20104016

DEENA

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("madrid_2017.csv")
df
```

Out[2]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2
0	2017-06-01 01:00:00	NaN	NaN	0.3	NaN	NaN	4.0	38.0	NaN	NaN	NaN	NaN	5.0
1	2017-06-01 01:00:00	0.6	NaN	0.3	0.4	0.08	3.0	39.0	NaN	71.0	22.0	9.0	7.0
2	2017-06-01 01:00:00	0.2	NaN	NaN	0.1	NaN	1.0	14.0	NaN	NaN	NaN	NaN	NaN
3	2017-06-01 01:00:00	NaN	NaN	0.2	NaN	NaN	1.0	9.0	NaN	91.0	NaN	NaN	NaN
4	2017-06-01 01:00:00	NaN	NaN	NaN	NaN	NaN	1.0	19.0	NaN	69.0	NaN	NaN	2.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
210115	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	27.0	NaN	65.0	NaN	NaN	NaN
210116	2017-08-01 00:00:00	NaN	NaN	0.2	NaN	NaN	1.0	14.0	NaN	NaN	73.0	NaN	7.0
210117	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	4.0	NaN	83.0	NaN	NaN	NaN
210118	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	11.0	NaN	78.0	NaN	NaN	NaN
210119	2017-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	1.0	14.0	NaN	77.0	60.0	NaN	NaN

210120 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: 4127 entries, 87457 to 158286
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     4127 non-null   object 
 1   BEN      4127 non-null   float64
 2   CH4      4127 non-null   float64
 3   CO       4127 non-null   float64
 4   EBE      4127 non-null   float64
 5   NMHC     4127 non-null   float64
 6   NO       4127 non-null   float64
 7   NO_2     4127 non-null   float64
 8   NOx      4127 non-null   float64
 9   O_3      4127 non-null   float64
 10  PM10     4127 non-null   float64
 11  PM25     4127 non-null   float64
 12  SO_2     4127 non-null   float64
 13  TCH      4127 non-null   float64
 14  TOL      4127 non-null   float64
 15  station  4127 non-null   int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 548.1+ KB
```

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

Out[6]:

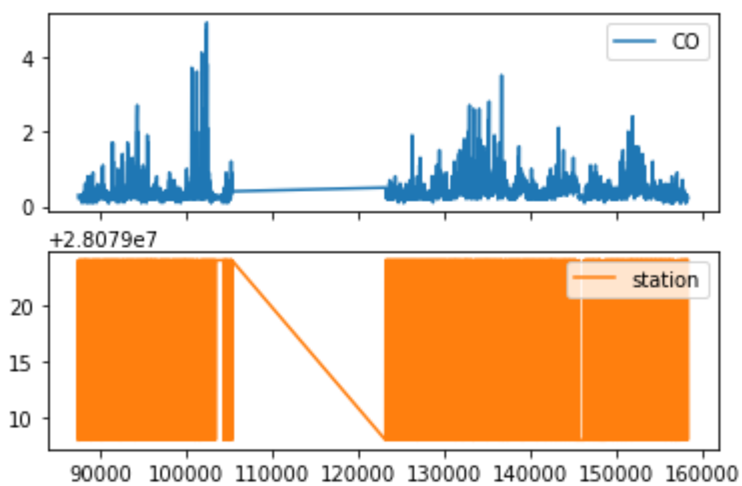
	CO	station
87457	0.3	28079008
87462	0.2	28079024
87481	0.2	28079008
87486	0.2	28079024
87505	0.2	28079008
...	...	...
158238	0.2	28079024
158257	0.3	28079008
158262	0.2	28079024
158281	0.2	28079008
158286	0.2	28079024

4127 rows × 2 columns

Line chart

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

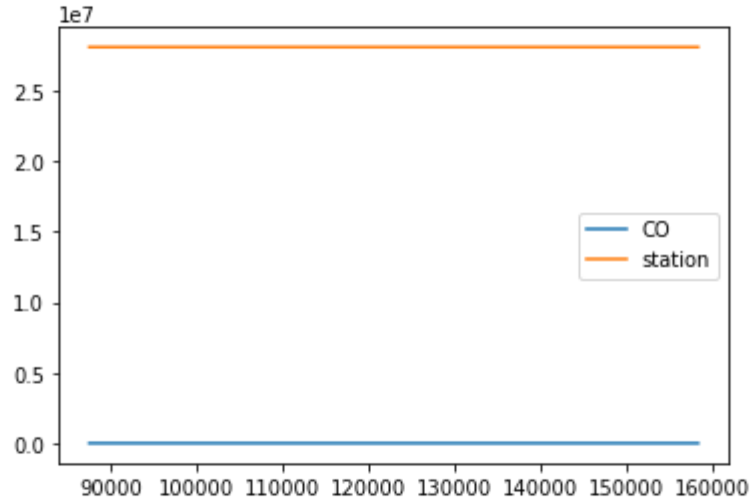
Out[7]: array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

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

Out[8]: <AxesSubplot:>

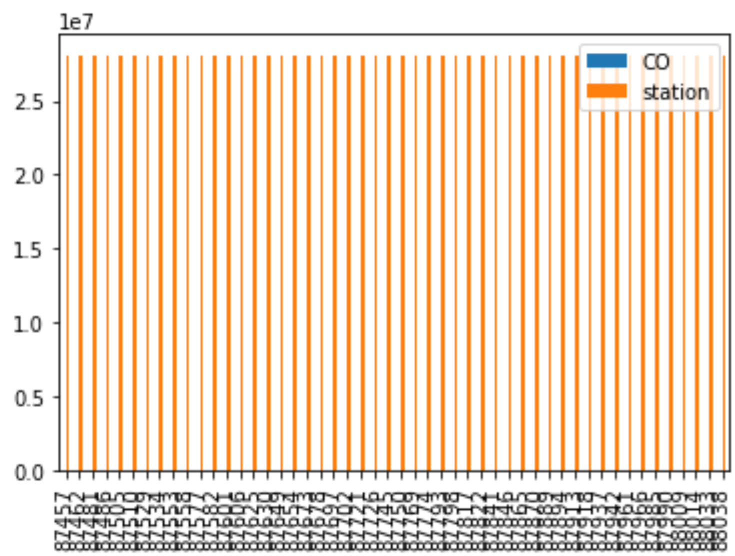


Bar chart

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

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

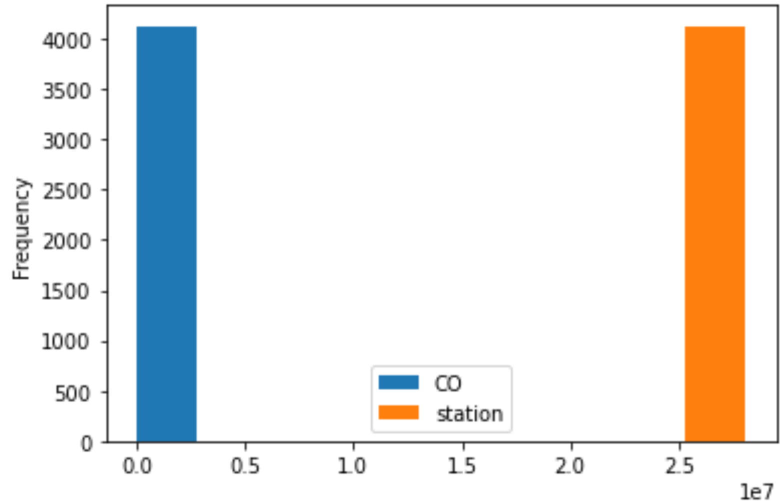
Out[10]: <AxesSubplot:>



Histogram

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

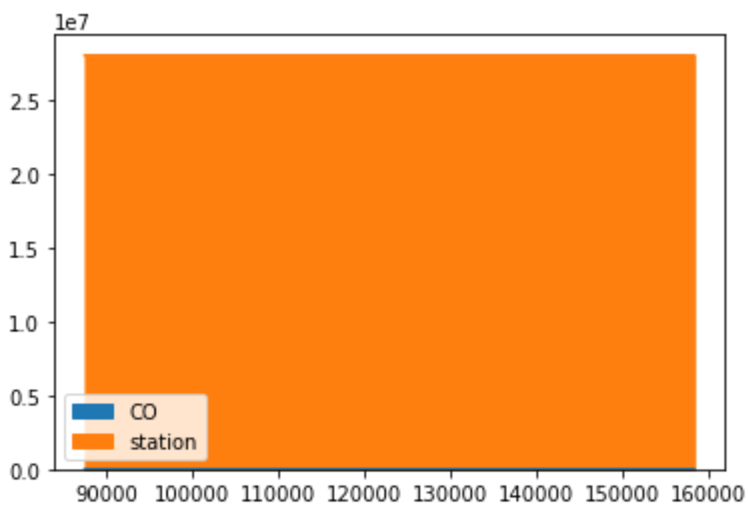
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

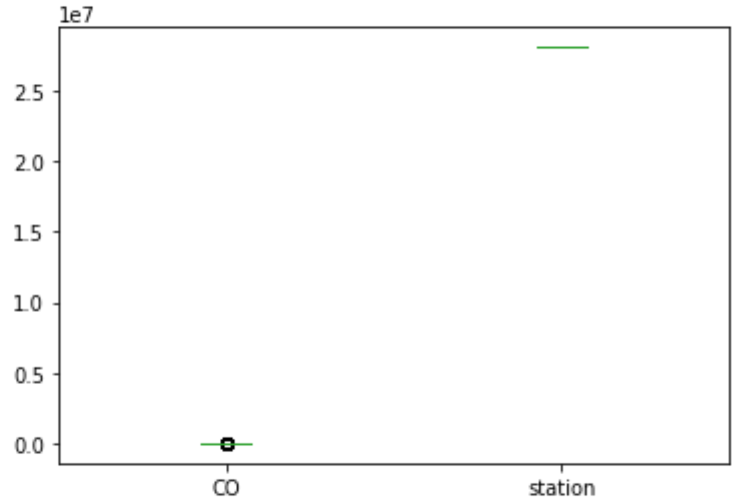
Out[12]: <AxesSubplot:>



Box chart

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

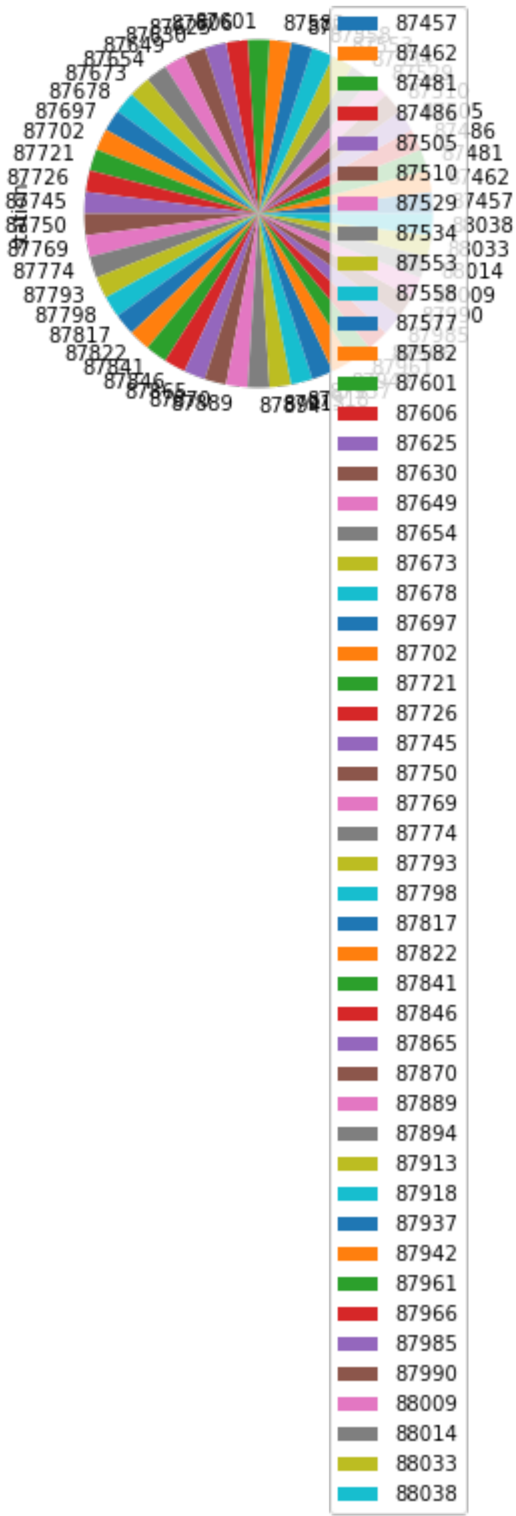
Out[13]: <AxesSubplot:>



Pie chart

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

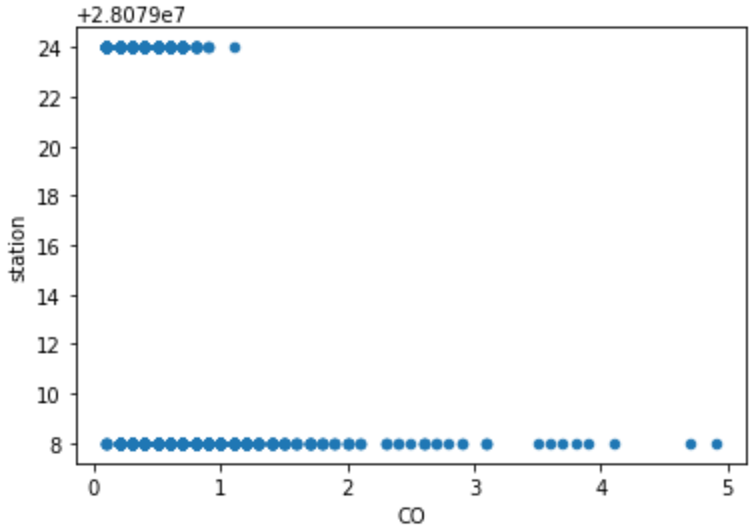
Out[14]: <AxesSubplot:ylabel='station'>



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: 4127 entries, 87457 to 158286
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     4127 non-null   object
 1   BEN      4127 non-null   float64
 2   CH4      4127 non-null   float64
 3   CO       4127 non-null   float64
 4   EBE      4127 non-null   float64
 5   NMHC     4127 non-null   float64
 6   NO       4127 non-null   float64
 7   NO_2     4127 non-null   float64
 8   NOx      4127 non-null   float64
 9   O_3      4127 non-null   float64
 10  PM10     4127 non-null   float64
 11  PM25     4127 non-null   float64
 12  SO_2     4127 non-null   float64
 13  TCH      4127 non-null   float64
 14  TOL      4127 non-null   float64
```

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

Out[17]:

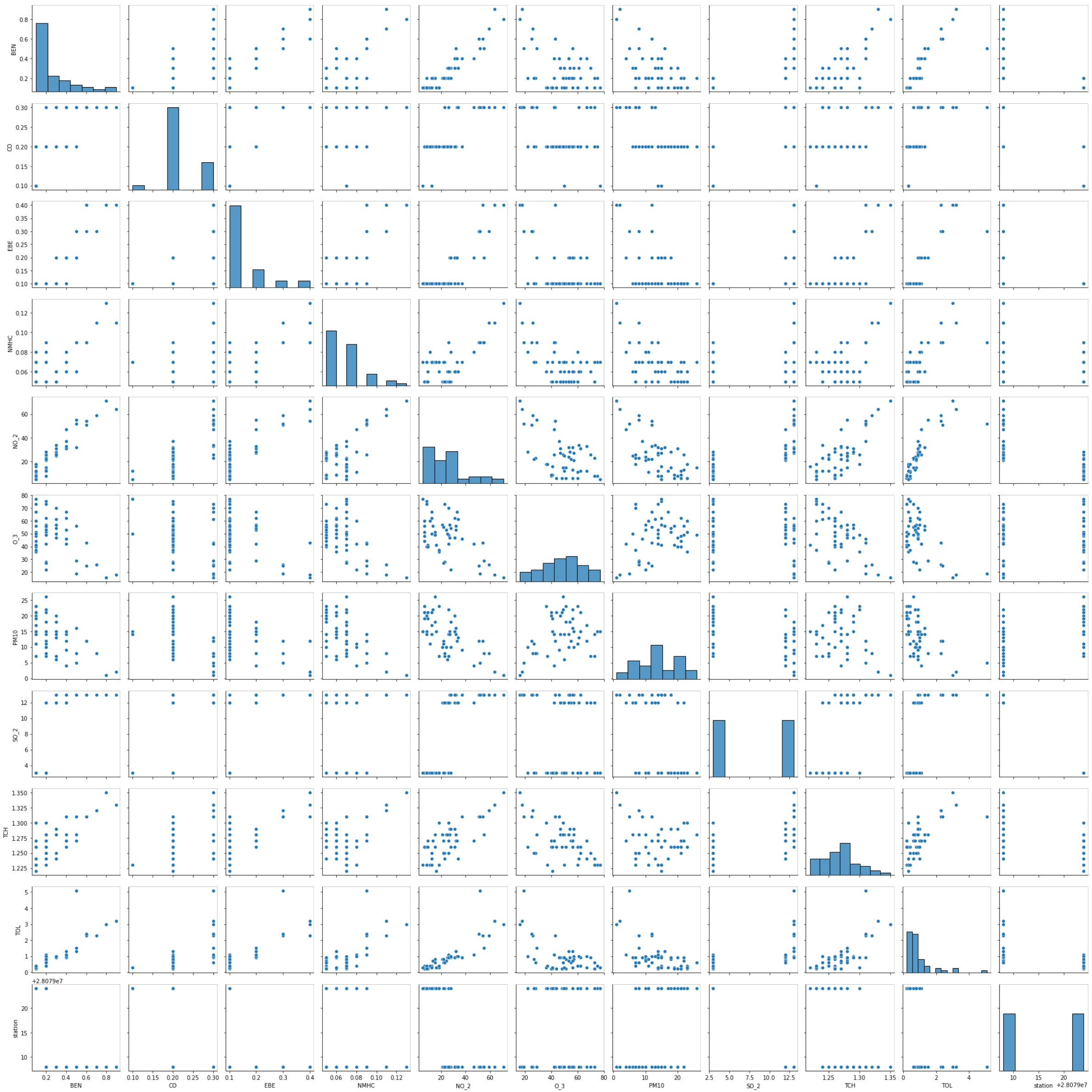
	BEN	CH4	CO	EBE	NMHC	NO	NO_
count	4127.000000	4127.000000	4127.000000	4127.000000	4127.000000	4127.000000	4127.0000
mean	0.919918	1.323732	0.417858	0.578168	0.097269	41.785316	58.0690
std	1.123078	0.215742	0.342871	0.962000	0.094035	71.118499	38.9741
min	0.100000	1.100000	0.100000	0.100000	0.000000	1.000000	1.0000
25%	0.300000	1.180000	0.200000	0.100000	0.050000	3.000000	30.0000
50%	0.600000	1.270000	0.300000	0.300000	0.080000	16.000000	54.0000
75%	1.100000	1.400000	0.500000	0.700000	0.110000	50.000000	78.0000
max	19.600000	3.630000	4.900000	16.700001	1.420000	879.000000	349.0000

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])
```

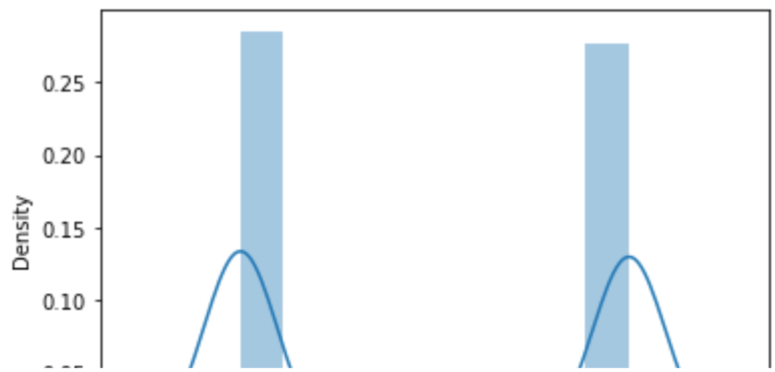
Out[19]: <seaborn.axisgrid.PairGrid at 0x1805bc38a00>



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

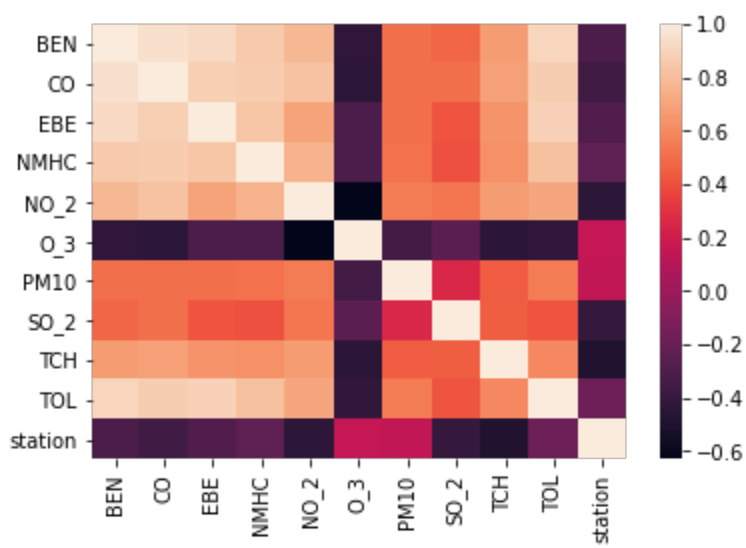
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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]: 28079040.856770415

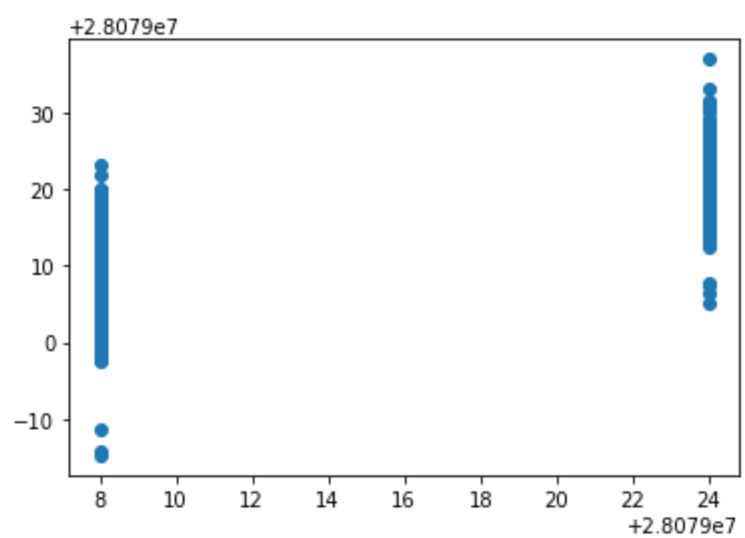
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	1.019110
CO	-2.375163
EBE	-2.213002
NMHC	35.191921
NO_2	-0.159668
O_3	-0.082740
PM10	0.344682
SO_2	-0.235895
TCH	-14.700262
TOL	0.197484

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

Out[27]: <matplotlib.collections.PathCollection at 0x1806423c610>



ACCURACY

In [28]:
```
lr.score(x_test,y_test)
```
Out[28]: 0.6164591813127407

In [29]:
```
lr.score(x_train,y_train)
```
Out[29]: 0.5980708486958615

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]: 0.5967752936426691

In [33]:
```
rr.score(x_train,y_train)
```
Out[33]: 0.582160213288050３

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]: 0.38943633294563806

Accuracy(Lasso)

In [36]:
```
la.score(x_test,y_test)
```
Out[36]: 0.4152858236600927

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.        , -0.16242119,
       -0.05522581,  0.32383153, -0.36053183, -0.        ,  0.11116606])
```

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

Out[39]:
```
28079022.910942458
```

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

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

Out[41]:
```
0.4732218662563411
```

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)))
```

```
4.953386156395066
33.58358989638402
5.795135019685393
```

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]:
```
(4127, 10)
```

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

Out[46]:
```
(4127,)
```

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]:
```
0.9437848315968016
```

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

Out[54]:
```
0.9999999999725541
```

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

Out[55]:
```
array([[1.00000000e+00, 2.74458959e-11]])
```

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]:
```
0.9736842105263158
```

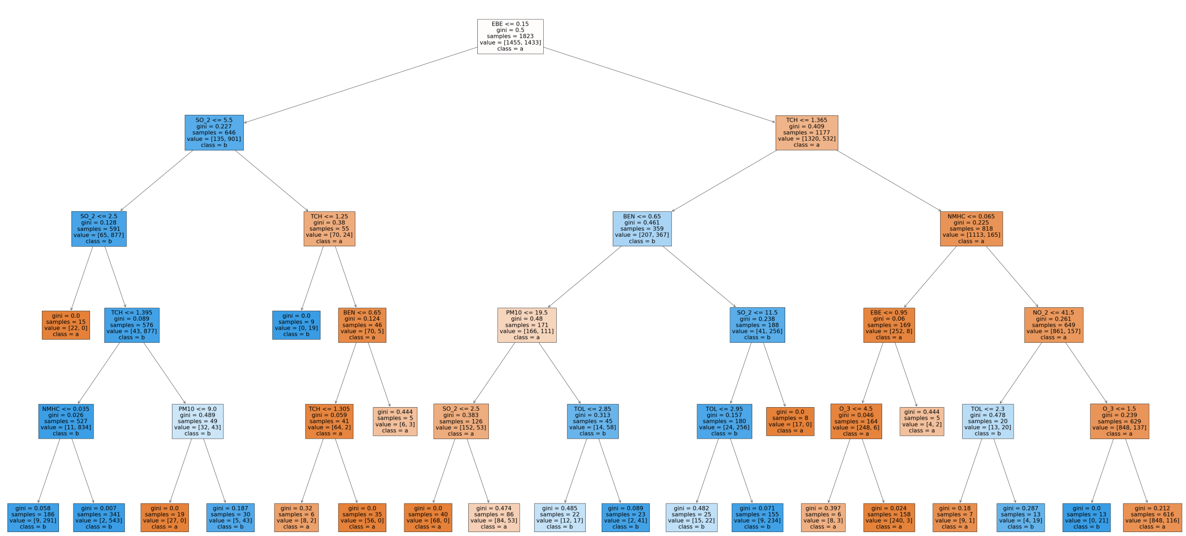
In [61]:
```
rfc_best=grid_search.best_estimator_
```

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(1922.0, 1993.2, 'EBE <= 0.15\ngini = 0.5\nsamples = 1823\nvalue = [145
5, 1433]\nclass = a'),
 Text(806.0, 1630.8000000000002, 'SO_2 <= 5.5\ngini = 0.227\nsamples = 646\nv
alue = [135, 901]\nclass = b'),
 Text(372.0, 1268.4, 'SO_2 <= 2.5\ngini = 0.128\nsamples = 591\nvalue = [65,
877]\nclass = b'),
 Text(248.0, 906.0, 'gini = 0.0\nsamples = 15\nvalue = [22, 0]\nclass = a'),
 Text(496.0, 906.0, 'TCH <= 1.395\ngini = 0.089\nsamples = 576\nvalue = [43,
877]\nclass = b'),
 Text(248.0, 543.5999999999999, 'NMHC <= 0.035\ngini = 0.026\nsamples = 527\n
value = [11, 834]\nclass = b'),
 Text(124.0, 181.19999999999982, 'gini = 0.058\nsamples = 186\nvalue = [9, 29
1]\nclass = b'),
 Text(372.0, 181.19999999999982, 'gini = 0.007\nsamples = 341\nvalue = [2, 54
3]\nclass = b'),
 Text(744.0, 543.5999999999999, 'PM10 <= 9.0\ngini = 0.489\nsamples = 49\nval
ue = [32, 43]\nclass = b'),
 Text(620.0, 181.19999999999982, 'gini = 0.0\nsamples = 19\nvalue = [27, 0]\n
class = a'),
 Text(868.0, 181.19999999999982, 'gini = 0.187\nsamples = 30\nvalue = [5, 4
3]\nclass = b'),
 Text(1240.0, 1268.4, 'TCH <= 1.25\ngini = 0.38\nsamples = 55\nvalue = [70, 2
4]\nclass = a'),
 Text(1116.0, 906.0, 'gini = 0.0\nsamples = 9\nvalue = [0, 19]\nclass = b'),
 Text(1364.0, 906.0, 'BEN <= 0.65\ngini = 0.124\nsamples = 46\nvalue = [70,
5]\nclass = a'),
 Text(1240.0, 543.5999999999999, 'TCH <= 1.305\ngini = 0.059\nsamples = 41\nv
alue = [64, 2]\nclass = a'),
 Text(1116.0, 181.19999999999982, 'gini = 0.32\nsamples = 6\nvalue = [8, 2]\n
class = a'),
 Text(1364.0, 181.19999999999982, 'gini = 0.0\nsamples = 35\nvalue = [56, 0]\
nclass = a'),
 Text(1488.0, 543.5999999999999, 'gini = 0.444\nsamples = 5\nvalue = [6, 3]\n
class = a'),
 Text(3038.0, 1630.8000000000002, 'TCH <= 1.365\ngini = 0.409\nsamples = 117
7\nvalue = [1320, 532]\nclass = a'),
 Text(2418.0, 1268.4, 'BEN <= 0.65\ngini = 0.461\nsamples = 359\nvalue = [20
7, 367]\nclass = b'),
 Text(1984.0, 906.0, 'PM10 <= 19.5\ngini = 0.48\nsamples = 171\nvalue = [166,
111]\nclass = a'),
 Text(1736.0, 543.5999999999999, 'SO_2 <= 2.5\ngini = 0.383\nsamples = 126\nv
alue = [152, 53]\nclass = a'),
 Text(1612.0, 181.19999999999982, 'gini = 0.0\nsamples = 40\nvalue = [68, 0]\
nclass = a'),
 Text(1860.0, 181.19999999999982, 'gini = 0.474\nsamples = 86\nvalue = [84, 5
3]\nclass = a'),
 Text(2232.0, 543.5999999999999, 'TOL <= 2.85\ngini = 0.313\nsamples = 45\nva
lue = [14, 58]\nclass = b'),
 Text(2108.0, 181.19999999999982, 'gini = 0.485\nsamples = 22\nvalue = [12, 1
7]\nclass = b'),
 Text(2356.0, 181.19999999999982, 'gini = 0.089\nsamples = 23\nvalue = [2, 4
```

```
1]\nclass = b'),
 Text(2852.0, 906.0, 'SO_2 <= 11.5\ngini = 0.238\nsamples = 188\nvalue = [41,
256]\nclass = b'),
 Text(2728.0, 543.5999999999999, 'TOL <= 2.95\ngini = 0.157\nsamples = 180\nv
alue = [24, 256]\nclass = b'),
 Text(2604.0, 181.19999999999982, 'gini = 0.482\nsamples = 25\nvalue = [15, 2
2]\nclass = b'),
 Text(2852.0, 181.19999999999982, 'gini = 0.071\nsamples = 155\nvalue = [9, 2
34]\nclass = b'),
 Text(2976.0, 543.5999999999999, 'gini = 0.0\nsamples = 8\nvalue = [17, 0]\nc
lass = a'),
 Text(3658.0, 1268.4, 'NMHC <= 0.065\ngini = 0.225\nsamples = 818\nvalue = [1
113, 165]\nclass = a'),
 Text(3348.0, 906.0, 'EBE <= 0.95\ngini = 0.06\nsamples = 169\nvalue = [252,
8]\nclass = a'),
 Text(3224.0, 543.5999999999999, 'O_3 <= 4.5\ngini = 0.046\nsamples = 164\nva
lue = [248, 6]\nclass = a'),
 Text(3100.0, 181.19999999999982, 'gini = 0.397\nsamples = 6\nvalue = [8, 3]\
nclass = a'),
 Text(3348.0, 181.19999999999982, 'gini = 0.024\nsamples = 158\nvalue = [240,
3]\nclass = a'),
 Text(3472.0, 543.5999999999999, 'gini = 0.444\nsamples = 5\nvalue = [4, 2]\n
class = a'),
 Text(3968.0, 906.0, 'NO_2 <= 41.5\ngini = 0.261\nsamples = 649\nvalue = [86
1, 157]\nclass = a'),
 Text(3720.0, 543.5999999999999, 'TOL <= 2.3\ngini = 0.478\nsamples = 20\nval
ue = [13, 20]\nclass = b'),
 Text(3596.0, 181.19999999999982, 'gini = 0.18\nsamples = 7\nvalue = [9, 1]\n
class = a'),
 Text(3844.0, 181.19999999999982, 'gini = 0.287\nsamples = 13\nvalue = [4, 1
9]\nclass = b'),
 Text(4216.0, 543.5999999999999, 'O_3 <= 1.5\ngini = 0.239\nsamples = 629\nva
lue = [848, 137]\nclass = a'),
 Text(4092.0, 181.19999999999982, 'gini = 0.0\nsamples = 13\nvalue = [0, 21]\
nclass = b'),
 Text(4340.0, 181.19999999999982, 'gini = 0.212\nsamples = 616\nvalue = [848,
116]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression :0.5980708486958615

Ridge Regression :0.38943633294563806

Lasso Regression :0.4152858236600927

ElasticNet Regression : 0.4732218662563411

Logistic Regression : 0.9437848315968016

Random Forest :0.9736842105263158

Random Forest is suitable for this dataset