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_2011.csv")
df = df1.head(1000)
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL
0	2011-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	84.0	NaN	NaN	NaN	6.0	NaN	NaN
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7
2	2011-11-01 01:00:00	2.9	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN	7.2
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN	NaN
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	2011-11-02 18:00:00	NaN	0.3	NaN	NaN	12.0	50.0	23.0	NaN	NaN	NaN	NaN	NaN
996	2011-11-02 18:00:00	NaN	NaN	NaN	NaN	6.0	31.0	NaN	37.0	NaN	1.0	NaN	NaN
997	2011-11-02 18:00:00	NaN	NaN	NaN	NaN	2.0	45.0	NaN	37.0	16.0	NaN	NaN	NaN
998	2011-11-02 18:00:00	NaN	NaN	NaN	NaN	10.0	49.0	NaN	40.0	13.0	NaN	NaN	NaN
999	2011-11-02 18:00:00	NaN	NaN	NaN	NaN	2.0	36.0	32.0	NaN	NaN	NaN	NaN	NaN

1000 rows × 14 columns

In [3]:

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

In [4]:

```
df.columns
```

Out[4]:

```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'P
M25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 84 entries, 1 to 990
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     84 non-null     object 
 1   BEN      84 non-null     float64
 2   CO       84 non-null     float64
 3   EBE      84 non-null     float64
 4   NMHC     84 non-null     float64
 5   NO       84 non-null     float64
 6   NO_2     84 non-null     float64
 7   O_3      84 non-null     float64
 8   PM10     84 non-null     float64
 9   PM25     84 non-null     float64
 10  SO_2     84 non-null     float64
 11  TCH      84 non-null     float64
 12  TOL      84 non-null     float64
 13  station  84 non-null     int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 9.8+ KB
```

In [6]:

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

Out[6]:

	CO	station
1	0.4	28079008
6	0.3	28079024
25	0.3	28079008
30	0.4	28079024
49	0.2	28079008
...	...	...
942	0.2	28079024
961	0.2	28079008
966	0.2	28079024
985	0.3	28079008
990	0.2	28079024

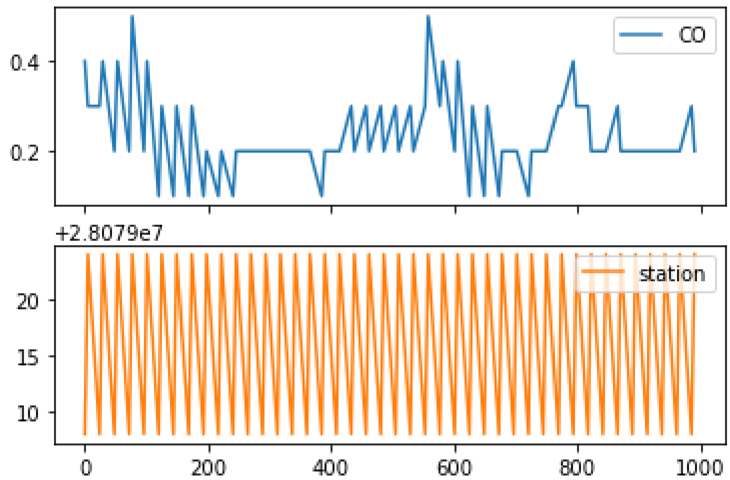
84 rows × 2 columns

In [7]:

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

Out[7]:

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

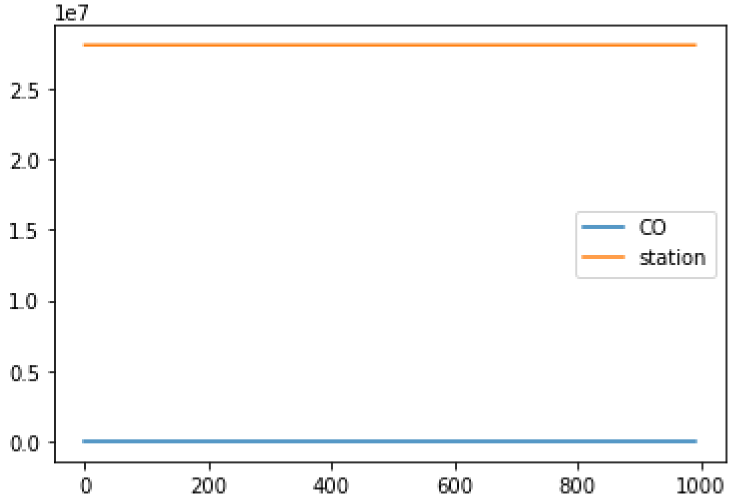


In [8]:

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

Out[8]:

<AxesSubplot:>



In [9]:

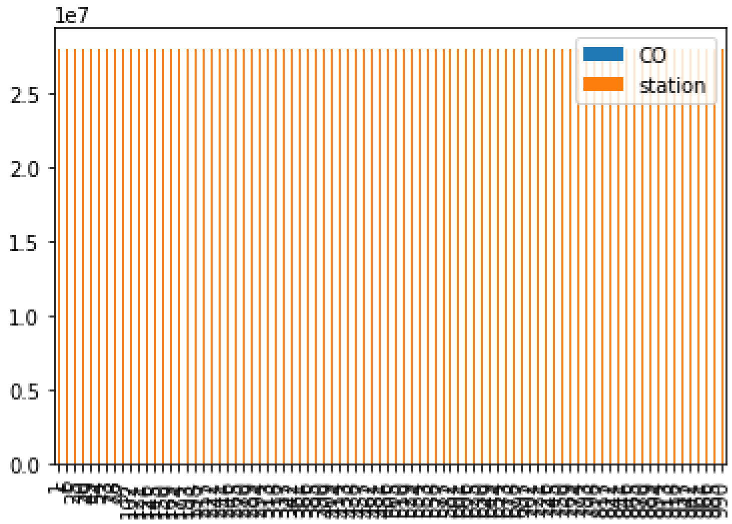
```
x = data[0:100]
```

In [10]:

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

Out[10]:

<AxesSubplot:>

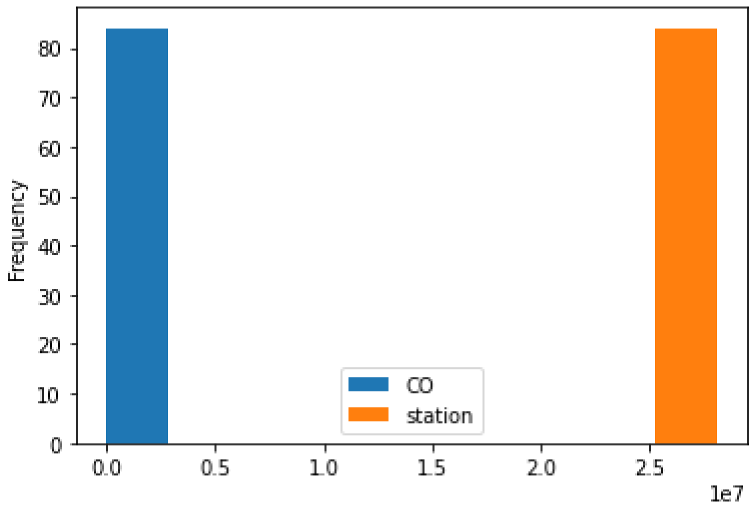


In [11]:

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

Out[11]:

<AxesSubplot:ylabel='Frequency'>

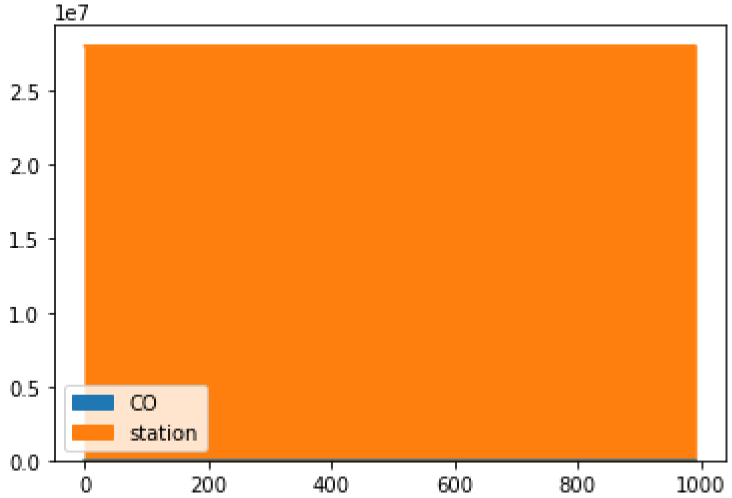


In [12]:

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

Out[12]:

<AxesSubplot:>

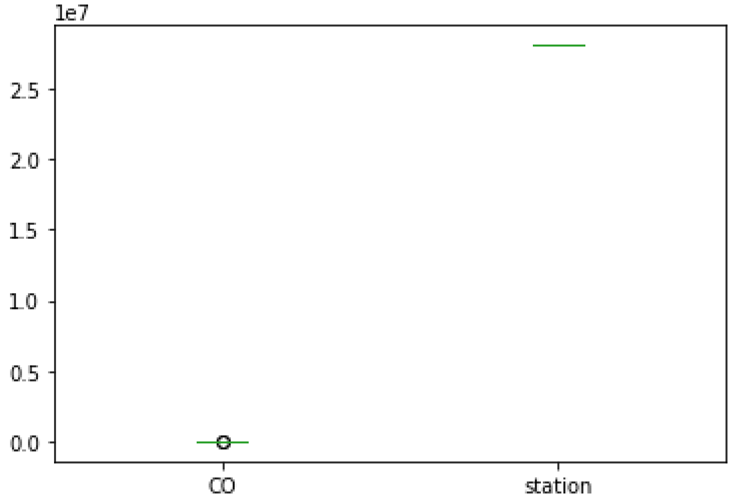


In [13]:

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

Out[13]:

<AxesSubplot:>



In [14]:

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

Out[14]:

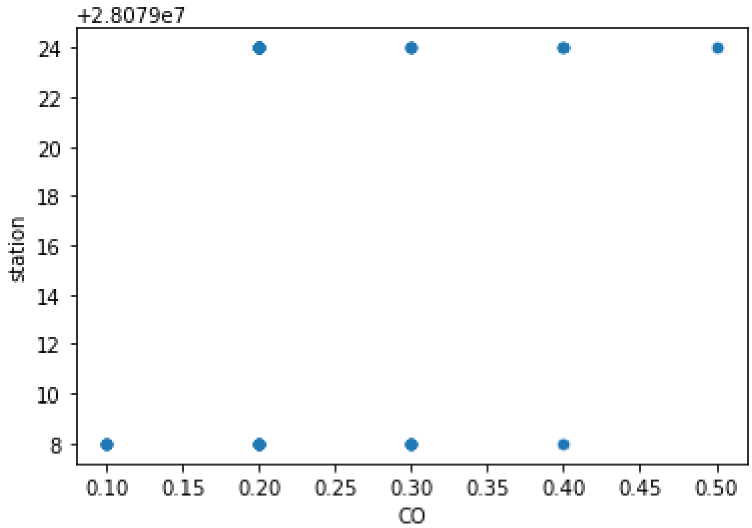
```
<AxesSubplot:ylabel='station'>
```

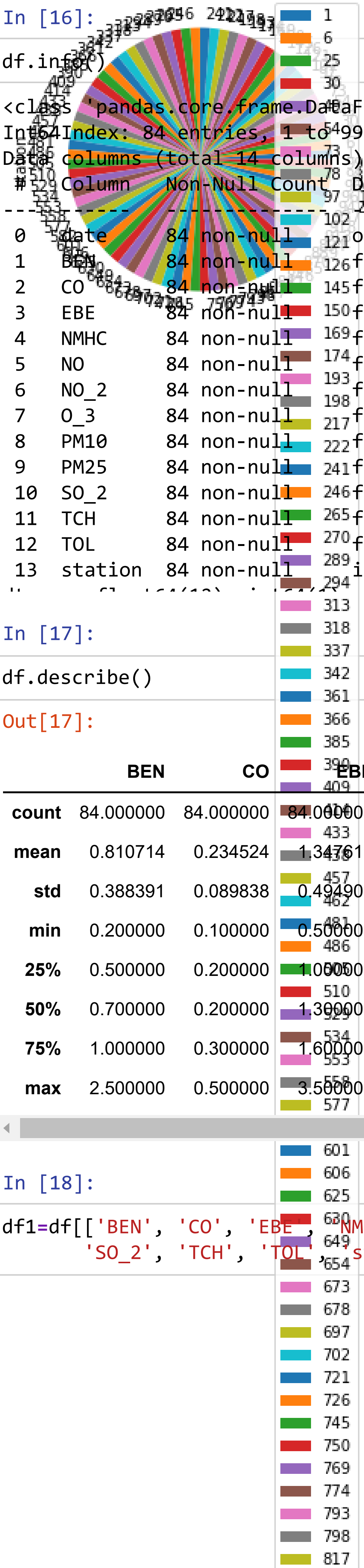
In [15]:

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

Out[15]:

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





In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 84 entries, 1 to 990
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     84 non-null     object
 1   BEN      84 non-null     float64
 2   CO       84 non-null     float64
 3   EBE      84 non-null     float64
 4   NMHC     84 non-null     float64
 5   NO       84 non-null     float64
 6   NO_2     84 non-null     float64
 7   O_3      84 non-null     float64
 8   PM10     84 non-null     float64
 9   PM25     84 non-null     float64
 10  SO_2     84 non-null     float64
 11  TCH      84 non-null     float64
 12  TOL      84 non-null     float64
 13  station  84 non-null     int64
```

In [17]:

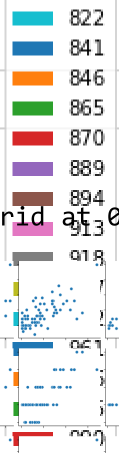
```
df.describe()
```

Out[17]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM1
count	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.00000
mean	0.810714	0.234524	1.347619	0.150476	16.178571	42.107143	22.404762	25.88095
std	0.388391	0.089838	0.494908	0.043490	14.639574	19.695280	16.990514	6.95147
min	0.200000	0.100000	0.500000	0.090000	1.000000	2.000000	3.000000	9.00000
25%	0.500000	0.200000	1.000000	0.120000	3.750000	32.000000	6.750000	22.00000
50%	0.700000	0.200000	1.300000	0.150000	13.500000	41.000000	20.500000	25.00000
75%	1.000000	0.300000	1.600000	0.170000	23.250000	55.250000	37.250000	30.00000
max	2.500000	0.500000	3.500000	0.260000	68.000000	92.000000	60.000000	41.00000

In [18]:

```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
        'SO_2', 'TCH', 'TOL', 'station']]
```

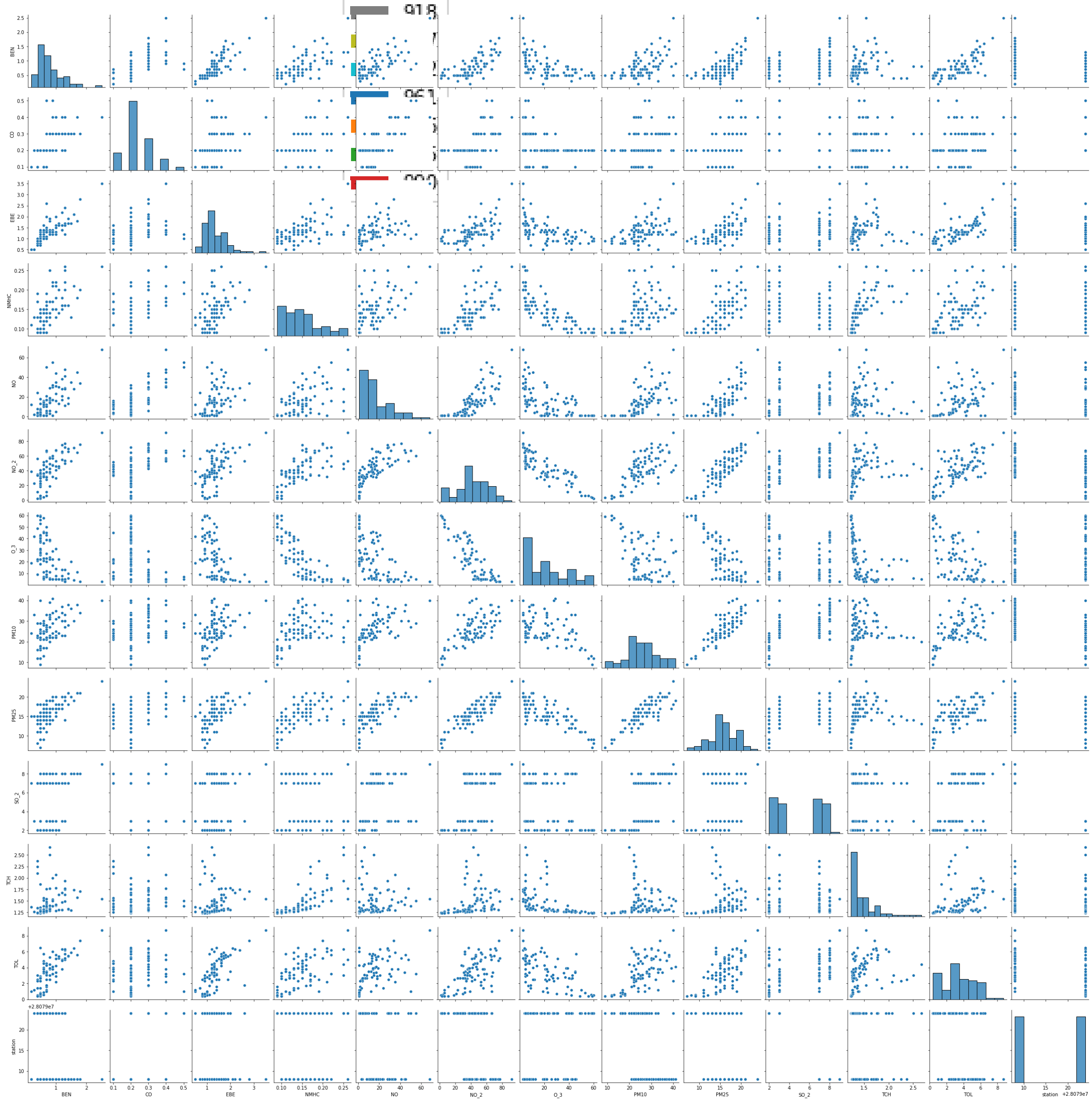


In [19]:

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

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x14d7323df40>
```



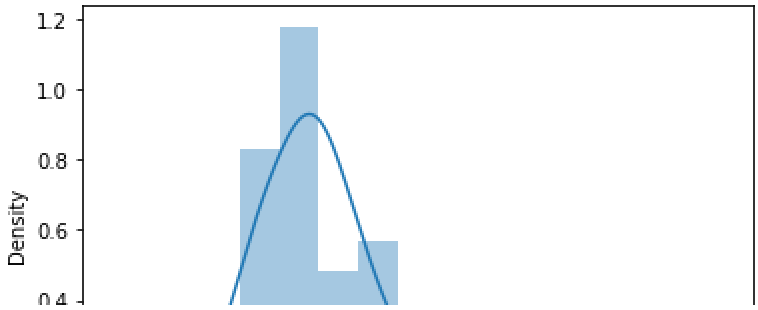
In [20]:

```
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'>
```

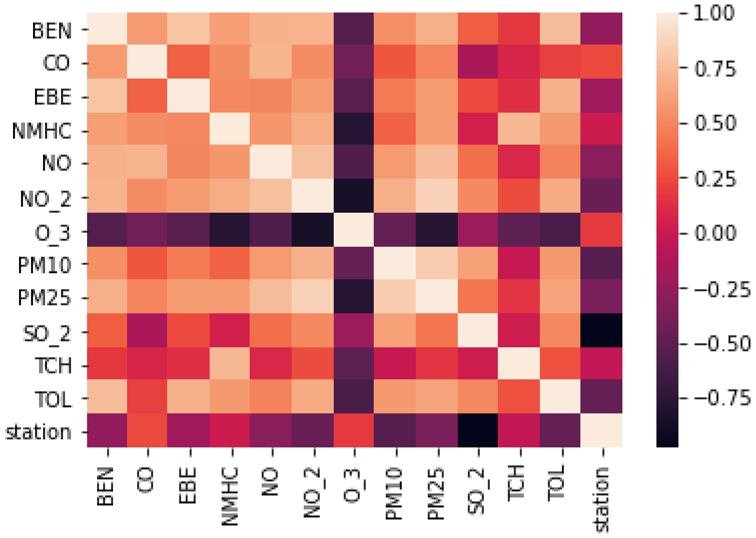


In [21]:

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

Out[21]:

```
<AxesSubplot:>
```



In [22]:

```
x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
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]:

```
7.450580596923828e-09
```

In [26]:

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

Out[26]:

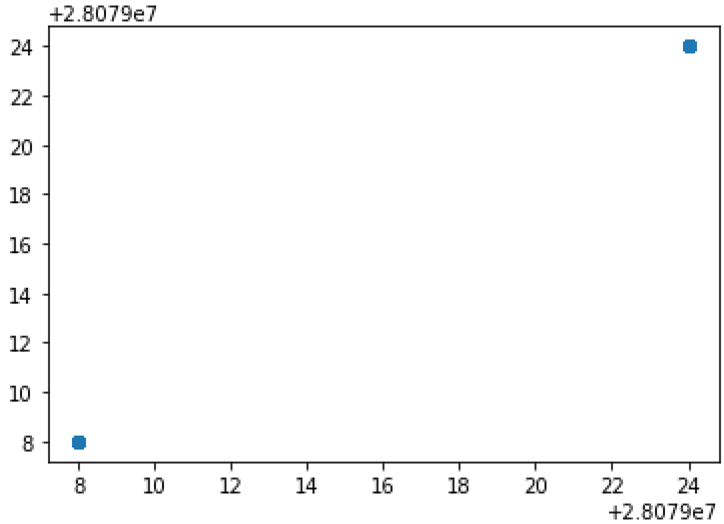
	Co-efficient
BEN	1.073708e-14
CO	2.564059e-14
EBE	1.953164e-15
NMHC	-9.599656e-14
NO	1.226602e-15
NO_2	1.517691e-15
O_3	-1.418615e-15
PM10	4.730335e-16
PM25	-6.585922e-16
SO_2	5.417656e-16
TCH	-1.837061e-15
TOL	-1.835073e-15
station	1.000000e+00

In [27]:

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

Out[27]:

```
<matplotlib.collections.PathCollection at 0x14d7d10d280>
```



In [28]:

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

Out[28]:

```
1.0
```

In [29]:

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

Out[29]:

```
1.0
```

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]:

```
0.9999230161399446
```

In [33]:

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

Out[33]:

```
0.9999523651403391
```

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]:

```
0.975527774182802
```

In [36]:

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

Out[36]:

```
0.9752222350920267
```

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.        , -0.        ,  0.        , -0.        ,
       -0.00134843,  0.        , -0.        , -0.        , -0.        ,
        0.        , -0.        ,  0.9832613 ])
```

In [39]:

```
e.intercept_
```

Out[39]:

```
470006.23367472365
```

In [40]:

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

In [41]:

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

Out[41]:

```
0.9997691923949762
```

In [42]:

```
from sklearn import metrics
```

In [43]:

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

```
0.01468428029121885
```

In [44]:

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

```
0.12117871220317061
```

In [45]:

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

```
0.11872569958751018
```

In [46]:

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

In [47]:

```
feature_matrix=df[['BEN']]
target_vector=df[ 'station']
```

In [48]:

```
feature_matrix.shape
```

Out[48]:

```
(84, 1)
```

In [49]:

```
target_vector.shape
```

Out[49]:

```
(84,)
```

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 [55]:

```
observation=[[1]]
```

In [56]:

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

```
[28079008]
```

In [57]:

```
logr.classes_
```

Out[57]:

```
array([28079008, 28079024], dtype=int64)
```

In [58]:

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

Out[58]:

```
0.5833333333333334
```

In [59]:

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

Out[59]:

```
0.6370043676371444
```

In [60]:

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

Out[60]:

```
array([[0.63700437, 0.36299563]])
```

In [61]:

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

In [62]:

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

Out[62]:

```
RandomForestClassifier()
```

In [63]:

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

In [64]:

```
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[64]:

```
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 [65]:

```
grid_search.best_score_
```

Out[65]:

```
1.0
```

In [66]:

```
rfc_best=grid_search.best_estimator_
```

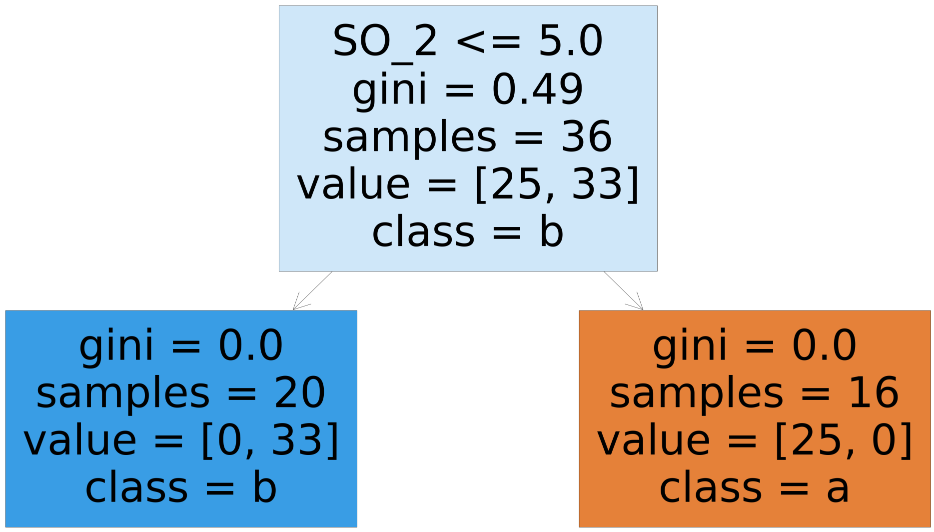
In [67]:

```
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[67]:

```
[Text(2232.0, 1630.8000000000002, 'SO_2 <= 5.0\ngini = 0.49\nsamples = 36
\nvalue = [25, 33]\nclass = b'),
 Text(1116.0, 543.5999999999999, 'gini = 0.0\nsamples = 20\nvalue = [0, 3
3]\nclass = b'),
 Text(3348.0, 543.5999999999999, 'gini = 0.0\nsamples = 16\nvalue = [25,
0]\nclass = a')]
```



logisticregression is best suitable for this dataset

In [ ]: