Importing Libraries

In [137]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing Datasets

In [138]:
```
df=pd.read_csv("madrid_2015.csv")
df
```

Out[138]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2015-10-01 01:00:00	NaN	0.8	NaN	NaN	90.0	82.0	NaN	NaN	NaN	10.0	NaN	NaN	28
1	2015-10-01 01:00:00	2.0	0.8	1.6	0.33	40.0	95.0	4.0	37.0	24.0	12.0	1.83	8.3	28
2	2015-10-01 01:00:00	3.1	NaN	1.8	NaN	29.0	97.0	NaN	NaN	NaN	NaN	NaN	7.1	28
3	2015-10-01 01:00:00	NaN	0.6	NaN	NaN	30.0	103.0	2.0	NaN	NaN	NaN	NaN	NaN	28
4	2015-10-01 01:00:00	NaN	NaN	NaN	NaN	95.0	96.0	2.0	NaN	NaN	9.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210091	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	11.0	33.0	53.0	NaN	NaN	NaN	NaN	NaN	28
210092	2015-08-01 00:00:00	NaN	0.2	NaN	NaN	1.0	5.0	NaN	26.0	NaN	10.0	NaN	NaN	28
210093	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	7.0	74.0	NaN	NaN	NaN	NaN	NaN	28
210094	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	3.0	7.0	65.0	NaN	NaN	NaN	NaN	NaN	28
210095	2015-08-01 00:00:00	NaN	NaN	NaN	NaN	1.0	9.0	54.0	29.0	NaN	NaN	NaN	NaN	28

210096 rows × 14 columns

Data Cleaning and Data Preprocessing

In [139]:
```
df=df.dropna()
```

In [140]:
```
df.columns
```

Out[140]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [141]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16026 entries, 1 to 210078
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16026 non-null  object 
 1   BEN      16026 non-null  float64
 2   CO       16026 non-null  float64
 3   EBE      16026 non-null  float64
 4   NMHC     16026 non-null  float64
 5   NO       16026 non-null  float64
 6   NO_2     16026 non-null  float64
 7   O_3      16026 non-null  float64
 8   PM10     16026 non-null  float64
 9   PM25     16026 non-null  float64
 10  SO_2     16026 non-null  float64
 11  TCH      16026 non-null  float64
 12  TOL      16026 non-null  float64
 13  station  16026 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.8+ MB
```

In [142]:
```
data=df[['CO' ,'station']]
data
```

Out[142]:

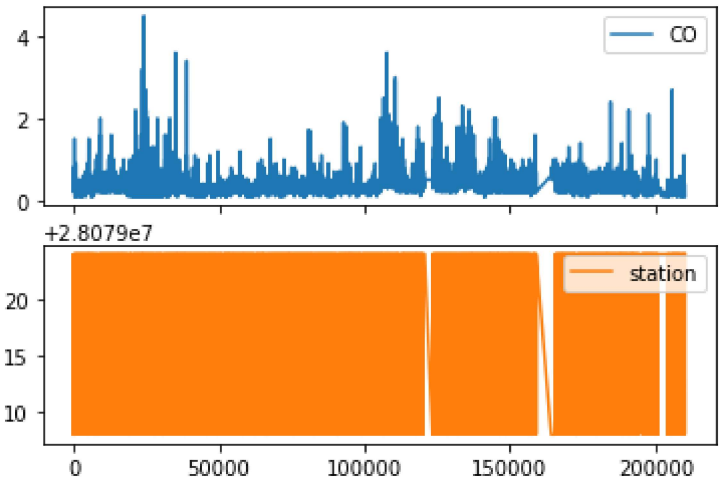
	CO	station
1	0.8	28079008
6	0.3	28079024
25	0.7	28079008
30	0.3	28079024
49	0.8	28079008
...	...	...
210030	0.1	28079024
210049	0.3	28079008
210054	0.1	28079024
210073	0.3	28079008
210078	0.1	28079024

16026 rows × 2 columns

Line chart

In [143]:
```
data.plot.line(subplots=True)
```

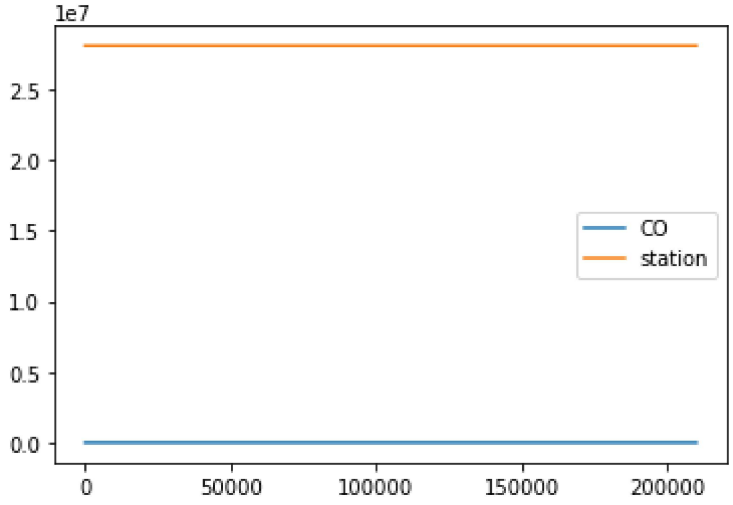
Out[143]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [144]:
```
data.plot.line()
```

Out[144]: <AxesSubplot:>

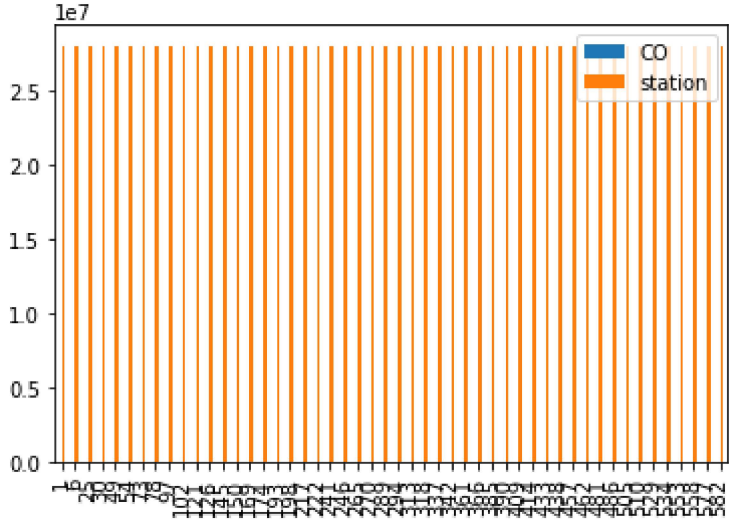


Bar chart

In [145]:
```
b=data[0:50]
```

In [146]:
```
b.plot.bar()
```

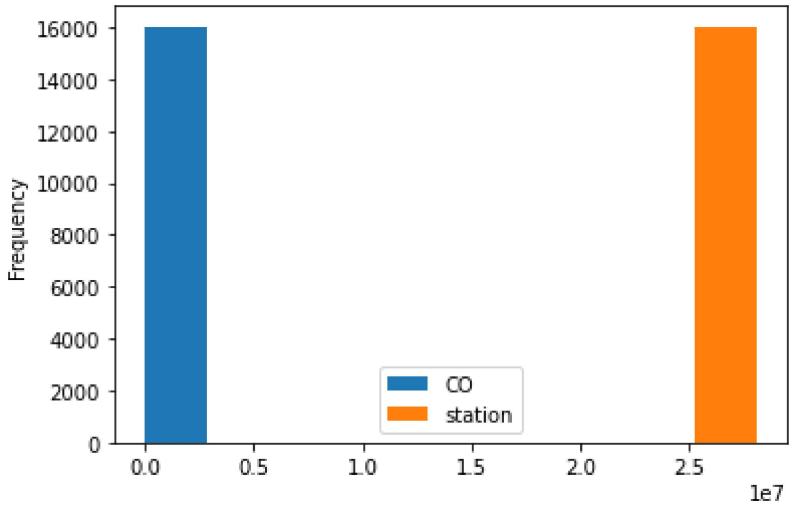
Out[146]: <AxesSubplot:>



Histogram

In [147]:
```
data.plot.hist()
```

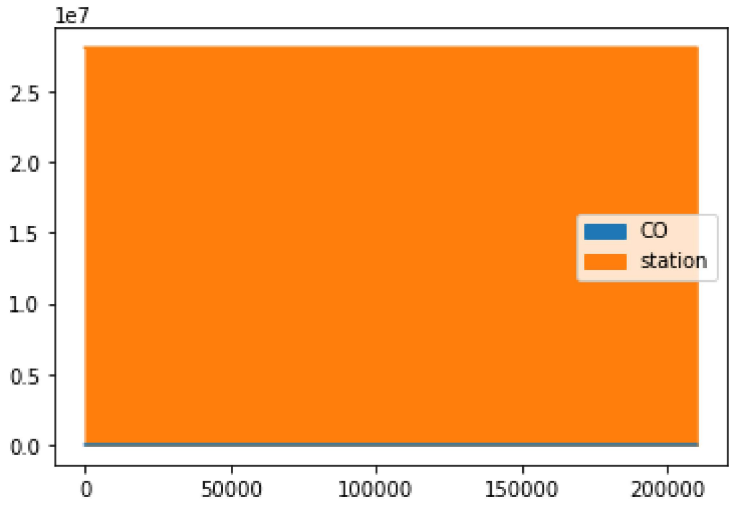
Out[147]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [148]:
```
data.plot.area()
```

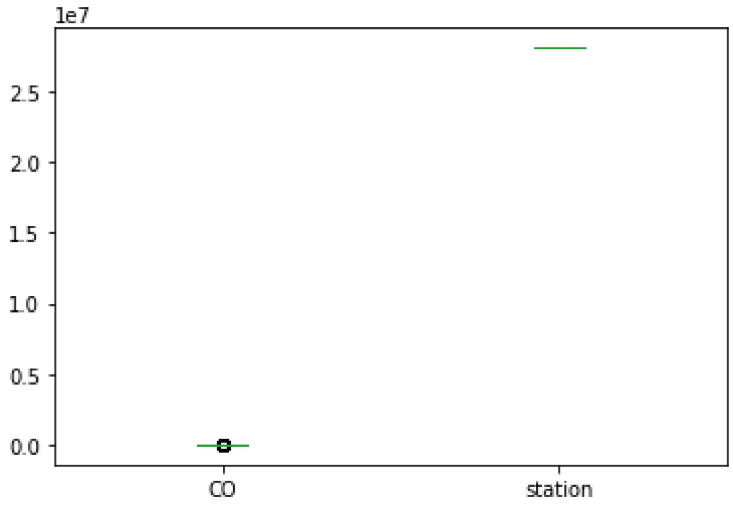
Out[148]: <AxesSubplot:>



Box chart

In [149]:
```
data.plot.box()
```

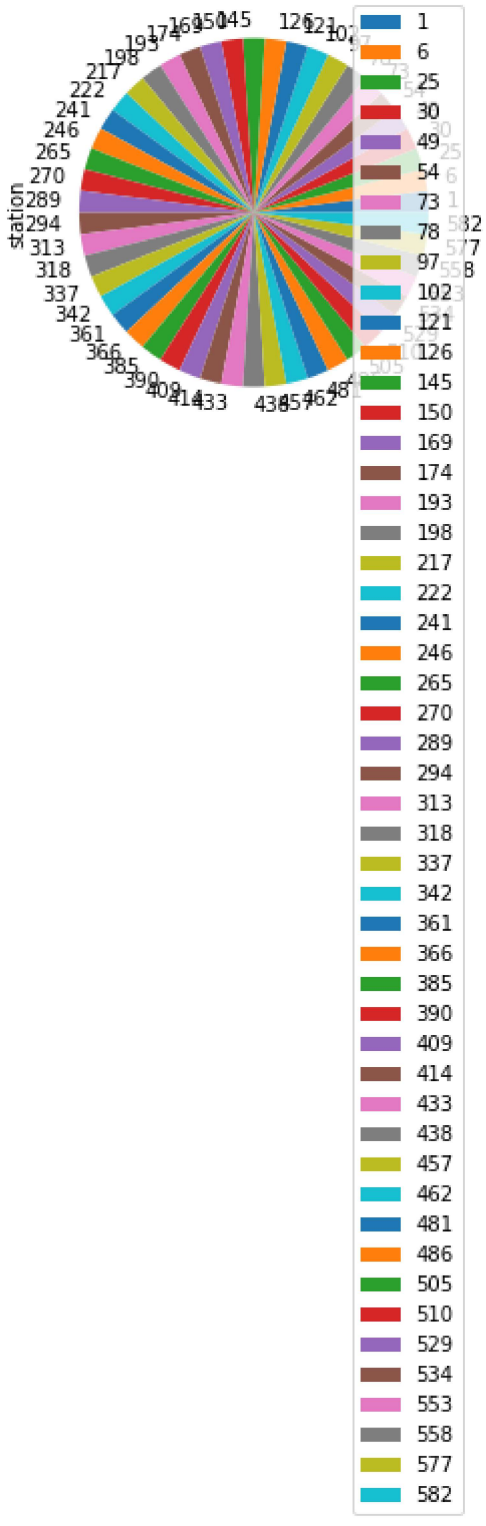
Out[149]: <AxesSubplot:>



Pie chart

In [150]:
```
b.plot.pie(y='station' )
```

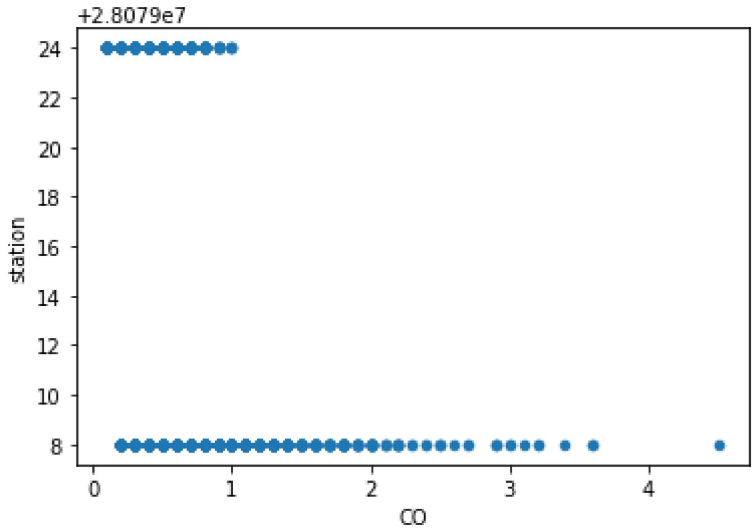
Out[150]: <AxesSubplot:ylabel='station'>



Scatter chart

In [151]:
```
data.plot.scatter(x='CO' ,y='station')
```

Out[151]:
```
<AxesSubplot:xlabel='CO', ylabel='station'>
```



In [152]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16026 entries, 1 to 210078
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16026 non-null  object 
 1   BEN      16026 non-null  float64
 2   CO       16026 non-null  float64
 3   EBE      16026 non-null  float64
 4   NMHC     16026 non-null  float64
 5   NO       16026 non-null  float64
 6   NO_2     16026 non-null  float64
 7   O_3      16026 non-null  float64
 8   PM10     16026 non-null  float64
 9   PM25     16026 non-null  float64
 10  SO_2     16026 non-null  float64
 11  TCH      16026 non-null  float64
 12  TOL      16026 non-null  float64
 13  station  16026 non-null  int64  
dtypes: float64(12), int64(1), object(1)
```

In [153]:
```
df.describe()
```

Out[153]:

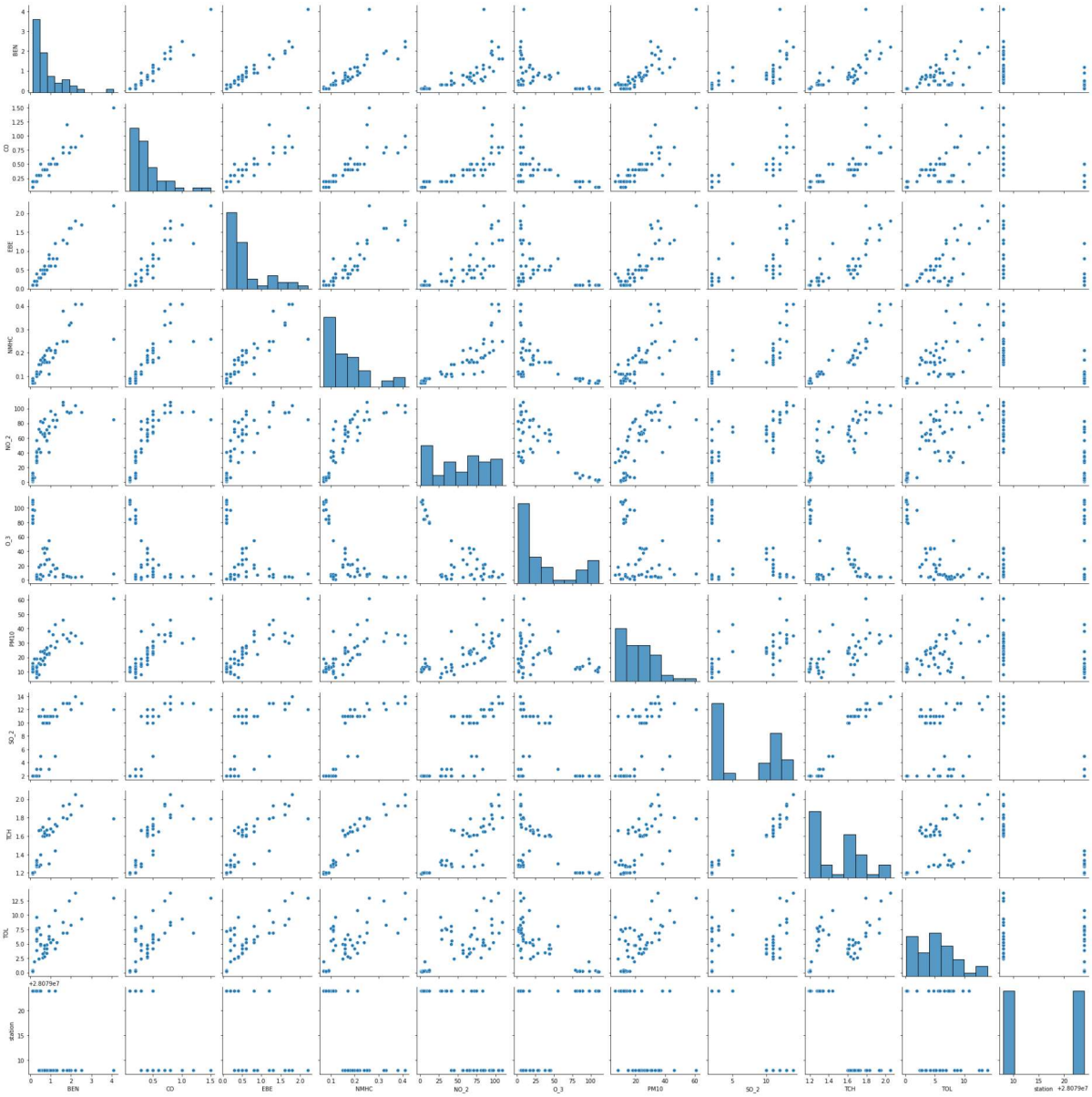
	BEN	CO	EBE	NMHC	NO	NO_2	
count	16026.000000	16026.000000	16026.000000	16026.000000	16026.000000	16026.000000	160
mean	0.504823	0.380594	0.394247	0.123099	23.842256	40.948771	
std	0.716896	0.260805	0.678592	0.092368	51.255660	33.236098	
min	0.100000	0.100000	0.100000	0.000000	1.000000	1.000000	
25%	0.100000	0.200000	0.100000	0.070000	1.000000	14.000000	
50%	0.200000	0.300000	0.100000	0.100000	6.000000	35.000000	
75%	0.700000	0.500000	0.400000	0.140000	24.000000	60.000000	
max	17.700001	4.500000	12.100000	1.090000	960.000000	369.000000	2

In [154]:
```
df1=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [155]:
```
sns.pairplot(df1[0:50])
```

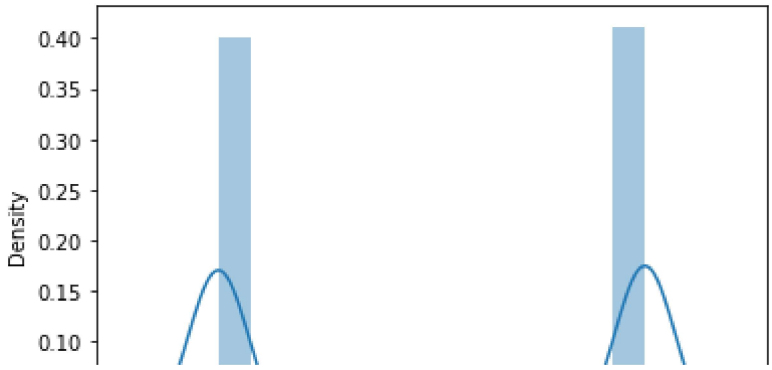
Out[155]: <seaborn.axisgrid.PairGrid at 0x14d3e2a5610>



In [156]:
```
sns.distplot(df1['station'])
```

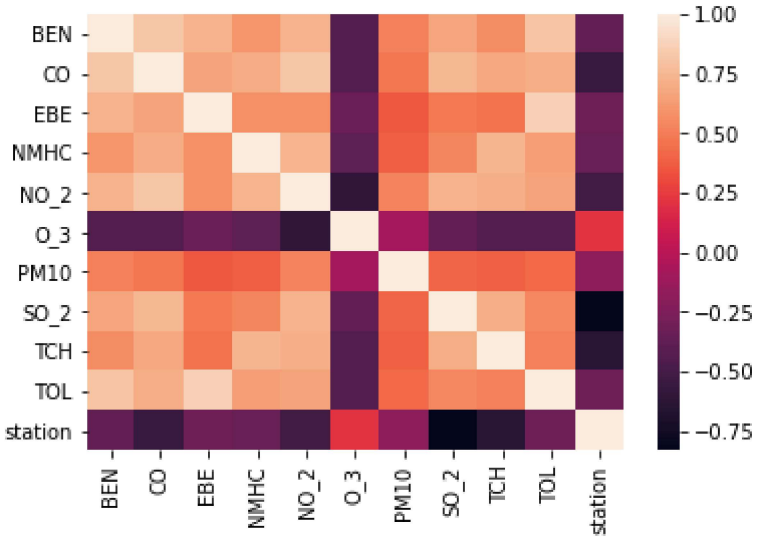
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[156]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [157]:
```
sns.heatmap(df1.corr())
```

Out[157]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [158]:
```
x=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [159]:
```
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)
```

Linear Regression

In [160]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[160]:
```
LinearRegression()
```

In [161]:
```
lr.intercept_
```

Out[161]:
```
28079038.0914884
```

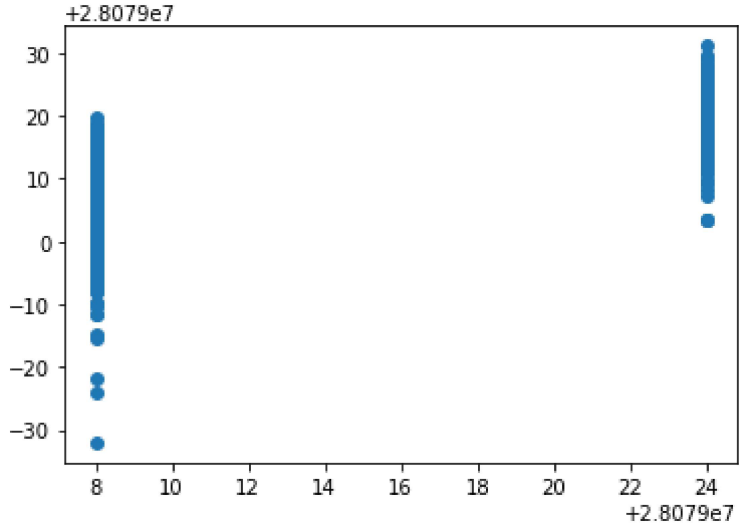
In [162]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[162]:

	Co-efficient
BEN	4.527468
CO	-6.927215
EBE	-1.141429
NMHC	24.260464
NO_2	0.000622
O_3	-0.017705
PM10	0.061161
SO_2	-1.184415
TCH	-11.127504
TOL	-0.039124

In [163]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[163]: <matplotlib.collections.PathCollection at 0x14d3d64fcd0>



ACCURACY

In [164]:
```
lr.score(x_test,y_test)
```

Out[164]: 0.799592638630653

In [165]:
```
lr.score(x_train,y_train)
```

Out[165]: 0.8116323469799842

Ridge and Lasso

In [166]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [167]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[167]: Ridge(alpha=10)

Accuracy(Ridge)

In [168]:
```
rr.score(x_test,y_test)
```

Out[168]: 0.7977007993462213

In [169]:
```
rr.score(x_train,y_train)
```

Out[169]:
```
0.809565586347635
```

Accuracy(Lasso)

In [170]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[170]:
```
Lasso(alpha=10)
```

In [171]:
```
la.score(x_train,y_train)
```

Out[171]:
```
0.6359313105656521
```

ElasticNet

In [172]:
```
la.score(x_test,y_test)
```

Out[172]:
```
0.6428282849894491
```

In [173]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[173]:
```
ElasticNet()
```

In [174]:
```
en.coef_
```

Out[174]:
```
array([ 0.04181634, -0.        , -0.        ,  0.        ,  0.        ,
       -0.01481954,  0.07432967, -1.27543538, -0.        ,  0.18608769])
```

In [175]:
```
en.intercept_
```

Out[175]:
```
28079024.083465632
```

In [176]:
```
prediction=en.predict(x_test)
```

In [177]:
```
en.score(x_test,y_test)
```

Out[177]:
```
0.730433930762691
```

Evaluation Metrics

In [178]:
```
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
3.2168449023358847
17.25021042279018
4.15333726330889
```

Logistic Regression

In [179]:
```
from sklearn.linear_model import LogisticRegression
```

In [180]:
```
feature_matrix=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [181]:
```
feature_matrix.shape
```

Out[181]:
```
(16026, 10)
```

In [182]:
```
target_vector.shape
```

Out[182]:
```
(16026,)
```

In [183]:
```
from sklearn.preprocessing import StandardScaler
```

In [184]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [185]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[185]:
```
LogisticRegression(max_iter=10000)
```

In [186]:
```
observation=[[1,2,3,4,5,6,7,8,9,10]]
```

In [187]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079008]
```

In [188]:
```
logr.classes_
```

Out[188]:
```
array([28079008, 28079024], dtype=int64)
```

In [189]:
```
logr.score(fs,target_vector)
```

Out[189]:
```
0.9947585174092101
```

In [190]:
```
logr.predict_proba(observation)[0][0]
```

Out[190]:
```
1.0
```

In [191]:
```
logr.predict_proba(observation)
```

Out[191]:
```
array([[1.00000000e+00, 5.69793111e-39]])
```

Random Forest

In [192]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [193]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[193]:
```
RandomForestClassifier()
```

In [194]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [195]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```

Out[195]:
```
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 [196]:
```
grid_search.best_score_
```

Out[196]:
```
0.9943840256730254
```

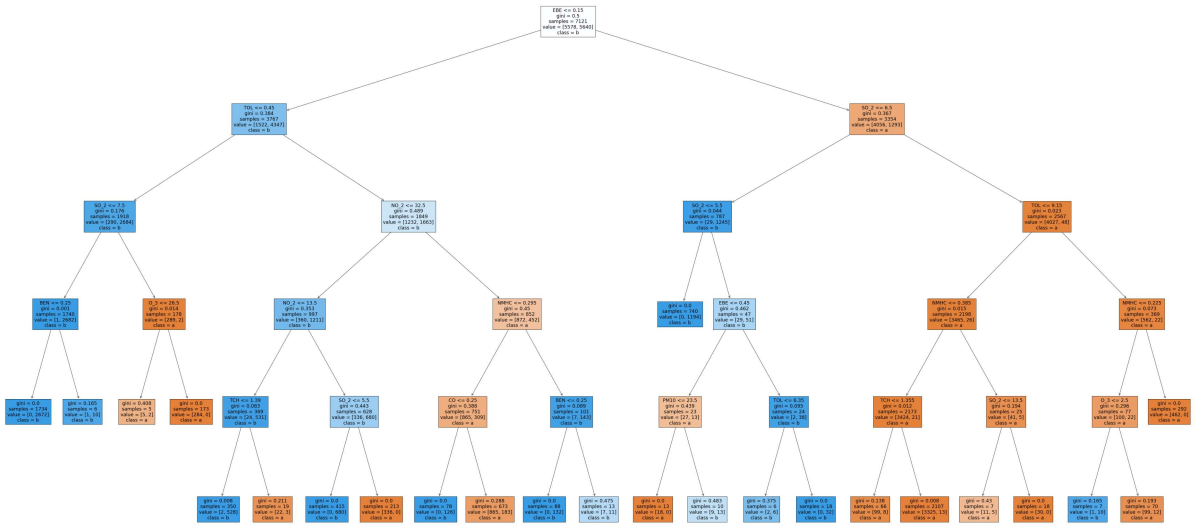
In [197]:
```
rfc_best=grid_search.best_estimator_
```

In [198]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b',
```

Out[198]:
```
[Text(2117.8636363636365, 1993.2, 'EBE <= 0.15\ngini = 0.5\nsamples = 7121\nv
alue = [5578, 5640]\nclass = b'),
 Text(963.8181818181818, 1630.8000000000002, 'TOL <= 0.45\ngini = 0.384\nsamp
les = 3767\nvalue = [1522, 4347]\nclass = b'),
 Text(405.8181818181818, 1268.4, 'SO_2 <= 7.5\ngini = 0.176\nsamples = 1918\n
value = [290, 2684]\nclass = b'),
 Text(202.9090909090909, 906.0, 'BEN <= 0.25\ngini = 0.001\nsamples = 1740\nv
alue = [1, 2682]\nclass = b'),
 Text(101.45454545454545, 543.5999999999999, 'gini = 0.0\nsamples = 1734\nval
ue = [0, 2672]\nclass = b'),
 Text(304.3636363636364, 543.5999999999999, 'gini = 0.165\nsamples = 6\nvalue
= [1, 10]\nclass = b'),
 Text(608.7272727272727, 906.0, 'O_3 <= 26.5\ngini = 0.014\nsamples = 178\nva
lue = [289, 2]\nclass = a'),
 Text(507.27272727272725, 543.5999999999999, 'gini = 0.408\nsamples = 5\nvalu
e = [5, 2]\nclass = a'),
 Text(710.1818181818181, 543.5999999999999, 'gini = 0.0\nsamples = 173\nvalue
= [284, 0]\nclass = a'),
 Text(1521.8181818181818, 1268.4, 'NO_2 <= 32.5\ngini = 0.489\nsamples = 1849
\nvalue = [1232, 1663]\nclass = b'),
 Text(1116.0, 906.0, 'NO_2 <= 13.5\ngini = 0.353\nsamples = 997\nvalue = [36
0, 1211]\nclass = b'),
 Text(913.0909090909091, 543.5999999999999, 'TCH <= 1.39\ngini = 0.083\nsampl
es = 369\nvalue = [24, 531]\nclass = b'),
 Text(811.6363636363636, 181.19999999999982, 'gini = 0.008\nsamples = 350\nva
lue = [2, 528]\nclass = b'),
 Text(1014.5454545454545, 181.19999999999982, 'gini = 0.211\nsamples = 19\nva
lue = [22, 3]\nclass = a'),
 Text(1318.909090909091, 543.5999999999999, 'SO_2 <= 5.5\ngini = 0.443\nsampl
es = 628\nvalue = [336, 680]\nclass = b'),
 Text(1217.4545454545455, 181.19999999999982, 'gini = 0.0\nsamples = 415\nval
ue = [0, 680]\nclass = b'),
 Text(1420.3636363636363, 181.19999999999982, 'gini = 0.0\nsamples = 213\nval
ue = [336, 0]\nclass = a'),
 Text(1927.6363636363635, 906.0, 'NMHC <= 0.295\ngini = 0.45\nsamples = 852\n
value = [872, 452]\nclass = a'),
 Text(1724.7272727272727, 543.5999999999999, 'CO <= 0.25\ngini = 0.388\nsampl
es = 751\nvalue = [865, 309]\nclass = a'),
 Text(1623.2727272727273, 181.19999999999982, 'gini = 0.0\nsamples = 78\nvalu
e = [0, 126]\nclass = b'),
 Text(1826.1818181818182, 181.19999999999982, 'gini = 0.288\nsamples = 673\nv
alue = [865, 183]\nclass = a'),
 Text(2130.5454545454545, 543.5999999999999, 'BEN <= 0.25\ngini = 0.089\nsamp
les = 101\nvalue = [7, 143]\nclass = b'),
 Text(2029.090909090909, 181.19999999999982, 'gini = 0.0\nsamples = 88\nvalue
= [0, 132]\nclass = b'),
 Text(2232.0, 181.19999999999982, 'gini = 0.475\nsamples = 13\nvalue = [7, 1
1]\nclass = b'),
 Text(3271.909090909091, 1630.8000000000002, 'SO_2 <= 6.5\ngini = 0.367\nsamp
les = 3354\nvalue = [4056, 1293]\nclass = a'),
 Text(2637.818181818182, 1268.4, 'SO_2 <= 5.5\ngini = 0.044\nsamples = 787\nv
alue = [29, 1245]\nclass = b'),
 Text(2536.3636363636365, 906.0, 'gini = 0.0\nsamples = 740\nvalue = [0, 119
4]\nclass = b'),
 Text(2739.272727272727, 906.0, 'EBE <= 0.45\ngini = 0.462\nsamples = 47\nval
ue = [29, 51]\nclass = b'),
 Text(2536.3636363636365, 543.5999999999999, 'PM10 <= 23.5\ngini = 0.439\nsam
```

```
ples = 23\nvalue = [27, 13]\nclass = a'),
 Text(2434.909090909091, 181.19999999999982, 'gini = 0.0\nsamples = 13\nvalue
= [18, 0]\nclass = a'),
 Text(2637.818181818182, 181.19999999999982, 'gini = 0.483\nsamples = 10\nval
ue = [9, 13]\nclass = b'),
 Text(2942.181818181818, 543.5999999999999, 'TOL <= 6.35\ngini = 0.095\nsampl
es = 24\nvalue = [2, 38]\nclass = b'),
 Text(2840.7272727272725, 181.19999999999982, 'gini = 0.375\nsamples = 6\nval
ue = [2, 6]\nclass = b'),
 Text(3043.6363636363635, 181.19999999999982, 'gini = 0.0\nsamples = 18\nvalu
e = [0, 32]\nclass = b'),
 Text(3906.0, 1268.4, 'TOL <= 9.15\ngini = 0.023\nsamples = 2567\nvalue = [40
27, 48]\nclass = a'),
 Text(3550.909090909091, 906.0, 'NMHC <= 0.385\ngini = 0.015\nsamples = 2198
\nvalue = [3465, 26]\nclass = a'),
 Text(3348.0, 543.5999999999999, 'TCH <= 1.355\ngini = 0.012\nsamples = 2173
\nvalue = [3424, 21]\nclass = a'),
 Text(3246.5454545454545, 181.19999999999982, 'gini = 0.138\nsamples = 66\nva
lue = [99, 8]\nclass = a'),
 Text(3449.4545454545455, 181.19999999999982, 'gini = 0.008\nsamples = 2107\n
value = [3325, 13]\nclass = a'),
 Text(3753.818181818182, 543.5999999999999, 'SO_2 <= 13.5\ngini = 0.194\nsamp
les = 25\nvalue = [41, 5]\nclass = a'),
 Text(3652.3636363636365, 181.19999999999982, 'gini = 0.43\nsamples = 7\nvalu
e = [11, 5]\nclass = a'),
 Text(3855.272727272727, 181.19999999999982, 'gini = 0.0\nsamples = 18\nvalue
= [30, 0]\nclass = a'),
 Text(4261.090909090909, 906.0, 'NMHC <= 0.225\ngini = 0.073\nsamples = 369\n
value = [562, 22]\nclass = a'),
 Text(4159.636363636364, 543.5999999999999, 'O_3 <= 2.5\ngini = 0.296\nsample
s = 77\nvalue = [100, 22]\nclass = a'),
 Text(4058.181818181818, 181.19999999999982, 'gini = 0.165\nsamples = 7\nvalu
e = [1, 10]\nclass = b'),
 Text(4261.090909090909, 181.19999999999982, 'gini = 0.193\nsamples = 70\nval
ue = [99, 12]\nclass = a'),
 Text(4362.545454545454, 543.5999999999999, 'gini = 0.0\nsamples = 292\nvalue
= [462, 0]\nclass = a')]
```



Conclusion

Accuracy

In [199]:
```
lr.score(x_train,y_train)
```

Out[199]: 0.8116323469799842

In [200]:
```
rr.score(x_train,y_train)
```

Out[200]: 0.809565586347635

In [201]:
```
la.score(x_train,y_train)
```

Out[201]: 0.6359313105656521

In [202]:
```
en.score(x_test,y_test)
```

Out[202]: 0.730433930762691

In [203]:
```
logr.score(fs,target_vector)
```

Out[203]: 0.9947585174092101

In [204]:
```
grid_search.best_score_
```

Out[204]: 0.9943840256730254

Logistic Regression is the suitable for this dataset