Importing Libraries

In [120]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing Datasets

In [121]:
```
df=pd.read_csv("madrid_2012.csv")
df
```

Out[121]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	7.0	18.0	NaN	NaN	NaN	2.0	NaN	NaN	28
1	2012-09-01 01:00:00	0.3	0.3	0.7	NaN	3.0	18.0	55.0	10.0	9.0	1.0	NaN	2.4	28
2	2012-09-01 01:00:00	0.4	NaN	0.7	NaN	2.0	10.0	NaN	NaN	NaN	NaN	NaN	1.5	28
3	2012-09-01 01:00:00	NaN	0.2	NaN	NaN	1.0	6.0	50.0	NaN	NaN	NaN	NaN	NaN	28
4	2012-09-01 01:00:00	NaN	NaN	NaN	NaN	1.0	13.0	54.0	NaN	NaN	3.0	NaN	NaN	28
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
210715	2012-03-01 00:00:00	NaN	0.6	NaN	NaN	37.0	84.0	14.0	NaN	NaN	NaN	NaN	NaN	28
210716	2012-03-01 00:00:00	NaN	0.4	NaN	NaN	5.0	76.0	NaN	17.0	NaN	7.0	NaN	NaN	28
210717	2012-03-01 00:00:00	NaN	NaN	NaN	0.34	3.0	41.0	24.0	NaN	NaN	NaN	1.34	NaN	28
210718	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	44.0	36.0	NaN	NaN	NaN	NaN	NaN	28
210719	2012-03-01 00:00:00	NaN	NaN	NaN	NaN	2.0	56.0	40.0	18.0	NaN	NaN	NaN	NaN	28

210720 rows × 14 columns

Data Cleaning and Data Preprocessing

In [122]:
```
df=df.dropna()
```

In [123]:
```
df.columns
```

Out[123]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [124]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10916 entries, 6 to 210702
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     10916 non-null  object 
 1   BEN      10916 non-null  float64
 2   CO       10916 non-null  float64
 3   EBE      10916 non-null  float64
 4   NMHC     10916 non-null  float64
 5   NO       10916 non-null  float64
 6   NO_2     10916 non-null  float64
 7   O_3      10916 non-null  float64
 8   PM10     10916 non-null  float64
 9   PM25     10916 non-null  float64
 10  SO_2     10916 non-null  float64
 11  TCH      10916 non-null  float64
 12  TOL      10916 non-null  float64
 13  station  10916 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.2+ MB
```

In [125]:
```
data=df[['CO' ,'station']]
data
```

Out[125]:

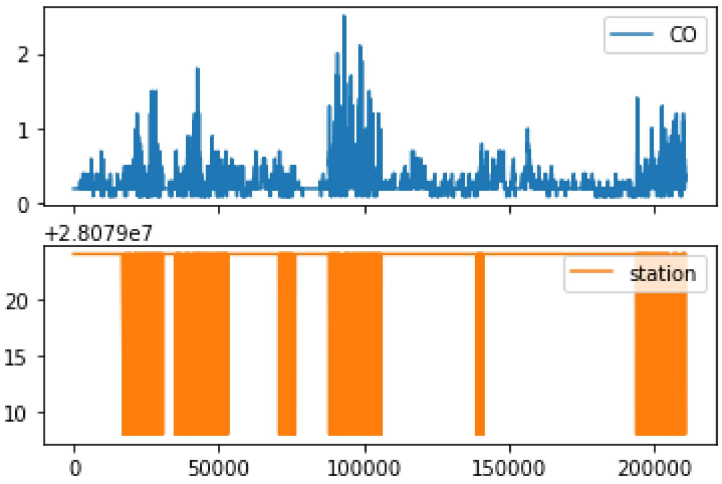
	CO	station
6	0.2	28079024
30	0.2	28079024
54	0.2	28079024
78	0.2	28079024
102	0.2	28079024
...	...	...
210654	0.3	28079024
210673	0.4	28079008
210678	0.3	28079024
210697	0.4	28079008
210702	0.3	28079024

10916 rows × 2 columns

Line chart

In [126]:
```
data.plot.line(subplots=True)
```

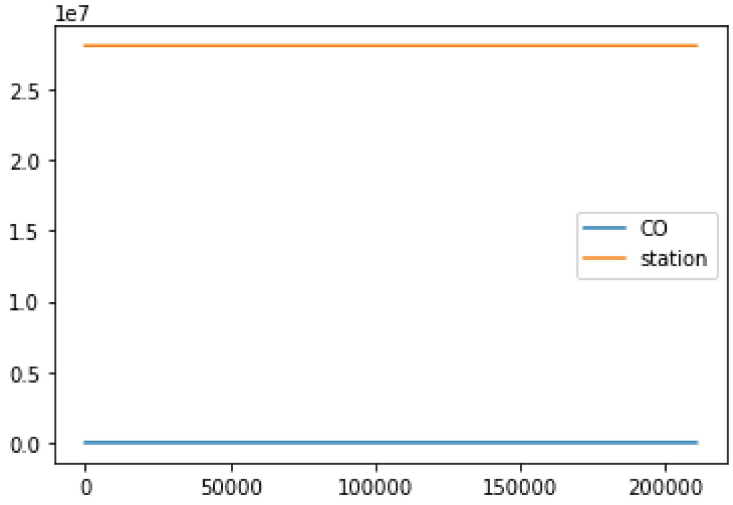
Out[126]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [127]:
```
data.plot.line()
```

Out[127]: <AxesSubplot:>

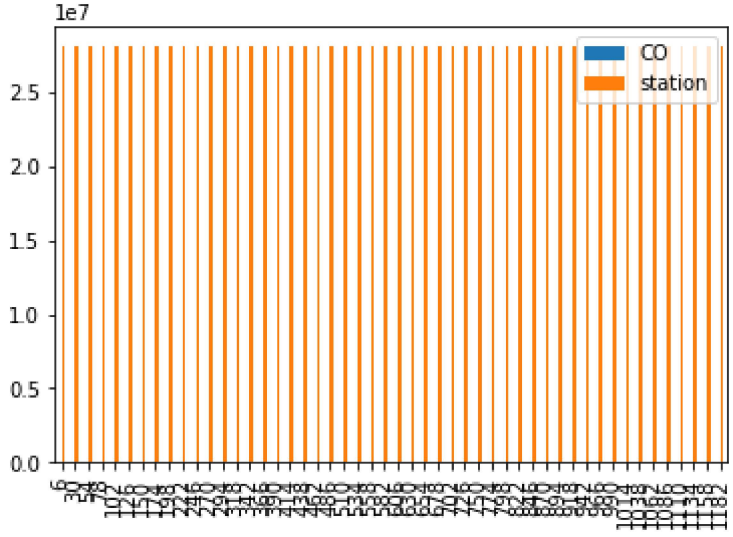


Bar chart

In [128]:
```
b=data[0:50]
```

In [129]:
```
b.plot.bar()
```

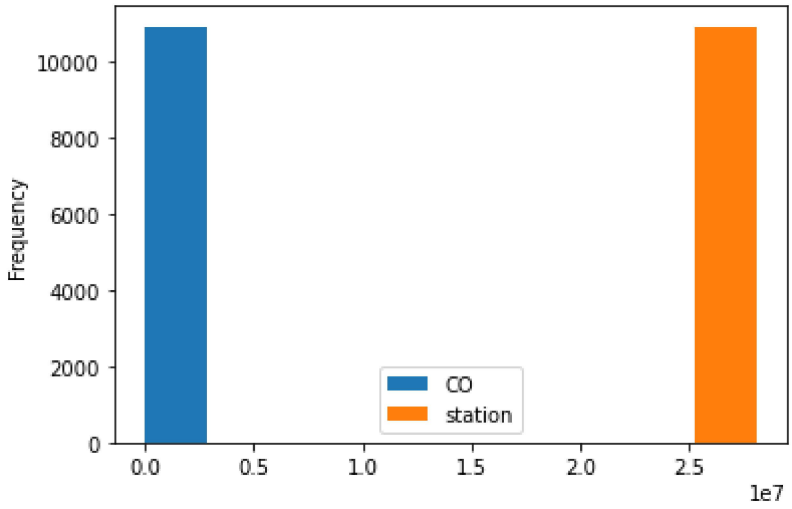
Out[129]: <AxesSubplot:>



Histogram

In [130]:
```
data.plot.hist()
```

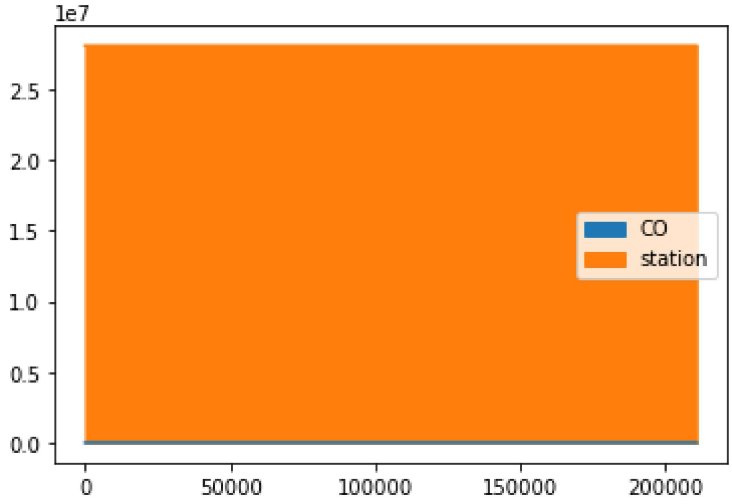
Out[130]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [131]:
```
data.plot.area()
```

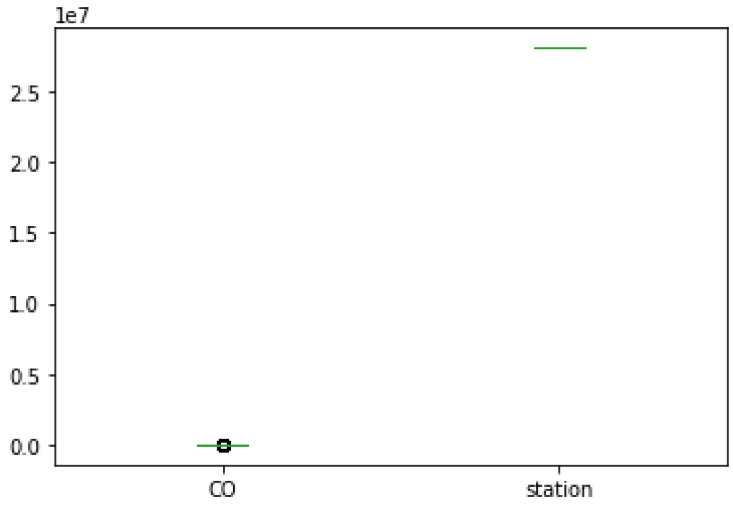
Out[131]: <AxesSubplot:>



Box chart

In [132]:
```
data.plot.box()
```

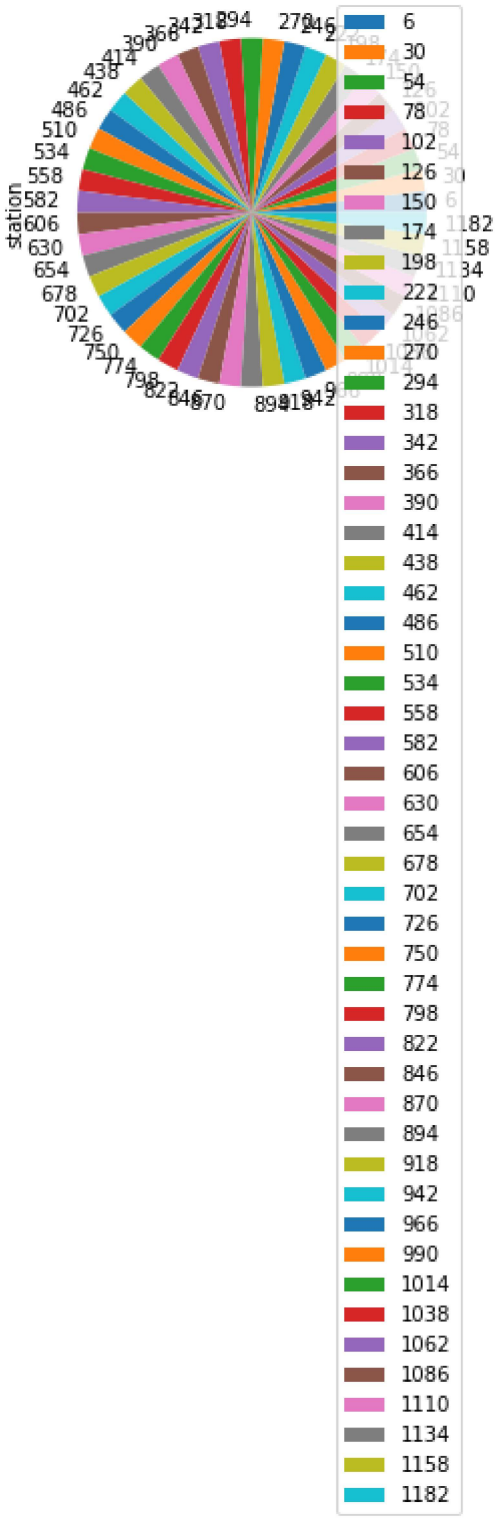
Out[132]: <AxesSubplot:>



Pie chart

In [133]:
```
b.plot.pie(y='station' )
```

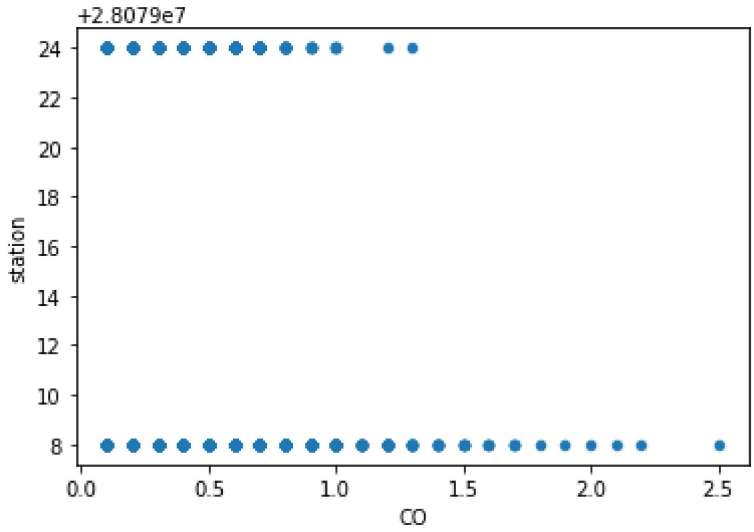
Out[133]: <AxesSubplot:ylabel='station'>



Scatter chart

In [134]:
```
data.plot.scatter(x='CO' ,y='station')
```

Out[134]: <AxesSubplot:xlabel='CO', ylabel='station'>



In [135]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10916 entries, 6 to 210702
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     10916 non-null  object 
 1   BEN      10916 non-null  float64
 2   CO       10916 non-null  float64
 3   EBE      10916 non-null  float64
 4   NMHC     10916 non-null  float64
 5   NO       10916 non-null  float64
 6   NO_2     10916 non-null  float64
 7   O_3      10916 non-null  float64
 8   PM10     10916 non-null  float64
 9   PM25     10916 non-null  float64
 10  SO_2     10916 non-null  float64
 11  TCH      10916 non-null  float64
 12  TOL      10916 non-null  float64
 13  station  10916 non-null  int64  
dtypes: float64(12), int64(1), object(1)
```

In [136]:
```
df.describe()
```

Out[136]:

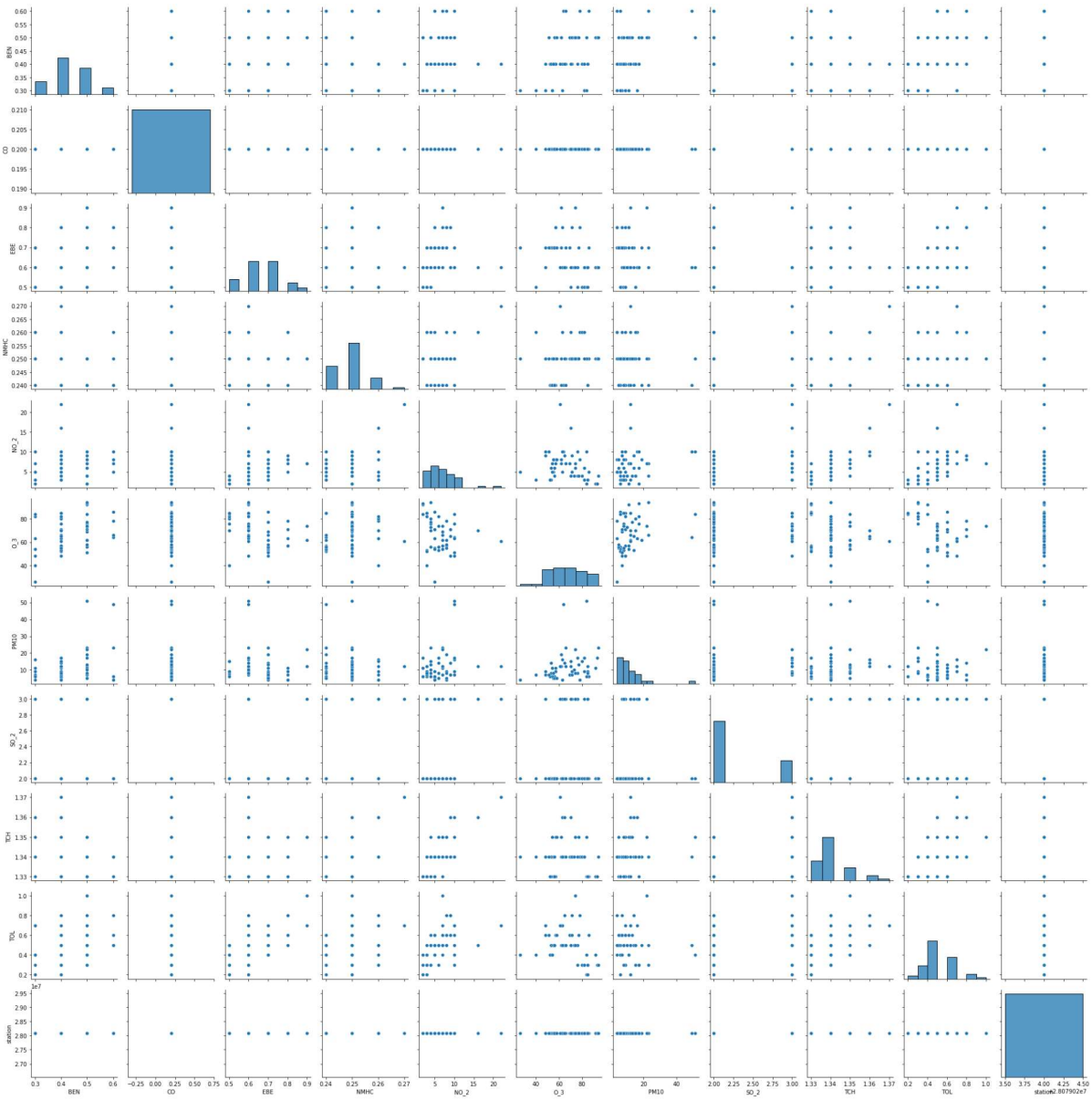
	BEN	CO	EBE	NMHC	NO	NO_2	
count	10916.000000	10916.000000	10916.000000	10916.000000	10916.000000	10916.000000	109
mean	0.784014	0.279333	0.992213	0.215755	18.795529	31.262642	
std	0.632755	0.167922	0.804554	0.075169	40.038872	27.234732	
min	0.100000	0.100000	0.100000	0.050000	0.000000	1.000000	
25%	0.400000	0.200000	0.500000	0.160000	1.000000	9.000000	
50%	0.600000	0.200000	0.800000	0.220000	3.000000	24.000000	
75%	0.900000	0.300000	1.200000	0.250000	18.000000	47.000000	
max	7.000000	2.500000	9.700000	0.670000	525.000000	225.000000	15

In [137]:
```
df1=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [138]:
```
sns.pairplot(df1[0:50])
```

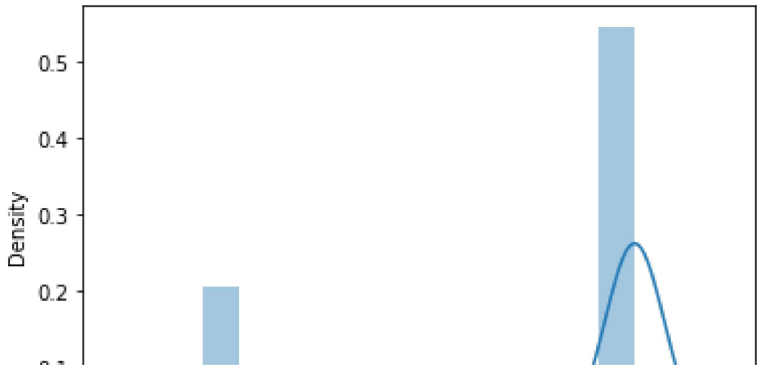
Out[138]: <seaborn.axisgrid.PairGrid at 0x21d7afe46d0>



In [139]:
```
sns.distplot(df1['station'])
```

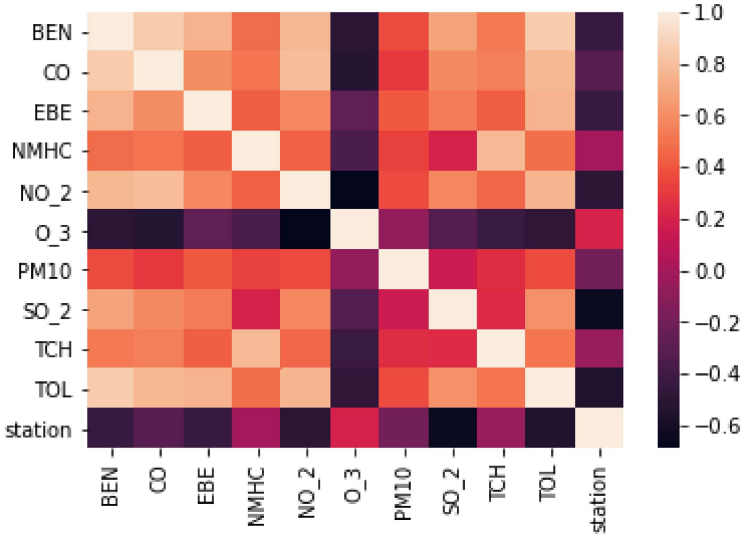
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio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

Out[139]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [140]:
```
sns.heatmap(df1.corr())
```

Out[140]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [141]:
```
x=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [142]:
```
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 [143]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[143]:
```
LinearRegression()
```

In [144]:
```
lr.intercept_
```

Out[144]:
```
28079019.526728842
```

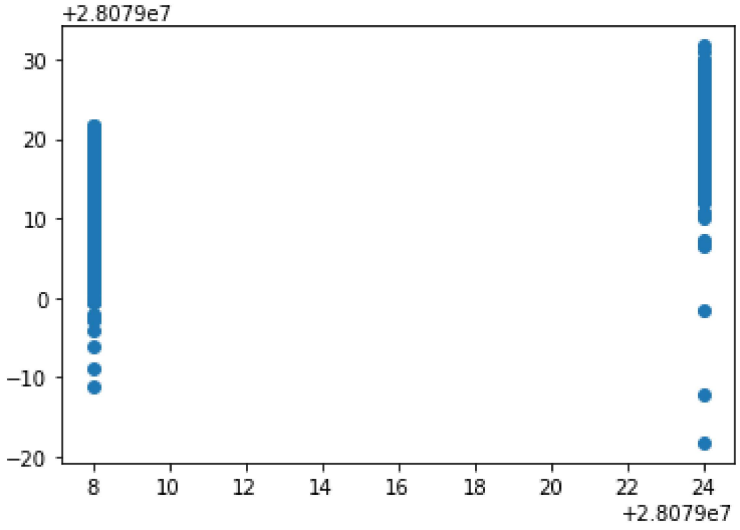
In [145]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[145]:

	Co-efficient
BEN	3.751643
CO	16.649119
EBE	-0.336712
NMHC	18.697049
NO_2	-0.115773
O_3	-0.028826
PM10	-0.009855
SO_2	-0.698860
TCH	0.568084
TOL	-1.478321

In [146]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[146]: <matplotlib.collections.PathCollection at 0x21d03255820>



ACCURACY

In [147]:
```
lr.score(x_test,y_test)
```

Out[147]: 0.6147514978938244

In [148]:
```
lr.score(x_train,y_train)
```

Out[148]: 0.6247137686462515

Ridge and Lasso

In [149]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [150]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[150]: Ridge(alpha=10)

Accuracy(Ridge)

In [151]:
```
rr.score(x_test,y_test)
```

Out[151]: 0.6129605549840603

In [152]:
```
rr.score(x_train,y_train)
```

Out[152]:
```
0.621965644818891
```

Accuracy(Lasso)

In [153]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[153]:
```
Lasso(alpha=10)
```

In [154]:
```
la.score(x_train,y_train)
```

Out[154]:
```
0.3692715607901955
```

ElasticNet

In [155]:
```
la.score(x_test,y_test)
```

Out[155]:
```
0.3555875585759102
```

In [156]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[156]:
```
ElasticNet()
```

In [157]:
```
en.coef_
```

Out[157]:
```
array([ 0.        ,  0.        ,  0.        ,  0.        , -0.06330338,
       -0.04022778,  0.00386877, -0.58130524,  0.        , -0.33620677])
```

In [158]:
```
en.intercept_
```

Out[158]:
```
28079026.81181796
```

In [159]:
```
prediction=en.predict(x_test)
```

In [160]:
```
en.score(x_test,y_test)
```

Out[160]:
```
0.4650923677984724
```

Evaluation Metrics

In [161]:
```
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)))
```

```
3.8288268151196814
27.308739633004265
5.225776462211551
```

Logistic Regression

In [162]:
```
from sklearn.linear_model import LogisticRegression
```

In [163]:
```
feature_matrix=df[['BEN', 'CO', 'EBE',  'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [164]:
```
feature_matrix.shape
```

Out[164]:
```
(10916, 10)
```

In [165]:
```
target_vector.shape
```

Out[165]:
```
(10916,)
```

In [166]:
```
from sklearn.preprocessing import StandardScaler
```

In [167]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [168]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[168]:
```
LogisticRegression(max_iter=10000)
```

In [169]:
```
observation=[[1,2,3,4,5,6,7,8,9,10]]
```

In [170]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079008]
```

In [171]:
```
logr.classes_
```

Out[171]:
```
array([28079008, 28079024], dtype=int64)
```

In [172]:
```
logr.score(fs,target_vector)
```

Out[172]:
```
0.9293697325027482
```

In [173]:
```
logr.predict_proba(observation)[0][0]
```

Out[173]:
```
1.0
```

In [174]:
```
logr.predict_proba(observation)
```

Out[174]:
```
array([[1.00000000e+00, 3.50349553e-26]])
```

Random Forest

In [175]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [176]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[176]:
```
RandomForestClassifier()
```

In [177]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [178]:
```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```

Out[178]:
```
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 [179]:
```
grid_search.best_score_
```

Out[179]:
```
0.9602150077211771
```

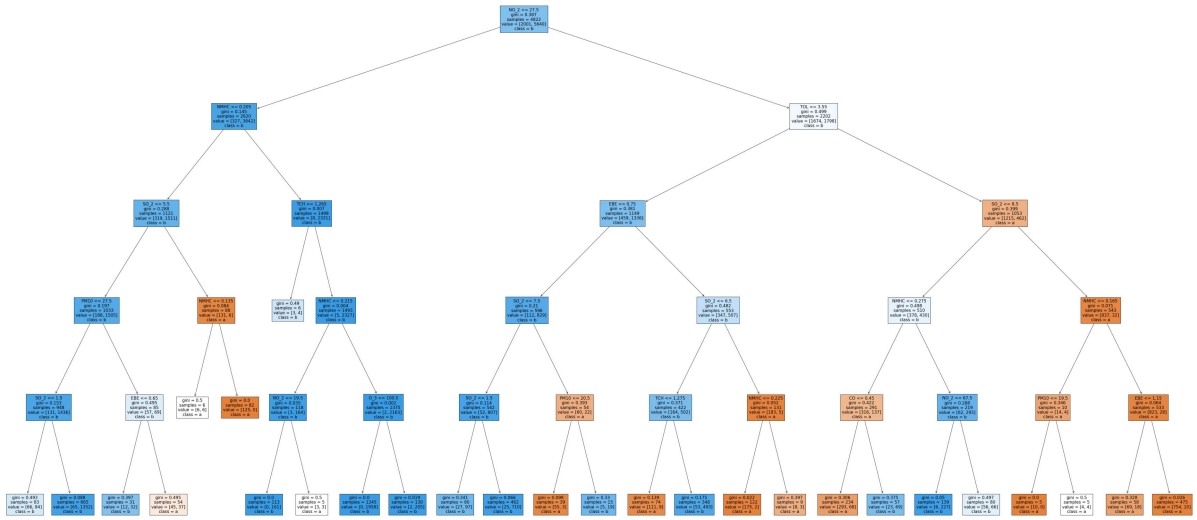
In [180]:
```
rfc_best=grid_search.best_estimator_
```

In [181]:
```
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[181]:
```
[Text(1953.0, 1993.2, 'NO_2 <= 27.5\ngini = 0.387\nsamples = 4822\nvalue = [2
001, 5640]\nclass = b'),
 Text(870.48, 1630.8000000000002, 'NMHC <= 0.205\ngini = 0.145\nsamples = 262
0\nvalue = [327, 3842]\nclass = b'),
 Text(580.32, 1268.4, 'SO_2 <= 5.5\ngini = 0.288\nsamples = 1121\nvalue = [31
9, 1511]\nclass = b'),
 Text(357.12, 906.0, 'PM10 <= 27.5\ngini = 0.197\nsamples = 1033\nvalue = [18
8, 1505]\nclass = b'),
 Text(178.56, 543.5999999999999, 'SO_2 <= 1.5\ngini = 0.153\nsamples = 948\nv
alue = [131, 1436]\nclass = b'),
 Text(89.28, 181.19999999999982, 'gini = 0.493\nsamples = 83\nvalue = [66, 8
4]\nclass = b'),
 Text(267.84000000000003, 181.19999999999982, 'gini = 0.088\nsamples = 865\nv
alue = [65, 1352]\nclass = b'),
 Text(535.6800000000001, 543.5999999999999, 'EBE <= 0.65\ngini = 0.495\nsampl
es = 85\nvalue = [57, 69]\nclass = b'),
 Text(446.4, 181.19999999999982, 'gini = 0.397\nsamples = 31\nvalue = [12, 3
2]\nclass = b'),
 Text(624.96, 181.19999999999982, 'gini = 0.495\nsamples = 54\nvalue = [45, 3
7]\nclass = a'),
 Text(803.52, 906.0, 'NMHC <= 0.135\ngini = 0.084\nsamples = 88\nvalue = [13
1, 6]\nclass = a'),
 Text(714.24, 543.5999999999999, 'gini = 0.5\nsamples = 6\nvalue = [6, 6]\ncl
ass = a'),
 Text(892.8, 543.5999999999999, 'gini = 0.0\nsamples = 82\nvalue = [125, 0]\n
class = a'),
 Text(1160.64, 1268.4, 'TCH <= 1.265\ngini = 0.007\nsamples = 1499\nvalue =
[8, 2331]\nclass = b'),
 Text(1071.3600000000001, 906.0, 'gini = 0.49\nsamples = 6\nvalue = [3, 4]\nc
lass = b'),
 Text(1249.92, 906.0, 'NMHC <= 0.215\ngini = 0.004\nsamples = 1493\nvalue =
[5, 2327]\nclass = b'),
 Text(1071.3600000000001, 543.5999999999999, 'NO_2 <= 19.5\ngini = 0.035\nsam
ples = 118\nvalue = [3, 164]\nclass = b'),
 Text(982.08, 181.19999999999982, 'gini = 0.0\nsamples = 113\nvalue = [0, 16
1]\nclass = b'),
 Text(1160.64, 181.19999999999982, 'gini = 0.5\nsamples = 5\nvalue = [3, 3]\n
class = a'),
 Text(1428.48, 543.5999999999999, 'O_3 <= 100.5\ngini = 0.002\nsamples = 1375
\nvalue = [2, 2163]\nclass = b'),
 Text(1339.2, 181.19999999999982, 'gini = 0.0\nsamples = 1245\nvalue = [0, 19
58]\nclass = b'),
 Text(1517.76, 181.19999999999982, 'gini = 0.019\nsamples = 130\nvalue = [2,
205]\nclass = b'),
 Text(3035.52, 1630.8000000000002, 'TOL <= 3.55\ngini = 0.499\nsamples = 2202
\nvalue = [1674, 1798]\nclass = b'),
 Text(2321.28, 1268.4, 'EBE <= 0.75\ngini = 0.381\nsamples = 1149\nvalue = [4
59, 1336]\nclass = b'),
 Text(1964.16, 906.0, 'SO_2 <= 7.5\ngini = 0.21\nsamples = 596\nvalue = [112,
829]\nclass = b'),
 Text(1785.6, 543.5999999999999, 'SO_2 <= 1.5\ngini = 0.114\nsamples = 542\nv
alue = [52, 807]\nclass = b'),
 Text(1696.32, 181.19999999999982, 'gini = 0.341\nsamples = 80\nvalue = [27,
97]\nclass = b'),
 Text(1874.88, 181.19999999999982, 'gini = 0.066\nsamples = 462\nvalue = [25,
710]\nclass = b'),
 Text(2142.7200000000003, 543.5999999999999, 'PM10 <= 20.5\ngini = 0.393\nsam
```

```
ples = 54\nvalue = [60, 22]\nclass = a'),
 Text(2053.44, 181.19999999999982, 'gini = 0.098\nsamples = 39\nvalue = [55,
3]\nclass = a'),
 Text(2232.0, 181.19999999999982, 'gini = 0.33\nsamples = 15\nvalue = [5, 19]
\nclass = b'),
 Text(2678.4, 906.0, 'SO_2 <= 6.5\ngini = 0.482\nsamples = 553\nvalue = [347,
507]\nclass = b'),
 Text(2499.84, 543.5999999999999, 'TCH <= 1.275\ngini = 0.371\nsamples = 422
\nvalue = [164, 502]\nclass = b'),
 Text(2410.56, 181.19999999999982, 'gini = 0.139\nsamples = 74\nvalue = [111,
9]\nclass = a'),
 Text(2589.12, 181.19999999999982, 'gini = 0.175\nsamples = 348\nvalue = [53,
493]\nclass = b'),
 Text(2856.96, 543.5999999999999, 'NMHC <= 0.225\ngini = 0.052\nsamples = 131
\nvalue = [183, 5]\nclass = a'),
 Text(2767.68, 181.19999999999982, 'gini = 0.022\nsamples = 122\nvalue = [17
5, 2]\nclass = a'),
 Text(2946.2400000000002, 181.19999999999982, 'gini = 0.397\nsamples = 9\nval
ue = [8, 3]\nclass = a'),
 Text(3749.76, 1268.4, 'SO_2 <= 8.5\ngini = 0.399\nsamples = 1053\nvalue = [1
215, 462]\nclass = a'),
 Text(3392.64, 906.0, 'NMHC <= 0.275\ngini = 0.498\nsamples = 510\nvalue = [3
78, 430]\nclass = b'),
 Text(3214.08, 543.5999999999999, 'CO <= 0.45\ngini = 0.422\nsamples = 291\nv
alue = [316, 137]\nclass = a'),
 Text(3124.8, 181.19999999999982, 'gini = 0.306\nsamples = 234\nvalue = [293,
68]\nclass = a'),
 Text(3303.36, 181.19999999999982, 'gini = 0.375\nsamples = 57\nvalue = [23,
69]\nclass = b'),
 Text(3571.2, 543.5999999999999, 'NO_2 <= 67.5\ngini = 0.288\nsamples = 219\n
value = [62, 293]\nclass = b'),
 Text(3481.92, 181.19999999999982, 'gini = 0.05\nsamples = 139\nvalue = [6, 2
27]\nclass = b'),
 Text(3660.48, 181.19999999999982, 'gini = 0.497\nsamples = 80\nvalue = [56,
66]\nclass = b'),
 Text(4106.88, 906.0, 'NMHC <= 0.165\ngini = 0.071\nsamples = 543\nvalue = [8
37, 32]\nclass = a'),
 Text(3928.32, 543.5999999999999, 'PM10 <= 19.5\ngini = 0.346\nsamples = 10\n
value = [14, 4]\nclass = a'),
 Text(3839.04, 181.19999999999982, 'gini = 0.0\nsamples = 5\nvalue = [10, 0]
\nclass = a'),
 Text(4017.6, 181.19999999999982, 'gini = 0.5\nsamples = 5\nvalue = [4, 4]\nc
lass = a'),
 Text(4285.4400000000005, 543.5999999999999, 'EBE <= 1.15\ngini = 0.064\nsamp
les = 533\nvalue = [823, 28]\nclass = a'),
 Text(4196.16, 181.19999999999982, 'gini = 0.328\nsamples = 58\nvalue = [69,
18]\nclass = a'),
 Text(4374.72, 181.19999999999982, 'gini = 0.026\nsamples = 475\nvalue = [75
4, 10]\nclass = a')]
```



Conclusion

Accuracy

In [182]:
```
lr.score(x_train,y_train)
```

Out[182]: 0.6247137686462515

In [183]:
```
rr.score(x_train,y_train)
```

Out[183]: 0.621965644818891

In [184]:
```
la.score(x_train,y_train)
```

Out[184]: 0.3692715607901955

In [185]:
```
en.score(x_test,y_test)
```

Out[185]: 0.4650923677984724

In [186]:
```
logr.score(fs,target_vector)
```

Out[186]: 0.9293697325027482

In [187]:
```
grid_search.best_score_
```

Out[187]: 0.9602150077211771

Random Forest is suitable for this dataset