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_2010.csv")
df = df1.head(1000)
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	Na
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	Na
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	Na
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.4100
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.6700
...	...	...	...	...	...	...	...	...	...	...	
995	2010-03-02 18:00:00	0.51	0.20	0.91	1.27	0.39	20.330000	22.940001	1.42	86.410004	14.2800
996	2010-03-02 18:00:00	NaN	NaN	NaN	NaN	0.13	28.370001	40.669998	NaN	73.480003	Na
997	2010-03-02 18:00:00	NaN	NaN	NaN	NaN	NaN	44.029999	50.509998	NaN	NaN	22.0499
998	2010-03-02 18:00:00	NaN	NaN	NaN	NaN	NaN	31.770000	37.040001	NaN	85.040001	Na
999	2010-03-02 18:00:00	NaN	NaN	NaN	NaN	NaN	32.500000	39.279999	NaN	NaN	16.3500

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: 83 entries, 11 to 995
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     83 non-null     object 
 1   BEN      83 non-null     float64
 2   CO       83 non-null     float64
 3   EBE      83 non-null     float64
 4   MXY      83 non-null     float64
 5   NMHC     83 non-null     float64
 6   NO_2     83 non-null     float64
 7   NOx      83 non-null     float64
 8   OXY      83 non-null     float64
 9   O_3      83 non-null     float64
 10  PM10     83 non-null     float64
 11  PM25     83 non-null     float64
 12  PXY      83 non-null     float64
 13  SO_2     83 non-null     float64
 14  TCH      83 non-null     float64
 15  TOL      83 non-null     float64
 16  station  83 non-null     int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 11.7+ KB
```

In [6]:

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

Out[6]:

	CO	station
11	0.18	28079024
23	0.23	28079099
35	0.17	28079024
47	0.21	28079099
59	0.16	28079024
...	...	...
947	0.20	28079024
959	0.31	28079099
971	0.18	28079024
983	0.31	28079099
995	0.20	28079024

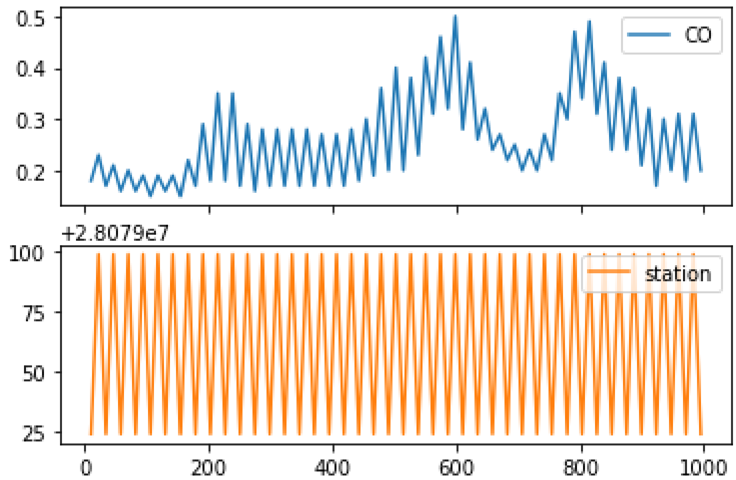
83 rows × 2 columns

In [7]:

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

Out[7]:

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

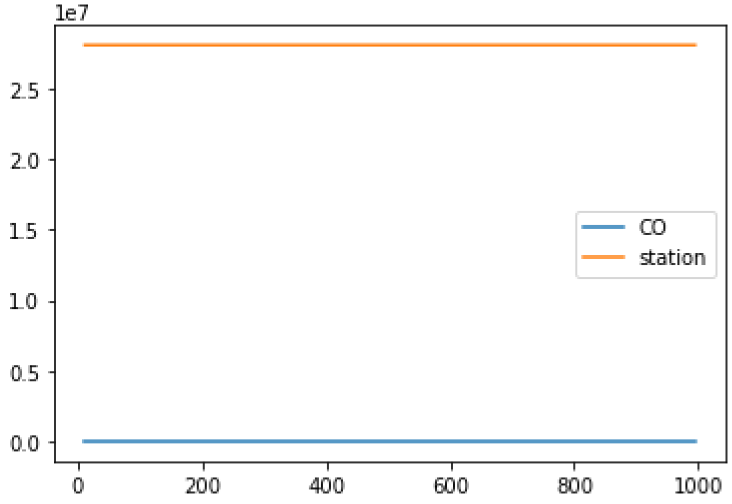


In [8]:

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

Out[8]:

<AxesSubplot:>



In [9]:

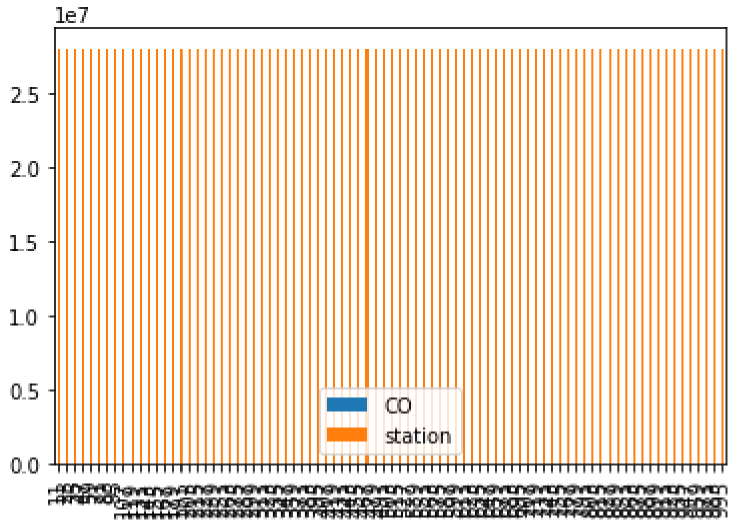
```
x = data[0:100]
```

In [10]:

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

Out[10]:

<AxesSubplot:>

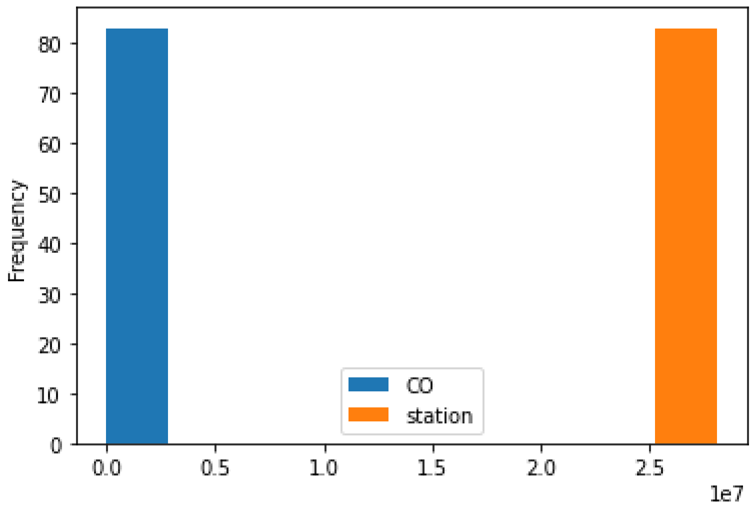


In [11]:

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

Out[11]:

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

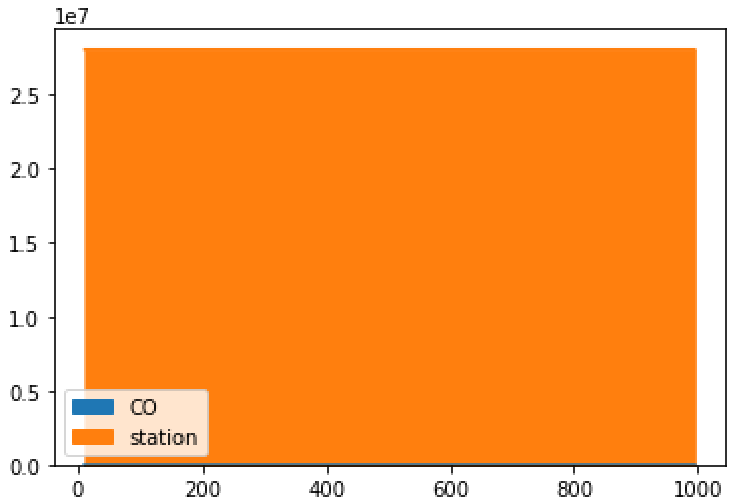


In [12]:

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

Out[12]:

