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_2014.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	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	3.0	10.0	NaN	NaN	NaN	3.0	NaN	NaN	2
1	2014-06-01 01:00:00	0.2	0.2	0.1	0.11	3.0	17.0	68.0	10.0	5.0	5.0	1.36	1.3	2
2	2014-06-01 01:00:00	0.3	NaN	0.1	NaN	2.0	6.0	NaN	NaN	NaN	NaN	NaN	1.1	2
3	2014-06-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	79.0	NaN	NaN	NaN	NaN	NaN	2
4	2014-06-01 01:00:00	NaN	NaN	NaN	NaN	1.0	6.0	75.0	NaN	NaN	4.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	2014-06-02 18:00:00	NaN	0.2	NaN	NaN	6.0	26.0	125.0	NaN	NaN	NaN	NaN	NaN	2
996	2014-06-02 18:00:00	NaN	NaN	NaN	NaN	7.0	36.0	NaN	17.0	NaN	4.0	NaN	NaN	2
997	2014-06-02 18:00:00	NaN	NaN	NaN	NaN	2.0	16.0	NaN	14.0	8.0	NaN	NaN	NaN	2
998	2014-06-02 18:00:00	NaN	NaN	NaN	NaN	5.0	31.0	NaN	14.0	7.0	NaN	NaN	NaN	2
999	2014-06-02 18:00:00	NaN	NaN	NaN	NaN	3.0	15.0	125.0	NaN	NaN	NaN	NaN	NaN	2

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: 82 entries, 1 to 990
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     82 non-null     object 
 1   BEN      82 non-null     float64
 2   CO       82 non-null     float64
 3   EBE      82 non-null     float64
 4   NMHC     82 non-null     float64
 5   NO       82 non-null     float64
 6   NO_2     82 non-null     float64
 7   O_3      82 non-null     float64
 8   PM10     82 non-null     float64
 9   PM25     82 non-null     float64
 10  SO_2     82 non-null     float64
 11  TCH      82 non-null     float64
 12  TOL      82 non-null     float64
 13  station  82 non-null     int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 9.6+ KB
```

In [6]:

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

Out[6]:

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

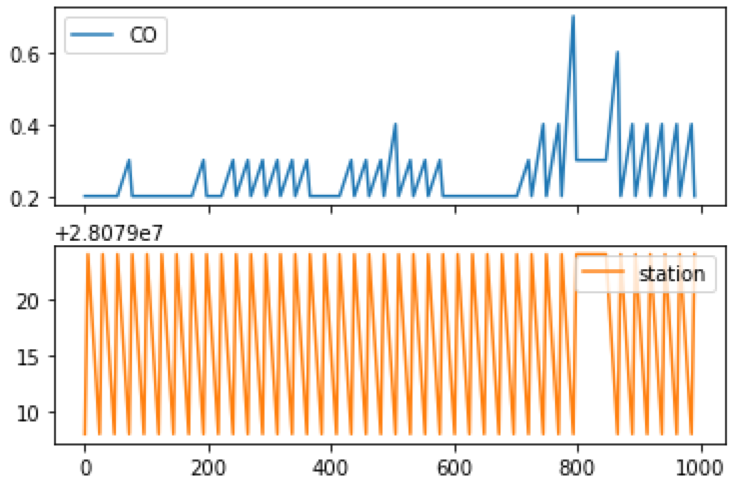
82 rows × 2 columns

In [7]:

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

Out[7]:

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

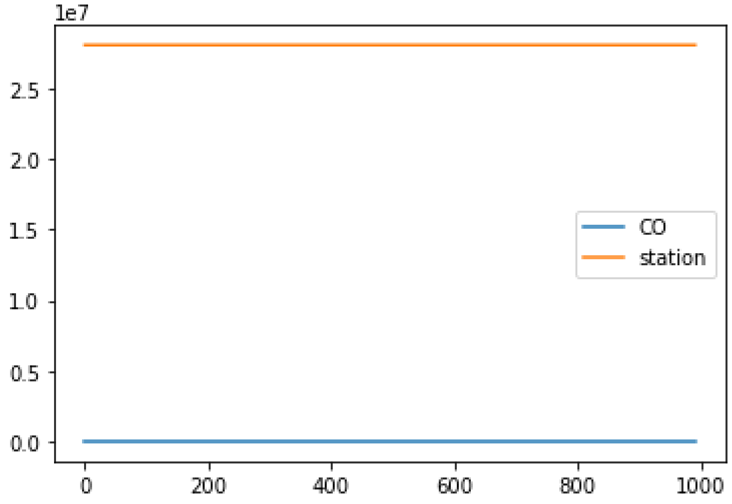


In [8]:

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

Out[8]:

<AxesSubplot:>



In [9]:

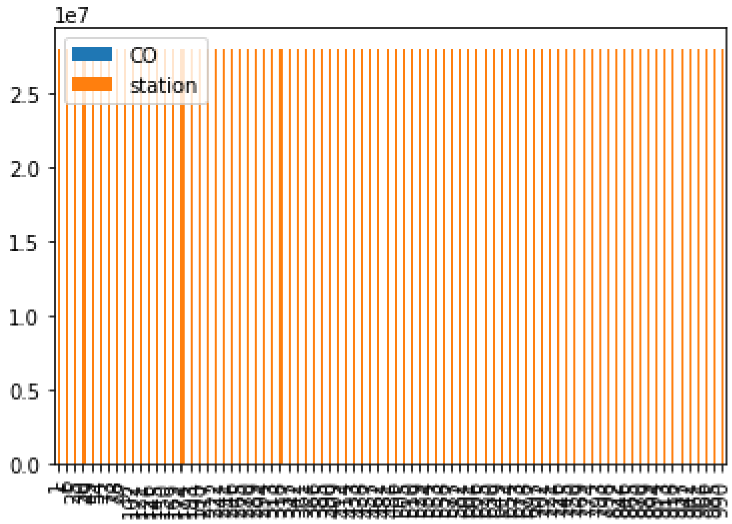
```
x = data[0:100]
```

In [10]:

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

Out[10]:

<AxesSubplot:>

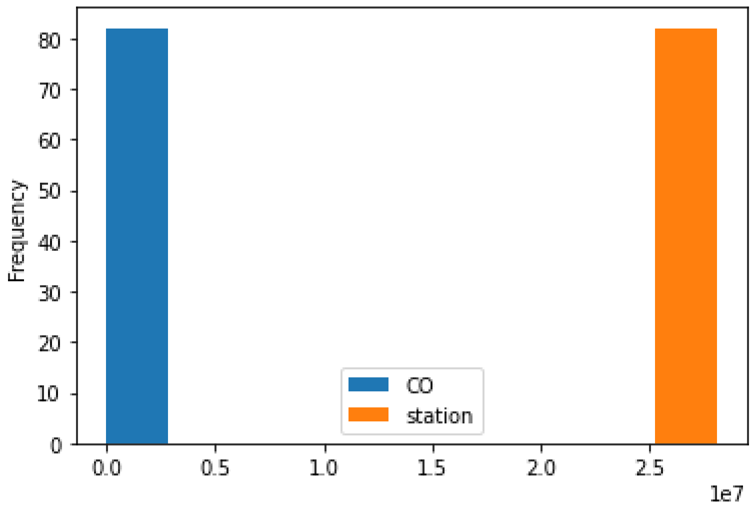


In [11]:

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

Out[11]:

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

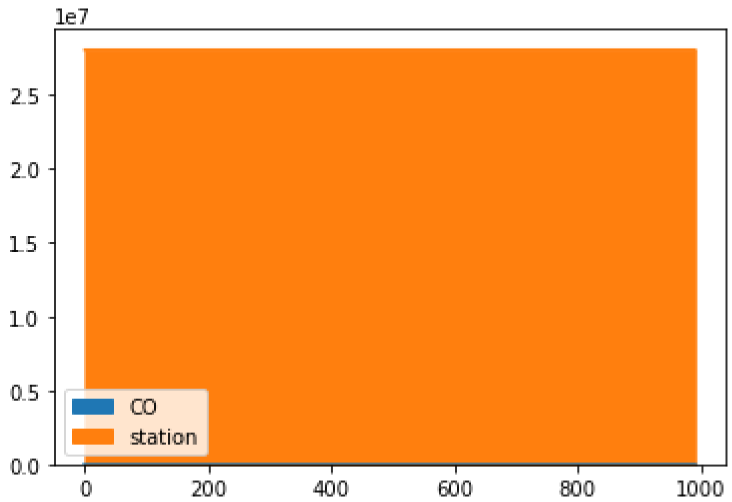


In [12]:

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

Out[12]:

```
<AxesSubplot:>
```

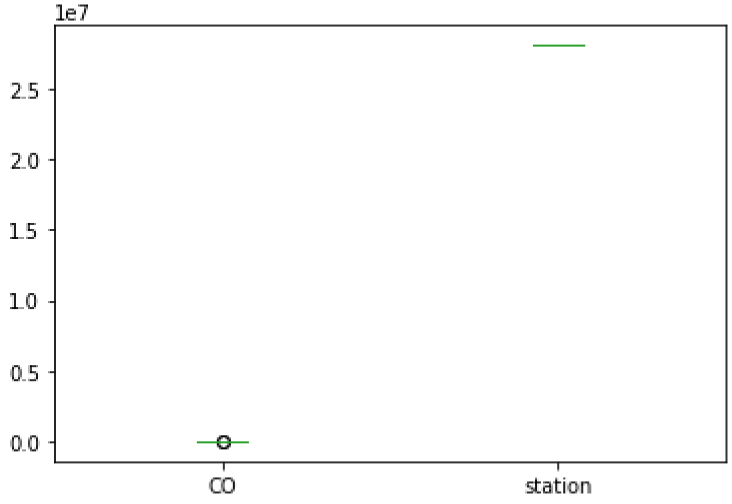


In [13]:

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

Out[13]:

<AxesSubplot:>



In [14]:

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

Out[14]:

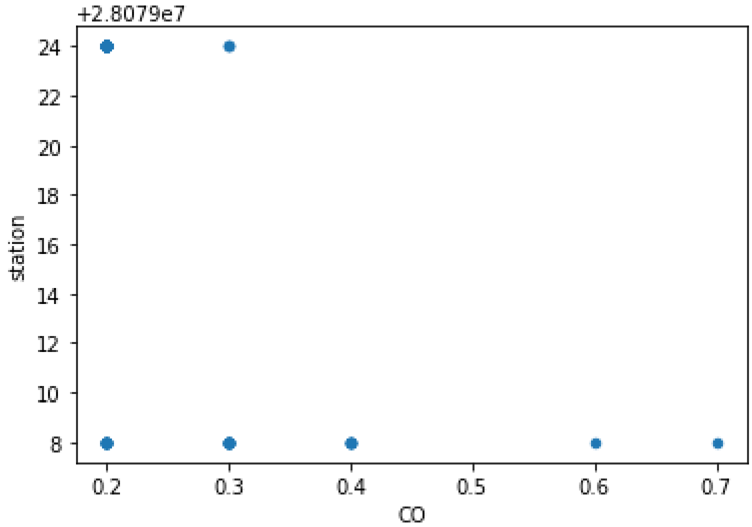
```
<AxesSubplot:ylabel='station'>
```

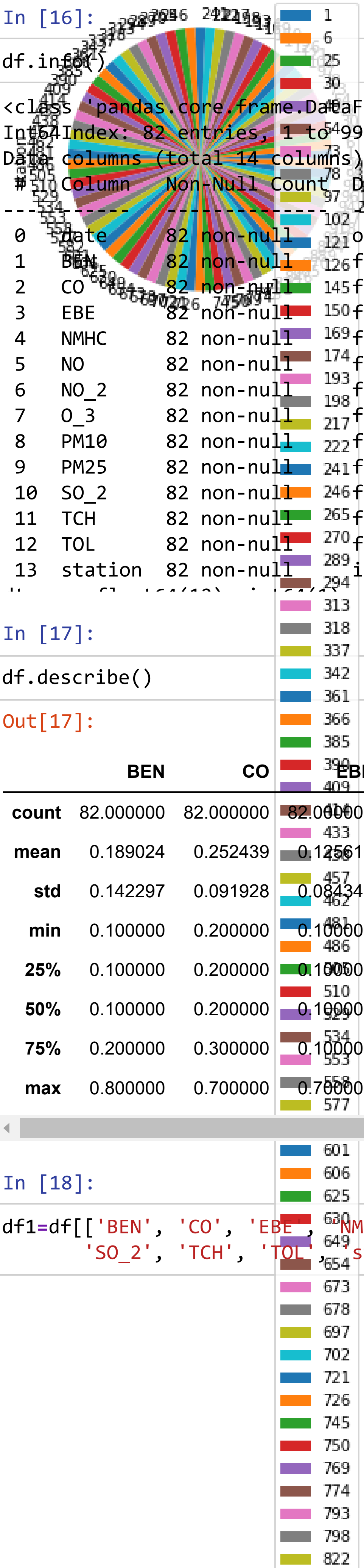
In [15]:

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

Out[15]:

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





In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 82 entries, 1 to 990
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     82 non-null     object
 1   BEN      82 non-null     float64
 2   CO       82 non-null     float64
 3   EBE      82 non-null     float64
 4   NMHC     82 non-null     float64
 5   NO       82 non-null     float64
 6   NO_2     82 non-null     float64
 7   O_3      82 non-null     float64
 8   PM10     82 non-null     float64
 9   PM25     82 non-null     float64
 10  SO_2     82 non-null     float64
 11  TCH      82 non-null     float64
 12  TOL      82 non-null     float64
 13  station  82 non-null     int64
```

In [17]:

```
df.describe()
```

Out[17]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM
count	82.000000	82.000000	82.000000	82.000000	82.000000	82.000000	82.000000	82.0000
mean	0.189024	0.252439	0.125610	0.187317	4.402439	16.341463	83.731707	12.5000
std	0.142297	0.091928	0.084343	0.060798	5.427092	13.573958	18.917004	6.9393
min	0.100000	0.200000	0.100000	0.100000	1.000000	1.000000	29.000000	4.0000
25%	0.100000	0.200000	0.100000	0.122500	1.000000	5.000000	74.250000	8.0000
50%	0.100000	0.200000	0.100000	0.225000	2.000000	13.500000	83.000000	11.0000
75%	0.200000	0.300000	0.100000	0.240000	5.000000	21.000000	94.000000	13.0000
max	0.800000	0.700000	0.700000	0.330000	24.000000	57.000000	136.000000	48.0000

In [18]:

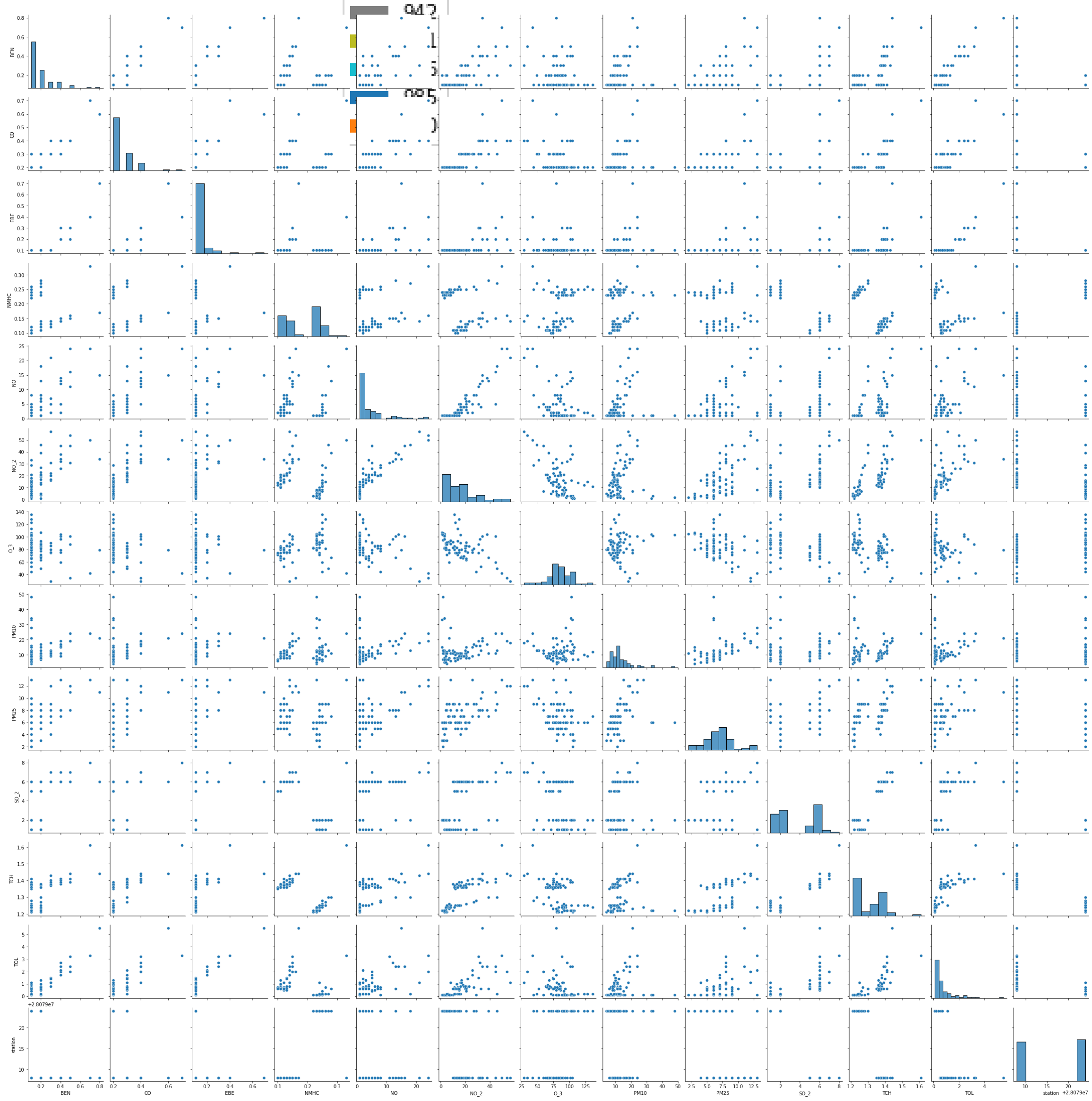
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
```

In [19]:

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

Out[19]:

<seaborn.axisgrid.PairGrid at 0x21b15615a60>



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]:

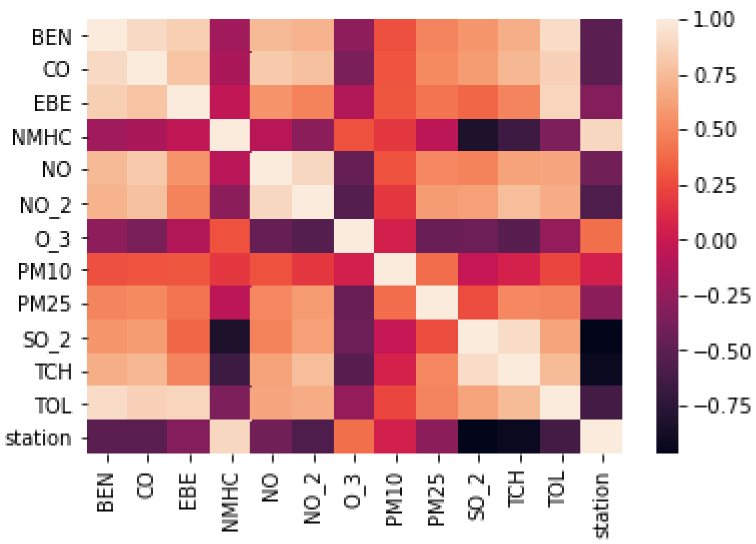
<AxesSubplot:xlabel='EBE', ylabel='Density'>

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]:

```
-1.862645149230957e-08
```

In [26]:

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

Out[26]:

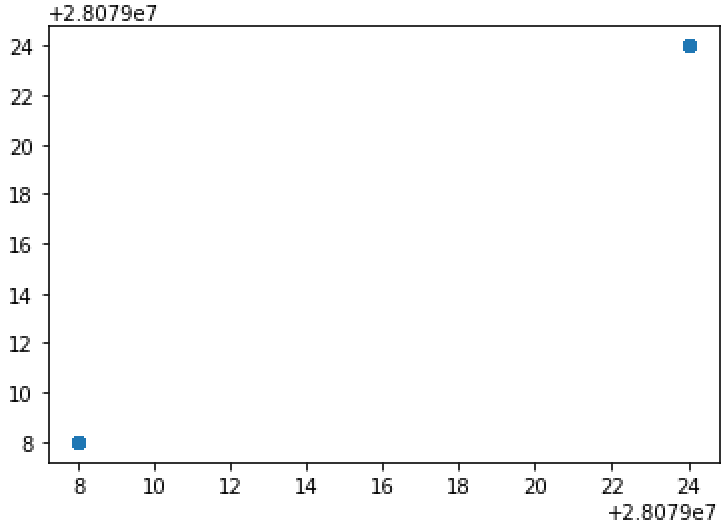
	Co-efficient
BEN	-2.429425e-15
CO	-1.894503e-14
EBE	-2.224266e-16
NMHC	2.107502e-14
NO	-2.737222e-16
NO_2	-1.475101e-16
O_3	1.694955e-16
PM10	8.878120e-16
PM25	-2.830153e-16
SO_2	-7.852760e-16
TCH	3.481835e-14
TOL	4.916475e-16
station	1.000000e+00

In [27]:

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

Out[27]:

```
<matplotlib.collections.PathCollection at 0x21b204f9b50>
```



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]:

```
0.9999616856415474
```

In [33]:

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

Out[33]:

```
0.9999464091286488
```

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]:

```
0.975205839493657
```

In [36]:

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

Out[36]:

```
0.9741204080244311
```

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([-0.00000000e+00, -0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
       -0.00000000e+00, -3.36576263e-03,  5.58859185e-04,  0.00000000e+00,
       -0.00000000e+00, -0.00000000e+00, -0.00000000e+00, -0.00000000e+00,
        9.80541138e-01])
```

In [39]:

```
e.intercept_
```

Out[39]:

```
546385.729947783
```

In [40]:

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

In [41]:

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

Out[41]:

```
0.9997111039704919
```

In [42]:

```
from sklearn import metrics
```

In [43]:

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

```
0.01822309930772365
```

In [44]:

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

```
0.13499296021542623
```

In [45]:

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

```
0.12863175049424172
```

In [46]:

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

In [47]:

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

In [48]:

```
feature_matrix.shape
```

Out[48]:

```
(82, 13)
```

In [49]:

```
target_vector.shape
```

Out[49]:

```
(82,)
```

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]]
```

In [54]:

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

```
[28079008]
```

In [55]:

```
logr.classes_
```

Out[55]:

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

In [56]:

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

Out[56]:

```
1.0
```

In [57]:

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

Out[57]:

```
0.9930679784665173
```

In [58]:

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

Out[58]:

```
array([[0.99306798, 0.00693202]])
```

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]:

```
1.0
```

In [64]:

```
rfc_best=grid_search.best_estimator_
```

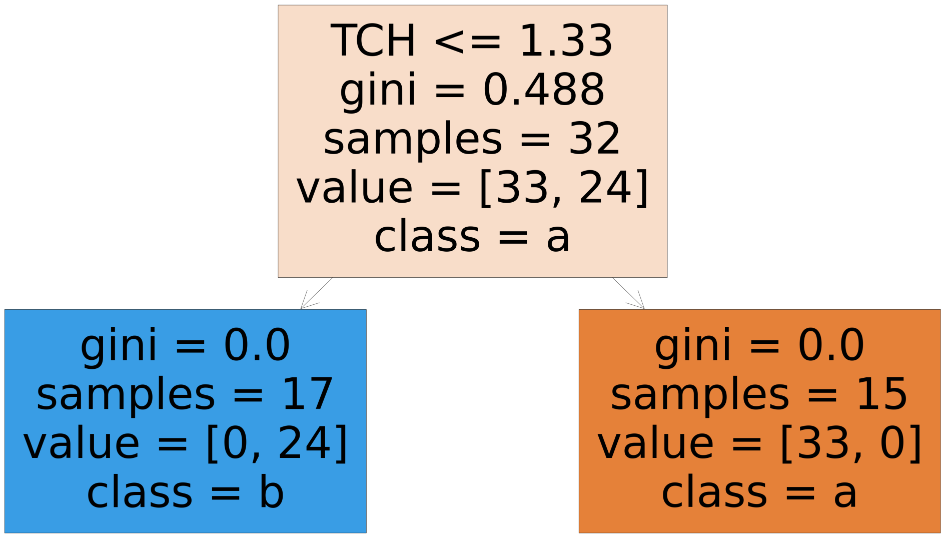
In [65]:

```
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, 'TCH <= 1.33\ngini = 0.488\nsamples = 32
\nvalue = [33, 24]\nclass = a'),
 Text(1116.0, 543.5999999999999, 'gini = 0.0\nsamples = 17\nvalue = [0, 2
4]\nclass = b'),
 Text(3348.0, 543.5999999999999, 'gini = 0.0\nsamples = 15\nvalue = [33,
0]\nclass = a')]
```



Logistic Regression is suitable

random forest is best suitable for this dataset

In [ ]: