# 20104028

# Heamnath N

# 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_2014.csv")
df
```

Out[2]:

| | date | BEN | CO | EBE | NMHC | NO | NO_2 | O_3 | PM10 | PM25 | SO_2 | TCH | TOL | statio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2014-06-01 01:00:00 | NaN | 0.2 | NaN | NaN | 3.0 | 10.0 | NaN | NaN | NaN | 3.0 | NaN | NaN | 2807900 |
| **1** | 2014-06-01 01:00:00 | 0.2 | 0.2 | 0.1 | 0.11 | 3.0 | 17.0 | 68.0 | 10.0 | 5.0 | 5.0 | 1.36 | 1.3 | 2807900 |
| **2** | 2014-06-01 01:00:00 | 0.3 | NaN | 0.1 | NaN | 2.0 | 6.0 | NaN | NaN | NaN | NaN | NaN | 1.1 | 2807901 |
| **3** | 2014-06-01 01:00:00 | NaN | 0.2 | NaN | NaN | 1.0 | 6.0 | 79.0 | NaN | NaN | NaN | NaN | NaN | 2807901 |
| **4** | 2014-06-01 01:00:00 | NaN | NaN | NaN | NaN | 1.0 | 6.0 | 75.0 | NaN | NaN | 4.0 | NaN | NaN | 2807901 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **210019** | 2014-09-01 00:00:00 | NaN | 0.5 | NaN | NaN | 20.0 | 84.0 | 29.0 | NaN | NaN | NaN | NaN | NaN | 2807905 |
| **210020** | 2014-09-01 00:00:00 | NaN | 0.3 | NaN | NaN | 1.0 | 22.0 | NaN | 15.0 | NaN | 6.0 | NaN | NaN | 2807905 |
| **210021** | 2014-09-01 00:00:00 | NaN | NaN | NaN | NaN | 1.0 | 13.0 | 70.0 | NaN | NaN | NaN | NaN | NaN | 2807905 |

| | date | BEN | CO | EBE | NMHC | NO | NO_2 | O_3 | PM10 | PM25 | SO_2 | TCH | TOL | statio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **210022** | 2014-09-01 00:00:00 | NaN | NaN | NaN | NaN | 3.0 | 38.0 | 42.0 | NaN | NaN | NaN | NaN | NaN | 2807905 |
| **210023** | 2014-09-01 00:00:00 | NaN | NaN | NaN | NaN | 1.0 | 26.0 | 65.0 | 11.0 | NaN | NaN | NaN | NaN | 2807906 |

210024 rows × 14 columns

# Data Cleaning and Data Preprocessing

In [3]:
```
df=df.dropna()
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13946 entries, 1 to 210006
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     13946 non-null  object 
 1   BEN      13946 non-null  float64
 2   CO       13946 non-null  float64
 3   EBE      13946 non-null  float64
 4   NMHC     13946 non-null  float64
 5   NO       13946 non-null  float64
 6   NO_2     13946 non-null  float64
 7   O_3      13946 non-null  float64
 8   PM10     13946 non-null  float64
 9   PM25     13946 non-null  float64
 10  SO_2     13946 non-null  float64
 11  TCH      13946 non-null  float64
 12  TOL      13946 non-null  float64
 13  station  13946 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.6+ MB
```

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

Out[6]:

| | CO | station |
|---|---|---|
| **1** | 0.2 | 28079008 |
| **6** | 0.2 | 28079024 |

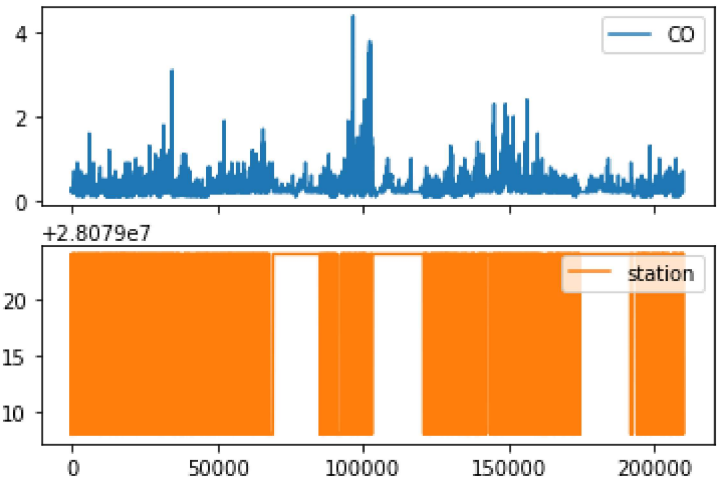
| | CO | station |
|---|---|---|
| **25** | 0.2 | 28079008 |
| **30** | 0.2 | 28079024 |
| **49** | 0.2 | 28079008 |
| **...** | ... | ... |
| **209958** | 0.2 | 28079024 |
| **209977** | 0.7 | 28079008 |
| **209982** | 0.2 | 28079024 |
| **210001** | 0.4 | 28079008 |
| **210006** | 0.2 | 28079024 |

13946 rows × 2 columns

# Line chart

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

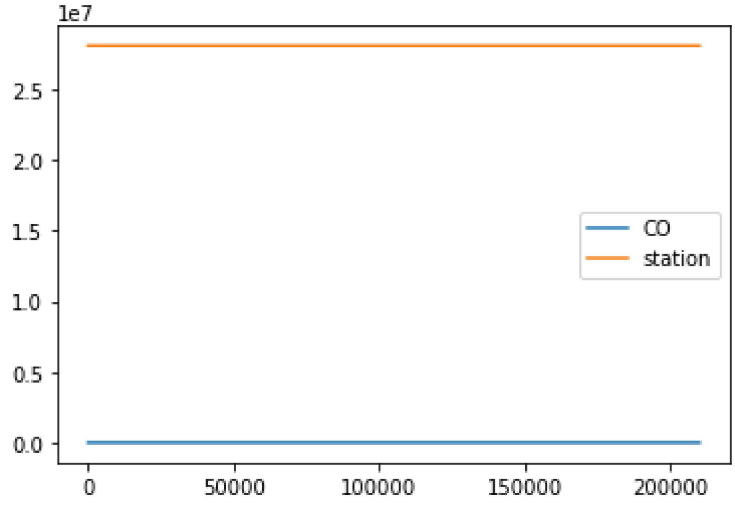
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```




# Line chart

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

Out[8]:
```
<AxesSubplot:>
```

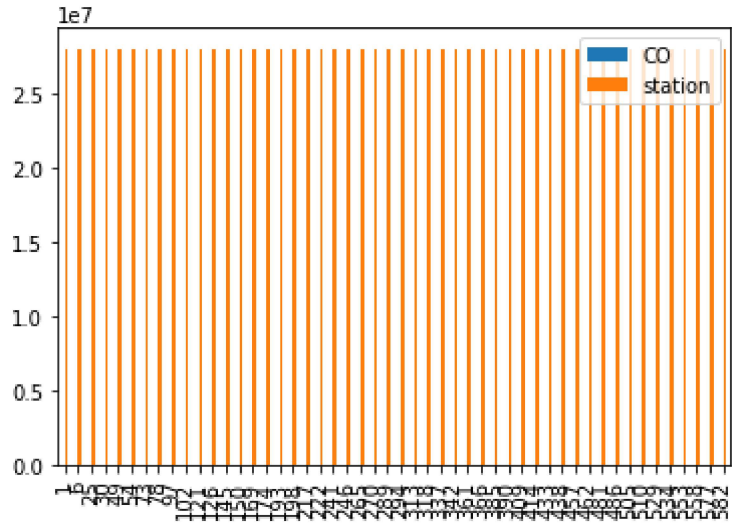



# Bar chart

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

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

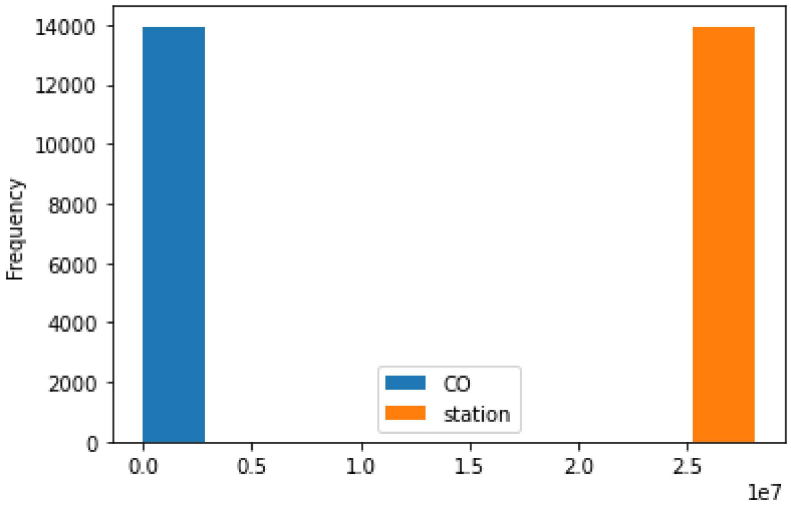
Out[10]: <AxesSubplot:>




# Histogram

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

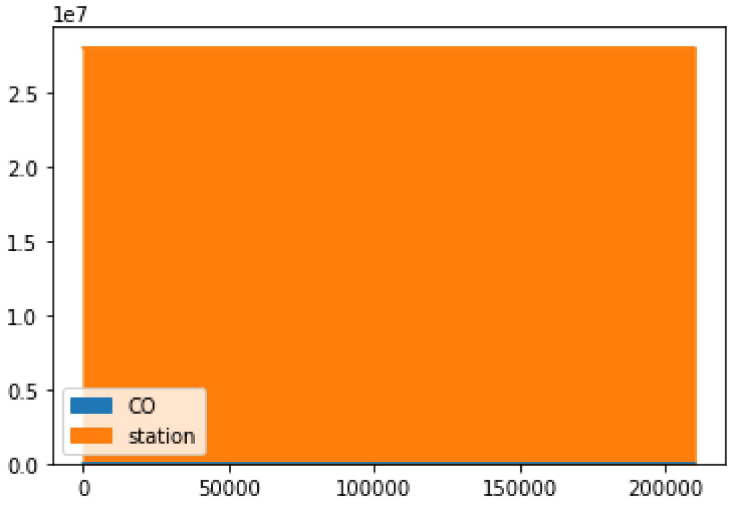
Out[11]: <AxesSubplot:ylabel='Frequency'>




# Area chart

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

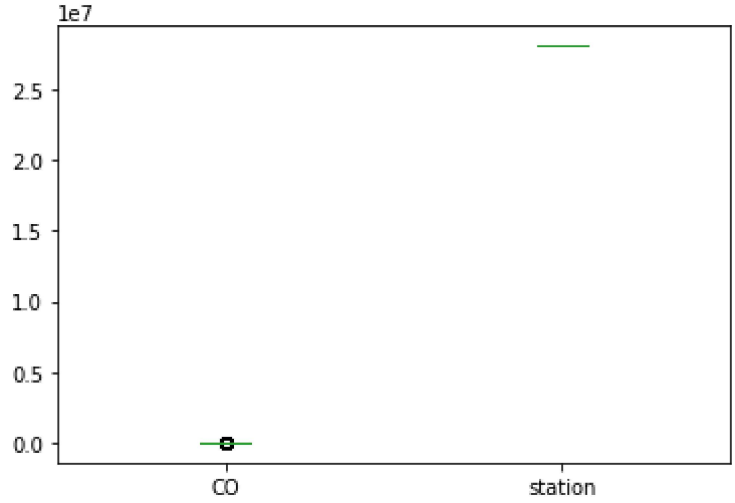
Out[12]: <AxesSubplot:>




# Box chart

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

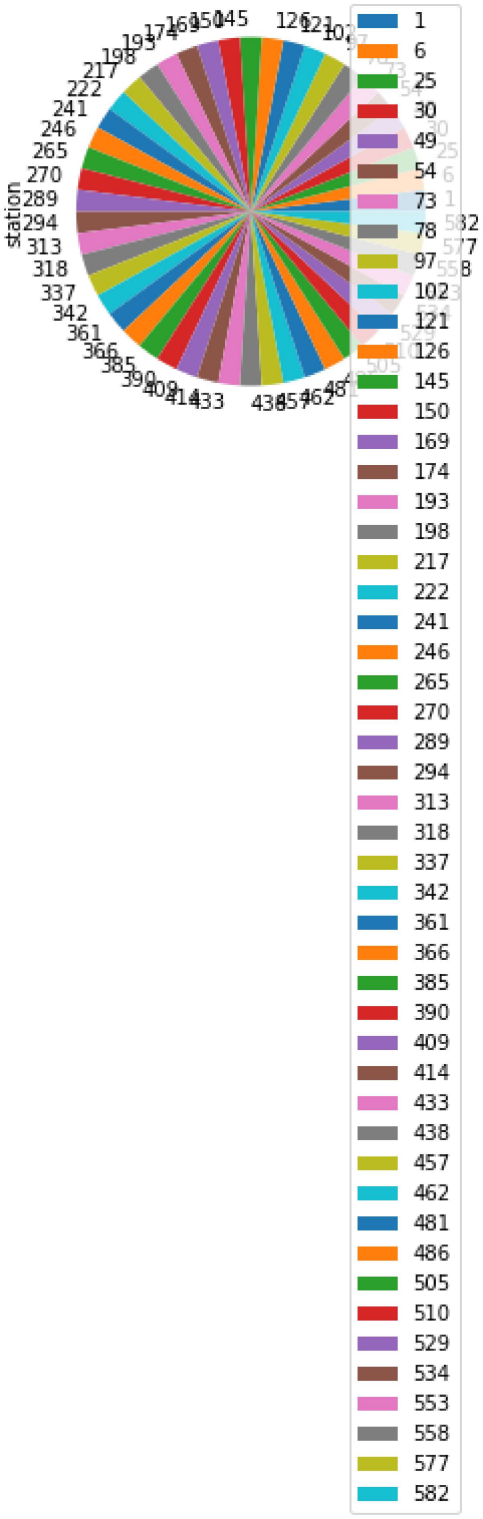
Out[13]: <AxesSubplot:>




# Pie chart

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

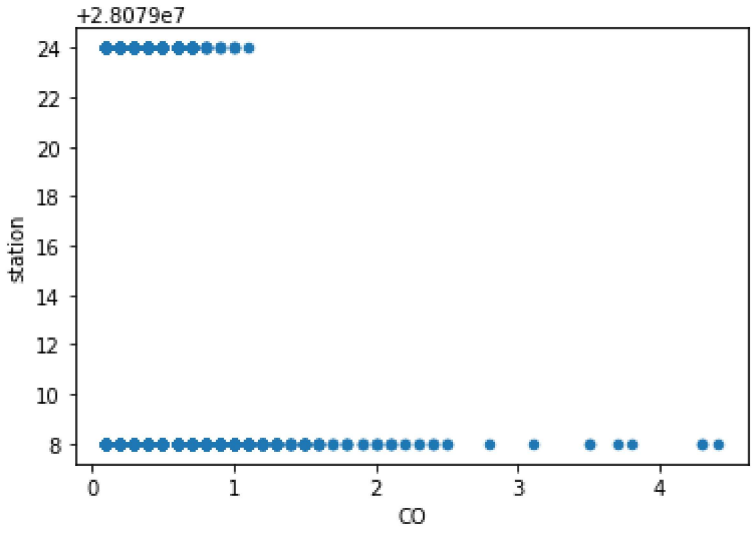
Out[14]: <AxesSubplot:ylabel='station'>




# 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: 13946 entries, 1 to 210006
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     13946 non-null  object 
 1   BEN      13946 non-null  float64
 2   CO       13946 non-null  float64
 3   EBE      13946 non-null  float64
 4   NMHC     13946 non-null  float64
 5   NO       13946 non-null  float64
 6   NO_2     13946 non-null  float64
 7   O_3      13946 non-null  float64
 8   PM10     13946 non-null  float64
 9   PM25     13946 non-null  float64
 10  SO_2     13946 non-null  float64
 11  TCH      13946 non-null  float64
 12  TOL      13946 non-null  float64
 13  station  13946 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.6+ MB
```

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

Out[17]:

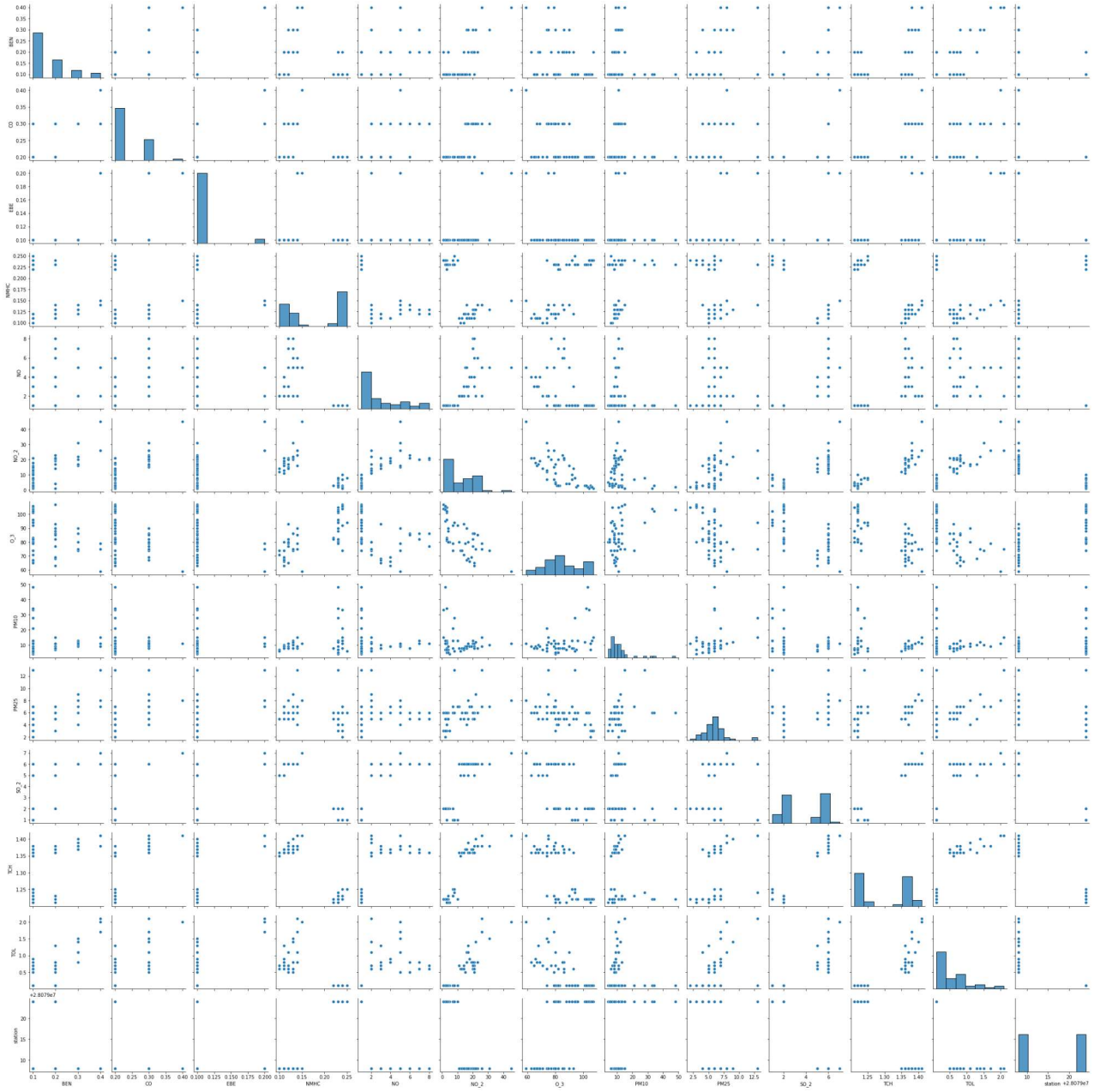
| | BEN | CO | EBE | NMHC | NO | NO_2 | O_3 |
|---|---|---|---|---|---|---|---|
| **count** | 13946.000000 | 13946.000000 | 13946.000000 | 13946.000000 | 13946.000000 | 13946.000000 | 13946.000000 |
| **mean** | 0.375921 | 0.314793 | 0.306016 | 0.222302 | 17.589129 | 34.240929 | 53.08238 |
| **std** | 0.555093 | 0.207375 | 0.635475 | 0.082403 | 39.432216 | 30.654229 | 33.48816 |
| **min** | 0.100000 | 0.100000 | 0.100000 | 0.060000 | 1.000000 | 1.000000 | 1.000000 |
| **25%** | 0.100000 | 0.200000 | 0.100000 | 0.160000 | 1.000000 | 10.000000 | 25.000000 |
| **50%** | 0.200000 | 0.300000 | 0.100000 | 0.230000 | 4.000000 | 27.000000 | 53.000000 |
| **75%** | 0.400000 | 0.400000 | 0.300000 | 0.260000 | 18.000000 | 51.000000 | 75.000000 |

| | BEN | CO | EBE | NMHC | NO | NO_2 | O_3 |
|---|---|---|---|---|---|---|---|
| **max** | 9.400000 | 4.400000 | 16.200001 | 1.290000 | 725.000000 | 346.000000 | 220.00000 |

# EDA AND VISUALIZATION

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

Out[18]: <seaborn.axisgrid.PairGrid at 0x2ba44d30f40>

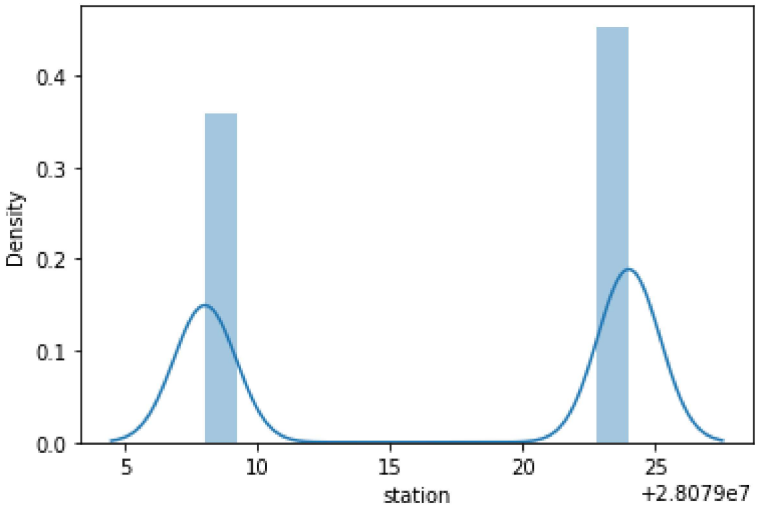


In [19]:
```
sns.distplot(df['station'])
```

```
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 adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
```

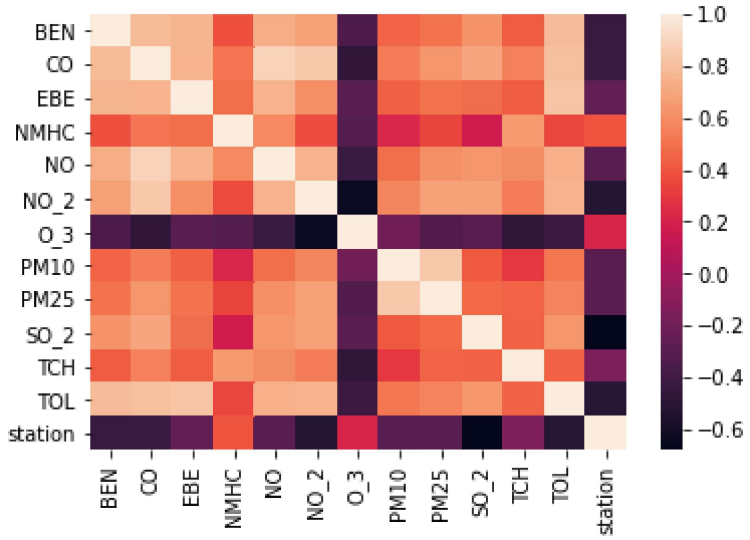
```
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

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




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

Out[20]: <AxesSubplot:>




# TO TRAIN THE MODEL AND MODEL BULDING

In [21]:
```
x=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       '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]:
```
28079024.546470482
```

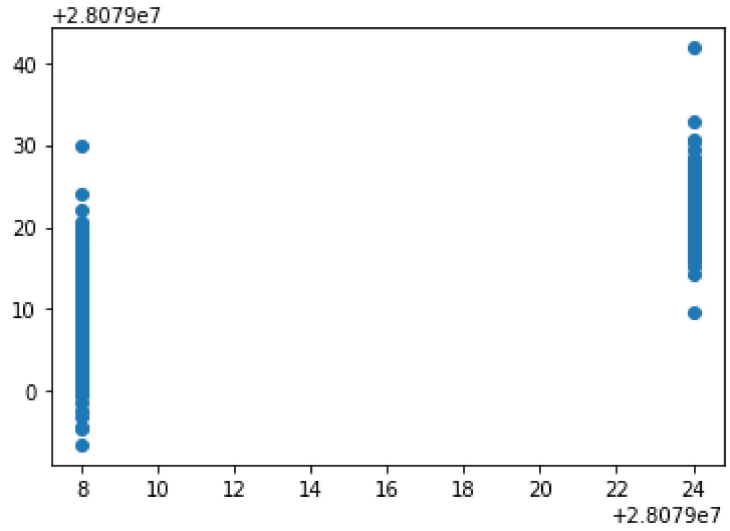
In [25]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[25]:

| | Co-efficient |
|---|---|
| **BEN** | -1.325877 |
| **CO** | -9.369359 |
| **EBE** | 0.299750 |
| **NMHC** | 80.899880 |
| **NO** | 0.029987 |
| **NO_2** | -0.038153 |
| **O_3** | 0.001825 |
| **PM10** | -0.018816 |
| **PM25** | 0.124421 |
| **SO_2** | -0.886393 |
| **TCH** | -12.631549 |
| **TOL** | -0.471852 |

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

Out[26]:
```
<matplotlib.collections.PathCollection at 0x2ba4fbfd6d0>
```




# ACCURACY

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

Out[27]: 0.8940375191851825

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

Out[28]: 0.8899739054060595

# 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]: 0.8717932607663217

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

Out[32]: 0.8671529216592928

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

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

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

Out[34]:
```
0.29813313311569833
```

# Accuracy(Lasso)

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

Out[35]:
```
0.29649352632089576
```

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([-0.        , -0.        ,  0.        ,  0.        ,  0.10752922,
       -0.10505727, -0.01547767, -0.        ,  0.06404522, -1.41543434,
        0.        , -0.58204658])
```

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

Out[38]:
```
28079026.650656264
```

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

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

Out[40]:
```
0.5991794130246936
```

# 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)))
```

```
4.137813632230472
25.347814849072286
5.034661344030231
```

# Logistic Regression

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

In [43]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

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

Out[44]: (13946, 12)

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

Out[45]: (13946,)

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=[[1,2,3,4,5,6,7,8,9,10,11,12]]
```

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

```
[28079008]
```

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

Out[51]: array([28079008, 28079024], dtype=int64)

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

Out[52]: 0.9930446006023232

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

Out[53]:
```
1.0
```

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

Out[54]:
```
array([[1.00000000e+00, 1.92542107e-22]])
```

# 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]:
```
0.9963122311001844
```

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

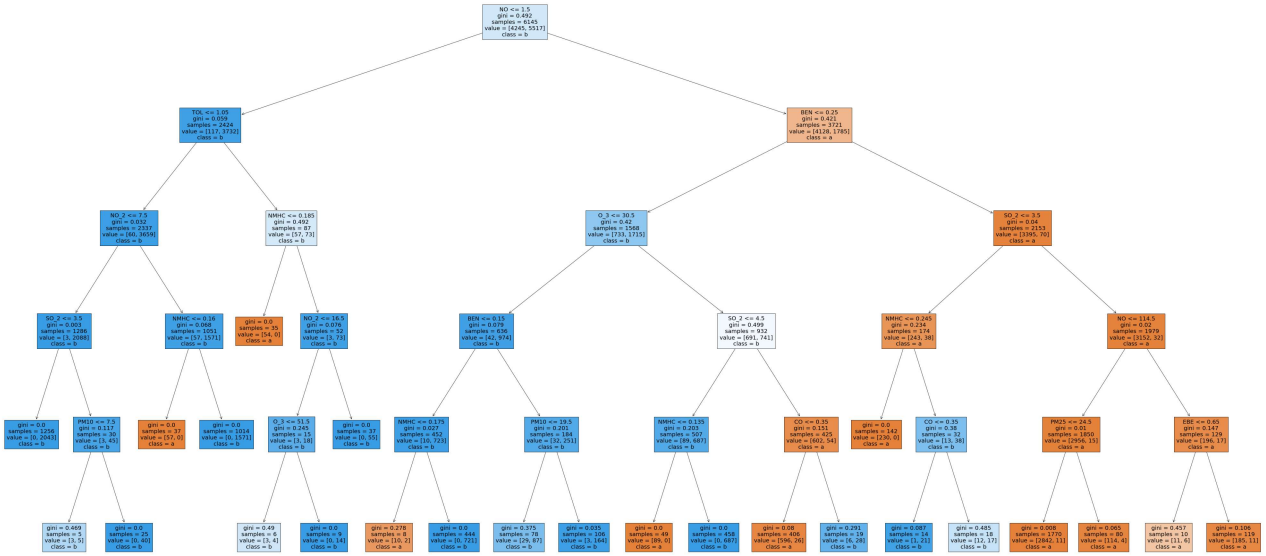
In [61]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b','c','d']
```

Out[61]:
```
[Text(1817.076923076923, 1993.2, 'NO <= 1.5\ngini = 0.492\nsamples = 6145\nvalue = [424
5, 5517]\nclass = b'),
 Text(744.0, 1630.8000000000002, 'TOL <= 1.05\ngini = 0.059\nsamples = 2424\nvalue = [11
7, 3732]\nclass = b'),
```

```
 Text(457.84615384615387, 1268.4, 'NO_2 <= 7.5\ngini = 0.032\nsamples = 2337\nvalue = [60, 3659]\nclass = b'),
 Text(228.92307692307693, 906.0, 'SO_2 <= 3.5\ngini = 0.003\nsamples = 1286\nvalue = [3, 2088]\nclass = b'),
 Text(114.46153846153847, 543.5999999999999, 'gini = 0.0\nsamples = 1256\nvalue = [0, 2043]\nclass = b'),
 Text(343.3846153846154, 543.5999999999999, 'PM10 <= 7.5\ngini = 0.117\nsamples = 30\nvalue = [3, 45]\nclass = b'),
 Text(228.92307692307693, 181.19999999999982, 'gini = 0.469\nsamples = 5\nvalue = [3, 5]\nclass = b'),
 Text(457.84615384615387, 181.19999999999982, 'gini = 0.0\nsamples = 25\nvalue = [0, 40]\nclass = b'),
 Text(686.7692307692308, 906.0, 'NMHC <= 0.16\ngini = 0.068\nsamples = 1051\nvalue = [57, 1571]\nclass = b'),
 Text(572.3076923076924, 543.5999999999999, 'gini = 0.0\nsamples = 37\nvalue = [57, 0]\nclass = a'),
 Text(801.2307692307693, 543.5999999999999, 'gini = 0.0\nsamples = 1014\nvalue = [0, 1571]\nclass = b'),
 Text(1030.1538461538462, 1268.4, 'NMHC <= 0.185\ngini = 0.492\nsamples = 87\nvalue = [57, 73]\nclass = b'),
 Text(915.6923076923077, 906.0, 'gini = 0.0\nsamples = 35\nvalue = [54, 0]\nclass = a'),
 Text(1144.6153846153848, 906.0, 'NO_2 <= 16.5\ngini = 0.076\nsamples = 52\nvalue = [3, 73]\nclass = b'),
 Text(1030.1538461538462, 543.5999999999999, 'O_3 <= 51.5\ngini = 0.245\nsamples = 15\nvalue = [3, 18]\nclass = b'),
 Text(915.6923076923077, 181.19999999999982, 'gini = 0.49\nsamples = 6\nvalue = [3, 4]\nclass = b'),
 Text(1144.6153846153848, 181.19999999999982, 'gini = 0.0\nsamples = 9\nvalue = [0, 14]\nclass = b'),
 Text(1259.076923076923, 543.5999999999999, 'gini = 0.0\nsamples = 37\nvalue = [0, 55]\nclass = b'),
 Text(2890.153846153846, 1630.8000000000002, 'BEN <= 0.25\ngini = 0.421\nsamples = 3721\nvalue = [4128, 1785]\nclass = a'),
 Text(2174.769230769231, 1268.4, 'O_3 <= 30.5\ngini = 0.42\nsamples = 1568\nvalue = [733, 1715]\nclass = b'),
 Text(1716.923076923077, 906.0, 'BEN <= 0.15\ngini = 0.079\nsamples = 636\nvalue = [42, 974]\nclass = b'),
 Text(1488.0, 543.5999999999999, 'NMHC <= 0.175\ngini = 0.027\nsamples = 452\nvalue = [10, 723]\nclass = b'),
 Text(1373.5384615384617, 181.19999999999982, 'gini = 0.278\nsamples = 8\nvalue = [10, 2]\nclass = a'),
 Text(1602.4615384615386, 181.19999999999982, 'gini = 0.0\nsamples = 444\nvalue = [0, 721]\nclass = b'),
 Text(1945.846153846154, 543.5999999999999, 'PM10 <= 19.5\ngini = 0.201\nsamples = 184\nvalue = [32, 251]\nclass = b'),
 Text(1831.3846153846155, 181.19999999999982, 'gini = 0.375\nsamples = 78\nvalue = [29, 87]\nclass = b'),
 Text(2060.3076923076924, 181.19999999999982, 'gini = 0.035\nsamples = 106\nvalue = [3, 164]\nclass = b'),
 Text(2632.6153846153848, 906.0, 'SO_2 <= 4.5\ngini = 0.499\nsamples = 932\nvalue = [691, 741]\nclass = b'),
 Text(2403.6923076923076, 543.5999999999999, 'NMHC <= 0.135\ngini = 0.203\nsamples = 507\nvalue = [89, 687]\nclass = b'),
 Text(2289.2307692307695, 181.19999999999982, 'gini = 0.0\nsamples = 49\nvalue = [89, 0]\nclass = a'),
 Text(2518.153846153846, 181.19999999999982, 'gini = 0.0\nsamples = 458\nvalue = [0, 687]\nclass = b'),
 Text(2861.538461538462, 543.5999999999999, 'CO <= 0.35\ngini = 0.151\nsamples = 425\nvalue = [602, 54]\nclass = a'),
 Text(2747.0769230769233, 181.19999999999982, 'gini = 0.08\nsamples = 406\nvalue = [596, 26]\nclass = a'),
 Text(2976.0, 181.19999999999982, 'gini = 0.291\nsamples = 19\nvalue = [6, 28]\nclass = b'),
 Text(3605.538461538462, 1268.4, 'SO_2 <= 3.5\ngini = 0.04\nsamples = 2153\nvalue = [3395, 70]\nclass = a'),
```

```
 Text(3204.923076923077, 906.0, 'NMHC <= 0.245\ngini = 0.234\nsamples = 174\nvalue = [24
3, 38]\nclass = a'),
 Text(3090.4615384615386, 543.5999999999999, 'gini = 0.0\nsamples = 142\nvalue = [230,
0]\nclass = a'),
 Text(3319.3846153846157, 543.5999999999999, 'CO <= 0.35\ngini = 0.38\nsamples = 32\nval
ue = [13, 38]\nclass = b'),
 Text(3204.923076923077, 181.19999999999982, 'gini = 0.087\nsamples = 14\nvalue = [1, 2
1]\nclass = b'),
 Text(3433.846153846154, 181.19999999999982, 'gini = 0.485\nsamples = 18\nvalue = [12, 1
7]\nclass = b'),
 Text(4006.153846153846, 906.0, 'NO <= 114.5\ngini = 0.02\nsamples = 1979\nvalue = [315
2, 32]\nclass = a'),
 Text(3777.2307692307695, 543.5999999999999, 'PM25 <= 24.5\ngini = 0.01\nsamples = 1850
\nvalue = [2956, 15]\nclass = a'),
 Text(3662.769230769231, 181.19999999999982, 'gini = 0.008\nsamples = 1770\nvalue = [284
2, 11]\nclass = a'),
 Text(3891.692307692308, 181.19999999999982, 'gini = 0.065\nsamples = 80\nvalue = [114,
4]\nclass = a'),
 Text(4235.076923076923, 543.5999999999999, 'EBE <= 0.65\ngini = 0.147\nsamples = 129\nv
alue = [196, 17]\nclass = a'),
 Text(4120.615384615385, 181.19999999999982, 'gini = 0.457\nsamples = 10\nvalue = [11,
6]\nclass = a'),
 Text(4349.538461538462, 181.19999999999982, 'gini = 0.106\nsamples = 119\nvalue = [185,
11]\nclass = a')]
```



# Conclusion

# Accuracy

In [62]:

```
print("Linear Regression:",lr.score(x_test,y_test))
print("Ridge Regression:",rr.score(x_test,y_test))
print("Lasso Regression",la.score(x_test,y_test))
print("ElasticNet Regression:",en.score(x_test,y_test))
print("Logistic Regression:",logr.score(fs,target_vector))
print("Random Forest:",grid_search.best_score_)
```

```
Linear Regression: 0.8940375191851825
Ridge Regression: 0.8717932607663217
Lasso Regression 0.29649352632089576
```

```
ElasticNet Regression: 0.5991794130246936
Logistic Regression: 0.9930446006023232
Random Forest: 0.9963122311001844
```

# Random Forest is suitable for this dataset