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_2009.csv")
df = df1.head(1000)
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2009-10-01 01:00:00	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002	NaN	50.680000	18.2
1	2009-10-01 01:00:00	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000	NaN	55.880001	10.5
2	2009-10-01 01:00:00	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001	NaN	49.060001	25.1
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.5
4	2009-10-01 01:00:00	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002	NaN	38.090000	23.7
...	...	...	...	...	...	...	...	...	...	...	
995	2009-10-02 16:00:00	0.79	0.81	1.08	2.42	0.17	16.629999	19.700001	0.92	133.899994	26.5
996	2009-10-02 16:00:00	NaN	0.34	NaN	NaN	NaN	63.080002	78.849998	NaN	77.169998	9.3
997	2009-10-02 16:00:00	0.34	NaN	0.18	NaN	0.27	102.500000	126.199997	NaN	53.279999	12.2
998	2009-10-02 16:00:00	1.76	NaN	1.15	NaN	0.19	82.989998	107.500000	NaN	NaN	21.8
999	2009-10-02 16:00:00	1.19	0.44	1.77	5.03	0.27	66.790001	89.110001	2.37	78.699997	14.6

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: 113 entries, 3 to 999
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     113 non-null    object 
 1   BEN      113 non-null    float64
 2   CO       113 non-null    float64
 3   EBE      113 non-null    float64
 4   MXY      113 non-null    float64
 5   NMHC     113 non-null    float64
 6   NO_2     113 non-null    float64
 7   NOx      113 non-null    float64
 8   OXY      113 non-null    float64
 9   O_3      113 non-null    float64
 10  PM10     113 non-null    float64
 11  PM25     113 non-null    float64
 12  PXY      113 non-null    float64
 13  SO_2     113 non-null    float64
 14  TCH      113 non-null    float64
 15  TOL      113 non-null    float64
 16  station  113 non-null    int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 15.9+ KB
```

In [6]:

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

Out[6]:

	CO	station
3	0.33	28079006
20	0.32	28079024
24	0.24	28079099
28	0.21	28079006
45	0.30	28079024
...	...	...
953	0.97	28079006
974	0.55	28079099
978	0.77	28079006
995	0.81	28079024
999	0.44	28079099

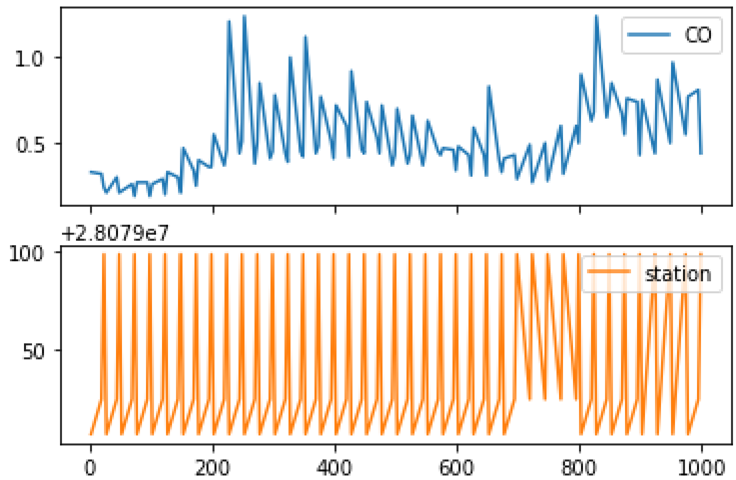
113 rows × 2 columns

In [7]:

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

Out[7]:

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

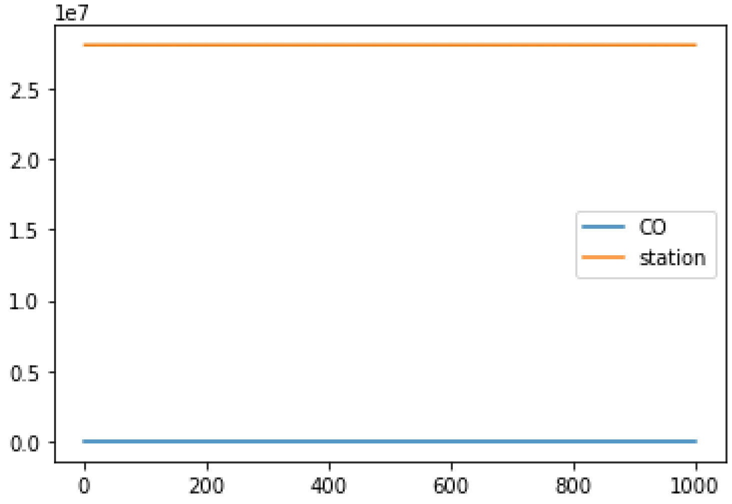


In [8]:

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

Out[8]:

<AxesSubplot:>



In [9]:

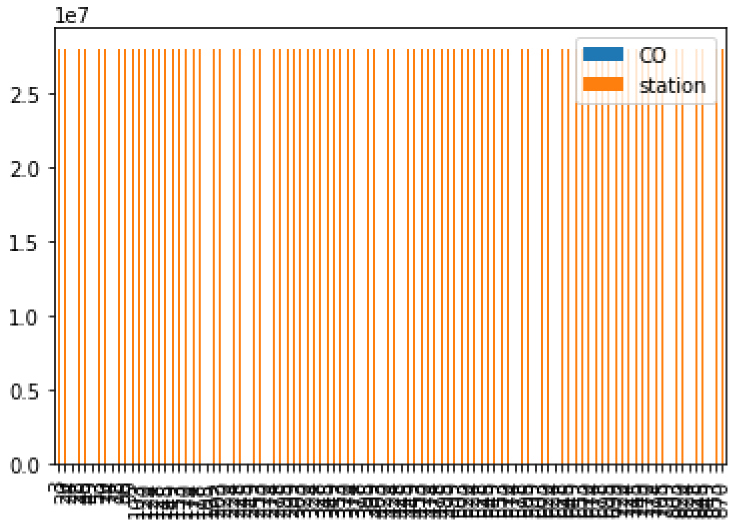
```
x = data[0:100]
```

In [10]:

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

Out[10]:

<AxesSubplot:>

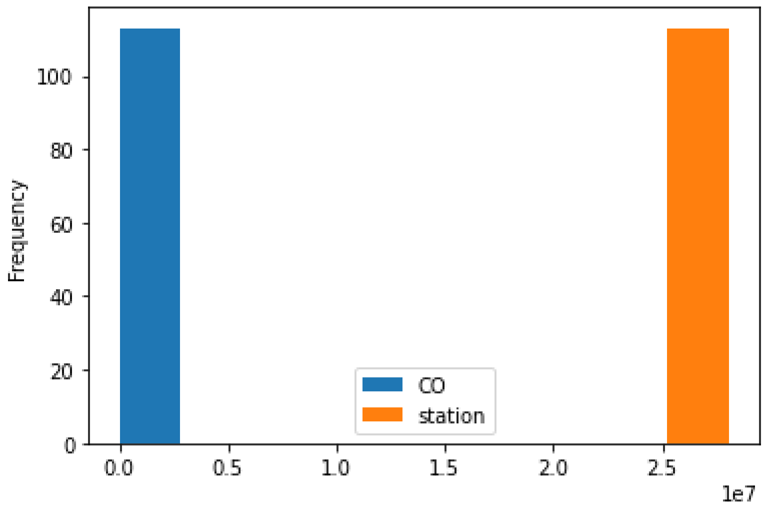


In [11]:

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

Out[11]:

<AxesSubplot:ylabel='Frequency'>

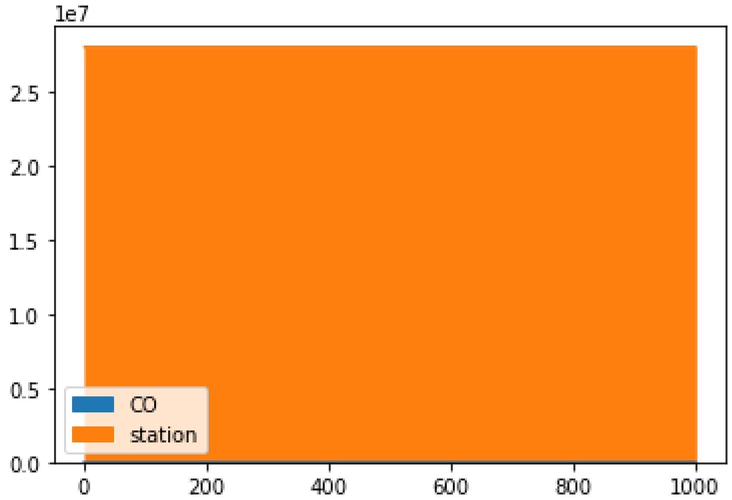


In [12]:

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

Out[12]:

<AxesSubplot:>

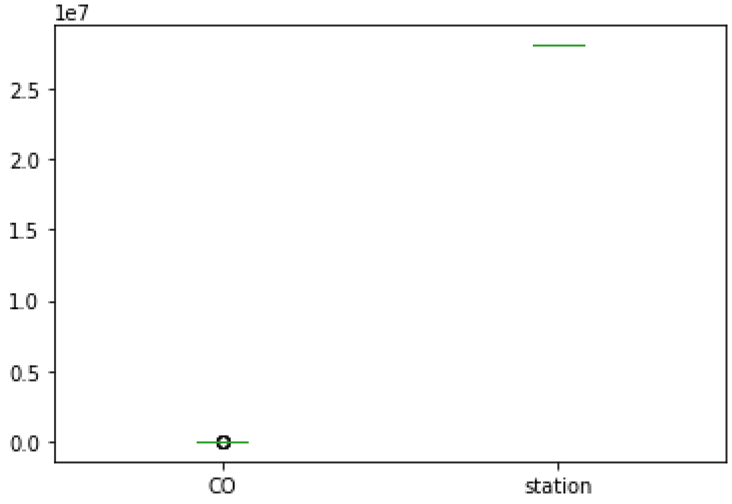


In [13]:

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

Out[13]:

<AxesSubplot:>



In [14]:

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

Out[14]:

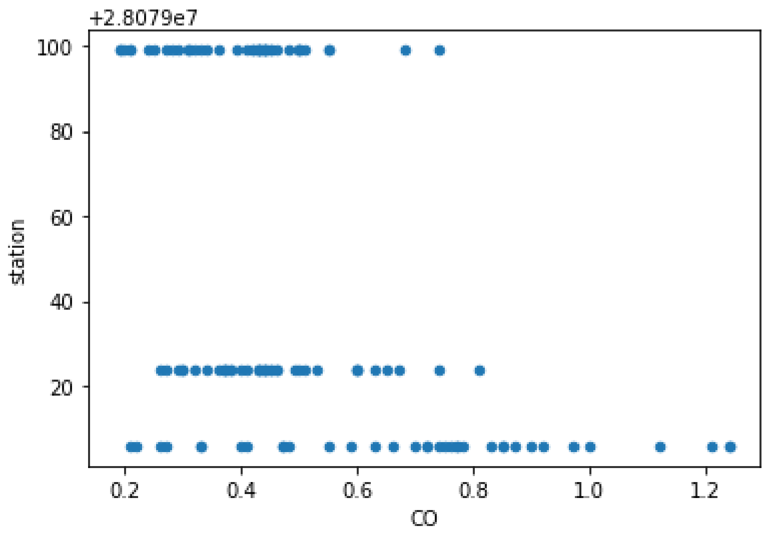
```
<AxesSubplot:ylabel='station'>
```

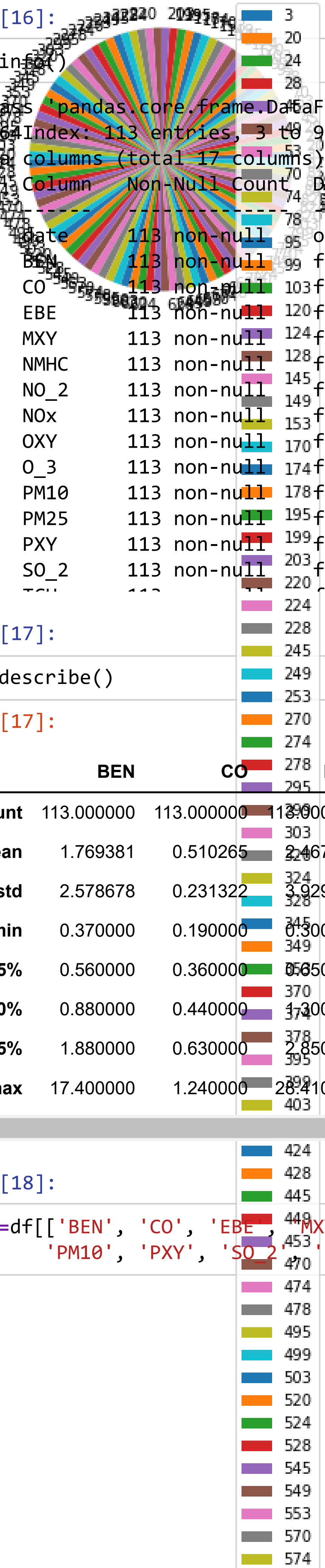
In [15]:

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

Out[15]:

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





In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 113 entries, 3 to 999
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     113 non-null    object 
 1   BEN      113 non-null    float64
 2   CO       113 non-null    float64
 3   EBE      113 non-null    float64
 4   MXY      113 non-null    float64
 5   NMHC     113 non-null    float64
 6   NO_2     113 non-null    float64
 7   NOx      113 non-null    float64
 8   OXY      113 non-null    float64
 9   O_3      113 non-null    float64
 10  PM10     113 non-null    float64
 11  PM25     113 non-null    float64
 12  PXY      113 non-null    float64
 13  SO_2     113 non-null    float64
```

In [17]:

```
df.describe()
```

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx
count	113.000000	113.000000	113.000000	113.000000	113.000000	113.000000	113.000000
mean	1.769381	0.510265	2.467876	5.415752	0.370177	63.396725	110.062832
std	2.578678	0.231322	3.929351	8.479730	0.326680	33.438821	74.424003
min	0.370000	0.190000	0.300000	0.800000	0.000000	9.590000	17.299999
25%	0.560000	0.360000	0.650000	1.490000	0.200000	33.869999	42.320000
50%	0.880000	0.440000	1.300000	3.080000	0.310000	63.830002	91.139999
75%	1.880000	0.630000	2.850000	5.370000	0.400000	84.279999	163.899994
max	17.400000	1.240000	28.410000	56.500000	2.110000	139.699997	321.000000

In [18]:

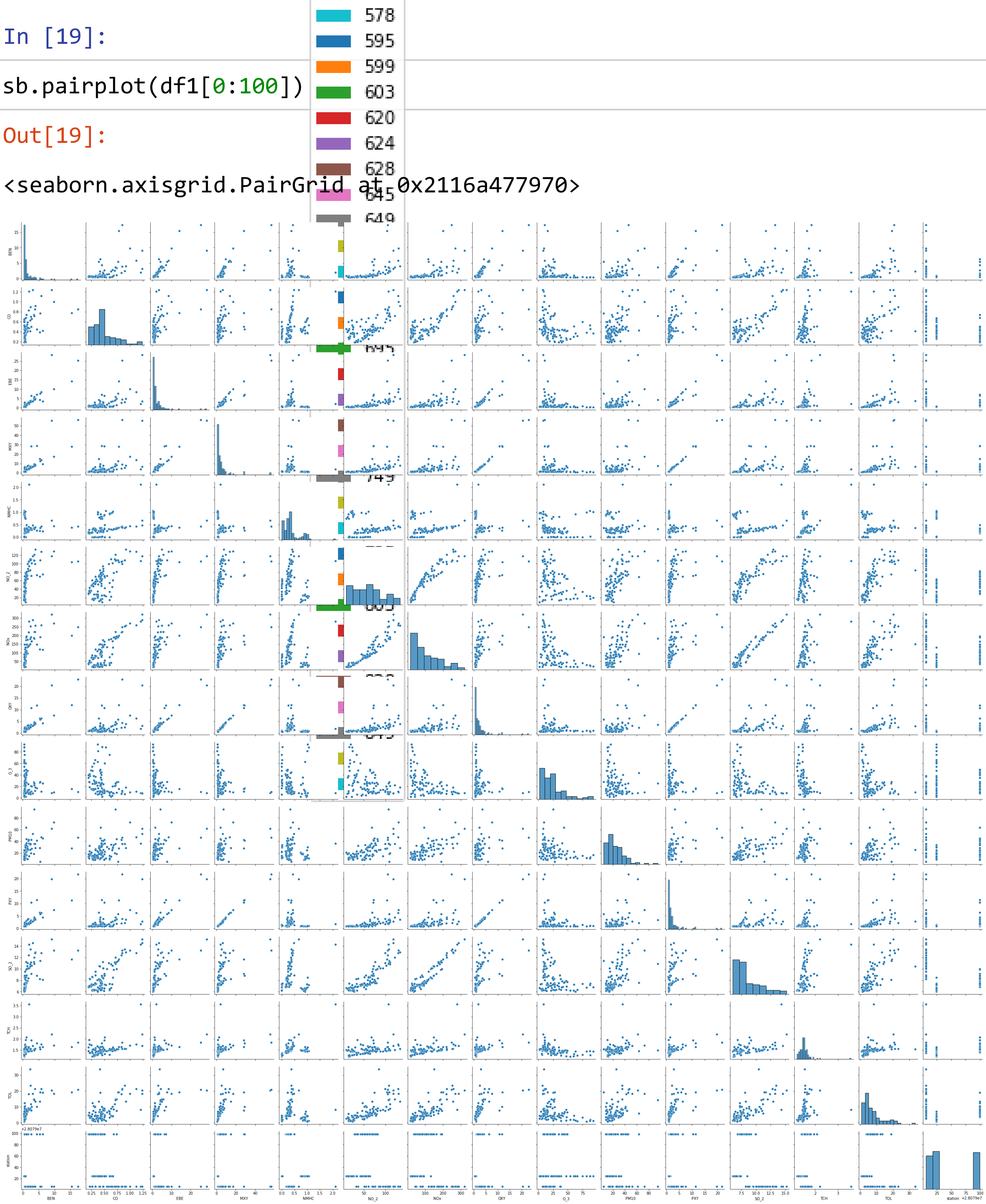
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [19]:

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

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x2116a477970>
```



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]:

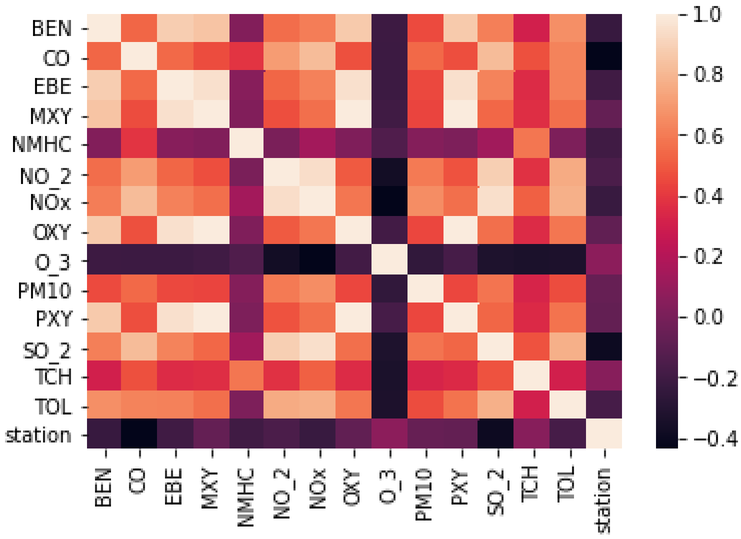
```
<AxesSubplot:xlabel='EBE', ylabel='Density'>
```

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]:

```
28079100.22381208
```

In [26]:

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

Out[26]:

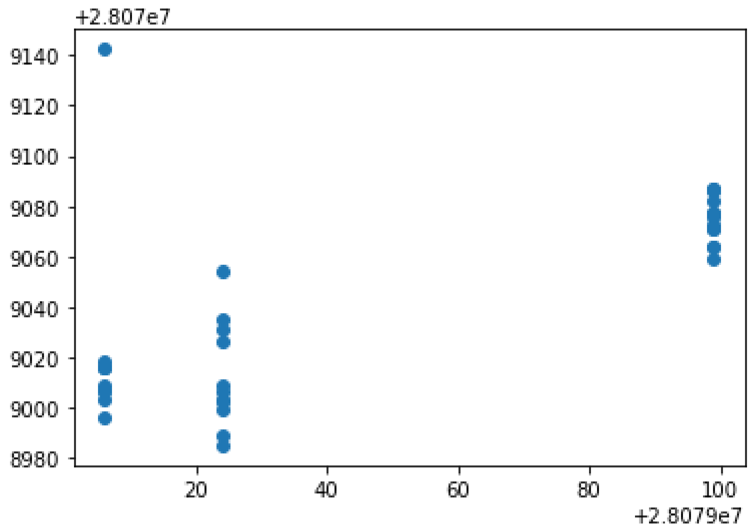
	Co-efficient
BEN	-1.360274
CO	-64.051480
EBE	-1.487121
MXY	1.341950
NMHC	-39.306053
NO_2	-0.094485
NOx	1.061408
OXY	28.606483
O_3	0.590976
PM10	0.029720
PXY	-33.186388
SO_2	-38.303033
TCH	125.718647
TOL	1.625493

In [27]:

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

Out[27]:

```
<matplotlib.collections.PathCollection at 0x211777996d0>
```



In [28]:

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

Out[28]:

```
0.41733633573776896
```

In [29]:

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

Out[29]:

```
0.6794802146180634
```

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]:

```
0.6397555824729599
```

In [33]:

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

Out[33]:

```
0.45989643171984795
```

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]:

```
0.20741891304373317
```

In [36]:

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

Out[36]:

```
0.16594174865040112
```

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([-3.26726666e+00, -1.99699639e+00, -5.41909317e+00,  3.00910204e+00,
       -9.33107644e-01,  3.34249634e-01,  1.03105750e-02,  1.16231361e+00,
       -1.16328459e-01,  2.56858178e-01,  0.00000000e+00, -1.15914991e+01,
        2.87191744e+00,  4.67234987e-01])
```

In [39]:

```
e.intercept_
```

Out[39]:

```
28079111.444218345
```

In [40]:

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

In [41]:

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

Out[41]:

```
0.45139736955364995
```

In [42]:

```
from sklearn import metrics
```

In [43]:

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

```
932.2376974129353
```

In [44]:

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

```
30.53256781557908
```

In [45]:

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

```
27.398654631065096
```

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]:

```
(113, 14)
```

In [49]:

```
target_vector.shape
```

Out[49]:

```
(113,)
```

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]:

```
0.9823008849557522
```

In [57]:

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

Out[57]:

```
0.9967096994596093
```

In [58]:

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

Out[58]:

```
array([[9.96709699e-01, 1.45852666e-40, 3.29030054e-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]:

```
0.8615384615384616
```

In [64]:

```
rfc_best=grid_search.best_estimator_
```

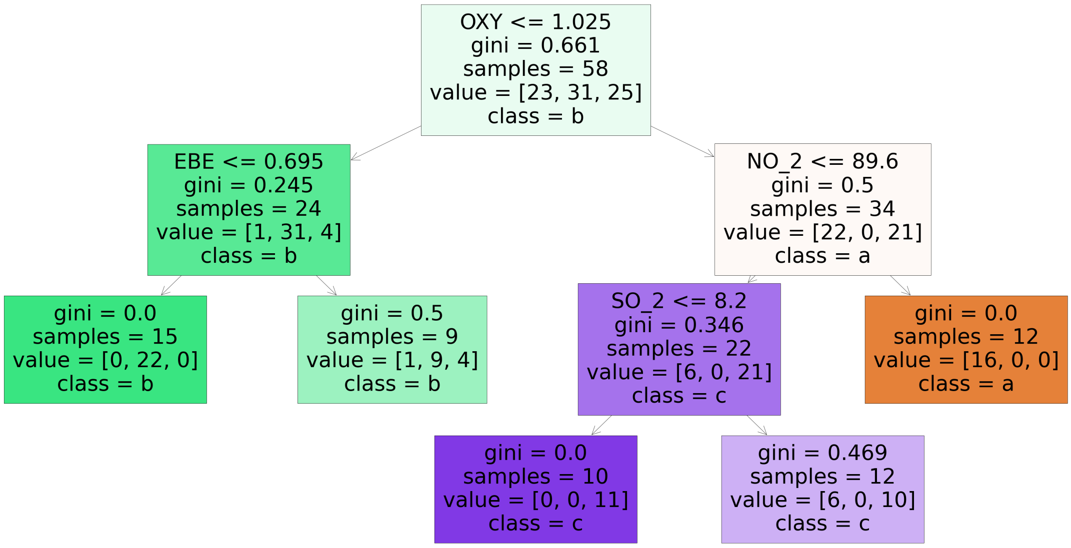
In [65]:

```
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, 'OXY <= 1.025\ngini = 0.661\nsamples = 5
8\nvalue = [23, 31, 25]\nclass = b'),
 Text(1116.0, 1359.0, 'EBE <= 0.695\ngini = 0.245\nsamples = 24\nvalue =
[1, 31, 4]\nclass = b'),
 Text(558.0, 815.4000000000001, 'gini = 0.0\nsamples = 15\nvalue = [0, 22,
0]\nclass = b'),
 Text(1674.0, 815.4000000000001, 'gini = 0.5\nsamples = 9\nvalue = [1, 9,
4]\nclass = b'),
 Text(3348.0, 1359.0, 'NO_2 <= 89.6\ngini = 0.5\nsamples = 34\nvalue = [2
2, 0, 21]\nclass = a'),
 Text(2790.0, 815.4000000000001, 'SO_2 <= 8.2\ngini = 0.346\nsamples = 22
\nvalue = [6, 0, 21]\nclass = c'),
 Text(2232.0, 271.79999999999995, 'gini = 0.0\nsamples = 10\nvalue = [0,
0, 11]\nclass = c'),
 Text(3348.0, 271.79999999999995, 'gini = 0.469\nsamples = 12\nvalue = [6,
0, 10]\nclass = c'),
 Text(3906.0, 815.4000000000001, 'gini = 0.0\nsamples = 12\nvalue = [16,
0, 0]\nclass = a')]
```



logistic regression is best suitable for this dataset

In [ ]: