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_2008.csv")
df = df1.head(1000)
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
0	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000	16.88
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999	19.04
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000	20.27
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001	10.85
4	2008-06-01 01:00:00	1.68	0.80	1.7	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.16
...	...	...	...	...	...	...	...	...	...	...	
995	2008-06-02 15:00:00	NaN	0.33	NaN	NaN	NaN	37.880001	62.980000	NaN	60.330002	19.68
996	2008-06-02 15:00:00	NaN	0.40	NaN	NaN	NaN	56.400002	90.769997	NaN	40.150002	14.34
997	2008-06-02 15:00:00	NaN	0.41	NaN	NaN	0.22	90.139999	165.800003	NaN	33.180000	24.76
998	2008-06-02 15:00:00	NaN	0.44	NaN	NaN	NaN	35.950001	54.740002	NaN	60.410000	15.06
999	2008-06-02 15:00:00	NaN	0.13	NaN	NaN	NaN	20.879999	30.410000	NaN	80.629997	16.33

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: 115 entries, 4 to 992
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     115 non-null    object 
 1   BEN      115 non-null    float64
 2   CO       115 non-null    float64
 3   EBE      115 non-null    float64
 4   MXY      115 non-null    float64
 5   NMHC     115 non-null    float64
 6   NO_2     115 non-null    float64
 7   NOx      115 non-null    float64
 8   OXY      115 non-null    float64
 9   O_3      115 non-null    float64
 10  PM10     115 non-null    float64
 11  PM25     115 non-null    float64
 12  PXY      115 non-null    float64
 13  SO_2     115 non-null    float64
 14  TCH      115 non-null    float64
 15  TOL      115 non-null    float64
 16  station  115 non-null    int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 16.2+ KB
```

In [6]:

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

Out[6]:

	CO	station
4	0.80	28079006
21	0.37	28079024
25	0.39	28079099
30	0.51	28079006
47	0.39	28079024
...	...	...
961	0.36	28079099
966	0.80	28079006
983	0.33	28079024
987	0.36	28079099
992	0.71	28079006

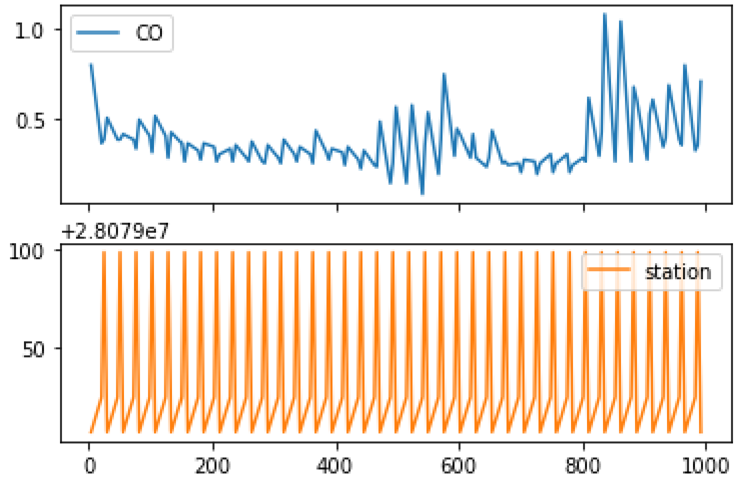
115 rows × 2 columns

In [7]:

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

Out[7]:

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

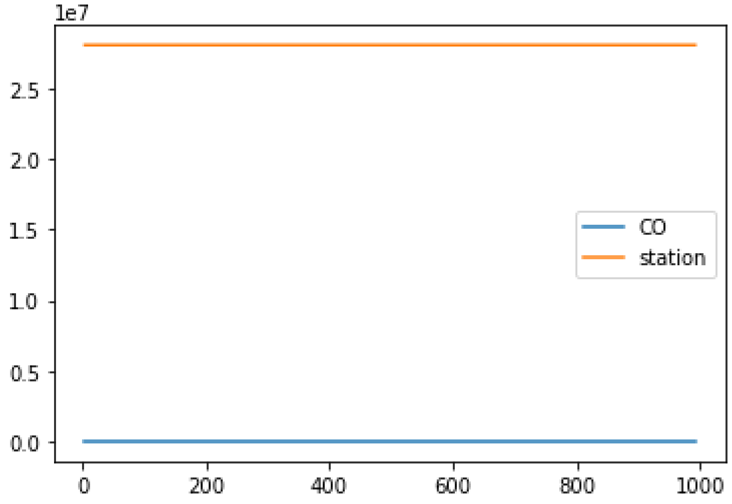


In [8]:

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

Out[8]:

<AxesSubplot:>



In [9]:

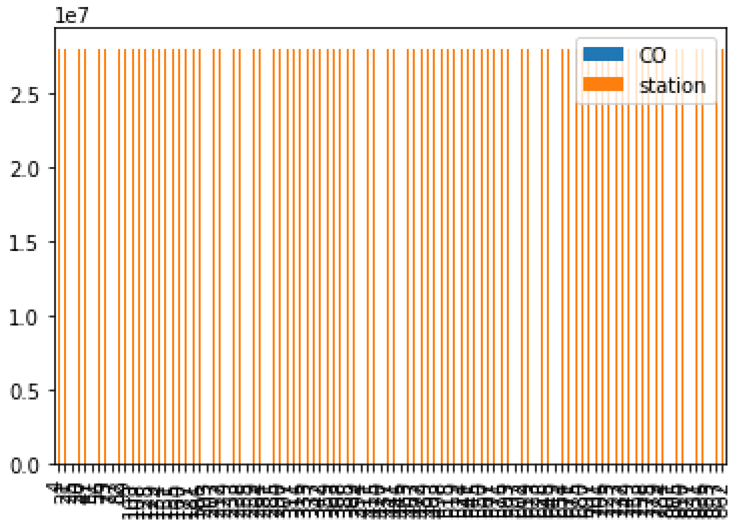
```
x = data[0:100]
```

In [10]:

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

Out[10]:

<AxesSubplot:>

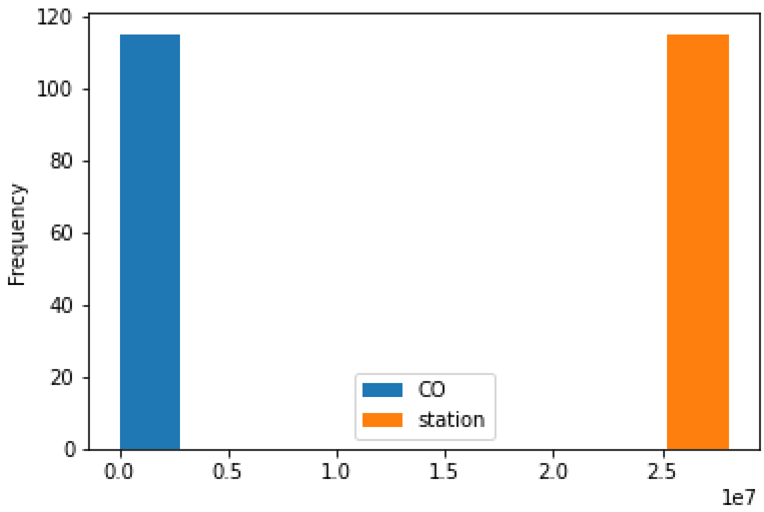


In [11]:

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

Out[11]:

<AxesSubplot:ylabel='Frequency'>

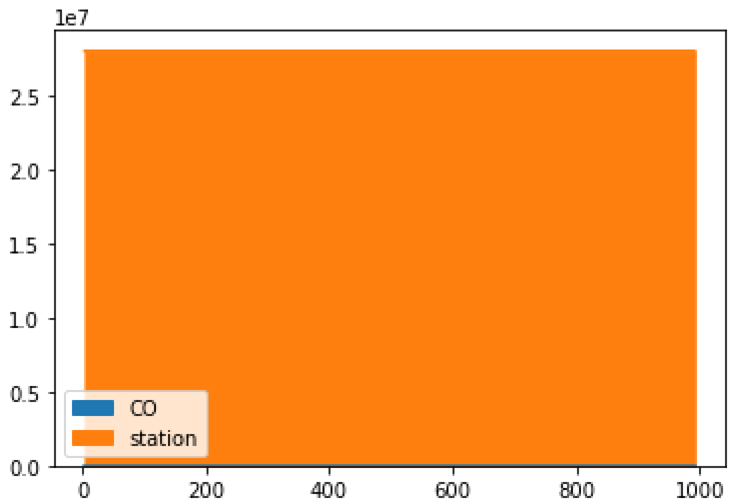


In [12]:

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

Out[12]:

<AxesSubplot:>

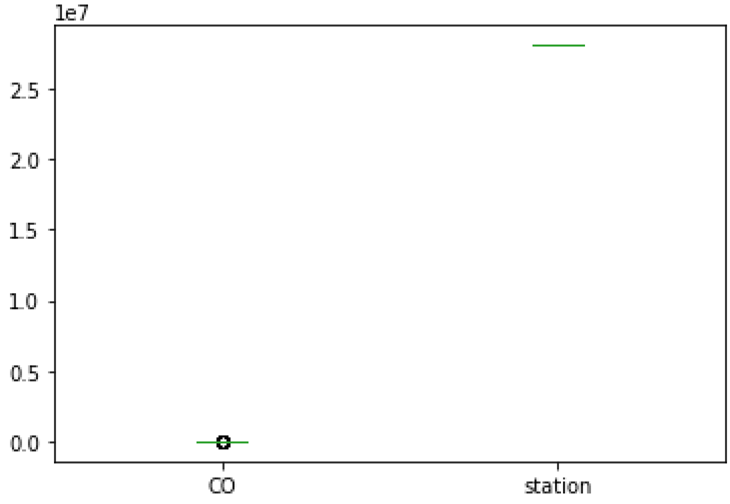


In [13]:

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

Out[13]:

<AxesSubplot:>



In [14]:

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

Out[14]:

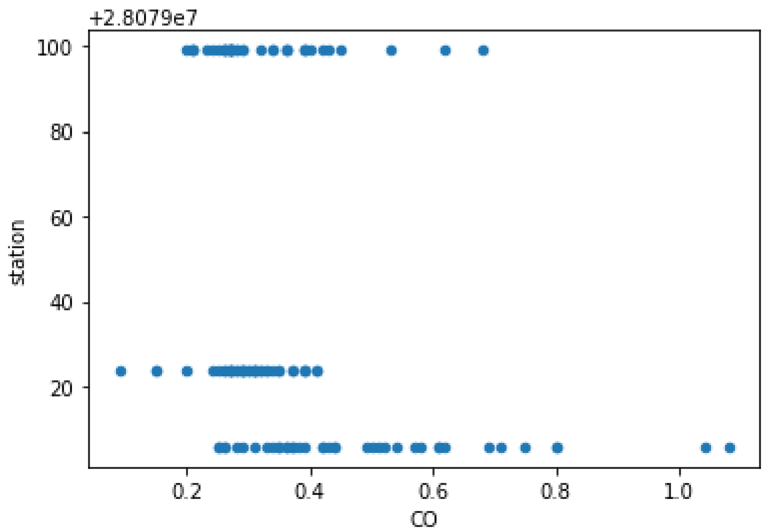
```
<AxesSubplot:ylabel='station'>
```

In [15]:

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

Out[15]:

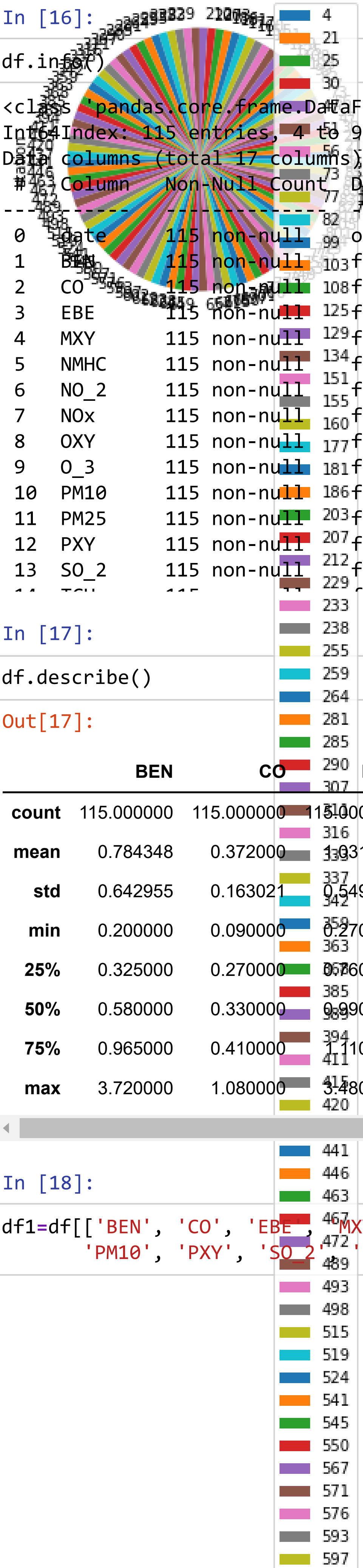
```
<AxesSubplot:xlabel='CO', ylabel='station'>
```



In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 115 entries, 4 to 992
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     115 non-null    object
 1   BEN      115 non-null    float64
 2   CO       115 non-null    float64
 3   EBE      115 non-null    float64
 4   MXY      115 non-null    float64
 5   NMHC     115 non-null    float64
 6   NO_2     115 non-null    float64
 7   NOx      115 non-null    float64
 8   OXY      115 non-null    float64
 9   O_3      115 non-null    float64
 10  PM10     115 non-null    float64
 11  PM25     115 non-null    float64
 12  PXY      115 non-null    float64
 13  SO_2     115 non-null    float64
```



In [17]:

```
df.describe()
```

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	
count	115.000000	115.000000	115.000000	115.000000	115.000000	115.000000	115.000000	1
mean	0.784348	0.372000	1.031739	1.581130	0.229739	41.478869	68.269826	
std	0.642955	0.163021	0.549865	1.258218	0.064841	25.367862	55.714292	
min	0.200000	0.090000	0.270000	0.280000	0.130000	7.220000	7.820000	
25%	0.325000	0.270000	0.760000	0.830000	0.180000	20.110000	21.025001	
50%	0.580000	0.330000	0.990000	1.200000	0.230000	39.529999	55.910000	
75%	0.965000	0.410000	1.110000	1.810000	0.280000	59.455000	94.840000	
max	3.720000	1.080000	3.480000	7.220000	0.400000	105.199997	261.299988	

In [18]:

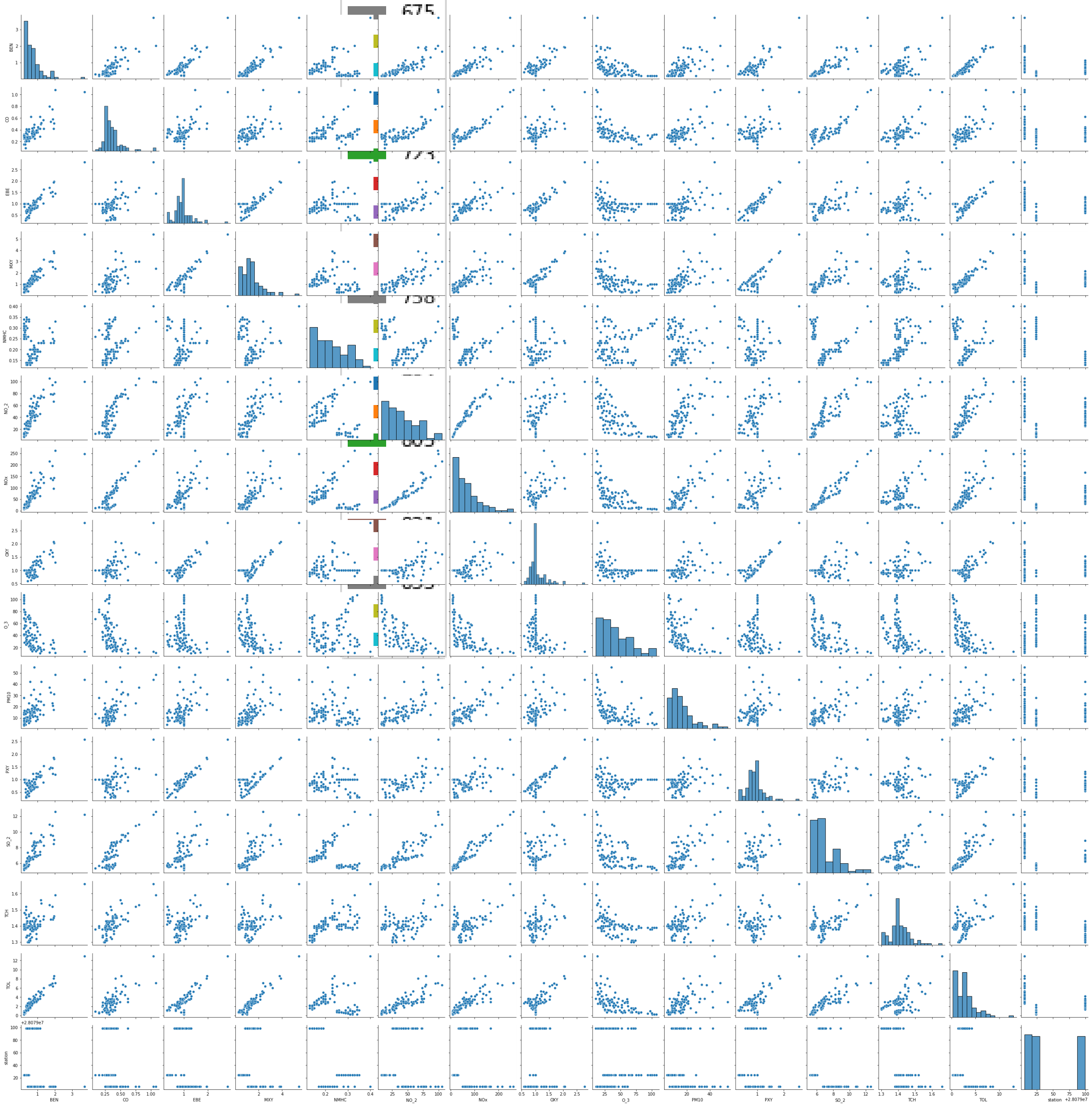
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [19]:

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

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x27b42a895e0>
```



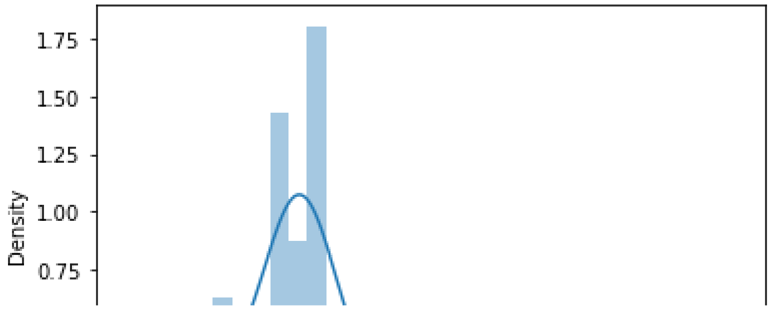
In [20]:

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

```
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='EBE', ylabel='Density'>
```

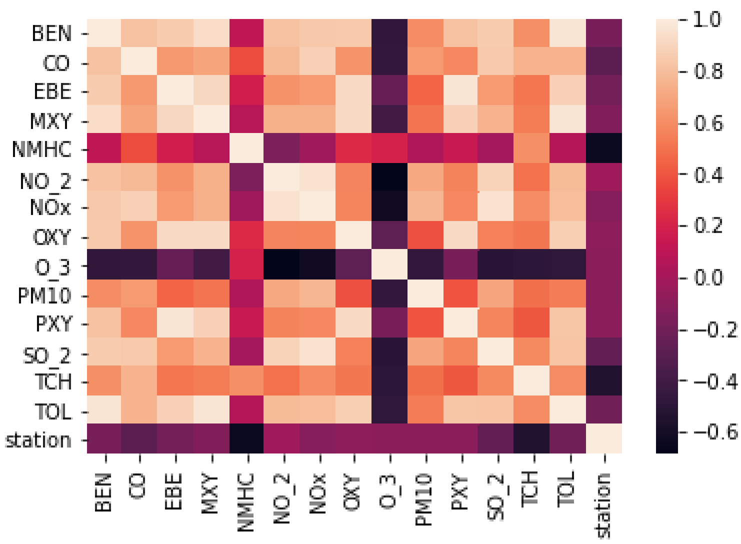


In [21]:

```
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]:

```
28079394.130565256
```

In [26]:

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

Out[26]:

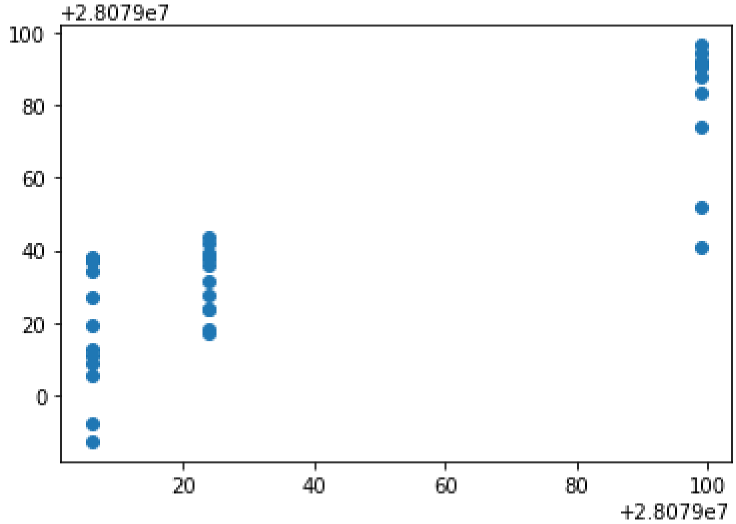
	Co-efficient
BEN	37.729564
CO	155.536772
EBE	-52.152137
MXY	4.971952
NMHC	-606.390468
NO_2	-1.184477
NOx	0.894408
OXY	31.328250
O_3	-0.159744
PM10	0.063305
PXY	34.473614
SO_2	-36.878906
TCH	-24.736933
TOL	-9.911559

In [27]:

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

Out[27]:

```
<matplotlib.collections.PathCollection at 0x27b4fc46820>
```



In [28]:

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

Out[28]:

```
0.7258584040919333
```

In [29]:

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

Out[29]:

```
0.8379774376260507
```

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]:

```
0.0402307309182115
```

In [33]:

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

Out[33]:

```
0.3975639924321591
```

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]:

```
0.2070558661849341
```

In [36]:

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

Out[36]:

```
-0.03696523676963026
```

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([-0.        , -0.29490992, -0.        ,  0.        , -1.41083051,
        1.44065892, -0.18754318,  0.60838148, -0.21984289, -0.74671763,
        1.77862973, -9.7443143 , -0.65729606, -4.60944056])
```

In [39]:

```
e.intercept_
```

Out[39]:

```
28079102.500512496
```

In [40]:

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

In [41]:

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

Out[41]:

```
-0.029350012320308316
```

In [42]:

```
from sklearn import metrics
```

In [43]:

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

```
1554.4563254624582
```

In [44]:

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

```
39.426594139773954
```

In [45]:

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

```
35.244250322771926
```

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]:

```
(115, 14)
```

In [49]:

```
target_vector.shape
```

Out[49]:

```
(115,)
```

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]:

```
1.0
```

In [58]:

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

Out[58]:

```
array([[1.00000000e+00, 2.29589992e-20, 8.12759877e-25]])
```

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]:

```
0.95
```

In [64]:

```
rfc_best=grid_search.best_estimator_
```

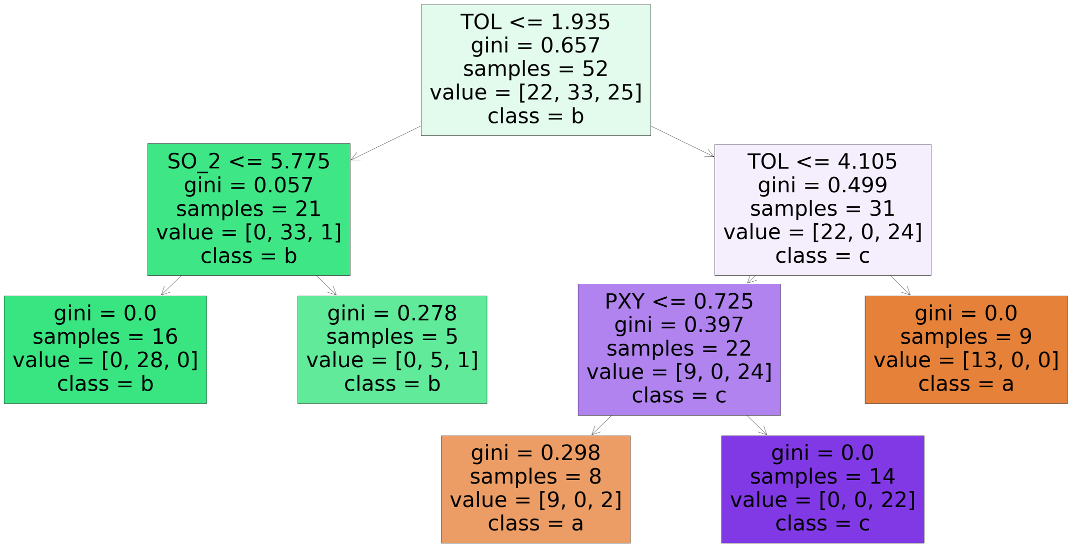
In [65]:

```
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, 1902.6000000000001, 'TOL <= 1.935\ngini = 0.657\nsamples = 5
2\nvalue = [22, 33, 25]\nclass = b'),
 Text(1116.0, 1359.0, 'SO_2 <= 5.775\ngini = 0.057\nsamples = 21\nvalue =
[0, 33, 1]\nclass = b'),
 Text(558.0, 815.4000000000001, 'gini = 0.0\nsamples = 16\nvalue = [0, 28,
0]\nclass = b'),
 Text(1674.0, 815.4000000000001, 'gini = 0.278\nsamples = 5\nvalue = [0,
5, 1]\nclass = b'),
 Text(3348.0, 1359.0, 'TOL <= 4.105\ngini = 0.499\nsamples = 31\nvalue =
[22, 0, 24]\nclass = c'),
 Text(2790.0, 815.4000000000001, 'PXY <= 0.725\ngini = 0.397\nsamples = 22
\nvalue = [9, 0, 24]\nclass = c'),
 Text(2232.0, 271.79999999999995, 'gini = 0.298\nsamples = 8\nvalue = [9,
0, 2]\nclass = a'),
 Text(3348.0, 271.79999999999995, 'gini = 0.0\nsamples = 14\nvalue = [0,
0, 22]\nclass = c'),
 Text(3906.0, 815.4000000000001, 'gini = 0.0\nsamples = 9\nvalue = [13, 0,
0]\nclass = a')]
```



random forest is best suitable for this model

In [ ]: