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

Out[2]:

| | date | BEN | CH4 | CO | EBE | NMHC | NO | NO_2 | NOx | O_3 | PM10 | PM25 | SO_2 | TCH | TOL | station |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2017-06-01 01:00:00 | 1.0 | 1.0 | 0.3 | 1.0 | 1.00 | 4.0 | 38.0 | 1.0 | 1.0 | 1.0 | 1.0 | 5.0 | 1.0 | 1.0 | 28079004 |
| **1** | 2017-06-01 01:00:00 | 0.6 | 1.0 | 0.3 | 0.4 | 0.08 | 3.0 | 39.0 | 1.0 | 71.0 | 22.0 | 9.0 | 7.0 | 1.4 | 2.9 | 28079008 |
| **2** | 2017-06-01 01:00:00 | 0.2 | 1.0 | 1.0 | 0.1 | 1.00 | 1.0 | 14.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.9 | 28079011 |
| **3** | 2017-06-01 01:00:00 | 1.0 | 1.0 | 0.2 | 1.0 | 1.00 | 1.0 | 9.0 | 1.0 | 91.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 28079016 |
| **4** | 2017-06-01 01:00:00 | 1.0 | 1.0 | 1.0 | 1.0 | 1.00 | 1.0 | 19.0 | 1.0 | 69.0 | 1.0 | 1.0 | 2.0 | 1.0 | 1.0 | 28079017 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **210115** | 2017-08-01 00:00:00 | 1.0 | 1.0 | 0.2 | 1.0 | 1.00 | 1.0 | 27.0 | 1.0 | 65.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 28079056 |
| **210116** | 2017-08-01 00:00:00 | 1.0 | 1.0 | 0.2 | 1.0 | 1.00 | 1.0 | 14.0 | 1.0 | 1.0 | 73.0 | 1.0 | 7.0 | 1.0 | 1.0 | 28079057 |
| **210117** | 2017-08-01 00:00:00 | 1.0 | 1.0 | 1.0 | 1.0 | 1.00 | 1.0 | 4.0 | 1.0 | 83.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 28079058 |
| **210118** | 2017-08-01 00:00:00 | 1.0 | 1.0 | 1.0 | 1.0 | 1.00 | 1.0 | 11.0 | 1.0 | 78.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 28079059 |
| **210119** | 2017-08-01 00:00:00 | 1.0 | 1.0 | 1.0 | 1.0 | 1.00 | 1.0 | 14.0 | 1.0 | 77.0 | 60.0 | 1.0 | 1.0 | 1.0 | 1.0 | 28079060 |

210120 rows × 16 columns

# Data Cleaning and Data Preprocessing

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

Out[3]:
```
Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210120 entries, 0 to 210119
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     210120 non-null  object 
 1   BEN      210120 non-null  float64
 2   CH4      210120 non-null  float64
 3   CO       210120 non-null  float64
 4   EBE      210120 non-null  float64
 5   NMHC     210120 non-null  float64
 6   NO       210120 non-null  float64
 7   NO_2     210120 non-null  float64
 8   NOx      210120 non-null  float64
 9   O_3      210120 non-null  float64
 10  PM10     210120 non-null  float64
 11  PM25     210120 non-null  float64
 12  SO_2     210120 non-null  float64
 13  TCH      210120 non-null  float64
 14  TOL      210120 non-null  float64
 15  station  210120 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 25.6+ MB
```

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

Out[5]:

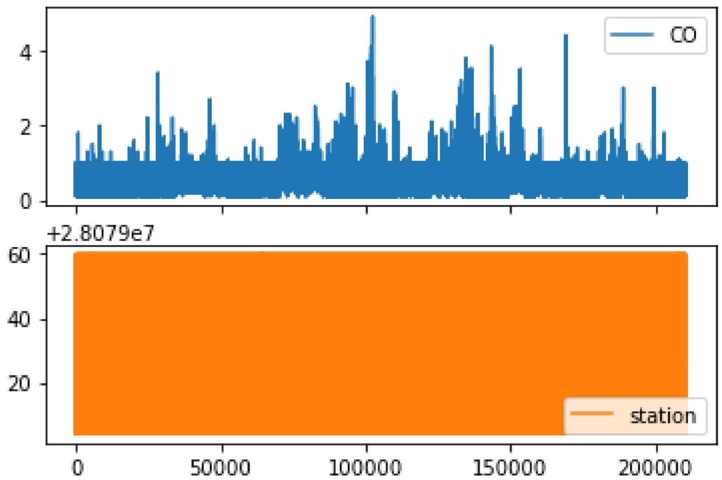
| | CO | station |
|---|---|---|
| **0** | 0.3 | 28079004 |
| **1** | 0.3 | 28079008 |
| **2** | 1.0 | 28079011 |
| **3** | 0.2 | 28079016 |
| **4** | 1.0 | 28079017 |
| **...** | ... | ... |
| **210115** | 0.2 | 28079056 |
| **210116** | 0.2 | 28079057 |
| **210117** | 1.0 | 28079058 |
| **210118** | 1.0 | 28079059 |
| **210119** | 1.0 | 28079060 |

210120 rows × 2 columns

# Line chart

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

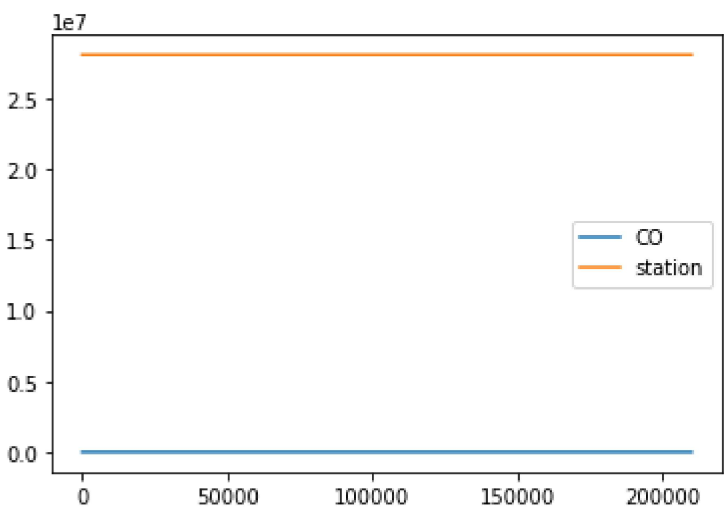
Out[6]: array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)




# Line chart

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

Out[7]: <AxesSubplot:>

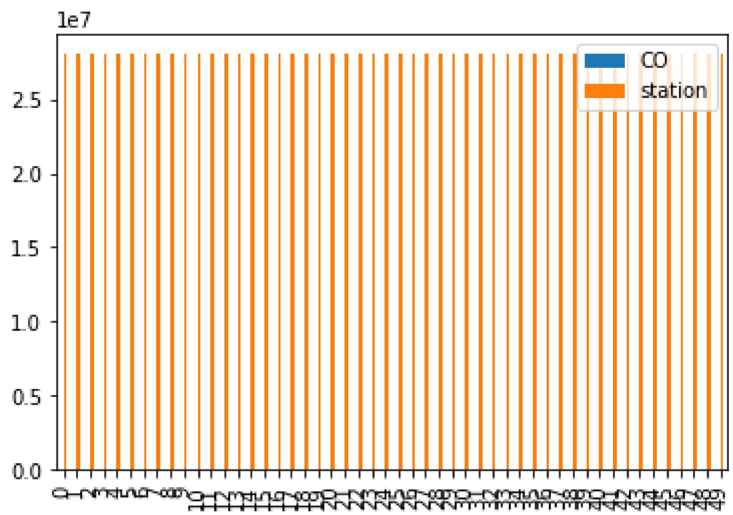


## Bar chart

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

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

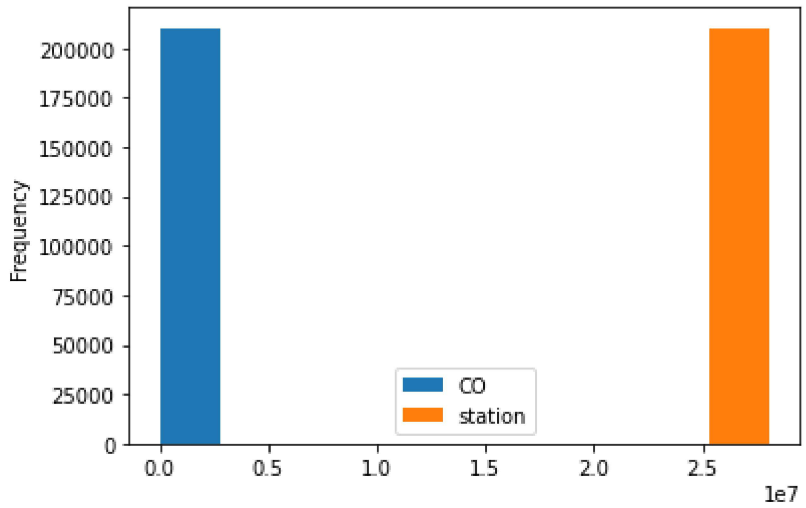
Out[9]: <AxesSubplot:>




## Histogram

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

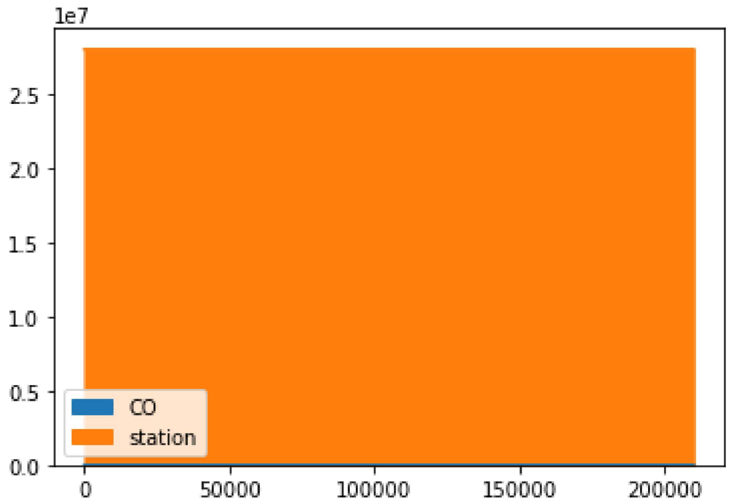
Out[10]: <AxesSubplot:ylabel='Frequency'>




## Area chart

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

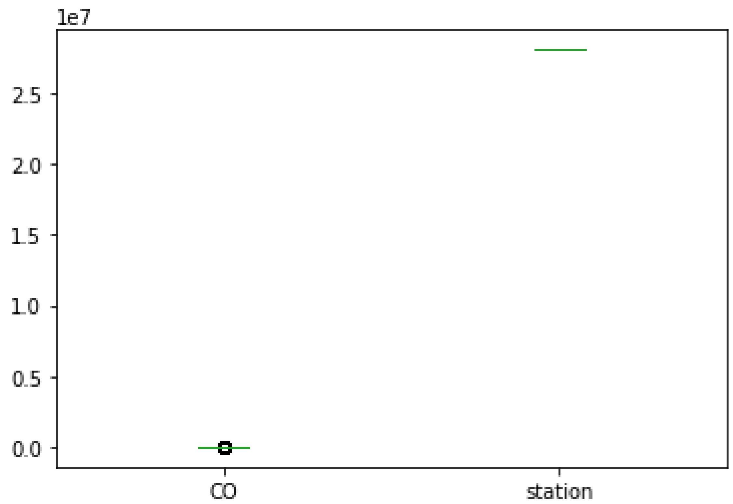
Out[11]: <AxesSubplot:>




# Box chart

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

Out[12]: <AxesSubplot:>

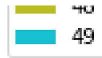



## Pie chart

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

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

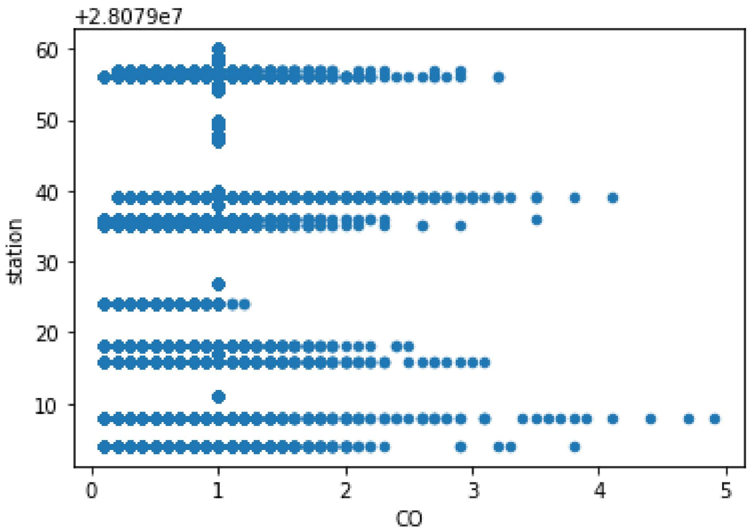

station
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47




# Scatter chart

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

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



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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210120 entries, 0 to 210119
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype
---  ------   --------------   -----
 0   date     210120 non-null  object
 1   BEN      210120 non-null  float64
 2   CH4      210120 non-null  float64
 3   CO       210120 non-null  float64
 4   EBE      210120 non-null  float64
 5   NMHC     210120 non-null  float64
 6   NO       210120 non-null  float64
 7   NO_2     210120 non-null  float64
 8   NOx      210120 non-null  float64
 9   O_3      210120 non-null  float64
 10  PM10     210120 non-null  float64
 11  PM25     210120 non-null  float64
 12  SO_2     210120 non-null  float64
 13  TCH      210120 non-null  float64
```

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

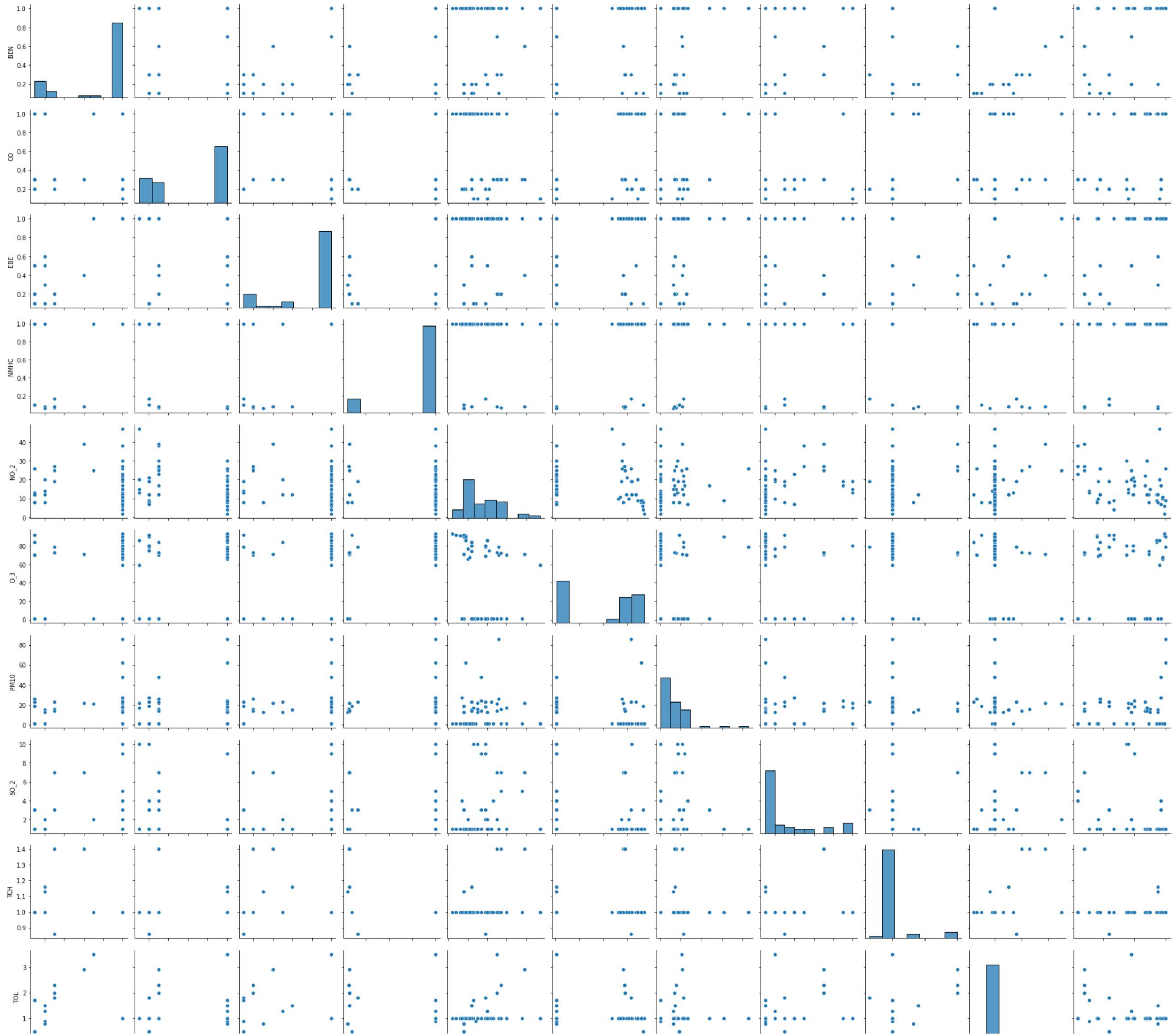
Out[16]:

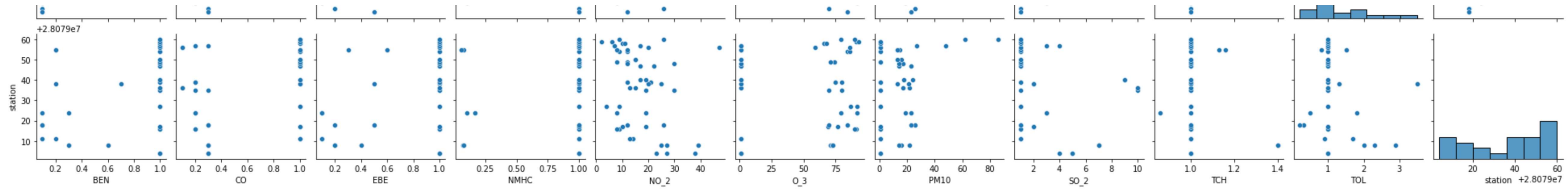
| | BEN | CH4 | CO | EBE | NMHC | NO | NO_2 | NOx | |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 210120.000000 | 210120.000000 | 210120.000000 | 210120.000000 | 210120.000000 | 210120.000000 | 210120.000000 | 210120.000000 | 21012 |
| **mean** | 0.903367 | 1.009799 | 0.736606 | 0.856069 | 0.894275 | 23.296673 | 41.377046 | 32.240248 | 2 |
| **std** | 0.415995 | 0.065702 | 0.356031 | 0.417742 | 0.286557 | 50.261335 | 32.420361 | 92.406059 | 3 |
| **min** | 0.100000 | 1.000000 | 0.100000 | 0.100000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 | |
| **25%** | 1.000000 | 1.000000 | 0.300000 | 1.000000 | 1.000000 | 2.000000 | 17.000000 | 1.000000 | |
| **50%** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 6.000000 | 33.000000 | 1.000000 | |
| **75%** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 20.000000 | 58.000000 | 5.000000 | 5 |
| **max** | 19.600000 | 3.630000 | 4.900000 | 38.299999 | 4.400000 | 973.000000 | 349.000000 | 1798.000000 | 19 |

In [17]:
```
df1=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO_2', 'O_3',
       'PM10', 'SO_2', 'TCH', 'TOL', 'station']]
```

# EDA AND VISUALIZATION

In [18]:
```
sns.pairplot(df1[0:50])
```
Out[18]:
```
<seaborn.axisgrid.PairGrid at 0x248e67d2a00>
```

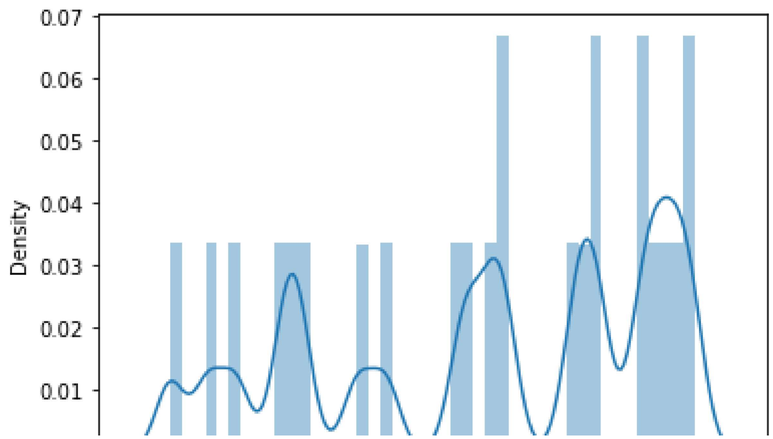

BEN
CO
EBE
NMHC
NO_2
O_3
PM10
SO_2
TCH
TOL



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

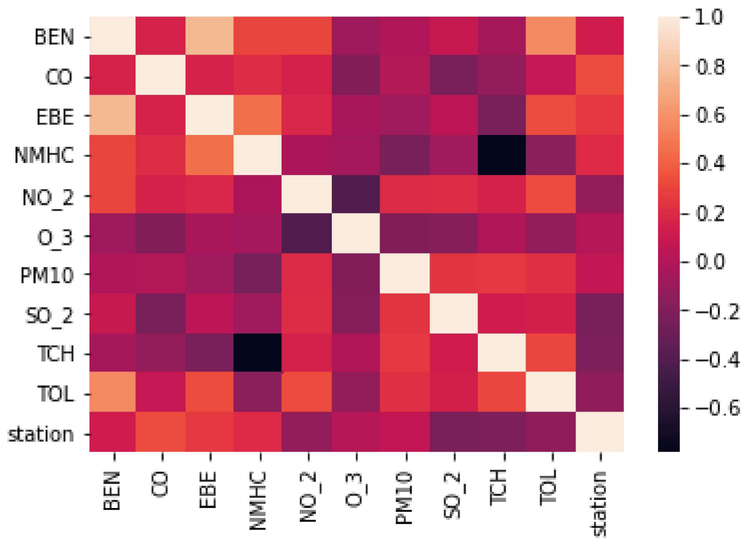
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a de
precated function and will be removed in a future version. Please adapt your code to use either `displot`
(a figure-level function with similar flexibility) or `histplot` (an axes-level function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

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



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

Out[20]: <AxesSubplot:>




## TO TRAIN THE MODEL AND MODEL BULDING

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

In [22]:
```
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 [23]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

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

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

Out[24]:
```
28079048.160485655
```

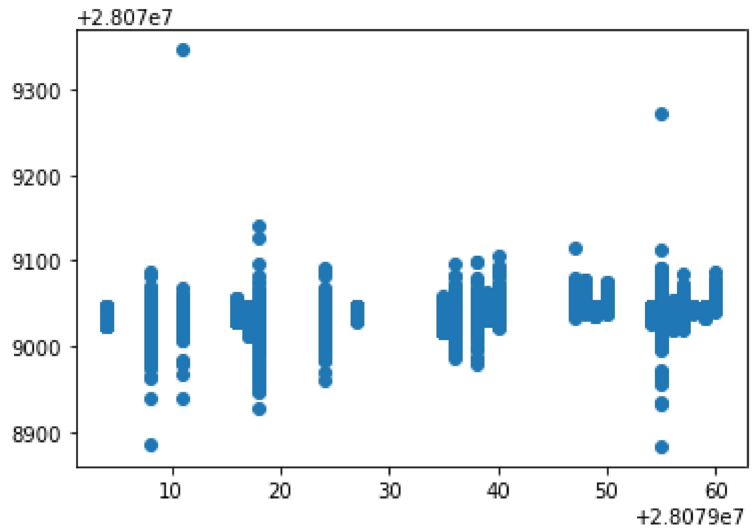
In [25]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[25]:

| | Co-efficient |
|---|---|
| **BEN** | 5.619874 |
| **CO** | 14.454426 |
| **EBE** | 12.471030 |
| **NMHC** | -11.440222 |
| **NO_2** | -0.082797 |
| **O_3** | -0.002817 |
| **PM10** | 0.256885 |
| **SO_2** | -0.641780 |
| **TCH** | -19.153854 |
| **TOL** | -2.512386 |

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

Out[26]: <matplotlib.collections.PathCollection at 0x248f4a1a2e0>




# ACCURACY

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

Out[27]: 0.307018741582505

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

Out[28]: 0.3047895675400857

# Ridge and Lasso

In [29]:
```
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]:
```
0.307001532632433
```

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

Out[32]:
```
0.3047883783577824
```

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]:
```
0.05924336469286362
```

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

Out[35]:
```
0.05759936849857572
```

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

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

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

Out[37]:
```
array([ 1.40691053,  2.0783251 ,  2.58378577,  0.43475407, -0.05869266,
       -0.02081401,  0.19385611, -0.87281204, -0.        , -1.09445944])
```

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

Out[38]:
```
28079037.862288464
```

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

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

Out[40]:
```
0.162953846501802
```

## Evaluation Metrics

In [41]:
```
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)))
```

```
13.61677829597041
260.075176054038
16.12684643859543
```

## Logistic Regression

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

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

In [44]:
```
feature_matrix.shape
```
Out[44]:
```
(210120, 10)
```

In [45]:
```
target_vector.shape
```
Out[45]:
```
(210120,)
```

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

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

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

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

In [50]:
```
prediction=logr.predict(observation)
print(prediction)
```
```
[28079018]
```

In [51]:
```
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]:
```
logr.score(fs,target_vector)
```
Out[52]:
```
0.6336855130401675
```

In [53]:
```
logr.predict_proba(observation)[0][0]
```
Out[53]:
```
3.1869195039911524e-203
```

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

Out[54]:
```
array([[3.18691950e-203, 2.22165855e-010, 4.62826902e-188,
        1.06625793e-151, 3.29027331e-092, 1.00000000e+000,
        2.21819034e-026, 4.03862272e-193, 1.48367814e-096,
        5.12259815e-077, 3.86118695e-047, 1.24733680e-161,
        5.76936555e-092, 3.74086730e-199, 3.15398013e-198,
        1.68143596e-199, 1.64434869e-201, 6.74349755e-192,
        2.71552113e-129, 8.96658400e-151, 1.98748253e-083,
        4.27215238e-205, 6.55150631e-197, 9.90226143e-095]])
```

# Random Forest

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

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

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

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

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

Out[59]:
```
0.6279948872752985
```

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','j'
```

Out[61]:
```
[Text(2232.0, 1993.2, 'CO <= 0.95\ngini = 0.958\nsamples = 92950\nvalue = [6283, 6147, 6121, 6147, 6104, 61
55, 6137, 6146, 6137\n6008, 6056, 6131, 6097, 6215, 6204, 6007, 5930, 6074\n6199, 6163, 6225, 6060, 6155, 6
183]\nclass = a'),
 Text(1116.0, 1630.8000000000002, 'NMHC <= 0.955\ngini = 0.9\nsamples = 37198\nvalue = [5896, 5834, 0, 595
7, 0, 5943, 6039, 0, 5923, 5838\n0, 5679, 0, 0, 0, 0, 0, 0, 0, 5781, 6089, 0, 0\n0]\nclass = u'),
 Text(558.0, 1268.4, 'TCH <= 1.345\ngini = 0.5\nsamples = 7395\nvalue = [0, 5810, 0, 0, 0, 0, 5921, 0, 0,
0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g'),
 Text(279.0, 906.0, 'SO_2 <= 10.5\ngini = 0.123\nsamples = 3378\nvalue = [0, 351, 0, 0, 0, 0, 4980, 0, 0,
0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g'),
 Text(139.5, 543.5999999999999, 'SO_2 <= 1.5\ngini = 0.042\nsamples = 3214\nvalue = [0, 108, 0, 0, 0, 0, 49
70, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g'),
 Text(69.75, 181.19999999999982, 'gini = 0.496\nsamples = 23\nvalue = [0, 19, 0, 0, 0, 0, 16, 0, 0, 0, 0,
0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = b'),
 Text(209.25, 181.19999999999982, 'gini = 0.035\nsamples = 3191\nvalue = [0, 89, 0, 0, 0, 0, 4954, 0, 0, 0,
0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = g'),
 Text(418.5, 543.5999999999999, 'BEN <= 0.45\ngini = 0.076\nsamples = 164\nvalue = [0, 243, 0, 0, 0, 0, 10,
0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = b'),
 Text(348.75, 181.19999999999982, 'gini = 0.0\nsamples = 105\nvalue = [0, 158, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]\nclass = b'),
```

# Conclusion

## Accuracy

***Linear Regression:0.3047895675400857***

***Ridge Regression:0.3047883783577824***

***Lasso Regression:0.05924336469286362***

***ElasticNet Regression:0.1629538465018O2***

***Logistic Regression:0.6336855130401675***

***Random Forest:0.6279948872752985***

## Logistic Regression is suitable for this dataset

```
In [ ]:
```