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_2015.csv")
df
```

Out[2]:

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

Out[6]:

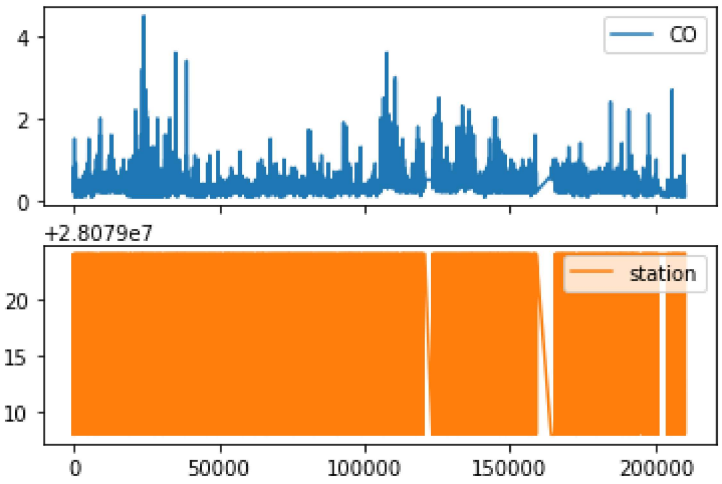
	CO	station
1	0.8	28079008
6	0.3	28079024
25	0.7	28079008
30	0.3	28079024
49	0.8	28079008
...	...	...
210030	0.1	28079024
210049	0.3	28079008
210054	0.1	28079024
210073	0.3	28079008
210078	0.1	28079024

16026 rows × 2 columns

Line chart

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

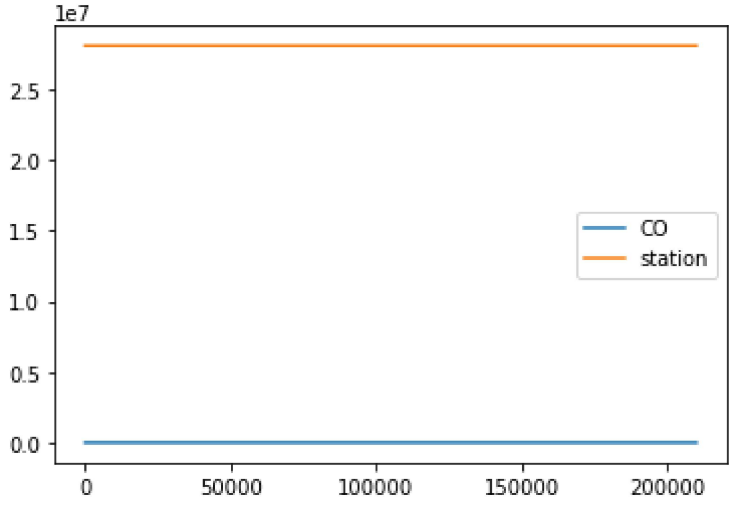
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

Out[8]: <AxesSubplot:>

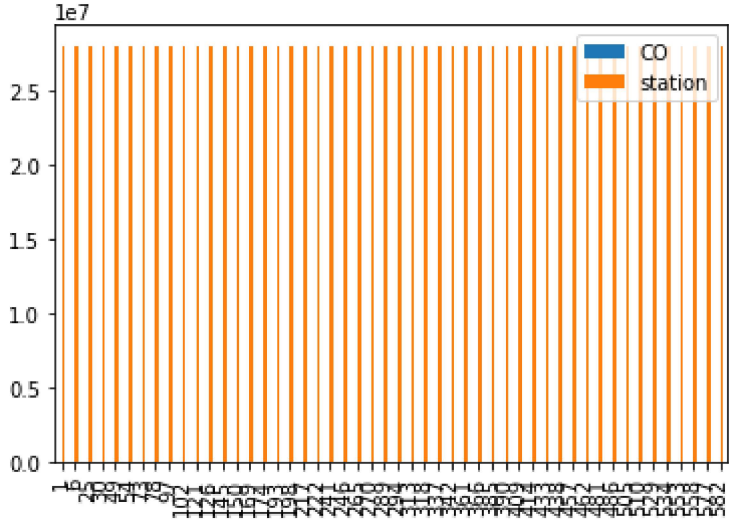


Bar chart

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

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

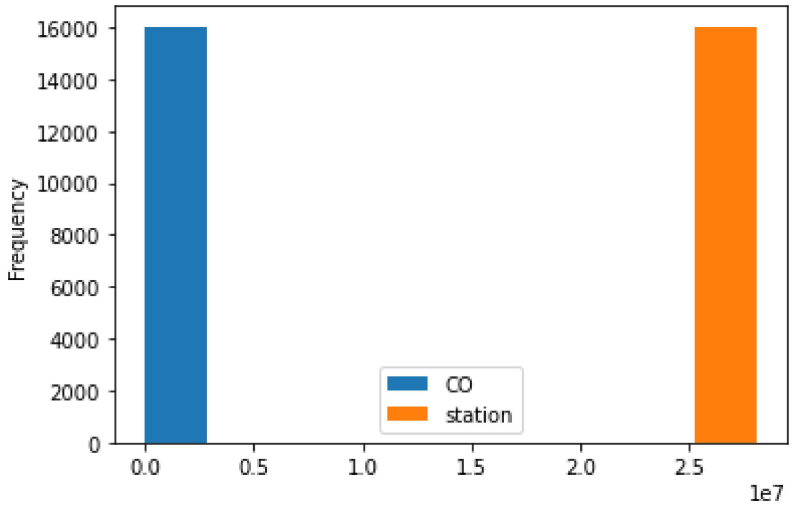
Out[10]: <AxesSubplot:>



Histogram

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

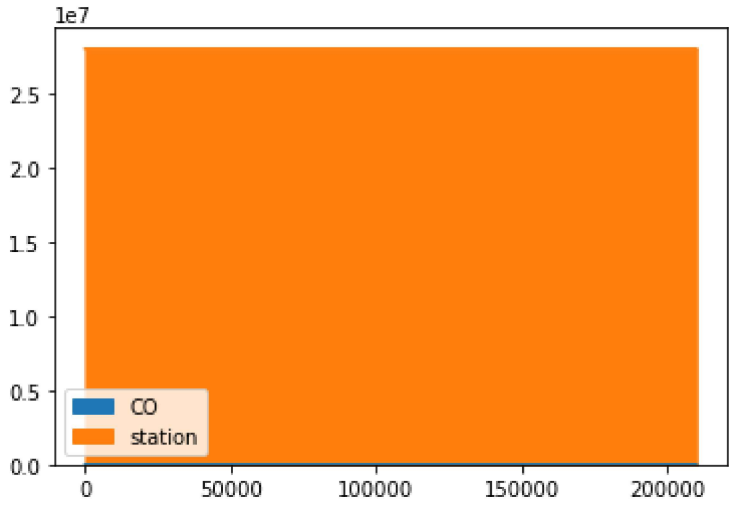
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

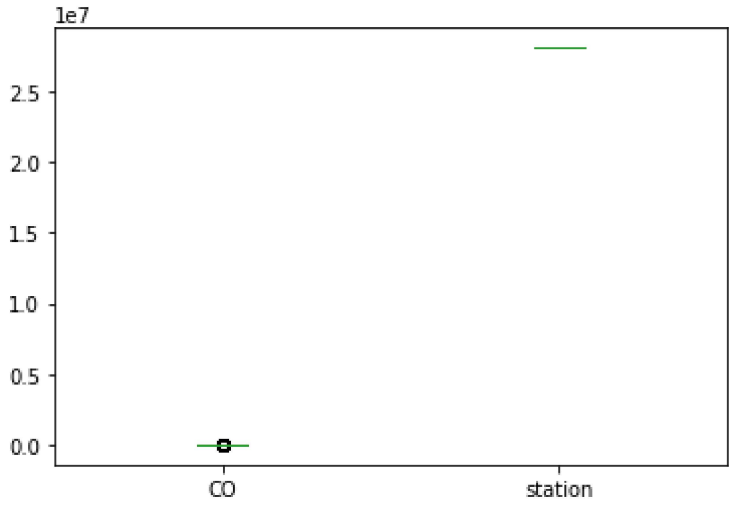
Out[12]: <AxesSubplot:>



Box chart

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

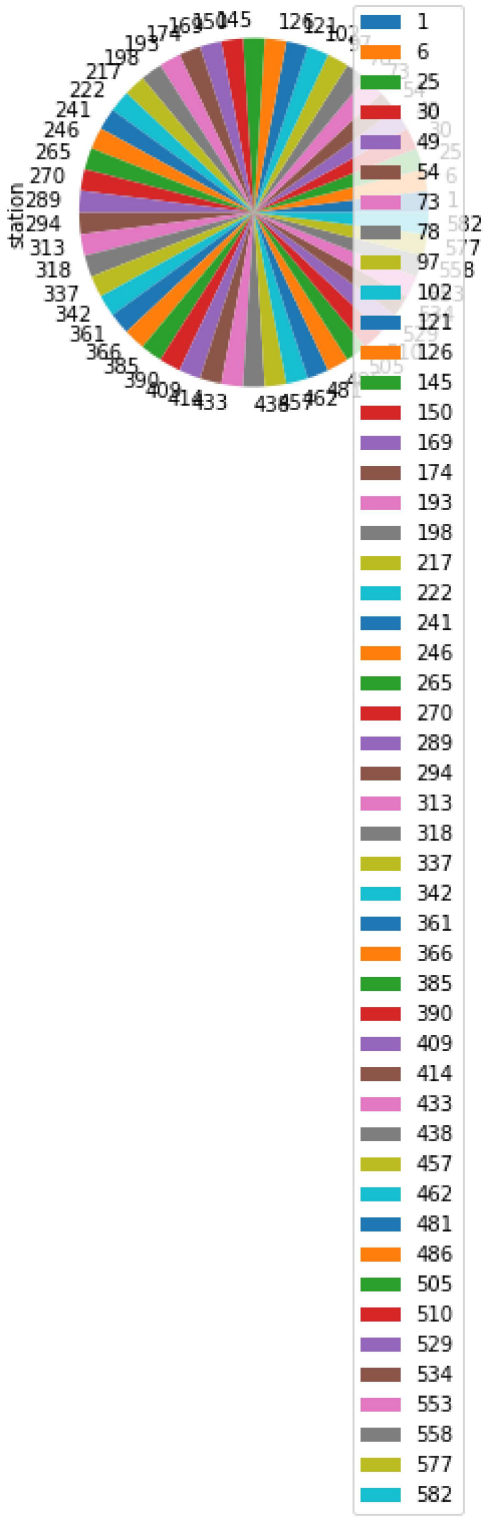
Out[13]: <AxesSubplot:>



Pie chart

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

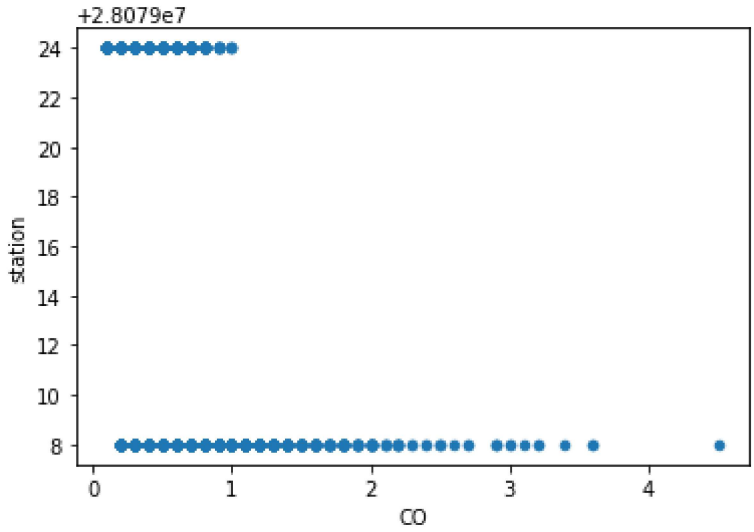
Out[14]: <AxesSubplot:ylabel='station'>



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

Out[17]:

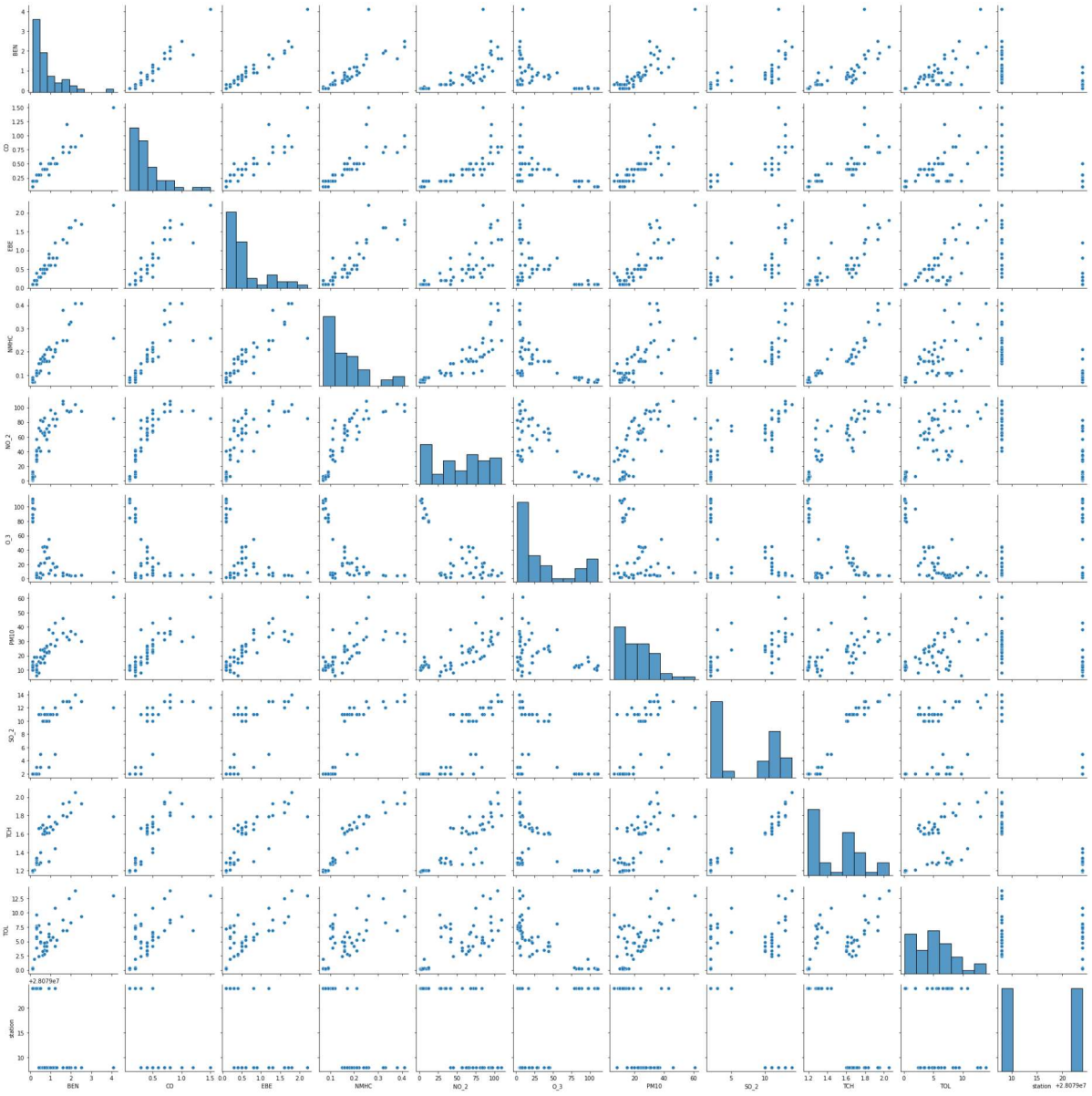
	BEN	CO	EBE	NMHC	NO	NO_2	
count	16026.000000	16026.000000	16026.000000	16026.000000	16026.000000	16026.000000	160
mean	0.504823	0.380594	0.394247	0.123099	23.842256	40.948771	
std	0.716896	0.260805	0.678592	0.092368	51.255660	33.236098	
min	0.100000	0.100000	0.100000	0.000000	1.000000	1.000000	
25%	0.100000	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 [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])
```

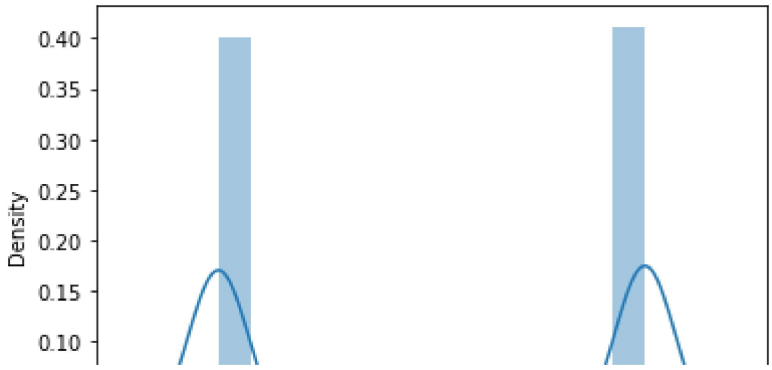
Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x1e3b4702f70>
```



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

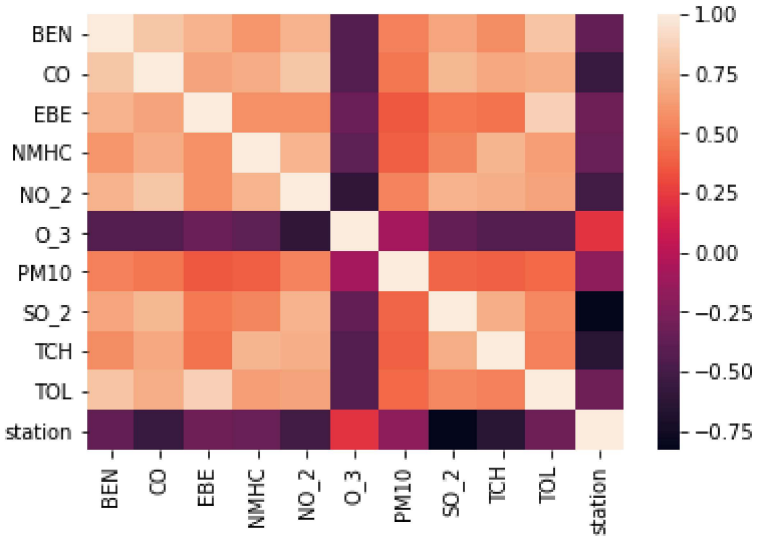
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
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]:
```
28079037.79794513
```

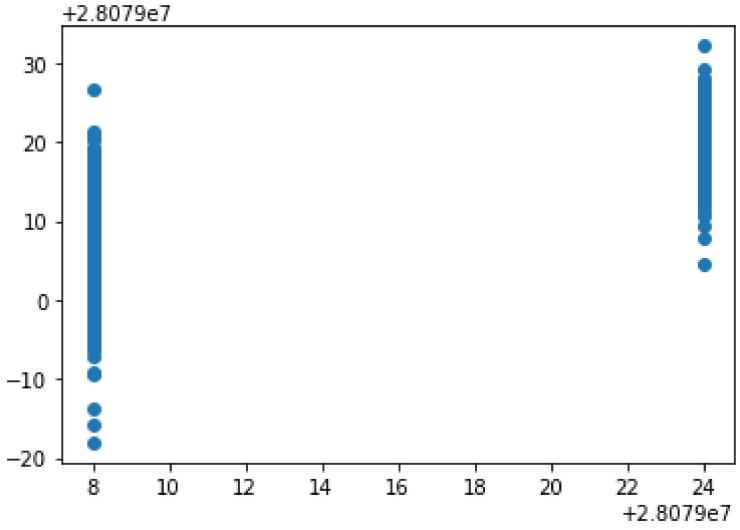
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	4.742921
CO	-6.833017
EBE	-1.122945
NMHC	24.072488
NO_2	-0.003839
O_3	-0.019734
PM10	0.062365
SO_2	-1.165376
TCH	-10.796944
TOL	-0.091051

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

Out[27]: <matplotlib.collections.PathCollection at 0x1e3bdcdf520>



ACCURACY

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

Out[28]: 0.8236019179020243

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

Out[29]: 0.8014642069655231

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]: 0.8215221224228407

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

Out[33]:
```
0.7994760584220166
```

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]:
```
0.6328420674894814
```

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

Out[36]:
```
0.6430421339351964
```

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.        ,
       -0.01653116,  0.07384199, -1.24429213, -0.        ,  0.14001623])
```

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

Out[39]:
```
28079024.125974357
```

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

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

Out[41]:
```
0.7470130026718783
```

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)))
```

```
3.2102805572444146
16.187735844351604
4.023398544060929
```

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]:
```
(16026, 10)
```

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

Out[46]:
```
(16026,)
```

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]:
```
0.9947585174092101
```

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

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

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

Out[55]:
```
array([[1.00000000e+00, 5.69793111e-39]])
```

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]:
```
0.9941165983241219
```

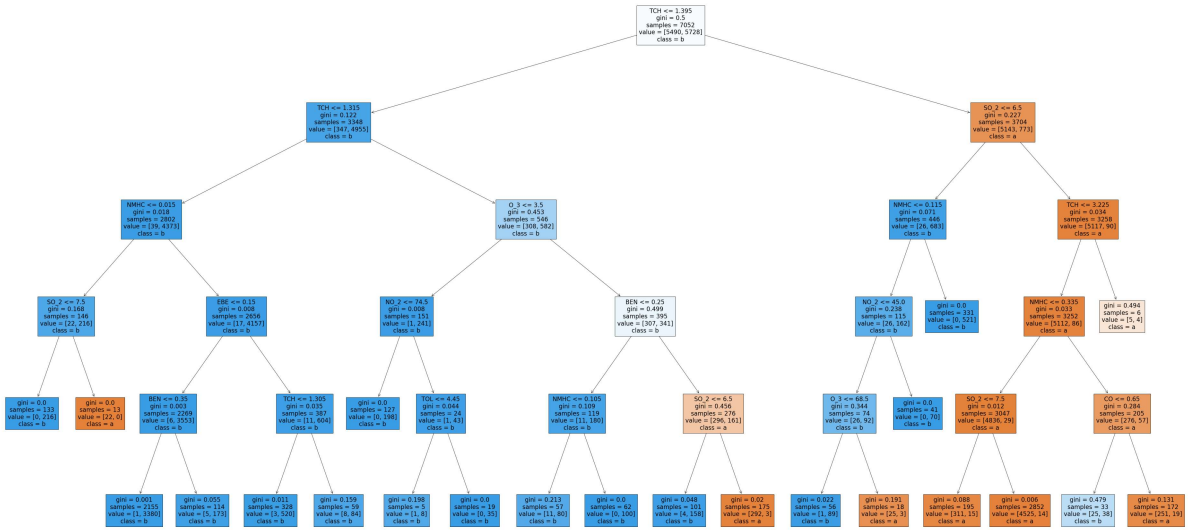
In [61]:
```
rfc_best=grid_search.best_estimator_
```

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(2518.9714285714285, 1993.2, 'TCH <= 1.395\ngini = 0.5\nsamples = 7052\n
value = [5490, 5728]\nclass = b'),
 Text(1275.4285714285716, 1630.8000000000002, 'TCH <= 1.315\ngini = 0.122\nsa
mples = 3348\nvalue = [347, 4955]\nclass = b'),
 Text(573.9428571428572, 1268.4, 'NMHC <= 0.015\ngini = 0.018\nsamples = 2802
\nvalue = [39, 4373]\nclass = b'),
 Text(255.0857142857143, 906.0, 'SO_2 <= 7.5\ngini = 0.168\nsamples = 146\nva
lue = [22, 216]\nclass = b'),
 Text(127.54285714285714, 543.5999999999999, 'gini = 0.0\nsamples = 133\nvalu
e = [0, 216]\nclass = b'),
 Text(382.62857142857143, 543.5999999999999, 'gini = 0.0\nsamples = 13\nvalue
= [22, 0]\nclass = a'),
 Text(892.8, 906.0, 'EBE <= 0.15\ngini = 0.008\nsamples = 2656\nvalue = [17,
4157]\nclass = b'),
 Text(637.7142857142858, 543.5999999999999, 'BEN <= 0.35\ngini = 0.003\nsampl
es = 2269\nvalue = [6, 3553]\nclass = b'),
 Text(510.1714285714286, 181.19999999999982, 'gini = 0.001\nsamples = 2155\nv
alue = [1, 3380]\nclass = b'),
 Text(765.2571428571429, 181.19999999999982, 'gini = 0.055\nsamples = 114\nva
lue = [5, 173]\nclass = b'),
 Text(1147.8857142857144, 543.5999999999999, 'TCH <= 1.305\ngini = 0.035\nsam
ples = 387\nvalue = [11, 604]\nclass = b'),
 Text(1020.3428571428572, 181.19999999999982, 'gini = 0.011\nsamples = 328\nv
alue = [3, 520]\nclass = b'),
 Text(1275.4285714285716, 181.19999999999982, 'gini = 0.159\nsamples = 59\nva
lue = [8, 84]\nclass = b'),
 Text(1976.9142857142858, 1268.4, 'O_3 <= 3.5\ngini = 0.453\nsamples = 546\nv
alue = [308, 582]\nclass = b'),
 Text(1530.5142857142857, 906.0, 'NO_2 <= 74.5\ngini = 0.008\nsamples = 151\n
value = [1, 241]\nclass = b'),
 Text(1402.9714285714285, 543.5999999999999, 'gini = 0.0\nsamples = 127\nvalu
e = [0, 198]\nclass = b'),
 Text(1658.057142857143, 543.5999999999999, 'TOL <= 4.45\ngini = 0.044\nsampl
es = 24\nvalue = [1, 43]\nclass = b'),
 Text(1530.5142857142857, 181.19999999999982, 'gini = 0.198\nsamples = 5\nval
ue = [1, 8]\nclass = b'),
 Text(1785.6, 181.19999999999982, 'gini = 0.0\nsamples = 19\nvalue = [0, 35]
\nclass = b'),
 Text(2423.3142857142857, 906.0, 'BEN <= 0.25\ngini = 0.499\nsamples = 395\nv
alue = [307, 341]\nclass = b'),
 Text(2168.2285714285713, 543.5999999999999, 'NMHC <= 0.105\ngini = 0.109\nsa
mples = 119\nvalue = [11, 180]\nclass = b'),
 Text(2040.6857142857143, 181.19999999999982, 'gini = 0.213\nsamples = 57\nva
lue = [11, 80]\nclass = b'),
 Text(2295.7714285714287, 181.19999999999982, 'gini = 0.0\nsamples = 62\nvalu
e = [0, 100]\nclass = b'),
 Text(2678.4, 543.5999999999999, 'SO_2 <= 6.5\ngini = 0.456\nsamples = 276\nv
alue = [296, 161]\nclass = a'),
 Text(2550.857142857143, 181.19999999999982, 'gini = 0.048\nsamples = 101\nva
lue = [4, 158]\nclass = b'),
 Text(2805.942857142857, 181.19999999999982, 'gini = 0.02\nsamples = 175\nval
ue = [292, 3]\nclass = a'),
 Text(3762.514285714286, 1630.8000000000002, 'SO_2 <= 6.5\ngini = 0.227\nsamp
les = 3704\nvalue = [5143, 773]\nclass = a'),
 Text(3443.657142857143, 1268.4, 'NMHC <= 0.115\ngini = 0.071\nsamples = 446
\nvalue = [26, 683]\nclass = b'),
 Text(3316.114285714286, 906.0, 'NO_2 <= 45.0\ngini = 0.238\nsamples = 115\nv
```

```
alue = [26, 162]\nclass = b'),
 Text(3188.5714285714284, 543.5999999999999, 'O_3 <= 68.5\ngini = 0.344\nsamp
les = 74\nvalue = [26, 92]\nclass = b'),
 Text(3061.0285714285715, 181.19999999999982, 'gini = 0.022\nsamples = 56\nva
lue = [1, 89]\nclass = b'),
 Text(3316.114285714286, 181.19999999999982, 'gini = 0.191\nsamples = 18\nval
ue = [25, 3]\nclass = a'),
 Text(3443.657142857143, 543.5999999999999, 'gini = 0.0\nsamples = 41\nvalue
= [0, 70]\nclass = b'),
 Text(3571.2, 906.0, 'gini = 0.0\nsamples = 331\nvalue = [0, 521]\nclass =
b'),
 Text(4081.3714285714286, 1268.4, 'TCH <= 3.225\ngini = 0.034\nsamples = 3258
\nvalue = [5117, 90]\nclass = a'),
 Text(3953.8285714285716, 906.0, 'NMHC <= 0.335\ngini = 0.033\nsamples = 3252
\nvalue = [5112, 86]\nclass = a'),
 Text(3698.7428571428572, 543.5999999999999, 'SO_2 <= 7.5\ngini = 0.012\nsamp
les = 3047\nvalue = [4836, 29]\nclass = a'),
 Text(3571.2, 181.19999999999982, 'gini = 0.088\nsamples = 195\nvalue = [311,
15]\nclass = a'),
 Text(3826.285714285714, 181.19999999999982, 'gini = 0.006\nsamples = 2852\nv
alue = [4525, 14]\nclass = a'),
 Text(4208.914285714286, 543.5999999999999, 'CO <= 0.65\ngini = 0.284\nsample
s = 205\nvalue = [276, 57]\nclass = a'),
 Text(4081.3714285714286, 181.19999999999982, 'gini = 0.479\nsamples = 33\nva
lue = [25, 38]\nclass = b'),
 Text(4336.457142857143, 181.19999999999982, 'gini = 0.131\nsamples = 172\nva
lue = [251, 19]\nclass = a'),
 Text(4208.914285714286, 906.0, 'gini = 0.494\nsamples = 6\nvalue = [5, 4]\nc
lass = a')]
```



Accuracy

Linear Regression

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

Out[63]: 0.8014642069655231

Ridge Regression

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

Out[64]: 0.7994760584220166

Lasso Regression

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

Out[65]: 0.6430421339351964

ElasticNet Regression

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

Out[66]: 0.7470130026718783

Logistic Regression

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

Out[67]: 0.9947585174092101

Random Forest

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

Out[68]: 0.9941165983241219

Conclusion

Logistic Regression Suitable for this dataset