```
<AxesSubplot:>
```

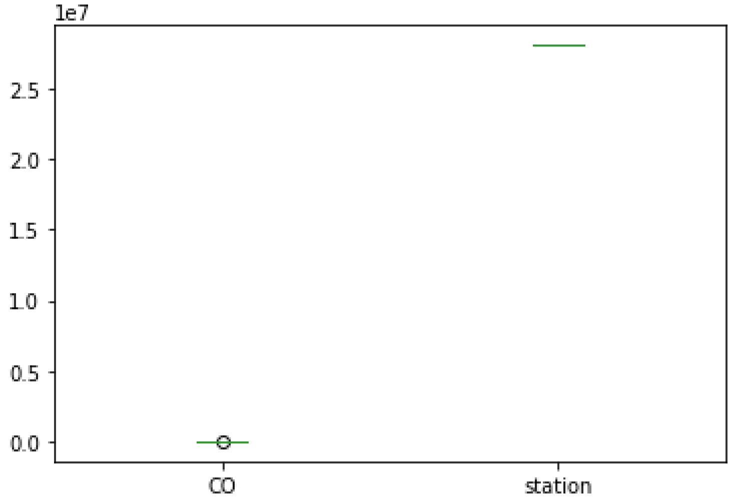


In [13]:

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

Out[13]:

<AxesSubplot:>



In [14]:

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

Out[14]:

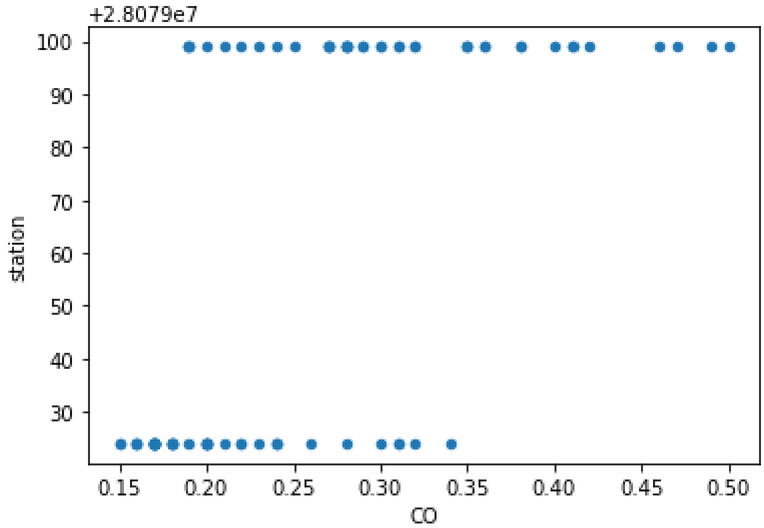
```
<AxesSubplot:ylabel='station'>
```

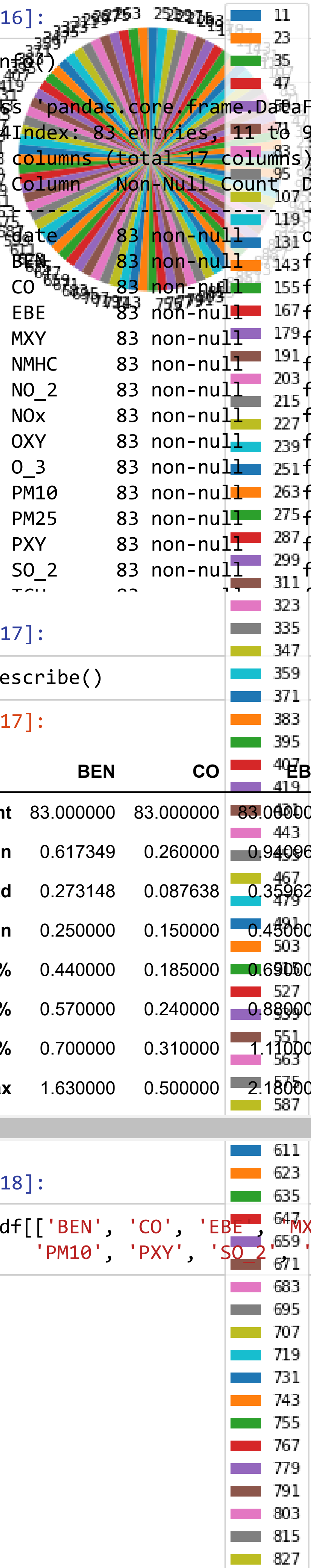
In [15]:

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

Out[15]:

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





In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 83 entries, 11 to 995
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     83 non-null     object
 1   BEN      83 non-null     float64
 2   CO       83 non-null     float64
 3   EBE      83 non-null     float64
 4   MXY      83 non-null     float64
 5   NMHC     83 non-null     float64
 6   NO_2     83 non-null     float64
 7   NOx      83 non-null     float64
 8   OXY      83 non-null     float64
 9   O_3      83 non-null     float64
 10  PM10     83 non-null     float64
 11  PM25     83 non-null     float64
 12  PXY      83 non-null     float64
 13  SO_2     83 non-null     float64
```

In [17]:

```
df.describe()
```

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	O)
count	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	83.000000	83.0000
mean	0.617349	0.260000	0.940964	0.837952	0.276988	29.832771	39.108795	1.0460
std	0.273148	0.087638	0.359629	0.300766	0.079599	22.784586	30.204628	0.1966
min	0.250000	0.150000	0.450000	0.380000	0.170000	1.290000	2.780000	0.9000
25%	0.440000	0.185000	0.690000	0.580000	0.210000	10.450000	13.015000	1.0000
50%	0.570000	0.240000	0.880000	1.000000	0.260000	25.110001	33.189999	1.0000
75%	0.700000	0.310000	1.110000	1.000000	0.330000	46.590000	57.724998	1.0000
max	1.630000	0.500000	2.180000	1.780000	0.470000	84.629997	117.300003	1.8900

In [18]:

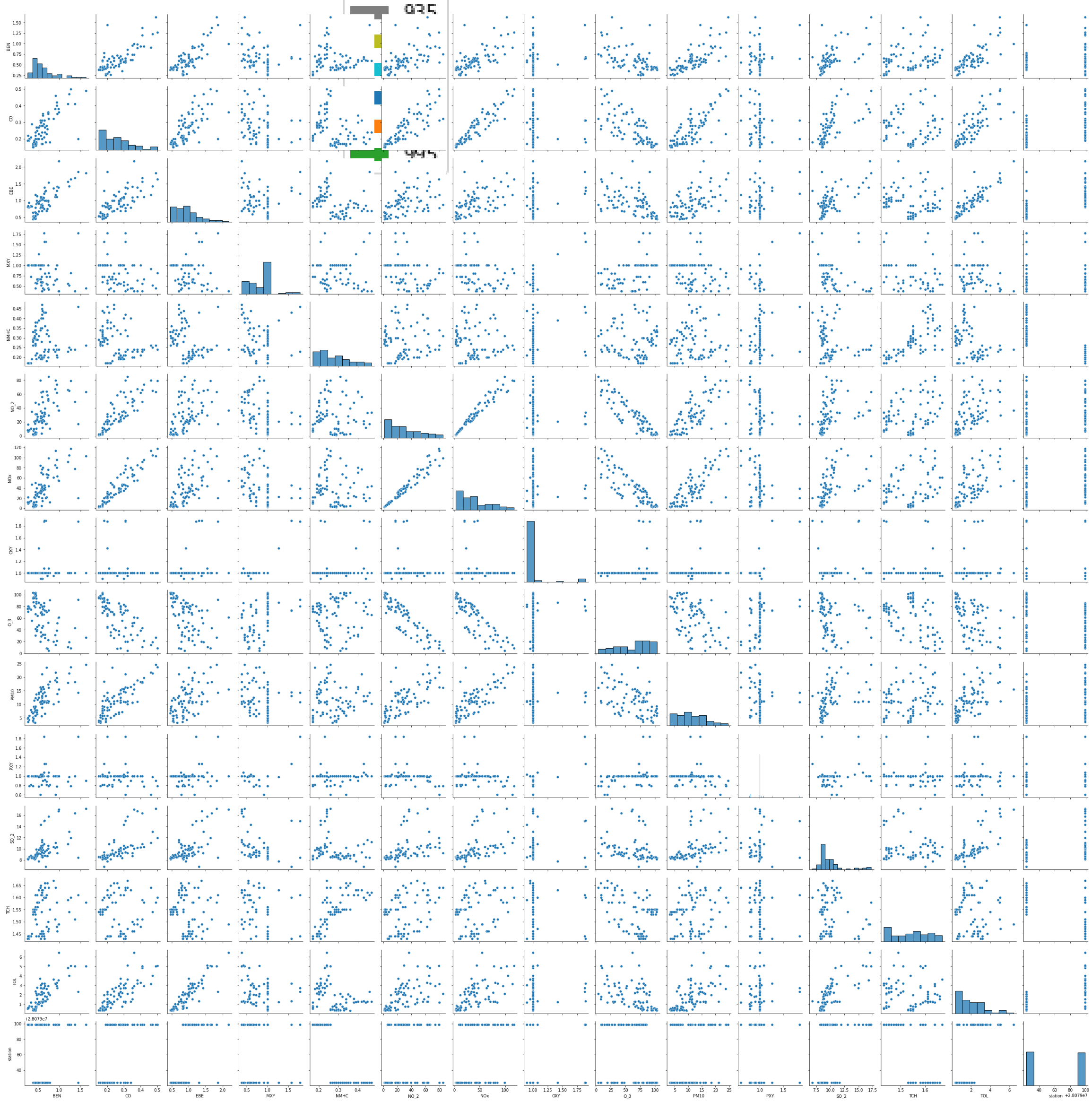
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [19]:

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

Out[19]:

<seaborn.axisgrid.PairGrid at 0x276432691c0>



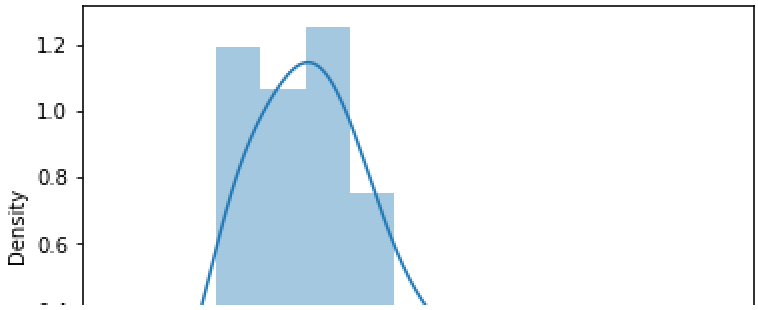
In [20]:

```
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'>
```

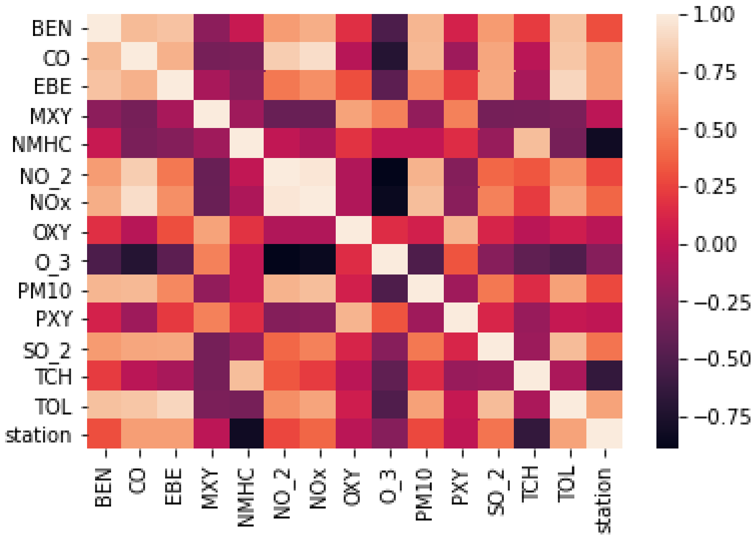


In [21]:

```
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]:

```
28079074.34038417
```

In [26]:

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

Out[26]:

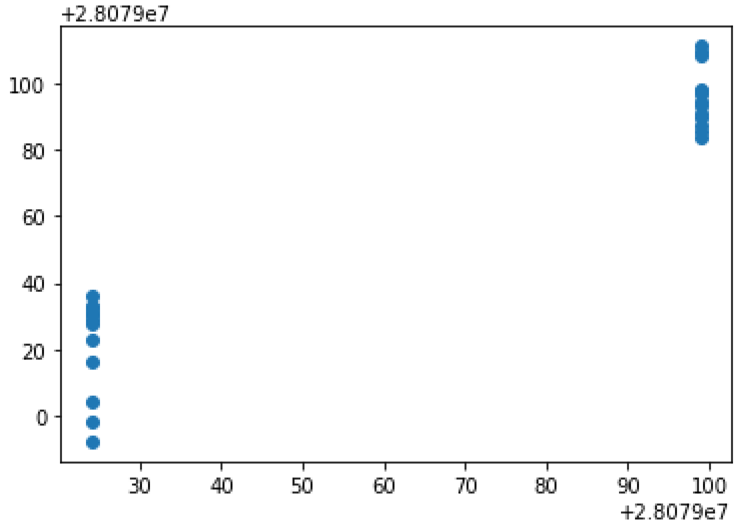
	Co-efficient
BEN	-52.384626
CO	532.029917
EBE	42.284118
MXY	-21.878818
NMHC	-242.407884
NO_2	-0.001104
NOx	-0.925168
OXY	26.514900
O_3	-0.085455
PM10	0.693944
PXY	17.850938
SO_2	-3.663803
TCH	-29.301800
TOL	-2.569086

In [27]:

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

Out[27]:

```
<matplotlib.collections.PathCollection at 0x27650566ee0>
```



In [28]:

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

Out[28]:

```
0.8954917688152905
```

In [29]:

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

Out[29]:

```
0.9531178497077009
```

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]:

```
0.5823870918034553
```

In [33]:

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

Out[33]:

```
0.6140046945202979
```

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]:

```
0.476659111651906
```

In [36]:

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

Out[36]:

```
0.43288567135492795
```

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([-2.17598345,  0.        ,  1.86730036,  1.47260202, -1.79421771,
       -2.99078316,  2.26850185, -0.        , -0.27221313, -1.2281746 ,
        0.        ,  0.25872199, -1.35687699,  8.79741552])
```

In [39]:

```
e.intercept_
```

Out[39]:

```
28079072.770726595
```

In [40]:

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

In [41]:

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

Out[41]:

```
0.5010724189970784
```

In [42]:

```
from sklearn import metrics
```

In [43]:

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

```
700.4943237281019
```

In [44]:

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

```
26.46685330234975
```

In [45]:

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

```
24.714530514627697
```

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]:

```
(83, 14)
```

In [49]:

```
target_vector.shape
```

Out[49]:

```
(83,)
```

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)
```

```
[28079024]
```

In [55]:

```
logr.classes_
```

Out[55]:

```
array([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]:

```
0.9987145936322467
```

In [58]:

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

Out[58]:

```
array([[0.99871459, 0.00128541]])
```

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_
```

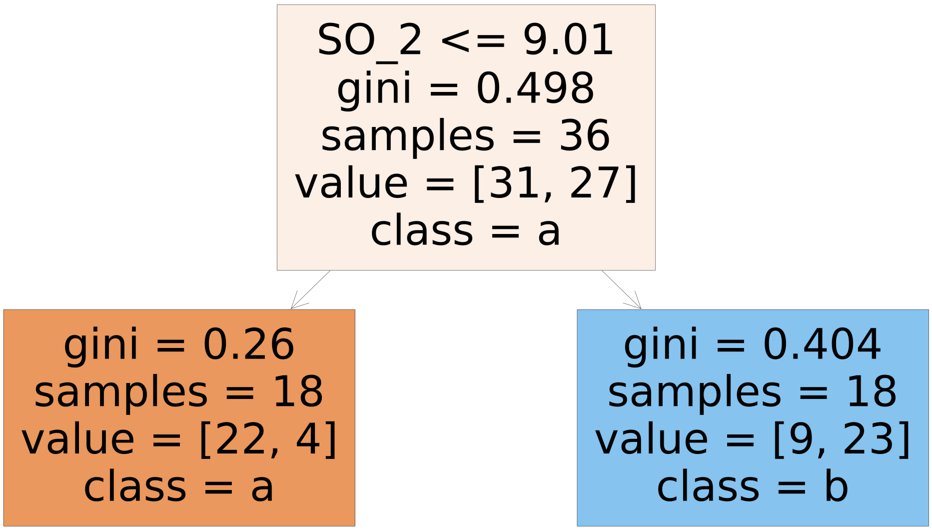
In [65]:

```
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, 'SO_2 <= 9.01\ngini = 0.498\nsamples = 3
6\nvalue = [31, 27]\nclass = a'),
 Text(1116.0, 543.5999999999999, 'gini = 0.26\nsamples = 18\nvalue = [22,
4]\nclass = a'),
 Text(3348.0, 543.5999999999999, 'gini = 0.404\nsamples = 18\nvalue = [9,
23]\nclass = b')]
```



random forest is best suitable for this dataset

In [ ]: