In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as pp
```

In [2]:

```
df1 = pd.read_csv(r"C:\Users\user\Desktop\c10\madrid_2004.csv")
df = df1.head(1000)
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.990
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.480
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.070
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.080
...	...	...	...	...	...	...	...	...	...	...	
995	2004-08-02 13:00:00	NaN	0.47	NaN	NaN	NaN	84.260002	146.100006	NaN	39.549999	53.759
996	2004-08-02 13:00:00	NaN	0.22	NaN	NaN	0.54	51.709999	69.400002	NaN	62.310001	65.510
997	2004-08-02 13:00:00	NaN	0.33	NaN	NaN	NaN	48.009998	64.550003	NaN	58.240002	38.389
998	2004-08-02 13:00:00	4.57	0.52	2.45	NaN	0.05	87.779999	138.199997	NaN	51.430000	53.259
999	2004-08-02 13:00:00	NaN	0.13	NaN	NaN	NaN	55.820000	75.260002	NaN	63.910000	49.150

1000 rows × 17 columns

In [3]:

```
df=df.dropna()
```

In [4]:

```
df.columns
```

Out[4]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O
_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 5 to 989
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     100 non-null    object 
 1   BEN      100 non-null    float64
 2   CO       100 non-null    float64
 3   EBE      100 non-null    float64
 4   MXY      100 non-null    float64
 5   NMHC     100 non-null    float64
 6   NO_2     100 non-null    float64
 7   NOx      100 non-null    float64
 8   OXY      100 non-null    float64
 9   O_3      100 non-null    float64
 10  PM10     100 non-null    float64
 11  PM25     100 non-null    float64
 12  PXY      100 non-null    float64
 13  SO_2     100 non-null    float64
 14  TCH      100 non-null    float64
 15  TOL      100 non-null    float64
 16  station  100 non-null    int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 14.1+ KB
```

In [6]:

```
data=df[['CO' ,'station']]
data
```

Out[6]:

	CO	station
5	0.63	28079006
22	0.36	28079024
26	0.46	28079099
32	0.67	28079006
49	0.30	28079024
...	...	...
955	0.42	28079099
961	1.25	28079006
979	0.22	28079024
983	0.44	28079099
989	1.24	28079006

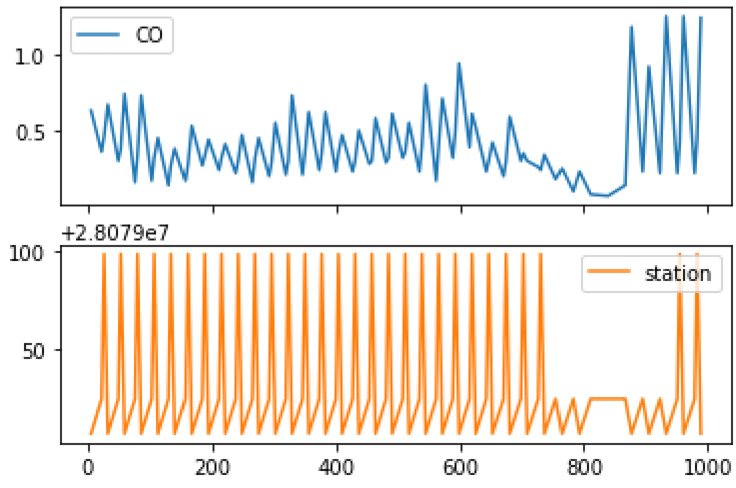
100 rows × 2 columns

In [7]:

```
data.plot.line(subplots=True)
```

Out[7]:

```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```

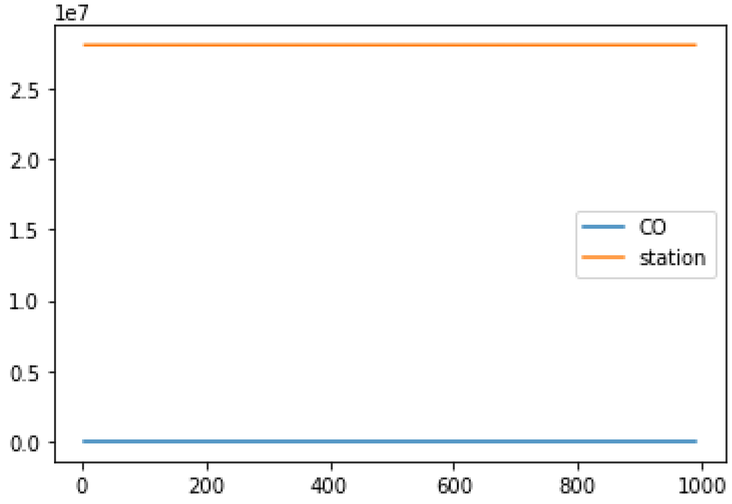


In [8]:

```
data.plot.line()
```

Out[8]:

