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_2016").fillna(1)
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	station
0	2016-11-01 01:00:00	1.0	0.7	1.0	1.00	153.0	77.0	1.0	1.0	1.0	7.0	1.00	1.0	28079004
1	2016-11-01 01:00:00	3.1	1.1	2.0	0.53	260.0	144.0	4.0	46.0	24.0	18.0	2.44	14.4	28079008
2	2016-11-01 01:00:00	5.9	1.0	7.5	1.00	297.0	139.0	1.0	1.0	1.0	1.0	1.00	26.0	28079011
3	2016-11-01 01:00:00	1.0	1.0	1.0	1.00	154.0	113.0	2.0	1.0	1.0	1.0	1.00	1.0	28079016
4	2016-11-01 01:00:00	1.0	1.0	1.0	1.00	275.0	127.0	2.0	1.0	1.0	18.0	1.00	1.0	28079017
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
209491	2016-07-01 00:00:00	1.0	0.2	1.0	1.00	2.0	29.0	73.0	1.0	1.0	1.0	1.00	1.0	28079056
209492	2016-07-01 00:00:00	1.0	0.3	1.0	1.00	1.0	29.0	1.0	36.0	1.0	5.0	1.00	1.0	28079057
209493	2016-07-01 00:00:00	1.0	1.0	1.0	1.00	1.0	19.0	71.0	1.0	1.0	1.0	1.00	1.0	28079058
209494	2016-07-01 00:00:00	1.0	1.0	1.0	1.00	6.0	17.0	85.0	1.0	1.0	1.0	1.00	1.0	28079059
209495	2016-07-01 00:00:00	1.0	1.0	1.0	1.00	2.0	46.0	61.0	34.0	1.0	1.0	1.00	1.0	28079060

209496 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: 209496 entries, 0 to 209495
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209496 non-null  object 
 1   BEN      209496 non-null  float64
 2   CO       209496 non-null  float64
 3   EBE      209496 non-null  float64
 4   NMHC     209496 non-null  float64
 5   NO       209496 non-null  float64
 6   NO_2     209496 non-null  float64
 7   O_3      209496 non-null  float64
 8   PM10     209496 non-null  float64
 9   PM25     209496 non-null  float64
 10  SO_2     209496 non-null  float64
 11  TCH      209496 non-null  float64
 12  TOL      209496 non-null  float64
 13  station  209496 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 22.4+ MB
```

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

Out[5]:

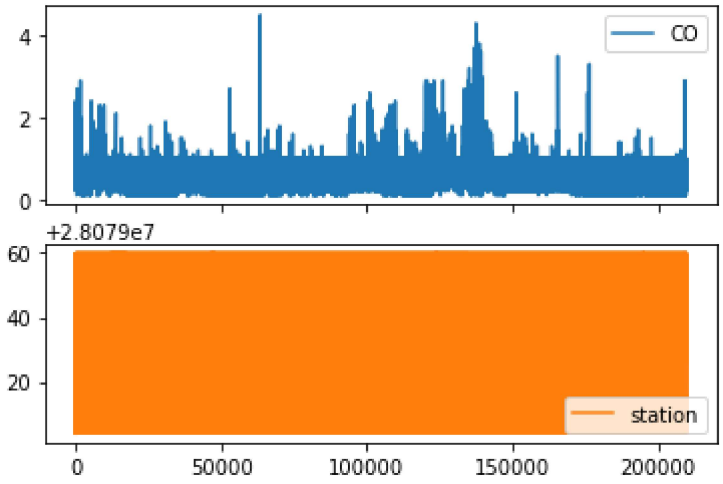
	CO	station
0	0.7	28079004
1	1.1	28079008
2	1.0	28079011
3	1.0	28079016
4	1.0	28079017
...	...	...
209491	0.2	28079056
209492	0.3	28079057
209493	1.0	28079058
209494	1.0	28079059
209495	1.0	28079060

209496 rows × 2 columns

Line chart

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

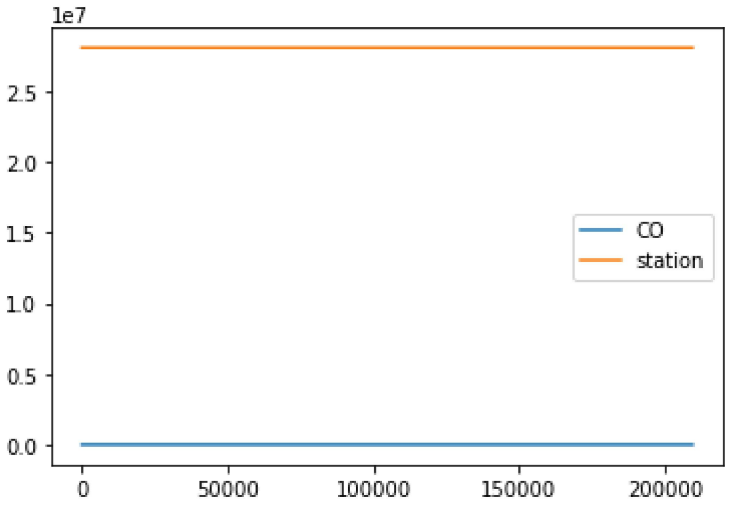
Out[6]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

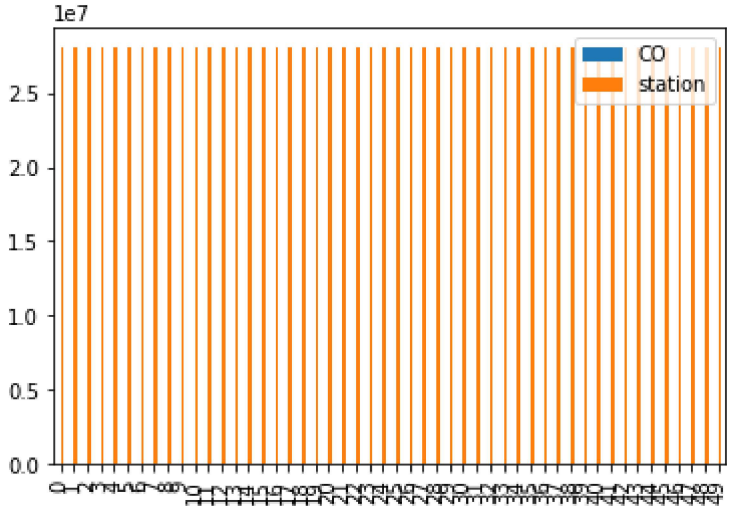


Bar chart

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

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

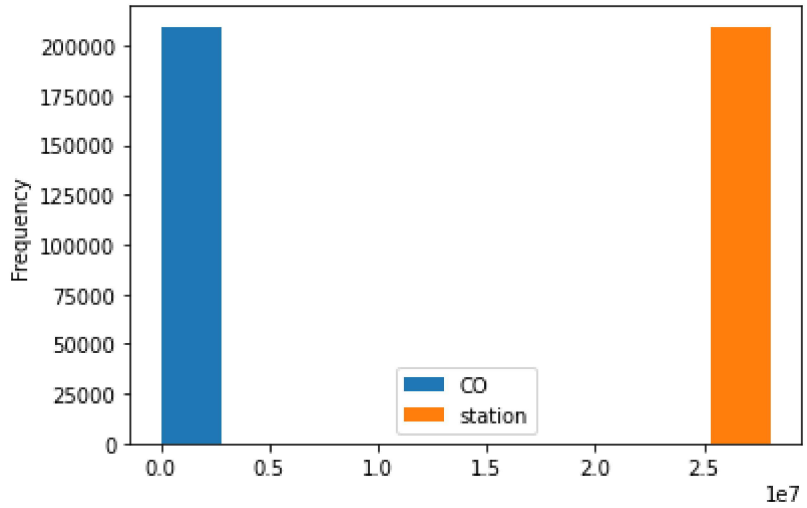
Out[9]: <AxesSubplot:>



Histogram

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

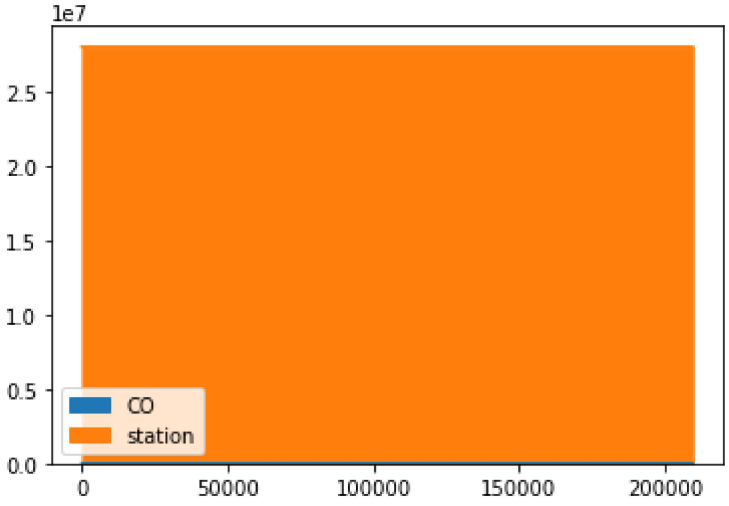
Out[10]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

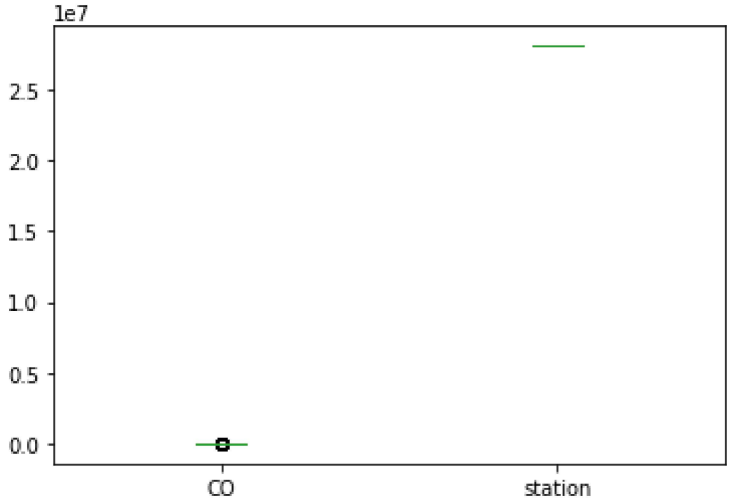
Out[11]: <AxesSubplot:>



Box chart

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

Out[12]: <AxesSubplot:>

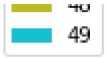


Pie chart

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

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

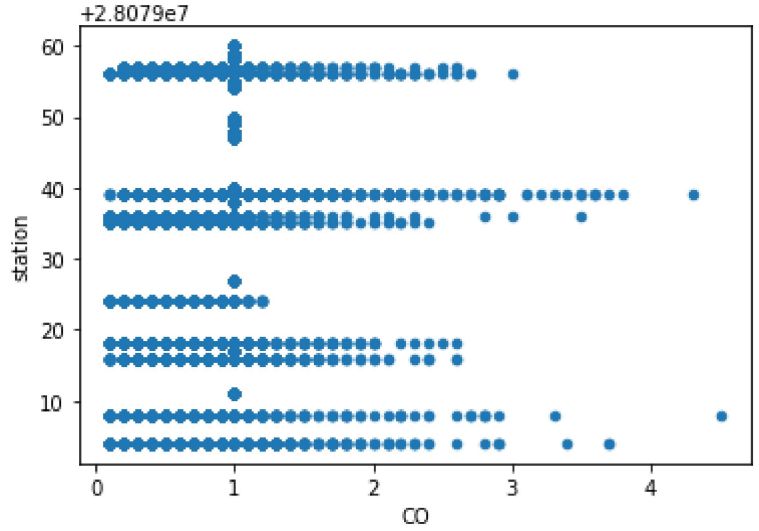

station
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47



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: 209496 entries, 0 to 209495
Data columns (total 14 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     209496 non-null  object 
 1   BEN      209496 non-null  float64
 2   CO       209496 non-null  float64
 3   EBE      209496 non-null  float64
 4   NMHC     209496 non-null  float64
 5   NO       209496 non-null  float64
 6   NO_2     209496 non-null  float64
 7   O_3      209496 non-null  float64
 8   PM10     209496 non-null  float64
 9   PM25     209496 non-null  float64
 10  SO_2     209496 non-null  float64
 11  TCH      209496 non-null  float64
 12  TOL      209496 non-null  float64
 13  station  209496 non-null  int64
```

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

Out[16]:

	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	
count	209496.000000	209496.000000	209496.000000	209496.000000	209496.000000	209496.000000	209496.000000	209496.000000	20949
mean	0.910953	0.735206	0.849691	0.891450	21.969622	38.401120	28.810326	9.914915	
std	0.450296	0.351168	0.428930	0.291514	46.440284	29.011518	35.052506	16.575640	
min	0.100000	0.100000	0.100000	0.000000	1.000000	1.000000	1.000000	0.000000	
25%	1.000000	0.300000	1.000000	1.000000	2.000000	16.000000	1.000000	1.000000	
50%	1.000000	1.000000	1.000000	1.000000	6.000000	31.000000	8.000000	1.000000	
75%	1.000000	1.000000	1.000000	1.000000	19.000000	54.000000	54.000000	14.000000	
max	21.400000	4.500000	27.400000	9.070000	957.000000	324.000000	196.000000	419.000000	21

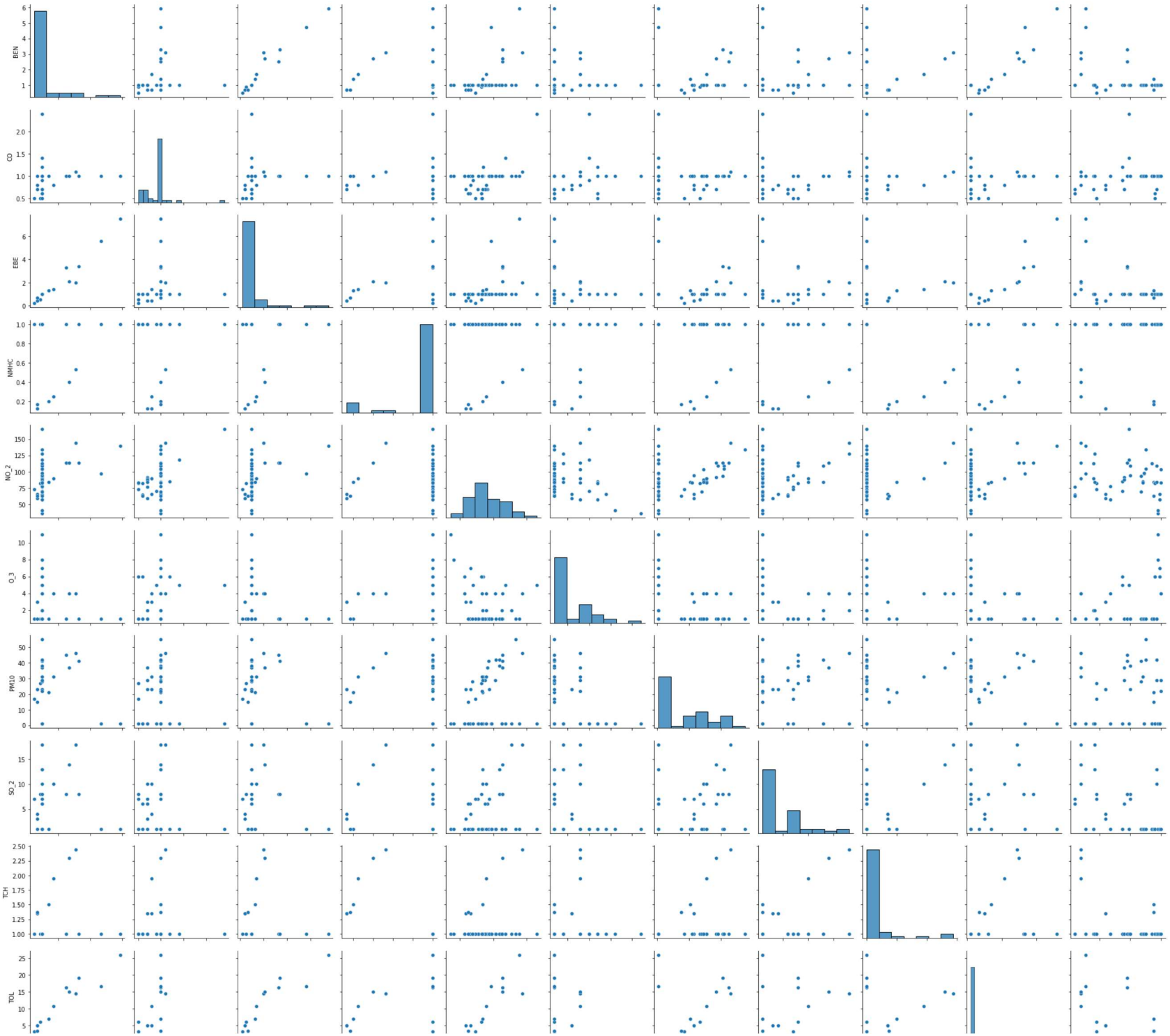
In [17]:

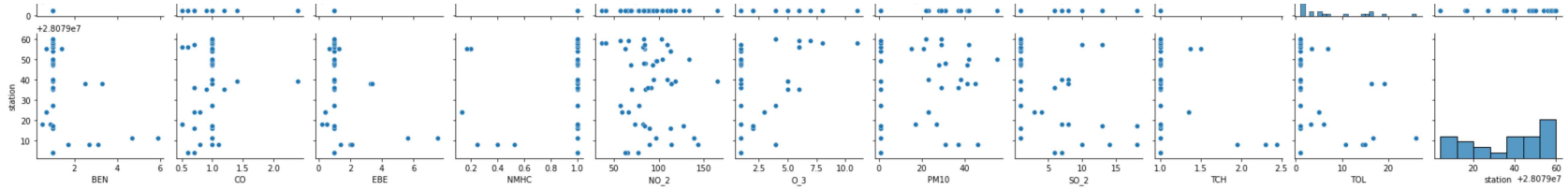
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

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

Out[18]:
```
<seaborn.axisgrid.PairGrid at 0x1ca81946bb0>
```

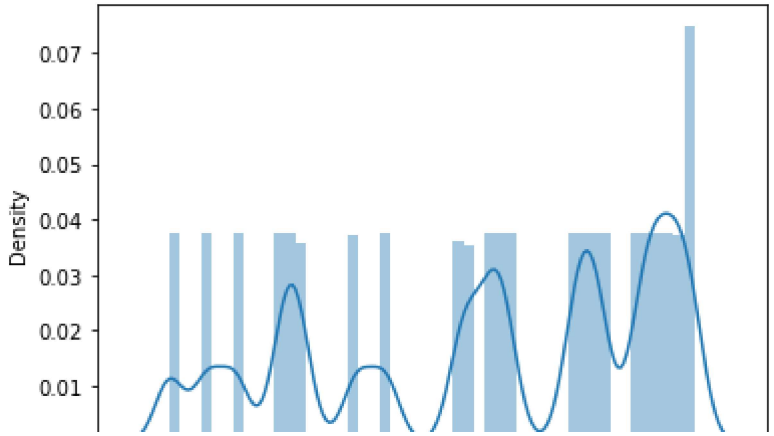

BEN
CO
EBE
NMHC
NO_2
O_3
PM10
SO_2
TCH
TOL



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

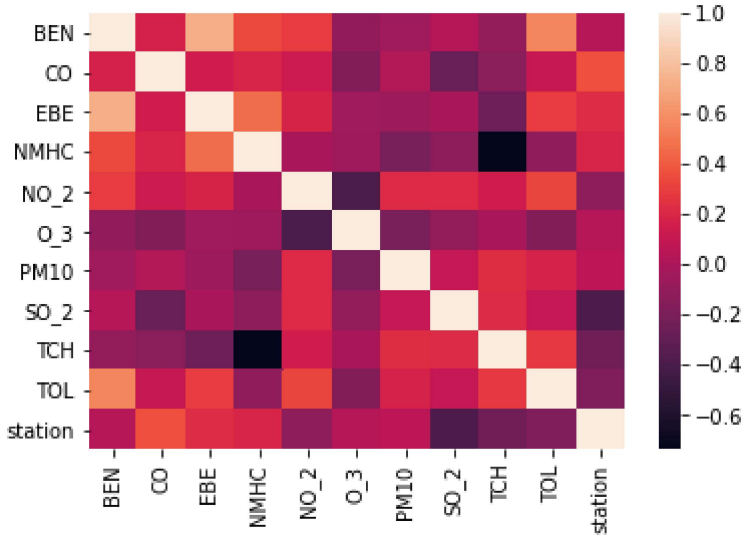
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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]:
```
28079036.995131824
```

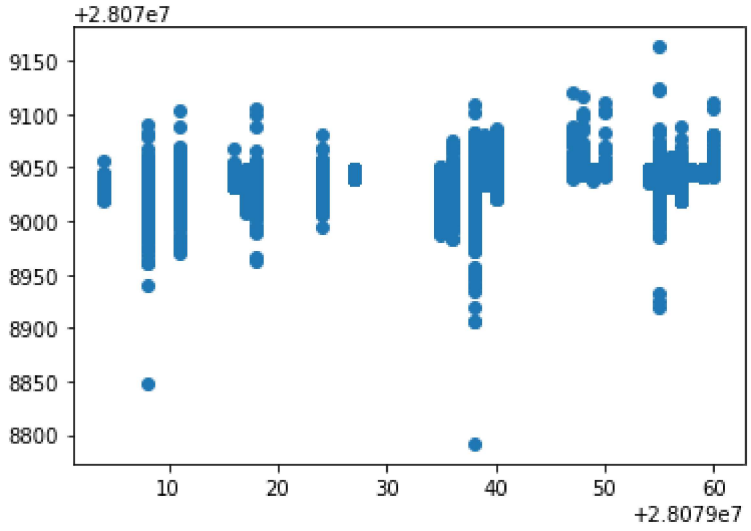
In [25]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[25]:

	Co-efficient
BEN	-2.202676
CO	14.499410
EBE	13.551078
NMHC	-4.274205
NO_2	-0.049590
O_3	0.017024
PM10	0.183913
SO_2	-0.965321
TCH	-8.749293
TOL	-2.298644

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

Out[26]: <matplotlib.collections.PathCollection at 0x1ca8e620670>



ACCURACY

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

Out[27]: 0.3511117831334015

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

Out[28]: 0.3438571482250735

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]: 0.3510983565285565

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

Out[32]: 0.3438568955546134

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]: 0.13987581805661864

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

Out[35]: 0.14130555009221213

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([ 0.48668289,  1.92168242,  2.26046231,  0.28344236, -0.02516436,
        0.00353064,  0.14763803, -1.26793493, -0.        , -1.39855091])
```

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

Out[38]:
```
28079039.88421178
```

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

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

Out[40]:
```
0.23173222960716588
```

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)))
```

```
12.941282699669419
238.6369132094814
15.447877304325065
```

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]:
```
(209496, 10)
```

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

Out[45]:
```
(209496,)
```

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]:
```
0.64236548669187
```

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

Out[53]:
```
6.277186881017425e-223
```

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

Out[54]:
```
array([[6.27718688e-223, 9.99998937e-001, 1.68715981e-174,
        1.61486551e-136, 5.94462895e-076, 1.06303387e-006,
        1.63098419e-029, 6.24785688e-152, 2.19406583e-097,
        8.66488996e-081, 2.53091816e-053, 2.25459599e-137,
        5.80702240e-078, 1.27873869e-179, 1.89251643e-179,
        8.48825655e-153, 7.40413626e-182, 1.53868538e-151,
        9.85444038e-115, 1.38314777e-137, 9.34291914e-084,
        1.43971255e-155, 2.39062594e-154, 5.43362264e-073]])
```

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]:
```
0.6475550149486835
```

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

In [61]:
```
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[61]:
```
[Text(2071.2203389830506, 1993.2, 'NMHC <= 0.97\ngini = 0.958\nsamples = 92622\nvalue = [6294, 6090, 6038,
6094, 6108, 6004, 6263, 6012, 5781\n5813, 6163, 6174, 6208, 6109, 6261, 6185, 6148, 6035\n6168, 6253, 6057,
6015, 6156, 6218]\nclass = a'),
 Text(889.0169491525423, 1630.8000000000002, 'SO_2 <= 1.5\ngini = 0.667\nsamples = 11504\nvalue = [0, 6058,
0, 0, 0, 0, 6169, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 6042, 0, 0, 0, 0, 0]\nclass = g'),
 Text(340.47457627118644, 1268.4, 'NMHC <= 0.045\ngini = 0.006\nsamples = 3831\nvalue = [0, 7, 0, 0, 0, 0,
11, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 6042, 0, 0, 0, 0, 0]\nclass = s'),
 Text(151.32203389830508, 906.0, 'NO_2 <= 21.5\ngini = 0.064\nsamples = 63\nvalue = [0, 0, 0, 0, 0, 0, 3,
0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 87, 0, 0, 0, 0, 0]\nclass = s'),
 Text(75.66101694915254, 543.5999999999999, 'gini = 0.139\nsamples = 29\nvalue = [0, 0, 0, 0, 0, 0, 3, 0,
0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 37, 0, 0, 0, 0, 0]\nclass = s'),
 Text(226.9830508474576, 543.5999999999999, 'gini = 0.0\nsamples = 34\nvalue = [0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0\n0, 0, 0, 0, 50, 0, 0, 0, 0, 0]\nclass = s'),
 Text(529.6271186440678, 906.0, 'BEN <= 0.75\ngini = 0.005\nsamples = 3768\nvalue = [0, 7, 0, 0, 0, 0, 8,
0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 5955, 0, 0, 0, 0, 0]\nclass = s'),
 Text(378.3050847457627, 543.5999999999999, 'PM10 <= 13.5\ngini = 0.003\nsamples = 3057\nvalue = [0, 4, 0,
0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 4836, 0, 0, 0, 0, 0]\nclass = s'),
 Text(302.64406779661016, 181.19999999999982, 'gini = 0.0\nsamples = 1505\nvalue = [0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 2422, 0, 0, 0, 0, 0]\nclass = s'),
```

Conclusion

Accuracy

Linear Regression:0.3438571482250735

Ridge Regression:0.3438568955546134

Lasso Regression:0.13987581805661864

ElasticNet Regression:0.23173222960716588

Logistic Regression:0.64236548669187

Random Forest:0.6475550149486835

Random Forest is suitable for this dataset