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_2006.csv")
df = df1.head(1000)
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM1(
0	2006-02-01 01:00:00	NaN	1.84	NaN	NaN	NaN	155.100006	490.100006	NaN	4.88	97.57000(
1	2006-02-01 01:00:00	1.68	1.01	2.38	6.36	0.32	94.339996	229.699997	3.04	7.10	25.82000(
2	2006-02-01 01:00:00	NaN	1.25	NaN	NaN	NaN	66.800003	192.000000	NaN	4.43	34.41999
3	2006-02-01 01:00:00	NaN	1.68	NaN	NaN	NaN	103.000000	407.799988	NaN	4.83	28.26000(
4	2006-02-01 01:00:00	NaN	1.31	NaN	NaN	NaN	105.400002	269.200012	NaN	6.99	54.18000(
...	...	...	...	...	...	...	...	...	...	...	..
995	2006-02-02 15:00:00	0.20	0.92	0.13	NaN	0.32	135.000000	271.299988	NaN	12.48	98.87999
996	2006-02-02 15:00:00	NaN	0.96	NaN	NaN	NaN	134.199997	230.300003	NaN	8.23	77.29000
997	2006-02-02 15:00:00	NaN	1.21	NaN	NaN	NaN	141.699997	251.399994	NaN	14.56	146.60000(
998	2006-02-02 15:00:00	NaN	1.38	NaN	NaN	0.35	83.239998	218.399994	NaN	8.91	94.30999
999	2006-02-02 15:00:00	NaN	1.22	NaN	NaN	NaN	83.820000	237.500000	NaN	8.03	91.66000

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: 114 entries, 5 to 993
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     114 non-null    object 
 1   BEN      114 non-null    float64
 2   CO       114 non-null    float64
 3   EBE      114 non-null    float64
 4   MXY      114 non-null    float64
 5   NMHC     114 non-null    float64
 6   NO_2     114 non-null    float64
 7   NOx      114 non-null    float64
 8   OXY      114 non-null    float64
 9   O_3      114 non-null    float64
 10  PM10     114 non-null    float64
 11  PM25     114 non-null    float64
 12  PXY      114 non-null    float64
 13  SO_2     114 non-null    float64
 14  TCH      114 non-null    float64
 15  TOL      114 non-null    float64
 16  station  114 non-null    int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 16.0+ KB
```

In [6]:

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

Out[6]:

	CO	station
5	1.69	28079006
22	0.79	28079024
25	1.47	28079099
31	0.85	28079006
48	0.79	28079024
...	...	...
961	1.25	28079099
967	1.40	28079006
984	0.83	28079024
987	1.02	28079099
993	1.43	28079006

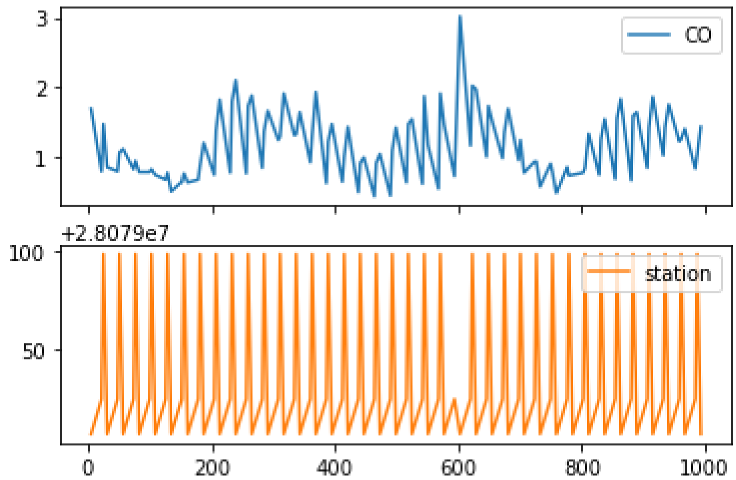
114 rows × 2 columns

In [7]:

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

Out[7]:

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

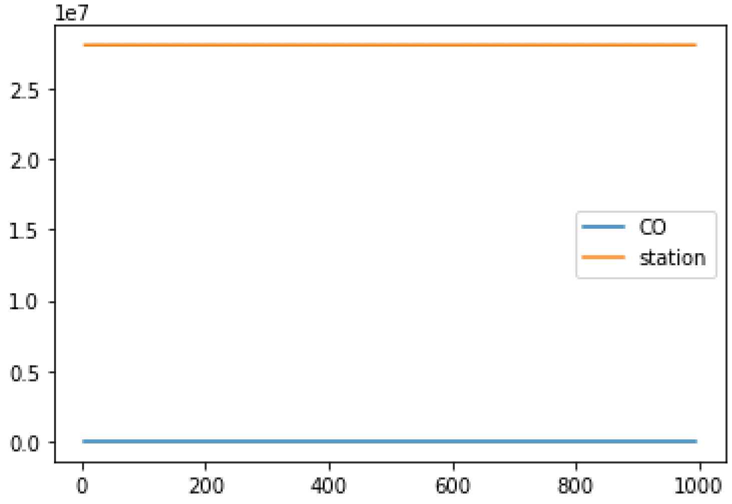


In [8]:

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

Out[8]:

<AxesSubplot:>



In [9]:

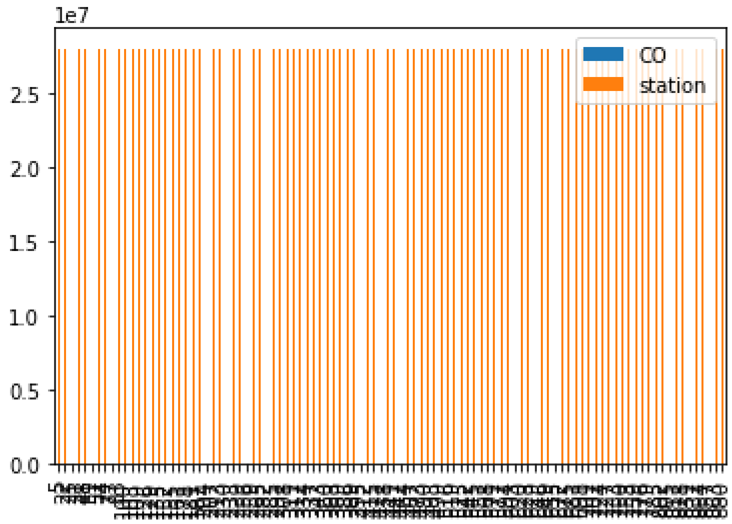
```
x = data[0:100]
```

In [10]:

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

Out[10]:

<AxesSubplot:>

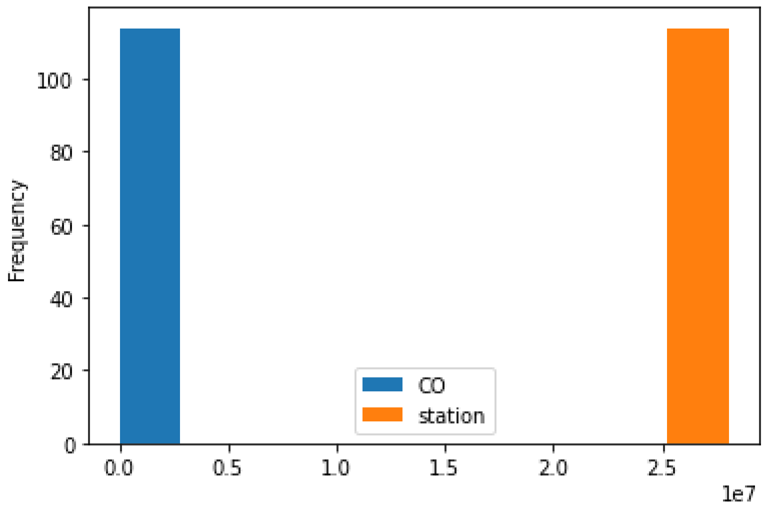


In [11]:

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

Out[11]:

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

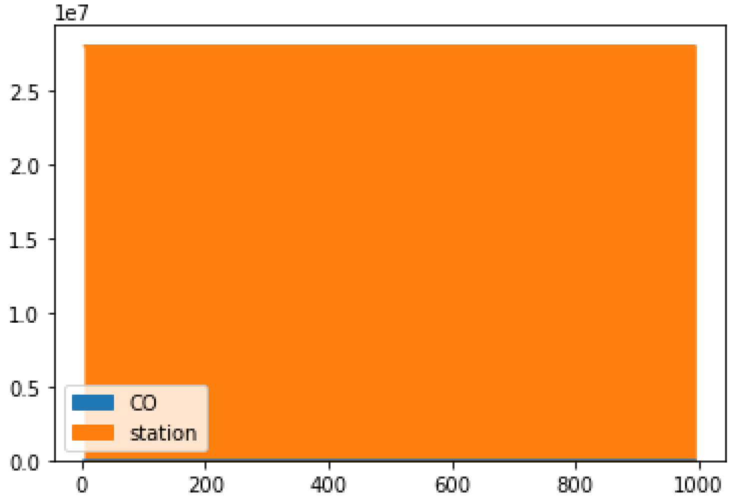


In [12]:

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

Out[12]:

