name = Divyanshu singh

id = 56

batch = DS2306

.

medical_cost_insurance

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.read_csv('medical_cost_insurance.csv')
df.head()
```

Out[2]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [5]:
```
df['sex'].nunique()
```

Out[5]:
```
2
```

In [6]:
```
df['smoker'].nunique()
```

Out[6]:
```
2
```

In [7]:
```
df['region'].nunique()
```

Out[7]:
```
4
```

In [8]:
```
df['sex'].value_counts()
```

Out[8]:
```
male      676
female    662
Name: sex, dtype: int64
```

In [9]:
```
df['sex']= pd.factorize(df.sex)[0]
df['sex'].value_counts()
```

Out[9]:
```
1    676
0    662
Name: sex, dtype: int64
```

In [10]:
```
df['smoker'].value_counts()
```

Out[10]:
```
no     1064
yes     274
Name: smoker, dtype: int64
```

In [11]:
```
df['smoker']= pd.factorize(df.smoker)[0]
df['smoker'].value_counts()
```

Out[11]:
```
1    1064
0     274
Name: smoker, dtype: int64
```

In [12]:
```
df['region'].value_counts()
```

Out[12]:
```
southeast    364
southwest    325
northwest    325
northeast    324
Name: region, dtype: int64
```

In [13]:
```
df['region'] = pd.factorize(df.region)[0]
df['region'].value_counts()
```

Out[13]:
```
1    364
0    325
2    325
3    324
Name: region, dtype: int64
```

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   int64  
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   int64  
 5   region    1338 non-null   int64  
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(5)
memory usage: 73.3 KB
```

In [15]:
```
df.describe()
```

Out[15]:

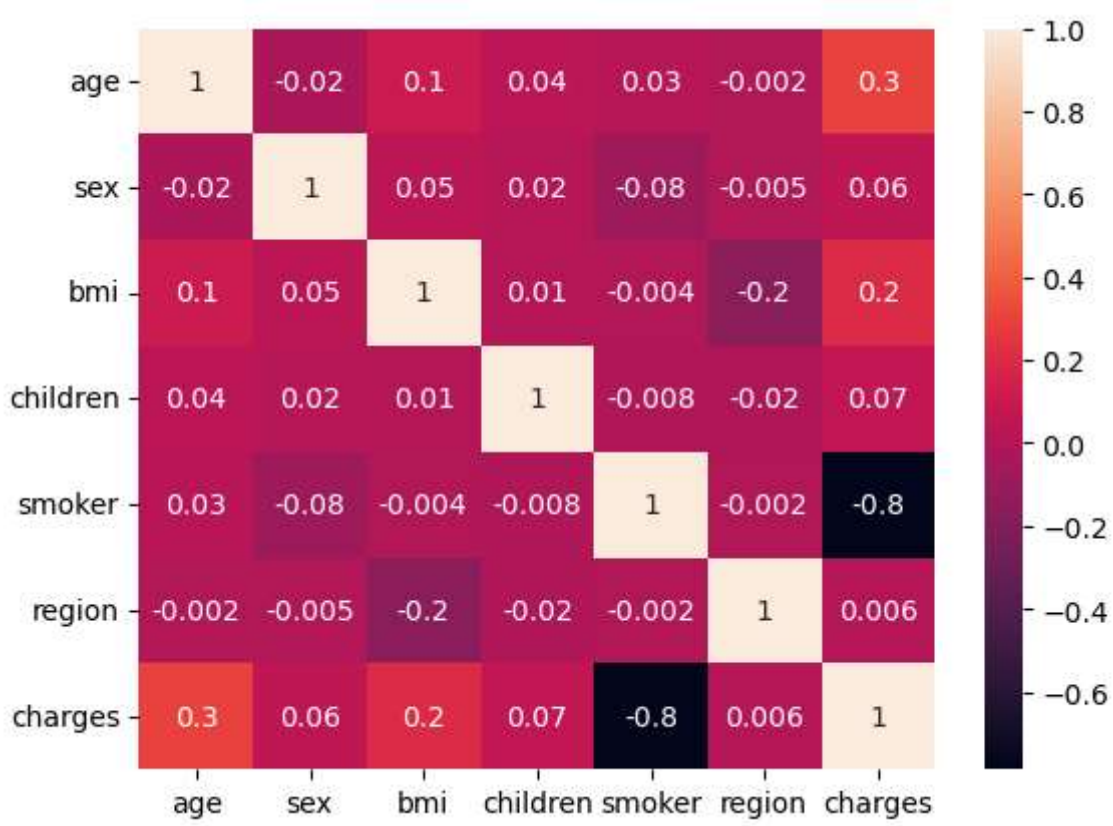
	age	sex	bmi	children	smoker	region	charges
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	0.505232	30.663397	1.094918	0.795217	1.484305	13270.422265
std	14.049960	0.500160	6.098187	1.205493	0.403694	1.104885	12110.011237
min	18.000000	0.000000	15.960000	0.000000	0.000000	0.000000	1121.873900
25%	27.000000	0.000000	26.296250	0.000000	1.000000	1.000000	4740.287150
50%	39.000000	1.000000	30.400000	1.000000	1.000000	1.000000	9382.033000
75%	51.000000	1.000000	34.693750	2.000000	1.000000	2.000000	16639.912515
max	64.000000	1.000000	53.130000	5.000000	1.000000	3.000000	63770.428010

.

In [16]:
```
import seaborn as sns
```

In [17]:
```
correlation = df.corr()
sns.heatmap(correlation, fmt='.1g', linecolor = 'red', annot=True)
```

Out[17]:
```
<Axes: >
```



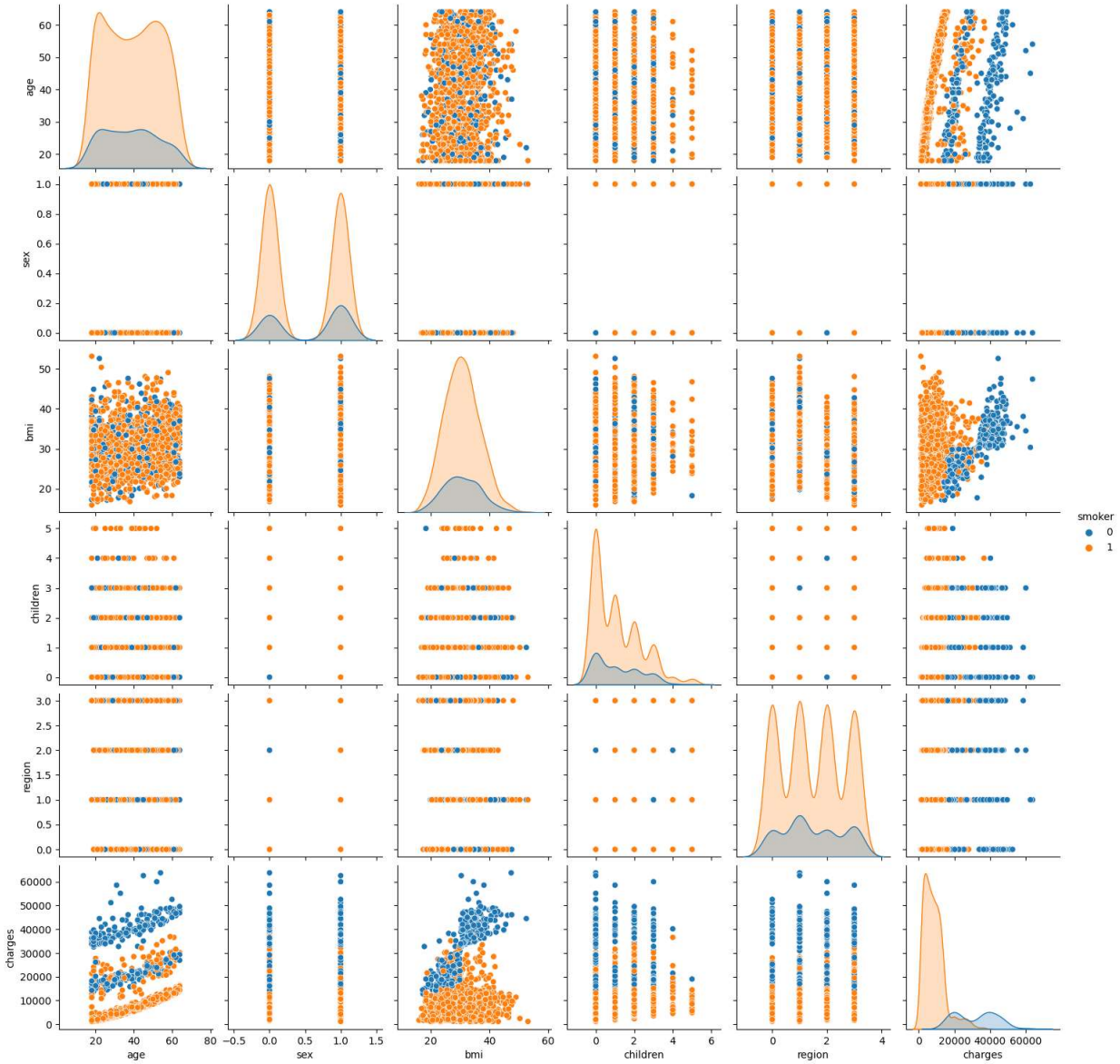
from the above chart we conclude that - smoker and charges are highly negatively correlated . age and bmi are positively corelated to charges.

.

In [18]:
```
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
```

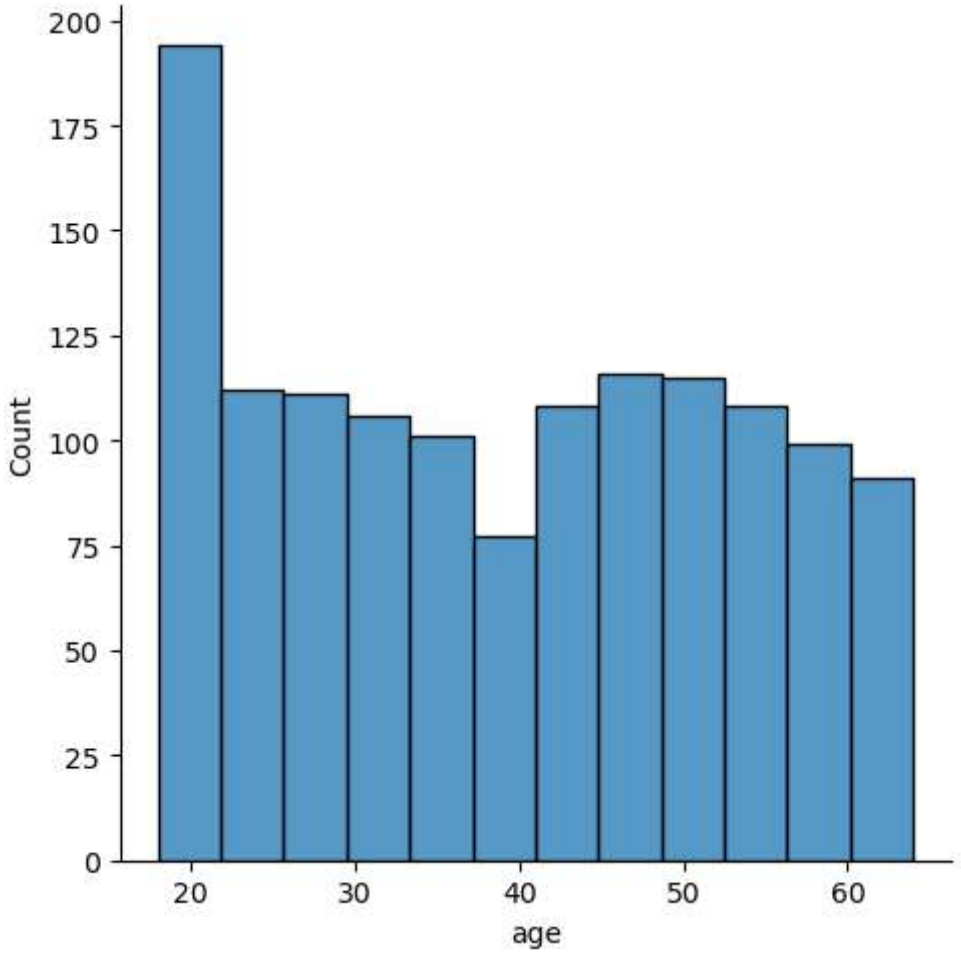
In [19]:
```
sns.pairplot(df,hue='smoker')
```

Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x1c1b710fd90>
```



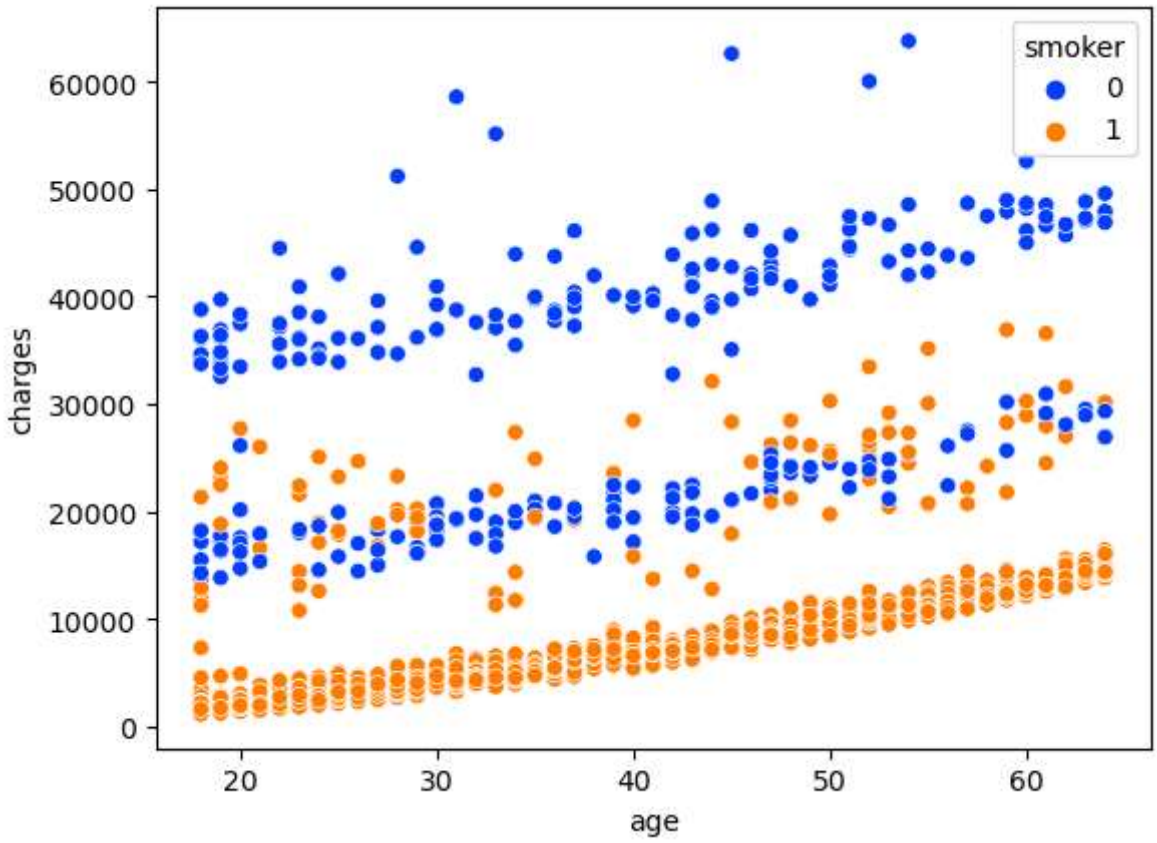
In [20]:
```
sns.displot(data=df, x='age')
```

Out[20]:
```
<seaborn.axisgrid.FacetGrid at 0x1c1bb146910>
```



In [21]:
```
sns.scatterplot(x='age',y='charges',data=df , hue='smoker' , palette='bright')
```

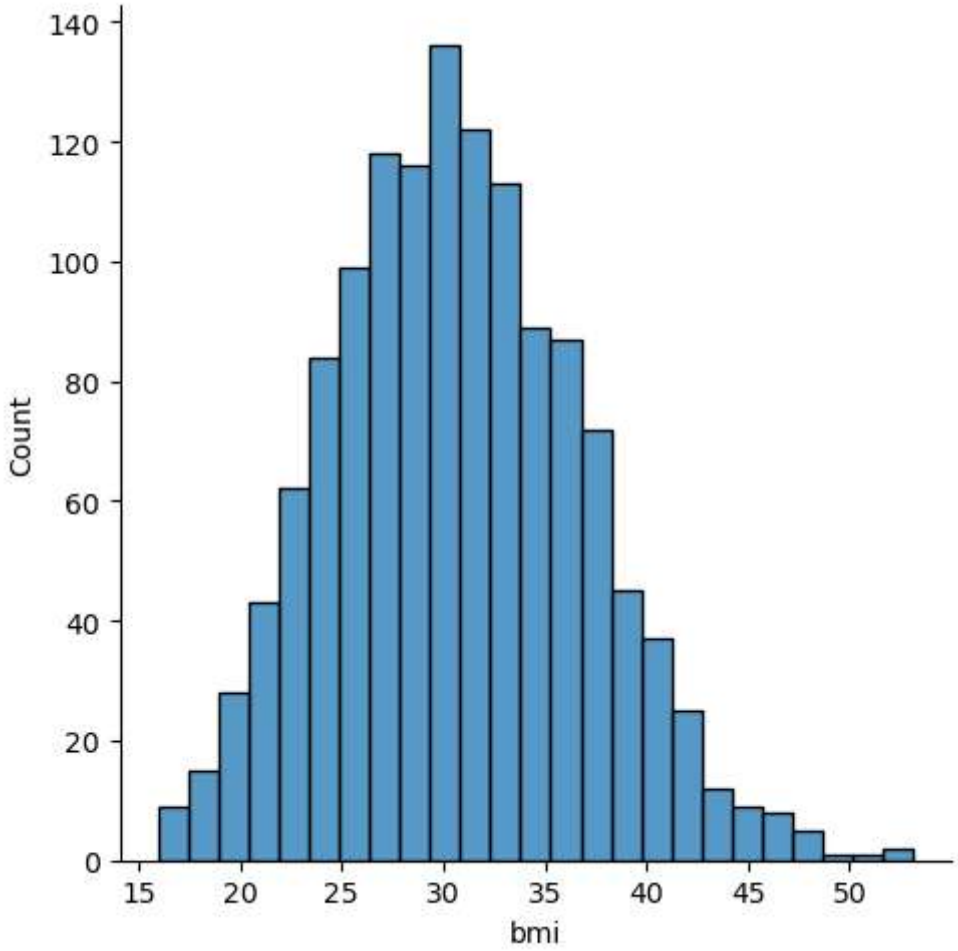
Out[21]:
```
<Axes: xlabel='age', ylabel='charges'>
```



.

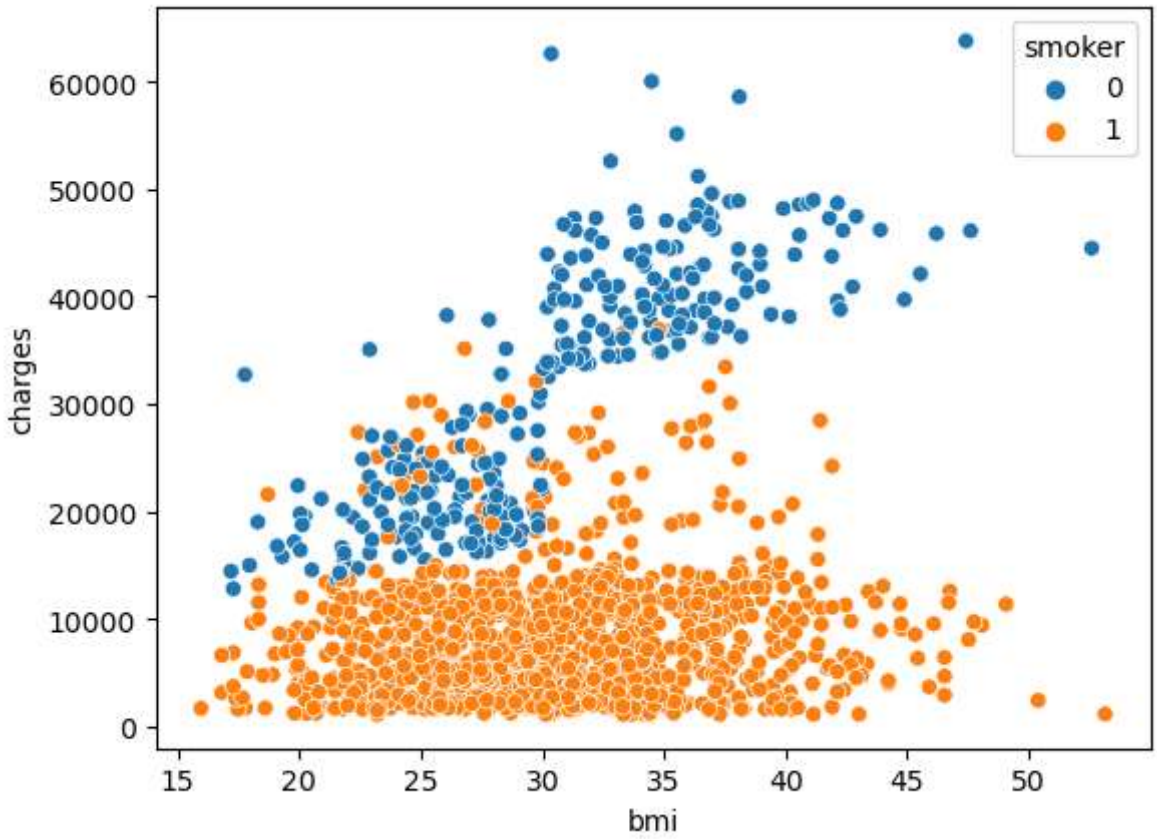
In [22]:
```
sns.displot(data=df, x='bmi')
```

Out[22]:
```
<seaborn.axisgrid.FacetGrid at 0x1c1b8920460>
```



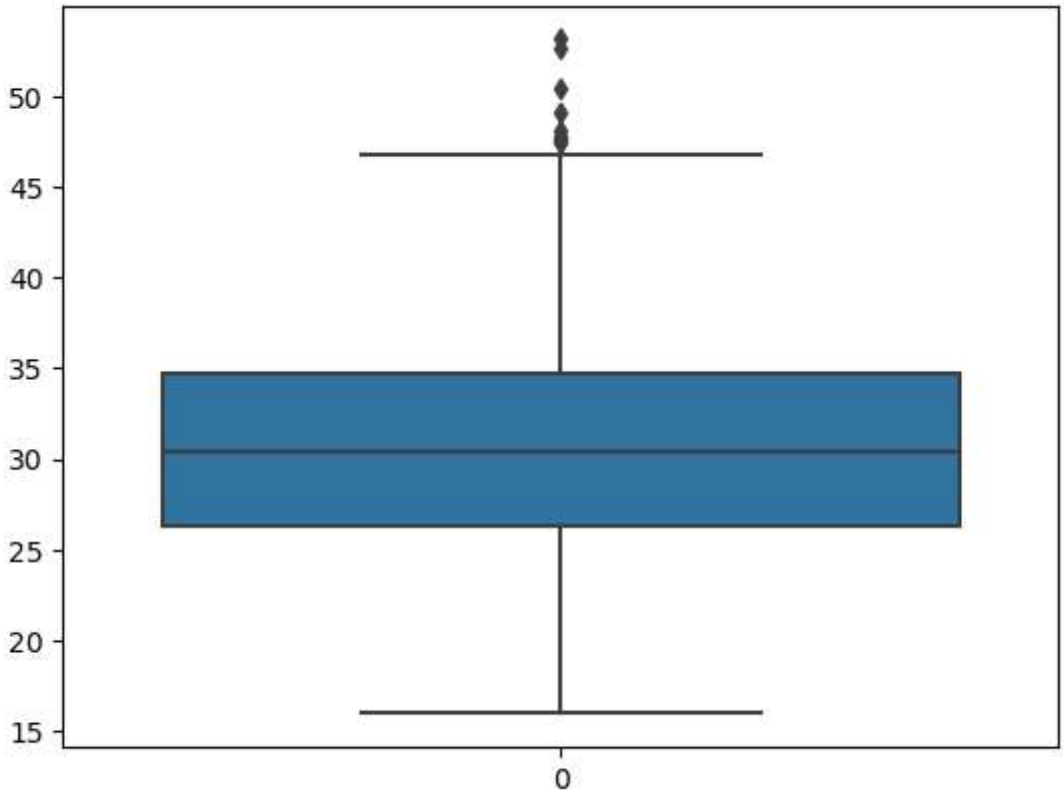
In [23]:
```
sns.scatterplot(x='bmi',y='charges',data=df , hue='smoker')
```

Out[23]:
```
<Axes: xlabel='bmi', ylabel='charges'>
```



In [24]:
```
sns.boxplot(df['bmi'])
```

Out[24]:
```
<Axes: >
```



there are outliers present in bmi it was shown in describe function and now its confirm through boxplot

.

iqr method

In [25]:
```
q1 = df['bmi'].quantile(.25)
q3 = df['bmi'].quantile(.75)

iqr = q3-q1
low = q1-1.5*iqr
up = q3+1.5*iqr

print(low)
print(up)
```

```
13.7
47.290000000000006
```

In [26]:
```
new_df = df.loc[(df['bmi']<up) & (df['bmi']>low)]
print(len(df))
print(len(new_df))
```

```
1338
1329
```

.

In [27]:
```
df.skew()
```

Out[27]:
```
age         0.055673
sex        -0.020951
bmi         0.284047
children    0.938380
smoker     -1.464766
region      0.038101
charges     1.515880
dtype: float64
```

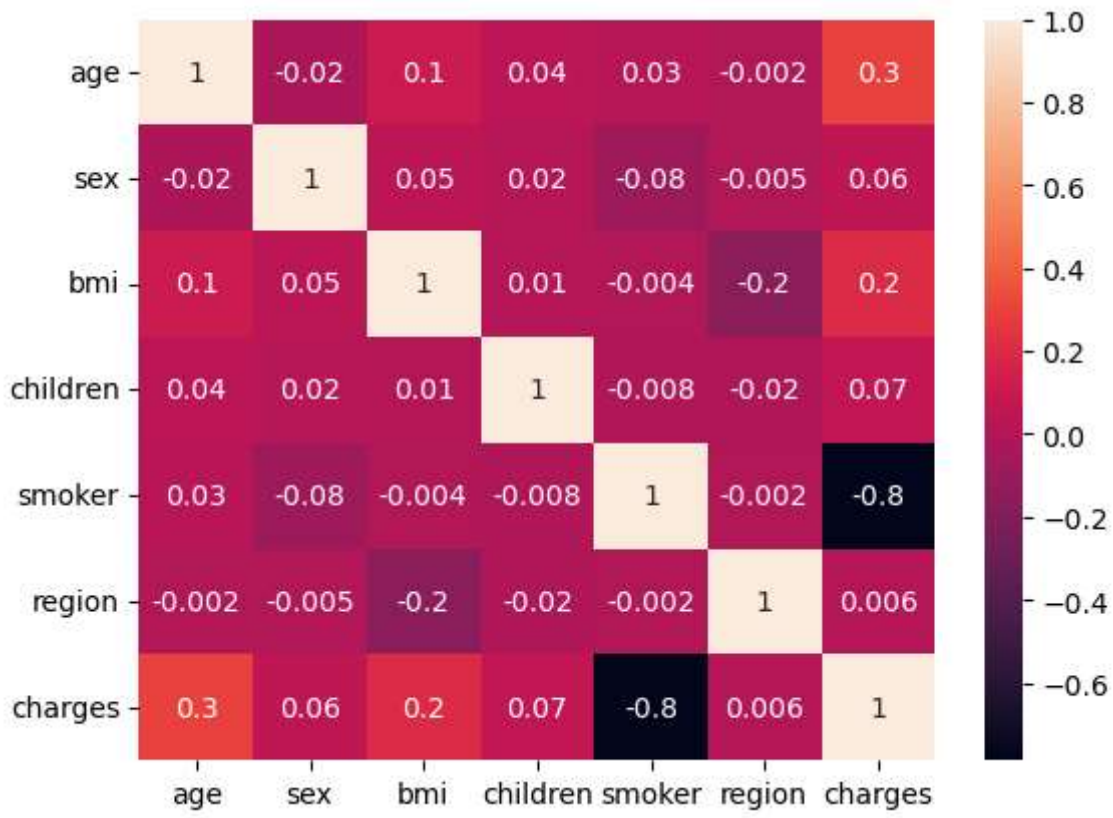
In [28]:
```
new_df.skew()
```

Out[28]:
```
age         0.058413
sex        -0.016573
bmi         0.157180
children    0.936628
smoker     -1.471424
region      0.031651
charges     1.500577
dtype: float64
```

.

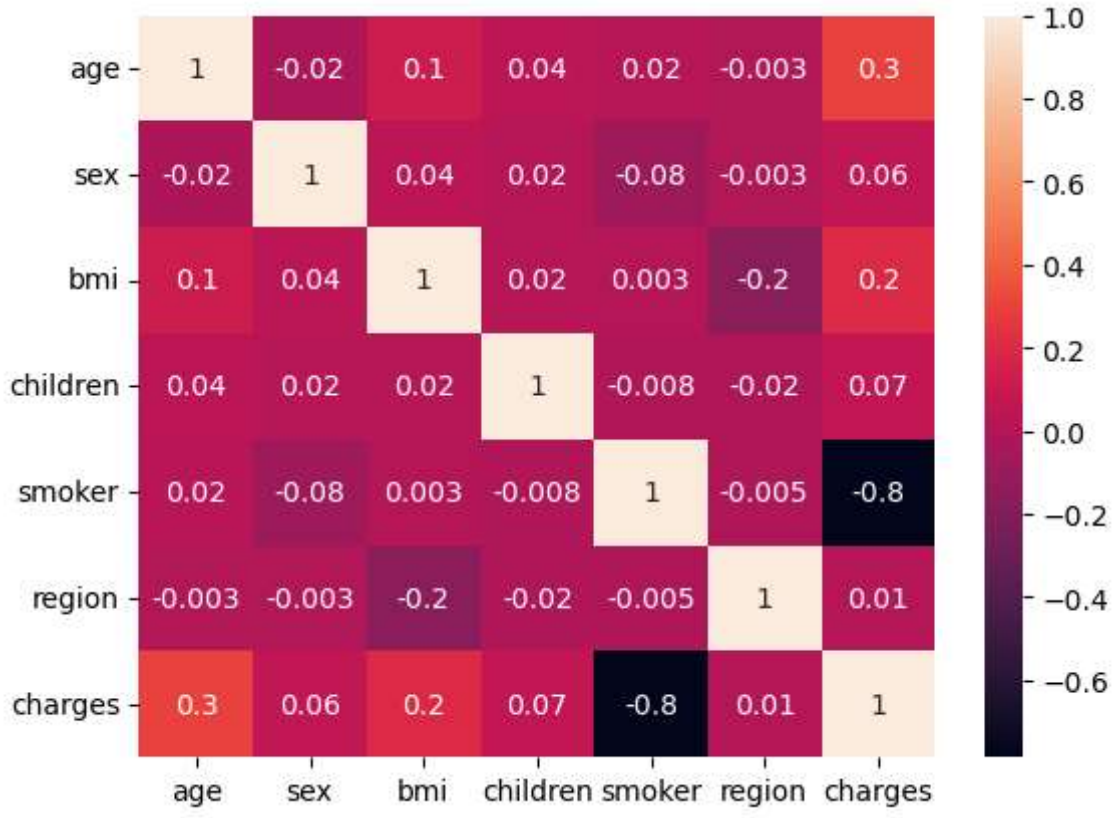
In [29]:
```
correlation = df.corr()
sns.heatmap(correlation, fmt='.1g', linecolor = 'red', annot=True)
```

Out[29]: <Axes: >



In [30]:
```
correlation = new_df.corr()
sns.heatmap(correlation, fmt='.1g', linecolor = 'red', annot=True)
```

Out[30]: <Axes: >



In [31]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

df

In [32]:
```
x= df.drop('charges',axis=1)
y= df['charges']
```

In [33]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.3, random_state=42)
```

In [34]:
```
x_train.shape
```

Out[34]:
```
(936, 6)
```

In [35]:
```
x_test.shape
```

Out[35]:
```
(402, 6)
```

In [36]:
```
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

In [37]:
```
svm = SVR()
svm.fit(x_train,y_train)
```

Out[37]:
```
SVR()
```

In [38]:
```
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[38]:
```
LinearRegression()
```

In [39]:
```
rfr= RandomForestRegressor()
rfr.fit(x_train,y_train)
```

Out[39]:
```
RandomForestRegressor()
```

In [40]:
```
gbr= GradientBoostingRegressor()
gbr.fit(x_train,y_train)
```

Out[40]:
```
GradientBoostingRegressor()
```

In [41]:
```
y_pred1 = svm.predict(x_test)
y_pred2 = lr.predict(x_test)
y_pred3 = rfr.predict(x_test)
y_pred4 = gbr.predict(x_test)
```

In [42]:
```
from sklearn import metrics
from sklearn.metrics import accuracy_score , confusion_matrix
```

In [43]:
```
svm1 = metrics.r2_score(y_test,y_pred1)
svm1
```

Out[43]:
```
-0.08170794773255219
```

In [44]:
```
lr2 = metrics.r2_score(y_test,y_pred2)
lr2
```

Out[44]:
```
0.7694415927057693
```

In [45]:
```
rfr3 = metrics.r2_score(y_test,y_pred3)
rfr3
```

Out[45]:
```
0.8529529920654574
```

In [46]:
```
gbr4 = metrics.r2_score(y_test,y_pred4)
gbr4
```

Out[46]:
```
0.8691639800395429
```

GradientBoostingRegressor gives the best performance

new_df

In [47]:
```
x2= new_df.drop('charges',axis=1)
y2= new_df['charges']

x_train,x_test,y_train,y_test = train_test_split(x2,y2, test_size=0.3, random_state=42
print(x_train.shape)
print(x_test.shape)
```

```
(930, 6)
(399, 6)
```

In [48]:
```
svm = SVR()
svm.fit(x_train,y_train)

lr= LinearRegression()
lr.fit(x_train,y_train)

rfr= RandomForestRegressor()
rfr.fit(x_train,y_train)
```

```
gbr= GradientBoostingRegressor()
gbr.fit(x_train,y_train)
```

Out[48]:
```
▾ GradientBoostingRegressor
GradientBoostingRegressor()
```

In [49]:
```
y_preda = svm.predict(x_test)
y_predb = lr.predict(x_test)
y_predc = rfr.predict(x_test)
y_predd = gbr.predict(x_test)
```

In [50]:
```
svma = metrics.r2_score(y_test,y_preda)
svma
```

Out[50]:
```
-0.07018426873340622
```

In [51]:
```
lrb = metrics.r2_score(y_test,y_predb)
lrb
```

Out[51]:
```
0.7716630723497743
```

In [52]:
```
rfrc = metrics.r2_score(y_test,y_predc)
rfrc
```

Out[52]:
```
0.8362618231460421
```

In [53]:
```
gbrd = metrics.r2_score(y_test,y_predd)
gbrd
```

Out[53]:
```
0.8600283315761992
```

In [54]:
```
from sklearn.model_selection import cross_val_score
```

In [55]:
```
recheck = cross_val_score(svm,x,y)
print(recheck)
print(recheck.mean())
```

```
[-0.11535857 -0.11037308 -0.07992918 -0.10416852 -0.10829545]
-0.10362495985825126
```

In [56]:
```
recheck = cross_val_score(lr,x,y)
print(recheck)
print(recheck.mean())
```

```
[0.76123983 0.70838663 0.77786772 0.73365206 0.7551335 ]
0.7472559473580954
```

In [57]:
```
recheck = cross_val_score(rfr,x,y)
print(recheck)
print(recheck.mean())
```

```
[0.84926824 0.77486337 0.86457641 0.82606821 0.84869856]
0.8326949597644859
```

In [58]:
```
recheck = cross_val_score(gbr,x,y)
print(recheck)
```

```
print(recheck.mean())
```

```
[0.87519987 0.79177595 0.89421047 0.84919251 0.85971592]
0.8540189433121006
```

.

GradientBoostingRegressor is the best

.

.

winequality-red.csv

In [59]:
```
df = pd.read_csv('winequality-red.csv')
df
```

Out[59]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

1599 rows × 12 columns

In [60]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

In [61]:
```
df.isnull().sum()
```

Out[61]:
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

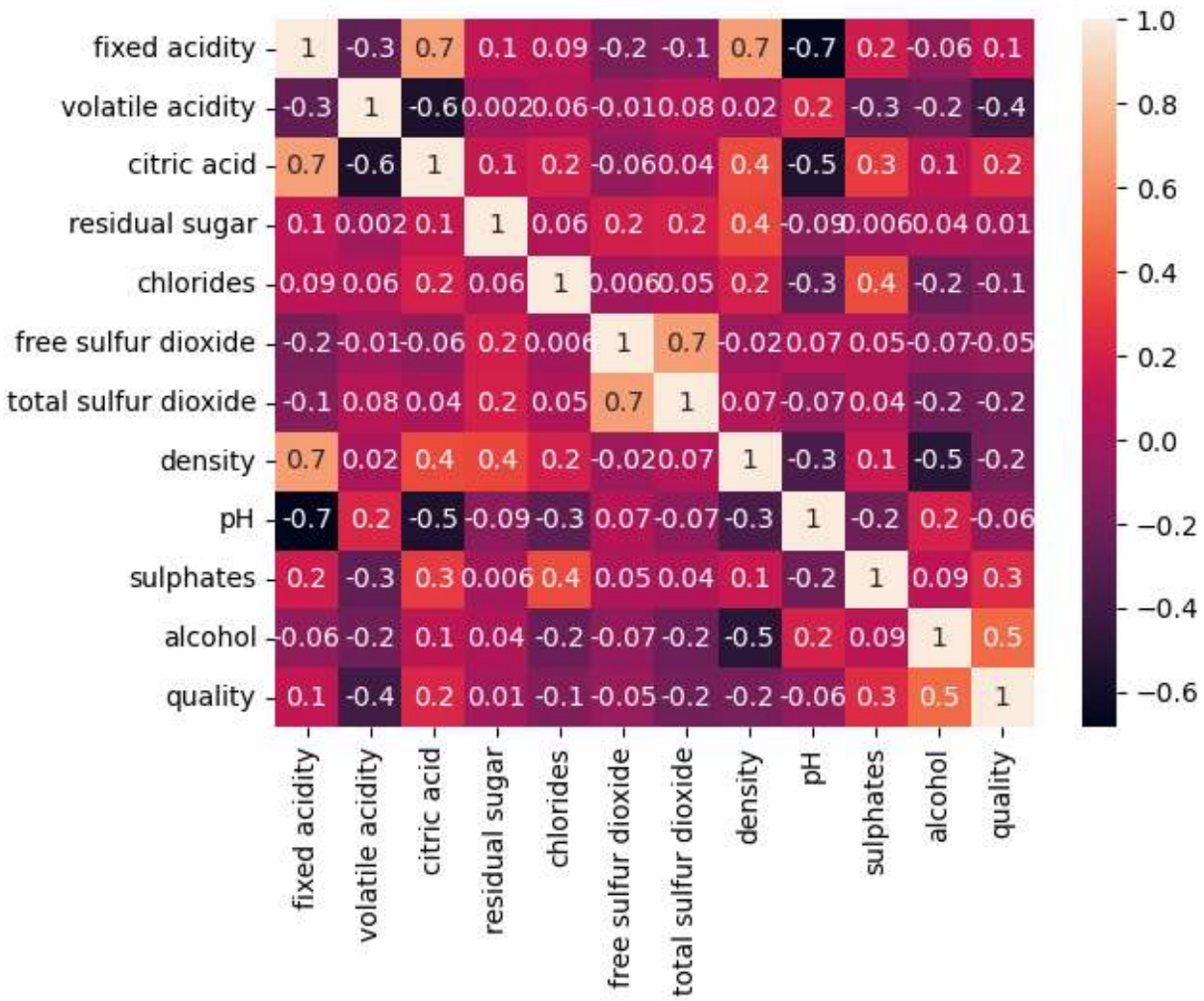
In [62]:
```
df.describe()
```

Out[62]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	15
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	

In [63]:
```
correlation = df.corr()
sns.heatmap(correlation, fmt='.1g', linecolor = 'red', annot=True)
```

Out[63]: <Axes: >



quality and alchohol,sulphates are positively related

quality and volatile acidity are negatively related

.

In [64]:
```
df['quality'].value_counts()
```

Out[64]:
```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

In [65]:
```
y3 = df['quality'].apply(lambda z: 1 if z>6 else 0)
y3
```

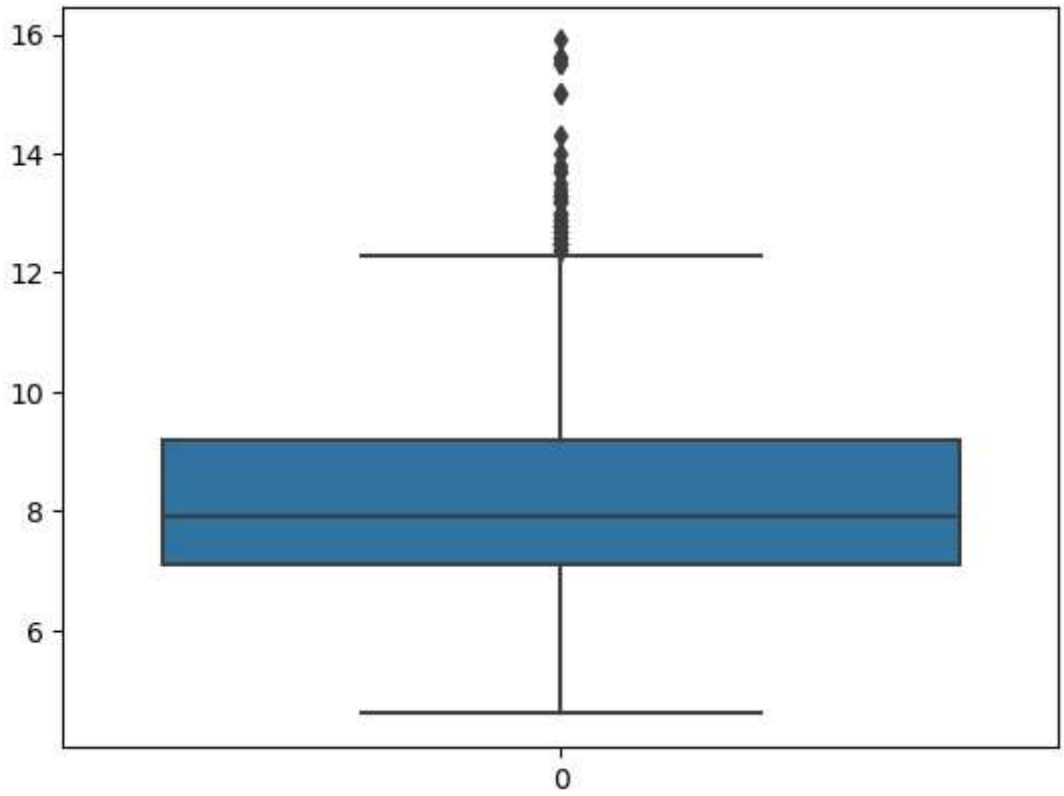
Out[65]:
```
0       0
1       0
2       0
3       0
4       0
       ..
1594    0
1595    0
1596    0
1597    0
1598    0
Name: quality, Length: 1599, dtype: int64
```

In [66]:
```
y3.value_counts()
```

Out[66]:
```
0    1382
1     217
Name: quality, dtype: int64
```

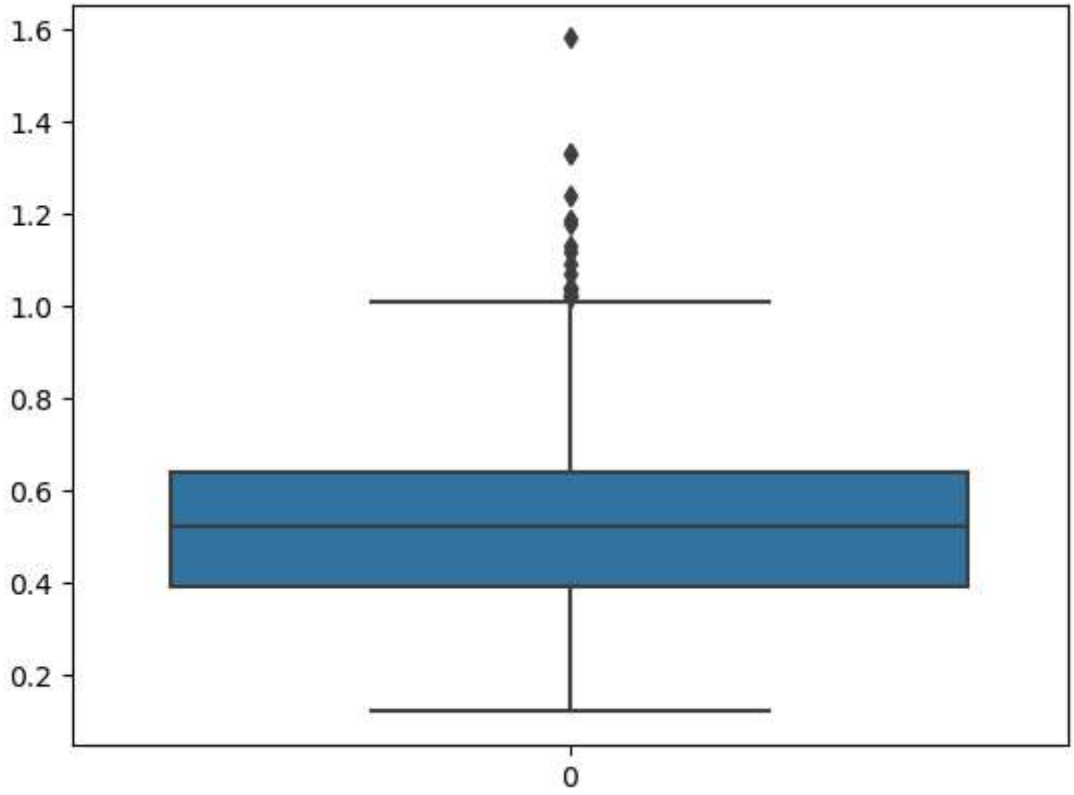
In [67]:
```
sns.boxplot(df['fixed acidity'])
```

Out[67]:
```
<Axes: >
```



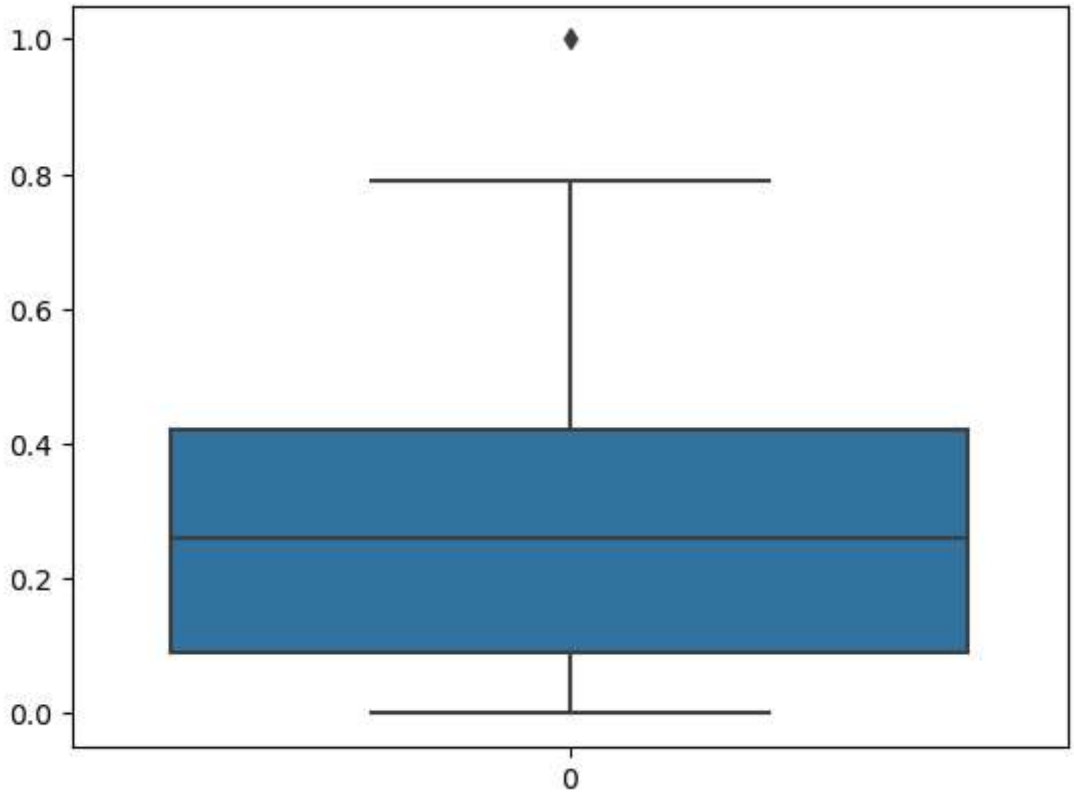
In [68]:
```
sns.boxplot(df['volatile acidity'])
```

Out[68]:
```
<Axes: >
```



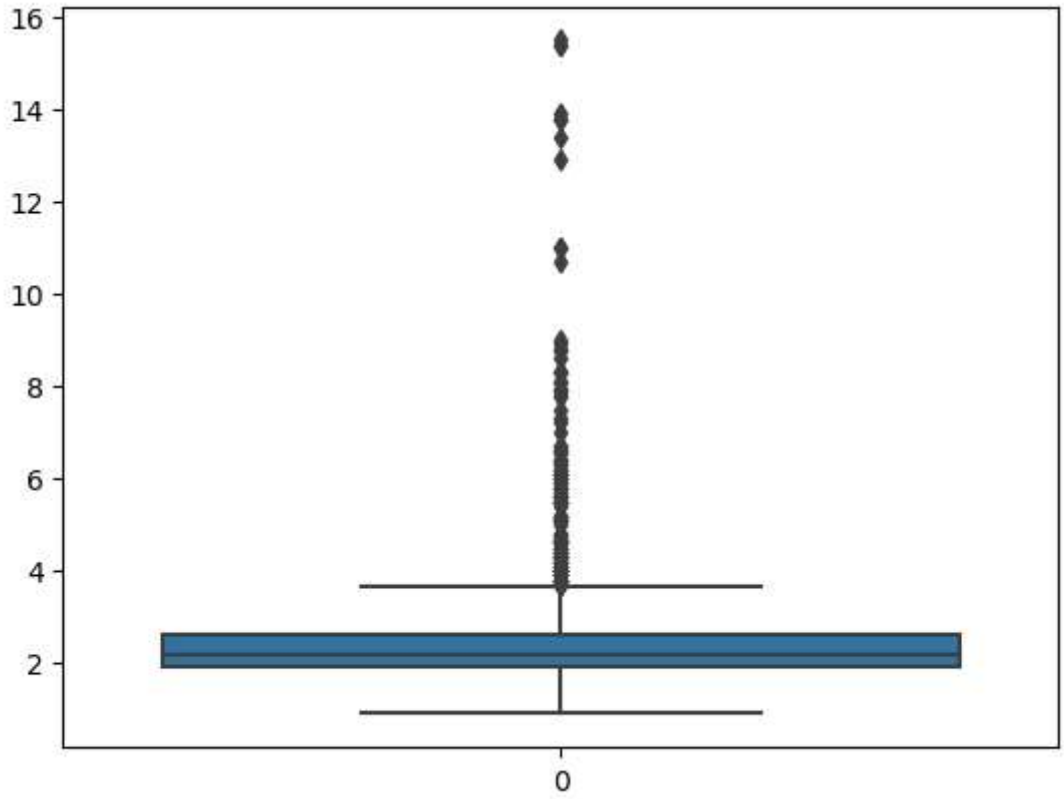
In [69]:
```
sns.boxplot(df['citric acid'])
```

Out[69]: <Axes: >



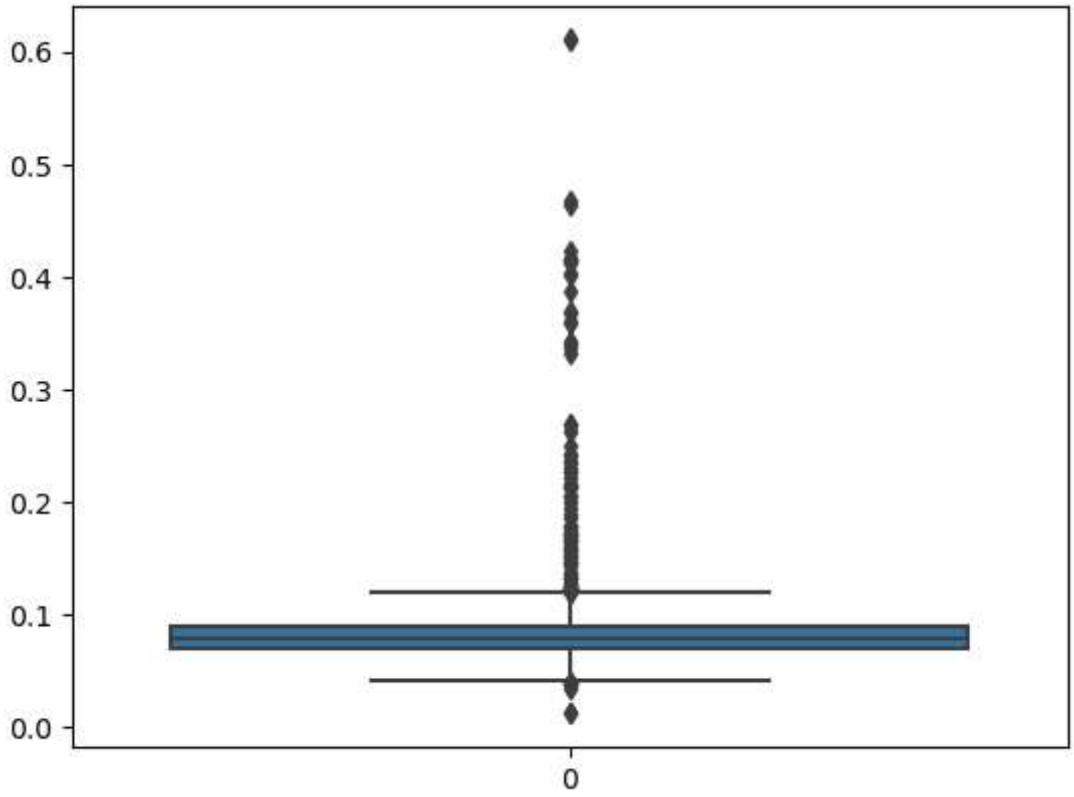
In [70]:
```
sns.boxplot(df['residual sugar'])
```

Out[70]: <Axes: >



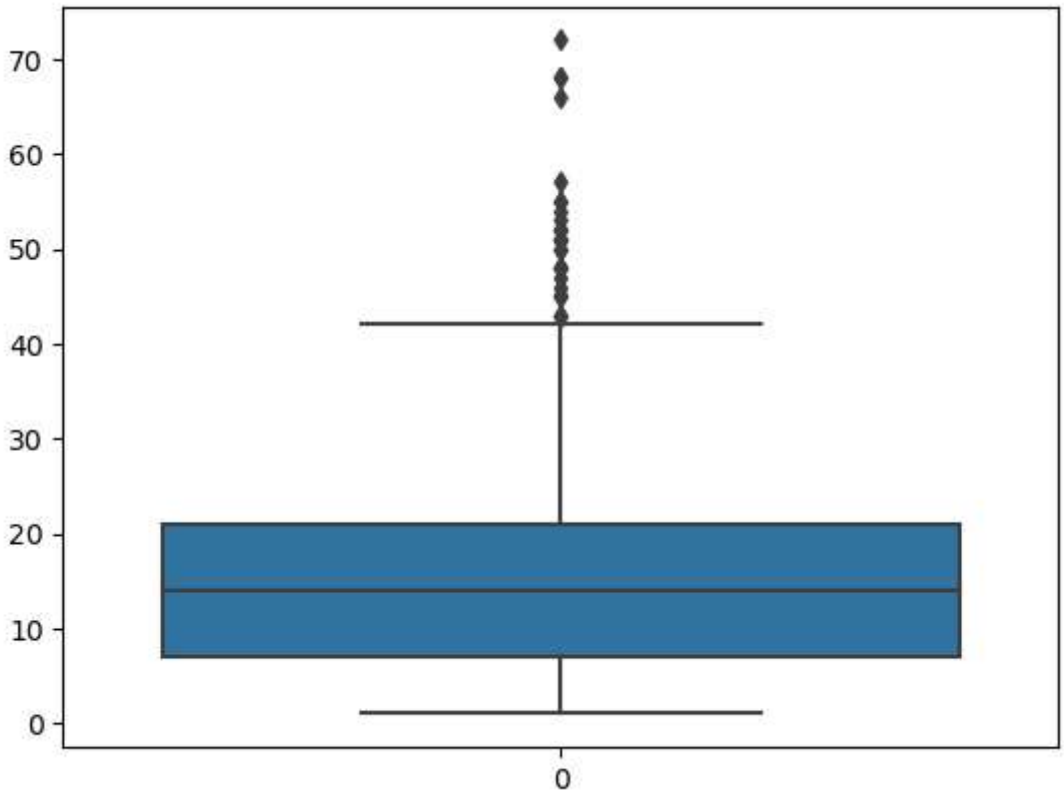
In [71]:
```
sns.boxplot(df['chlorides'])
```

Out[71]:
```
<Axes: >
```



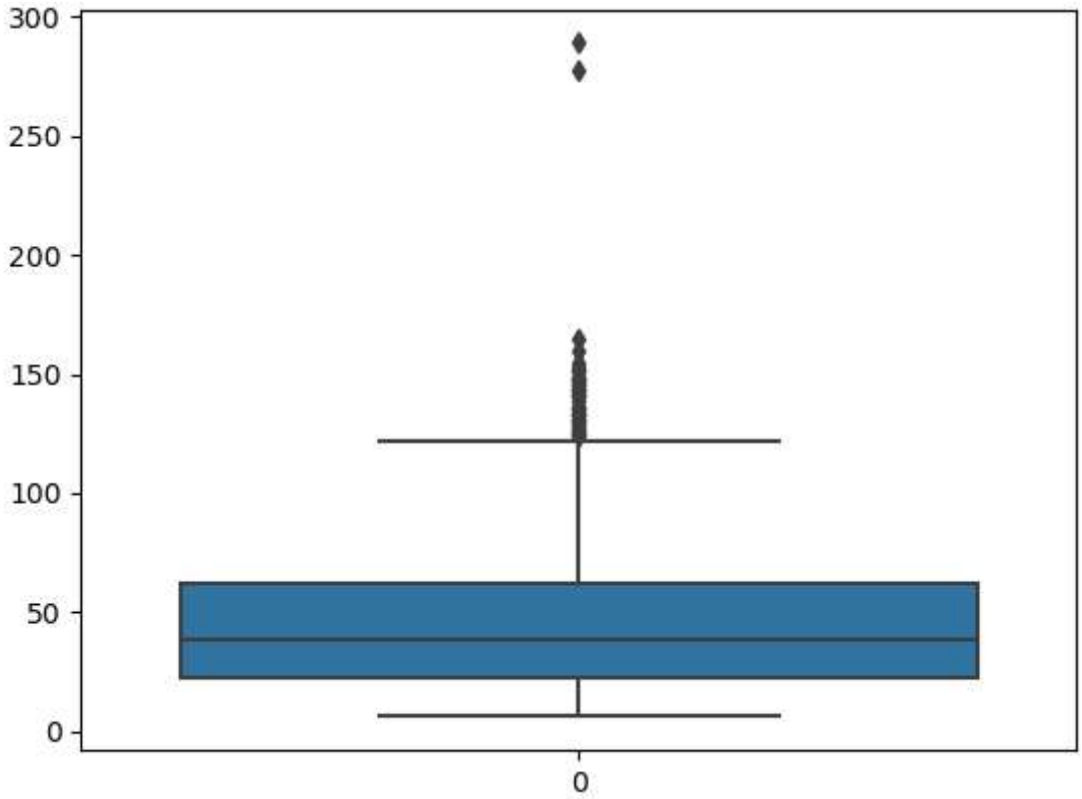
In [72]:
```
sns.boxplot(df['free sulfur dioxide'])
```

Out[72]:
```
<Axes: >
```



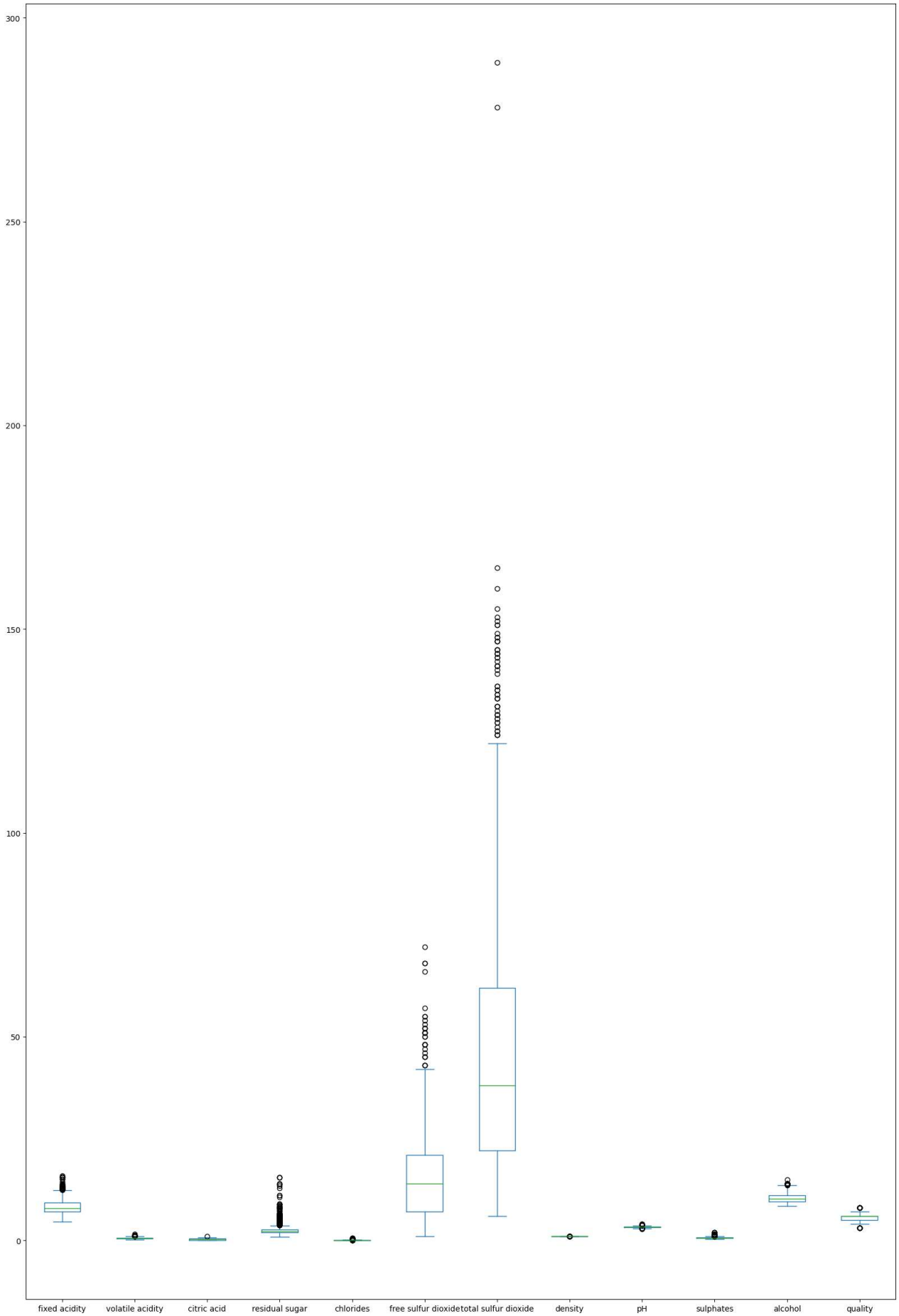
In [73]:
```
sns.boxplot(df['total sulfur dioxide'])
```

Out[73]:
```
<Axes: >
```



In [74]:
```
df.plot(kind='box',figsize=(20,30))
```

Out[74]: <Axes: >


300
250
200
150
100
50
0
fixed acidity
volatile acidity
citric acid
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
quality

there are outiers present in many columns

In [75]:
```
df.skew()
```

Out[75]:
```
fixed acidity           0.982751
volatile acidity        0.671593
citric acid             0.318337
residual sugar          4.540655
chlorides               5.680347
free sulfur dioxide     1.250567
total sulfur dioxide    1.515531
density                 0.071288
pH                      0.193683
sulphates               2.428672
alcohol                 0.860829
quality                 0.217802
dtype: float64
```

there are skewness present in many columns

In [76]:
```
x3=df.drop('quality',axis = 1)
```

In [77]:
```
from sklearn.ensemble import ExtraTreesClassifier
etc= ExtraTreesClassifier()
etc.fit(x3,y3)
```

Out[77]:
```
▾ ExtraTreesClassifier
ExtraTreesClassifier()
```

In [78]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x3,y3, test_size=0.3, random_state=42
print(x_train.shape)
print(x_test.shape)
```

```
(1119, 11)
(480, 11)
```

In [79]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

In [82]:
```
from sklearn.svm import SVC
model= SVC()
model.fit(x_train,y_train)
modely_pred = model.predict(x_test)

from sklearn.metrics import accuracy_score , confusion_matrix
print(accuracy_score(y_test,modely_pred))
```

```
0.88125
```

In [83]:
```
from sklearn.ensemble import RandomForestClassifier
rfc= RandomForestClassifier()
rfc.fit(x_train,y_train)
rfcy_pred = rfc.predict(x_test)

print(accuracy_score(y_test,rfcy_pred))
```

```
0.8770833333333333
```

In [84]:
```
from sklearn.tree import DecisionTreeClassifier
dtc= DecisionTreeClassifier()
dtc.fit(x_train,y_train)
dtcy_pred = dtc.predict(x_test)

print(accuracy_score(y_test,dtcy_pred))
```

```
0.85
```

In [86]:
```
from sklearn.naive_bayes import GaussianNB
gs = GaussianNB()
gs.fit(x_train,y_train)
gsy_pred = gs.predict(x_test)

print(accuracy_score(y_test,gsy_pred))
```

```
0.8395833333333333
```

In [87]:
```
from sklearn.ensemble import GradientBoostingClassifier , AdaBoostClassifier , Bagging
```

In [88]:
```
gbc = GradientBoostingClassifier()
gbc.fit(x_train,y_train)
gbcy_pred = gbc.predict(x_test)

print(accuracy_score(y_test,gbcy_pred))
```

```
0.8645833333333334
```

In [89]:
```
abc = AdaBoostClassifier()
abc.fit(x_train,y_train)
abcy_pred = abc.predict(x_test)

print(accuracy_score(y_test,abcy_pred))
```

```
0.8645833333333334
```

In [90]:
```
bc = BaggingClassifier()
bc.fit(x_train,y_train)
bcy_pred = bc.predict(x_test)

print(accuracy_score(y_test,bcy_pred))
```

```
0.8708333333333333
```

random forest classifier is best till now

In [91]:
```
from sklearn.model_selection import cross_val_score
```

In [92]:
```
recheck = cross_val_score(model,x3,y3)
print(recheck)
print(recheck.mean())
print("difference",accuracy_score(y_test,modely_pred) - recheck.mean())
```

```
[0.865625   0.865625   0.8625     0.8625     0.86520376]
0.8642907523510971
difference 0.016959247648902886
```

In [93]:
```
recheck = cross_val_score(rfc,x3,y3)
print(recheck)
print(recheck.mean())
print("difference",accuracy_score(y_test,rfcy_pred) - recheck.mean())
```

```
[0.875      0.859375   0.871875   0.865625   0.88401254]
0.8711775078369908
difference 0.005905825496342554
```

In [94]:
```
recheck = cross_val_score(dtc,x3,y3)
print(recheck)
print(recheck.mean())
print("difference",accuracy_score(y_test,dtcy_pred) - recheck.mean())
```

```
[0.871875   0.734375   0.8625     0.759375   0.78683386]
0.8029917711598745
difference 0.047008228840125454
```

In [95]:
```
recheck = cross_val_score(gs,x3,y3)
print(recheck)
print(recheck.mean())
print("difference",accuracy_score(y_test,gsy_pred) - recheck.mean())
```

```
[0.871875   0.69375    0.896875   0.703125   0.86833856]
0.8067927115987461
difference 0.032790621734587244
```

In [96]:
```
recheck = cross_val_score(gbc,x3,y3)
print(recheck)
print(recheck.mean())
print("difference",accuracy_score(y_test,gbcy_pred) - recheck.mean())
```

```
[0.8875     0.846875   0.878125   0.84375    0.87460815]
0.8661716300940439
difference -0.0015882967607104836
```

In [97]:
```
recheck = cross_val_score(abc,x3,y3)
print(recheck)
print(recheck.mean())
print("difference",accuracy_score(y_test,abcy_pred) - recheck.mean())
```

```
[0.88125    0.80625    0.89375    0.84375    0.86206897]
0.8574137931034482
difference 0.007169540229885141
```