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_2016.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2016-11-01 01:00:00	NaN	0.7	NaN	NaN	153.0	77.0	NaN	NaN	NaN	7.0	NaN	NaN	2
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4	2
2	2016-11-01 01:00:00	5.9	NaN	7.5	NaN	297.0	139.0	NaN	NaN	NaN	NaN	NaN	26.0	2
3	2016-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	113.0	2.0	NaN	NaN	NaN	NaN	NaN	2
4	2016-11-01 01:00:00	NaN	NaN	NaN	NaN	275.0	127.0	2.0	NaN	NaN	18.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209491	2016-07-01 00:00:00	NaN	0.2	NaN	NaN	2.0	29.0	73.0	NaN	NaN	NaN	NaN	NaN	2
209492	2016-07-01 00:00:00	NaN	0.3	NaN	NaN	1.0	29.0	NaN	36.0	NaN	5.0	NaN	NaN	2
209493	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	1.0	19.0	71.0	NaN	NaN	NaN	NaN	NaN	2
209494	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	6.0	17.0	85.0	NaN	NaN	NaN	NaN	NaN	2
209495	2016-07-01 00:00:00	NaN	NaN	NaN	NaN	2.0	46.0	61.0	34.0	NaN	NaN	NaN	NaN	2

209496 rows × 14 columns

Data Cleaning and Data Preprocessing

In [3]:
```
df=df.dropna()
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16932 entries, 1 to 209478
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16932 non-null  object 
 1   BEN      16932 non-null  float64
 2   CO       16932 non-null  float64
 3   EBE      16932 non-null  float64
 4   NMHC     16932 non-null  float64
 5   NO       16932 non-null  float64
 6   NO_2     16932 non-null  float64
 7   O_3      16932 non-null  float64
 8   PM10     16932 non-null  float64
 9   PM25     16932 non-null  float64
 10  SO_2     16932 non-null  float64
 11  TCH      16932 non-null  float64
 12  TOL      16932 non-null  float64
 13  station  16932 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.9+ MB
```

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

Out[6]:

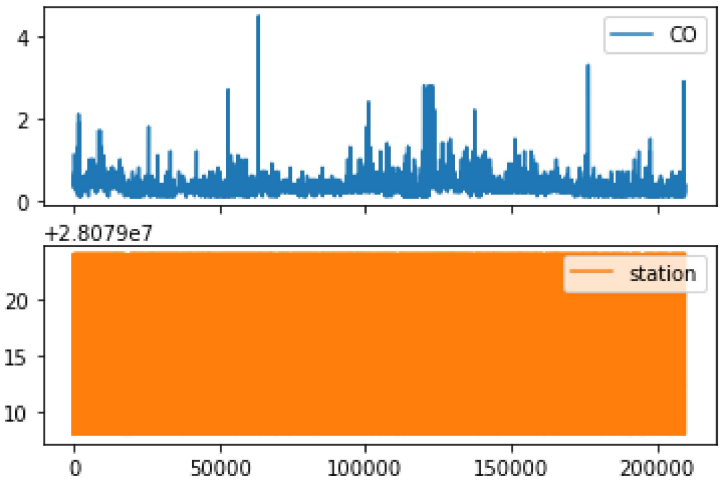
	CO	station
1	1.1	28079008
6	0.8	28079024
25	1.0	28079008
30	0.7	28079024
49	0.8	28079008
...	...	...
209430	0.2	28079024
209449	0.4	28079008
209454	0.2	28079024
209473	0.4	28079008
209478	0.2	28079024

16932 rows × 2 columns

Line chart

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

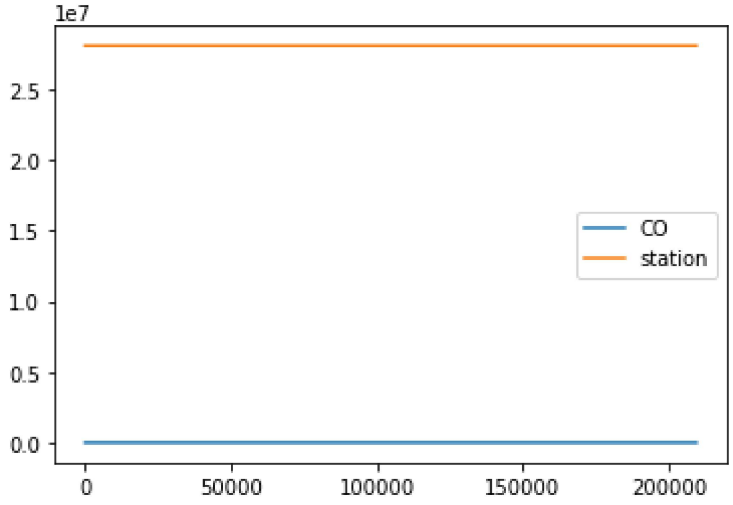
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

Out[8]: <AxesSubplot:>

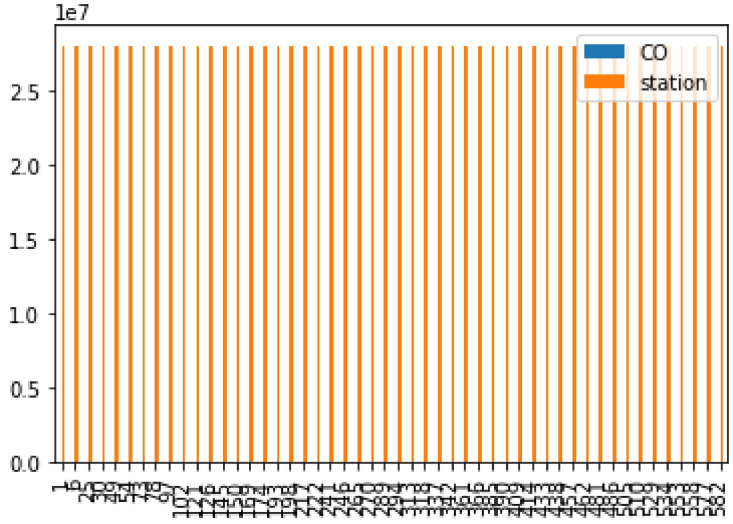


Bar chart

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

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

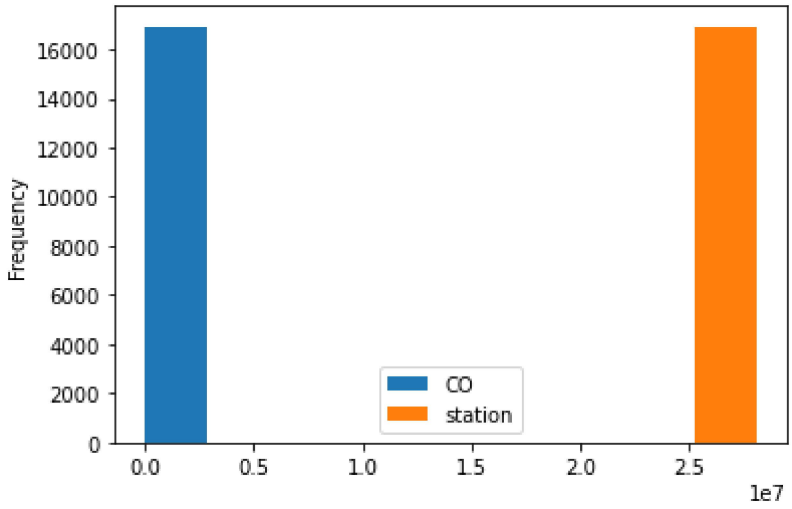
Out[10]: <AxesSubplot:>



Histogram

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

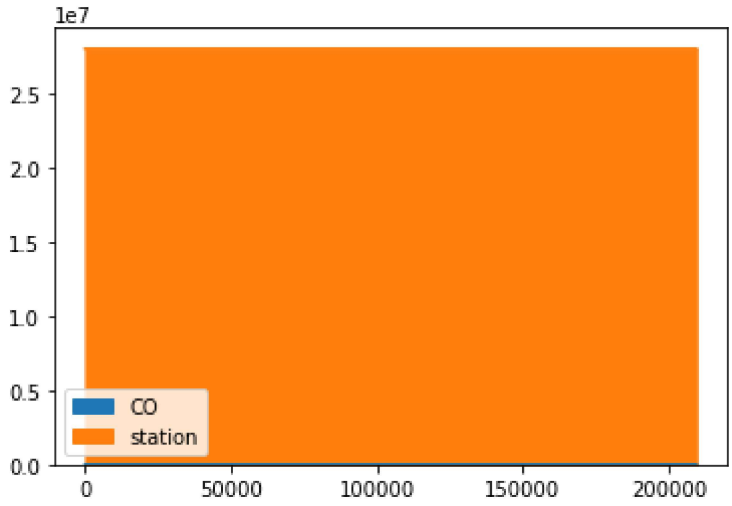
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

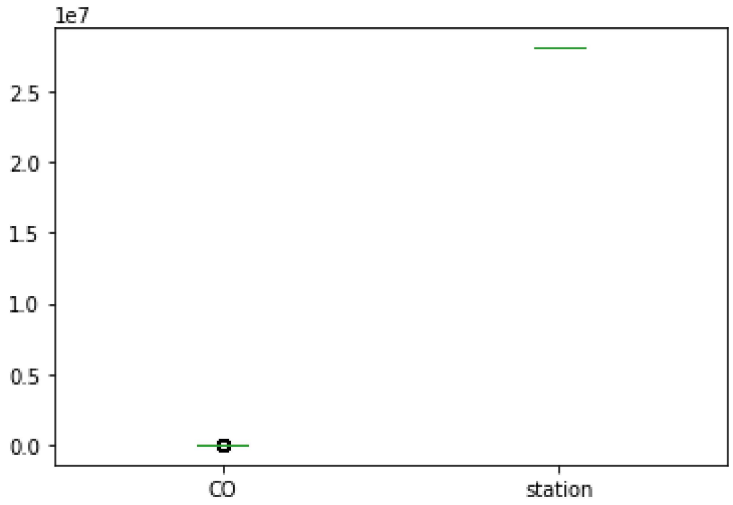
Out[12]: <AxesSubplot:>



Box chart

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

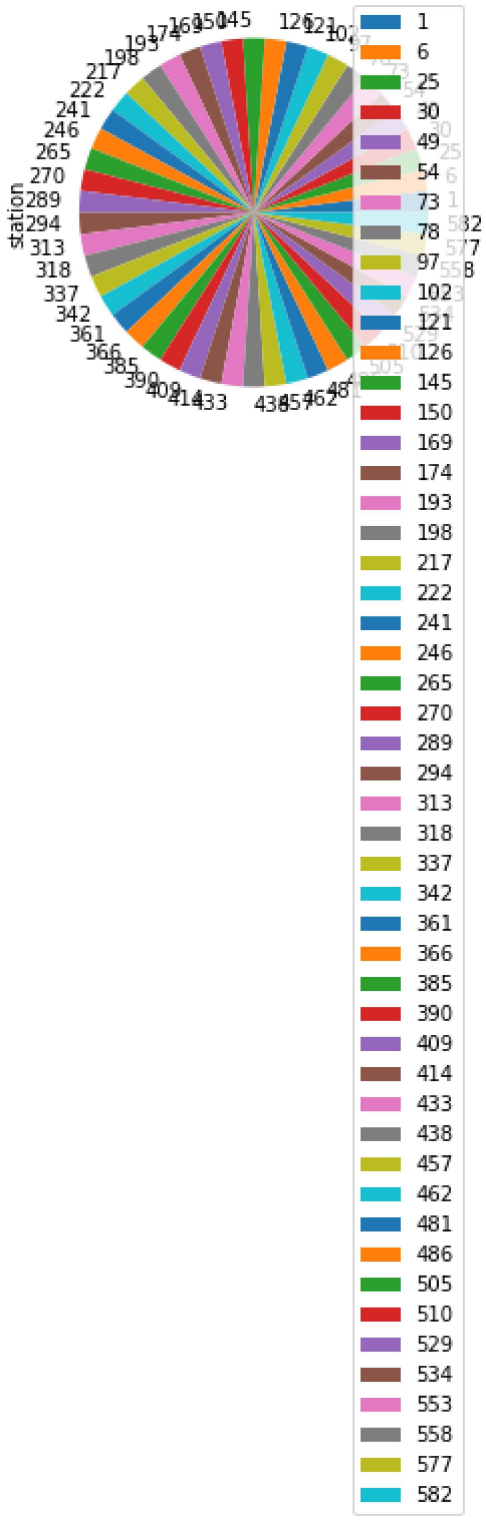
Out[13]: <AxesSubplot:>



Pie chart

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

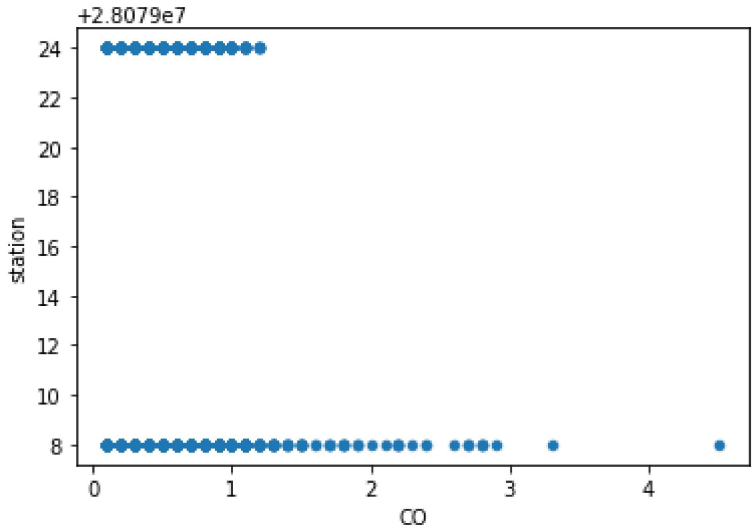
Out[14]: <AxesSubplot:ylabel='station'>



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: 16932 entries, 1 to 209478
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16932 non-null  object 
 1   BEN      16932 non-null  float64
 2   CO       16932 non-null  float64
 3   EBE      16932 non-null  float64
 4   NMHC     16932 non-null  float64
 5   NO       16932 non-null  float64
 6   NO_2     16932 non-null  float64
 7   O_3      16932 non-null  float64
 8   PM10     16932 non-null  float64
 9   PM25     16932 non-null  float64
 10  SO_2     16932 non-null  float64
 11  TCH      16932 non-null  float64
 12  TOL      16932 non-null  float64
 13  station  16932 non-null  int64  
dtypes: float64(12), int64(1), object(1)
```

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

Out[17]:

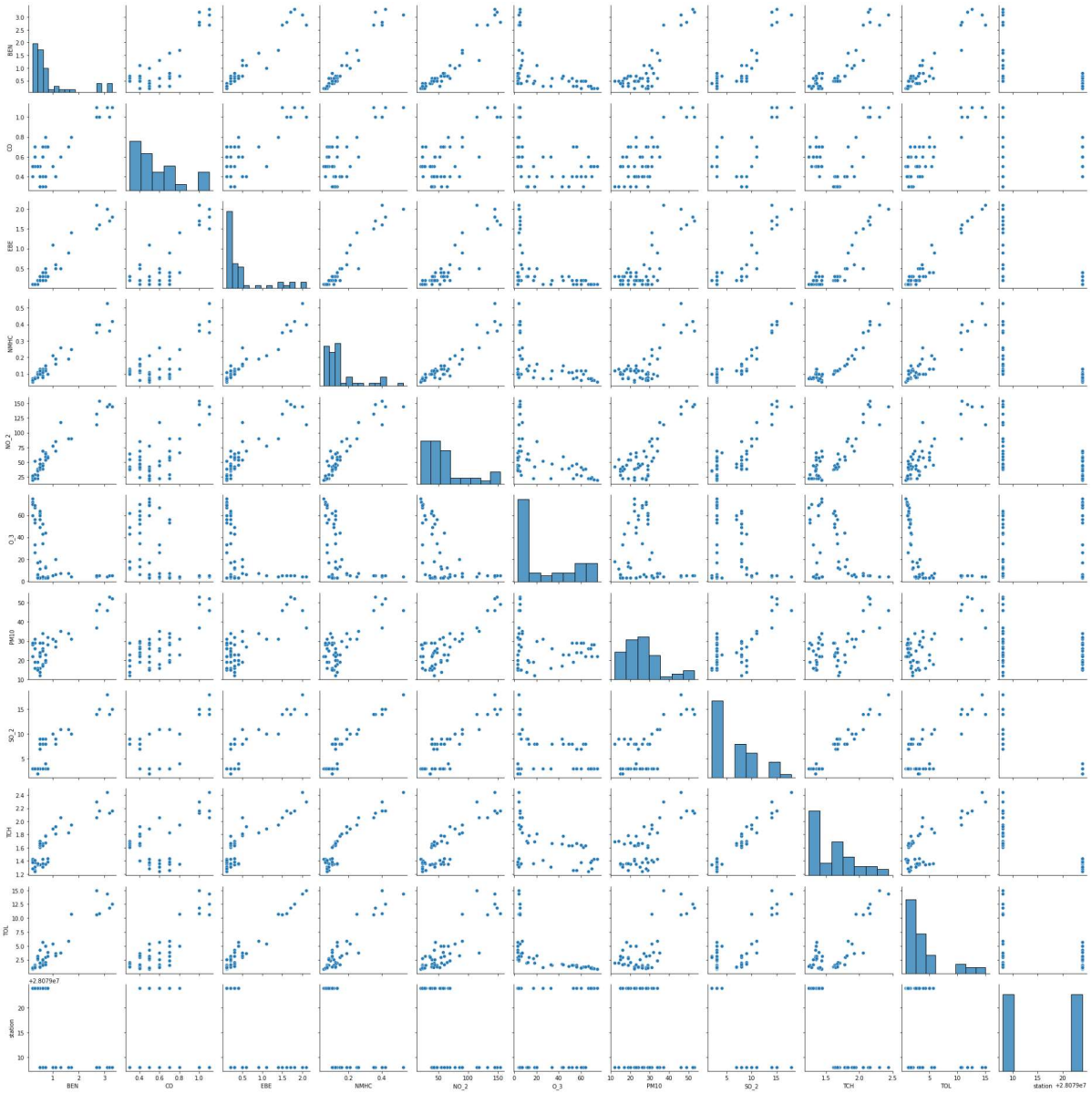
	BEN	CO	EBE	NMHC	NO	NO_2	
count	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	16932.000000	169
mean	0.537970	0.349941	0.298955	0.099913	20.815734	39.373376	
std	0.599479	0.203807	0.450204	0.079850	40.986063	31.170307	
min	0.100000	0.100000	0.100000	0.000000	1.000000	1.000000	
25%	0.200000	0.200000	0.100000	0.050000	1.000000	14.000000	
50%	0.400000	0.300000	0.200000	0.090000	7.000000	34.000000	
75%	0.700000	0.400000	0.300000	0.120000	23.000000	58.000000	
max	12.300000	4.500000	13.500000	2.210000	829.000000	319.000000	1

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

EDA AND VISUALIZATION

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

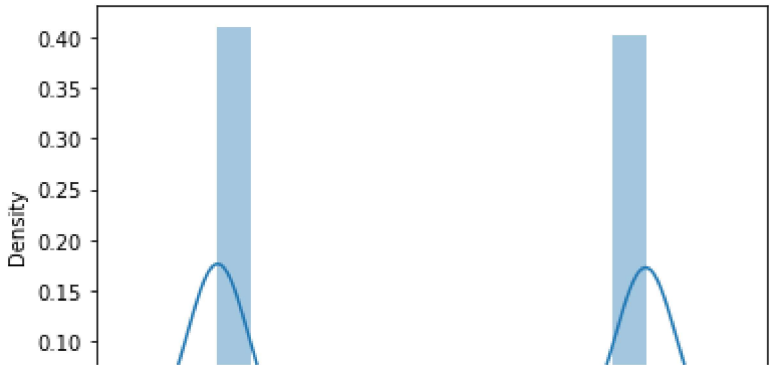
Out[19]: <seaborn.axisgrid.PairGrid at 0x217182acb80>



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

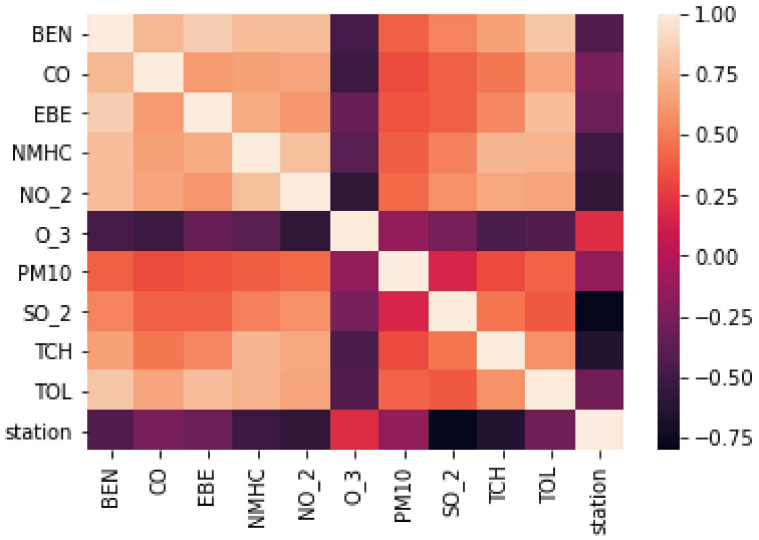
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
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','NMHC', 'NO_2','O_3',
      'PM10', '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]:
```
28079041.10516898
```

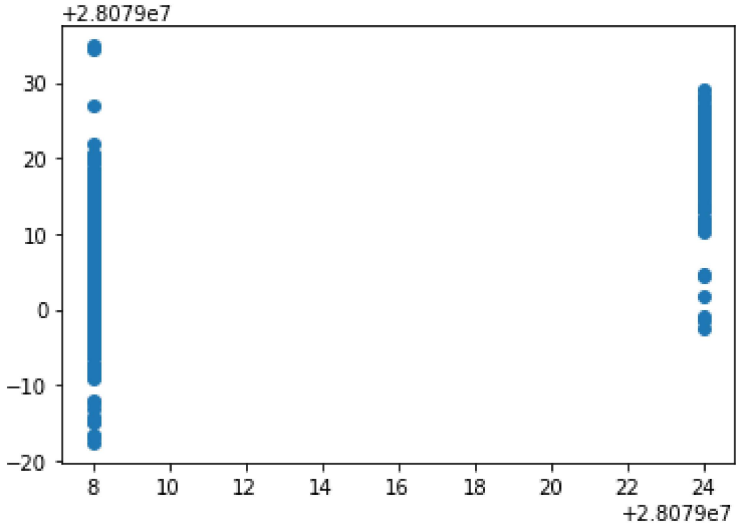
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	1.798987
CO	5.440386
EBE	0.027010
NMHC	5.276424
NO_2	-0.067407
O_3	-0.027303
PM10	0.020046
SO_2	-0.834781
TCH	-13.792112
TOL	0.316362

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

Out[27]: <matplotlib.collections.PathCollection at 0x217207edc10>



ACCURACY

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

Out[28]: 0.7913181693429105

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

Out[29]: 0.7954969948906043

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]: 0.7916266110767852

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

Out[33]:
```
0.7953055224216964
```

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]:
```
0.6154179664592716
```

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

Out[36]:
```
0.6185084690471969
```

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([ 0.        ,  0.        ,  0.        , -0.        , -0.08290022,
       -0.02526972,  0.02192113, -0.84091817, -0.        ,  0.26696177])
```

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

Out[39]:
```
28079025.80926477
```

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

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

Out[41]:
```
0.6889695555591844
```

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)))
```

```
3.4955207436748847
19.90369916706919
4.461356202666313
```

Logistic Regression

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

In [44]:
```
feature_matrix=df[['BEN', 'CO', 'EBE','NMHC', 'NO_2','O_3',
       'PM10', 'SO_2', 'TCH', 'TOL' ]]
target_vector=df[ 'station']
```

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

Out[45]:
```
(16932, 10)
```

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

Out[46]:
```
(16932,)
```

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]]
```

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

```
[28079008]
```

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

Out[52]:
```
array([28079008, 28079024], dtype=int64)
```

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

Out[53]:
```
0.9923812898653437
```

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

Out[54]:
```
1.0
```

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

Out[55]:
```
array([[1.0000000e+00, 1.6336121e-46]])
```

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]:
```
0.9952750590617617
```

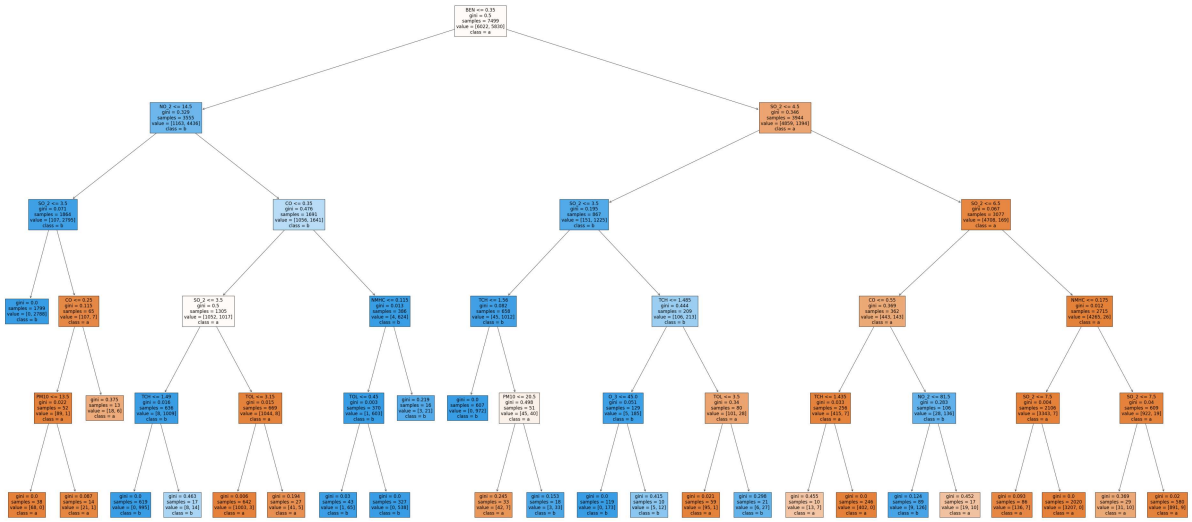
In [61]:
```
rfc_best=grid_search.best_estimator_
```

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(1795.304347826087, 1993.2, 'BEN <= 0.35\ngini = 0.5\nsamples = 7499\nva
lue = [6022, 5830]\nclass = a'),
 Text(655.0434782608695, 1630.8000000000002, 'NO_2 <= 14.5\ngini = 0.329\nsam
ples = 3555\nvalue = [1163, 4436]\nclass = b'),
 Text(194.08695652173913, 1268.4, 'SO_2 <= 3.5\ngini = 0.071\nsamples = 1864
\nvalue = [107, 2795]\nclass = b'),
 Text(97.04347826086956, 906.0, 'gini = 0.0\nsamples = 1799\nvalue = [0, 278
8]\nclass = b'),
 Text(291.1304347826087, 906.0, 'CO <= 0.25\ngini = 0.115\nsamples = 65\nvalu
e = [107, 7]\nclass = a'),
 Text(194.08695652173913, 543.5999999999999, 'PM10 <= 13.5\ngini = 0.022\nsam
ples = 52\nvalue = [89, 1]\nclass = a'),
 Text(97.04347826086956, 181.19999999999982, 'gini = 0.0\nsamples = 38\nvalue
= [68, 0]\nclass = a'),
 Text(291.1304347826087, 181.19999999999982, 'gini = 0.087\nsamples = 14\nval
ue = [21, 1]\nclass = a'),
 Text(388.17391304347825, 543.5999999999999, 'gini = 0.375\nsamples = 13\nval
ue = [18, 6]\nclass = a'),
 Text(1116.0, 1268.4, 'CO <= 0.35\ngini = 0.476\nsamples = 1691\nvalue = [105
6, 1641]\nclass = b'),
 Text(776.3478260869565, 906.0, 'SO_2 <= 3.5\ngini = 0.5\nsamples = 1305\nval
ue = [1052, 1017]\nclass = a'),
 Text(582.2608695652174, 543.5999999999999, 'TCH <= 1.49\ngini = 0.016\nsampl
es = 636\nvalue = [8, 1009]\nclass = b'),
 Text(485.2173913043478, 181.19999999999982, 'gini = 0.0\nsamples = 619\nvalu
e = [0, 995]\nclass = b'),
 Text(679.304347826087, 181.19999999999982, 'gini = 0.463\nsamples = 17\nvalu
e = [8, 14]\nclass = b'),
 Text(970.4347826086956, 543.5999999999999, 'TOL <= 3.15\ngini = 0.015\nsampl
es = 669\nvalue = [1044, 8]\nclass = a'),
 Text(873.391304347826, 181.19999999999982, 'gini = 0.006\nsamples = 642\nval
ue = [1003, 3]\nclass = a'),
 Text(1067.4782608695652, 181.19999999999982, 'gini = 0.194\nsamples = 27\nva
lue = [41, 5]\nclass = a'),
 Text(1455.6521739130435, 906.0, 'NMHC <= 0.115\ngini = 0.013\nsamples = 386
\nvalue = [4, 624]\nclass = b'),
 Text(1358.608695652174, 543.5999999999999, 'TOL <= 0.45\ngini = 0.003\nsampl
es = 370\nvalue = [1, 603]\nclass = b'),
 Text(1261.5652173913043, 181.19999999999982, 'gini = 0.03\nsamples = 43\nval
ue = [1, 65]\nclass = b'),
 Text(1455.6521739130435, 181.19999999999982, 'gini = 0.0\nsamples = 327\nval
ue = [0, 538]\nclass = b'),
 Text(1552.695652173913, 543.5999999999999, 'gini = 0.219\nsamples = 16\nvalu
e = [3, 21]\nclass = b'),
 Text(2935.5652173913045, 1630.8000000000002, 'SO_2 <= 4.5\ngini = 0.346\nsam
ples = 3944\nvalue = [4859, 1394]\nclass = a'),
 Text(2183.478260869565, 1268.4, 'SO_2 <= 3.5\ngini = 0.195\nsamples = 867\nv
alue = [151, 1225]\nclass = b'),
 Text(1843.8260869565217, 906.0, 'TCH <= 1.56\ngini = 0.082\nsamples = 658\nv
alue = [45, 1012]\nclass = b'),
 Text(1746.782608695652, 543.5999999999999, 'gini = 0.0\nsamples = 607\nvalue
= [0, 972]\nclass = b'),
 Text(1940.8695652173913, 543.5999999999999, 'PM10 <= 20.5\ngini = 0.498\nsam
ples = 51\nvalue = [45, 40]\nclass = a'),
 Text(1843.8260869565217, 181.19999999999982, 'gini = 0.245\nsamples = 33\nva
lue = [42, 7]\nclass = a'),
 Text(2037.9130434782608, 181.19999999999982, 'gini = 0.153\nsamples = 18\nva
```

```
lue = [3, 33]\nclass = b'),
 Text(2523.1304347826085, 906.0, 'TCH <= 1.485\ngini = 0.444\nsamples = 209\n
value = [106, 213]\nclass = b'),
 Text(2329.0434782608695, 543.5999999999999, 'O_3 <= 45.0\ngini = 0.051\nsamp
les = 129\nvalue = [5, 185]\nclass = b'),
 Text(2232.0, 181.19999999999982, 'gini = 0.0\nsamples = 119\nvalue = [0, 17
3]\nclass = b'),
 Text(2426.086956521739, 181.19999999999982, 'gini = 0.415\nsamples = 10\nval
ue = [5, 12]\nclass = b'),
 Text(2717.217391304348, 543.5999999999999, 'TOL <= 3.5\ngini = 0.34\nsamples
= 80\nvalue = [101, 28]\nclass = a'),
 Text(2620.173913043478, 181.19999999999982, 'gini = 0.021\nsamples = 59\nval
ue = [95, 1]\nclass = a'),
 Text(2814.2608695652175, 181.19999999999982, 'gini = 0.298\nsamples = 21\nva
lue = [6, 27]\nclass = b'),
 Text(3687.6521739130435, 1268.4, 'SO_2 <= 6.5\ngini = 0.067\nsamples = 3077
\nvalue = [4708, 169]\nclass = a'),
 Text(3299.478260869565, 906.0, 'CO <= 0.55\ngini = 0.369\nsamples = 362\nval
ue = [443, 143]\nclass = a'),
 Text(3105.391304347826, 543.5999999999999, 'TCH <= 1.435\ngini = 0.033\nsamp
les = 256\nvalue = [415, 7]\nclass = a'),
 Text(3008.3478260869565, 181.19999999999982, 'gini = 0.455\nsamples = 10\nva
lue = [13, 7]\nclass = a'),
 Text(3202.4347826086955, 181.19999999999982, 'gini = 0.0\nsamples = 246\nval
ue = [402, 0]\nclass = a'),
 Text(3493.565217391304, 543.5999999999999, 'NO_2 <= 81.5\ngini = 0.283\nsamp
les = 106\nvalue = [28, 136]\nclass = b'),
 Text(3396.5217391304345, 181.19999999999982, 'gini = 0.124\nsamples = 89\nva
lue = [9, 126]\nclass = b'),
 Text(3590.608695652174, 181.19999999999982, 'gini = 0.452\nsamples = 17\nval
ue = [19, 10]\nclass = a'),
 Text(4075.8260869565215, 906.0, 'NMHC <= 0.175\ngini = 0.012\nsamples = 2715
\nvalue = [4265, 26]\nclass = a'),
 Text(3881.7391304347825, 543.5999999999999, 'SO_2 <= 7.5\ngini = 0.004\nsamp
les = 2106\nvalue = [3343, 7]\nclass = a'),
 Text(3784.695652173913, 181.19999999999982, 'gini = 0.093\nsamples = 86\nval
ue = [136, 7]\nclass = a'),
 Text(3978.782608695652, 181.19999999999982, 'gini = 0.0\nsamples = 2020\nval
ue = [3207, 0]\nclass = a'),
 Text(4269.913043478261, 543.5999999999999, 'SO_2 <= 7.5\ngini = 0.04\nsample
s = 609\nvalue = [922, 19]\nclass = a'),
 Text(4172.869565217391, 181.19999999999982, 'gini = 0.369\nsamples = 29\nval
ue = [31, 10]\nclass = a'),
 Text(4366.95652173913, 181.19999999999982, 'gini = 0.02\nsamples = 580\nvalu
e = [891, 9]\nclass = a')]
```



Accuracy

Linear Regression

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

Out[63]: 0.7954969948906043

Ridge Regression

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

Out[64]: 0.7953055224216964

Lasso Regression

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

Out[65]: 0.6185084690471969

ElasticNet Regression

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

Out[66]: 0.6889695555591844

Logistic Regression

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

Out[67]:
```
0.9923812898653437
```

Random Forest

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

Out[68]:
```
0.9952750590617617
```

Conclusion

RandomForest for this dataset