Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing Datasets

In [2]:
```
df=pd.read_csv(r"E:\154\C10_air\csvs_per_year\csvs_per_year\madrid_2008.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
0	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000	16.88
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999	19.04
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000	20.27
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001	10.85
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.16
...	...	...	...	...	...	...	...	...	...	...	
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002	5.45
226388	2008-11-01 00:00:00	NaN	0.30	NaN	NaN	NaN	41.880001	48.500000	NaN	35.830002	15.02
226389	2008-11-01 00:00:00	0.25	NaN	0.56	NaN	0.11	83.610001	102.199997	NaN	14.130000	17.54
226390	2008-11-01 00:00:00	0.54	NaN	2.70	NaN	0.18	70.639999	81.860001	NaN	NaN	11.91
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000	12.69

226392 rows × 17 columns

Data Cleaning and Data Preprocessing

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: 25631 entries, 4 to 226391
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     25631 non-null  object 
 1   BEN      25631 non-null  float64
 2   CO       25631 non-null  float64
 3   EBE      25631 non-null  float64
 4   MXY      25631 non-null  float64
 5   NMHC     25631 non-null  float64
 6   NO_2     25631 non-null  float64
 7   NOx      25631 non-null  float64
 8   OXY      25631 non-null  float64
 9   O_3      25631 non-null  float64
 10  PM10     25631 non-null  float64
 11  PM25     25631 non-null  float64
 12  PXY      25631 non-null  float64
 13  SO_2     25631 non-null  float64
 14  TCH      25631 non-null  float64
 15  TOL      25631 non-null  float64
 16  station  25631 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.5+ MB
```

In [6]:
```
data=df[['CO' ,'station']]
data
```

Out[6]:

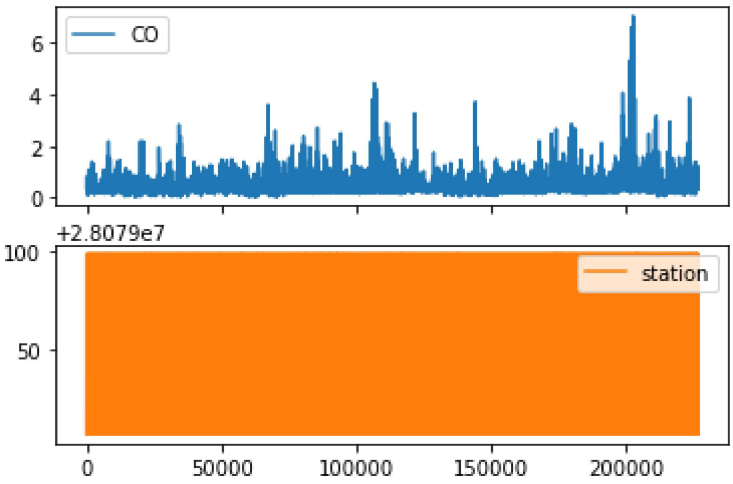
	CO	station
4	0.80	28079006
21	0.37	28079024
25	0.39	28079099
30	0.51	28079006
47	0.39	28079024
...	...	...
226362	0.35	28079024
226366	0.46	28079099
226371	0.53	28079006
226387	0.30	28079024
226391	0.36	28079099

25631 rows × 2 columns

Line chart

In [7]:
```
data.plot.line(subplots=True)
```

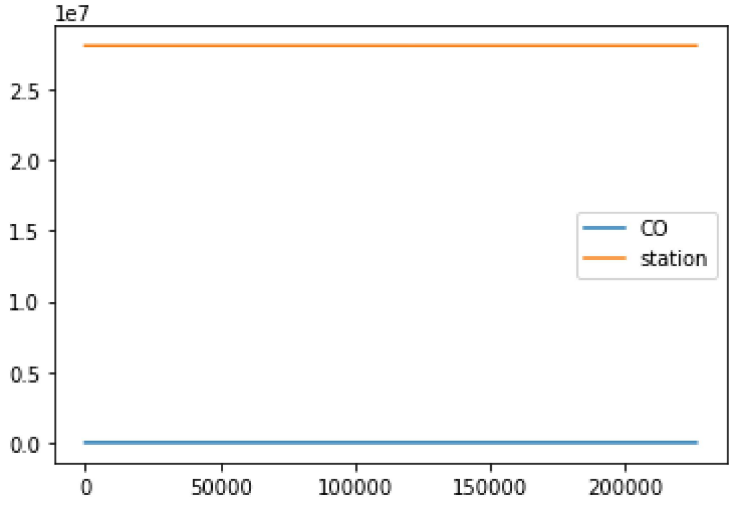
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [8]:
```
data.plot.line()
```

Out[8]: <AxesSubplot:>

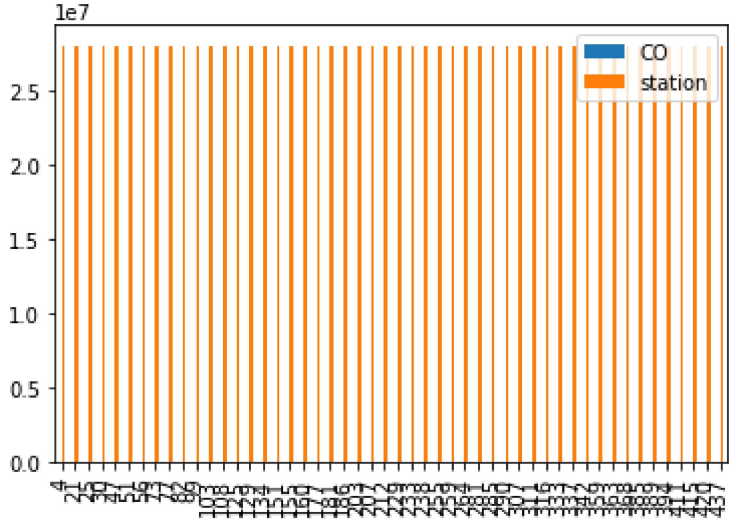


Bar chart

In [9]:
```
b=data[0:50]
```

In [10]:
```
b.plot.bar()
```

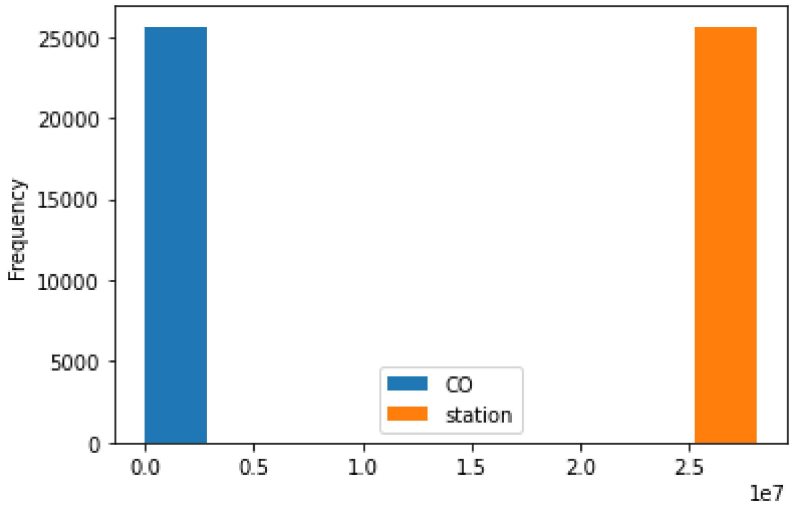
Out[10]: <AxesSubplot:>



Histogram

In [11]:
```
data.plot.hist()
```

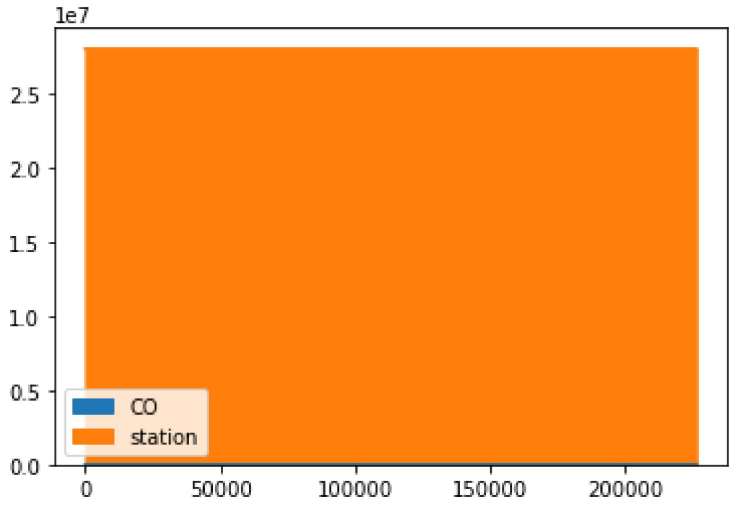
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [12]:
```
data.plot.area()
```

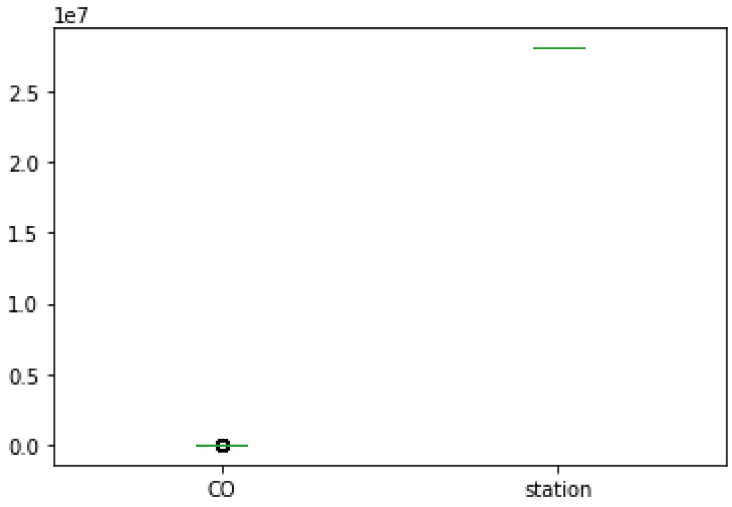
Out[12]: <AxesSubplot:>



Box chart

In [13]:
```
data.plot.box()
```

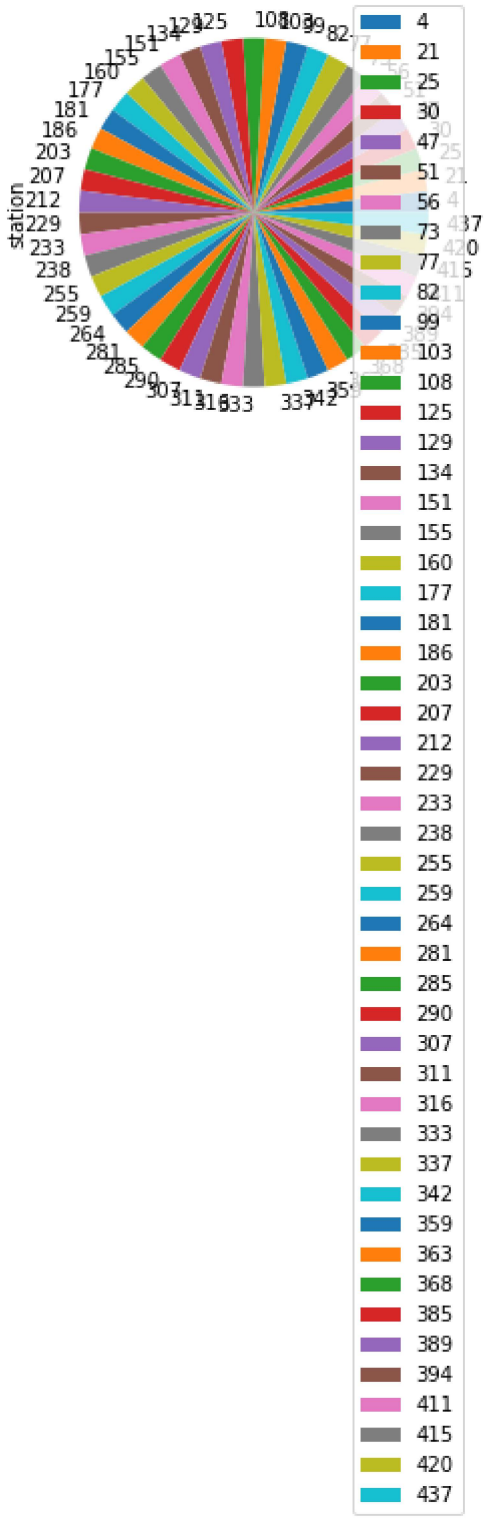
Out[13]: <AxesSubplot:>



Pie chart

In [14]:
```
b.plot.pie(y='station' )
```

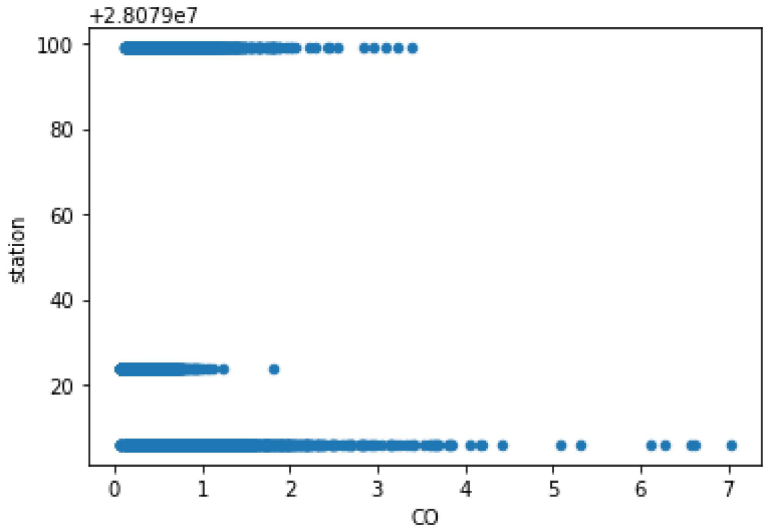
Out[14]: <AxesSubplot:ylabel='station'>



Scatter chart

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: 25631 entries, 4 to 226391
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     25631 non-null  object 
 1   BEN      25631 non-null  float64
 2   CO       25631 non-null  float64
 3   EBE      25631 non-null  float64
 4   MXY      25631 non-null  float64
 5   NMHC     25631 non-null  float64
 6   NO_2     25631 non-null  float64
 7   NOx      25631 non-null  float64
 8   OXY      25631 non-null  float64
 9   O_3      25631 non-null  float64
 10  PM10     25631 non-null  float64
 11  PM25     25631 non-null  float64
 12  PXY      25631 non-null  float64
 13  SO_2     25631 non-null  float64
```

In [17]:
```
df.describe()
```

Out[17]:

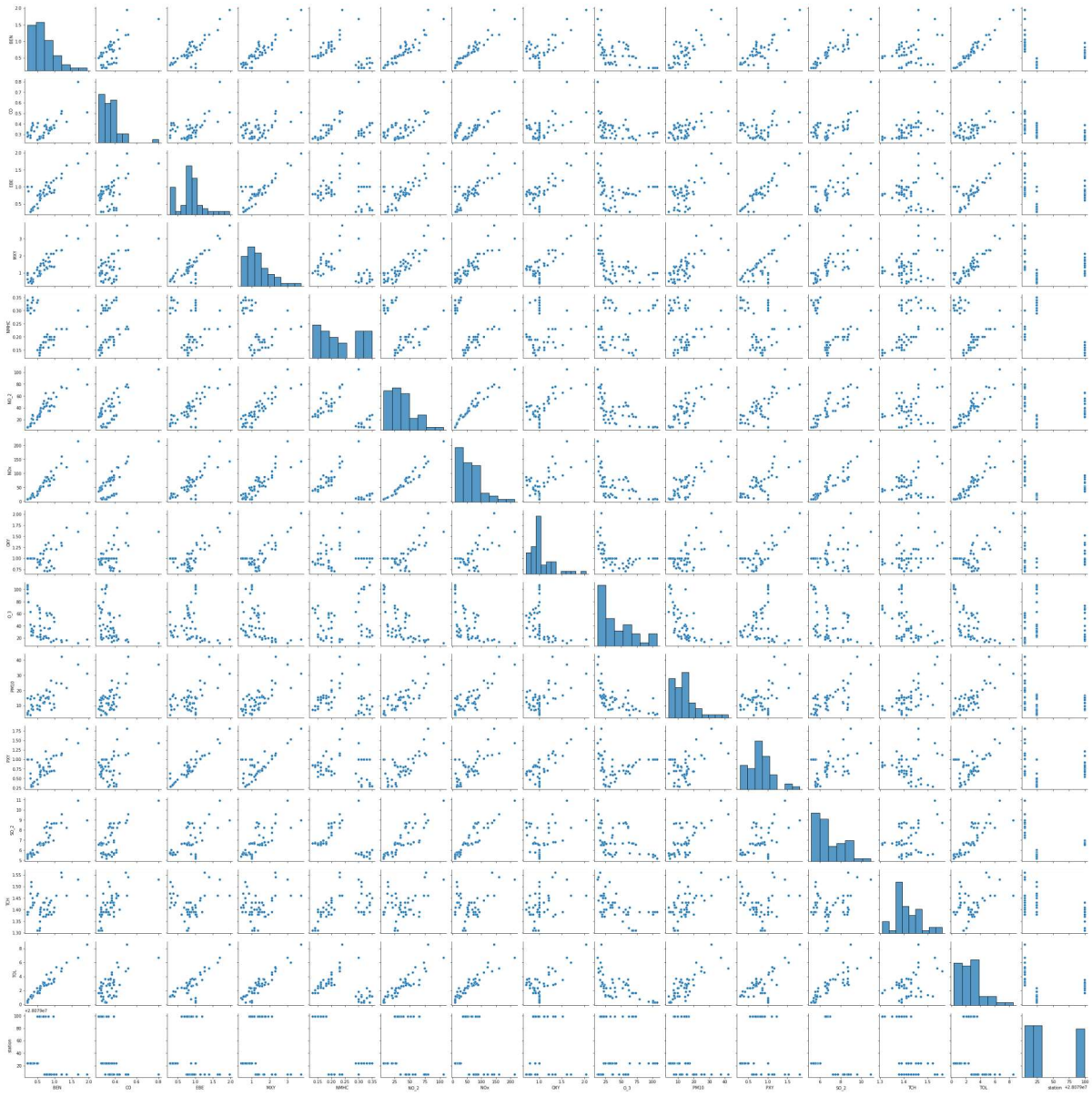
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	25631.000000	25631.000000	25631.000000	25631.000000	25631.000000	25631.000000	256
mean	1.090541	0.440632	1.352355	2.446045	0.213323	54.225261	
std	1.146461	0.317853	1.118191	2.390023	0.123409	38.164647	1
min	0.100000	0.060000	0.170000	0.240000	0.000000	0.240000	
25%	0.430000	0.260000	0.740000	1.000000	0.130000	25.719999	
50%	0.750000	0.350000	1.000000	1.620000	0.190000	48.000000	
75%	1.320000	0.510000	1.580000	3.105000	0.270000	74.924999	1
max	27.230000	7.030000	26.740000	55.889999	1.760000	554.900024	20

In [18]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [19]:
```
sns.pairplot(df1[0:50])
```

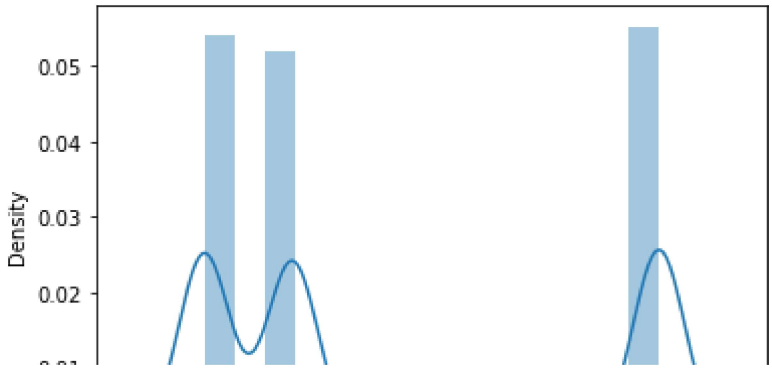
Out[19]: <seaborn.axisgrid.PairGrid at 0x22b5636c790>



In [20]:
```
sns.distplot(df1['station'])
```

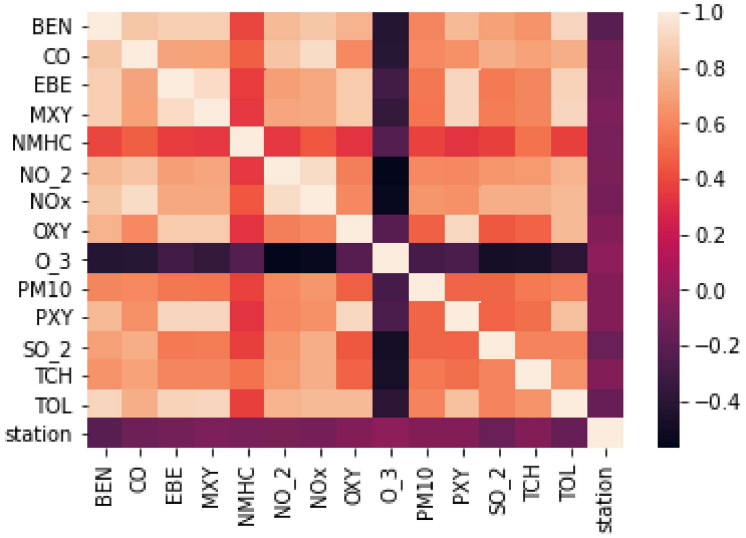
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio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

Out[20]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [21]:
```
sns.heatmap(df1.corr())
```

Out[21]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

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)
```

Linear Regression

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]:
```
28079032.17408309
```

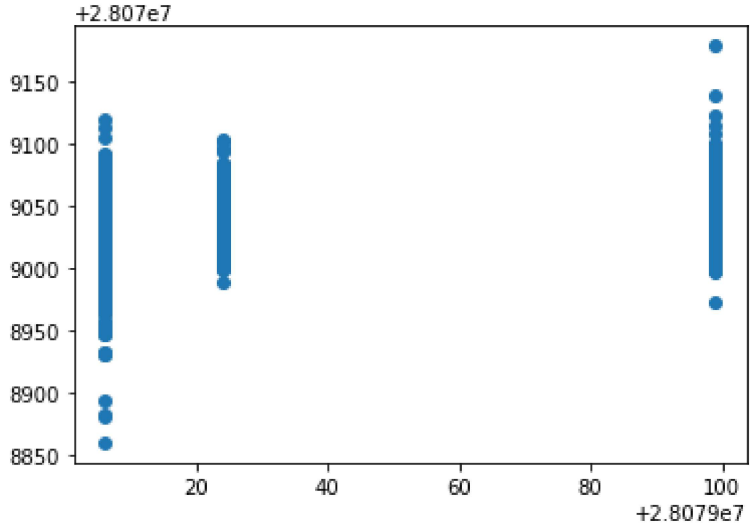
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-24.386911
CO	-1.950706
EBE	-0.009643
MXY	7.288423
NMHC	-27.978200
NO_2	-0.018595
NOx	0.120010
OXY	4.272955
O_3	-0.140344
PM10	0.132375
PXY	1.670614
SO_2	-0.610132
TCH	19.753147
TOL	-2.201672

In [27]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[27]: <matplotlib.collections.PathCollection at 0x22b64d233d0>



ACCURACY

In [28]:
```
lr.score(x_test,y_test)
```

Out[28]: 0.15016532031810825

In [29]:
```
lr.score(x_train,y_train)
```

Out[29]: 0.1402895068053709

Ridge and Lasso

In [30]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [31]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[31]: Ridge(alpha=10)

Accuracy(Ridge)

In [32]:
```
rr.score(x_test,y_test)
```

Out[32]: 0.15019122409133046

In [33]:
```
rr.score(x_train,y_train)
```

Out[33]:
```
0.14026212970550966
```

In [34]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[34]:
```
Lasso(alpha=10)
```

Accuracy(Lasso)

In [35]:
```
la.score(x_train,y_train)
```

Out[35]:
```
0.04246575606939873
```

In [36]:
```
la.score(x_test,y_test)
```

Out[36]:
```
0.04086515932538837
```

ElasticNet

In [37]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[37]:
```
ElasticNet()
```

In [38]:
```
en.coef_
```

Out[38]:
```
array([-4.33338231, -0.        ,  0.        ,  3.29726493, -0.        ,
        0.07539493,  0.02249436,  1.66568737, -0.15395961,  0.13218482,
        1.58890585, -0.95025962,  0.        , -2.71429632])
```

In [39]:
```
en.intercept_
```

Out[39]:
```
28079057.05604101
```

In [40]:
```
prediction=en.predict(x_test)
```

In [41]:
```
en.score(x_test,y_test)
```

Out[41]:
```
0.09221243307974003
```

Evaluation Metrics

In [42]:
```
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)))
```

```
35.72433405711675
1489.77399472722
38.597590530073504
```

Logistic Regression

In [43]:
```
from sklearn.linear_model import LogisticRegression
```

In [44]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [45]:
```
feature_matrix.shape
```

Out[45]:
```
(25631, 14)
```

In [46]:
```
target_vector.shape
```

Out[46]:
```
(25631,)
```

In [47]:
```
from sklearn.preprocessing import StandardScaler
```

In [48]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [49]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[49]:
```
LogisticRegression(max_iter=10000)
```

In [50]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [51]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079099]
```

In [52]:
```
logr.classes_
```

Out[52]:
```
array([28079006, 28079024, 28079099], dtype=int64)
```

In [53]:
```
logr.score(fs,target_vector)
```

Out[53]:
```
0.794194530061254
```

In [54]:
```
logr.predict_proba(observation)[0][0]
```

Out[54]:
```
8.321803242555043e-09
```

In [55]:
```
logr.predict_proba(observation)
```

Out[55]:
```
array([[8.32180324e-09, 1.19114634e-13, 9.99999992e-01]])
```

Random Forest

In [56]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [57]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[57]:
```
RandomForestClassifier()
```

In [58]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [59]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="ac
grid_search.fit(x_train,y_train)
```

Out[59]:
```
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 [60]:
```
grid_search.best_score_
```

Out[60]:
```
0.8490051369537444
```

In [61]:
```
rfc_best=grid_search.best_estimator_
```

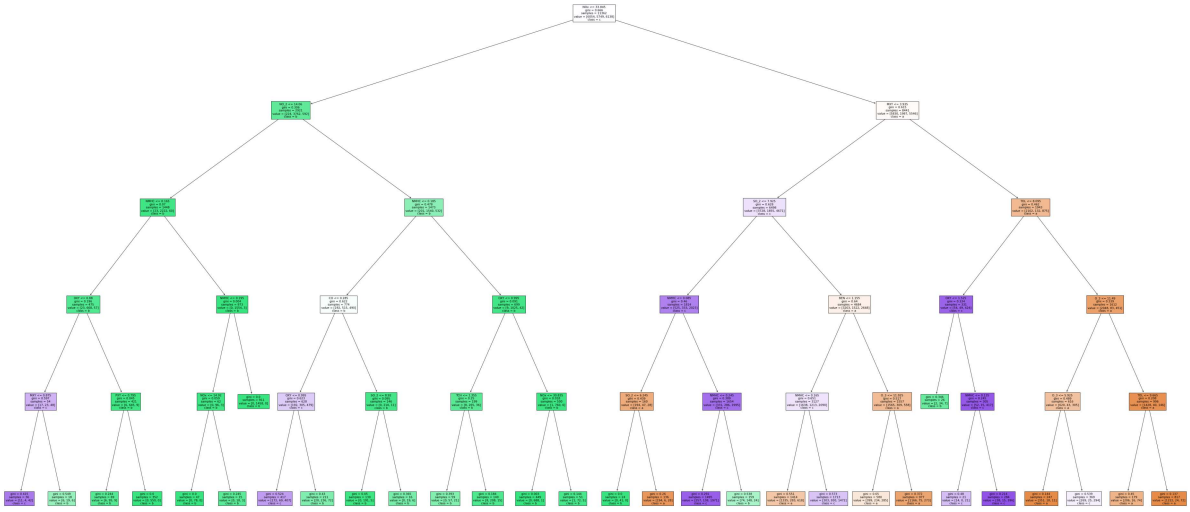
In [62]:
```
from sklearn.tree import plot_tree

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

Out[62]:
```
[Text(2232.0, 1993.2, 'NOx <= 33.845\ngini = 0.666\nsamples = 11362\nvalue =
[6054, 5749, 6138]\nclass = c'),
 Text(1096.0714285714284, 1630.8000000000002, 'NO_2 <= 14.06\ngini = 0.306\ns
amples = 2921\nvalue = [224, 3762, 592]\nclass = b'),
 Text(597.8571428571428, 1268.4, 'NMHC <= 0.165\ngini = 0.07\nsamples = 1448
\nvalue = [23, 2222, 60]\nclass = b'),
 Text(318.85714285714283, 906.0, 'OXY <= 0.88\ngini = 0.196\nsamples = 475\nv
alue = [23, 668, 57]\nclass = b'),
 Text(159.42857142857142, 543.5999999999999, 'MXY <= 0.875\ngini = 0.597\nsam
ples = 54\nvalue = [17, 23, 48]\nclass = c'),
 Text(79.71428571428571, 181.19999999999982, 'gini = 0.415\nsamples = 36\nval
ue = [11, 4, 42]\nclass = c'),
 Text(239.1428571428571, 181.19999999999982, 'gini = 0.549\nsamples = 18\nval
ue = [6, 19, 6]\nclass = b'),
 Text(478.2857142857142, 543.5999999999999, 'PXY <= 0.795\ngini = 0.045\nsamp
les = 421\nvalue = [6, 645, 9]\nclass = b'),
 Text(398.57142857142856, 181.19999999999982, 'gini = 0.244\nsamples = 69\nva
lue = [6, 95, 9]\nclass = b'),
 Text(558.0, 181.19999999999982, 'gini = 0.0\nsamples = 352\nvalue = [0, 550,
0]\nclass = b'),
 Text(876.8571428571428, 906.0, 'NMHC <= 0.195\ngini = 0.004\nsamples = 973\n
value = [0, 1554, 3]\nclass = b'),
 Text(797.1428571428571, 543.5999999999999, 'NOx <= 14.92\ngini = 0.059\nsamp
les = 62\nvalue = [0, 96, 3]\nclass = b'),
 Text(717.4285714285713, 181.19999999999982, 'gini = 0.0\nsamples = 47\nvalue
= [0, 78, 0]\nclass = b'),
 Text(876.8571428571428, 181.19999999999982, 'gini = 0.245\nsamples = 15\nval
ue = [0, 18, 3]\nclass = b'),
 Text(956.5714285714284, 543.5999999999999, 'gini = 0.0\nsamples = 911\nvalue
= [0, 1458, 0]\nclass = b'),
 Text(1594.2857142857142, 1268.4, 'NMHC <= 0.185\ngini = 0.478\nsamples = 147
3\nvalue = [201, 1540, 532]\nclass = b'),
 Text(1275.4285714285713, 906.0, 'CO <= 0.285\ngini = 0.622\nsamples = 774\nv
alue = [192, 515, 490]\nclass = b'),
 Text(1116.0, 543.5999999999999, 'OXY <= 0.995\ngini = 0.623\nsamples = 628\n
value = [192, 305, 479]\nclass = c'),
 Text(1036.2857142857142, 181.19999999999982, 'gini = 0.524\nsamples = 417\nv
alue = [172, 69, 407]\nclass = c'),
 Text(1195.7142857142856, 181.19999999999982, 'gini = 0.43\nsamples = 211\nva
lue = [20, 236, 72]\nclass = b'),
 Text(1434.8571428571427, 543.5999999999999, 'SO_2 <= 8.92\ngini = 0.095\nsam
ples = 146\nvalue = [0, 210, 11]\nclass = b'),
 Text(1355.142857142857, 181.19999999999982, 'gini = 0.05\nsamples = 130\nval
ue = [0, 191, 5]\nclass = b'),
 Text(1514.5714285714284, 181.19999999999982, 'gini = 0.365\nsamples = 16\nva
lue = [0, 19, 6]\nclass = b'),
 Text(1913.1428571428569, 906.0, 'OXY <= 0.995\ngini = 0.091\nsamples = 699\n
value = [9, 1025, 42]\nclass = b'),
 Text(1753.7142857142856, 543.5999999999999, 'TCH <= 1.355\ngini = 0.25\nsamp
les = 199\nvalue = [8, 265, 36]\nclass = b'),
 Text(1673.9999999999998, 181.19999999999982, 'gini = 0.393\nsamples = 59\nva
lue = [0, 57, 21]\nclass = b'),
 Text(1833.4285714285713, 181.19999999999982, 'gini = 0.184\nsamples = 140\nv
alue = [8, 208, 15]\nclass = b'),
 Text(2072.5714285714284, 543.5999999999999, 'NOx <= 30.835\ngini = 0.018\nsa
mples = 500\nvalue = [1, 760, 6]\nclass = b'),
 Text(1992.8571428571427, 181.19999999999982, 'gini = 0.003\nsamples = 449\nv
```

```
alue = [0, 688, 1]\nclass = b'),
 Text(2152.285714285714, 181.19999999999982, 'gini = 0.144\nsamples = 51\nval
ue = [1, 72, 5]\nclass = b'),
 Text(3367.928571428571, 1630.8000000000002, 'MXY <= 3.935\ngini = 0.615\nsam
ples = 8441\nvalue = [5830, 1987, 5546]\nclass = a'),
 Text(2869.7142857142853, 1268.4, 'SO_2 <= 7.925\ngini = 0.628\nsamples = 649
8\nvalue = [3728, 1855, 4671]\nclass = c'),
 Text(2550.8571428571427, 906.0, 'NMHC <= 0.085\ngini = 0.46\nsamples = 1814
\nvalue = [525, 333, 2023]\nclass = c'),
 Text(2391.428571428571, 543.5999999999999, 'SO_2 <= 6.245\ngini = 0.439\nsam
ples = 160\nvalue = [194, 47, 28]\nclass = a'),
 Text(2311.7142857142853, 181.19999999999982, 'gini = 0.0\nsamples = 24\nvalu
e = [0, 41, 0]\nclass = b'),
 Text(2471.142857142857, 181.19999999999982, 'gini = 0.26\nsamples = 136\nval
ue = [194, 6, 28]\nclass = a'),
 Text(2710.285714285714, 543.5999999999999, 'NMHC <= 0.245\ngini = 0.389\nsam
ples = 1654\nvalue = [331, 286, 1995]\nclass = c'),
 Text(2630.5714285714284, 181.19999999999982, 'gini = 0.291\nsamples = 1495\n
value = [257, 138, 1971]\nclass = c'),
 Text(2790.0, 181.19999999999982, 'gini = 0.538\nsamples = 159\nvalue = [74,
148, 24]\nclass = b'),
 Text(3188.5714285714284, 906.0, 'BEN <= 1.155\ngini = 0.64\nsamples = 4684\n
value = [3203, 1522, 2648]\nclass = a'),
 Text(3029.142857142857, 543.5999999999999, 'NMHC <= 0.165\ngini = 0.651\nsam
ples = 3127\nvalue = [1638, 1213, 2090]\nclass = c'),
 Text(2949.428571428571, 181.19999999999982, 'gini = 0.551\nsamples = 1414\nv
alue = [1335, 283, 618]\nclass = a'),
 Text(3108.8571428571427, 181.19999999999982, 'gini = 0.573\nsamples = 1713\n
value = [303, 930, 1472]\nclass = c'),
 Text(3347.9999999999995, 543.5999999999999, 'O_3 <= 11.925\ngini = 0.517\nsa
mples = 1557\nvalue = [1565, 309, 558]\nclass = a'),
 Text(3268.285714285714, 181.19999999999982, 'gini = 0.65\nsamples = 580\nval
ue = [399, 234, 285]\nclass = a'),
 Text(3427.7142857142853, 181.19999999999982, 'gini = 0.372\nsamples = 977\nv
alue = [1166, 75, 273]\nclass = a'),
 Text(3866.142857142857, 1268.4, 'TOL <= 8.095\ngini = 0.462\nsamples = 1943
\nvalue = [2102, 132, 875]\nclass = a'),
 Text(3587.142857142857, 906.0, 'OXY <= 1.525\ngini = 0.334\nsamples = 331\nv
alue = [54, 49, 424]\nclass = c'),
 Text(3507.428571428571, 543.5999999999999, 'gini = 0.346\nsamples = 26\nvalu
e = [2, 34, 7]\nclass = b'),
 Text(3666.8571428571427, 543.5999999999999, 'NMHC <= 0.135\ngini = 0.245\nsa
mples = 305\nvalue = [52, 15, 417]\nclass = c'),
 Text(3587.142857142857, 181.19999999999982, 'gini = 0.48\nsamples = 21\nvalu
e = [14, 0, 21]\nclass = c'),
 Text(3746.5714285714284, 181.19999999999982, 'gini = 0.214\nsamples = 284\nv
alue = [38, 15, 396]\nclass = c'),
 Text(4145.142857142857, 906.0, 'O_3 <= 11.49\ngini = 0.339\nsamples = 1612\n
value = [2048, 83, 451]\nclass = a'),
 Text(3985.7142857142853, 543.5999999999999, 'O_3 <= 5.925\ngini = 0.489\nsam
ples = 616\nvalue = [620, 43, 305]\nclass = a'),
 Text(3905.9999999999995, 181.19999999999982, 'gini = 0.144\nsamples = 247\nv
alue = [351, 18, 11]\nclass = a'),
 Text(4065.428571428571, 181.19999999999982, 'gini = 0.539\nsamples = 369\nva
lue = [269, 25, 294]\nclass = c'),
 Text(4304.571428571428, 543.5999999999999, 'TOL <= 9.665\ngini = 0.208\nsamp
les = 996\nvalue = [1428, 40, 146]\nclass = a'),
```

```
 Text(4224.857142857142, 181.19999999999982, 'gini = 0.45\nsamples = 179\nval
ue = [206, 16, 74]\nclass = a'),
 Text(4384.285714285714, 181.19999999999982, 'gini = 0.137\nsamples = 817\nva
lue = [1222, 24, 72]\nclass = a')]
```



Accuracy

Linear Regression

In [63]:
```
lr.score(x_train,y_train)
```

Out[63]: 0.1402895068053709

Ridge Regression

In [64]:
```
rr.score(x_train,y_train)
```

Out[64]: 0.14026212970550966

Lasso Regression

In [65]:
```
la.score(x_test,y_test)
```

Out[65]: 0.04086515932538837

ElasticNet Regression

In [66]:
```
en.score(x_test,y_test)
```

Out[66]:
```
0.09221243307974003
```

Logistic Regression

In [67]:
```
logr.score(fs,target_vector)
```

Out[67]:
```
0.794194530061254
```

Random Forest

In [68]:
```
grid_search.best_score_
```

Out[68]:
```
0.8490051369537444
```

Conclusion

Random Forest is suitable for this dataset