ASSIGNMENT-2 -Data Wrangling 2

Kaustubh Shrikant Kabra

ERP Number :- 38

TE Comp 1

Import libraries

In [79]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [80]:
```
df=pd.read_csv("student-por.csv")
```

In [81]:
```
df.head()
```

Out[81]:

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

5 rows × 33 columns

1 school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
2 sex - student's sex (binary: 'F' - female or 'M' - male)
3 age - student's age (numeric: from 15 to 22)
4 address - student's home address type (binary: 'U' - urban or 'R' - rural)
5 famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)
6 Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
7 Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education)
8 Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education)
9 Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
10 Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or

police), 'at_home' or 'other')
11 reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')
12 guardian - student's guardian (nominal: 'mother', 'father' or 'other')
13 traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
14 studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
15 failures - number of past class failures (numeric: n if 1<=n<3, else 4)
16 schoolsup - extra educational support (binary: yes or no)
17 famsup - family educational support (binary: yes or no)
18 paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
19 activities - extra-curricular activities (binary: yes or no)
20 nursery - attended nursery school (binary: yes or no)
21 higher - wants to take higher education (binary: yes or no)
22 internet - Internet access at home (binary: yes or no)
23 romantic - with a romantic relationship (binary: yes or no)
24 famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
25 freetime - free time after school (numeric: from 1 - very low to 5 - very high)
26 goout - going out with friends (numeric: from 1 - very low to 5 - very high)
27 Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28 Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
29 health - current health status (numeric: from 1 - very bad to 5 - very good)
30 absences - number of school absences (numeric: from 0 to 93)

these grades are related with the course subject, Math or Portuguese:

31 G1 - first period grade (numeric: from 0 to 20)
31 G2 - second period grade (numeric: from 0 to 20)
32 G3 - final grade (numeric: from 0 to 20, output target)

Scanning For Missing Values

In [82]:
```
df.isnull().sum()
```

Out[82]:
```
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 [83]:
```
df.duplicated().sum()
```

Out[83]:
```
0
```

In [84]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 649 entries, 0 to 648
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   school      649 non-null    object
 1   sex         649 non-null    object
 2   age         649 non-null    int64 
 3   address     649 non-null    object
 4   famsize     649 non-null    object
 5   Pstatus     649 non-null    object
 6   Medu        649 non-null    int64 
 7   Fedu        649 non-null    int64 
 8   Mjob        649 non-null    object
 9   Fjob        649 non-null    object
 10  reason      649 non-null    object
 11  guardian    649 non-null    object
 12  traveltime  649 non-null    int64 
 13  studytime   649 non-null    int64 
 14  failures    649 non-null    int64 
 15  schoolsup   649 non-null    object
 16  famsup      649 non-null    object
 17  paid        649 non-null    object
 18  activities  649 non-null    object
 19  nursery     649 non-null    object
 20  higher      649 non-null    object
```

```
 21  internet    649 non-null    object
 22  romantic    649 non-null    object
 23  famrel      649 non-null    int64
 24  freetime    649 non-null    int64
 25  goout       649 non-null    int64
 26  Dalc        649 non-null    int64
 27  Walc        649 non-null    int64
 28  health      649 non-null    int64
 29  absences    649 non-null    int64
 30  G1          649 non-null    int64
 31  G2          649 non-null    int64
 32  G3          649 non-null    int64
dtypes: int64(16), object(17)
memory usage: 167.4+ KB
```

In [85]:
```
df.describe()
```

Out[85]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime
count	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000
mean	16.744222	2.514638	2.306626	1.568567	1.930663	0.221880	3.930663	3.180277
std	1.218138	1.134552	1.099931	0.748660	0.829510	0.593235	0.955717	1.051093
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000
25%	16.000000	2.000000	1.000000	1.000000	1.000000	0.000000	4.000000	3.000000
50%	17.000000	2.000000	2.000000	1.000000	2.000000	0.000000	4.000000	3.000000
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000	4.000000
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	5.000000

In [86]:
```
data=df.select_dtypes(include='int64')
```

In [87]:
```
data.head()
```

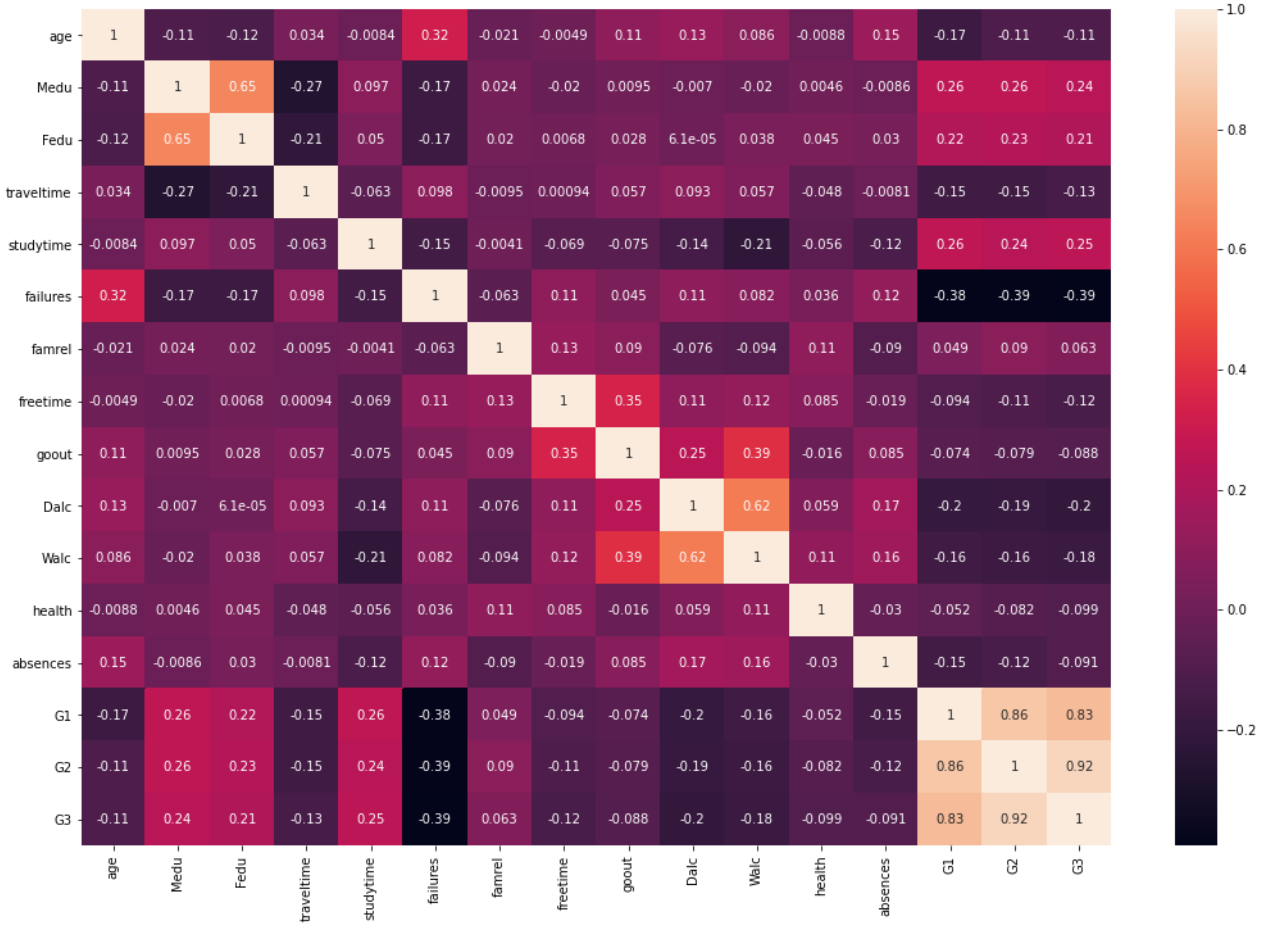
Out[87]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc	health	abs
0	18	4	4	2	2	0	4	3	4	1	1	3	
1	17	1	1	1	2	0	5	3	3	1	1	3	
2	15	1	1	1	2	0	4	3	2	2	3	3	
3	15	4	2	1	3	0	3	2	2	1	1	5	
4	16	3	3	1	2	0	4	3	2	1	2	5	

In [88]:
```
columns=data.columns
```

In [89]:

```
plt.figure(figsize=(18,12))
sns.heatmap(df.corr(),cbar=True,annot=True)
plt.show()
```



StandardScaler

StandardScaler follows Standard Normal Distribution (SND). Therefore, it makes mean = 0 and scales the data to unit variance.

In [90]:
```
from sklearn.preprocessing import StandardScaler
Scaler=StandardScaler()
```

In [91]:
```
copy_data=data.copy()
copy_data=Scaler.fit_transform(data,y=None)
copy_data=pd.DataFrame(copy_data,columns=data.columns)
copy_data.head()
```

Out[91]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	
0	1.031695	1.310216	1.540715	0.576718	0.083653	-0.374305	0.072606	-0.171647	0.693785	-0.
1	0.210137	-1.336039	-1.188832	-0.760032	0.083653	-0.374305	1.119748	-0.171647	-0.157380	-0.
2	-1.432980	-1.336039	-1.188832	-0.760032	0.083653	-0.374305	0.072606	-0.171647	-1.008546	0.

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	
3	-1.432980	1.310216	-0.278983	-0.760032	1.290114	-0.374305	-0.974536	-1.123771	-1.008546	-0.
4	-0.611422	0.428131	0.630866	-0.760032	0.083653	-0.374305	0.072606	-0.171647	-1.008546	-0.

In [92]:
```
copy_data.describe()
```

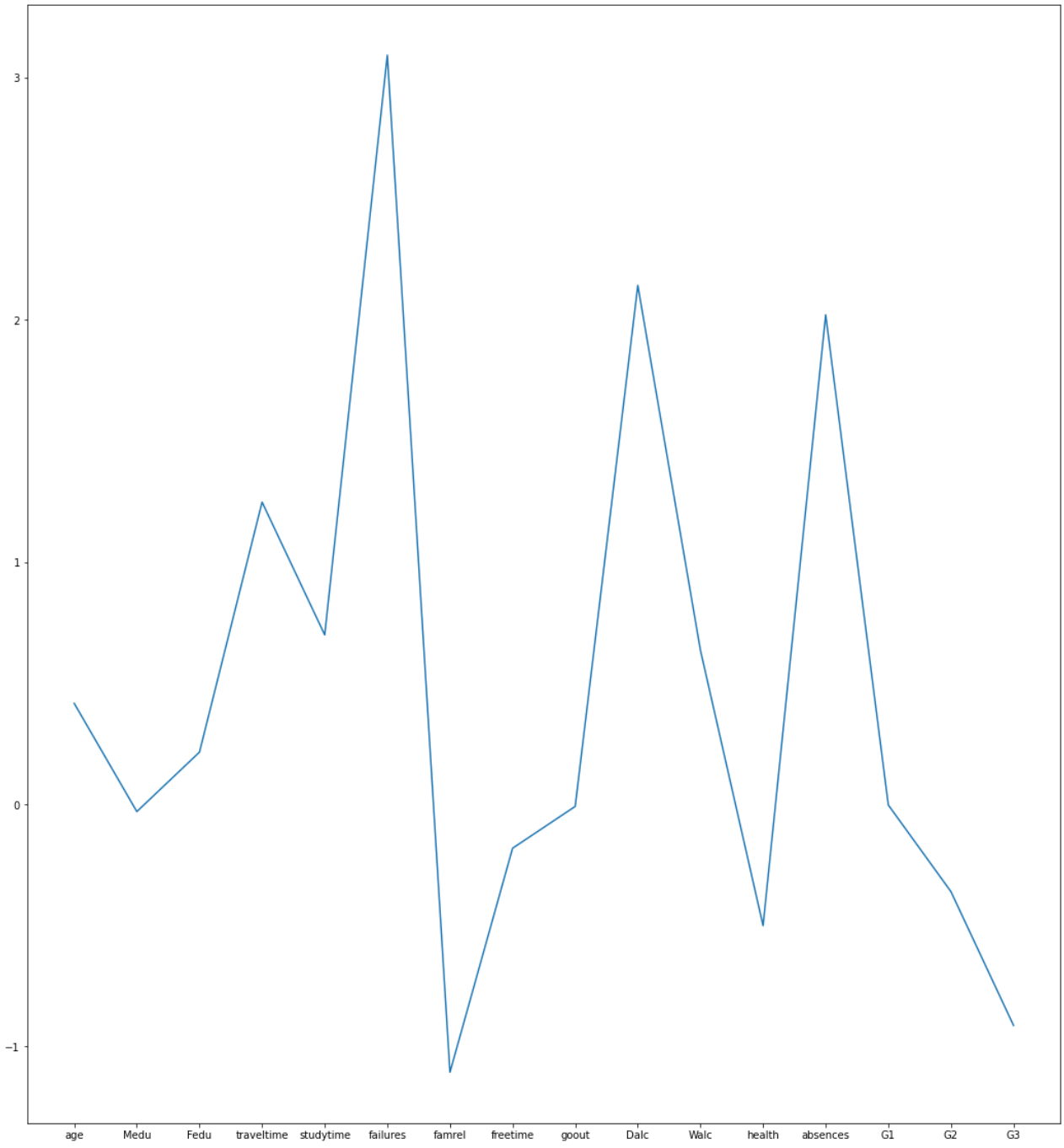
Out[92]:

	age	Medu	Fedu	traveltime	studytime	failures	
count	6.490000e+02	6.490000e+02	6.490000e+02	6.490000e+02	6.490000e+02	6.490000e+02	6.490
mean	-5.053311e-16	-5.787187e-16	2.468493e-16	-7.349026e-16	1.662769e-16	-5.109763e-16	-2.047
std	1.000771e+00	1.000771e+00	1.000771e+00	1.000771e+00	1.000771e+00	1.000771e+00	1.000
min	-1.432980e+00	-2.218124e+00	-2.098682e+00	-7.600319e-01	-1.122808e+00	-3.743051e-01	-3.068
25%	-6.114218e-01	-4.539544e-01	-1.188832e+00	-7.600319e-01	-1.122808e+00	-3.743051e-01	7.260
50%	2.101367e-01	-4.539544e-01	-2.789831e-01	-7.600319e-01	8.365295e-02	-3.743051e-01	7.260
75%	1.031695e+00	1.310216e+00	6.308662e-01	5.767180e-01	8.365295e-02	-3.743051e-01	1.119
max	4.317929e+00	1.310216e+00	1.540715e+00	3.250218e+00	2.496576e+00	4.686612e+00	1.119

Skewness

refers to a distortion or asymmetry that deviates from the symmetrical bell curve, or normal distribution, in a set of data.

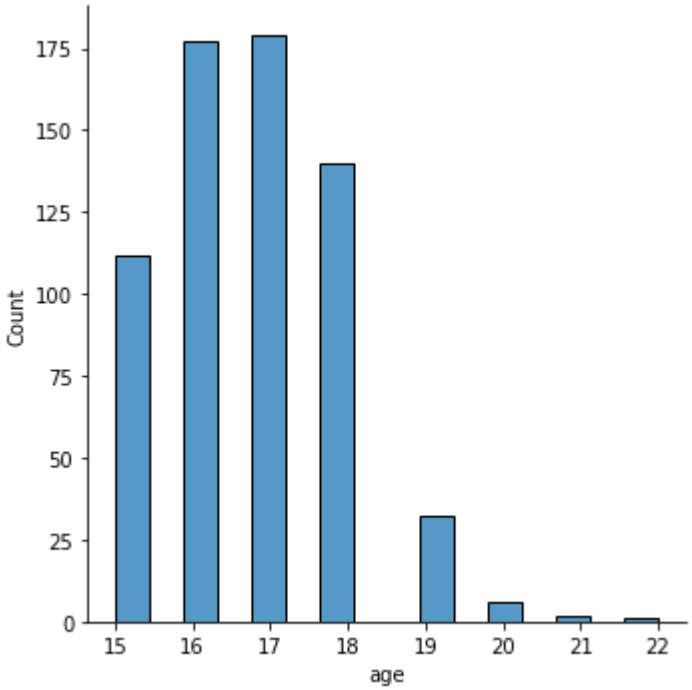
In [93]:
```
plt.figure(figsize=(18,20))
plt.plot(data.skew())
plt.show()
```

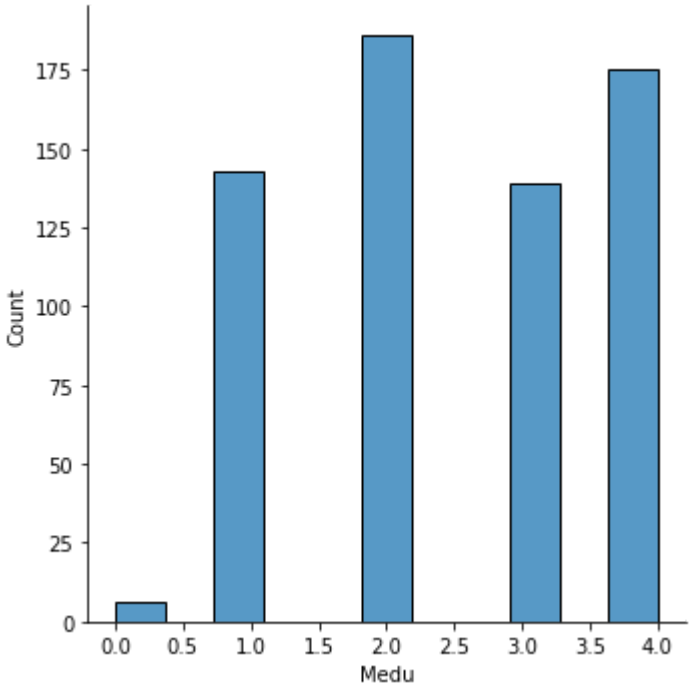


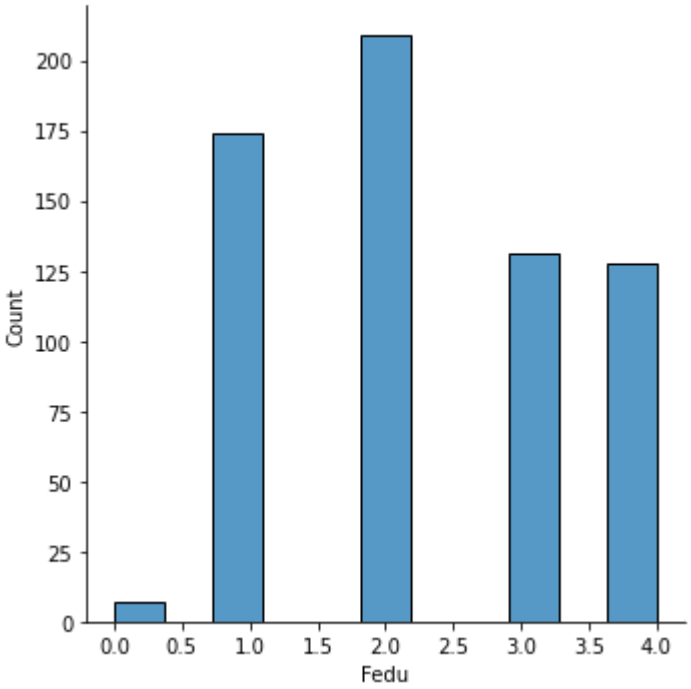
In [94]:

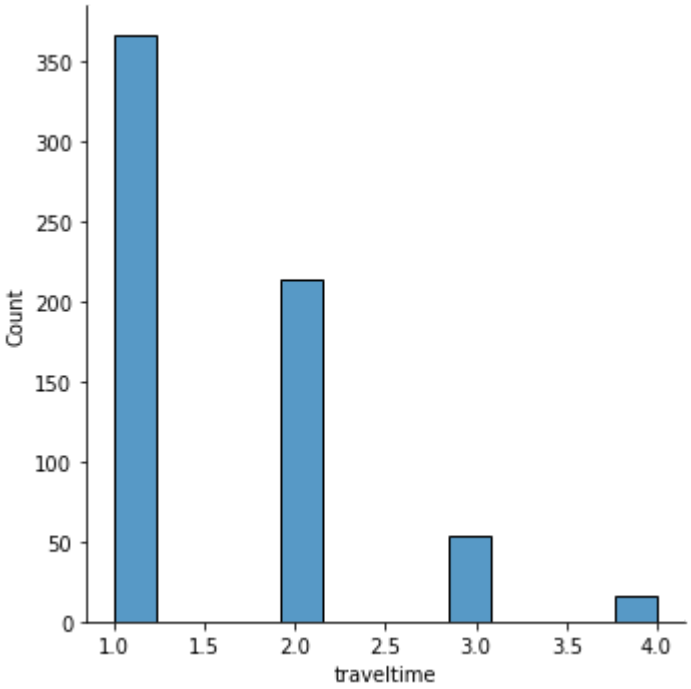
```
warnings.filterwarnings('ignore')
features_=data.columns.values[:]
fig=plt.figure(figsize=(20,10))
for columns, feature in enumerate(features_):
    sns.displot(data[feature])
plt.show()
```

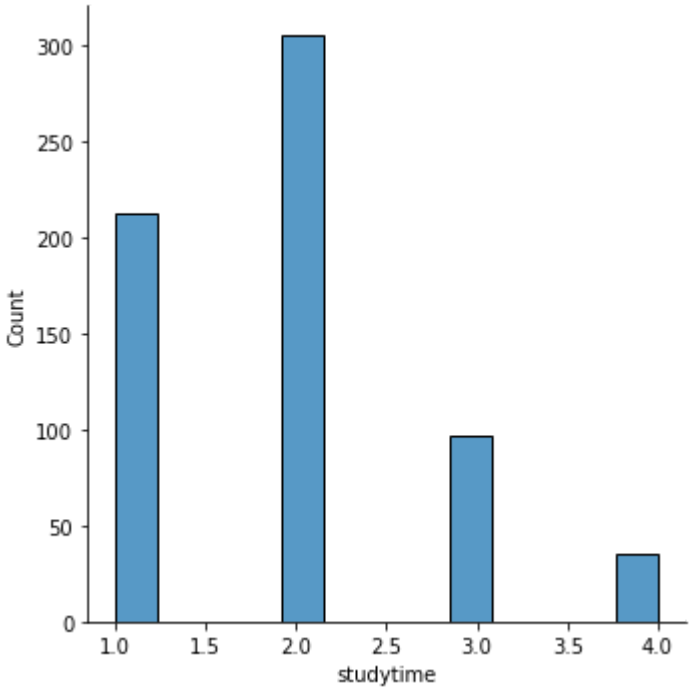
```
<Figure size 1440x720 with 0 Axes>
```

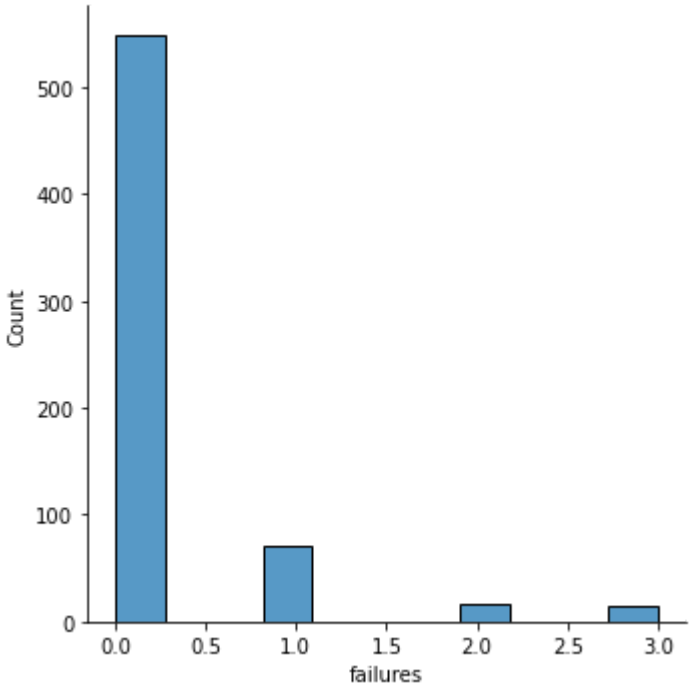

Count
175
150
125
100
75
50
25
0
15
16
17
18
19
20
21
22
age

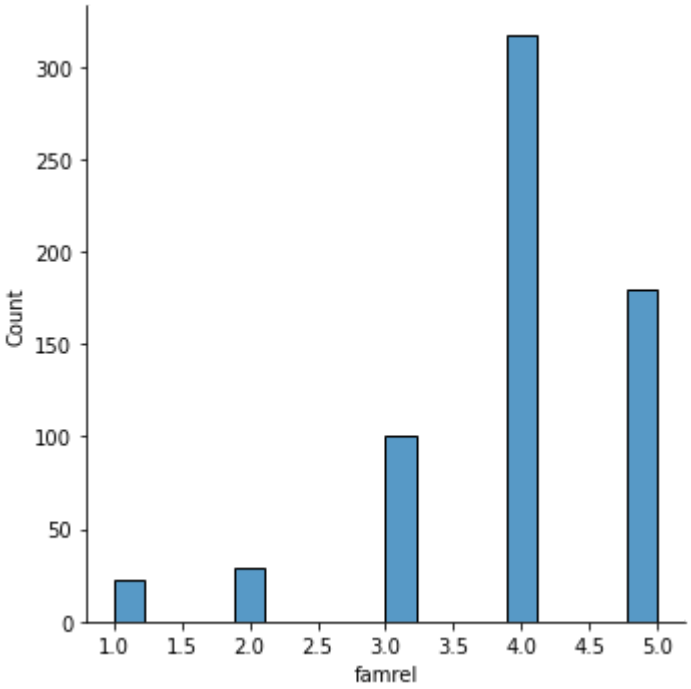

Count
175
150
125
100
75
50
25
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Medu

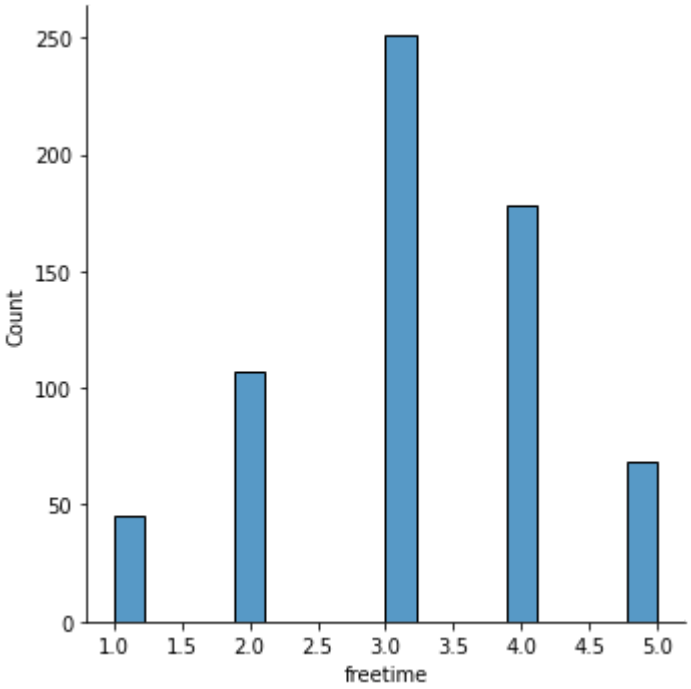

Count
200
175
150
125
100
75
50
25
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Fedu

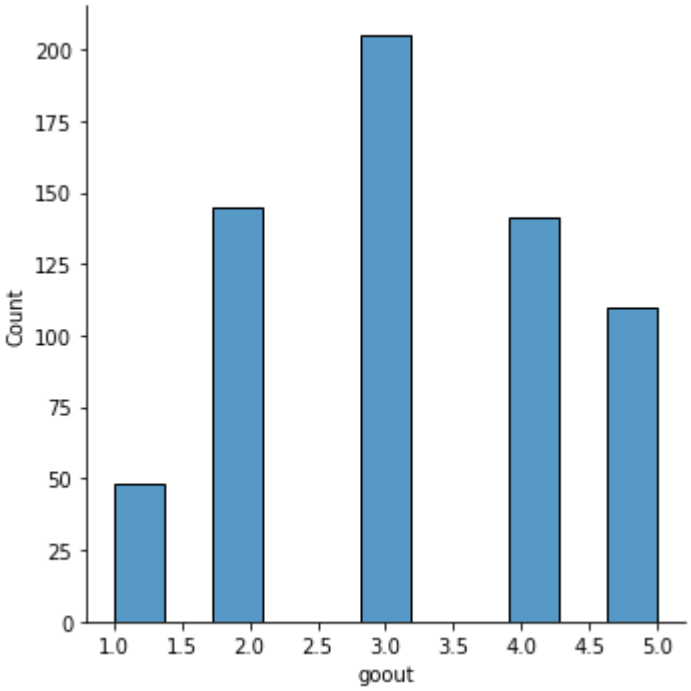

Count
350
300
250
200
150
100
50
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
traveltime

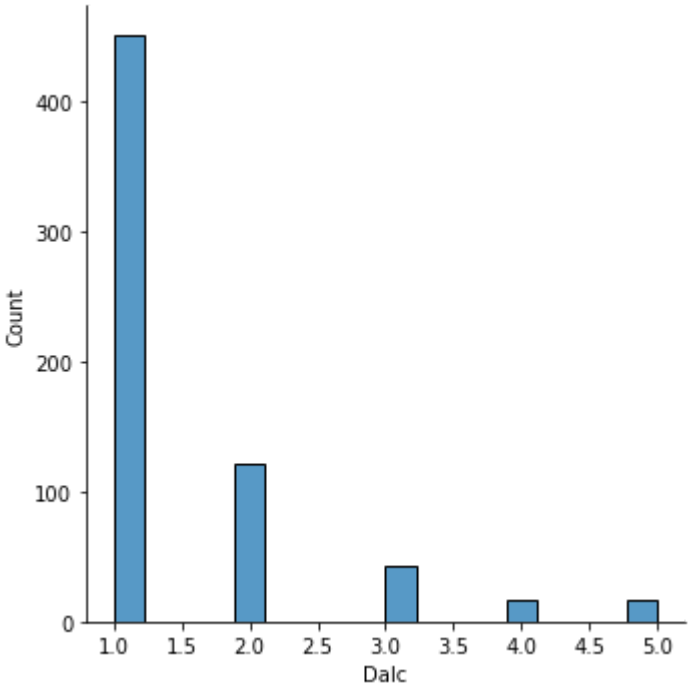

300
250
200
150
100
50
0
Count
1.0
1.5
2.0
2.5
3.0
3.5
4.0
studytime

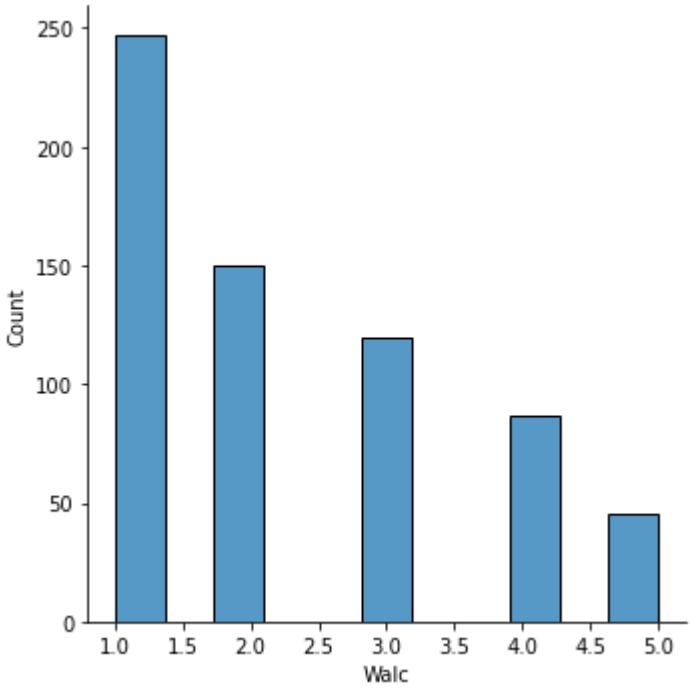

500
400
300
200
100
0
Count
0.0
0.5
1.0
1.5
2.0
2.5
3.0
failures

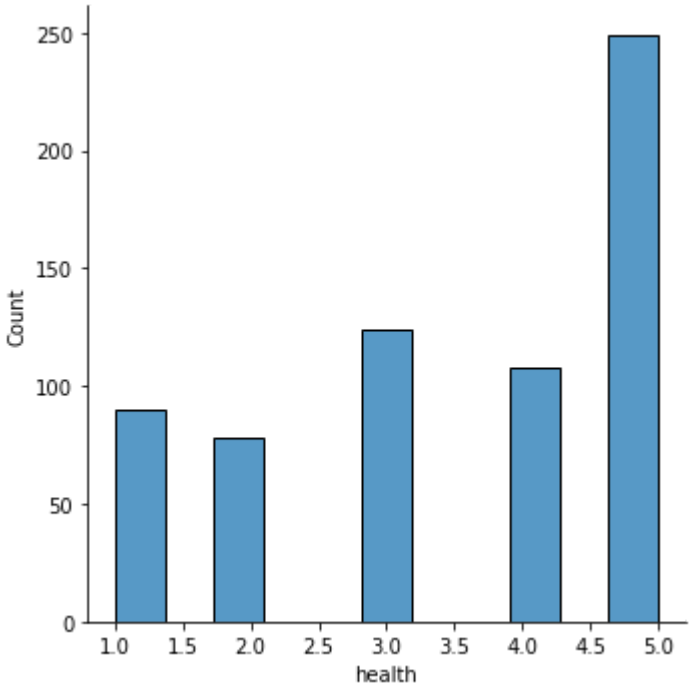

Count
300
250
200
150
100
50
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
famrel

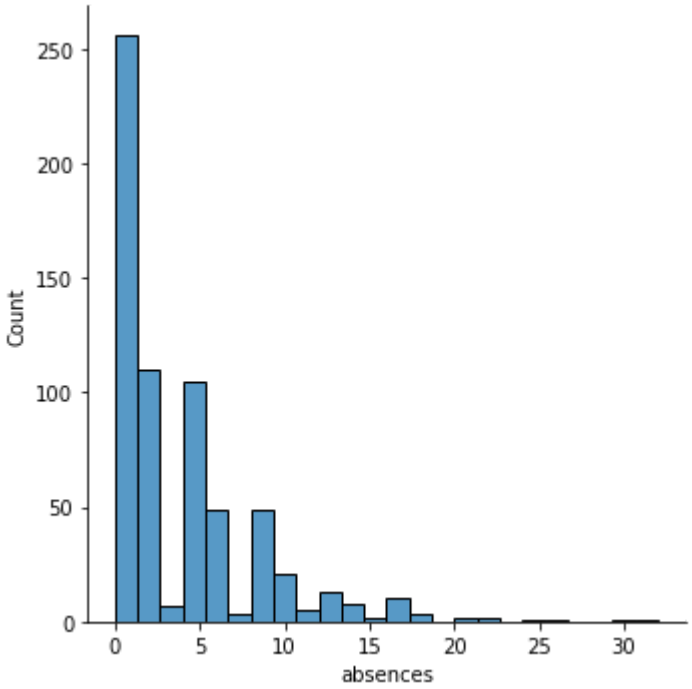

Count
250
200
150
100
50
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
freetime

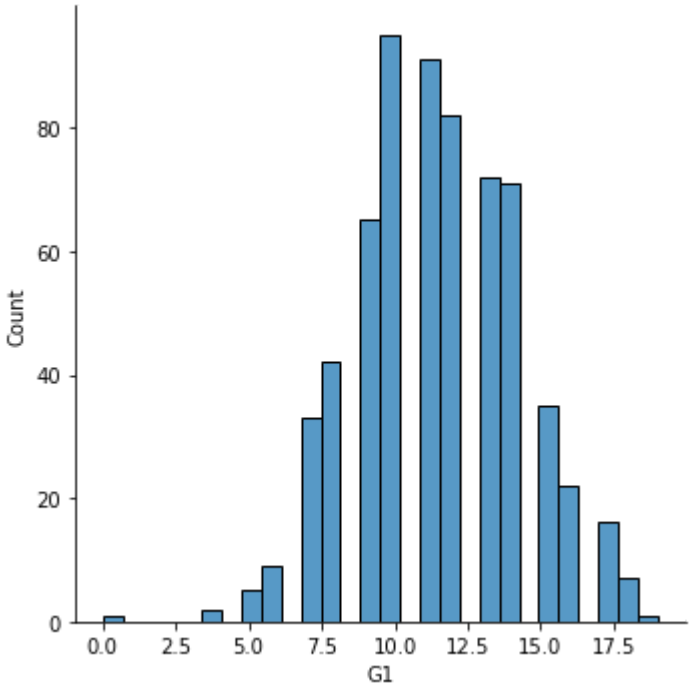

Count
goout
0
25
50
75
100
125
150
175
200
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

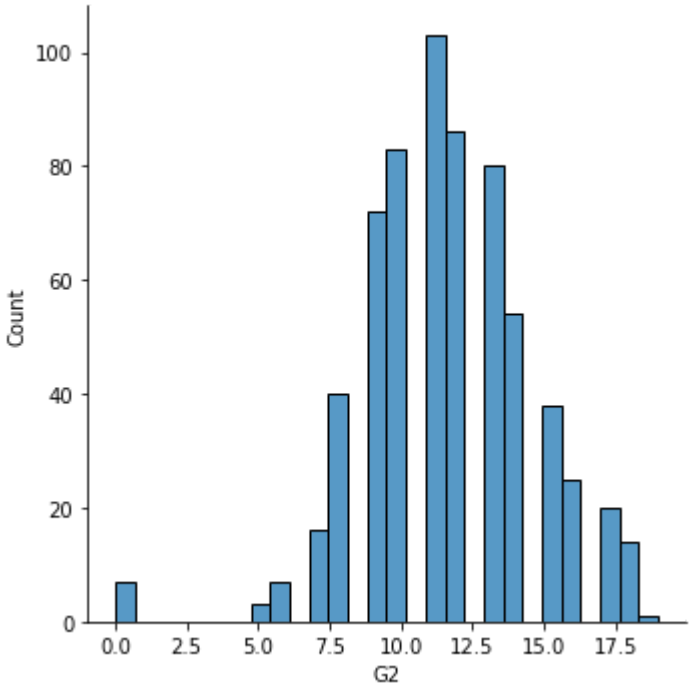

Count
Dalc
0
100
200
300
400
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

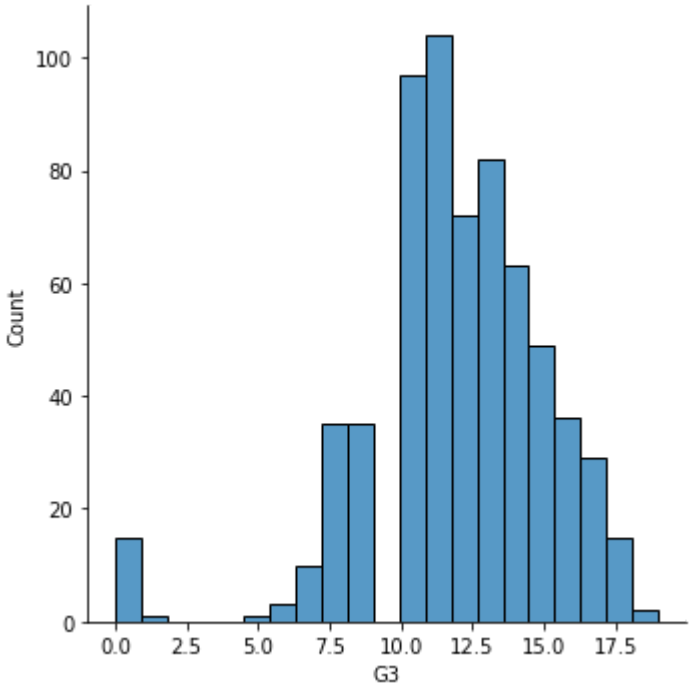

250
200
150
100
50
0
Count
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Walc


250
200
150
100
50
0
Count
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
health


250
200
150
100
50
0
Count
0
5
10
15
20
25
30
absences


80
60
40
20
0
Count
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
G1

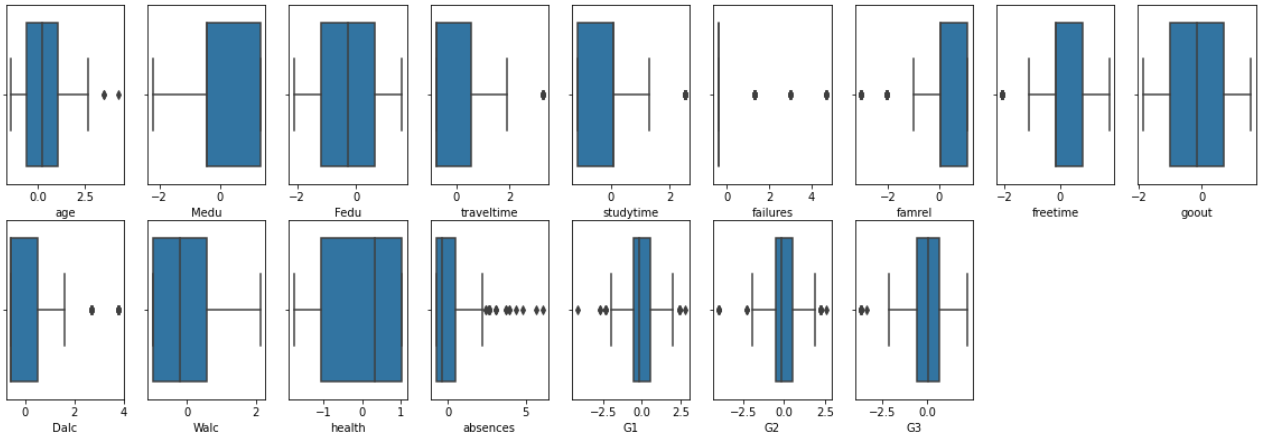




Scanning For Outliers

In [95]:
```
import warnings
```

In [97]:
```
warnings.filterwarnings('ignore')
features_=copy_data.columns.values[:]
fig=plt.figure(figsize=(20,10))
for columns, feature in enumerate(features_):
    fig.add_subplot(3,9,columns+1)
    sns.boxplot(copy_data[feature],data=copy_data)
plt.show()
```



In [99]:
```
def detect(data):
    outliers=[]
    threshhold=3
    mean=np.mean(data)
    std=np.std(data)
    for i in data:
        zscore=(i-mean)/std
        if(np.abs(zscore)>3):
            outliers.append(i)
    return outliers
```

In [100…
```
absences=detect(copy_data['absences'])
print(absences)
```

```
[4.3863974667881624, 3.9551010565244673, 6.111583107842941, 5.6802866975792465, 3.739452
85139262, 3.092508235997078, 4.817693877051857, 3.9551010565244673, 3.092508235997078,
3.092508235997078, 3.73945285139262]
```

In [101…
```
def replace(data):
    outliers=[]
    threshhold=3
    mean=np.mean(data)
    std=np.std(data)
    for i in data:
        zscore=(i-mean)/std
        if(zscore>3):
            data.replace(i,3,inplace=True)
            outliers.append(i)
        if(zscore<-3):
            data.replace(i,-3,inplace=True)
            outliers.append(i)
    return outliers
```

In [102…
```
replace(copy_data['absences'])
```

Out[102…
```
[4.3863974667881624,
 3.9551010565244673,
 6.111583107842941,
 5.6802866975792465,
 3.73945285139262,
```

```
3.092508235997078,
4.817693877051857]
```

In [103…

```
sns.boxplot(copy_data['absences'])
```

Out[103…

```
<AxesSubplot:xlabel='absences'>
```