<AxesSubplot:>



In [9]:

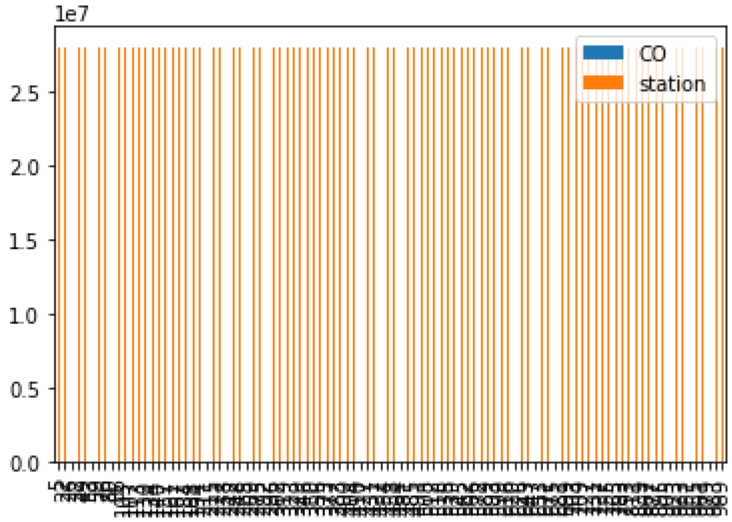
```
x = data[0:100]
```

In [10]:

```
x.plot.bar()
```

Out[10]:

<AxesSubplot:>

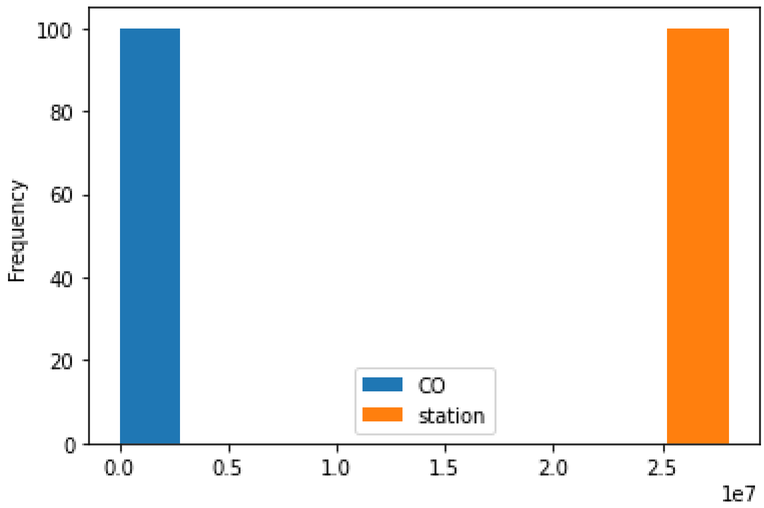


In [11]:

```
data.plot.hist()
```

Out[11]:

```
<AxesSubplot:ylabel='Frequency'>
```

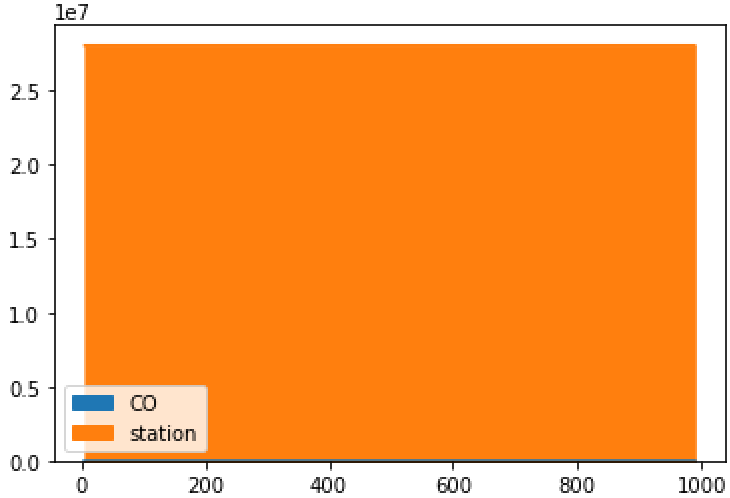


In [12]:

```
data.plot.area()
```

Out[12]:

```
<AxesSubplot:>
```

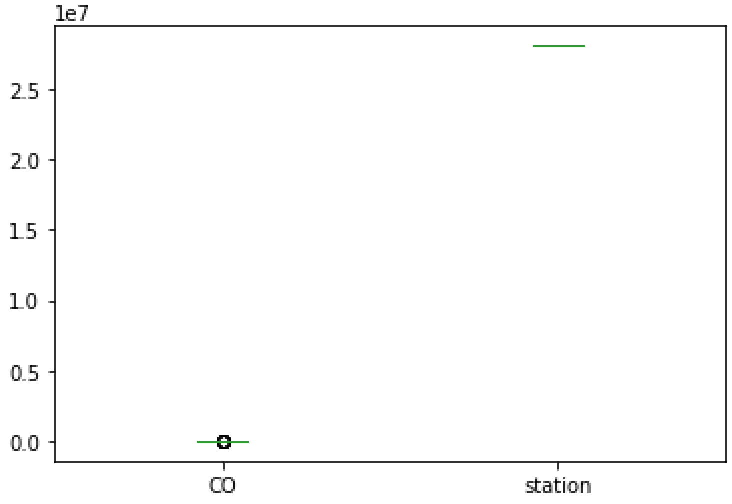


In [13]:

```
data.plot.box()
```

Out[13]:

<AxesSubplot:>



In [14]:

```
x.plot.pie(y='station' )
```

Out[14]:

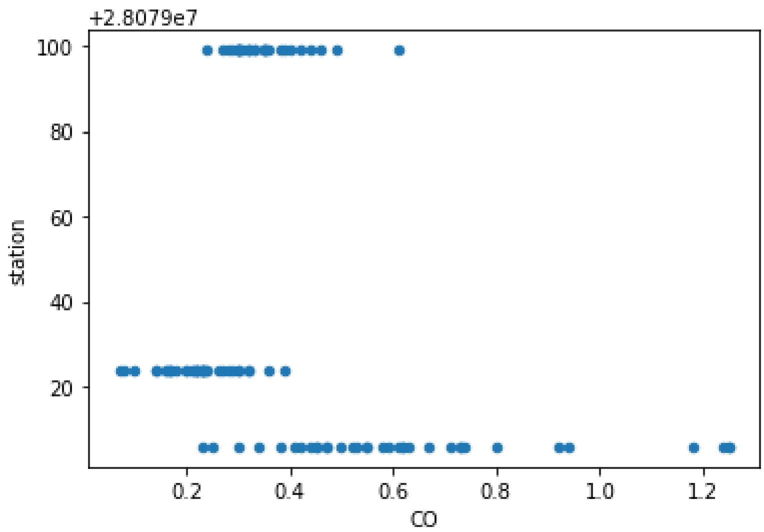
```
<AxesSubplot:ylabel='station'>
```

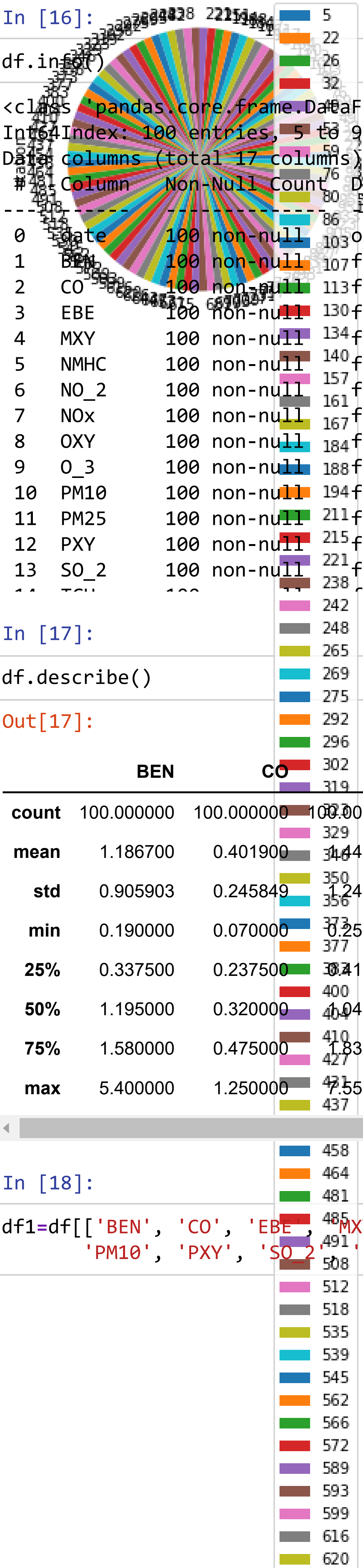
In [15]:

```
data.plot.scatter(x='CO' ,y='station')
```

Out[15]:

```
<AxesSubplot:xlabel='CO', ylabel='station'>
```





In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 5 to 989
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     100 non-null    object
 1   BEN      100 non-null    float64
 2   CO       100 non-null    float64
 3   EBE      100 non-null    float64
 4   MXY      100 non-null    float64
 5   NMHC     100 non-null    float64
 6   NO_2     100 non-null    float64
 7   NOx      100 non-null    float64
 8   OXY      100 non-null    float64
 9   O_3      100 non-null    float64
 10  PM10     100 non-null    float64
 11  PM25     100 non-null    float64
 12  PXY      100 non-null    float64
 13  SO_2     100 non-null    float64
```

In [17]:

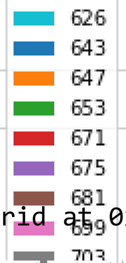
```
df.describe()
```

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
mean	1.186700	0.401900	1.442300	2.388100	0.093200	58.240300	82.661400
std	0.905903	0.245849	1.247769	2.270227	0.072389	30.625082	60.783607
min	0.190000	0.070000	0.250000	0.230000	0.000000	4.540000	4.630000
25%	0.337500	0.237500	0.415000	0.572500	0.040000	35.147499	41.542499
50%	1.195000	0.320000	1.040000	1.935000	0.065000	53.105001	63.610001
75%	1.580000	0.475000	1.832500	3.095000	0.150000	74.535000	108.750002
max	5.400000	1.250000	7.550000	14.080000	0.300000	135.000000	279.799988

In [18]:

```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

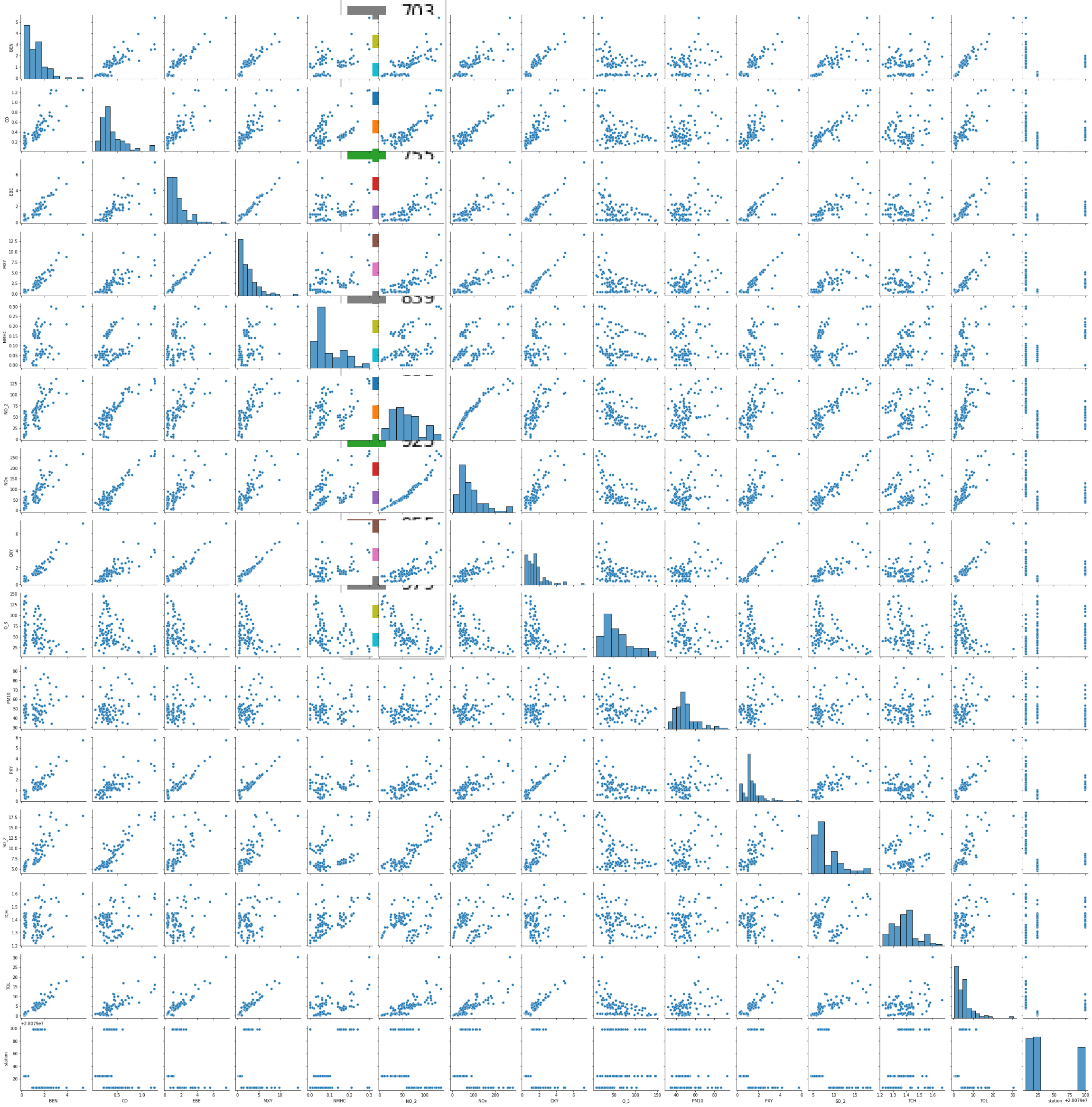


In [19]:

```
sb.pairplot(df1[0:100])
```

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x20c0e681310>
```



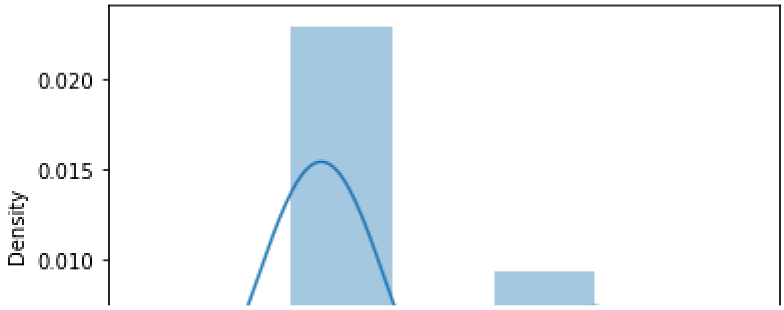
In [20]:

```
sb.distplot(df1['station'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:255
7: FutureWarning: `distplot` is a deprecated function and will be remove
d in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-l
evel function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[20]:

```
<AxesSubplot:xlabel='station', ylabel='Density'>
```

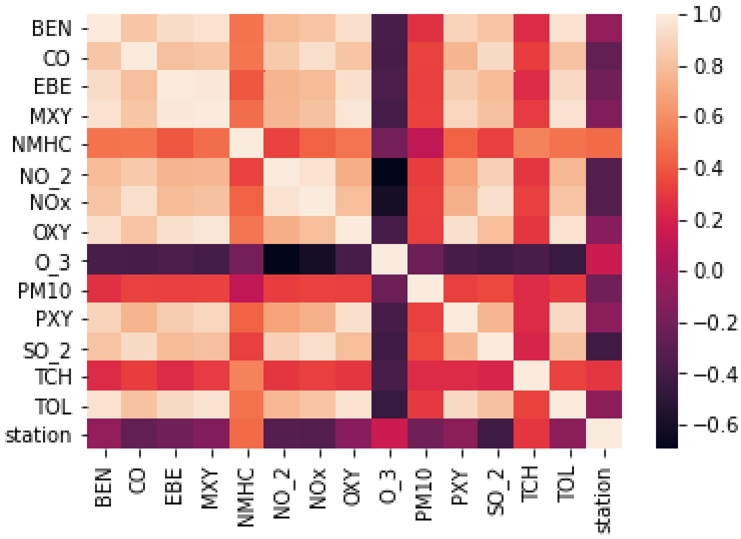


In [21]:

```
sb.heatmap(df1.corr())
```

Out[21]:

```
<AxesSubplot:>
```



In [22]:

```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [23]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [24]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[24]:

```
LinearRegression()
```

In [25]:

```
lr.intercept_
```

Out[25]:

```
28078950.82491036
```

In [26]:

```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

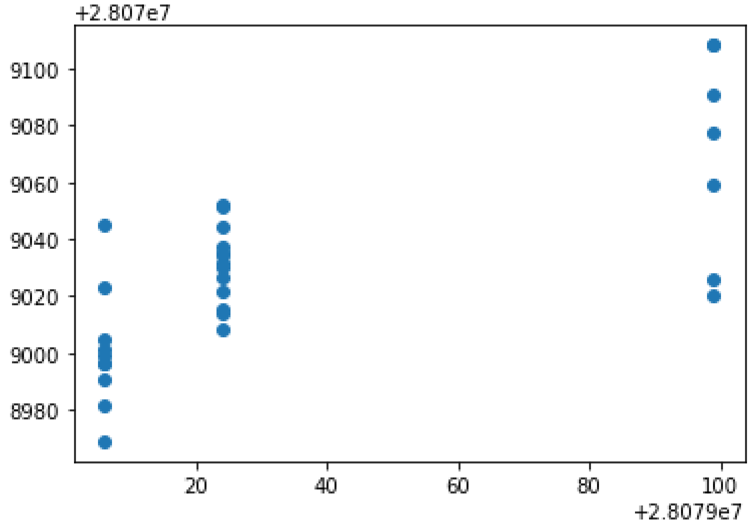
	Co-efficient
BEN	61.204819
CO	-25.447483
EBE	-13.844988
MXY	4.489134
NMHC	352.403422
NO_2	-0.083407
NOx	-0.468326
OXY	-14.863733
O_3	-0.171765
PM10	-0.167580
PXY	12.890044
SO_2	-1.868694
TCH	81.800858
TOL	-6.188675

In [27]:

```
prediction =lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[27]:

```
<matplotlib.collections.PathCollection at 0x20c1d299eb0>
```



In [28]:

```
lr.score(x_test,y_test)
```

Out[28]:

```
0.45557467769186066
```

In [29]:

```
lr.score(x_train,y_train)
```

Out[29]:

```
0.791158550959588
```

In [30]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [31]:

```
r=Ridge(alpha=10)
r.fit(x_train,y_train)
```

Out[31]:

```
Ridge(alpha=10)
```

In [32]:

```
r.score(x_test,y_test)
```

Out[32]:

```
0.02223624112412015
```

In [33]:

```
r.score(x_train,y_train)
```

Out[33]:

```
0.48343210561710614
```

In [34]:

```
l=Lasso(alpha=10)
l.fit(x_train,y_train)
```

Out[34]:

```
Lasso(alpha=10)
```

In [35]:

```
l.score(x_train,y_train)
```

Out[35]:

```
0.2970261802948704
```

In [36]:

```
l.score(x_test,y_test)
```

Out[36]:

```
-0.2989841161260207
```

In [37]:

```
from sklearn.linear_model import ElasticNet
e=ElasticNet()
e.fit(x_train,y_train)
```

Out[37]:

```
ElasticNet()
```

In [38]:

```
e.coef_
```

Out[38]:

```
array([ 4.1214755 ,  0.        , -2.75673309,  0.81676049,  1.5259761 ,
       -0.62194175,  0.31148644,  1.53679929,  0.31526474, -0.1891754 ,
        2.19191466, -8.99692726,  0.79847353,  2.9275602 ])
```

In [39]:

```
e.intercept_
```

Out[39]:

```
28079094.52590053
```

In [40]:

```
prediction=e.predict(x_test)
```

In [41]:

```
e.score(x_test,y_test)
```

Out[41]:

```
-0.12667668616532257
```

In [42]:

```
from sklearn import metrics
```

In [43]:

```
print(metrics.mean_squared_error(y_test,prediction))
```

```
1451.4412409524768
```

In [44]:

```
print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
38.09778524996534
```

In [45]:

```
print(metrics.mean_absolute_error(y_test,prediction))
```

```
33.73625044661264
```

In [46]:

```
from sklearn.linear_model import LogisticRegression
```

In [47]:

```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [48]:

```
feature_matrix.shape
```

Out[48]:

```
(100, 14)
```

In [49]:

```
target_vector.shape
```

Out[49]:

```
(100,)
```

In [50]:

```
from sklearn.preprocessing import StandardScaler
```

In [51]:

```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [52]:

```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[52]:

```
LogisticRegression(max_iter=10000)
```

In [53]:

```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [54]:

```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079006]
```

In [55]:

```
logr.classes_
```

Out[55]:

```
array([28079006, 28079024, 28079099], dtype=int64)
```

In [56]:

```
logr.score(fs,target_vector)
```

Out[56]:

```
1.0
```

In [57]:

```
logr.predict_proba(observation)[0][0]
```

Out[57]:

```
0.9901352554131048
```

In [58]:

```
logr.predict_proba(observation)
```

Out[58]:

```
array([[9.90135255e-01, 4.89040800e-29, 9.86474459e-03]])
```

In [59]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [60]:

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[60]:

```
RandomForestClassifier()
```

In [61]:

```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [62]:

```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[62]:

```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [63]:

```
grid_search.best_score_
```

Out[63]:

```
0.9857142857142858
```

In [64]:

```
rfc_best=grid_search.best_estimator_
```

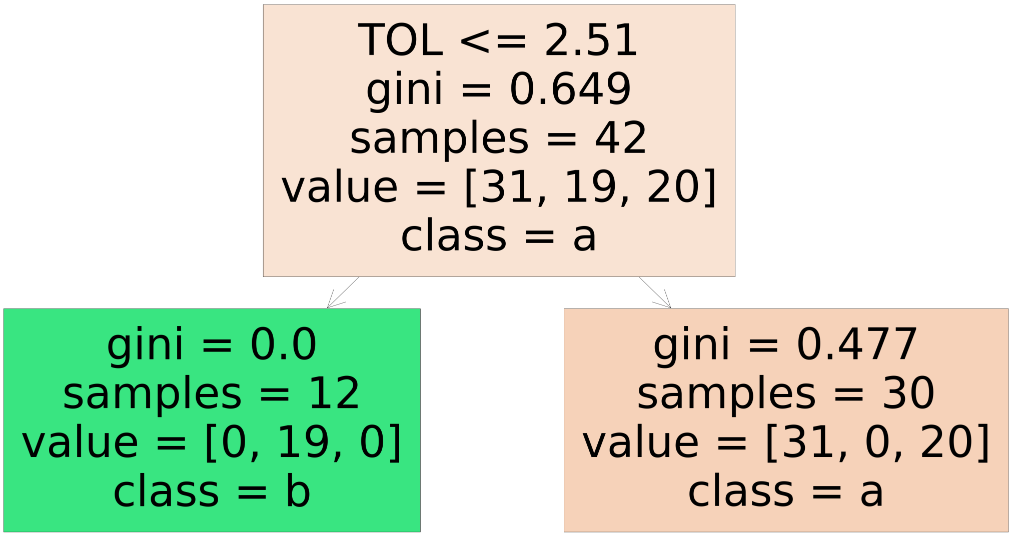
In [65]:

```
from sklearn.tree import plot_tree

pp.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b','c','d'],f
```

Out[65]:

```
[Text(2232.0, 1630.8000000000002, 'TOL <= 2.51\ngini = 0.649\nsamples = 42
\nvalue = [31, 19, 20]\nclass = a'),
 Text(1116.0, 543.5999999999999, 'gini = 0.0\nsamples = 12\nvalue = [0, 1
9, 0]\nclass = b'),
 Text(3348.0, 543.5999999999999, 'gini = 0.477\nsamples = 30\nvalue = [31,
0, 20]\nclass = a')]
```



random forest is best suitable for this data set

In [ ]: