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_2018.csv")
df
```

Out[2]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	T
0	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	29.0	31.0	NaN	NaN	NaN	2.0	N
1	2018-03-01 01:00:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0	1
2	2018-03-01 01:00:00	0.4	NaN	NaN	0.2	NaN	4.0	41.0	47.0	NaN	NaN	NaN	NaN	N
3	2018-03-01 01:00:00	NaN	NaN	0.3	NaN	NaN	1.0	35.0	37.0	54.0	NaN	NaN	NaN	N
4	2018-03-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	27.0	29.0	49.0	NaN	NaN	3.0	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
69091	2018-02-01 00:00:00	NaN	NaN	0.5	NaN	NaN	66.0	91.0	192.0	1.0	35.0	22.0	NaN	N
69092	2018-02-01 00:00:00	NaN	NaN	0.7	NaN	NaN	87.0	107.0	241.0	NaN	29.0	NaN	15.0	N
69093	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	28.0	48.0	91.0	2.0	NaN	NaN	NaN	N
69094	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	141.0	103.0	320.0	2.0	NaN	NaN	NaN	N
69095	2018-02-01 00:00:00	NaN	NaN	NaN	NaN	NaN	69.0	96.0	202.0	3.0	26.0	NaN	NaN	N

69096 rows × 16 columns

Data Cleaning and Data Preprocessing

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

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

Out[4]:
```
Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4562 entries, 1 to 69078
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     4562 non-null   object 
 1   BEN      4562 non-null   float64
 2   CH4      4562 non-null   float64
 3   CO       4562 non-null   float64
 4   EBE      4562 non-null   float64
 5   NMHC     4562 non-null   float64
 6   NO       4562 non-null   float64
 7   NO_2     4562 non-null   float64
 8   NOx      4562 non-null   float64
 9   O_3      4562 non-null   float64
 10  PM10     4562 non-null   float64
 11  PM25     4562 non-null   float64
 12  SO_2     4562 non-null   float64
 13  TCH      4562 non-null   float64
 14  TOL      4562 non-null   float64
 15  station  4562 non-null   int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 605.9+ KB
```

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

Out[6]:

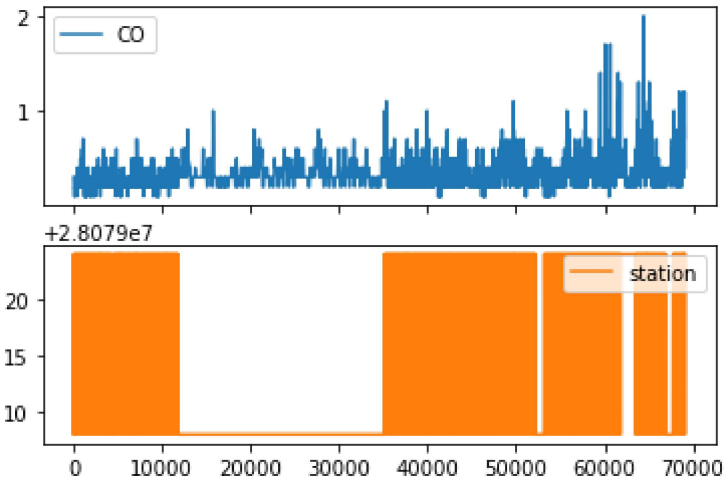
	CO	station
1	0.3	28079008
6	0.2	28079024
25	0.2	28079008
30	0.2	28079024
49	0.2	28079008
...	...	...
69030	0.7	28079024
69049	1.2	28079008
69054	0.6	28079024
69073	1.0	28079008
69078	0.4	28079024

4562 rows × 2 columns

Line chart

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

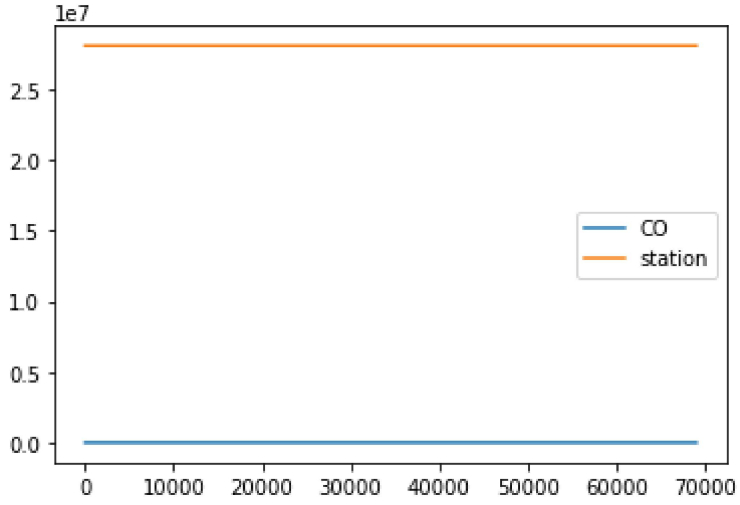
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

Out[8]: <AxesSubplot:>

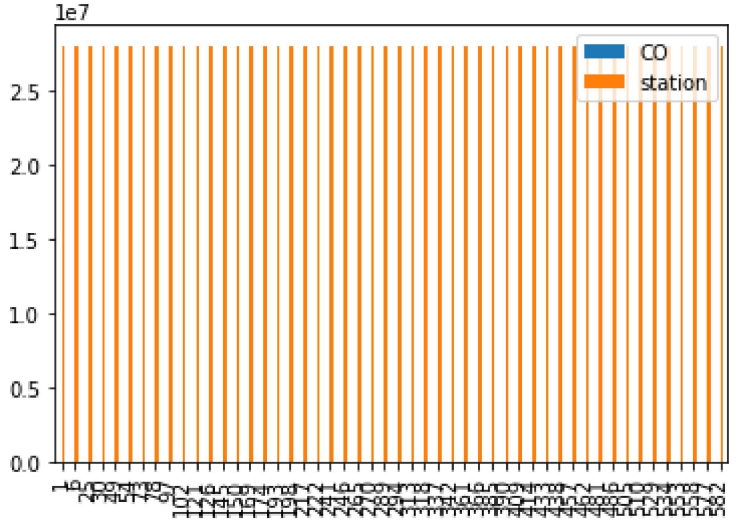


Bar chart

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

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

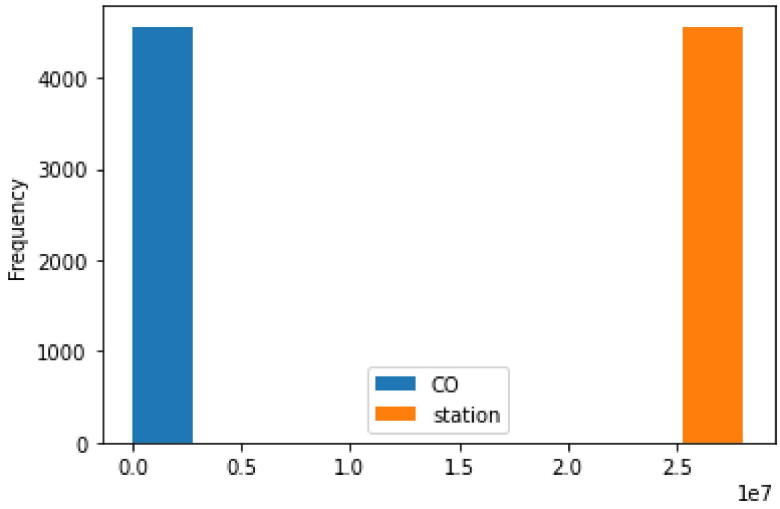
Out[10]: <AxesSubplot:>



Histogram

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

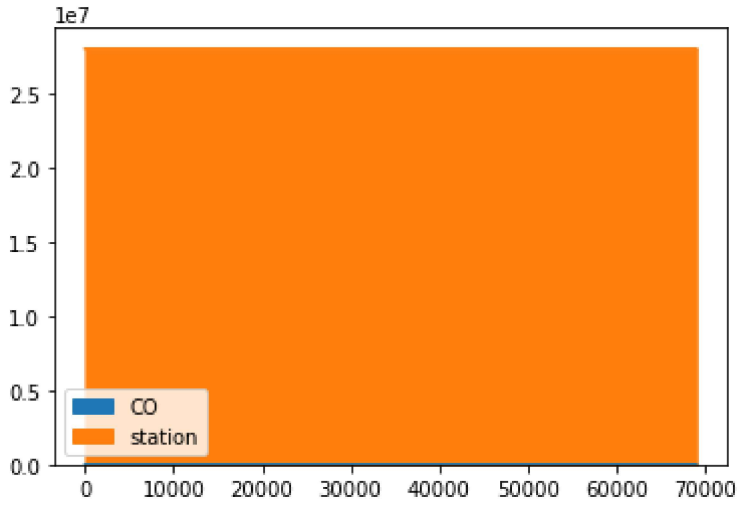
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

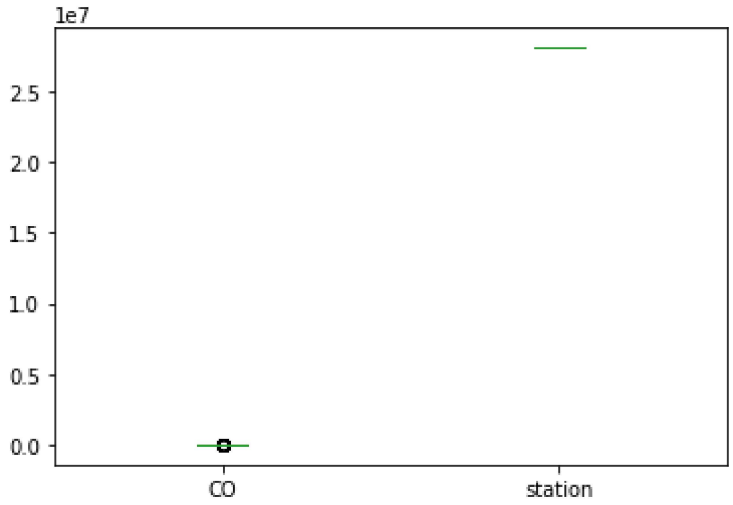
Out[12]: <AxesSubplot:>



Box chart

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

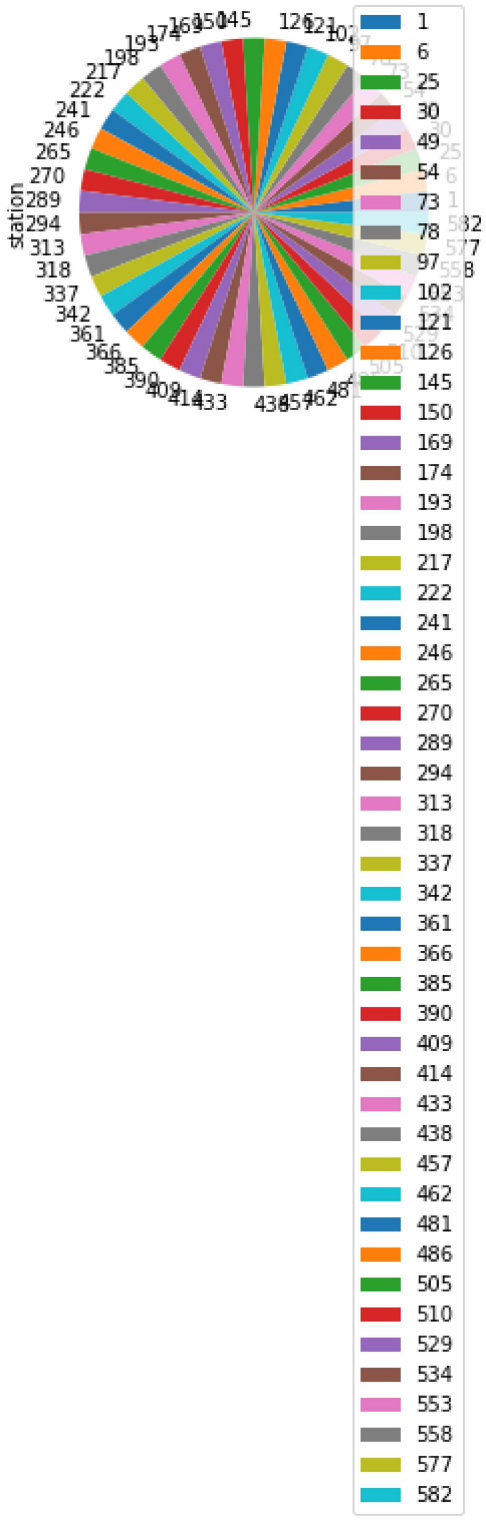
Out[13]:
```
<AxesSubplot:>
```



Pie chart

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

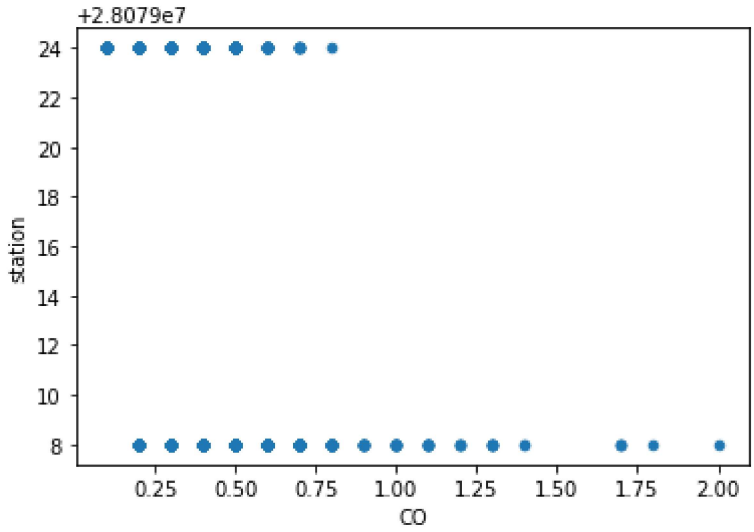
Out[14]: <AxesSubplot:ylabel='station'>



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: 4562 entries, 1 to 69078
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     4562 non-null   object
 1   BEN      4562 non-null   float64
 2   CH4      4562 non-null   float64
 3   CO       4562 non-null   float64
 4   EBE      4562 non-null   float64
 5   NMHC     4562 non-null   float64
 6   NO       4562 non-null   float64
 7   NO_2     4562 non-null   float64
 8   NOx      4562 non-null   float64
 9   O_3      4562 non-null   float64
 10  PM10     4562 non-null   float64
 11  PM25     4562 non-null   float64
 12  SO_2     4562 non-null   float64
 13  TCH      4562 non-null   float64
```

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

Out[17]:

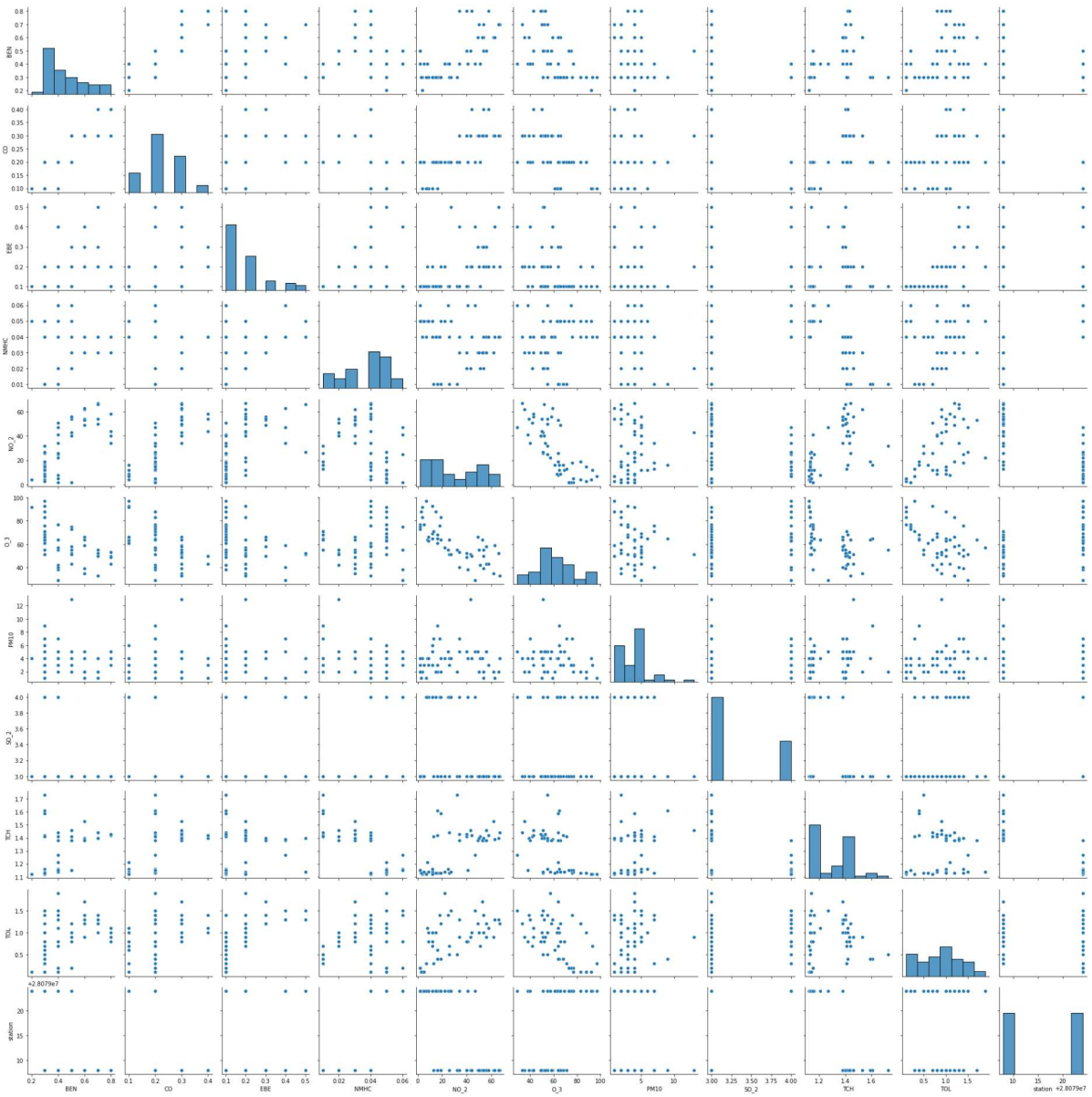
	BEN	CH4	CO	EBE	NMHC	NO	NO_2
count	4562.00000	4562.000000	4562.000000	4562.000000	4562.000000	4562.000000	4562.000000
mean	0.69349	1.329163	0.330579	0.286782	0.056773	21.742218	44.152126
std	0.46832	0.214399	0.161489	0.354442	0.037711	35.539531	30.234015
min	0.10000	0.020000	0.100000	0.100000	0.000000	1.000000	1.000000
25%	0.40000	1.120000	0.200000	0.100000	0.030000	1.000000	20.000000
50%	0.60000	1.390000	0.300000	0.200000	0.050000	9.000000	41.000000
75%	0.90000	1.420000	0.400000	0.300000	0.070000	27.000000	64.000000
max	6.60000	3.920000	2.000000	7.400000	0.490000	431.000000	184.000000

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])
```

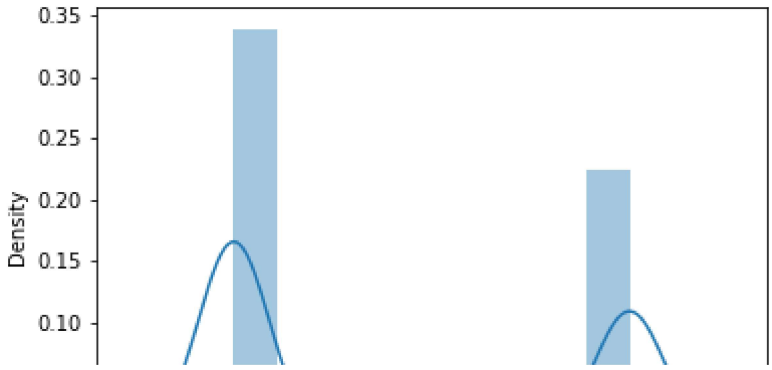
Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x1c398da5f40>
```



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

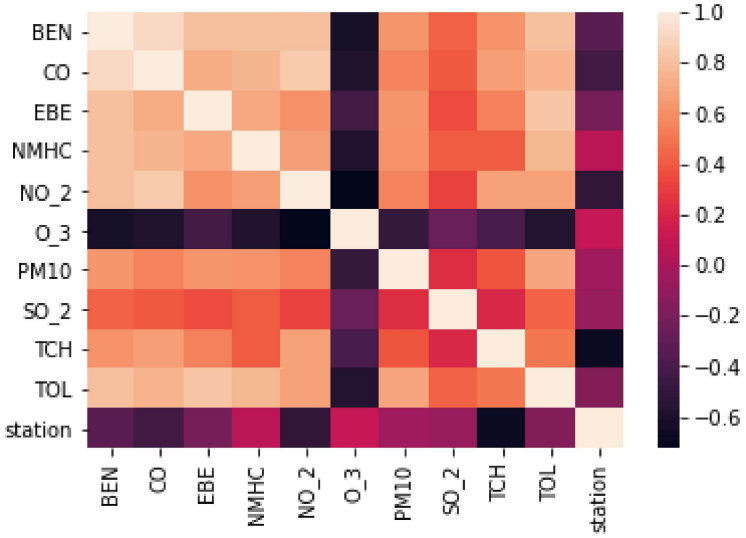
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
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]:
```
28079043.077610146
```

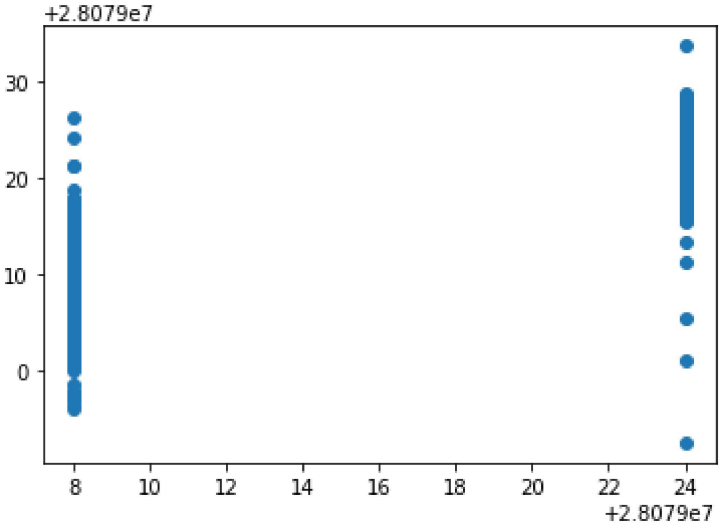
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-0.900528
CO	-17.574130
EBE	0.658867
NMHC	157.307734
NO_2	-0.159591
O_3	-0.084248
PM10	0.108194
SO_2	-0.073791
TCH	-15.489496
TOL	-0.131789

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

Out[27]:
```
<matplotlib.collections.PathCollection at 0x1c3a0170a60>
```



ACCURACY

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

Out[28]:
```
0.7824772546668642
```

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

Out[29]:
```
0.8144965924867783
```

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]:
```
0.6595624273972378
```

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

Out[33]:
```
0.6882866952427864
```

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]:
```
0.42703522156155893
```

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

Out[36]:
```
0.3736038180074345
```

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.        , -0.27014363,
       -0.14760838,  0.24707476,  0.07711933, -0.04893418,  0.03587408])
```

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

Out[39]:
```
28079029.42740041
```

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

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

Out[41]:
```
0.42085331346280364
```

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)))
```

```
5.104064504162206
35.41358840461507
5.950931725756486
```

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]:
```
(4562, 10)
```

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

Out[46]:
```
(4562,)
```

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]:
```
0.9888206926786497
```

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

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

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

Out[55]:
```
array([[1.00000000e+00, 1.42669593e-19]])
```

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]:
```
0.9924829685359297
```

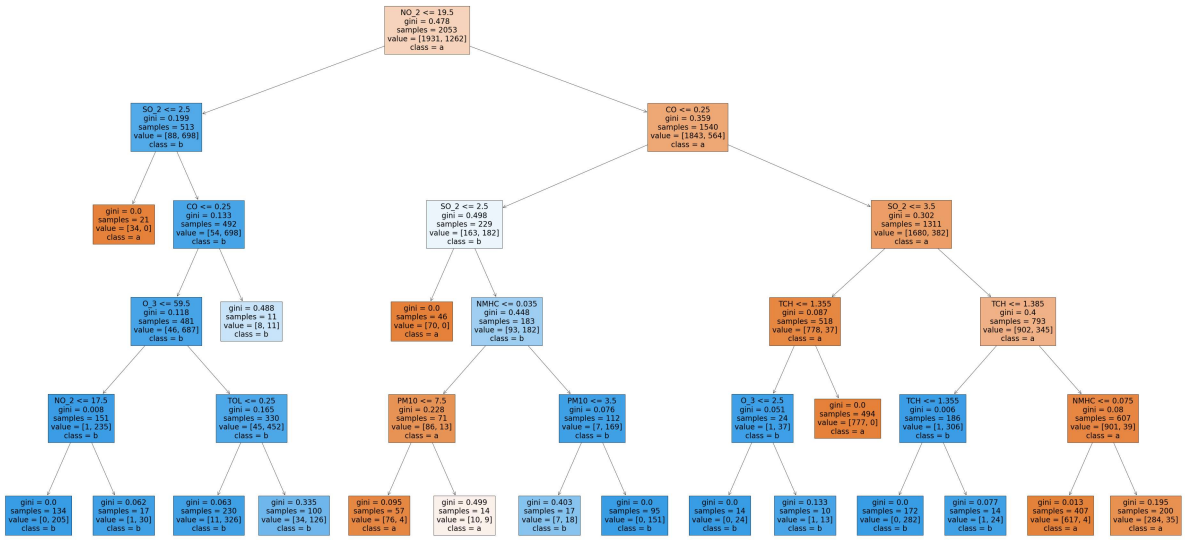
In [61]:
```
rfc_best=grid_search.best_estimator_
```

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(1614.2142857142856, 1993.2, 'NO_2 <= 19.5\ngini = 0.478\nsamples = 2053
\nvalue = [1931, 1262]\nclass = a'),
 Text(637.7142857142857, 1630.8000000000002, 'SO_2 <= 2.5\ngini = 0.199\nsamp
les = 513\nvalue = [88, 698]\nclass = b'),
 Text(478.2857142857142, 1268.4, 'gini = 0.0\nsamples = 21\nvalue = [34, 0]\n
class = a'),
 Text(797.1428571428571, 1268.4, 'CO <= 0.25\ngini = 0.133\nsamples = 492\nva
lue = [54, 698]\nclass = b'),
 Text(637.7142857142857, 906.0, 'O_3 <= 59.5\ngini = 0.118\nsamples = 481\nva
lue = [46, 687]\nclass = b'),
 Text(318.85714285714283, 543.5999999999999, 'NO_2 <= 17.5\ngini = 0.008\nsam
ples = 151\nvalue = [1, 235]\nclass = b'),
 Text(159.42857142857142, 181.19999999999982, 'gini = 0.0\nsamples = 134\nval
ue = [0, 205]\nclass = b'),
 Text(478.2857142857142, 181.19999999999982, 'gini = 0.062\nsamples = 17\nval
ue = [1, 30]\nclass = b'),
 Text(956.5714285714284, 543.5999999999999, 'TOL <= 0.25\ngini = 0.165\nsampl
es = 330\nvalue = [45, 452]\nclass = b'),
 Text(797.1428571428571, 181.19999999999982, 'gini = 0.063\nsamples = 230\nva
lue = [11, 326]\nclass = b'),
 Text(1116.0, 181.19999999999982, 'gini = 0.335\nsamples = 100\nvalue = [34,
126]\nclass = b'),
 Text(956.5714285714284, 906.0, 'gini = 0.488\nsamples = 11\nvalue = [8, 11]
\nclass = b'),
 Text(2590.7142857142853, 1630.8000000000002, 'CO <= 0.25\ngini = 0.359\nsamp
les = 1540\nvalue = [1843, 564]\nclass = a'),
 Text(1753.7142857142856, 1268.4, 'SO_2 <= 2.5\ngini = 0.498\nsamples = 229\n
value = [163, 182]\nclass = b'),
 Text(1594.2857142857142, 906.0, 'gini = 0.0\nsamples = 46\nvalue = [70, 0]\n
class = a'),
 Text(1913.1428571428569, 906.0, 'NMHC <= 0.035\ngini = 0.448\nsamples = 183
\nvalue = [93, 182]\nclass = b'),
 Text(1594.2857142857142, 543.5999999999999, 'PM10 <= 7.5\ngini = 0.228\nsamp
les = 71\nvalue = [86, 13]\nclass = a'),
 Text(1434.8571428571427, 181.19999999999982, 'gini = 0.095\nsamples = 57\nva
lue = [76, 4]\nclass = a'),
 Text(1753.7142857142856, 181.19999999999982, 'gini = 0.499\nsamples = 14\nva
lue = [10, 9]\nclass = a'),
 Text(2232.0, 543.5999999999999, 'PM10 <= 3.5\ngini = 0.076\nsamples = 112\nv
alue = [7, 169]\nclass = b'),
 Text(2072.5714285714284, 181.19999999999982, 'gini = 0.403\nsamples = 17\nva
lue = [7, 18]\nclass = b'),
 Text(2391.428571428571, 181.19999999999982, 'gini = 0.0\nsamples = 95\nvalue
= [0, 151]\nclass = b'),
 Text(3427.7142857142853, 1268.4, 'SO_2 <= 3.5\ngini = 0.302\nsamples = 1311
\nvalue = [1680, 382]\nclass = a'),
 Text(3029.142857142857, 906.0, 'TCH <= 1.355\ngini = 0.087\nsamples = 518\nv
alue = [778, 37]\nclass = a'),
 Text(2869.7142857142853, 543.5999999999999, 'O_3 <= 2.5\ngini = 0.051\nsampl
es = 24\nvalue = [1, 37]\nclass = b'),
 Text(2710.285714285714, 181.19999999999982, 'gini = 0.0\nsamples = 14\nvalue
= [0, 24]\nclass = b'),
 Text(3029.142857142857, 181.19999999999982, 'gini = 0.133\nsamples = 10\nval
ue = [1, 13]\nclass = b'),
 Text(3188.5714285714284, 543.5999999999999, 'gini = 0.0\nsamples = 494\nvalu
e = [777, 0]\nclass = a'),
 Text(3826.2857142857138, 906.0, 'TCH <= 1.385\ngini = 0.4\nsamples = 793\nva
```

```
lue = [902, 345]\nclass = a'),
 Text(3507.428571428571, 543.5999999999999, 'TCH <= 1.355\ngini = 0.006\nsamp
les = 186\nvalue = [1, 306]\nclass = b'),
 Text(3347.9999999999995, 181.19999999999982, 'gini = 0.0\nsamples = 172\nval
ue = [0, 282]\nclass = b'),
 Text(3666.8571428571427, 181.19999999999982, 'gini = 0.077\nsamples = 14\nva
lue = [1, 24]\nclass = b'),
 Text(4145.142857142857, 543.5999999999999, 'NMHC <= 0.075\ngini = 0.08\nsamp
les = 607\nvalue = [901, 39]\nclass = a'),
 Text(3985.7142857142853, 181.19999999999982, 'gini = 0.013\nsamples = 407\nv
alue = [617, 4]\nclass = a'),
 Text(4304.571428571428, 181.19999999999982, 'gini = 0.195\nsamples = 200\nva
lue = [284, 35]\nclass = a')]
```



Accuracy

Linear Regression

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

Out[63]: 0.8144965924867783

Ridge Regression

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

Out[64]: 0.6882866952427864

Lasso Regression

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

Out[65]:
```
0.3736038180074345
```

ElasticNet Regression

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

Out[66]:
```
0.42085331346280364
```

Logistic Regression

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

Out[67]:
```
0.9888206926786497
```

Random Forest

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

Out[68]:
```
0.9924829685359297
```

Conclusion

RandomForest for this dataset