Importing Libraries

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

Importing Datasets

In [3]:
```
df=pd.read_csv(r"E:\154\C10_air\csvs_per_year\csvs_per_year\madrid_2001.csv")
df
```

Out[3]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	105.0
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.5
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	100.0
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	69.7
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	75.1
...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000	47.8
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000	26.8
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.7
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	37.8
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.3

217872 rows × 16 columns

Data Cleaning and Data Preprocessing

In [4]:
```
df=df.dropna()
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29669 entries, 1 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     29669 non-null  object 
 1   BEN      29669 non-null  float64
 2   CO       29669 non-null  float64
 3   EBE      29669 non-null  float64
 4   MXY      29669 non-null  float64
 5   NMHC     29669 non-null  float64
 6   NO_2     29669 non-null  float64
 7   NOx      29669 non-null  float64
 8   OXY      29669 non-null  float64
 9   O_3      29669 non-null  float64
 10  PM10     29669 non-null  float64
 11  PXY      29669 non-null  float64
 12  SO_2     29669 non-null  float64
 13  TCH      29669 non-null  float64
 14  TOL      29669 non-null  float64
 15  station  29669 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 3.8+ MB
```

In [7]:
```
data=df[['CO' ,'station']]
data
```

Out[7]:

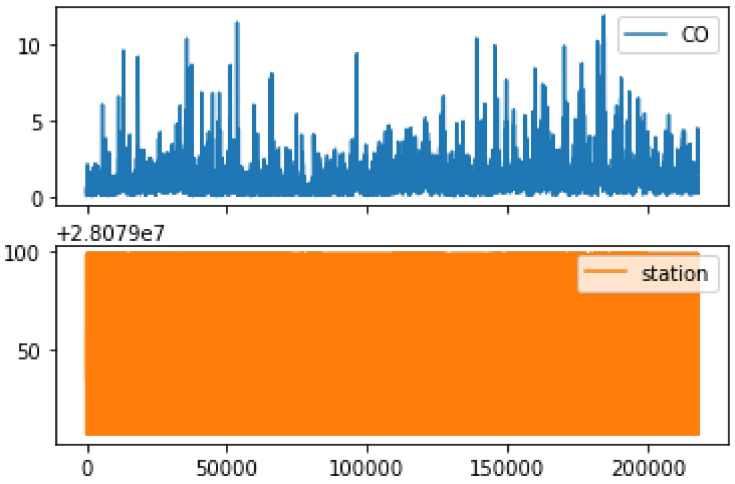
	CO	station
1	0.34	28079035
5	0.63	28079006
21	0.43	28079024
23	0.34	28079099
25	0.06	28079035
...	...	...
217829	4.48	28079006
217847	2.65	28079099
217849	1.22	28079035
217853	1.83	28079006
217871	1.62	28079099

29669 rows × 2 columns

Line chart

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

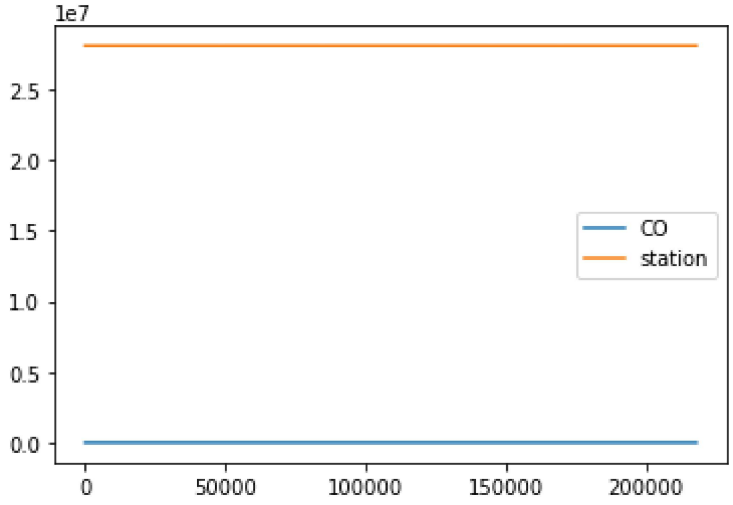
Out[8]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [9]:
```
data.plot.line()
```

Out[9]: <AxesSubplot:>

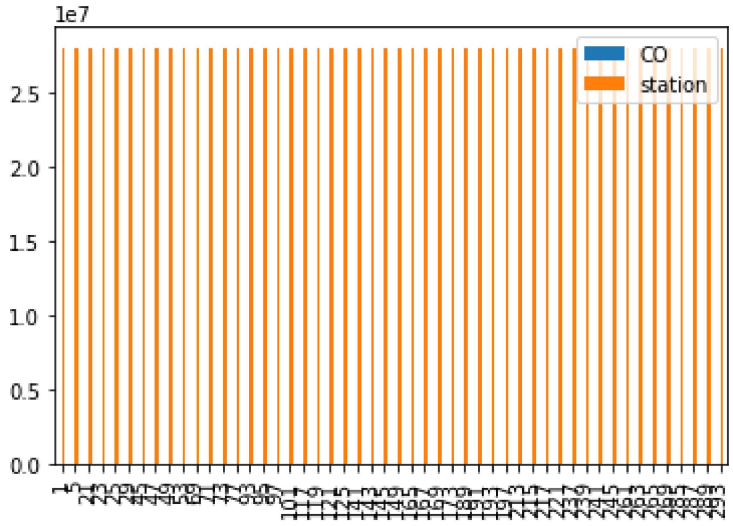


Bar chart

In [10]:
```
b=data[0:50]
```

In [11]:
```
b.plot.bar()
```

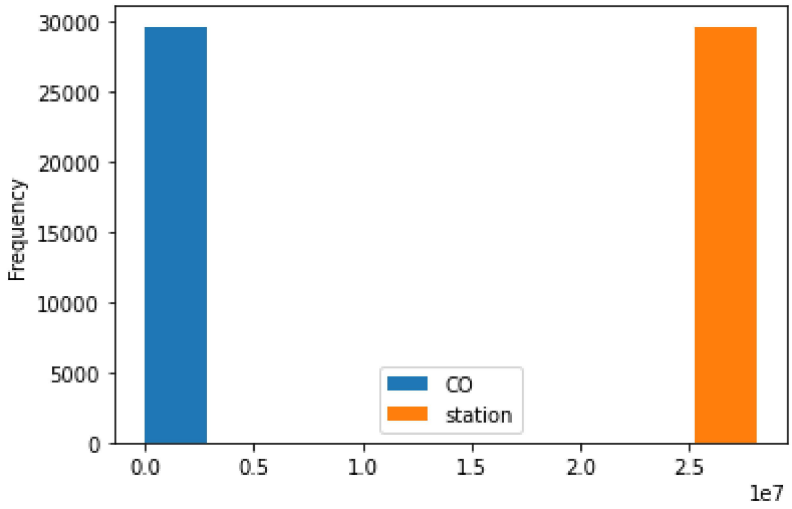
Out[11]: <AxesSubplot:>



Histogram

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

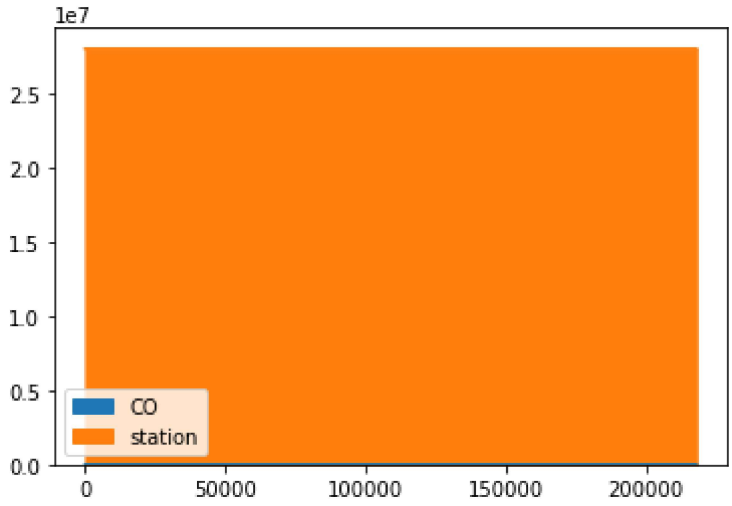
Out[12]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

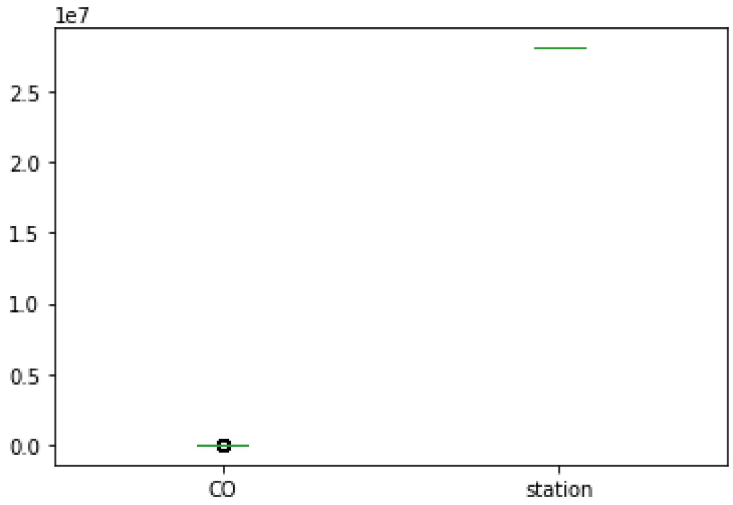
Out[13]: <AxesSubplot:>



Box chart

In [14]:
```
data.plot.box()
```

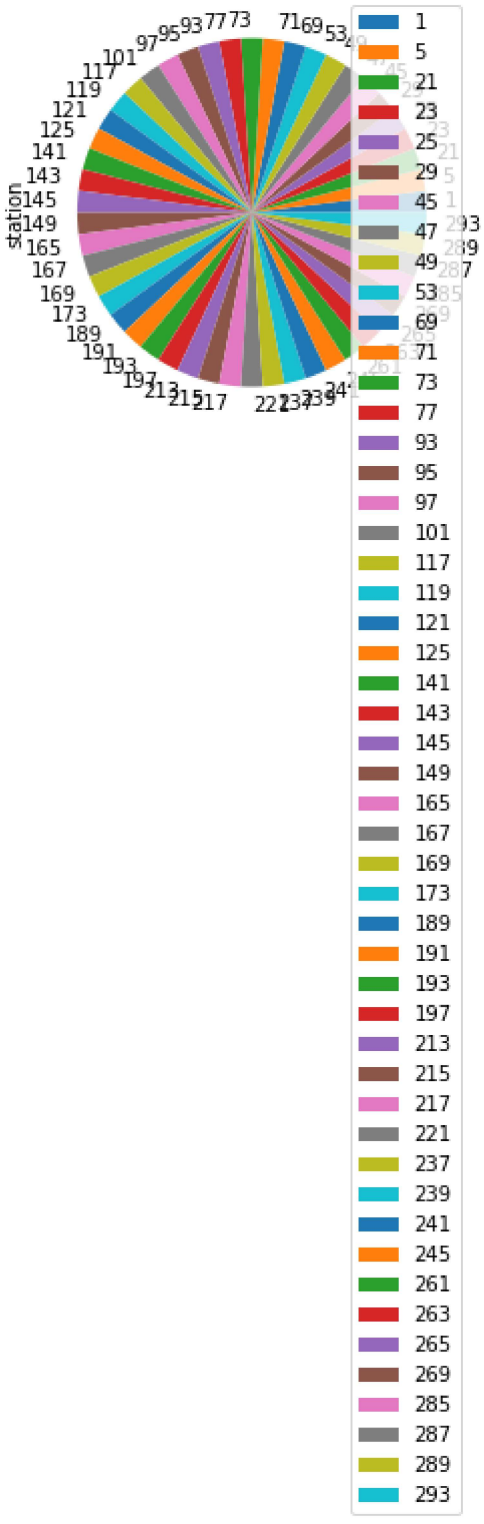
Out[14]: <AxesSubplot:>



Pie chart

In [15]:
```
b.plot.pie(y='station' )
```

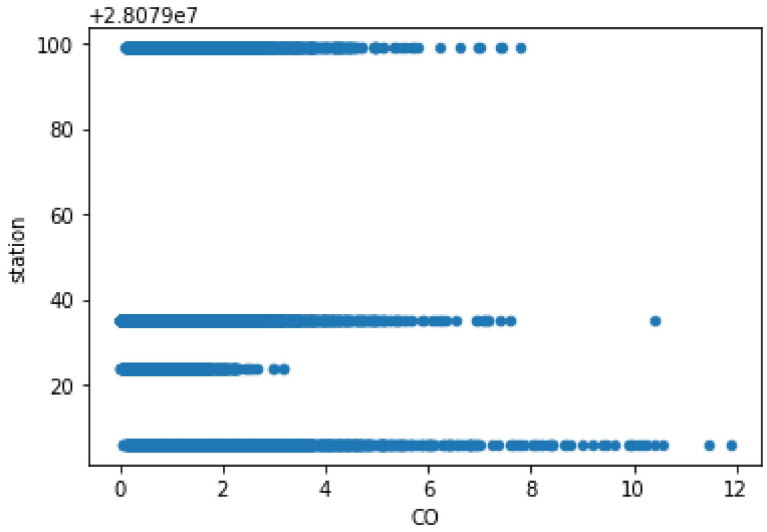
Out[15]: <AxesSubplot:ylabel='station'>



Scatter chart

In [16]:
```
data.plot.scatter(x='CO' ,y='station')
```

Out[16]: <AxesSubplot:xlabel='CO', ylabel='station'>



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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29669 entries, 1 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     29669 non-null  object 
 1   BEN      29669 non-null  float64
 2   CO       29669 non-null  float64
 3   EBE      29669 non-null  float64
 4   MXY      29669 non-null  float64
 5   NMHC     29669 non-null  float64
 6   NO_2     29669 non-null  float64
 7   NOx      29669 non-null  float64
 8   OXY      29669 non-null  float64
 9   O_3      29669 non-null  float64
 10  PM10     29669 non-null  float64
 11  PXY      29669 non-null  float64
 12  SO_2     29669 non-null  float64
 13  TCH      29669 non-null  float64
```

In [18]:
```
df.describe()
```

Out[18]:

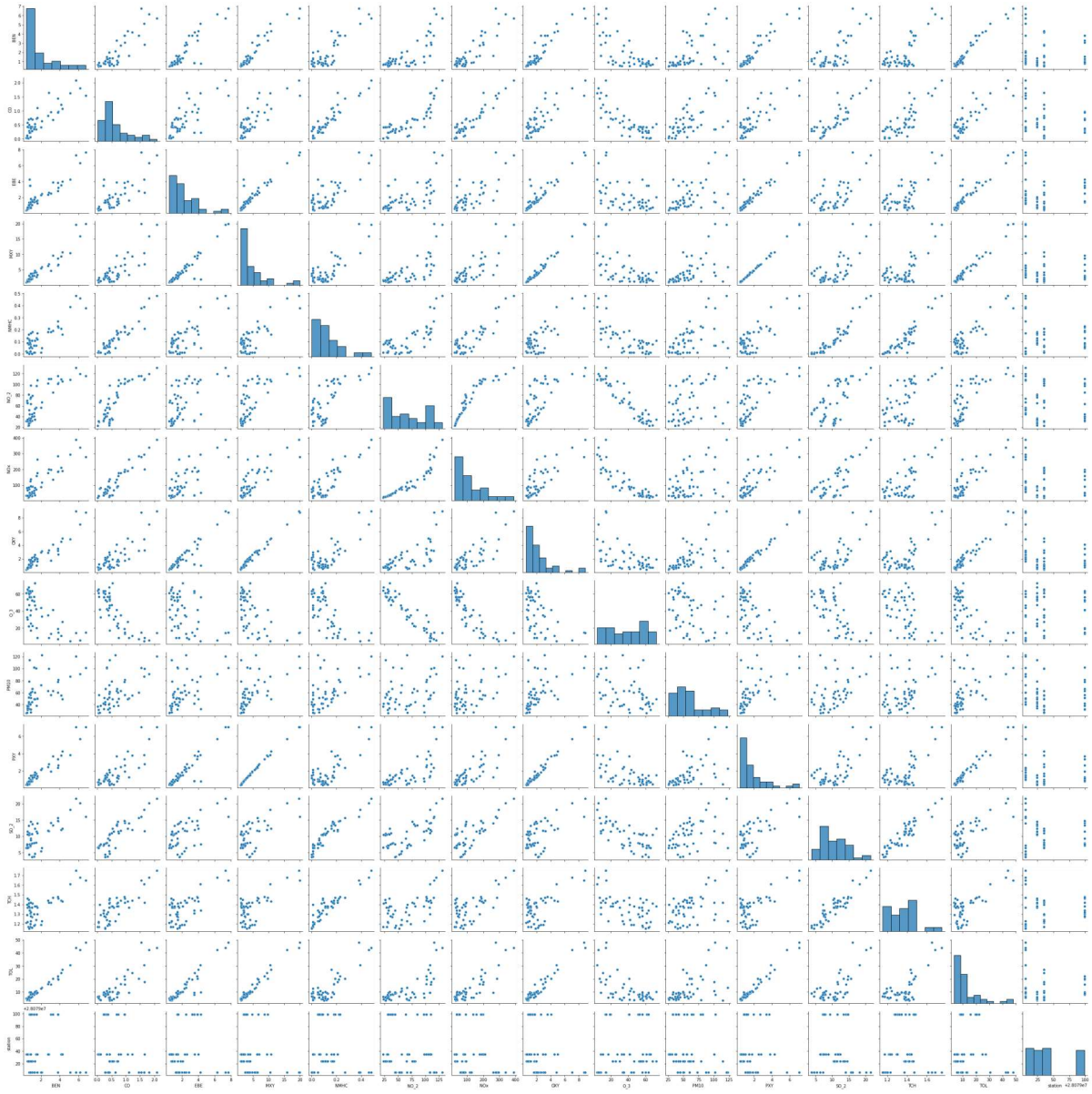
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	29669.000000	29669.000000	29669.000000	29669.000000	29669.000000	29669.000000	296
mean	3.361895	1.005413	3.580229	8.113086	0.195222	67.652292	1
std	3.176669	0.863135	3.744496	7.909701	0.192585	34.003120	1
min	0.100000	0.000000	0.140000	0.210000	0.000000	1.180000	
25%	1.280000	0.470000	1.390000	3.040000	0.080000	44.299999	
50%	2.510000	0.760000	2.600000	5.830000	0.140000	64.449997	1
75%	4.420000	1.270000	4.580000	10.640000	0.250000	86.540001	2
max	54.560001	11.890000	77.260002	150.600006	2.880000	292.700012	19

In [19]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [20]:
```
sns.pairplot(df1[0:50])
```

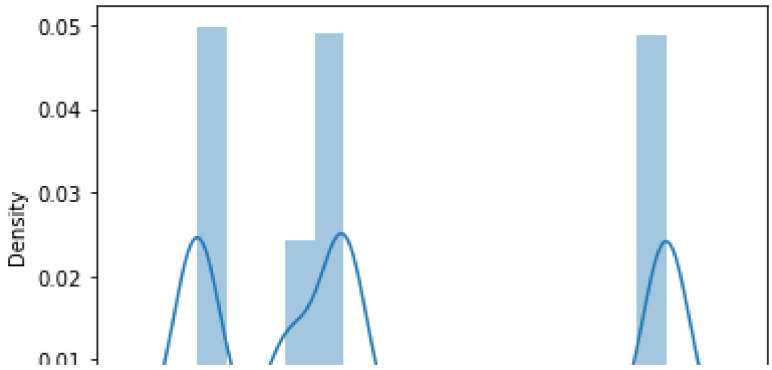
Out[20]:
```
<seaborn.axisgrid.PairGrid at 0x1b991b818e0>
```



In [21]:
```
sns.distplot(df1['station'])
```

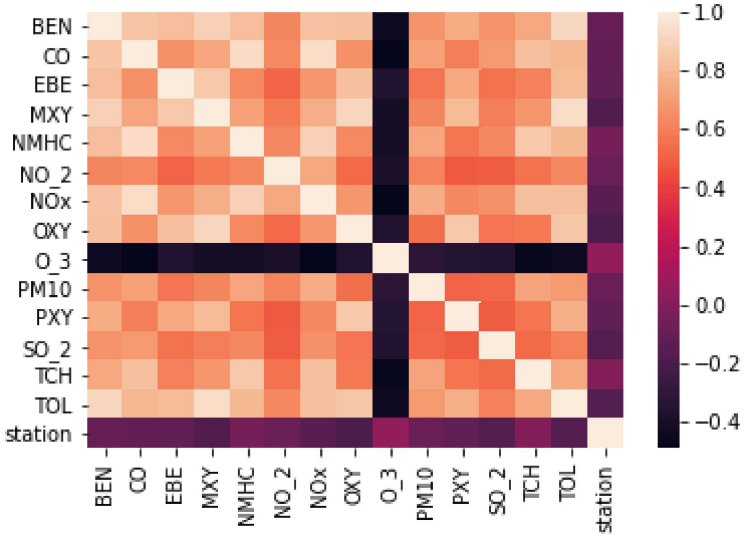
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio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

Out[21]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [22]:
```
sns.heatmap(df1.corr())
```

Out[22]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [23]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [24]:
```
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 [25]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[25]:
```
LinearRegression()
```

In [26]:
```
lr.intercept_
```

Out[26]:
```
28079004.836290125
```

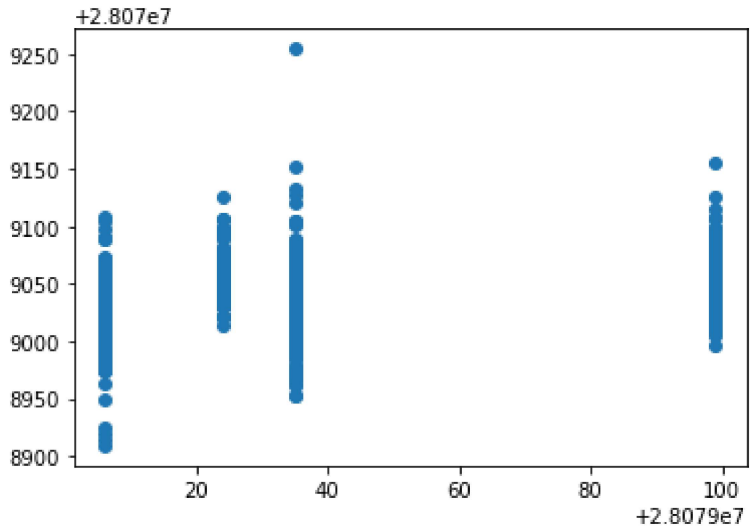
In [27]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[27]:

	Co-efficient
BEN	6.724656
CO	-14.582654
EBE	0.663762
MXY	-0.150799
NMHC	77.599840
NO_2	0.125805
NOx	-0.092027
OXY	-2.795297
O_3	-0.025750
PM10	-0.053761
PXY	1.250145
SO_2	-0.298080
TCH	38.618963
TOL	-1.147030

In [28]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[28]: <matplotlib.collections.PathCollection at 0x1b9a2314be0>



ACCURACY

In [29]:
```
lr.score(x_test,y_test)
```

Out[29]: 0.18043158754129462

In [30]:
```
lr.score(x_train,y_train)
```

Out[30]: 0.15792771685222995

Ridge and Lasso

In [31]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [32]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[32]: Ridge(alpha=10)

Accuracy(Ridge)

In [33]:
```
rr.score(x_test,y_test)
```

Out[33]: 0.17966058804213192

In [34]:
```
rr.score(x_train,y_train)
```

Out[34]:
```
0.15770785489393768
```

In [35]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[35]:
```
Lasso(alpha=10)
```

In [36]:
```
la.score(x_train,y_train)
```

Out[36]:
```
0.038751624051942435
```

Accuracy(Lasso)

In [37]:
```
la.score(x_test,y_test)
```

Out[37]:
```
0.039710554684473154
```

In [38]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[38]:
```
ElasticNet()
```

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

Out[39]:
```
array([ 4.7229001 ,  0.        ,  0.58235622, -0.28692874,  0.03103695,
        0.0680557 , -0.03453948, -2.28857848, -0.03188161,  0.07727591,
        0.8262449 , -0.32295882,  1.18717788, -0.64775069])
```

In [40]:
```
en.intercept_
```

Out[40]:
```
28079048.79378794
```

In [41]:
```
prediction=en.predict(x_test)
```

In [42]:
```
en.score(x_test,y_test)
```

Out[42]:
```
0.1092601656644483
```

Evaluation Metrics

In [43]:
```
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)))
```

```
30.355249049040093
1208.5886420155277
34.7647614980389
```

Logistic Regression

In [44]:
```
from sklearn.linear_model import LogisticRegression
```

In [45]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

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

Out[46]:
```
(29669, 14)
```

In [47]:
```
target_vector.shape
```

Out[47]:
```
(29669,)
```

In [48]:
```
from sklearn.preprocessing import StandardScaler
```

In [49]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [50]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[50]:
```
LogisticRegression(max_iter=10000)
```

In [51]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [52]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079035]
```

In [53]:
```
logr.classes_
```

Out[53]:
```
array([28079006, 28079024, 28079035, 28079099], dtype=int64)
```

In [54]:
```
logr.score(fs,target_vector)
```

Out[54]:
```
0.8087229094340894
```

In [55]:
```
logr.predict_proba(observation)[0][0]
```

Out[55]:
```
1.724527777144498e-43
```

In [56]:
```
logr.predict_proba(observation)
```

Out[56]:
```
array([[1.72452778e-43, 2.43756289e-56, 9.99998565e-01, 1.43537418e-06]])
```

Random Forest

In [57]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [58]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[58]:
```
RandomForestClassifier()
```

In [64]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [*]:
```
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)
```

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

In [62]:
```
rfc_best=grid_search.best_estimator_
```

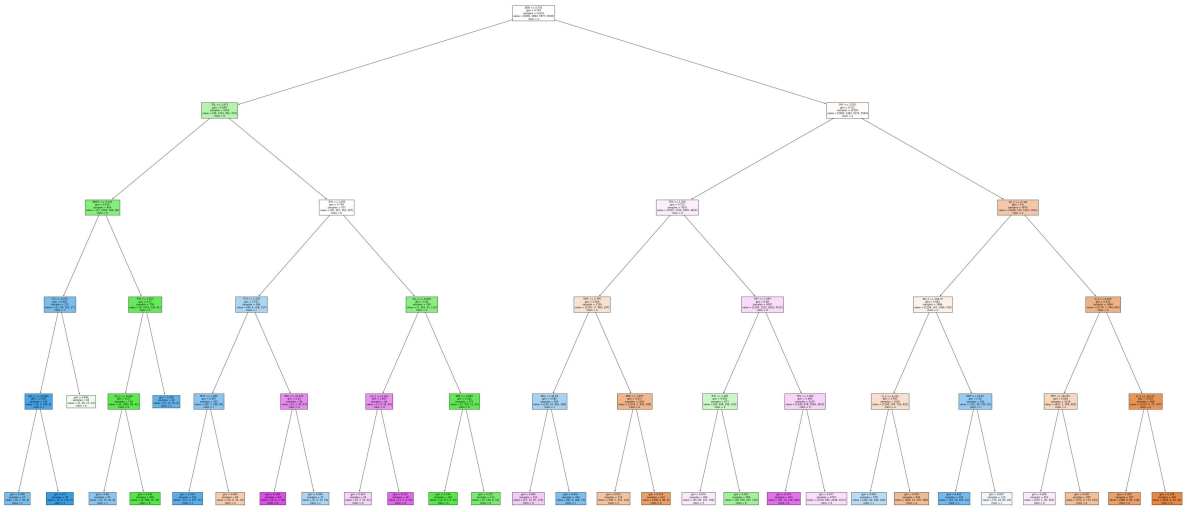
In [63]:
```
from sklearn.tree import plot_tree

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

Out[63]:
```
[Text(2012.7857142857142, 1993.2, 'BEN <= 0.755\ngini = 0.734\nsamples = 1311
9\nvalue = [6081, 2892, 5877, 5918]\nclass = a'),
 Text(836.9999999999999, 1630.8000000000002, 'TOL <= 2.475\ngini = 0.594\nsam
ples = 1616\nvalue = [98, 1410, 662, 335]\nclass = b'),
 Text(398.57142857142856, 1268.4, 'NMHC <= 0.035\ngini = 0.419\nsamples = 909
\nvalue = [17, 1043, 299, 68]\nclass = b'),
 Text(239.1428571428571, 906.0, 'CO <= 0.215\ngini = 0.465\nsamples = 151\nva
lue = [11, 28, 161, 27]\nclass = c'),
 Text(159.42857142857142, 543.5999999999999, 'NO_2 <= 19.885\ngini = 0.144\ns
amples = 105\nvalue = [6, 2, 146, 4]\nclass = c'),
 Text(79.71428571428571, 181.19999999999982, 'gini = 0.458\nsamples = 27\nval
ue = [6, 2, 30, 4]\nclass = c'),
 Text(239.1428571428571, 181.19999999999982, 'gini = 0.0\nsamples = 78\nvalue
= [0, 0, 116, 0]\nclass = c'),
 Text(318.85714285714283, 543.5999999999999, 'gini = 0.694\nsamples = 46\nval
ue = [5, 26, 15, 23]\nclass = b'),
 Text(558.0, 906.0, 'TCH <= 1.425\ngini = 0.27\nsamples = 758\nvalue = [6, 10
15, 138, 41]\nclass = b'),
 Text(478.2857142857142, 543.5999999999999, 'SO_2 <= 8.065\ngini = 0.21\nsamp
les = 718\nvalue = [6, 1003, 83, 41]\nclass = b'),
 Text(398.57142857142856, 181.19999999999982, 'gini = 0.48\nsamples = 52\nval
ue = [0, 15, 44, 6]\nclass = c'),
 Text(558.0, 181.19999999999982, 'gini = 0.142\nsamples = 666\nvalue = [6, 98
8, 39, 35]\nclass = b'),
 Text(637.7142857142857, 543.5999999999999, 'gini = 0.294\nsamples = 40\nvalu
e = [0, 12, 55, 0]\nclass = c'),
 Text(1275.4285714285713, 1268.4, 'TCH <= 1.265\ngini = 0.704\nsamples = 707
\nvalue = [81, 367, 363, 267]\nclass = b'),
 Text(956.5714285714284, 906.0, 'TCH <= 1.235\ngini = 0.572\nsamples = 368\nv
alue = [80, 4, 328, 157]\nclass = c'),
 Text(797.1428571428571, 543.5999999999999, 'MXY <= 1.855\ngini = 0.497\nsamp
les = 282\nvalue = [80, 2, 288, 60]\nclass = c'),
 Text(717.4285714285713, 181.19999999999982, 'gini = 0.254\nsamples = 193\nva
lue = [10, 2, 257, 31]\nclass = c'),
 Text(876.8571428571428, 181.19999999999982, 'gini = 0.603\nsamples = 89\nval
ue = [70, 0, 31, 29]\nclass = a'),
 Text(1116.0, 543.5999999999999, 'NOx <= 47.415\ngini = 0.43\nsamples = 86\nv
alue = [0, 2, 40, 97]\nclass = d'),
 Text(1036.2857142857142, 181.19999999999982, 'gini = 0.189\nsamples = 54\nva
lue = [0, 2, 7, 78]\nclass = d'),
 Text(1195.7142857142856, 181.19999999999982, 'gini = 0.464\nsamples = 32\nva
lue = [0, 0, 33, 19]\nclass = c'),
 Text(1594.2857142857142, 906.0, 'SO_2 <= 8.965\ngini = 0.44\nsamples = 339\n
value = [1, 363, 35, 110]\nclass = b'),
 Text(1434.8571428571427, 543.5999999999999, 'SO_2 <= 6.925\ngini = 0.493\nsa
mples = 64\nvalue = [1, 8, 24, 65]\nclass = d'),
 Text(1355.142857142857, 181.19999999999982, 'gini = 0.613\nsamples = 26\nval
ue = [0, 7, 15, 21]\nclass = d'),
 Text(1514.5714285714284, 181.19999999999982, 'gini = 0.333\nsamples = 38\nva
lue = [1, 1, 9, 44]\nclass = d'),
 Text(1753.7142857142856, 543.5999999999999, 'EBE <= 0.825\ngini = 0.241\nsam
ples = 275\nvalue = [0, 355, 11, 45]\nclass = b'),
 Text(1673.9999999999998, 181.19999999999982, 'gini = 0.144\nsamples = 162\nv
alue = [0, 217, 5, 13]\nclass = b'),
 Text(1833.4285714285713, 181.19999999999982, 'gini = 0.351\nsamples = 113\nv
alue = [0, 138, 6, 32]\nclass = b'),
 Text(3188.5714285714284, 1630.8000000000002, 'PXY <= 3.555\ngini = 0.711\nsa
```

```
mples = 11503\nvalue = [5983, 1482, 5215, 5583]\nclass = a'),
 Text(2550.8571428571427, 1268.4, 'TCH <= 1.255\ngini = 0.712\nsamples = 7633
\nvalue = [2555, 1339, 3805, 4422]\nclass = d'),
 Text(2232.0, 906.0, 'MXY <= 2.795\ngini = 0.594\nsamples = 1536\nvalue = [12
55, 17, 883, 297]\nclass = a'),
 Text(2072.5714285714284, 543.5999999999999, 'NOx <= 43.19\ngini = 0.585\nsam
ples = 495\nvalue = [139, 16, 463, 188]\nclass = c'),
 Text(1992.8571428571427, 181.19999999999982, 'gini = 0.665\nsamples = 153\nv
alue = [57, 12, 67, 115]\nclass = d'),
 Text(2152.285714285714, 181.19999999999982, 'gini = 0.452\nsamples = 342\nva
lue = [82, 4, 396, 73]\nclass = c'),
 Text(2391.428571428571, 543.5999999999999, 'BEN <= 1.975\ngini = 0.471\nsamp
les = 1041\nvalue = [1116, 1, 420, 109]\nclass = a'),
 Text(2311.7142857142853, 181.19999999999982, 'gini = 0.523\nsamples = 719\nv
alue = [708, 1, 331, 101]\nclass = a'),
 Text(2471.142857142857, 181.19999999999982, 'gini = 0.316\nsamples = 322\nva
lue = [408, 0, 89, 8]\nclass = a'),
 Text(2869.7142857142853, 906.0, 'PXY <= 1.085\ngini = 0.69\nsamples = 6097\n
value = [1300, 1322, 2922, 4125]\nclass = d'),
 Text(2710.285714285714, 543.5999999999999, 'TCH <= 1.305\ngini = 0.659\nsamp
les = 877\nvalue = [60, 644, 358, 310]\nclass = b'),
 Text(2630.5714285714284, 181.19999999999982, 'gini = 0.676\nsamples = 309\nv
alue = [30, 85, 161, 194]\nclass = d'),
 Text(2790.0, 181.19999999999982, 'gini = 0.551\nsamples = 568\nvalue = [30,
559, 197, 116]\nclass = b'),
 Text(3029.142857142857, 543.5999999999999, 'TOL <= 7.005\ngini = 0.664\nsamp
les = 5220\nvalue = [1240, 678, 2564, 3815]\nclass = d'),
 Text(2949.428571428571, 181.19999999999982, 'gini = 0.372\nsamples = 423\nva
lue = [30, 13, 116, 544]\nclass = d'),
 Text(3108.8571428571427, 181.19999999999982, 'gini = 0.677\nsamples = 4797\n
value = [1210, 665, 2448, 3271]\nclass = d'),
 Text(3826.2857142857138, 1268.4, 'SO_2 <= 27.08\ngini = 0.6\nsamples = 3870
\nvalue = [3428, 143, 1410, 1161]\nclass = a'),
 Text(3507.428571428571, 906.0, 'NO_2 <= 104.75\ngini = 0.662\nsamples = 1886
\nvalue = [1258, 142, 1066, 519]\nclass = a'),
 Text(3347.9999999999995, 543.5999999999999, 'O_3 <= 8.225\ngini = 0.651\nsam
ples = 1530\nvalue = [1148, 104, 716, 452]\nclass = a'),
 Text(3268.285714285714, 181.19999999999982, 'gini = 0.663\nsamples = 576\nva
lue = [202, 92, 446, 164]\nclass = c'),
 Text(3427.7142857142853, 181.19999999999982, 'gini = 0.543\nsamples = 954\nv
alue = [946, 12, 270, 288]\nclass = a'),
 Text(3666.8571428571427, 543.5999999999999, 'MXY <= 14.91\ngini = 0.56\nsamp
les = 356\nvalue = [110, 38, 350, 67]\nclass = c'),
 Text(3587.142857142857, 181.19999999999982, 'gini = 0.433\nsamples = 234\nva
lue = [34, 20, 265, 41]\nclass = c'),
 Text(3746.5714285714284, 181.19999999999982, 'gini = 0.667\nsamples = 122\nv
alue = [76, 18, 85, 26]\nclass = c'),
 Text(4145.142857142857, 906.0, 'O_3 <= 8.525\ngini = 0.474\nsamples = 1984\n
value = [2170, 1, 344, 642]\nclass = a'),
 Text(3985.7142857142853, 543.5999999999999, 'MXY <= 18.525\ngini = 0.604\nsa
mples = 1018\nvalue = [833, 1, 269, 495]\nclass = a'),
 Text(3905.9999999999995, 181.19999999999982, 'gini = 0.609\nsamples = 423\nv
alue = [263, 1, 90, 302]\nclass = d'),
 Text(4065.428571428571, 181.19999999999982, 'gini = 0.556\nsamples = 595\nva
lue = [570, 0, 179, 193]\nclass = a'),
 Text(4304.571428571428, 543.5999999999999, 'O_3 <= 18.125\ngini = 0.253\nsam
ples = 966\nvalue = [1337, 0, 75, 147]\nclass = a'),
```

```
 Text(4224.857142857142, 181.19999999999982, 'gini = 0.352\nsamples = 527\nva
lue = [688, 0, 65, 118]\nclass = a'),
 Text(4384.285714285714, 181.19999999999982, 'gini = 0.108\nsamples = 439\nva
lue = [649, 0, 10, 29]\nclass = a')]
```



Conclusion

Accuracy

In [63]:
```
# Linear Regression:0.16831590812021358
# Ridge Regression:0.16809856624085817
# Lasso Regression:0.039641720127249425
# ElasticNet Regression:0.09743364213542827
# Logistic Regression:0.8087229094340894
# Random Forest:0.7291987673343605
```

Logistic Regression is suitable for this dataset