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_2016.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	2016-11-01 01:00:00	NaN	0.7	NaN	NaN	153.0	77.0	NaN	NaN	NaN	7.0	NaN	NaN
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4
2	2016-11-01 01:00:00	5.9	NaN	7.5	NaN	297.0	139.0	NaN	NaN	NaN	NaN	NaN	26.0
3	2016-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	113.0	2.0	NaN	NaN	NaN	NaN	NaN
4	2016-11-01 01:00:00	NaN	NaN	NaN	NaN	275.0	127.0	2.0	NaN	NaN	18.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	2016-11-02 18:00:00	NaN	0.4	NaN	NaN	24.0	66.0	24.0	NaN	NaN	NaN	NaN	NaN
996	2016-11-02 18:00:00	NaN	NaN	NaN	NaN	9.0	56.0	NaN	35.0	NaN	5.0	NaN	NaN
997	2016-11-02 18:00:00	NaN	NaN	NaN	NaN	4.0	42.0	NaN	26.0	12.0	NaN	NaN	NaN
998	2016-11-02 18:00:00	NaN	NaN	NaN	NaN	25.0	70.0	NaN	34.0	13.0	NaN	NaN	NaN
999	2016-11-02 18:00:00	NaN	NaN	NaN	NaN	1.0	28.0	53.0	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	1.1	28079008
6	0.8	28079024
25	1.0	28079008
30	0.7	28079024
49	0.8	28079008
...	...	...
942	0.2	28079024
961	0.5	28079008
966	0.2	28079024
985	0.5	28079008
990	0.2	28079024

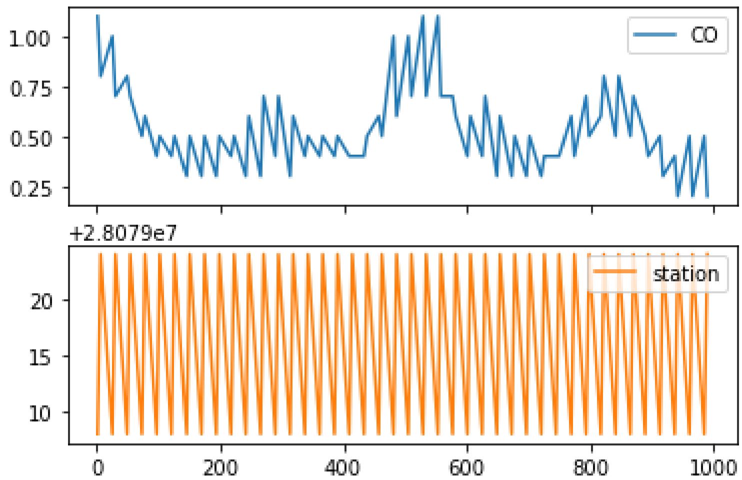
84 rows × 2 columns

In [7]:

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

Out[7]:

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

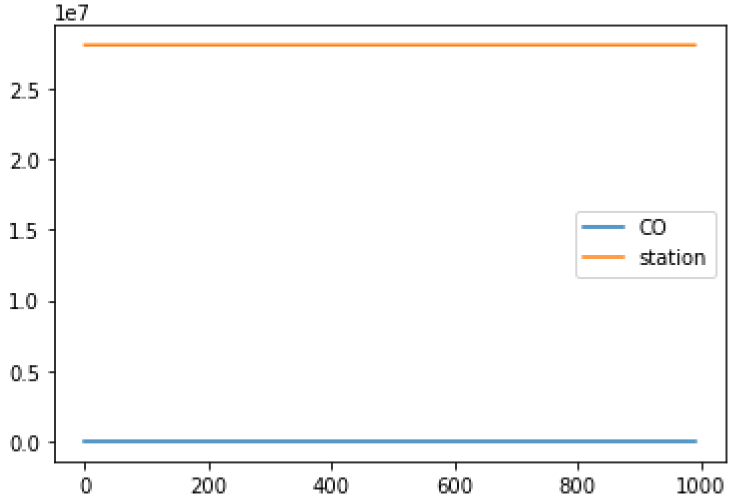


In [8]:

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

Out[8]:

<AxesSubplot:>



In [9]:

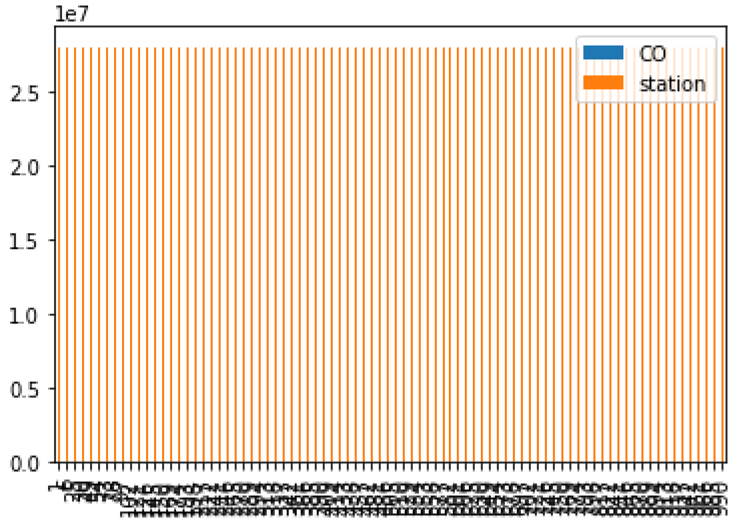
```
x = data[0:100]
```

In [10]:

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

Out[10]:

<AxesSubplot:>

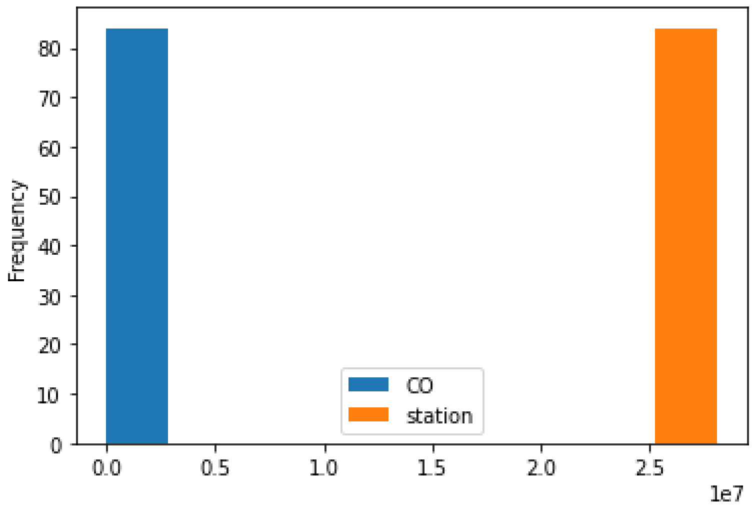


In [11]:

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

Out[11]:

<AxesSubplot:ylabel='Frequency'>

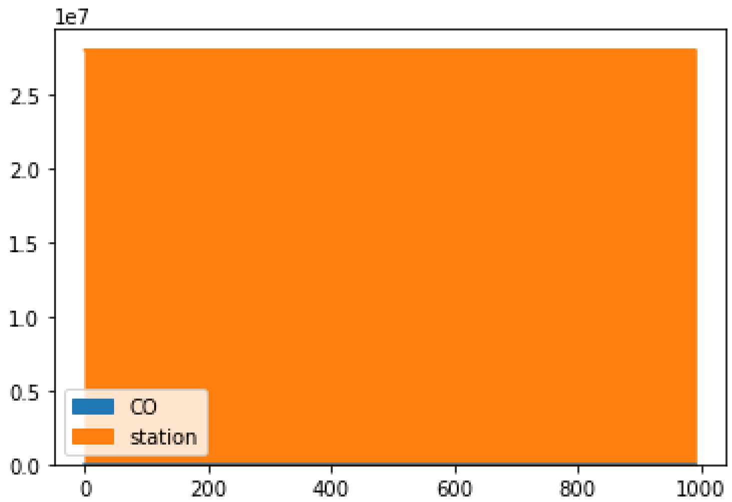


In [12]:

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

Out[12]:

<AxesSubplot:>

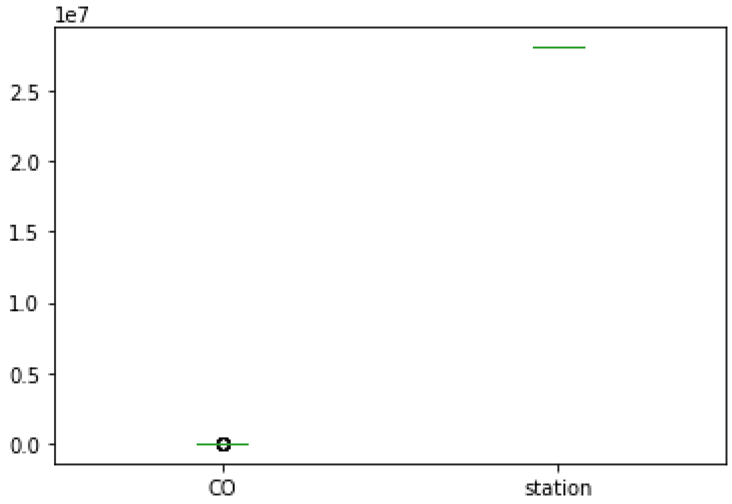


In [13]:

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

Out[13]:

<AxesSubplot:>



In [14]:

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

Out[14]:

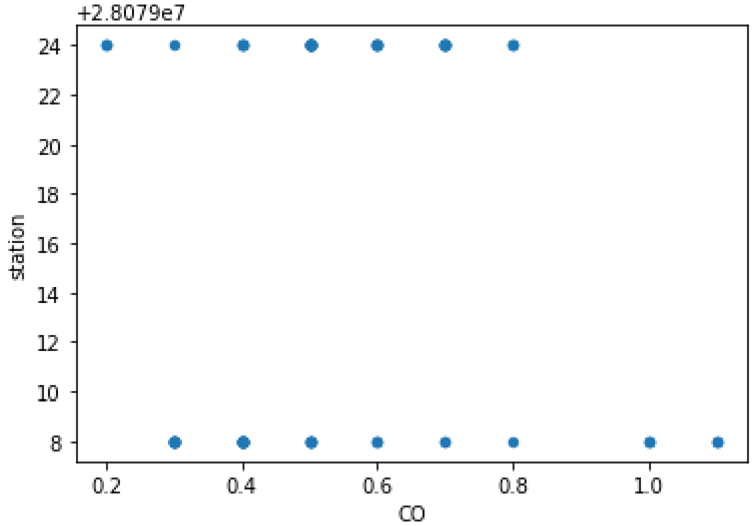
```
<AxesSubplot:ylabel='station'>
```

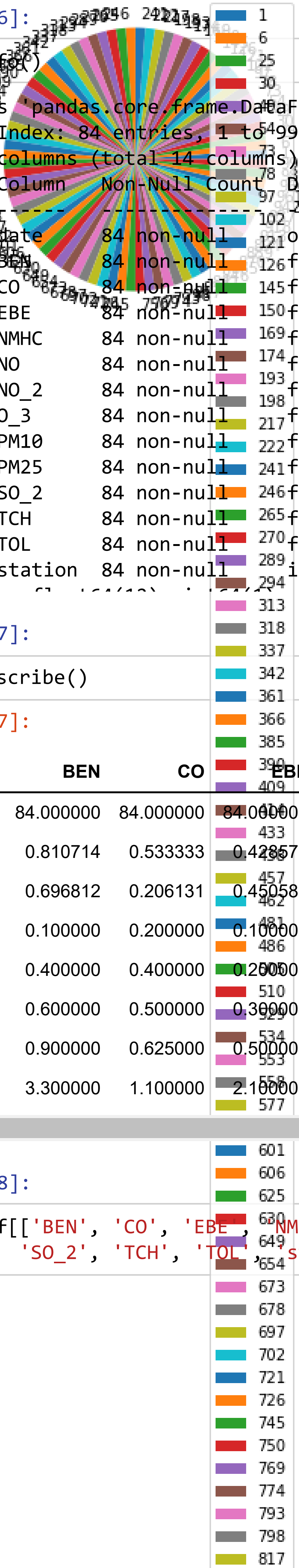
In [15]:

```
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	PI
count	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.000
mean	0.810714	0.533333	0.428571	0.140000	34.940476	55.642857	24.726190	27.916
std	0.696812	0.206131	0.450588	0.089375	48.969711	31.532979	23.506652	9.406
min	0.100000	0.200000	0.100000	0.050000	1.000000	14.000000	3.000000	12.000
25%	0.400000	0.400000	0.200000	0.080000	8.000000	34.250000	4.750000	22.000
50%	0.600000	0.500000	0.300000	0.120000	15.500000	48.500000	15.000000	27.000
75%	0.900000	0.625000	0.500000	0.150000	41.500000	69.000000	43.250000	33.250
max	3.300000	1.100000	2.100000	0.530000	260.000000	154.000000	75.000000	53.000

In [18]:

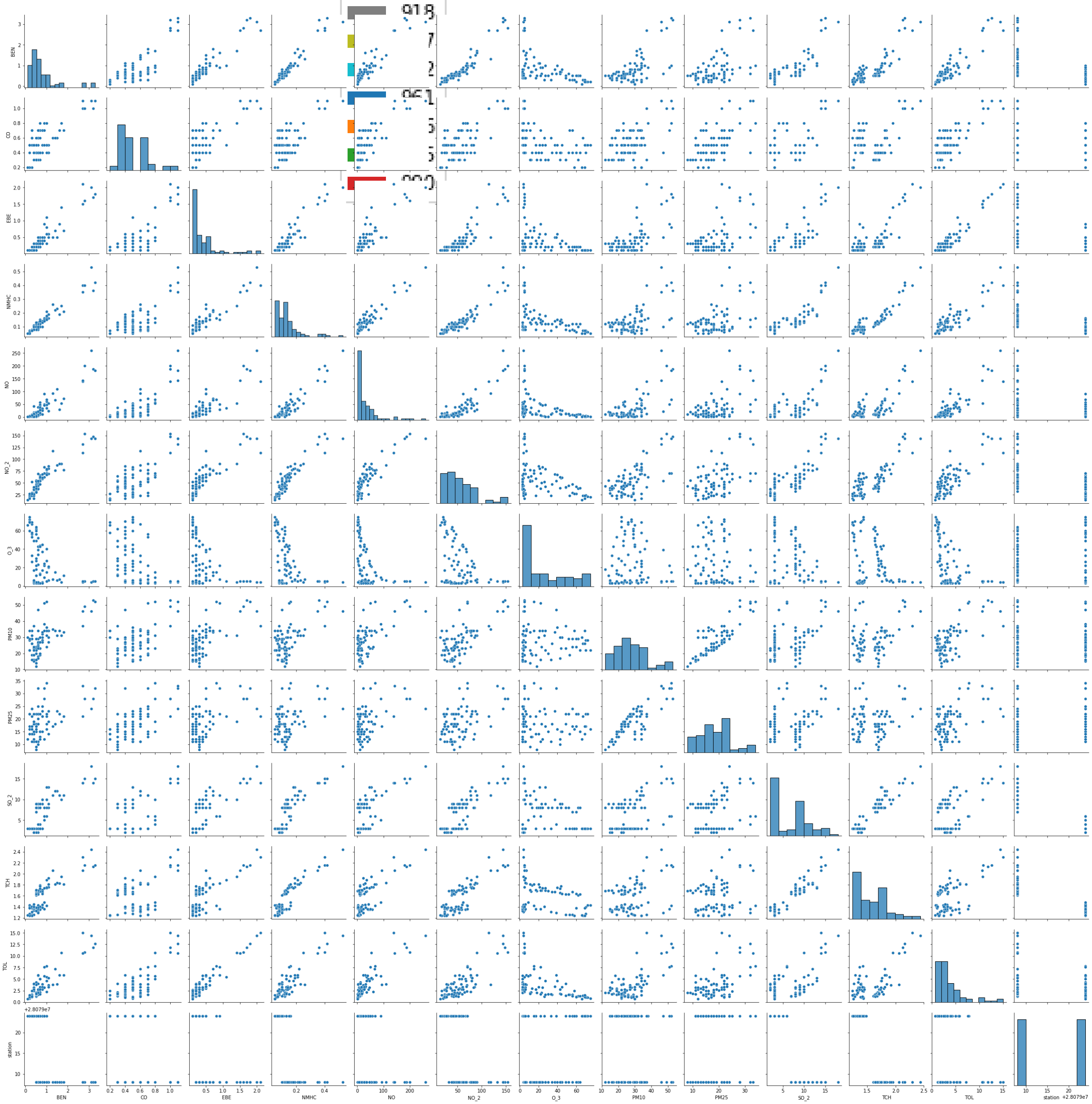
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station']]
```

In [19]:

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

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x1c2012f8c40>
```



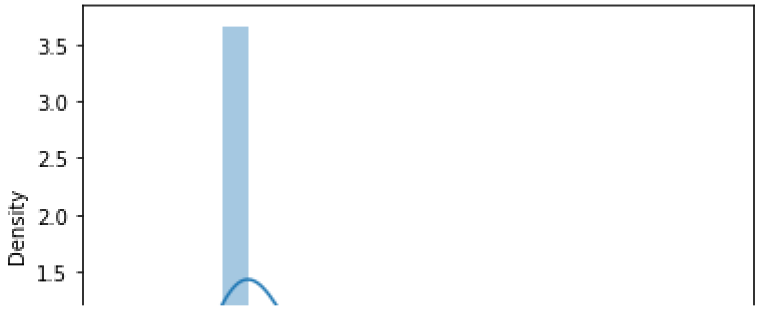
In [20]:

```
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'>

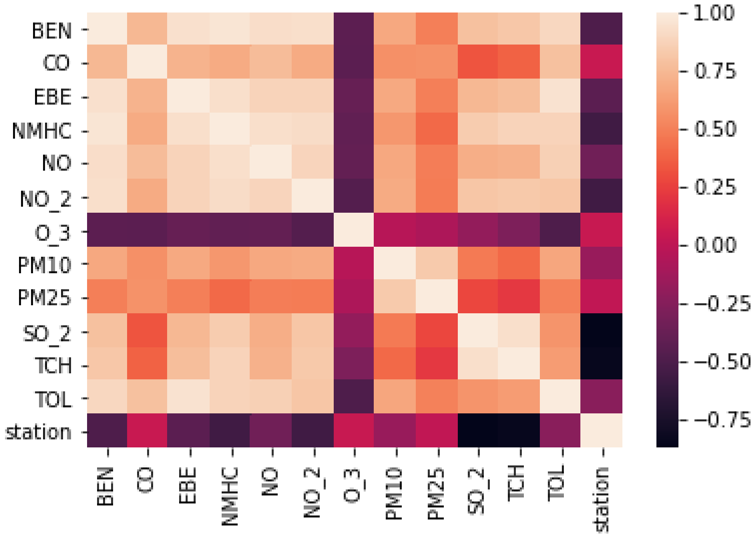


In [21]:

```
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]:

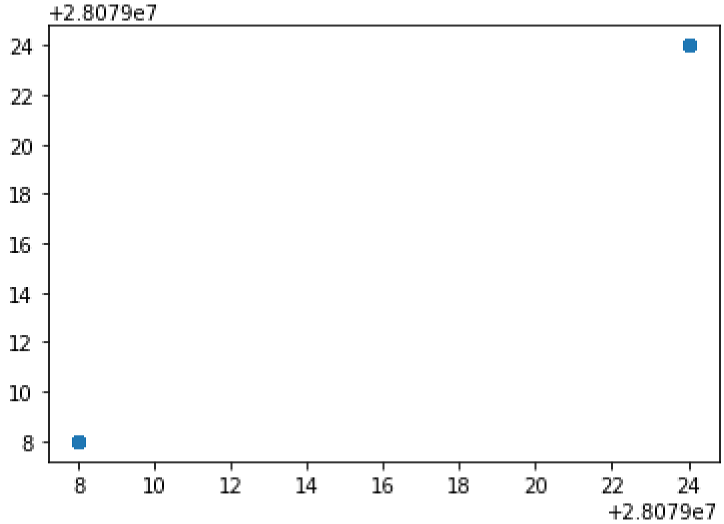
	Co-efficient
BEN	-3.238009e-15
CO	-4.990300e-15
EBE	-6.489599e-15
NMHC	6.480721e-14
NO	-1.834609e-17
NO_2	2.349135e-16
O_3	1.709426e-16
PM10	-8.613797e-17
PM25	1.505229e-16
SO_2	-2.153905e-16
TCH	-1.158721e-15
TOL	-9.261846e-17
station	1.000000e+00

In [27]:

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

Out[27]:

```
<matplotlib.collections.PathCollection at 0x1c20bfc7f40>
```



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]:

```
0.999809337721146
```

In [33]:

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

Out[33]:

```
0.99991459335366
```

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]:

```
0.9736271705434829
```

In [36]:

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

Out[36]:

```
0.9758175842587881
```

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, -2.70112598e-03, -7.62677321e-04,  0.00000000e+00,
        0.00000000e+00, -0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
        9.79924041e-01])
```

In [39]:

```
e.intercept_
```

Out[39]:

```
563713.33394951
```

In [40]:

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

In [41]:

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

Out[41]:

```
0.9997505655686159
```

In [42]:

```
from sklearn import metrics
```

In [43]:

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

```
0.01445243758647134
```

In [44]:

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

```
0.12021829139723847
```

In [45]:

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

```
0.11217871136390246
```

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]:

```
(84, 13)
```

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 [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]:

```
0.8385094351160735
```

In [58]:

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

Out[58]:

```
array([[0.83850944, 0.16149056]])
```

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_
```

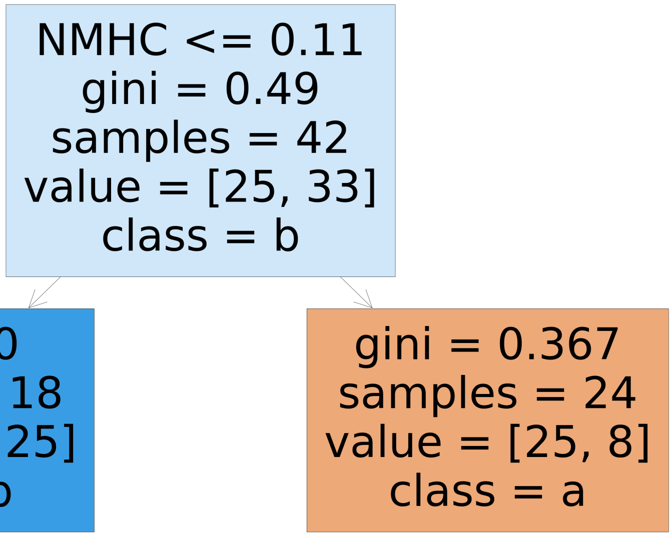
In [65]:

```
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, 'NMHC <= 0.11\ngini = 0.49\nsamples = 42
\nvalue = [25, 33]\nclass = b'),
 Text(1116.0, 543.5999999999999, 'gini = 0.0\nsamples = 18\nvalue = [0, 2
5]\nclass = b'),
 Text(3348.0, 543.5999999999999, 'gini = 0.367\nsamples = 24\nvalue = [25,
8]\nclass = a')]
```



random forest is the best suitable for this model

In [ ]:

```

```