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_2013").fillna(1)
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2013-11-01 01:00:00	1.0	0.6	1.0	1.0	135.0	74.0	1.0	1.0	1.0	7.0	1.0	1.0	28079004
1	2013-11-01 01:00:00	1.5	0.5	1.3	1.0	71.0	83.0	2.0	23.0	16.0	12.0	1.0	8.3	28079008
2	2013-11-01 01:00:00	3.9	1.0	2.8	1.0	49.0	70.0	1.0	1.0	1.0	1.0	1.0	9.0	28079011
3	2013-11-01 01:00:00	1.0	0.5	1.0	1.0	82.0	87.0	3.0	1.0	1.0	1.0	1.0	1.0	28079016
4	2013-11-01 01:00:00	1.0	1.0	1.0	1.0	242.0	111.0	2.0	1.0	1.0	12.0	1.0	1.0	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209875	2013-03-01 00:00:00	1.0	0.4	1.0	1.0	8.0	39.0	52.0	1.0	1.0	1.0	1.0	1.0	28079056
209876	2013-03-01 00:00:00	1.0	0.4	1.0	1.0	1.0	11.0	1.0	6.0	1.0	2.0	1.0	1.0	28079057
209877	2013-03-01 00:00:00	1.0	1.0	1.0	1.0	2.0	4.0	75.0	1.0	1.0	1.0	1.0	1.0	28079058
209878	2013-03-01 00:00:00	1.0	1.0	1.0	1.0	2.0	11.0	52.0	1.0	1.0	1.0	1.0	1.0	28079059
209879	2013-03-01 00:00:00	1.0	1.0	1.0	1.0	1.0	10.0	75.0	3.0	1.0	1.0	1.0	1.0	28079060

209880 rows × 14 columns

Data Cleaning and Data Preprocessing

In [3]:
```
1 df.columns
```

Out[3]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [4]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      209880 non-null  float64
 2   CO       209880 non-null  float64
 3   EBE      209880 non-null  float64
 4   NMHC     209880 non-null  float64
 5   NO       209880 non-null  float64
 6   NO_2     209880 non-null  float64
 7   O_3      209880 non-null  float64
 8   PM10     209880 non-null  float64
 9   PM25     209880 non-null  float64
 10  SO_2     209880 non-null  float64
 11  TCH      209880 non-null  float64
 12  TOL      209880 non-null  float64
 13  station  209880 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

In [5]:
```
data=df[['CO' ,'station']]
data
```

Out[5]:

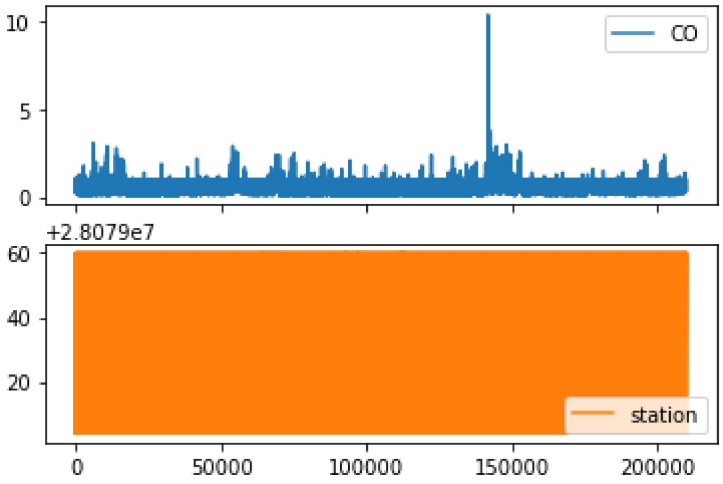
	CO	station
0	0.6	28079004
1	0.5	28079008
2	1.0	28079011
3	0.5	28079016
4	1.0	28079017
...	...	...
209875	0.4	28079056
209876	0.4	28079057
209877	1.0	28079058
209878	1.0	28079059
209879	1.0	28079060

209880 rows × 2 columns

Line chart

In [6]:
```
1 data.plot.line(subplots=True)
```

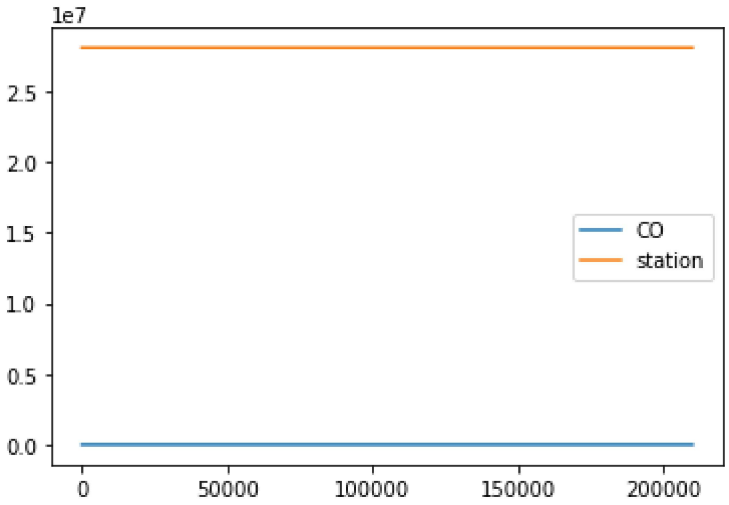
Out[6]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [7]:
```
1 data.plot.line()
```

Out[7]:
```
<AxesSubplot:>
```

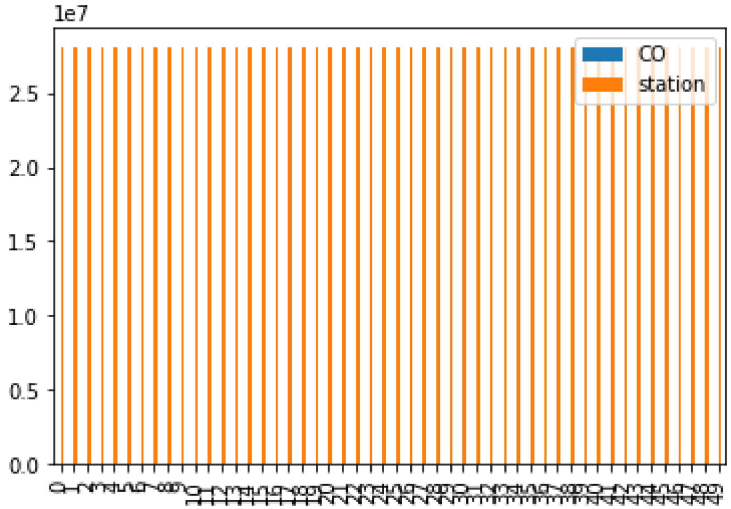


Bar chart

In [8]:
```
1 b=data[0:50]
```

In [9]:
```
1 b.plot.bar()
```

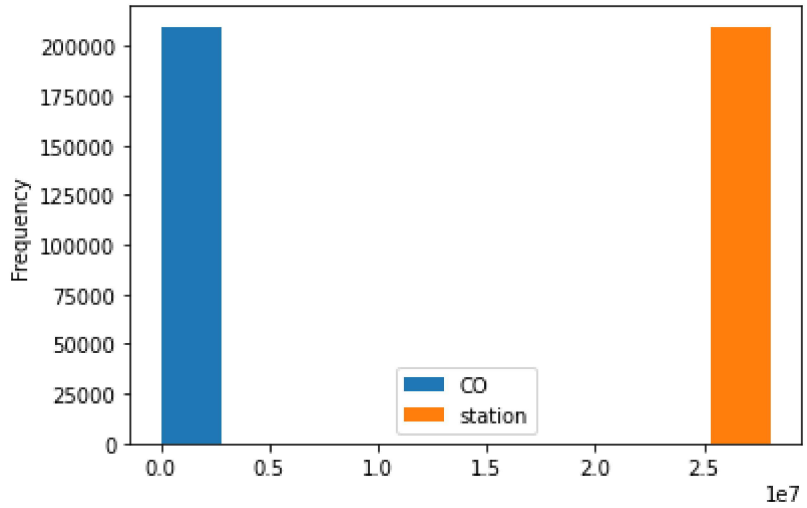
Out[9]: <AxesSubplot:>



Histogram

In [10]:
```
1 data.plot.hist()
```

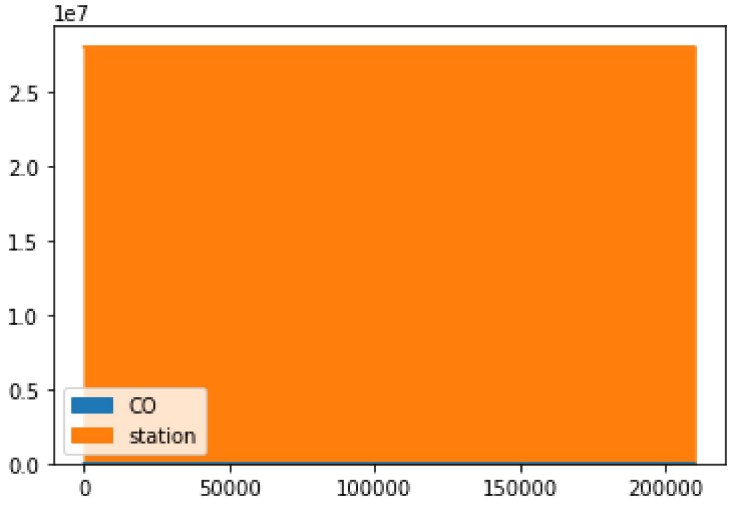
Out[10]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [11]:
```
1 data.plot.area()
```

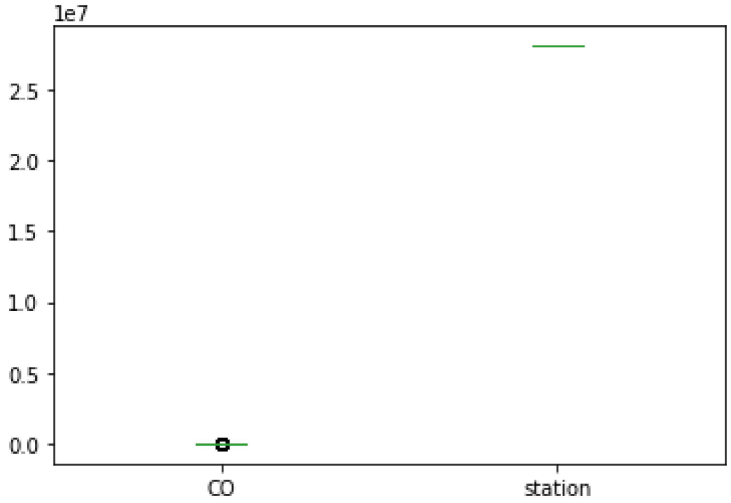
Out[11]: <AxesSubplot:>



Box chart

In [12]:
```
1 data.plot.box()
```

Out[12]: <AxesSubplot:>

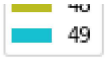


Pie chart

In [13]:
```
1 b.plot.pie(y='station' )
```

Out[13]:
```
<AxesSubplot:ylabel='station'>
```

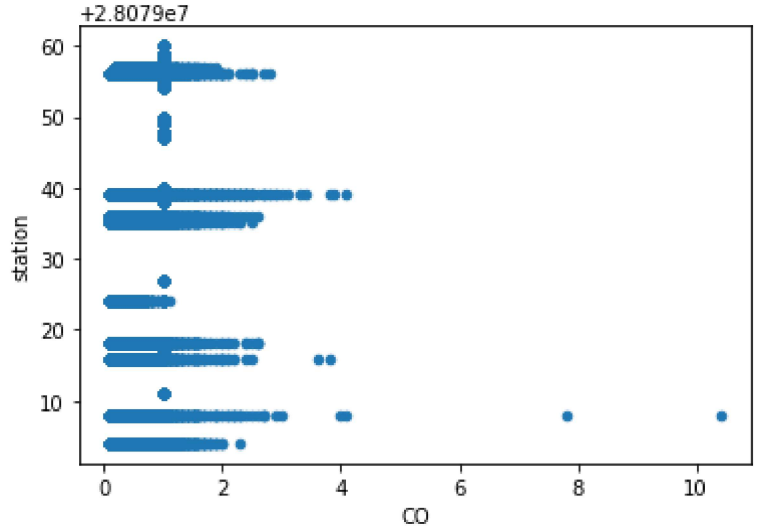

station
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47



Scatter chart

In [14]:
```
1 data.plot.scatter(x='CO' ,y='station')
```

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



In [15]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209880 entries, 0 to 209879
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209880 non-null  object 
 1   BEN      209880 non-null  float64
 2   CO       209880 non-null  float64
 3   EBE      209880 non-null  float64
 4   NMHC     209880 non-null  float64
 5   NO       209880 non-null  float64
 6   NO_2     209880 non-null  float64
 7   O_3      209880 non-null  float64
 8   PM10     209880 non-null  float64
 9   PM25     209880 non-null  float64
 10  SO_2     209880 non-null  float64
 11  TCH      209880 non-null  float64
 12  TOL      209880 non-null  float64
 13  station  209880 non-null  int64
```

In [16]:
```
1 df.describe()
```

Out[16]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	
count	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	209880.000000	20988
mean	0.931014	0.721695	0.954744	0.900223	20.101401	34.586402	29.461235	9.636635	
std	0.430684	0.361528	0.301074	0.267139	44.319112	27.866588	35.362880	13.492716	
min	0.100000	0.100000	0.100000	0.040000	1.000000	1.000000	1.000000	1.000000	
25%	1.000000	0.300000	1.000000	1.000000	2.000000	14.000000	1.000000	1.000000	
50%	1.000000	1.000000	1.000000	1.000000	5.000000	27.000000	8.000000	1.000000	
75%	1.000000	1.000000	1.000000	1.000000	17.000000	48.000000	54.000000	14.000000	
max	12.100000	10.400000	11.800000	1.000000	1081.000000	388.000000	226.000000	232.000000	6

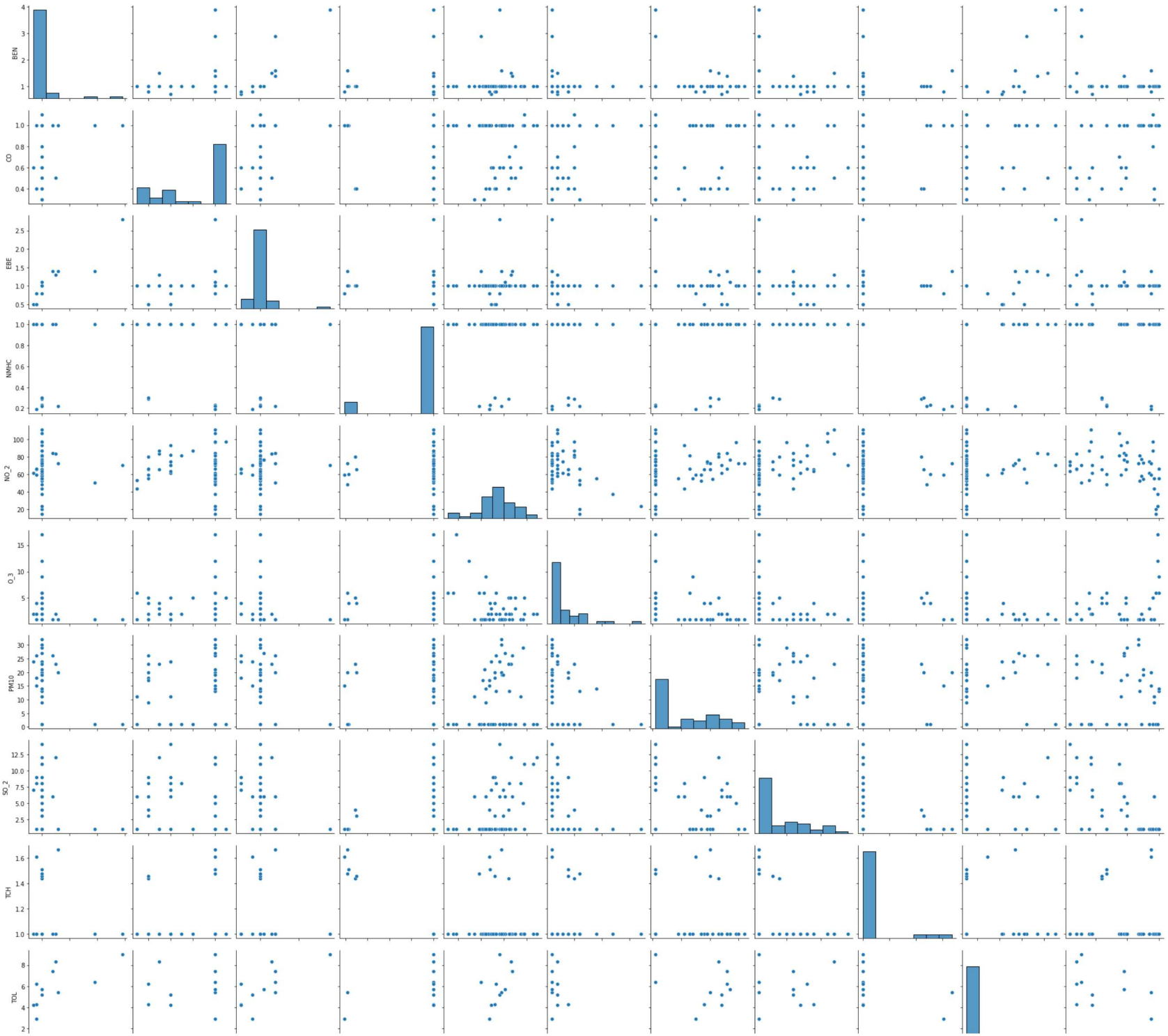
In [17]:

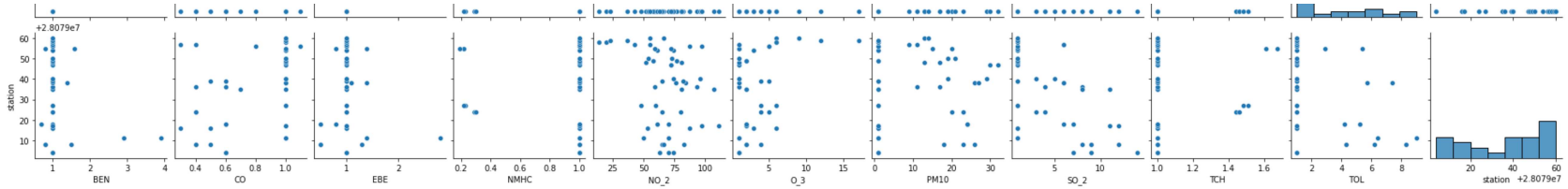
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

In [18]:
```
1 sns.pairplot(df1[0:50])
```

Out[18]:
```
<seaborn.axisgrid.PairGrid at 0x2a8a5de63a0>
```

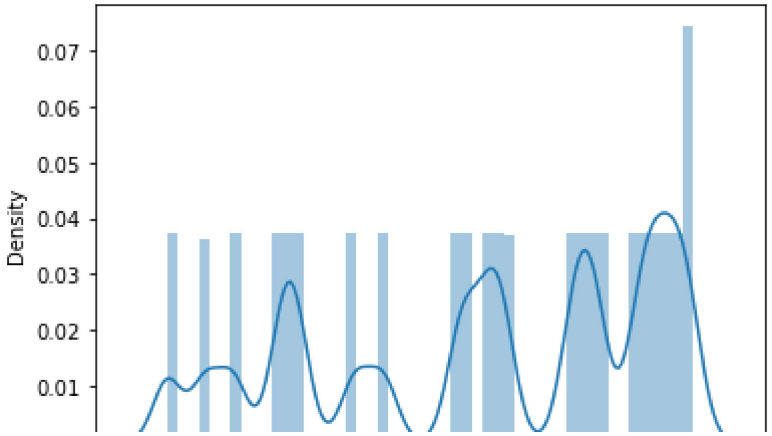

BEN
CO
EBE
NMHC
NO_2
O_3
PM10
SO_2
TCH
TOL



In [19]:
```
1 sns.distplot(df1['station'])
```

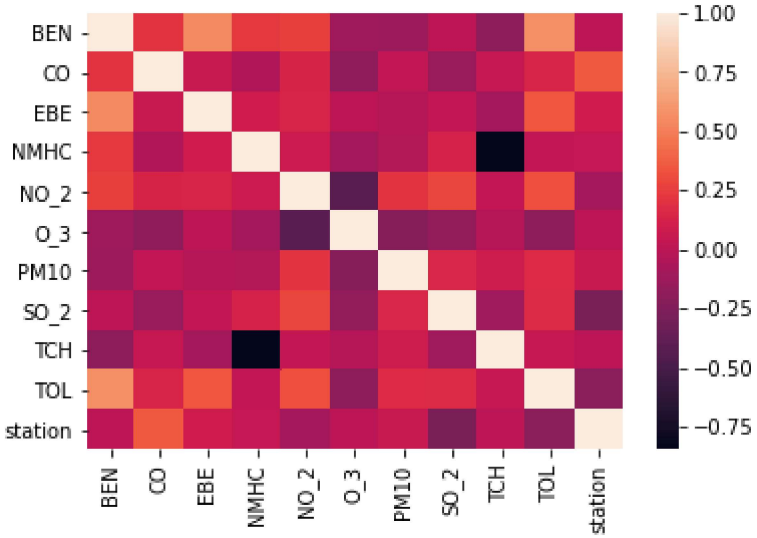
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[19]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [20]:
```
1 sns.heatmap(df1.corr())
```

Out[20]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [21]:
```
1 x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
2        'PM10', 'SO_2', 'TCH', 'TOL']]
3 y=df['station']
```

In [22]:
```
1 from sklearn.model_selection import train_test_split
2 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

Linear Regression

In [23]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[23]:
```
LinearRegression()
```

In [24]:
```
lr.intercept_
```

Out[24]:
```
28078973.97552531
```

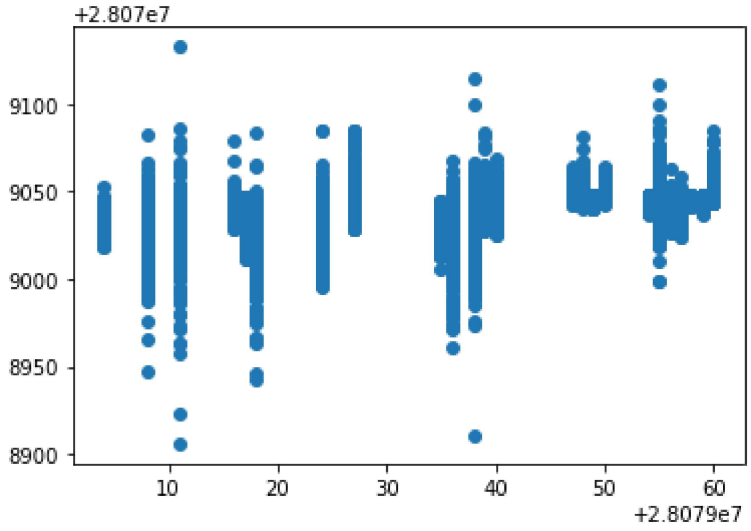
In [25]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[25]:

	Co-efficient
BEN	2.170661
CO	18.565557
EBE	10.169270
NMHC	18.714867
NO_2	-0.055843
O_3	0.010340
PM10	0.203776
SO_2	-0.918896
TCH	27.352854
TOL	-3.683764

In [26]:
```
1 prediction =lr.predict(x_test)
2 plt.scatter(y_test,prediction)
```

Out[26]: <matplotlib.collections.PathCollection at 0x2a8b2aaaf40>



ACCURACY

In [27]:
```
1 lr.score(x_test,y_test)
```

Out[27]: 0.2949171647085367

In [28]:
```
1 lr.score(x_train,y_train)
```

Out[28]: 0.30185667154770124

Ridge and Lasso

In [29]:
```
1 from sklearn.linear_model import Ridge,Lasso
```

In [30]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[30]:
```
Ridge(alpha=10)
```

Accuracy(Ridge)

In [31]:
```
rr.score(x_test,y_test)
```

Out[31]:
```
0.29494142338374996
```

In [32]:
```
rr.score(x_train,y_train)
```

Out[32]:
```
0.3018534587234435
```

In [33]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[33]:
```
Lasso(alpha=10)
```

Accuracy(Lasso)

In [34]:
```
la.score(x_train,y_train)
```

Out[34]:
```
0.04417764487861486
```

In [35]:
```
la.score(x_test,y_test)
```

Out[35]:
```
0.0451150683967122
```

In [36]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[36]:
```
ElasticNet()
```

In [37]:
```
1 en.coef_
```

Out[37]:
```
array([ 0.40679568,  2.73813615,  0.52062804,  0.        , -0.02063318,
       -0.01609616,  0.16005498, -1.26779535, -0.        , -1.65558469])
```

In [38]:
```
1 en.intercept_
```

Out[38]:
```
28079039.89703601
```

In [39]:
```
1 prediction=en.predict(x_test)
```

In [40]:
```
1 en.score(x_test,y_test)
```

Out[40]:
```
0.15589323549099032
```

Evaluation Metrics

In [41]:
```
1 from sklearn import metrics
2 print(metrics.mean_absolute_error(y_test,prediction))
3 print(metrics.mean_squared_error(y_test,prediction))
4 print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
13.717678915380002
261.1312078507424
16.159554692216687
```

Logistic Regression

In [42]:
```
1 from sklearn.linear_model import LogisticRegression
```

In [43]:
```
1 feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
2        'PM10', 'SO_2', 'TCH', 'TOL']]
3 target_vector=df[ 'station']
```

In [44]:
```
1 feature_matrix.shape
```

Out[44]:
```
(209880, 10)
```

In [45]:
```
1 target_vector.shape
```

Out[45]:
```
(209880,)
```

In [46]:
```
1 from sklearn.preprocessing import StandardScaler
```

In [47]:
```
1 fs=StandardScaler().fit_transform(feature_matrix)
```

In [48]:
```
1 logr=LogisticRegression(max_iter=10000)
2 logr.fit(fs,target_vector)
```

Out[48]:
```
LogisticRegression(max_iter=10000)
```

In [49]:
```
1 observation=[[1,2,3,4,5,6,7,8,9,10]]
```

In [50]:
```
1 prediction=logr.predict(observation)
2 print(prediction)
```

```
[28079008]
```

In [51]:
```
1 logr.classes_
```

Out[51]:
```
array([28079004, 28079008, 28079011, 28079016, 28079017, 28079018,
       28079024, 28079027, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079047, 28079048, 28079049, 28079050, 28079054,
       28079055, 28079056, 28079057, 28079058, 28079059, 28079060],
      dtype=int64)
```

In [52]:
```
1 logr.score(fs,target_vector)
```

Out[52]:
```
0.6612921669525443
```

In [53]:
```
1 logr.predict_proba(observation)[0][0]
```

Out[53]:
```
9.49253547859177e-217
```

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

Out[54]:
```
array([[9.49253548e-217, 6.03969072e-001, 1.69773000e-169,
        1.44179094e-134, 1.71060740e-074, 3.96021369e-001,
        9.55808997e-006, 5.22717178e-089, 5.48319507e-081,
        1.32436170e-079, 1.07294134e-076, 3.50636612e-129,
        1.69529056e-079, 3.82520459e-158, 4.22872970e-161,
        3.57928159e-187, 2.10845766e-164, 8.33937392e-188,
        1.12752042e-082, 7.42692411e-129, 7.66872499e-080,
        6.30044443e-191, 4.32093567e-191, 3.26054498e-071]])
```

Random Forest

In [55]:
```
1 from sklearn.ensemble import RandomForestClassifier
```

In [56]:
```
1 rfc=RandomForestClassifier()
2 rfc.fit(x_train,y_train)
```

Out[56]:
```
RandomForestClassifier()
```

In [57]:
```
1 parameters={'max_depth':[1,2,3,4,5],
2             'min_samples_leaf':[5,10,15,20,25],
3             'n_estimators':[10,20,30,40,50]
4 }
```

In [58]:
```
1 from sklearn.model_selection import GridSearchCV
2 grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
3 grid_search.fit(x_train,y_train)
```

Out[58]:
```
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 [59]:
```
1 grid_search.best_score_
```

Out[59]:
```
0.6938182362710664
```

In [60]:
```
rfc_best=grid_search.best_estimator_
```

In [64]:
```
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','c','d','e','f','g','h','i
```

Out[64]:
```
[Text(2441.25, 1993.2, 'O_3 <= 1.5\ngini = 0.958\nsamples = 92940\nvalue = [6119, 6081, 6160, 6255, 6027, 6
217, 6100, 6021, 6107\n6258, 6128, 6195, 6072, 6137, 5955, 6129, 6154, 5950\n6157, 6222, 6014, 6205, 6141,
6112]\nclass = j'),
 Text(1278.75, 1630.8000000000002, 'PM10 <= 1.5\ngini = 0.904\nsamples = 39486\nvalue = [6119, 43, 6160, 1
6, 64, 144, 26, 18, 205, 6258\n6128, 39, 6072, 6137, 5955, 357, 6154, 44, 6157, 60\n6014, 100, 39, 75]\ncla
ss = j'),
 Text(697.5, 1268.4, 'TOL <= 1.05\ngini = 0.587\nsamples = 8552\nvalue = [6119, 39, 6160, 16, 64, 2, 9, 18,
205, 3, 24, 39\n40, 26, 24, 357, 64, 44, 45, 60, 30, 100, 39, 8]\nclass = c'),
 Text(372.0, 906.0, 'SO_2 <= 1.5\ngini = 0.355\nsamples = 4830\nvalue = [6119, 39, 303, 16, 64, 2, 9, 18, 2
05, 3, 7, 39\n40, 26, 24, 357, 64, 44, 36, 60, 30, 100, 39, 8]\nclass = a'),
 Text(186.0, 543.5999999999999, 'TOL <= 0.95\ngini = 0.845\nsamples = 803\nvalue = [20, 39, 303, 16, 32, 2,
9, 18, 8, 3, 4, 39, 15\n26, 24, 357, 64, 44, 36, 60, 2, 100, 39, 8]\nclass = p'),
 Text(93.0, 181.19999999999982, 'gini = 0.118\nsamples = 115\nvalue = [0, 0, 178, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0\n0, 0, 0, 0, 12, 0, 0, 0, 0, 0]\nclass = c'),
 Text(279.0, 181.19999999999982, 'gini = 0.852\nsamples = 688\nvalue = [20, 39, 125, 16, 32, 2, 9, 18, 8,
3, 4, 39, 15\n26, 24, 357, 64, 44, 24, 60, 2, 100, 39, 8]\nclass = p'),
 Text(558.0, 543.5999999999999, 'NO_2 <= 50.5\ngini = 0.086\nsamples = 4027\nvalue = [6099, 0, 0, 0, 32, 0,
0, 0, 197, 0, 3, 0, 25\n0, 0, 0, 0, 0, 0, 0, 28, 0, 0, 0]\nclass = a'),
 Text(465.0, 181.19999999999982, 'gini = 0.04\nsamples = 2604\nvalue = [3982, 0, 0, 0, 4, 0, 0, 0, 26, 0,
```

Conclusion

Accuracy

Linear Regression:0.30185667154770124

Ridge Regression:0.3018534587234435

Lasso Regression:0.04417764487861486

ElasticNet Regression:0.15589323549099032

Logistic Regression:0.6612921669525443

Random Forest:0.6938182362710664

Random Forest is suitable for this dataset