```
<AxesSubplot:>
```

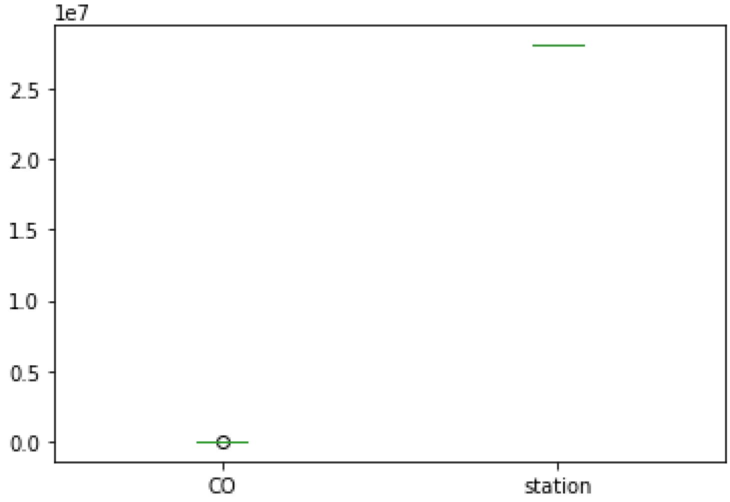


In [13]:

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

Out[13]:

<AxesSubplot:>



In [14]:

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

Out[14]:

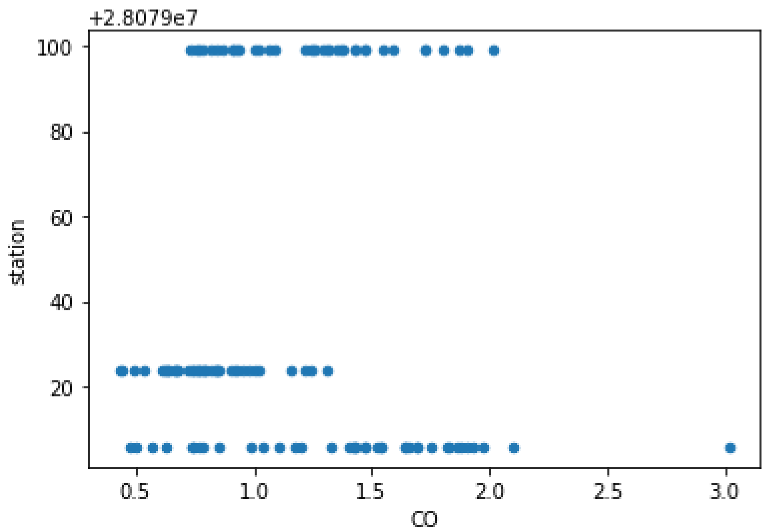
```
<AxesSubplot:ylabel='station'>
```

In [15]:

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

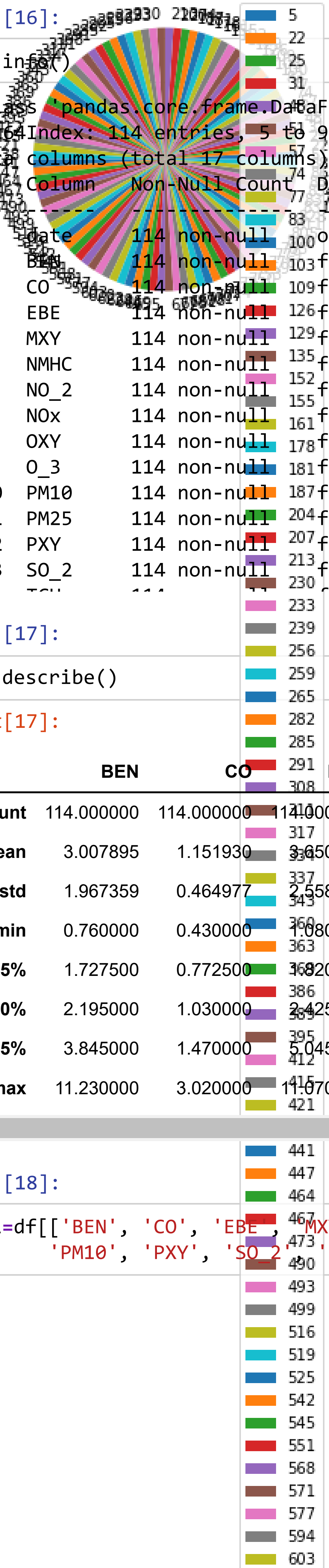
Out[15]:

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



In [16]:

```
df.info()
```




```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 114 entries, 5 to 993
Data columns (total 17 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   date      114 non-null    object
 1   BEN       114 non-null    float64
 2   CO        114 non-null    float64
 3   EBE       114 non-null    float64
 4   MXY       114 non-null    float64
 5   NMHC      114 non-null    float64
 6   NO_2      114 non-null    float64
 7   NOx       114 non-null    float64
 8   OXY       114 non-null    float64
 9   O_3       114 non-null    float64
 10  PM10      114 non-null    float64
 11  PM25      114 non-null    float64
 12  PXY       114 non-null    float64
 13  SO_2      114 non-null    float64
```

In [17]:

```
df.describe()
```

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx
count	114.000000	114.000000	114.000000	114.000000	114.000000	114.000000	114.000000
mean	3.007895	1.151930	3.650088	9.002018	0.282018	90.532982	256.107017
std	1.967359	0.464977	2.558025	4.874614	0.119432	30.213987	128.404652
min	0.760000	0.430000	1.080000	1.860000	0.090000	35.290001	55.320000
25%	1.727500	0.772500	1.820000	5.097500	0.180000	65.037500	144.549999
50%	2.195000	1.030000	2.425000	7.730000	0.270000	92.469997	239.650002
75%	3.845000	1.470000	5.045000	11.307500	0.360000	112.775002	337.174988
max	11.230000	3.020000	11.070000	22.920000	0.740000	164.500000	832.500000

In [18]:

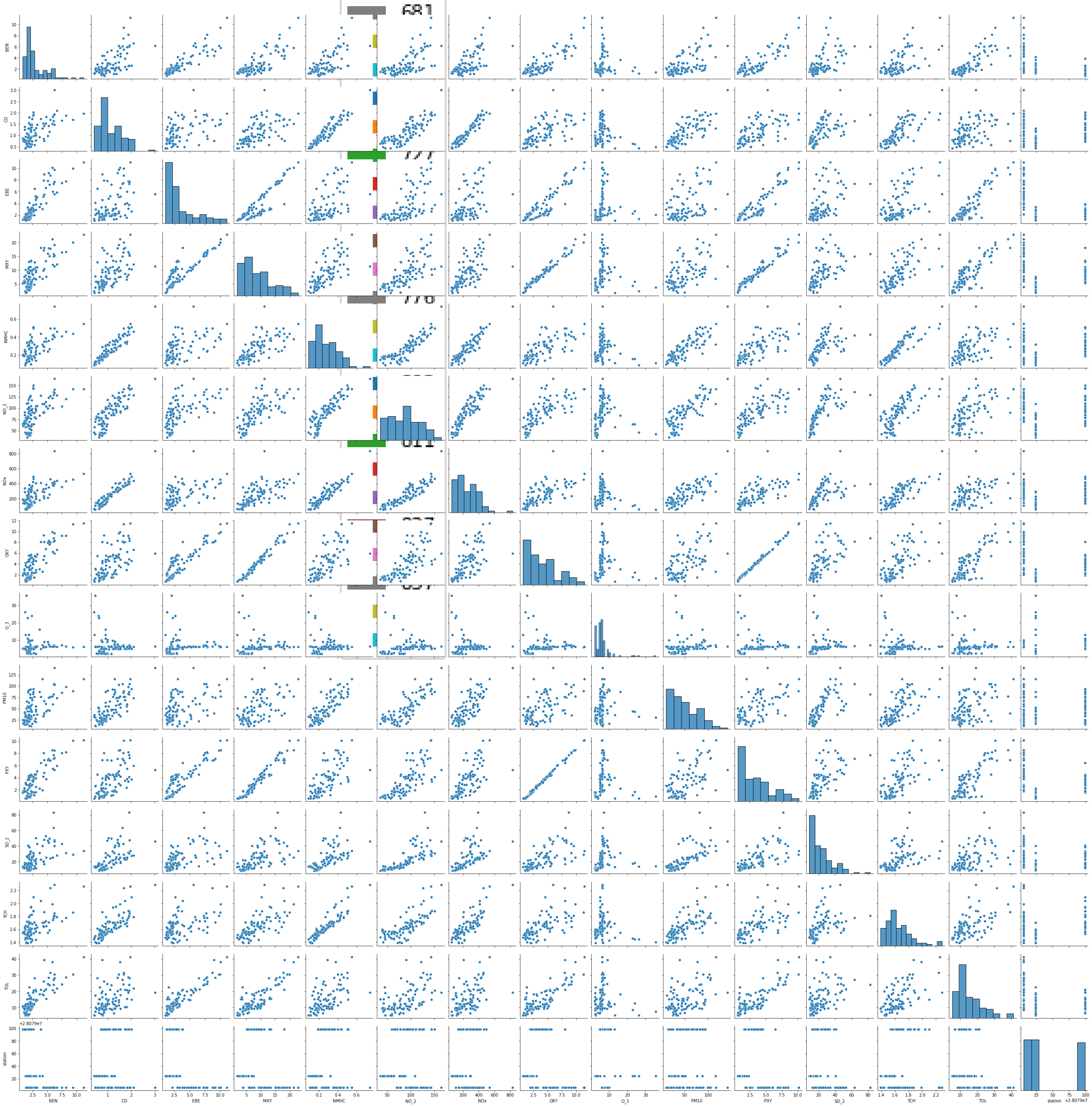
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
        'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [19]:

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

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x193376c9370>
```



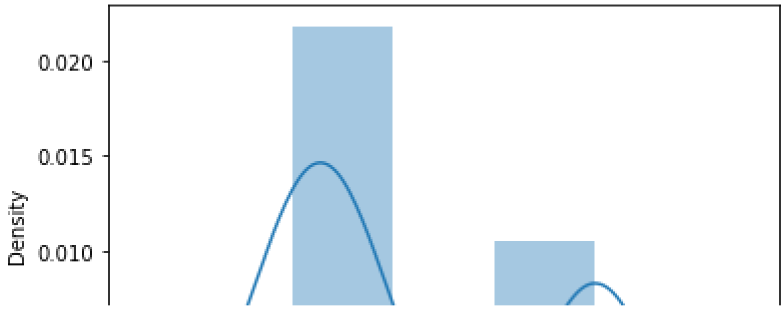
In [20]:

```
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'>

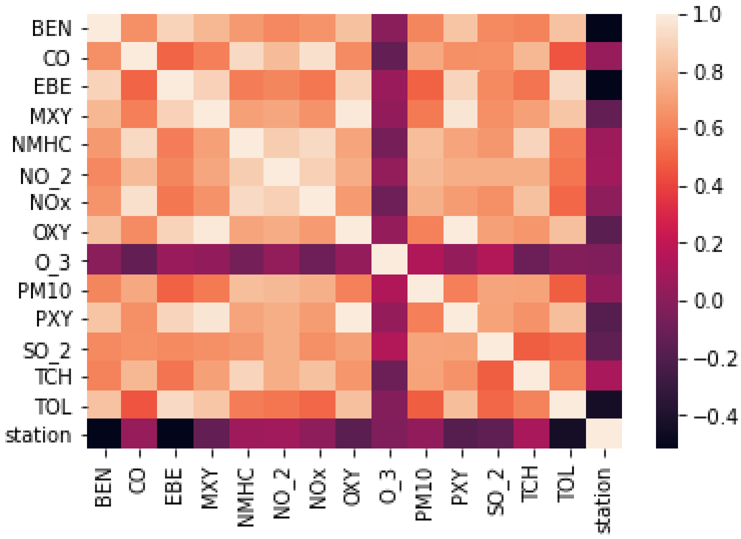


In [21]:

```
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]:

```
28079094.969658807
```

In [26]:

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

Out[26]:

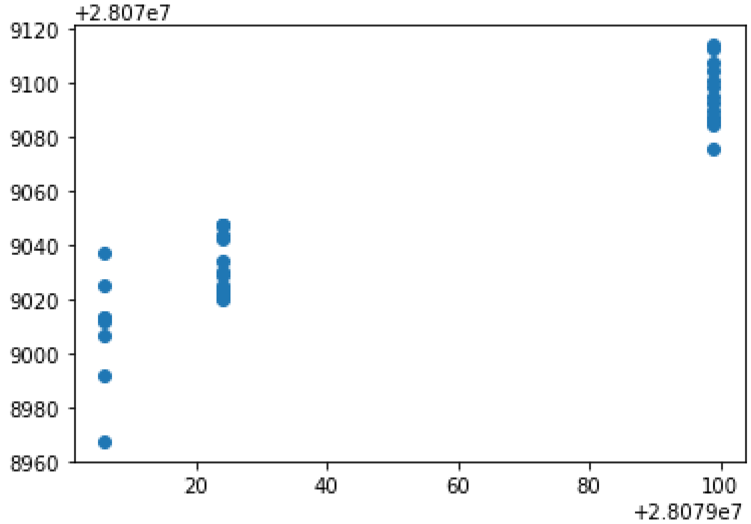
	Co-efficient
BEN	-10.748896
CO	-2.015224
EBE	-29.410909
MXY	14.833981
NMHC	199.300105
NO_2	-0.129263
NOx	-0.012314
OXY	-47.881637
O_3	0.416031
PM10	0.113235
PXY	49.165460
SO_2	-0.274870
TCH	-46.015310
TOL	1.182023

In [27]:

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

Out[27]:

```
<matplotlib.collections.PathCollection at 0x1934484cb50>
```



In [28]:

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

Out[28]:

```
0.8669452029017192
```

In [29]:

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

Out[29]:

```
0.9061843657440546
```

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]:

```
0.8867524775822146
```

In [33]:

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

Out[33]:

```
0.8664819679629829
```

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]:

```
0.6923145899784232
```

In [36]:

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

Out[36]:

```
0.5633408788029934
```

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([ -7.99133153,   0.        , -11.59480875,   6.14978637,
         0.        ,   0.52304156,   0.01210836,   1.34122031,
        -0.05790393,   0.20451887,   0.81326317,  -0.72421383,
         0.        ,  -1.85215883])
```

In [39]:

```
e.intercept_
```

Out[39]:

```
28079028.84873901
```

In [40]:

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

In [41]:

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

Out[41]:

```
0.7754908047419946
```

In [42]:

```
from sklearn import metrics
```

In [43]:

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

```
359.12930728220454
```

In [44]:

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

```
18.95070730295322
```

In [45]:

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

```
16.161460589085305
```

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]:

```
(114, 14)
```

In [49]:

```
target_vector.shape
```

Out[49]:

```
(114,)
```

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]:

```
0.9261448787367386
```

In [58]:

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

Out[58]:

```
array([[9.26144879e-01, 3.39405518e-16, 7.38551213e-02]])
```

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]:

```
0.8733974358974359
```

In [64]:

```
rfc_best=grid_search.best_estimator_
```

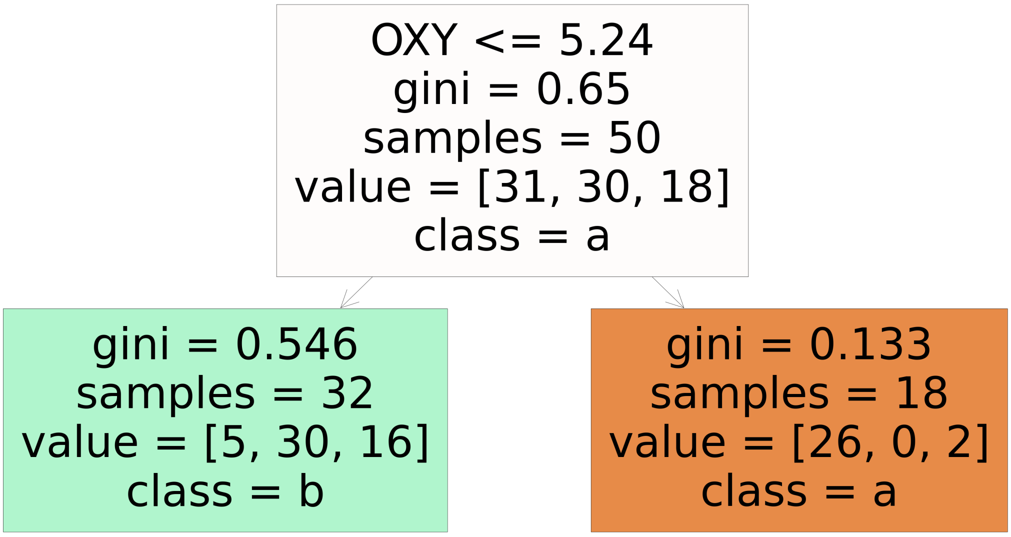
In [65]:

```
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, 'OXY <= 5.24\ngini = 0.65\nsamples = 50
\nvalue = [31, 30, 18]\nclass = a'),
 Text(1116.0, 543.5999999999999, 'gini = 0.546\nsamples = 32\nvalue = [5,
30, 16]\nclass = b'),
 Text(3348.0, 543.5999999999999, 'gini = 0.133\nsamples = 18\nvalue = [26,
0, 2]\nclass = a')]
```



random forest is best suitable for this dataset

In [ ]: