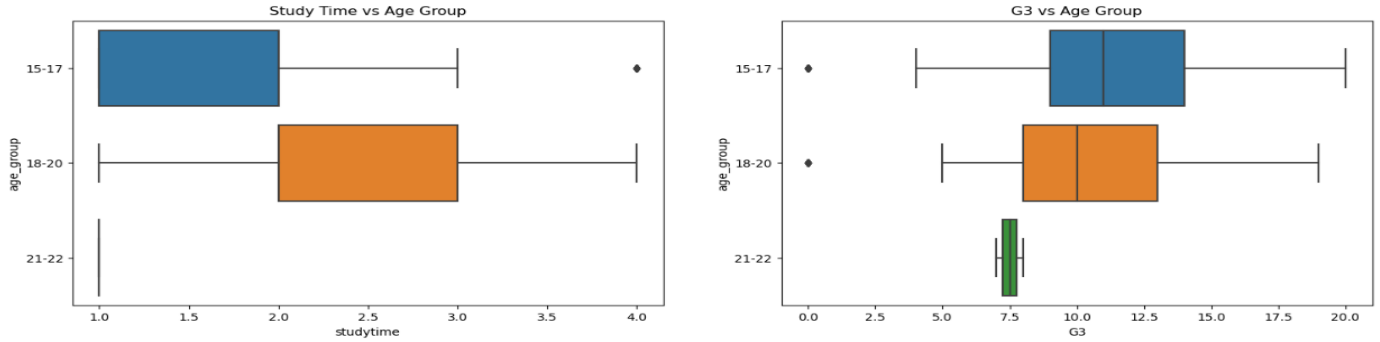
19)Question 19

This code creates age groups by categorizing ‘age’ column into specified age bins.Creates 2 boxplot by subplotting,first for relationship between studytime and age group,second for G3 and age group.The result prints the counts of individuals in each age group,shows boxplot,that 21-22 age groups are not study hard,etc



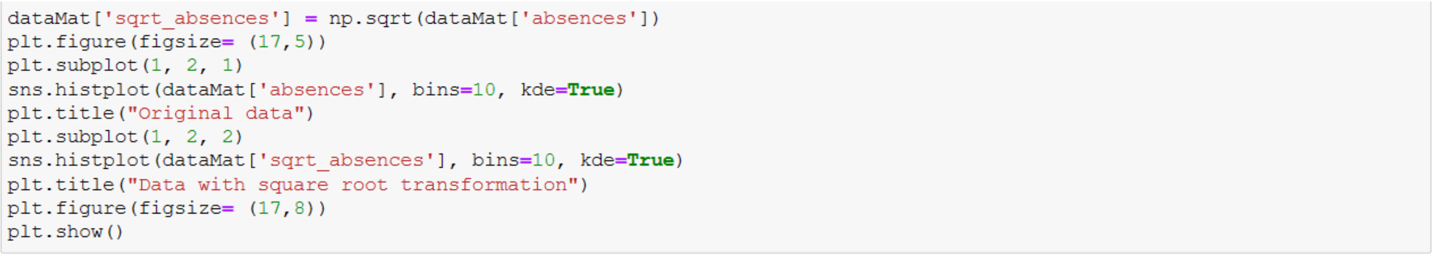
```
ages = [15,18,21,23]
label = ['15-17','18-20','21-22']
dataMat['age_group']=pd.cut(dataMat['age'],bins = ages,labels = label,right = False)
print(dataMat['age_group'].value_counts())
plt.figure(figsize=(18, 5))
plt.subplot(1, 2, 1)
sns.boxplot(x='studytime', y='age_group', data=dataMat)
plt.title('Study Time vs Age Group')
plt.subplot(1, 2, 2)
sns.boxplot(x='G3', y='age_group', data=dataMat)
plt.title('G3 vs Age Group')
plt.show()
```

```
15-17    284
18-20    109
21-22      2
Name: age_group, dtype: int64
```

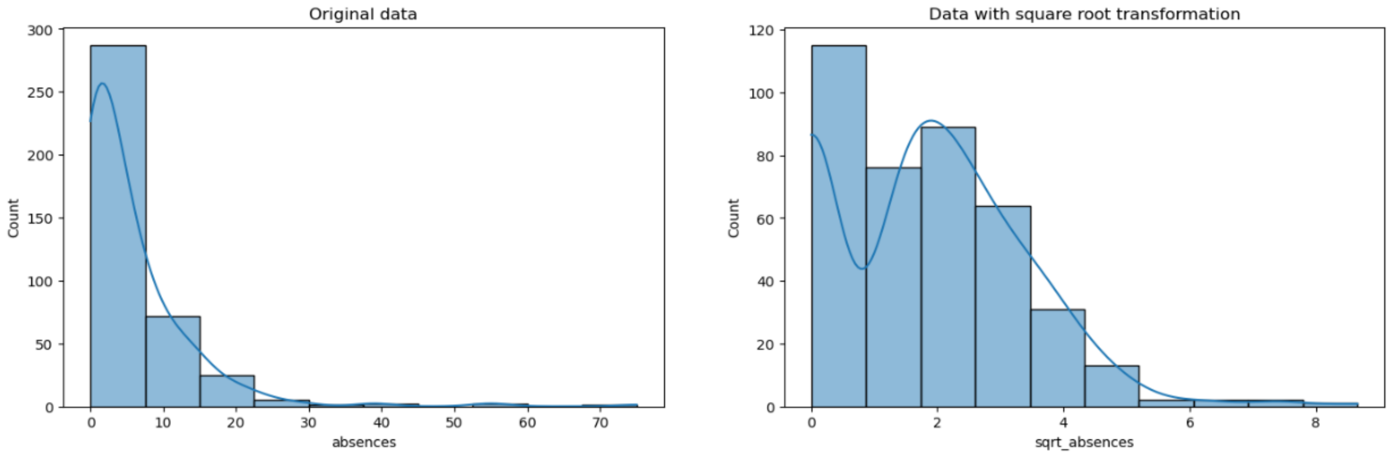


20)Question 20

This code does square root transformation to the absences,draws histograms which compares transformed data with original.The result shows 2 histograms ,by them we can see that with square root transformation gives better visualization of distribution.

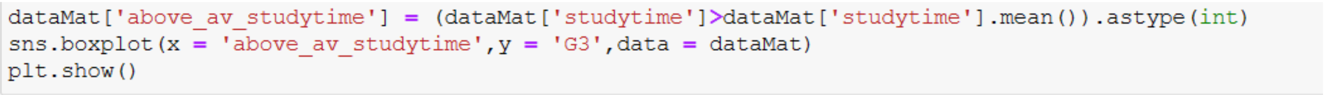


```
dataMat['sqrt_absences'] = np.sqrt(dataMat['absences'])
plt.figure(figsize= (17,5))
plt.subplot(1, 2, 1)
sns.histplot(dataMat['absences'], bins=10, kde=True)
plt.title("Original data")
plt.subplot(1, 2, 2)
sns.histplot(dataMat['sqrt_absences'], bins=10, kde=True)
plt.title("Data with square root transformation")
plt.figure(figsize= (17,8))
plt.show()
```

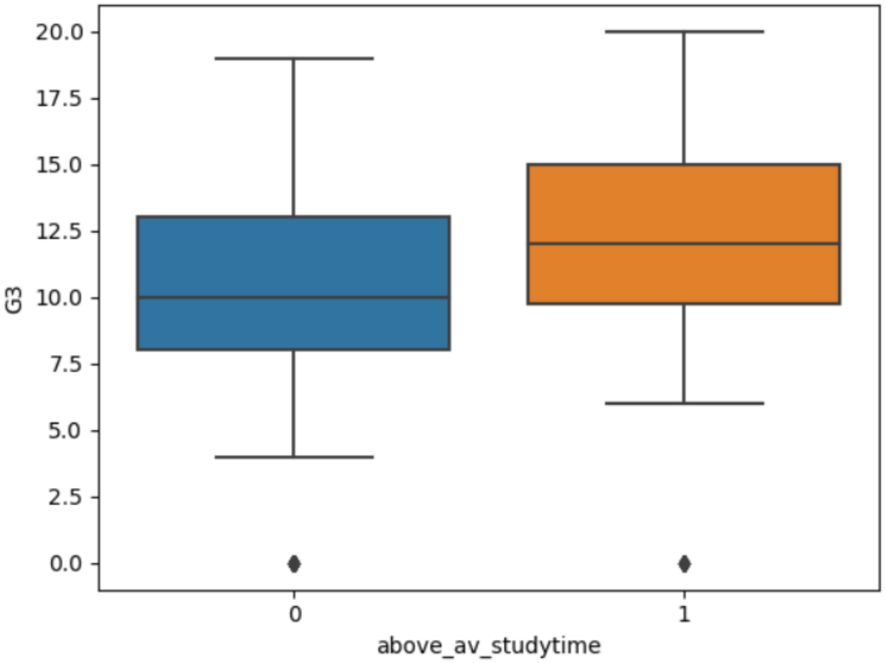


21)Question 21

This code creates new column which takes values that are greater than average value of studytime ,draws boxplot to show relation of them with G3.The result returns boxplot ,which shows students with above-average studytime have good grades.0 means less than average studytime,1 means greater than average studytime

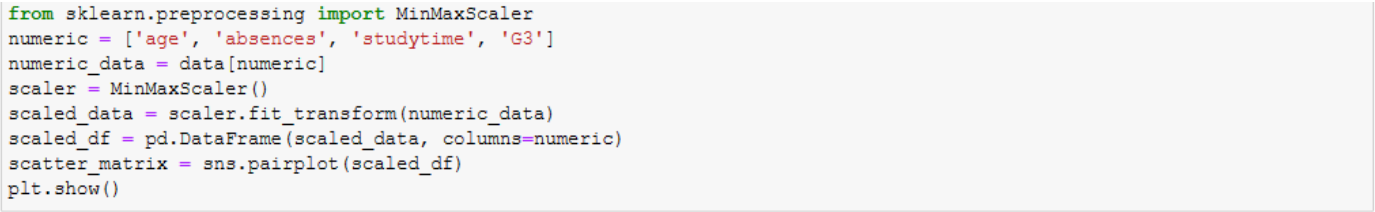


```
dataMat['above_av_studytime'] = (dataMat['studytime']>dataMat['studytime'].mean()).astype(int)
sns.boxplot(x = 'above_av_studytime',y = 'G3',data = dataMat)
plt.show()
```

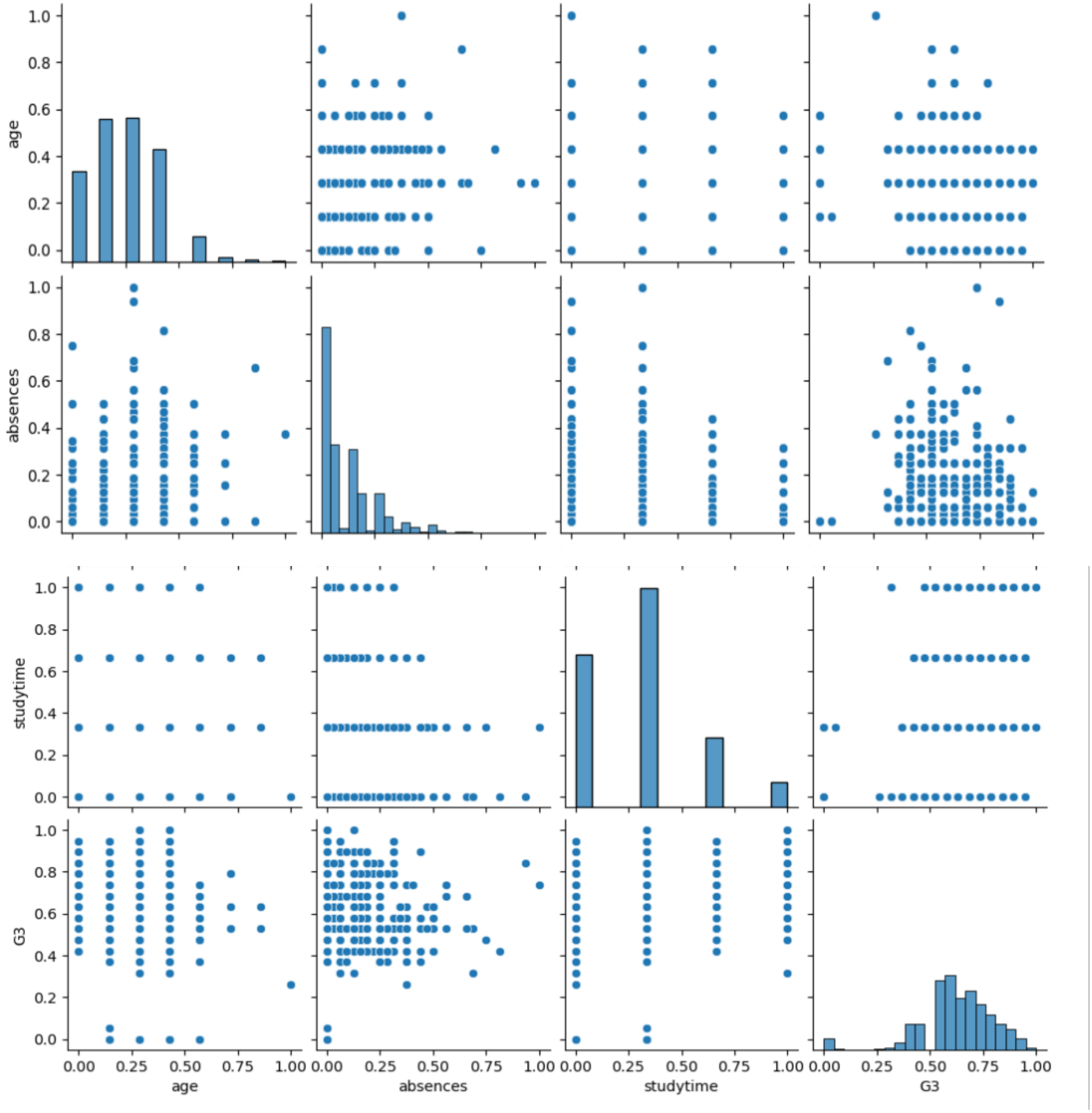


22)Question 22

This code imports library and takes numeric values and does MinMaxScaler,for them does scatter matrix.The result returns scatter matrix,which shows scaled data.It gives better visualization between numeric values and G3 comparing with non scaled values.

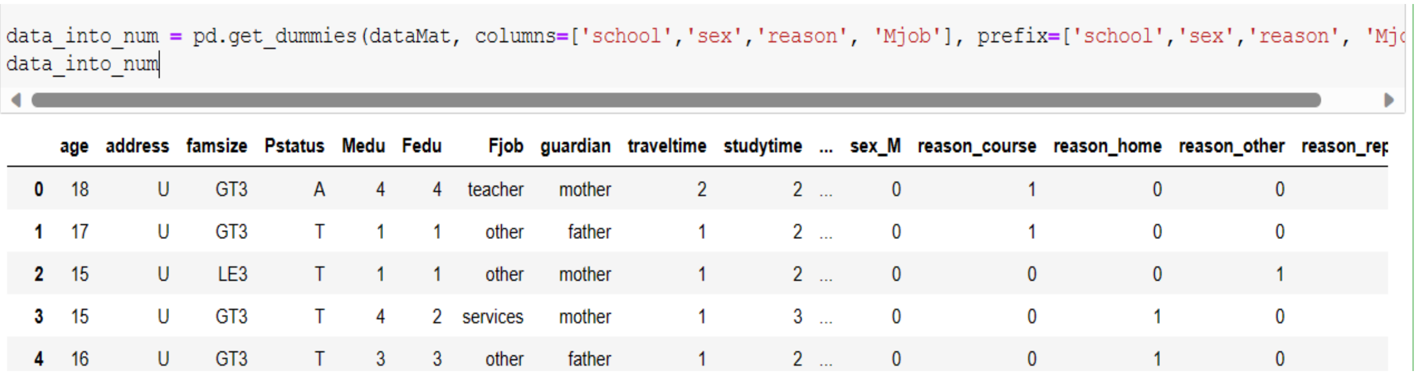


```
from sklearn.preprocessing import MinMaxScaler
numeric = ['age', 'absences', 'studytime', 'G3']
numeric_data = data[numeric]
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(numeric_data)
scaled_df = pd.DataFrame(scaled_data, columns=numeric)
scatter_matrix = sns.pairplot(scaled_df)
plt.show()
```



23)Question 23

This code converts categorical variables into numeric by one-hot encodeing.The result returns data with changed values.This method sperates every columns into their values ,1 means true,0 means false.It is better for analyzing.

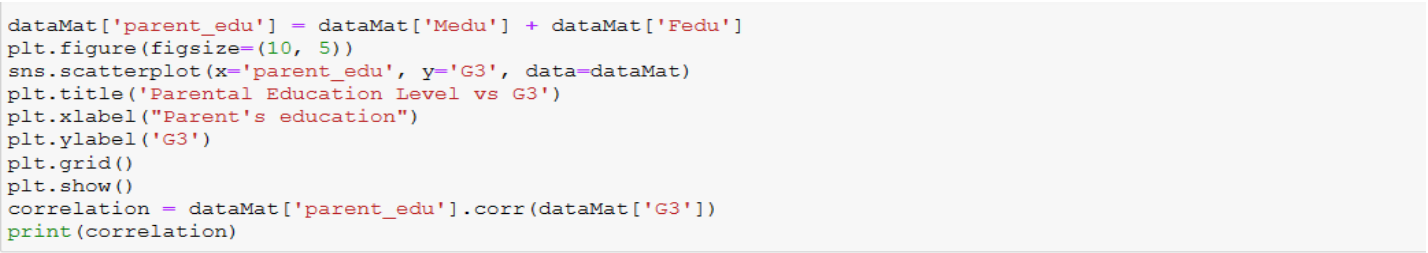


```
data_into_num = pd.get_dummies(dataMat, columns=['school','sex','reason', 'Mjob'], prefix=['school','sex','reason', 'Mj
data_into_num
```

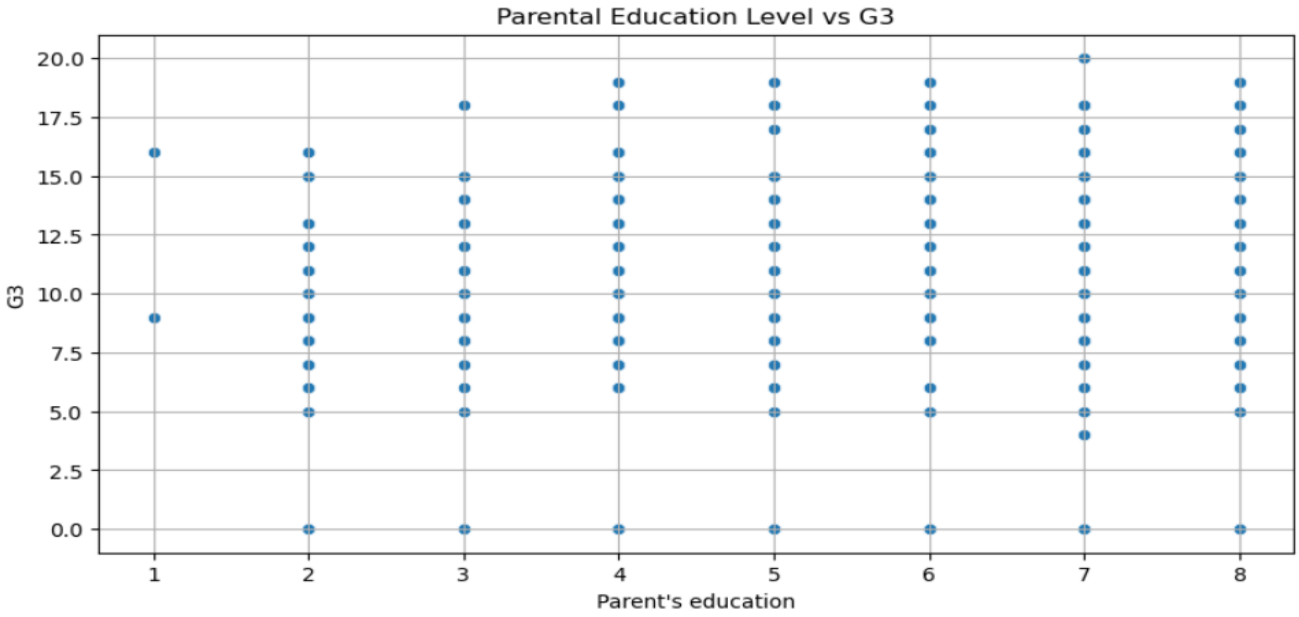
	age	address	famsize	Pstatus	Medu	Fedu	Fjob	guardian	traveltime	studytime	...	sex_M	reason_course	reason_home	reason_other	reason_rep
0	18	U	GT3	A	4	4	teacher	mother	2	2	...	0	1	0	0	
1	17	U	GT3	T	1	1	other	father	1	2	...	0	1	0	0	
2	15	U	LE3	T	1	1	other	mother	1	2	...	0	0	0	1	
3	15	U	GT3	T	4	2	services	mother	1	3	...	0	0	1	0	
4	16	U	GT3	T	3	3	other	father	1	2	...	0	0	1	0	

24)Question 24

This code adds Medu and Fedu values into new column ‘parent_edu’ ,then generates scatter plot to show correlation of ‘parent_edu’ and ‘G3’.In addition, code calculates correlation coefficient .The result returns scatter plot and coefficient ,which is weak positive:if parent_edu is more,G3 is also more.



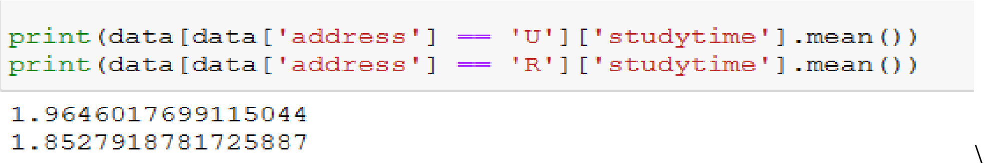
```
dataMat['parent_edu'] = dataMat['Medu'] + dataMat['Fedu']
plt.figure(figsize=(10, 5))
sns.scatterplot(x='parent_edu', y='G3', data=dataMat)
plt.title('Parental Education Level vs G3')
plt.xlabel("Parent's education")
plt.ylabel('G3')
plt.grid()
plt.show()
correlation = dataMat['parent_edu'].corr(dataMat['G3'])
print(correlation)
```



```
0.20522443411453914
```

25)Question 25

This code calculates the mean of studytime of Urban students,then Rural students.The result returns their means,they are approximately same, but Urban’s are little bit more.



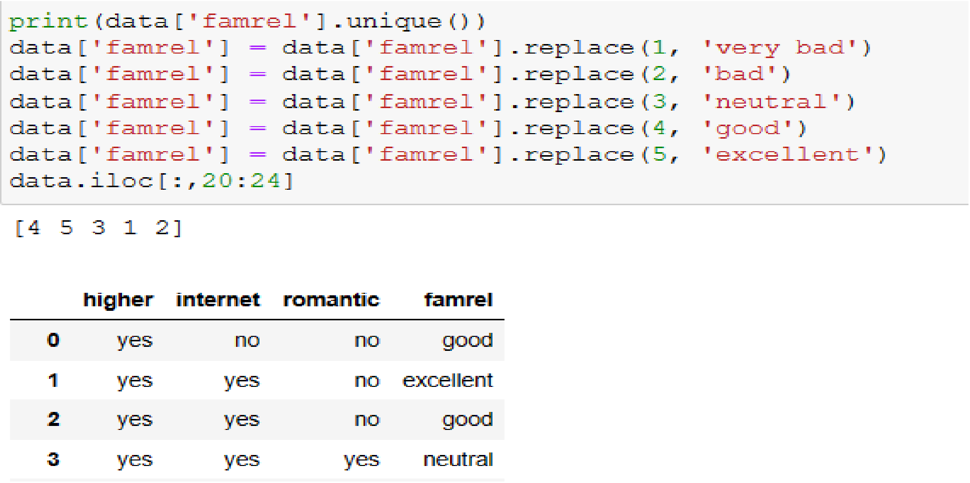
```
print(data[data['address'] == 'U']['studytime'].mean())
print(data[data['address'] == 'R']['studytime'].mean())
```

```
1.9646017699115044
1.8527918781725887
```

\

26)Question 26

This code finds unique values of family relation.Then converts each numeric values of famrel to words.The result returns unique values of famrel,then data with changed famrel column.It is more readable and understandable to broader audience.



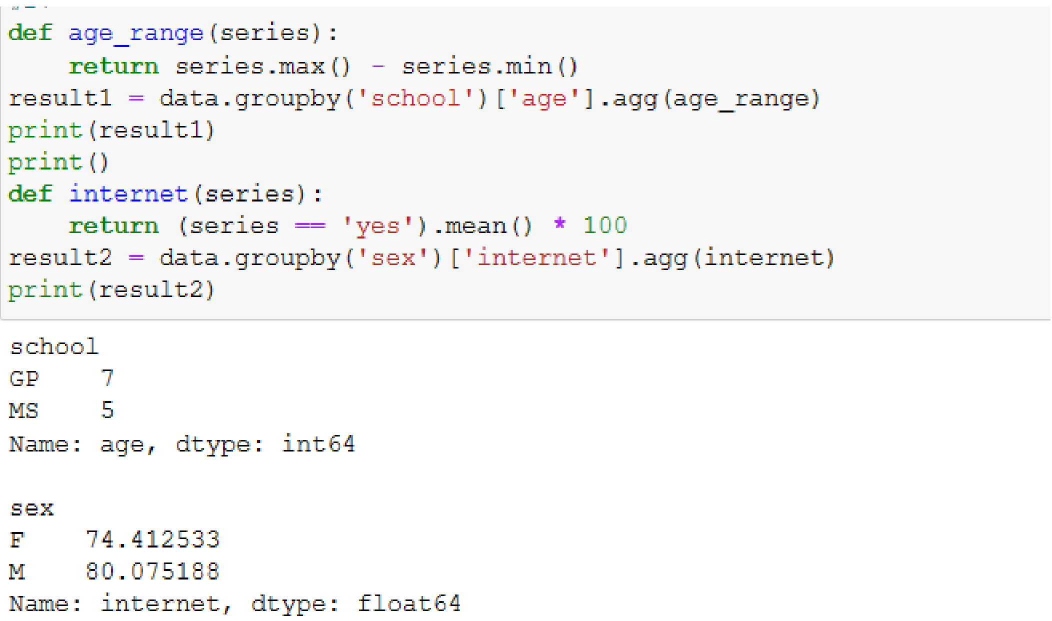
```
print(data['famrel'].unique())
data['famrel'] = data['famrel'].replace(1, 'very bad')
data['famrel'] = data['famrel'].replace(2, 'bad')
data['famrel'] = data['famrel'].replace(3, 'neutral')
data['famrel'] = data['famrel'].replace(4, 'good')
data['famrel'] = data['famrel'].replace(5, 'excellent')
data.iloc[:,20:24]
```

```
[4 5 3 1 2]
```

	higher	internet	romantic	famrel
0	yes	no	no	good
1	yes	yes	no	excellent
2	yes	yes	no	good
3	yes	yes	yes	neutral

27)Question 27

This code creates function which finds the range of ages ,also creates second function to determine percentage of students having internet.The result returns ranges of ages in different school, percentages of having internet students by gender.

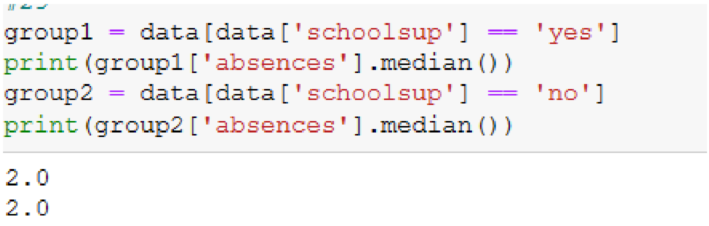


```
def age_range(series):
    return series.max() - series.min()
result1 = data.groupby('school')['age'].agg(age_range)
print(result1)
print()
def internet(series):
    return (series == 'yes').mean() * 100
result2 = data.groupby('sex')['internet'].agg(internet)
print(result2)
```

```
school
GP    7
MS    5
Name: age, dtype: int64

sex
F    74.412533
M    80.075188
Name: internet, dtype: float64
```

29)Question 29

This code creates two groups ,first is students with schoolsup ,second is without,calculates their absence median.The result returns two group’s median which they are same.

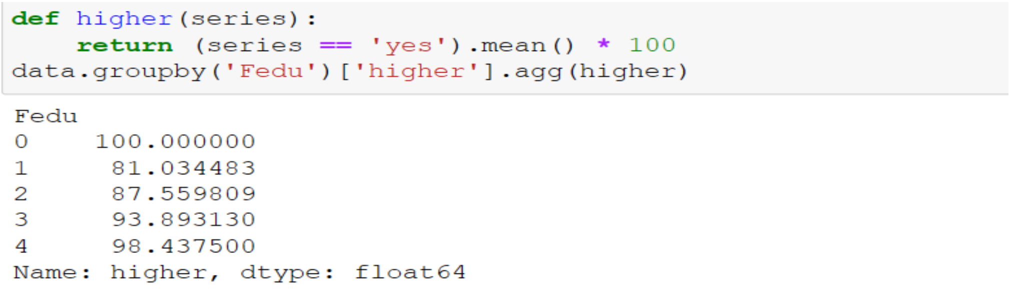


```
group1 = data[data['schoolsup'] == 'yes']
print(group1['absences'].median())
group2 = data[data['schoolsup'] == 'no']
print(group2['absences'].median())
```

```
2.0
2.0
```

30)Question 30

This code creates function higher which calculates percentages of students which want to take higher education.The result returns percentages by groups of Father education level.

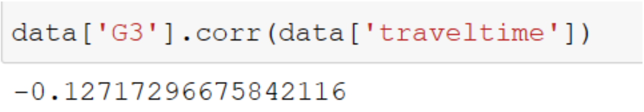


```
def higher(series):
    return (series == 'yes').mean() * 100
data.groupby('Fedu')['higher'].agg(higher)
```

```
Fedu
0    100.000000
1     81.034483
2     87.559809
3     93.893130
4     98.437500
Name: higher, dtype: float64
```

31)Question 31

This code calculates correlation coefficient of G3 and traveltime.The result returns coefficient ,that is near to negative:it means if G3 is lower,traveltime is more

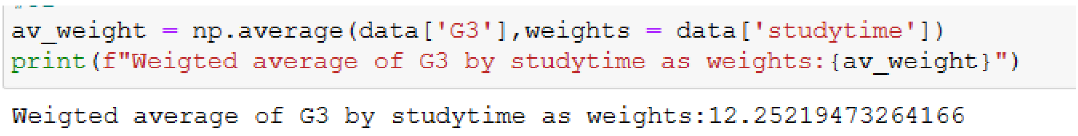


```
data['G3'].corr(data['traveltime'])
```

```
-0.12717296675842116
```

32)Question 32

This code calculates average of G3 by the weight of studytime.If there is more studytime,it gives weight to studytime.The result returns weighted average

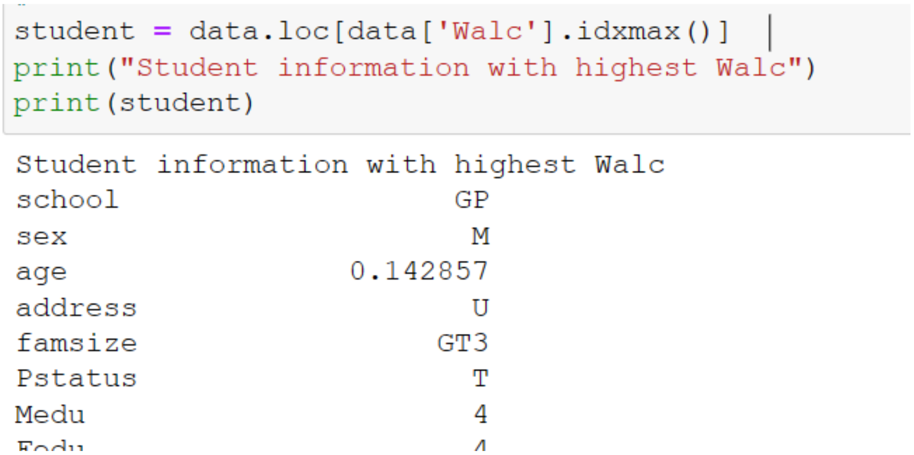


```
av_weight = np.average(data['G3'],weights = data['studytime'])
print(f"Weigted average of G3 by studytime as weights:{av_weight}")
```

```
Weigted average of G3 by studytime as weights:12.25219473264166
```

33)Question 33

This code takes first index of highest ‘Walc’ value.The result returns information of this student

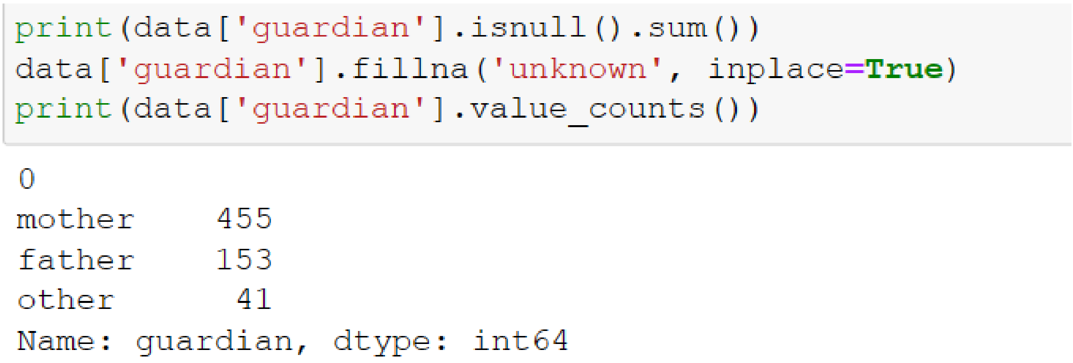


```
student = data.loc[data['Walc'].idxmax()]
print("Student information with highest Walc")
print(student)
```

```
Student information with highest Walc
school                       GP
sex                           M
age                    0.142857
address                       U
famsize                     GT3
Pstatus                       T
Medu                          4
Fedu                          4
```

34)Question 34

This code checks in guardian column has null values or not .In our case , there is no null values,but I decided to do filling:replacing null values with word ‘unknown’.The result returns number of null values,number of guardian by group

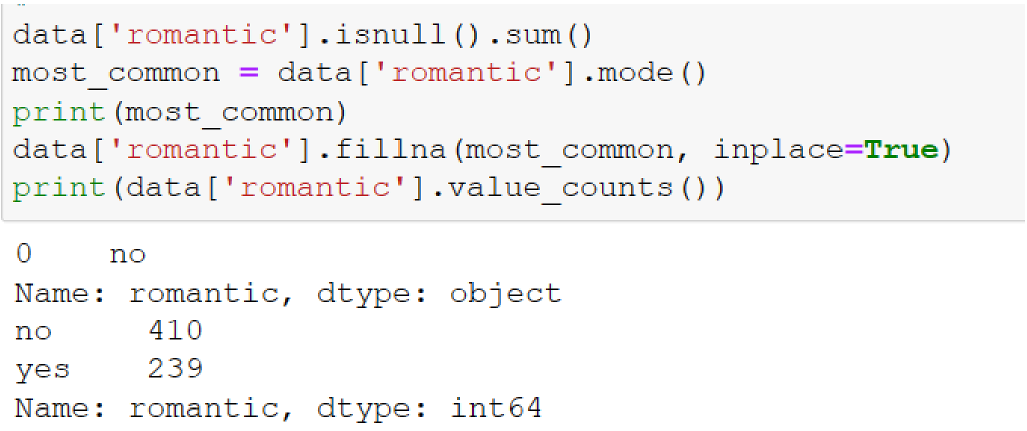


```
print(data['guardian'].isnull().sum())
data['guardian'].fillna('unknown', inplace=True)
print(data['guardian'].value_counts())
```

```
0
mother    455
father    153
other      41
Name: guardian, dtype: int64
```

35)Question 35

This code firstly checks in romantic column has null value or not,in our case there is no null values.But I decided to write code for filling null values:it takes mode of romantic and fills null values with this mode.The result prints count of null value,most common romantic value,the number of romantic values by group:no/yes

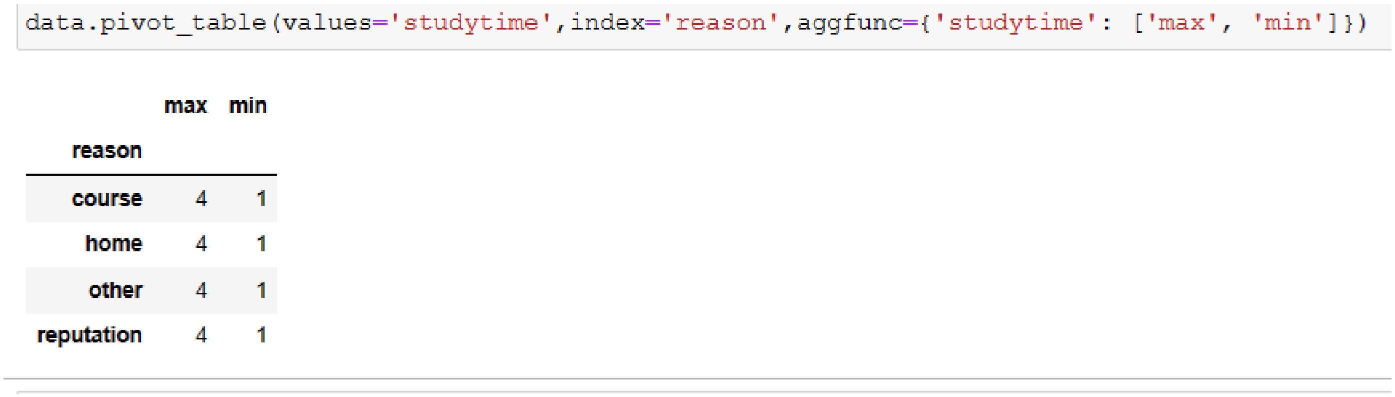


```
data['romantic'].isnull().sum()
most_common = data['romantic'].mode()
print(most_common)
data['romantic'].fillna(most_common, inplace=True)
print(data['romantic'].value_counts())
```

```
0    no
Name: romantic, dtype: object
no     410
yes    239
Name: romantic, dtype: int64
```

36)Question 36

This code creates pivot table ,sets index reason ,at the column shows studytime’s min and max values,on the value sets studytime value.The result returns pivot table

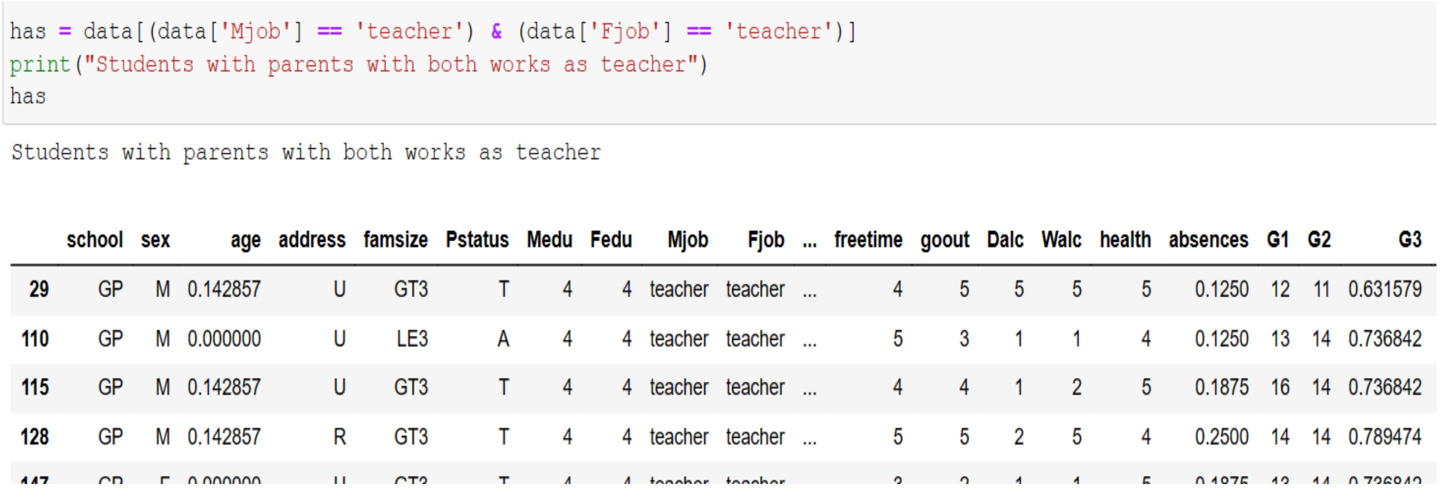


```
data.pivot_table(values='studytime',index='reason',aggfunc={'studytime': ['max', 'min']})
```

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

37)Question 37

This code takes from data rows,where both Mjob and Fjob are teacher.The result prints data which displays only ‘teacher’ at Mjob and Fjob .



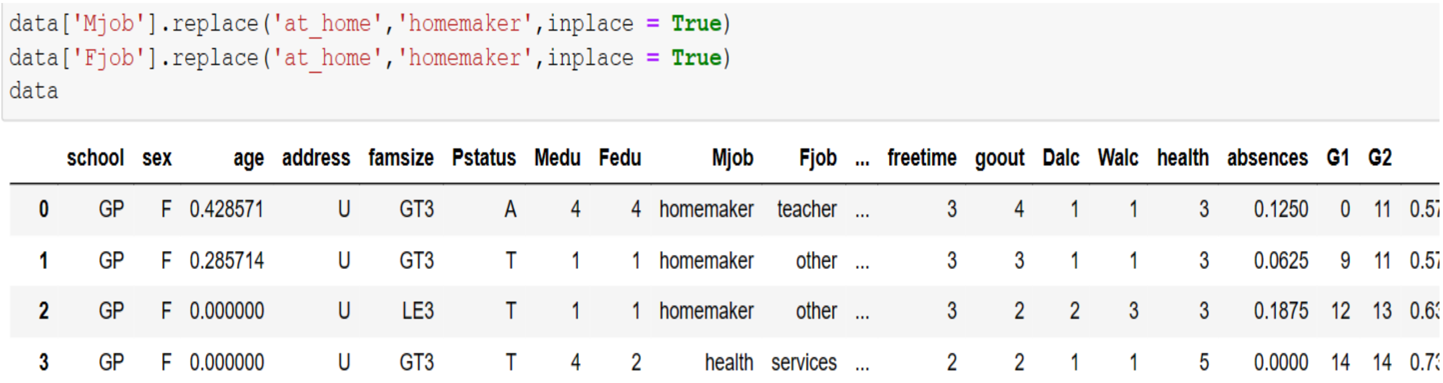
```
has = data[(data['Mjob'] == 'teacher') & (data['Fjob'] == 'teacher')]
print("Students with parents with both works as teacher")
has
```

```
Students with parents with both works as teacher
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
29	GP	M	0.142857	U	GT3	T	4	4	teacher	teacher	...	4	5	5	5	5	0.1250	12	11	0.631579
110	GP	M	0.000000	U	LE3	A	4	4	teacher	teacher	...	5	3	1	1	4	0.1250	13	14	0.736842
115	GP	M	0.142857	U	GT3	T	4	4	teacher	teacher	...	4	4	1	2	5	0.1875	16	14	0.736842
128	GP	M	0.142857	R	GT3	T	4	4	teacher	teacher	...	5	5	2	5	4	0.2500	14	14	0.789474
147	GP	F	0.000000	U	GT3	T	4	4	teacher	teacher	...	3	2	1	1	5	0.1875	13	14	0.736842

38)Question 38

This code takes ‘at_home’ value of Mjob,then Fjob and replaces with ‘homemaker’ value by not changing dataframe.The result returns data with changed values of ‘at_home’

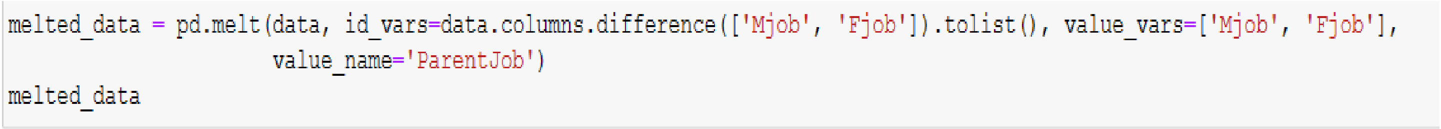


```
data['Mjob'].replace('at_home','homemaker',inplace = True)
data['Fjob'].replace('at_home','homemaker',inplace = True)
data
```

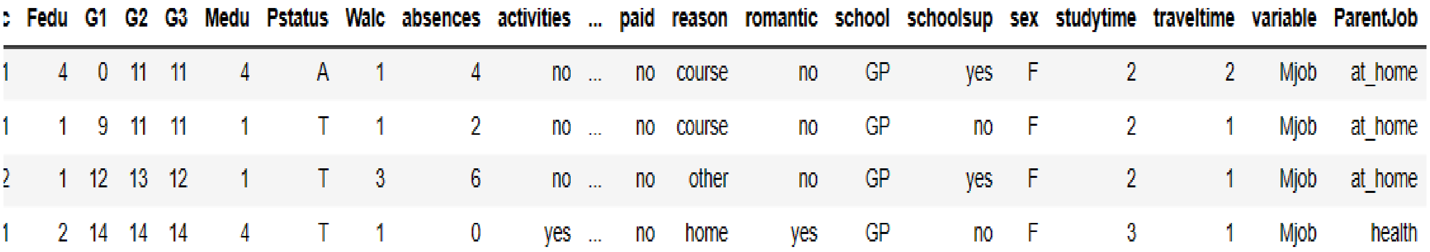
	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	freetime	goout	Dalc	Walc	health	absences	G1	G2	
0	GP	F	0.428571	U	GT3	A	4	4	homemaker	teacher	...	3	4	1	1	3	0.1250	0	11	0.57
1	GP	F	0.285714	U	GT3	T	1	1	homemaker	other	...	3	3	1	1	3	0.0625	9	11	0.57
2	GP	F	0.000000	U	LE3	T	1	1	homemaker	other	...	3	2	2	3	3	0.1875	12	13	0.63
3	GP	F	0.000000	U	GT3	T	4	2	health	services	...	2	2	1	1	5	0.0000	14	14	0.73

39)Question 39

This code creates new dataframe that melts data of two columns into one ‘ParentJob’ column.The result returns new melted data with ParentJob column



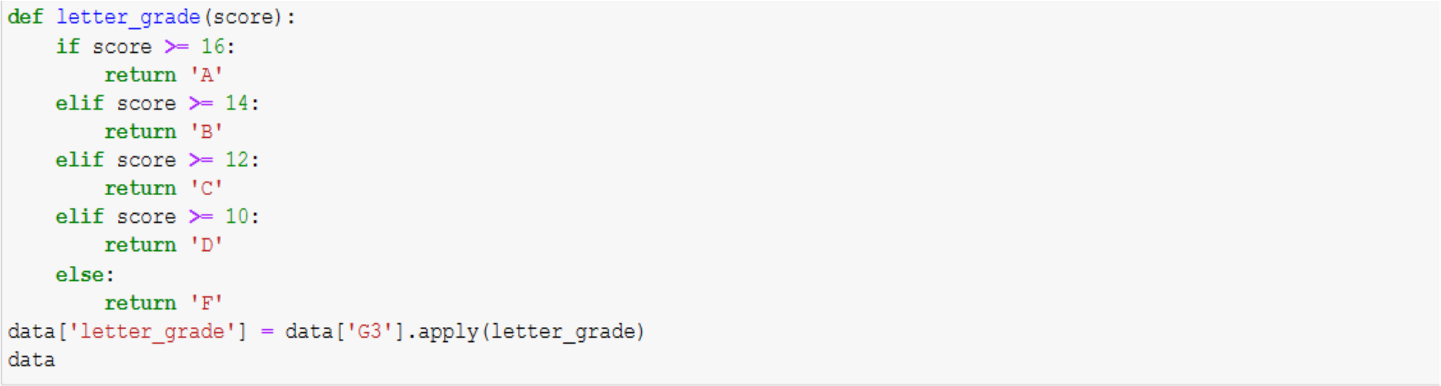
```
melted_data = pd.melt(data, id_vars=data.columns.difference(['Mjob', 'Fjob']).tolist(), value_vars=['Mjob', 'Fjob'],
                      value_name='ParentJob')
melted_data
```



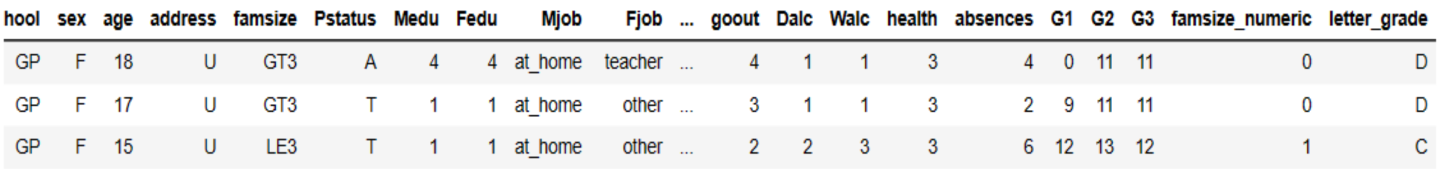
:	Fedu	G1	G2	G3	Medu	Pstatus	Walc	absences	activities	...	paid	reason	romantic	school	schoolsup	sex	studytime	traveltime	variable	ParentJob
1	4	0	11	11	4	A	1	4	no	...	no	course	no	GP	yes	F	2	2	Mjob	at_home
1	1	9	11	11	1	T	1	2	no	...	no	course	no	GP	no	F	2	1	Mjob	at_home
2	1	12	13	12	1	T	3	6	no	...	no	other	no	GP	yes	F	2	1	Mjob	at_home
1	2	14	14	14	4	T	1	0	yes	...	no	home	yes	GP	no	F	3	1	Mjob	health

40)Question 40

This code writes functions which converts scores to letters with if and elif,else statements,then applies this function to new column ‘letter_grade’.The result returns data with new column ,that shows letters for grades.



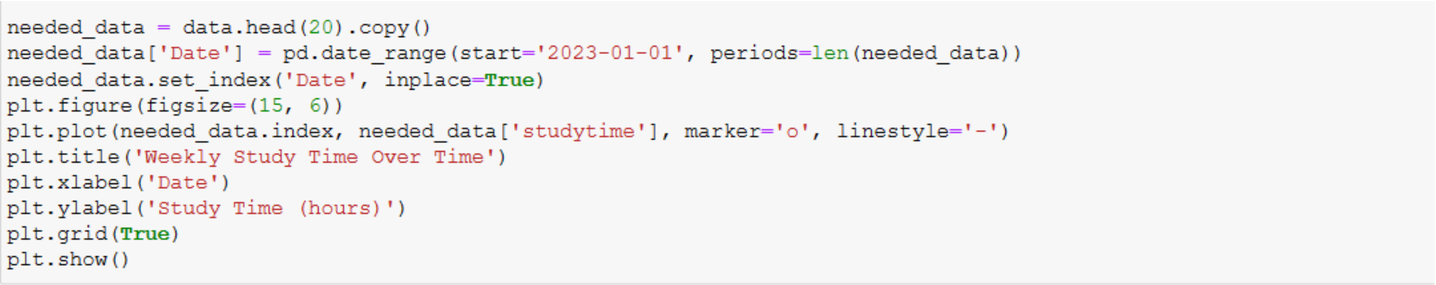
```
def letter_grade(score):
    if score >= 16:
        return 'A'
    elif score >= 14:
        return 'B'
    elif score >= 12:
        return 'C'
    elif score >= 10:
        return 'D'
    else:
        return 'F'
data['letter_grade'] = data['G3'].apply(letter_grade)
data
```



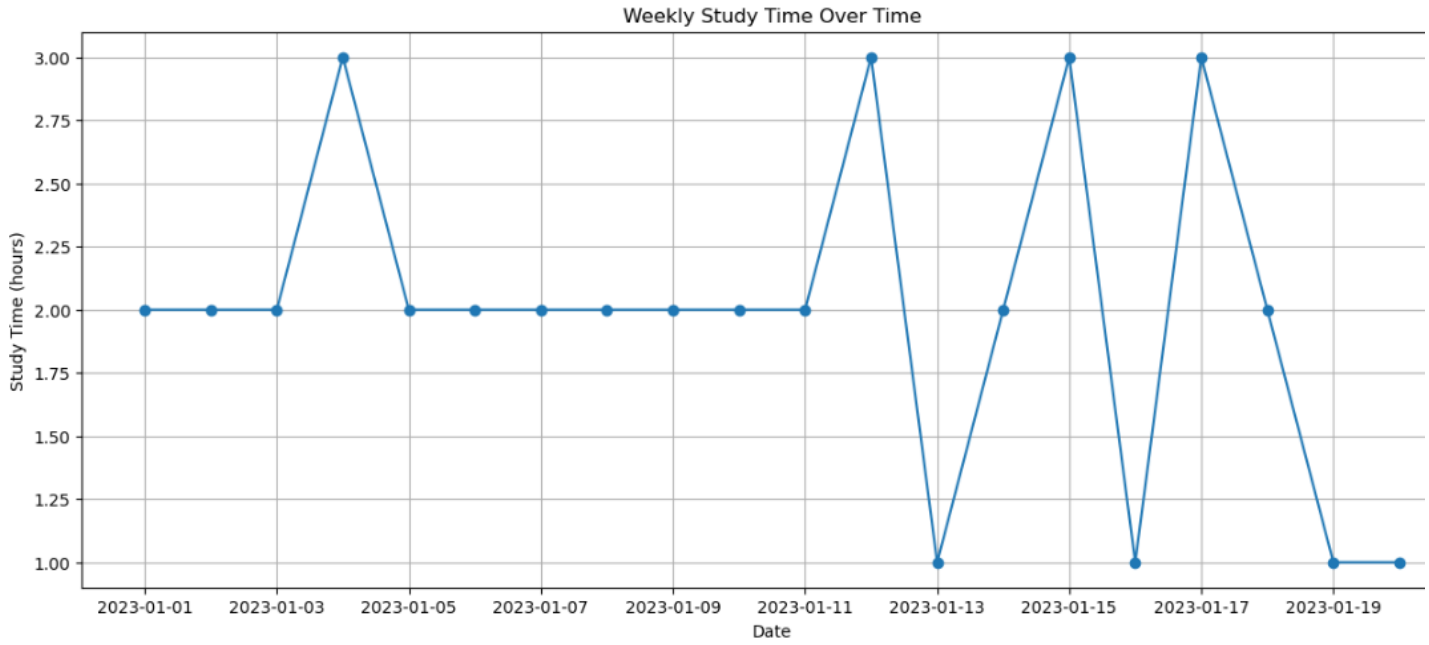
hool	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	goout	Dalc	Walc	health	absences	G1	G2	G3	famsize_numeric	letter_grade
GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	1	1	3	4	0	11	11	0	D
GP	F	17	U	GT3	T	1	1	at_home	other	...	3	1	1	3	2	9	11	11	0	D
GP	F	15	U	LE3	T	1	1	at_home	other	...	2	2	3	3	6	12	13	12	1	C

41)Question 41

This code takes first 20 rows of data,to show good graph,then randomize date for new column,as index takes Date,then creates time series plot with studytime overtime of specific student.The result returns plot

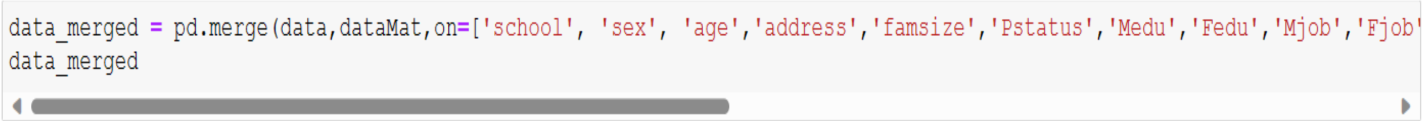


```
needed_data = data.head(20).copy()
needed_data['Date'] = pd.date_range(start='2023-01-01', periods=len(needed_data))
needed_data.set_index('Date', inplace=True)
plt.figure(figsize=(15, 6))
plt.plot(needed_data.index, needed_data['studytime'], marker='o', linestyle='-')
plt.title('Weekly Study Time Over Time')
plt.xlabel('Date')
plt.ylabel('Study Time (hours)')
plt.grid(True)
plt.show()
```

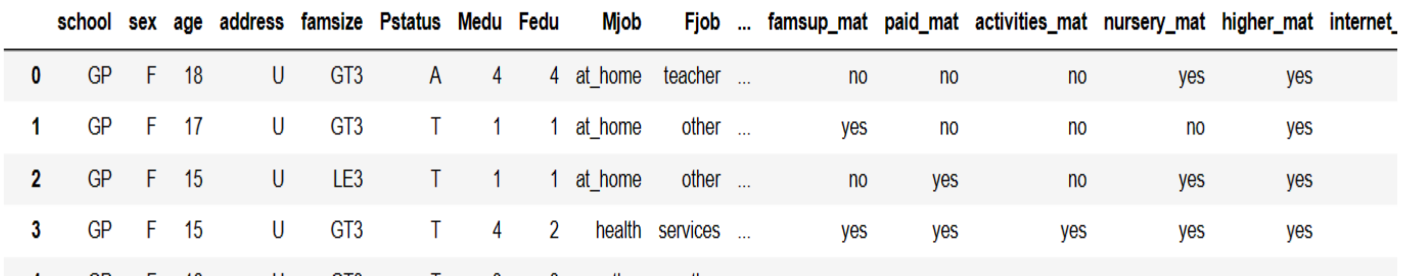


42)Question 42

This code merges two datasets by same columns to show the students who appear in both datasets.The result returns new merged dataset.



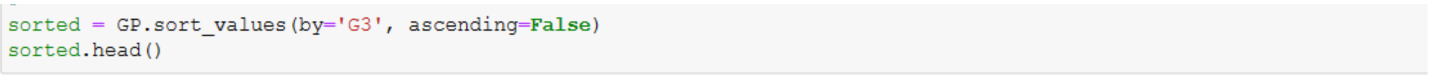
```
data_merged = pd.merge(data,dataMat,on=['school', 'sex', 'age','address','famsize','Pstatus','Medu','Fedu','Mjob','Fjob'
data_merged
```



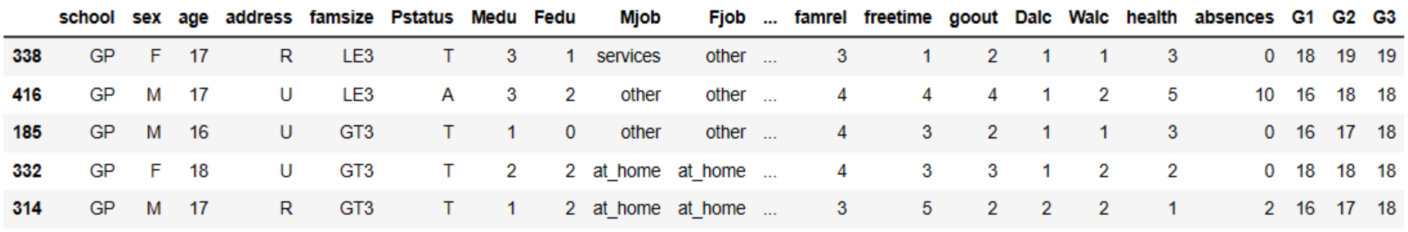
	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famsup_mat	paid_mat	activities_mat	nursery_mat	higher_mat	internet_
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	no	no	no	yes	yes	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	yes	no	no	no	yes	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	no	yes	no	yes	yes	
3	GP	F	15	U	GT3	T	4	2	health	services	...	yes	yes	yes	yes	yes	

43)Question 43

This code takes GP dataframe and uses sort_values to sort in descending order Final grades.The result show top 5 students with high final grades



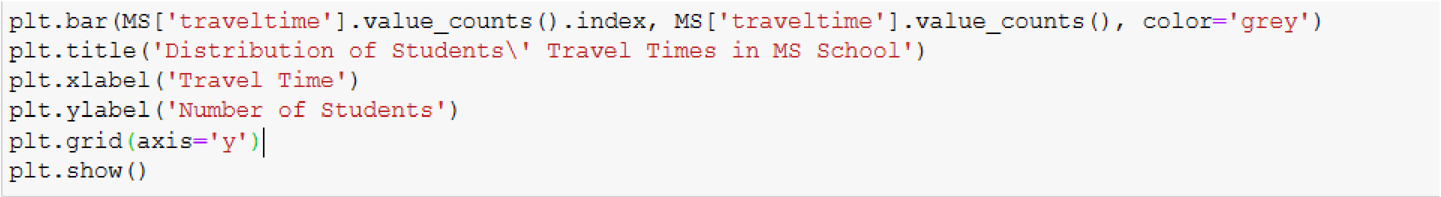
```
sorted = GP.sort_values(by='G3', ascending=False)
sorted.head()
```



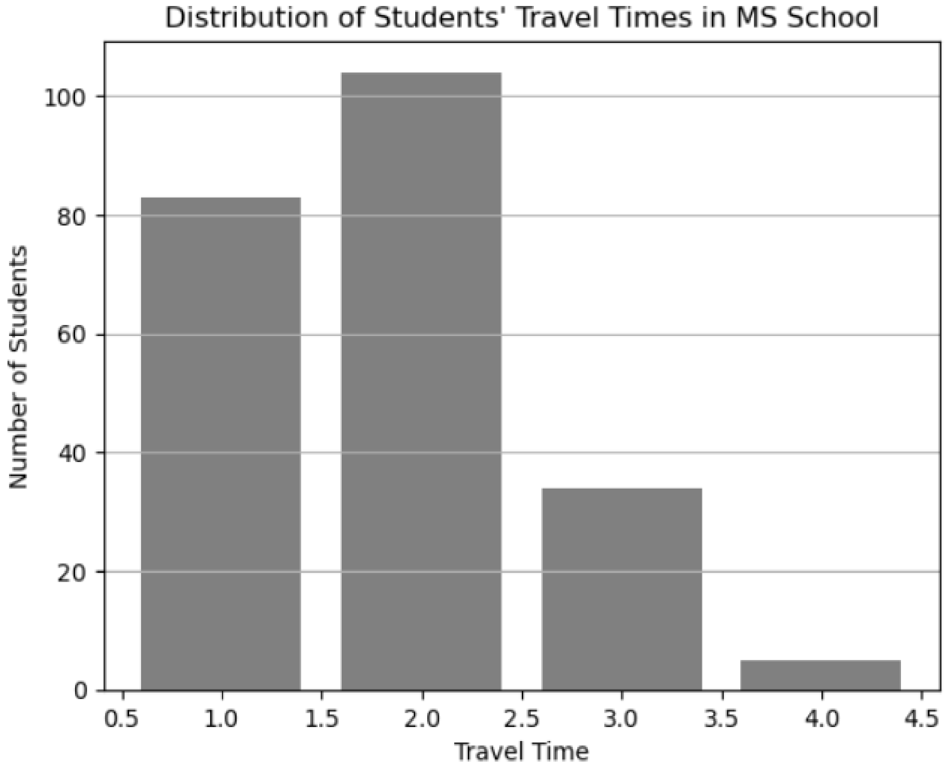
	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
338	GP	F	17	R	LE3	T	3	1	services	other	...	3	1	2	1	1	3	0	18	19	19
416	GP	M	17	U	LE3	A	3	2	other	other	...	4	4	4	1	2	5	10	16	18	18
185	GP	M	16	U	GT3	T	1	0	other	other	...	4	3	2	1	1	3	0	16	17	18
332	GP	F	18	U	GT3	T	2	2	at_home	at_home	...	4	3	3	1	2	2	0	18	18	18
314	GP	M	17	R	GT3	T	1	2	at_home	at_home	...	3	5	2	2	2	1	2	16	17	18

44)Question 44

This code creates bar chart :one side for number of students by calculating values of MS,one side for traveltime indexes.The result shows distribution of student’s travel time by bar chart.

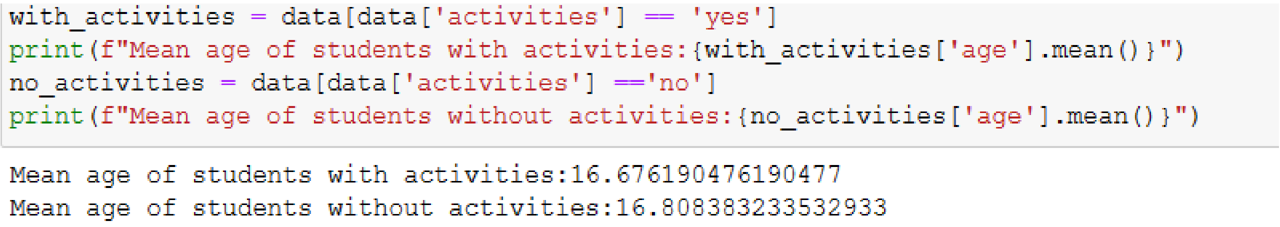


```
plt.bar(MS['traveltime'].value_counts().index, MS['traveltime'].value_counts(), color='grey')
plt.title('Distribution of Students\' Travel Times in MS School')
plt.xlabel('Travel Time')
plt.ylabel('Number of Students')
plt.grid(axis='y')
plt.show()
```



45)Question 45

This code creates two dataframes ,one for students with activies,one for without activities and calculates their age means.The result returns Mean age of these groups.

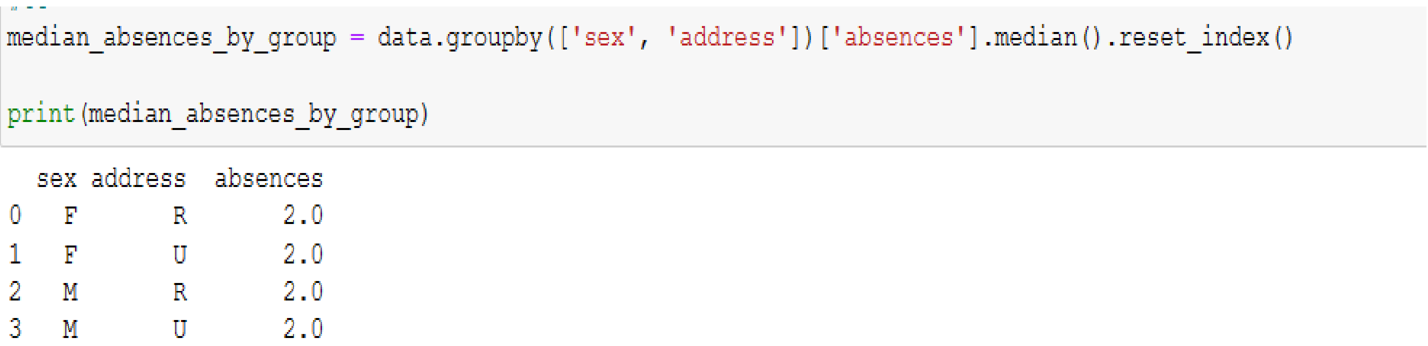


```
with_activities = data[data['activities'] == 'yes']
print(f"Mean age of students with activities:{with_activities['age'].mean()}")
no_activities = data[data['activities'] =='no']
print(f"Mean age of students without activities:{no_activities['age'].mean()}")
```

```
Mean age of students with activities:16.676190476190477
Mean age of students without activities:16.808383233532933
```

46)Question 46

This code groups data by sex and address,finds their median.The result returns absence median of each sex and address groups.

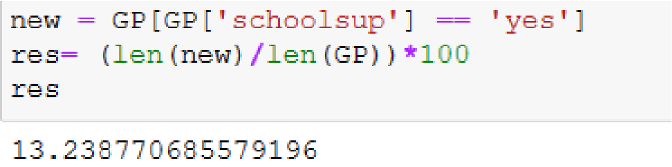


```
median_absences_by_group = data.groupby(['sex', 'address'])['absences'].median().reset_index()

print(median_absences_by_group)
```

```
  sex address  absences
0   F       R       2.0
1   F       U       2.0
2   M       R       2.0
3   M       U       2.0
```

47)Question 47

This code creates new dataframe from GP dataframe,that uses schoolsup.Then calculates probability by formula.The result returns the percentage of students who have schoolsup in the GP

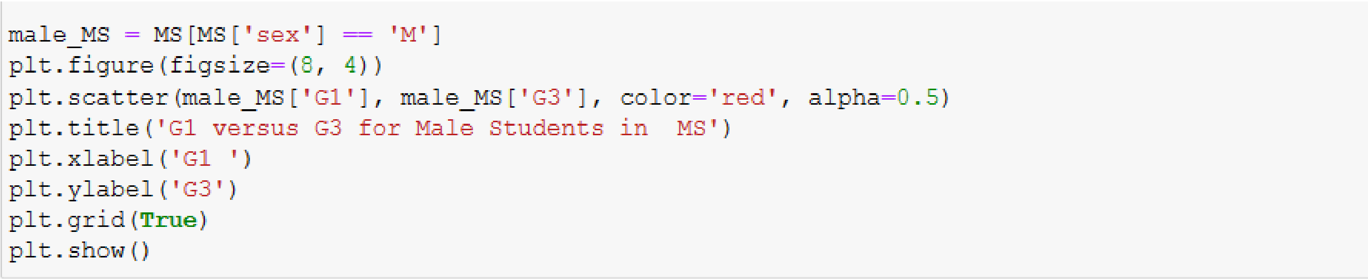


```
new = GP[GP['schoolsup'] == 'yes']
res= (len(new)/len(GP))*100
res
```

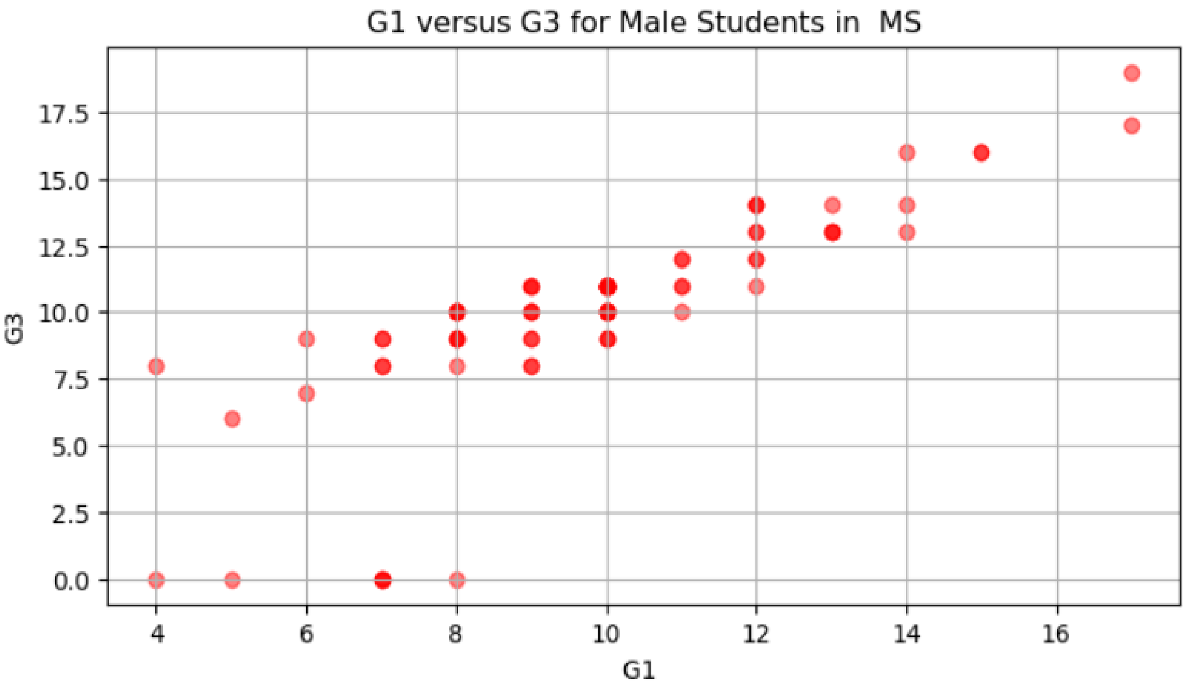
```
13.238770685579196
```

48)Question 48

This code takes male Ms students,creates scatter plot of G1 versus G3 of those students.The result returns scatter plot of grades

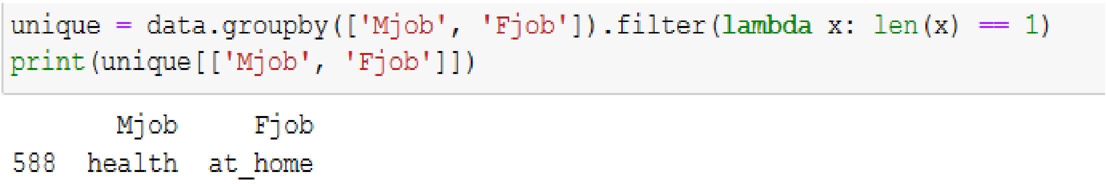


```
male_MS = MS[MS['sex'] == 'M']
plt.figure(figsize=(8, 4))
plt.scatter(male_MS['G1'], male_MS['G3'], color='red', alpha=0.5)
plt.title('G1 versus G3 for Male Students in  MS')
plt.xlabel('G1 ')
plt.ylabel('G3')
plt.grid(True)
plt.show()
```



49)Question 49

This code groups Mjob and Fjob, filters by unique combination.The result returns students containing the combination of Mjob and Fjob is unique.

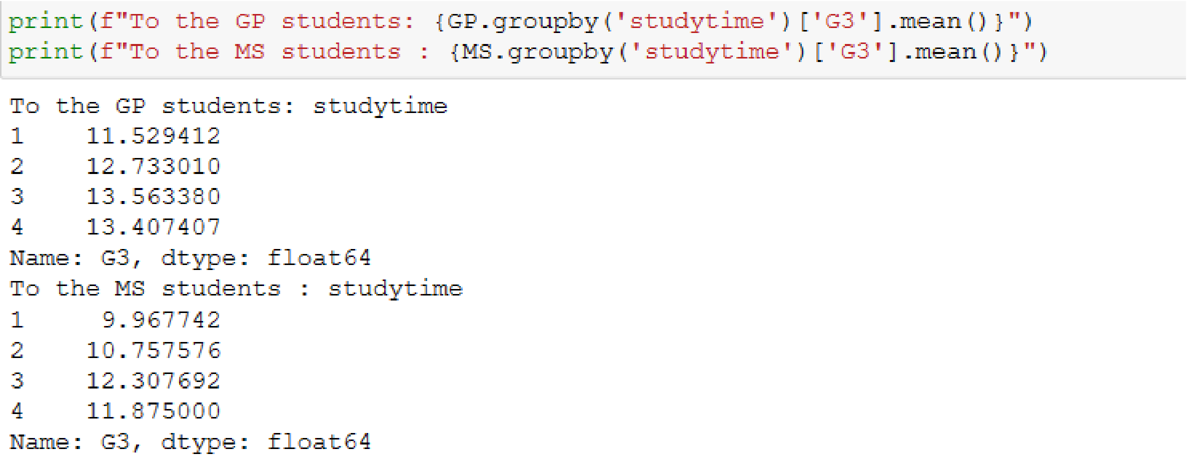


```
unique = data.groupby(['Mjob', 'Fjob']).filter(lambda x: len(x) == 1)
print(unique[['Mjob', 'Fjob']])
```

```
       Mjob     Fjob
588  health  at_home
```

50)Question 50

This code calculates average G3 by grouping studytime of GP and MS students.The result returns GP student’s average G3 by values of studytime,then same things with MS students.



```python
print(f"To the GP students: {GP.groupby('studytime')['G3'].mean()}")
print(f"To the MS students : {MS.groupby('studytime')['G3'].mean()}")
```

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