In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
data = pd.read_csv("student_alcohol.csv")
```

In [3]:

```
data.head()
```

Out[3]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	fre
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	

5 rows × 33 columns

In [4]:

```
data.tail()
```

Out[4]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	f
390	MS	M	20	U	LE3	A	2	2	services	services	...	5	
391	MS	M	17	U	LE3	T	3	1	services	services	...	2	
392	MS	M	21	R	GT3	T	1	1	other	other	...	5	
393	MS	M	18	R	LE3	T	3	2	services	other	...	4	
394	MS	M	19	U	LE3	T	1	1	other	at_home	...	3	

5 rows × 33 columns

In [5]:

```
data.shape
```

Out[5]:

```
(395, 33)
```

In [6]:

```
data.columns
```

Out[6]:

```
Index(['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'F
edu',
       'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime',
       'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery',
       'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'D
alc',
       'Walc', 'health', 'absences', 'G1', 'G2', 'G3'],
      dtype='object')
```

In [7]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   school      395 non-null    object
 1   sex         395 non-null    object
 2   age         395 non-null    int64 
 3   address     395 non-null    object
 4   famsize     395 non-null    object
 5   Pstatus     395 non-null    object
 6   Medu        395 non-null    int64 
 7   Fedu        395 non-null    int64 
 8   Mjob        395 non-null    object
 9   Fjob        395 non-null    object
 10  reason      395 non-null    object
 11  guardian    395 non-null    object
 12  traveltime  395 non-null    int64 
 13  studytime   395 non-null    int64 
 14  failures    395 non-null    int64 
 15  schoolsup   395 non-null    object
 16  famsup      395 non-null    object
 17  paid        395 non-null    object
 18  activities  395 non-null    object
 19  nursery     395 non-null    object
 20  higher      395 non-null    object
 21  internet    395 non-null    object
 22  romantic    395 non-null    object
 23  famrel      395 non-null    int64 
 24  freetime    395 non-null    int64 
 25  goout       395 non-null    int64 
 26  Dalc        395 non-null    int64 
 27  Walc        395 non-null    int64 
 28  health      395 non-null    int64 
 29  absences    395 non-null    int64 
 30  G1          395 non-null    int64 
 31  G2          395 non-null    int64 
 32  G3          395 non-null    int64 
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
```

In [8]:

```
data.describe()
```

Out[8]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	
count	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	3
mean	16.696203	2.749367	2.521519	1.448101	2.035443	0.334177	3.944304	
std	1.276043	1.094735	1.088201	0.697505	0.839240	0.743651	0.896659	
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	
25%	16.000000	2.000000	2.000000	1.000000	1.000000	0.000000	4.000000	
50%	17.000000	3.000000	2.000000	1.000000	2.000000	0.000000	4.000000	
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000	
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	

In [9]:

```
data.isnull().sum()
```

Out[9]:

```
school        0
sex           0
age           0
address       0
famsize       0
Pstatus       0
Medu          0
Fedu          0
Mjob          0
Fjob          0
reason        0
guardian      0
traveltime    0
studytime     0
failures      0
schoolsup     0
famsup        0
paid          0
activities    0
nursery       0
higher        0
internet      0
romantic      0
famrel        0
freetime      0
goout         0
Dalc          0
Walc          0
health        0
absences      0
G1            0
G2            0
G3            0
dtype: int64
```

In [15]:

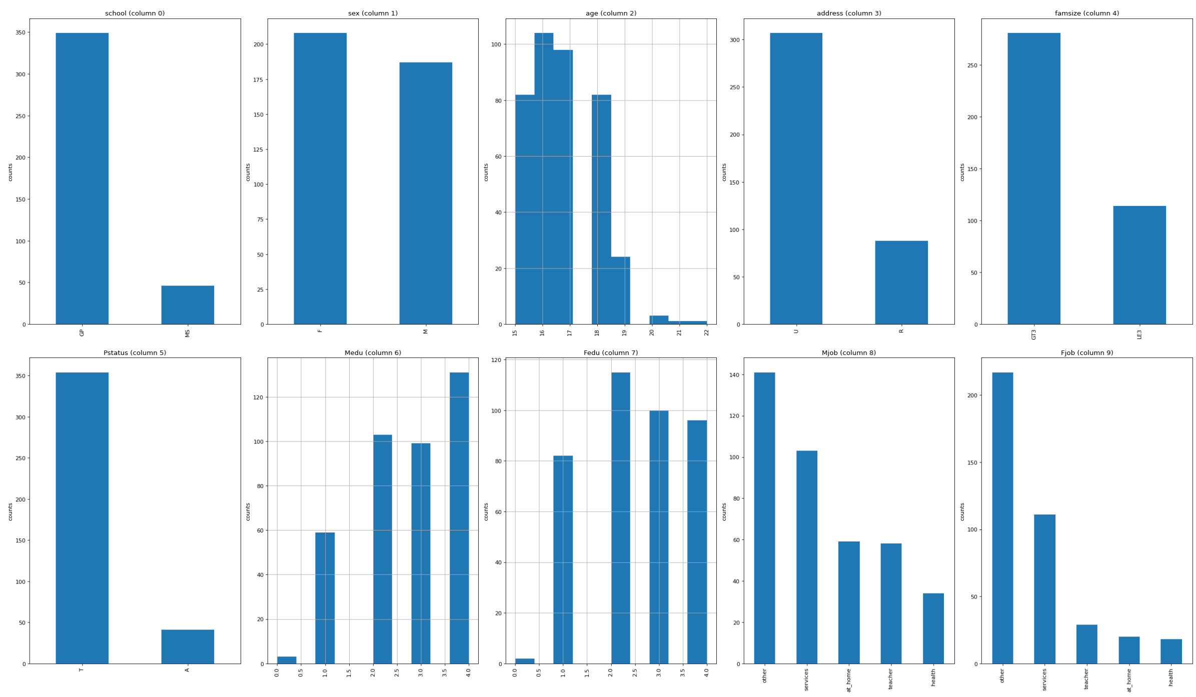
```
import warnings
warnings.filterwarnings('ignore')
```

In [16]:

```
def plotPerColumnDistribution(data, nGraphShown, nGraphPerRow):
    nunique = data.nunique()
    data = data[[col for col in data if nunique[col] > 1 and nunique[col] < 50]]
    nRow, nCol = data.shape
    columnNames = list(data)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGraphRow), dpi = 80, facec
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = data.iloc[:, i]
        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
    plt.show()
```

In [17]:

```
plotPerColumnDistribution(data, 10, 5)
```



In [22]:

```
data.corr()
```

Out[22]:

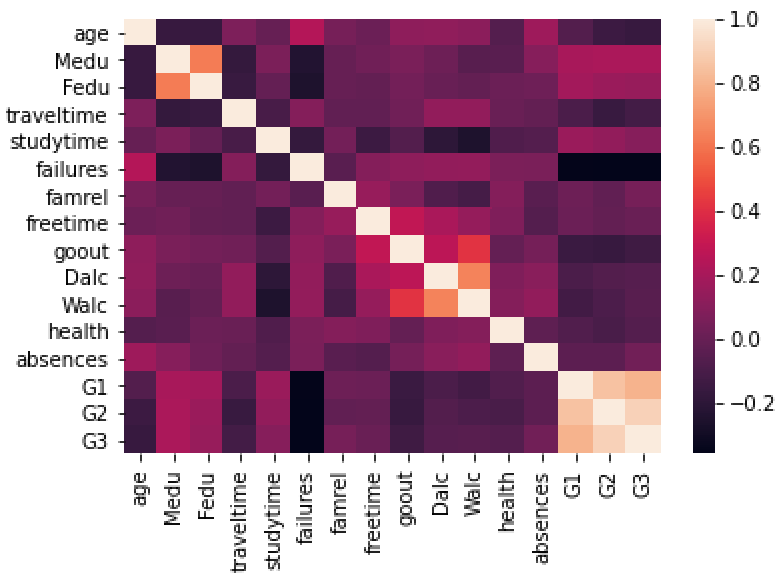
	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetim
age	1.000000	-0.163658	-0.163438	0.070641	-0.004140	0.243665	0.053940	0.01643
Medu	-0.163658	1.000000	0.623455	-0.171639	0.064944	-0.236680	-0.003914	0.03089
Fedu	-0.163438	0.623455	1.000000	-0.158194	-0.009175	-0.250408	-0.001370	-0.01284
traveltime	0.070641	-0.171639	-0.158194	1.000000	-0.100909	0.092239	-0.016808	-0.01702
studytime	-0.004140	0.064944	-0.009175	-0.100909	1.000000	-0.173563	0.039731	-0.14319
failures	0.243665	-0.236680	-0.250408	0.092239	-0.173563	1.000000	-0.044337	0.09198
famrel	0.053940	-0.003914	-0.001370	-0.016808	0.039731	-0.044337	1.000000	0.15070
freetime	0.016434	0.030891	-0.012846	-0.017025	-0.143198	0.091987	0.150701	1.00000
goout	0.126964	0.064094	0.043105	0.028540	-0.063904	0.124561	0.064568	0.28501
Dalc	0.131125	0.019834	0.002386	0.138325	-0.196019	0.136047	-0.077594	0.20900
Walc	0.117276	-0.047123	-0.012631	0.134116	-0.253785	0.141962	-0.113397	0.14782
health	-0.062187	-0.046878	0.014742	0.007501	-0.075616	0.065827	0.094056	0.07573
absences	0.175230	0.100285	0.024473	-0.012944	-0.062700	0.063726	-0.044354	-0.05807
G1	-0.064081	0.205341	0.190270	-0.093040	0.160612	-0.354718	0.022168	0.01261
G2	-0.143474	0.215527	0.164893	-0.153198	0.135880	-0.355896	-0.018281	-0.01377
G3	-0.161579	0.217147	0.152457	-0.117142	0.097820	-0.360415	0.051363	0.01130

In [23]:

```
sns.heatmap(data.corr())
```

Out[23]:

<AxesSubplot:>

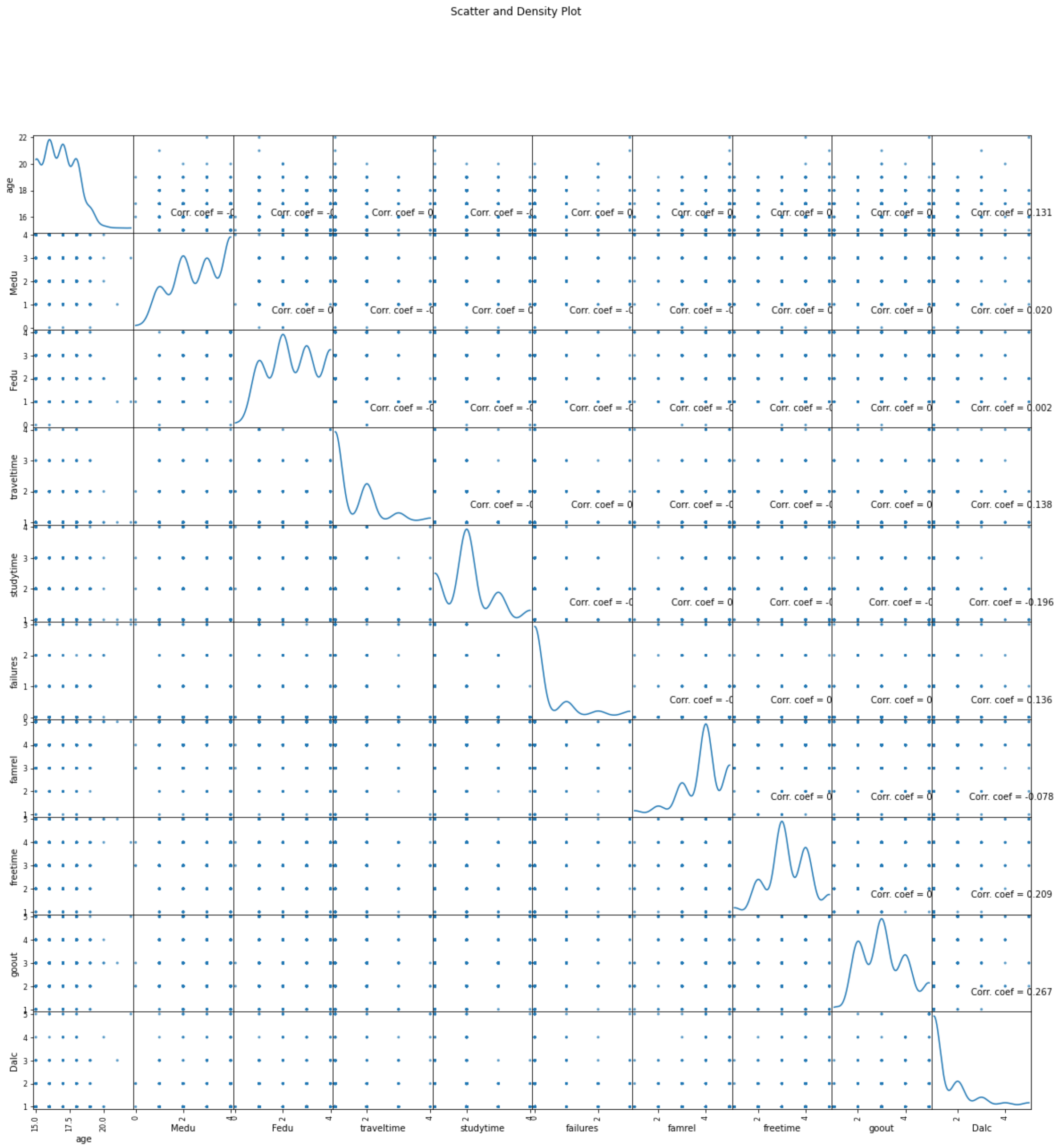


In [24]:

```
def plotScatterMatrix(data, plotSize, textSize):
    data = data.select_dtypes(include =[np.number])
    data = data.dropna('columns')
    data = data[[col for col in data if data[col].nunique() > 1]]
    columnNames = list(data)
    if len(columnNames) > 10:
        columnNames = columnNames[:10]
    data = data[columnNames]
    ax = pd.plotting.scatter_matrix(data, alpha=0.75, figsize=[plotSize, plotSize], diag
    corrs = data.corr().values
    for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
        ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2), xycoords='axes
    plt.suptitle('Scatter and Density Plot')
    plt.show()
```

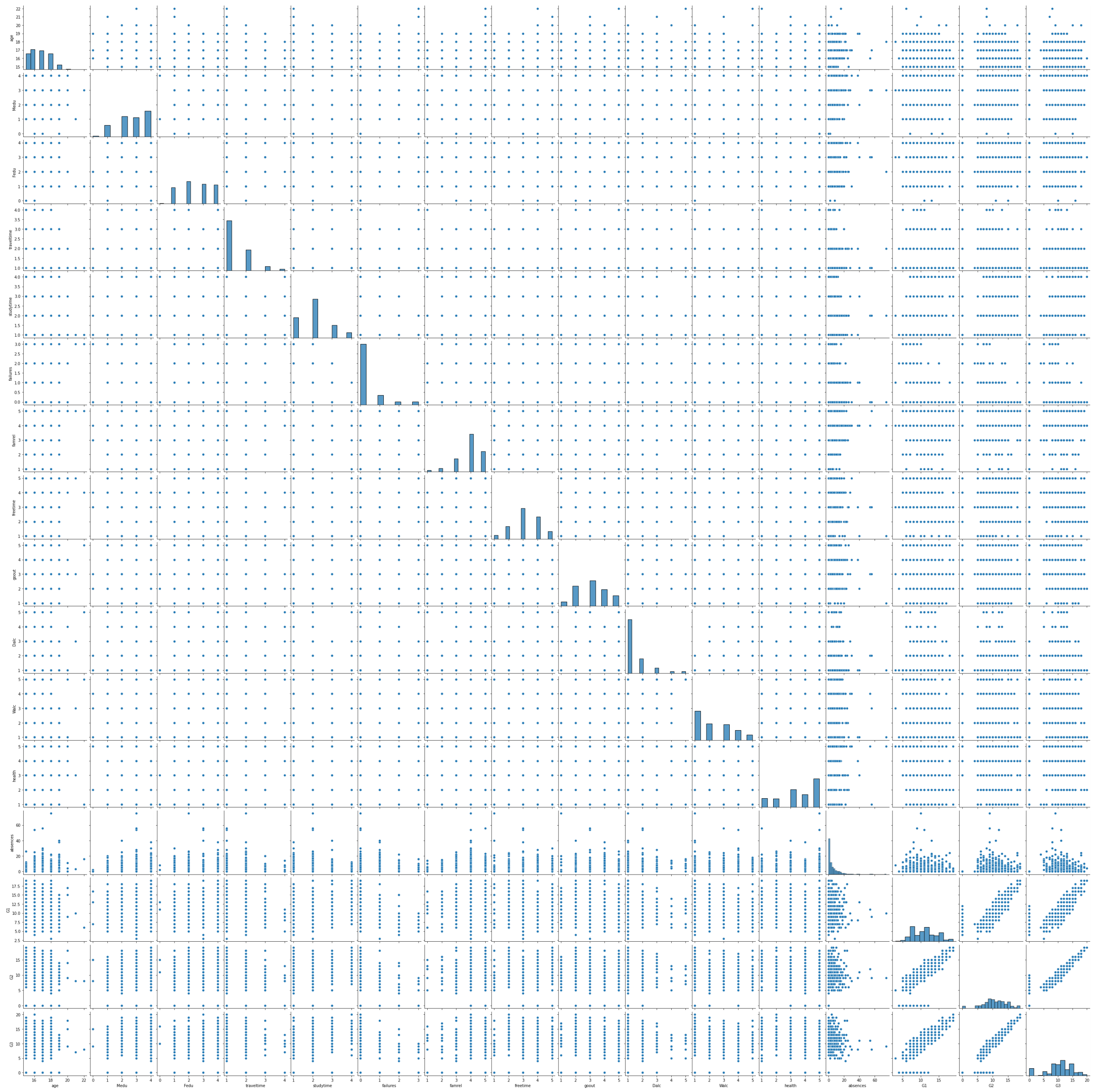
In [25]:

```
plotScatterMatrix(data, 20, 10)
```



In [26]:

```
sns.pairplot(data);
```



In [27]:

```
data.nunique()
```

Out[27]:

```
school         2
sex            2
age            8
address        2
famsize        2
Pstatus        2
Medu           5
Fedu           5
Mjob           5
Fjob           5
reason         4
guardian       3
traveltime     4
studytime      4
failures       4
schoolsup      2
famsup         2
paid           2
activities     2
nursery        2
higher         2
internet       2
romantic       2
famrel         5
freetime       5
goout          5
Dalc           5
Walc           5
health         5
absences      34
G1            17
G2            17
G3            18
dtype: int64
```

In [32]:

```
data['sex'].value_counts()
```

Out[32]:

```
F    208
M    187
Name: sex, dtype: int64
```

In [28]:

```
data['activities'].value_counts()
```

Out[28]:

```
yes    201
no     194
Name: activities, dtype: int64
```

In [29]:

```
data['internet'].value_counts()
```

Out[29]:

```
yes    329
no      66
Name: internet, dtype: int64
```

In [31]:

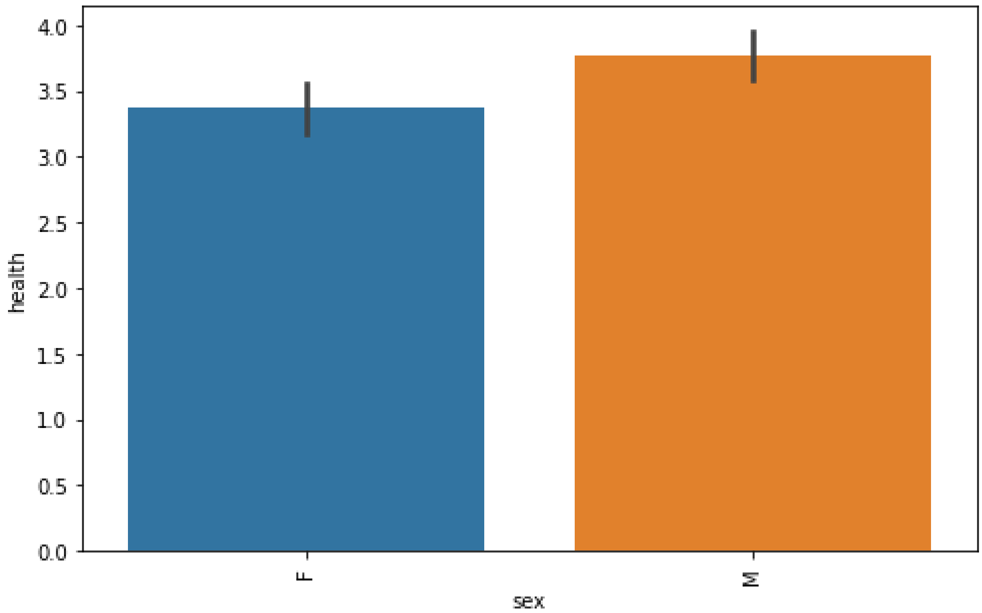
```
data['romantic'].value_counts()
```

Out[31]:

```
no     263
yes    132
Name: romantic, dtype: int64
```

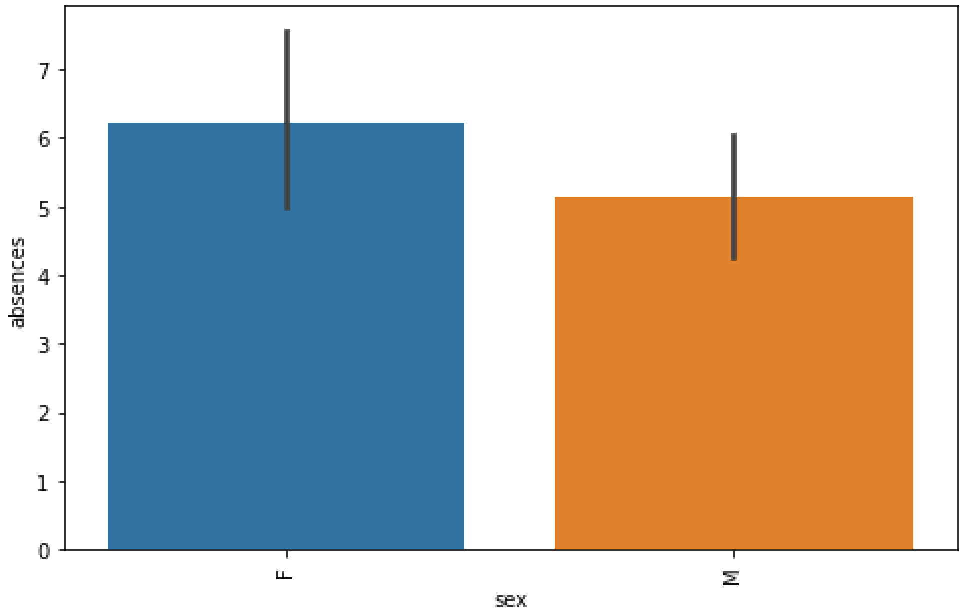
In [34]:

```
plt.figure(figsize=(8,5))
sns.barplot(x = data.sex, y = data.health, data = data)
plt.xticks(rotation = 90)
plt.show()
```



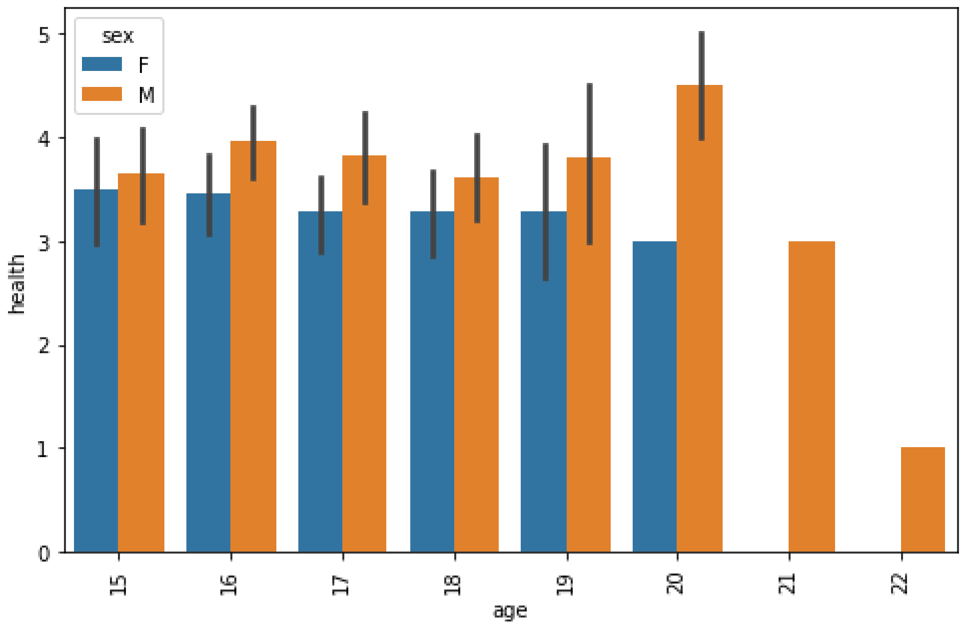
In [35]:

```
plt.figure(figsize=(8,5))
sns.barplot(x = data.sex, y = data.absences, data = data)
plt.xticks(rotation = 90)
plt.show()
```



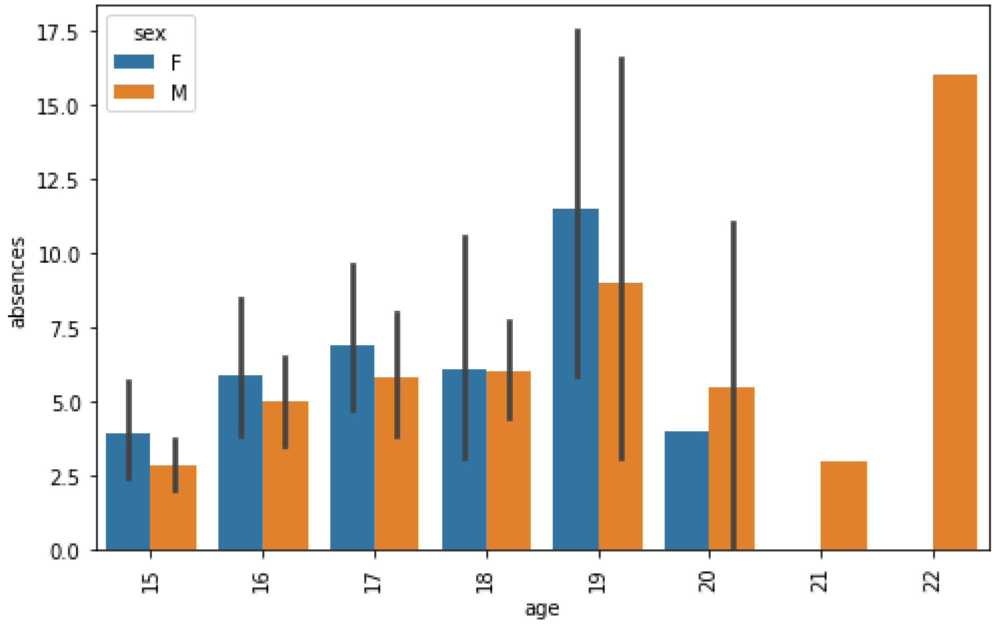
In [36]:

```
plt.figure(figsize=(8,5))
sns.barplot(x = data.age, y = data.health, data = data, hue = data.sex)
plt.xticks(rotation = 90)
plt.show()
```



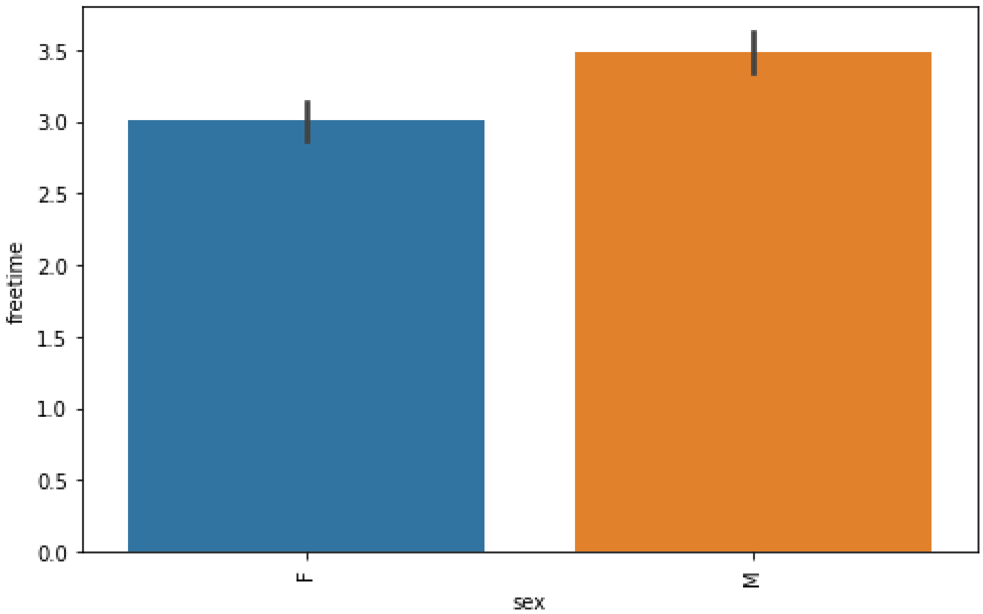
In [37]:

```
plt.figure(figsize=(8,5))
sns.barplot(x = data.age, y = data.absences, data = data, hue = data.sex)
plt.xticks(rotation = 90)
plt.show()
```



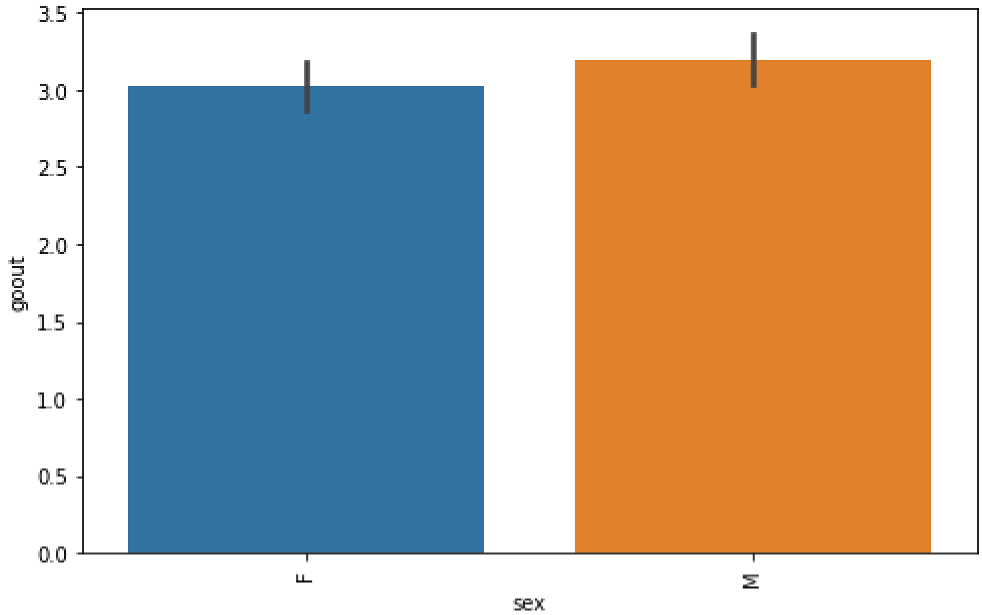
In [38]:

```
plt.figure(figsize=(8,5))
sns.barplot(x = data.sex, y = data.freetime, data = data)
plt.xticks(rotation = 90)
plt.show()
```



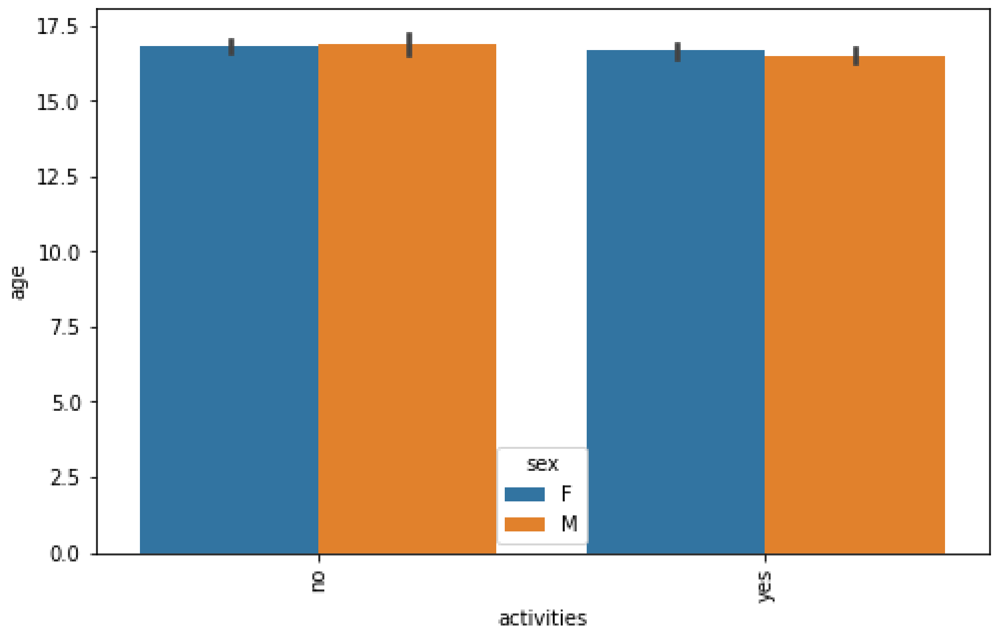
In [39]:

```
plt.figure(figsize=(8,5))
sns.barplot(x = data.sex, y = data.goout, data = data)
plt.xticks(rotation = 90)
plt.show()
```



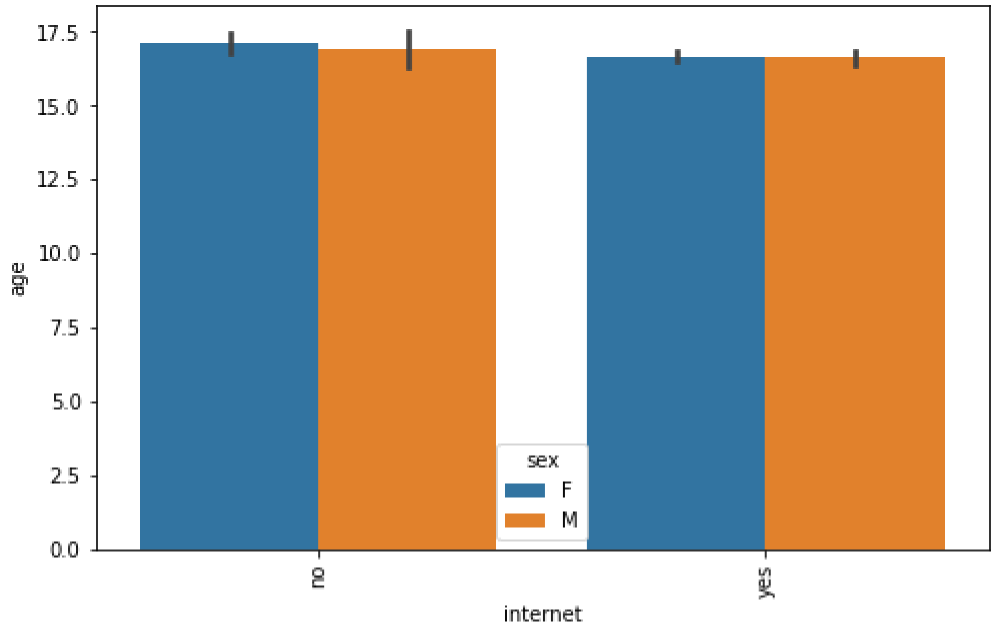
In [41]:

```
plt.figure(figsize=(8,5))
sns.barplot(x = data.activities, y = data.age, data = data, hue = data.sex)
plt.xticks(rotation = 90)
plt.show()
```



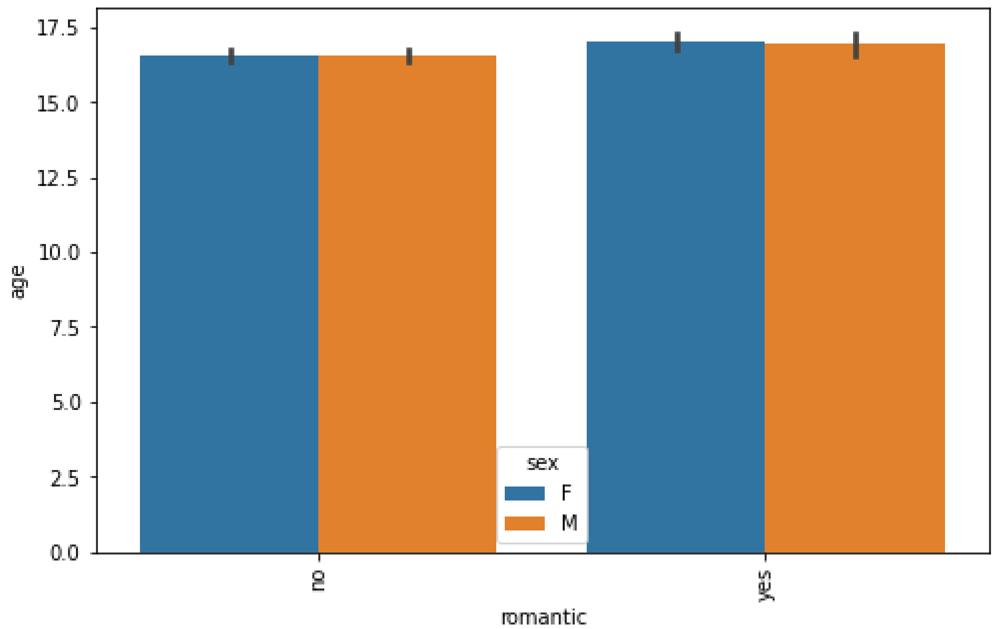
In [42]:

```
plt.figure(figsize=(8,5))
sns.barplot(x = data.internet, y = data.age, data = data, hue = data.sex)
plt.xticks(rotation = 90)
plt.show()
```



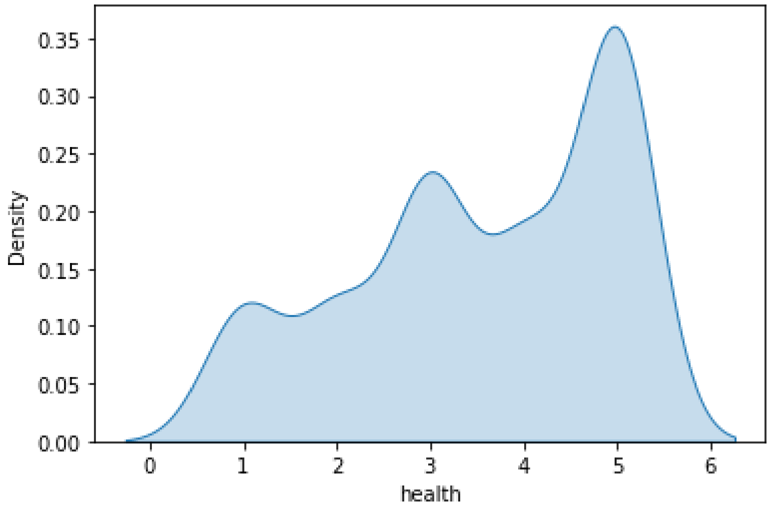
In [43]:

```
plt.figure(figsize=(8,5))
sns.barplot(x = data.romantic, y = data.age, data = data, hue = data.sex)
plt.xticks(rotation = 90)
plt.show()
```



In [44]:

```
sns.kdeplot(x='health', data=data, shade=True);
```



In [45]:

```
data.columns
```

Out[45]:

```
Index(['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'F
edu',
       'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime',
       'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery',
       'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'D
alc',
       'Walc', 'health', 'absences', 'G1', 'G2', 'G3'],
      dtype='object')
```

In [46]:

```
new_data=data.drop(columns=['school', 'sex','address', 'famsize', 'Pstatus', 'Medu', 'Fe
       'Mjob', 'Fjob', 'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activities
       'higher', 'internet', 'romantic'])
```

In [47]:

```
new_data.head()
```

Out[47]:

	age	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc	health	absences	G
0	18	2	2	0	4	3	4	1	1	3	6	
1	17	1	2	0	5	3	3	1	1	3	4	
2	15	1	2	3	4	3	2	2	3	3	10	
3	15	1	3	0	3	2	2	1	1	5	2	1
4	16	1	2	0	4	3	2	1	2	5	4	

In [72]:

```
x = new_data.drop(['G1','G2', 'G3'], axis = 1)
y = new_data.G1
```

In [73]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [74]:

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
```

In [75]:

```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[75]:

```
LinearRegression()
```

In [76]:

```
y_pred = model.predict(X_test)
```

In [77]:

```
print("Training Accuracy :", model.score(X_train, y_train))
print("Testing Accuracy :", model.score(X_test, y_test))
```

```
Training Accuracy : 0.17038753061066803
Testing Accuracy : 0.03497866263159799
```

In [78]:

```
x = new_data.drop(['G1','G2', 'G3'], axis = 1)
y = new_data.G2
```

In [79]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [80]:

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
```

In [81]:

```
model1 = LinearRegression()
model1.fit(X_train, y_train)
```

Out[81]:

```
LinearRegression()
```

In [82]:

```
y_pred = model1.predict(X_test)
```

In [83]:

```
print("Training Accuracy :", model1.score(X_train, y_train))
print("Testing Accuracy :", model1.score(X_test, y_test))
```

```
Training Accuracy : 0.17640743035507134
Testing Accuracy : 0.1321195349551918
```

In [84]:

```
x = new_data.drop(['G1','G2', 'G3'], axis = 1)
y = new_data.G3
```

In [85]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [86]:

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
```

In [87]:

```
model2 = LinearRegression()
model2.fit(X_train, y_train)
```

Out[87]:

```
LinearRegression()
```

In [88]:

```
y_pred = model2.predict(X_test)
```

In [89]:

```
print("Training Accuracy :", model2.score(X_train, y_train))
print("Testing Accuracy :", model2.score(X_test, y_test))
```

```
Training Accuracy : 0.2062699218905163
Testing Accuracy : 0.03259803470131295
```