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_2002.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990002
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.85	20.980000
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440001
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180000
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330002
...	...	...	...	...	...	...	...	...	...	...	...
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750000
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.389999
217293	2002-11-01 00:00:00	1.37	0.58	1.17	2.37	0.15	65.389999	107.699997	1.30	9.11	9.640000
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	NaN
217295	2002-11-01 00:00:00	3.11	1.17	3.00	7.77	0.26	80.110001	180.300003	2.25	7.38	29.240000

217296 rows × 16 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', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32381 entries, 1 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     32381 non-null  object 
 1   BEN      32381 non-null  float64
 2   CO       32381 non-null  float64
 3   EBE      32381 non-null  float64
 4   MXY      32381 non-null  float64
 5   NMHC     32381 non-null  float64
 6   NO_2     32381 non-null  float64
 7   NOx      32381 non-null  float64
 8   OXY      32381 non-null  float64
 9   O_3      32381 non-null  float64
 10  PM10     32381 non-null  float64
 11  PXY      32381 non-null  float64
 12  SO_2     32381 non-null  float64
 13  TCH      32381 non-null  float64
 14  TOL      32381 non-null  float64
 15  station  32381 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 4.2+ MB
```

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

Out[6]:

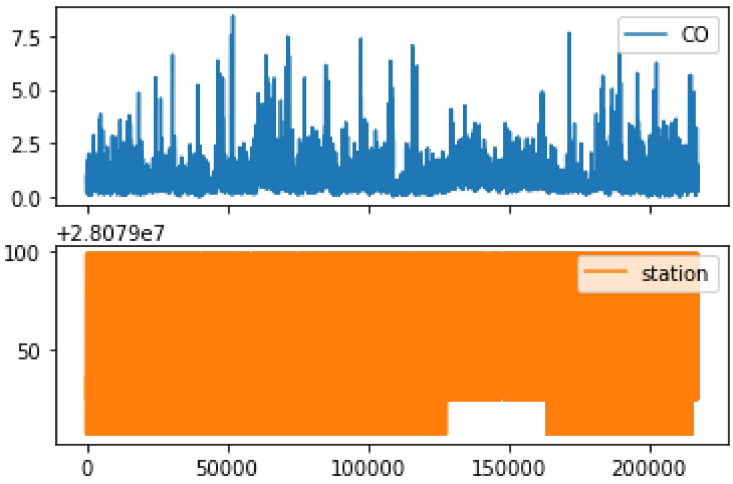
	CO	station
1	0.71	28079035
5	0.72	28079006
22	0.80	28079024
24	1.04	28079099
26	0.53	28079035
...	...	...
217269	0.28	28079024
217271	1.30	28079099
217273	0.97	28079035
217293	0.58	28079024
217295	1.17	28079099

32381 rows × 2 columns

Line chart

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

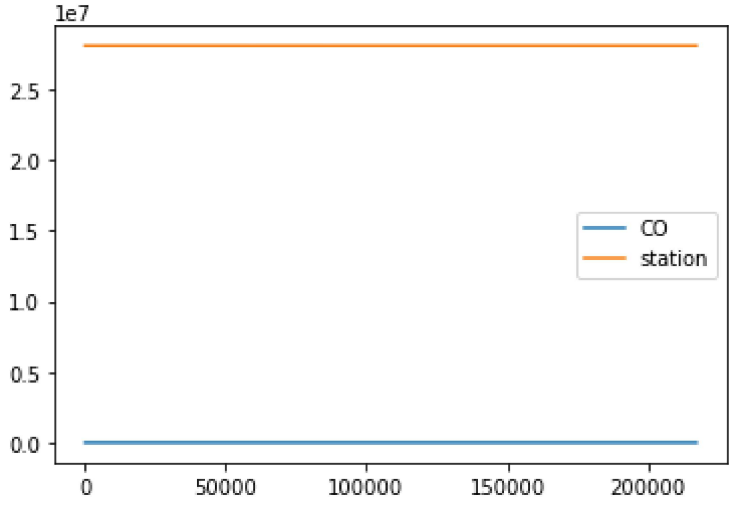
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

Out[8]: <AxesSubplot:>

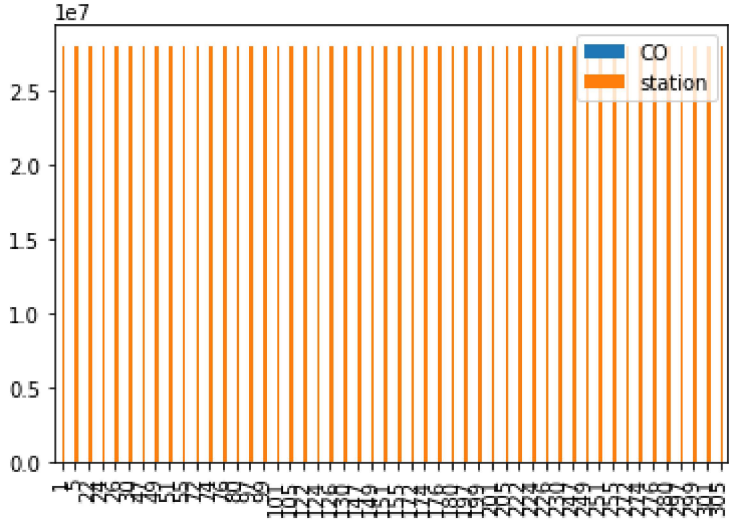


Bar chart

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

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

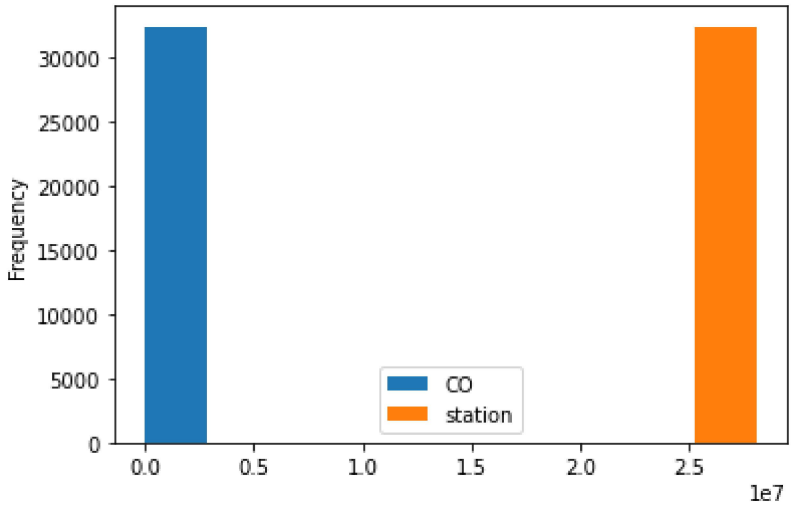
Out[10]: <AxesSubplot:>



Histogram

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

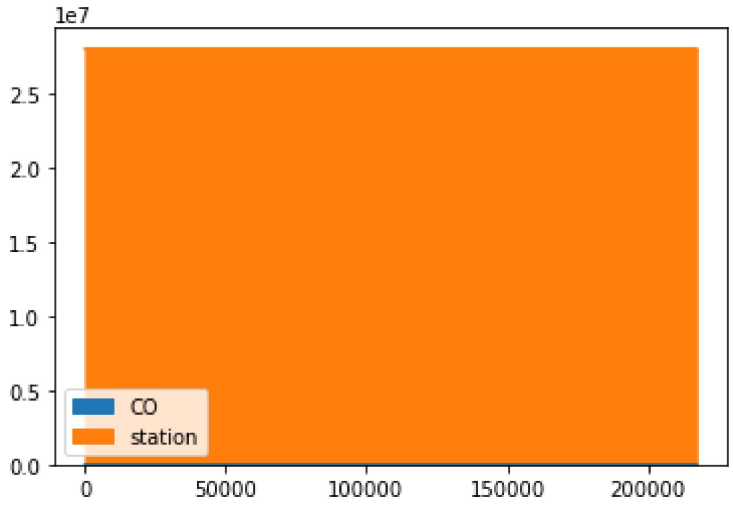
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

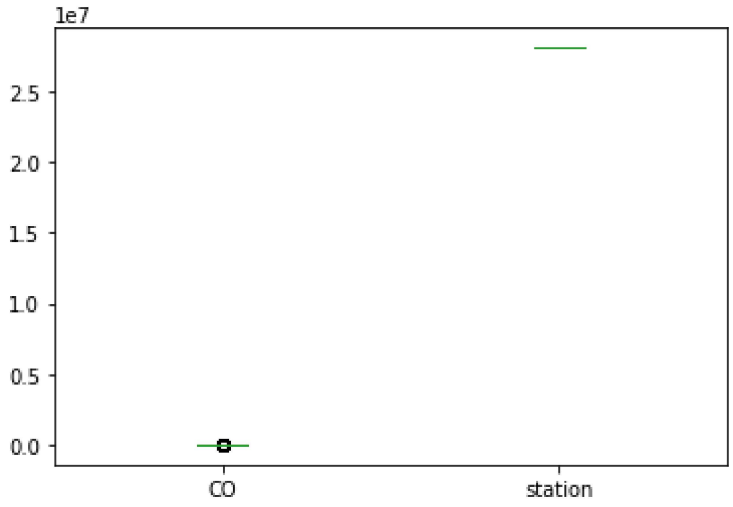
Out[12]: <AxesSubplot:>



Box chart

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

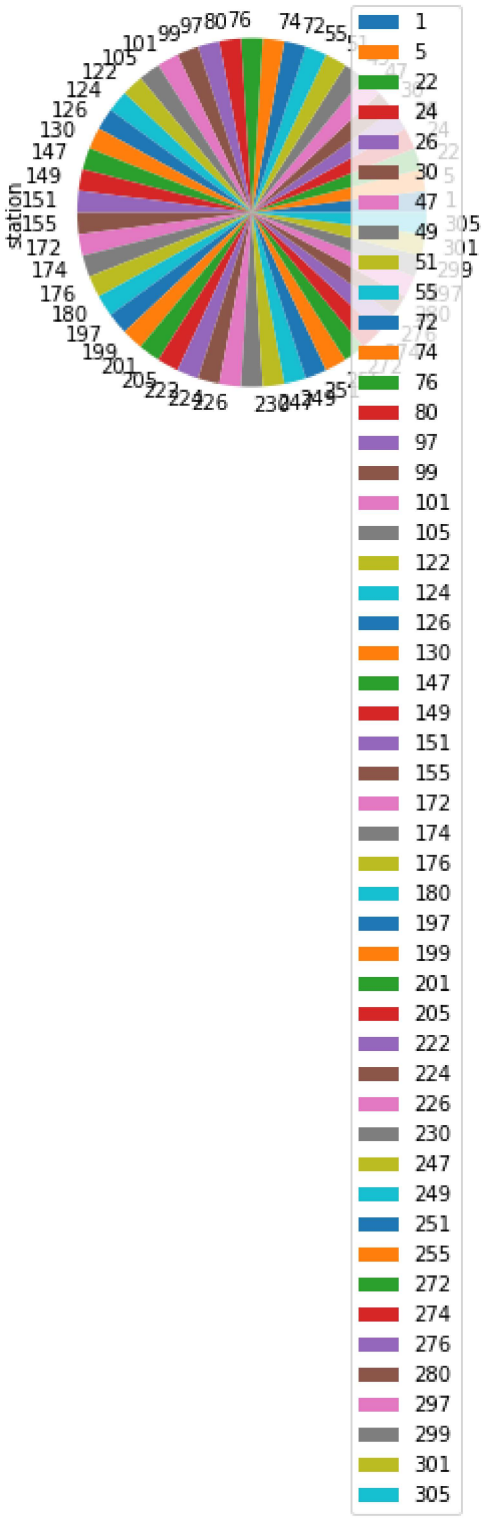
Out[13]: <AxesSubplot:>



Pie chart

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

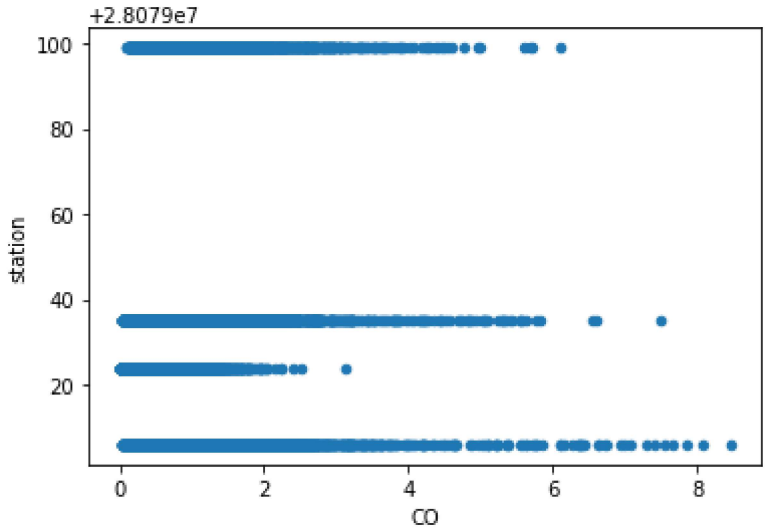
Out[14]: <AxesSubplot:ylabel='station'>



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'>
RangeIndex: 32381 entries, 0 to 32380
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     32381 non-null  object
 1   BEN      32381 non-null  float64
 2   CO       32381 non-null  float64
 3   EBE      32381 non-null  float64
 4   MXY      32381 non-null  float64
 5   NMHC     32381 non-null  float64
 6   NO_2     32381 non-null  float64
 7   NOx      32381 non-null  float64
 8   OXY      32381 non-null  float64
 9   O_3      32381 non-null  float64
 10  PM10     32381 non-null  float64
 11  PXY      32381 non-null  float64
 12  SO_2     32381 non-null  float64
 13  TCH      32381 non-null  float64
 14  TOL      32381 non-null  float64
 15  station  32381 non-null  int64
dtypes: float64(14), int64(1), object(1)
memory usage: 4.2+ MB
```

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

Out[17]:

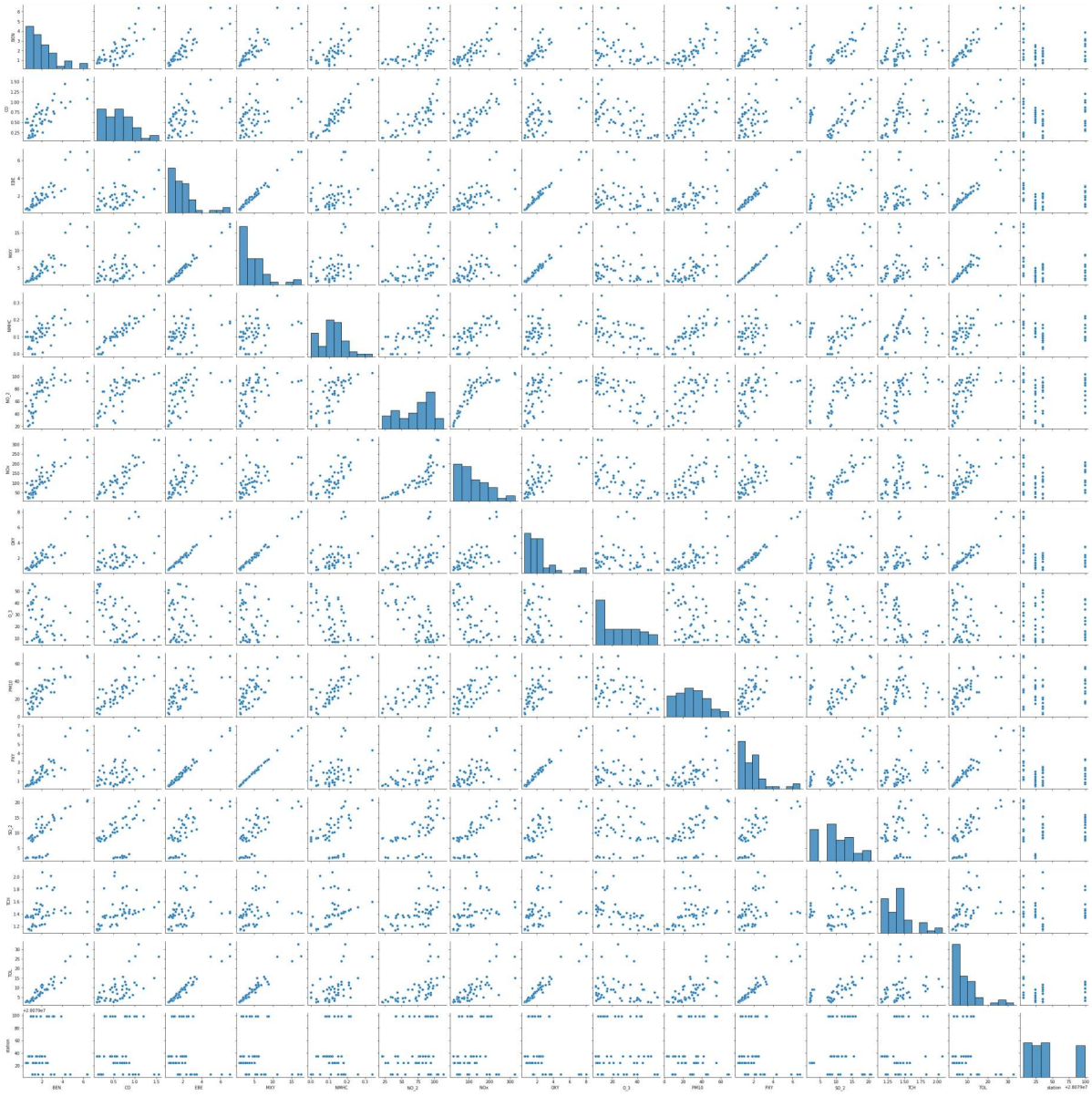
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000	323
mean	2.479155	0.787323	2.914004	7.013636	0.155827	58.936796	1
std	2.280959	0.610810	2.667881	6.774365	0.135731	31.472733	1
min	0.180000	0.000000	0.180000	0.190000	0.000000	0.890000	
25%	0.970000	0.420000	1.140000	2.420000	0.080000	35.660000	
50%	1.840000	0.620000	2.130000	5.140000	0.130000	57.160000	
75%	3.250000	0.980000	3.830000	9.420000	0.200000	78.769997	1
max	32.660000	8.460000	41.740002	99.879997	2.700000	263.600006	13

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])
```

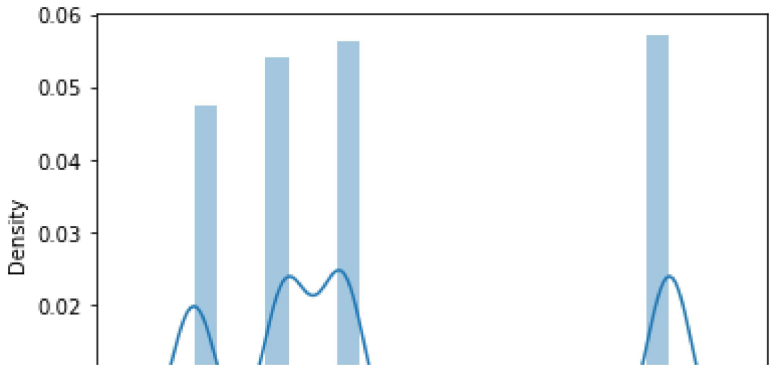
Out[19]: <seaborn.axisgrid.PairGrid at 0x29434385520>



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

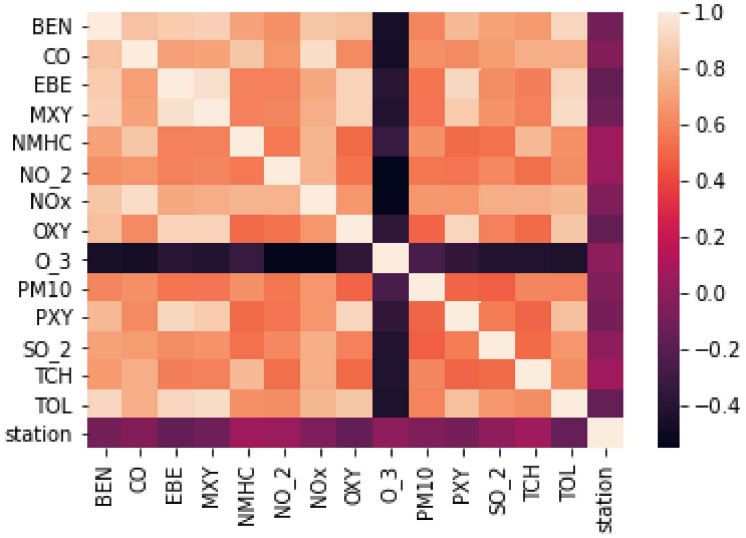
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
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]:
```
28078990.493966755
```

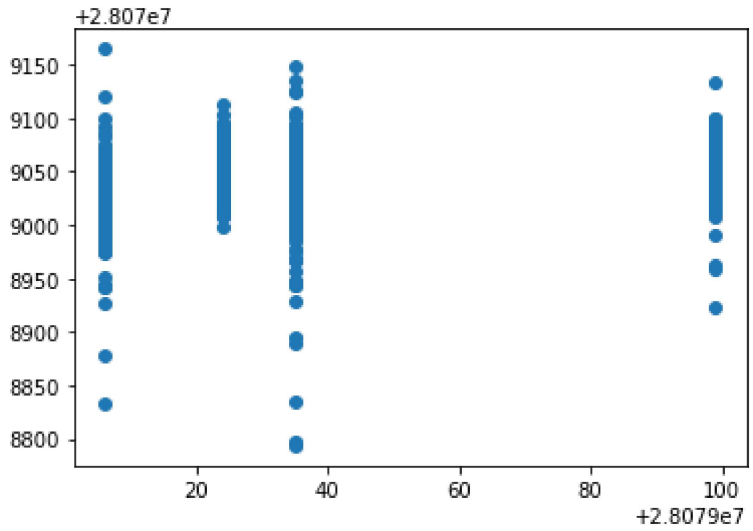
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	2.194318
CO	-11.893185
EBE	-12.054644
MXY	4.360416
NMHC	81.218933
NO_2	0.252798
NOx	-0.105985
OXY	-4.807837
O_3	-0.029851
PM10	-0.119406
PXY	7.462782
SO_2	0.627465
TCH	41.064831
TOL	-1.521883

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

Out[27]:
```
<matplotlib.collections.PathCollection at 0x29443b9e610>
```



ACCURACY

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

Out[28]:
```
0.19768123573291807
```

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

Out[29]:
```
0.1985162380075609
```

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]:
```
0.1966749815090698
```

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

Out[33]:
```
0.19832591348995832
```

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

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

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

Out[35]:
```
0.057575026649134275
```

Accuracy(Lasso)

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

Out[36]:
```
0.05997806773327663
```

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([ 1.10834557,  0.        , -3.01595239,  1.74457769,  0.21078396,
        0.227755  , -0.03039195, -2.33939424, -0.025651  ,  0.00432294,
        2.11101202,  0.42563526,  1.07850176, -1.23722711])
```

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

Out[39]:
```
28079038.3589574
```

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

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

Out[41]:
```
0.09690788147985696
```

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)))
```

```
28.530682568811933
1116.5848821706143
33.415339025223346
```

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]:
```
(32381, 14)
```

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

Out[46]:
```
(32381,)
```

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)
```

```
[28079035]
```

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

Out[52]:
```
array([28079006, 28079024, 28079035, 28079099], dtype=int64)
```

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

Out[53]:
```
0.8480899292795158
```

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

Out[54]:
```
2.5638972732451705e-10
```

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

Out[55]:
```
array([[2.56389727e-10, 3.44199742e-71, 1.00000000e+00, 1.43898646e-13]])
```

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]:
```
0.7738462895967528
```

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

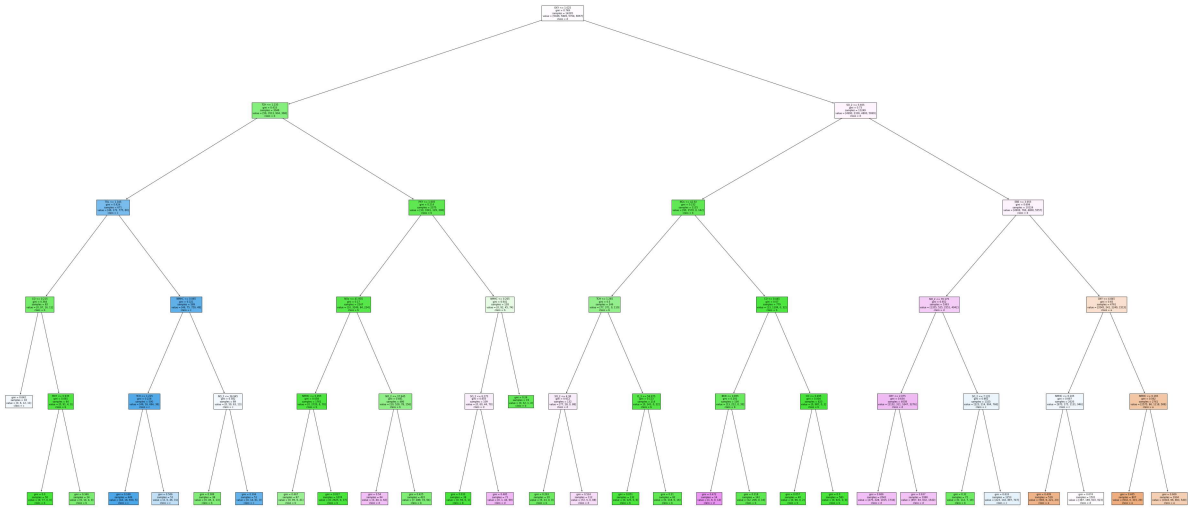
In [62]:
```
from sklearn.tree import plot_tree

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

Out[62]:
```
[Text(2114.5263157894738, 1993.2, 'OXY <= 1.025\ngini = 0.749\nsamples = 1429
5\nvalue = [5046, 5849, 5704, 6067]\nclass = d'),
 Text(1018.1052631578947, 1630.8000000000002, 'TCH <= 1.235\ngini = 0.433\nsa
mples = 3046\nvalue = [56, 3513, 904, 368]\nclass = b'),
 Text(430.7368421052631, 1268.4, 'TOL <= 1.345\ngini = 0.426\nsamples = 671\n
value = [46, 172, 775, 60]\nclass = c'),
 Text(156.6315789473684, 906.0, 'CO <= 0.215\ngini = 0.364\nsamples = 82\nval
ue = [0, 97, 16, 11]\nclass = b'),
 Text(78.3157894736842, 543.5999999999999, 'gini = 0.642\nsamples = 18\nvalue
= [0, 6, 12, 11]\nclass = c'),
 Text(234.9473684210526, 543.5999999999999, 'MXY <= 0.935\ngini = 0.081\nsamp
les = 64\nvalue = [0, 91, 4, 0]\nclass = b'),
 Text(156.6315789473684, 181.19999999999982, 'gini = 0.0\nsamples = 50\nvalue
= [0, 77, 0, 0]\nclass = b'),
 Text(313.2631578947368, 181.19999999999982, 'gini = 0.346\nsamples = 14\nval
ue = [0, 14, 4, 0]\nclass = b'),
 Text(704.8421052631578, 906.0, 'NMHC <= 0.065\ngini = 0.321\nsamples = 589\n
value = [46, 75, 759, 49]\nclass = c'),
 Text(548.2105263157895, 543.5999999999999, 'TCH <= 1.215\ngini = 0.226\nsamp
les = 500\nvalue = [46, 16, 696, 36]\nclass = c'),
 Text(469.8947368421052, 181.19999999999982, 'gini = 0.169\nsamples = 449\nva
lue = [44, 16, 650, 5]\nclass = c'),
 Text(626.5263157894736, 181.19999999999982, 'gini = 0.506\nsamples = 51\nval
ue = [2, 0, 46, 31]\nclass = c'),
 Text(861.4736842105262, 543.5999999999999, 'NO_2 <= 36.945\ngini = 0.582\nsa
mples = 89\nvalue = [0, 59, 63, 13]\nclass = c'),
 Text(783.1578947368421, 181.19999999999982, 'gini = 0.389\nsamples = 38\nval
ue = [0, 45, 2, 13]\nclass = b'),
 Text(939.7894736842104, 181.19999999999982, 'gini = 0.304\nsamples = 51\nval
ue = [0, 14, 61, 0]\nclass = c'),
 Text(1605.4736842105262, 1268.4, 'PXY <= 1.005\ngini = 0.214\nsamples = 2375
\nvalue = [10, 3341, 129, 308]\nclass = b'),
 Text(1331.3684210526314, 906.0, 'NOx <= 41.935\ngini = 0.17\nsamples = 2247
\nvalue = [10, 3249, 84, 234]\nclass = b'),
 Text(1174.7368421052631, 543.5999999999999, 'NMHC <= 0.055\ngini = 0.058\nsa
mples = 1761\nvalue = [0, 2720, 6, 78]\nclass = b'),
 Text(1096.421052631579, 181.19999999999982, 'gini = 0.467\nsamples = 87\nval
ue = [0, 95, 6, 41]\nclass = b'),
 Text(1253.0526315789473, 181.19999999999982, 'gini = 0.027\nsamples = 1674\n
value = [0, 2625, 0, 37]\nclass = b'),
 Text(1488.0, 543.5999999999999, 'NO_2 <= 37.945\ngini = 0.481\nsamples = 486
\nvalue = [10, 529, 78, 156]\nclass = b'),
 Text(1409.6842105263156, 181.19999999999982, 'gini = 0.54\nsamples = 64\nval
ue = [3, 40, 4, 62]\nclass = d'),
 Text(1566.3157894736842, 181.19999999999982, 'gini = 0.425\nsamples = 422\nv
alue = [7, 489, 74, 94]\nclass = b'),
 Text(1879.5789473684208, 906.0, 'NMHC <= 0.205\ngini = 0.641\nsamples = 128
\nvalue = [0, 92, 45, 74]\nclass = b'),
 Text(1801.2631578947367, 543.5999999999999, 'SO_2 <= 6.275\ngini = 0.655\nsa
mples = 109\nvalue = [0, 60, 44, 70]\nclass = d'),
 Text(1722.9473684210525, 181.19999999999982, 'gini = 0.033\nsamples = 38\nva
lue = [0, 59, 0, 1]\nclass = b'),
 Text(1879.5789473684208, 181.19999999999982, 'gini = 0.485\nsamples = 71\nva
lue = [0, 1, 44, 69]\nclass = d'),
 Text(1957.8947368421052, 543.5999999999999, 'gini = 0.24\nsamples = 19\nvalu
e = [0, 32, 1, 4]\nclass = b'),
 Text(3210.9473684210525, 1630.8000000000002, 'SO_2 <= 5.655\ngini = 0.73\nsa
```

```
mples = 11249\nvalue = [4990, 2336, 4800, 5699]\nclass = d'),
 Text(2584.4210526315787, 1268.4, 'NOx <= 42.92\ngini = 0.232\nsamples = 1125
\nvalue = [90, 1570, 0, 142]\nclass = b'),
 Text(2271.157894736842, 906.0, 'TCH <= 1.265\ngini = 0.5\nsamples = 346\nval
ue = [77, 371, 0, 110]\nclass = b'),
 Text(2114.5263157894738, 543.5999999999999, 'SO_2 <= 4.38\ngini = 0.612\nsam
ples = 132\nvalue = [77, 28, 0, 88]\nclass = d'),
 Text(2036.2105263157894, 181.19999999999982, 'gini = 0.293\nsamples = 20\nva
lue = [5, 23, 0, 0]\nclass = b'),
 Text(2192.842105263158, 181.19999999999982, 'gini = 0.524\nsamples = 112\nva
lue = [72, 5, 0, 88]\nclass = d'),
 Text(2427.7894736842104, 543.5999999999999, 'O_3 <= 54.975\ngini = 0.113\nsa
mples = 214\nvalue = [0, 343, 0, 22]\nclass = b'),
 Text(2349.4736842105262, 181.19999999999982, 'gini = 0.051\nsamples = 132\nv
alue = [0, 225, 0, 6]\nclass = b'),
 Text(2506.1052631578946, 181.19999999999982, 'gini = 0.21\nsamples = 82\nval
ue = [0, 118, 0, 16]\nclass = b'),
 Text(2897.6842105263154, 906.0, 'CO <= 0.445\ngini = 0.07\nsamples = 779\nva
lue = [13, 1199, 0, 32]\nclass = b'),
 Text(2741.052631578947, 543.5999999999999, 'BEN <= 0.855\ngini = 0.292\nsamp
les = 156\nvalue = [13, 212, 0, 30]\nclass = b'),
 Text(2662.736842105263, 181.19999999999982, 'gini = 0.472\nsamples = 14\nval
ue = [1, 6, 0, 14]\nclass = d'),
 Text(2819.368421052631, 181.19999999999982, 'gini = 0.218\nsamples = 142\nva
lue = [12, 206, 0, 16]\nclass = b'),
 Text(3054.315789473684, 543.5999999999999, 'CO <= 0.495\ngini = 0.004\nsampl
es = 623\nvalue = [0, 987, 0, 2]\nclass = b'),
 Text(2976.0, 181.19999999999982, 'gini = 0.057\nsamples = 40\nvalue = [0, 6
6, 0, 2]\nclass = b'),
 Text(3132.6315789473683, 181.19999999999982, 'gini = 0.0\nsamples = 583\nval
ue = [0, 921, 0, 0]\nclass = b'),
 Text(3837.4736842105262, 1268.4, 'EBE <= 3.055\ngini = 0.694\nsamples = 1012
4\nvalue = [4900, 766, 4800, 5557]\nclass = d'),
 Text(3524.210526315789, 906.0, 'NO_2 <= 70.175\ngini = 0.652\nsamples = 5363
\nvalue = [1355, 525, 2551, 4042]\nclass = d'),
 Text(3367.578947368421, 543.5999999999999, 'OXY <= 2.075\ngini = 0.634\nsamp
les = 4038\nvalue = [1132, 311, 1647, 3276]\nclass = d'),
 Text(3289.2631578947367, 181.19999999999982, 'gini = 0.606\nsamples = 2042\n
value = [275, 228, 1015, 1734]\nclass = d'),
 Text(3445.894736842105, 181.19999999999982, 'gini = 0.637\nsamples = 1996\nv
alue = [857, 83, 632, 1542]\nclass = d'),
 Text(3680.8421052631575, 543.5999999999999, 'SO_2 <= 7.135\ngini = 0.662\nsa
mples = 1325\nvalue = [223, 214, 904, 766]\nclass = c'),
 Text(3602.5263157894733, 181.19999999999982, 'gini = 0.32\nsamples = 72\nval
ue = [0, 112, 7, 19]\nclass = b'),
 Text(3759.1578947368416, 181.19999999999982, 'gini = 0.633\nsamples = 1253\n
value = [223, 102, 897, 747]\nclass = c'),
 Text(4150.736842105262, 906.0, 'OXY <= 4.845\ngini = 0.65\nsamples = 4761\nv
alue = [3545, 241, 2249, 1515]\nclass = a'),
 Text(3994.1052631578946, 543.5999999999999, 'NMHC <= 0.105\ngini = 0.697\nsa
mples = 2020\nvalue = [970, 175, 1131, 946]\nclass = c'),
 Text(3915.7894736842104, 181.19999999999982, 'gini = 0.439\nsamples = 519\nv
alue = [583, 6, 221, 23]\nclass = a'),
 Text(4072.4210526315787, 181.19999999999982, 'gini = 0.674\nsamples = 1501\n
value = [387, 169, 910, 923]\nclass = d'),
 Text(4307.368421052632, 543.5999999999999, 'NMHC <= 0.165\ngini = 0.562\nsam
ples = 2741\nvalue = [2575, 66, 1118, 569]\nclass = a'),
```

```
 Text(4229.0526315789475, 181.19999999999982, 'gini = 0.405\nsamples = 807\nv
alue = [932, 0, 315, 29]\nclass = a'),
 Text(4385.684210526316, 181.19999999999982, 'gini = 0.609\nsamples = 1934\nv
alue = [1643, 66, 803, 540]\nclass = a')]
```



Conclusion

Accuracy

In [63]:
```
# Linear Regression:0.1985162380075609
# Ridge Regression:0.16809856624085817
# Lasso Regression:0.057575026649134275
# ElasticNet Regression:0.09690788147985696
# Logistic Regression:0.8480899292795158
# Random Forest:0.7738462895967528
```

Logistic Regression is suitable for this dataset

In [ ]:
```

```