# 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(r"C:\Users\user\Downloads\madrid_2008.csv")
df
```

Out[2]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 | PM2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 01-06-2008 01:00 | NaN | 0.47 | NaN | NaN | NaN | 83.089996 | 120.699997 | NaN | 16.990000 | 16.889999 | 10.4 |
| **1** | 01-06-2008 01:00 | NaN | 0.59 | NaN | NaN | NaN | 94.820000 | 130.399994 | NaN | 17.469999 | 19.040001 | Na |
| **2** | 01-06-2008 01:00 | NaN | 0.55 | NaN | NaN | NaN | 75.919998 | 104.599998 | NaN | 13.470000 | 20.270000 | Na |
| **3** | 01-06-2008 01:00 | NaN | 0.36 | NaN | NaN | NaN | 61.029999 | 66.559998 | NaN | 23.110001 | 10.850000 | Na |
| **4** | 01-06-2008 01:00 | 1.68 | 0.80 | 1.70 | 3.01 | 0.30 | 105.199997 | 214.899994 | 1.61 | 12.120000 | 37.160000 | 21.9 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **226387** | 01-11-2008 00:00 | 0.48 | 0.30 | 0.57 | 1.00 | 0.31 | 13.050000 | 14.160000 | 0.91 | 57.400002 | 5.450000 | 5.1 |
| **226388** | 01-11-2008 00:00 | NaN | 0.30 | NaN | NaN | NaN | 41.880001 | 48.500000 | NaN | 35.830002 | 15.020000 | Na |
| **226389** | 01-11-2008 00:00 | 0.25 | NaN | 0.56 | NaN | 0.11 | 83.610001 | 102.199997 | NaN | 14.130000 | 17.540001 | 13.9 |

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 | PM2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **226390** | 01-11-2008 00:00 | 0.54 | NaN | 2.70 | NaN | 0.18 | 70.639999 | 81.860001 | NaN | NaN | 11.910000 | Na |
| **226391** | 01-11-2008 00:00 | 0.75 | 0.36 | 1.20 | 2.75 | 0.16 | 58.240002 | 74.239998 | 1.64 | 31.910000 | 12.690000 | 11.4 |

226392 rows × 17 columns

# Data Cleaning and Data Preprocessing

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

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

Out[4]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25631 entries, 4 to 226391
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     25631 non-null  object 
 1   BEN      25631 non-null  float64
 2   CO       25631 non-null  float64
 3   EBE      25631 non-null  float64
 4   MXY      25631 non-null  float64
 5   NMHC     25631 non-null  float64
 6   NO_2     25631 non-null  float64
 7   NOx      25631 non-null  float64
 8   OXY      25631 non-null  float64
 9   O_3      25631 non-null  float64
 10  PM10     25631 non-null  float64
 11  PM25     25631 non-null  float64
 12  PXY      25631 non-null  float64
 13  SO_2     25631 non-null  float64
 14  TCH      25631 non-null  float64
 15  TOL      25631 non-null  float64
 16  station  25631 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.5+ MB
```

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

Out[6]:

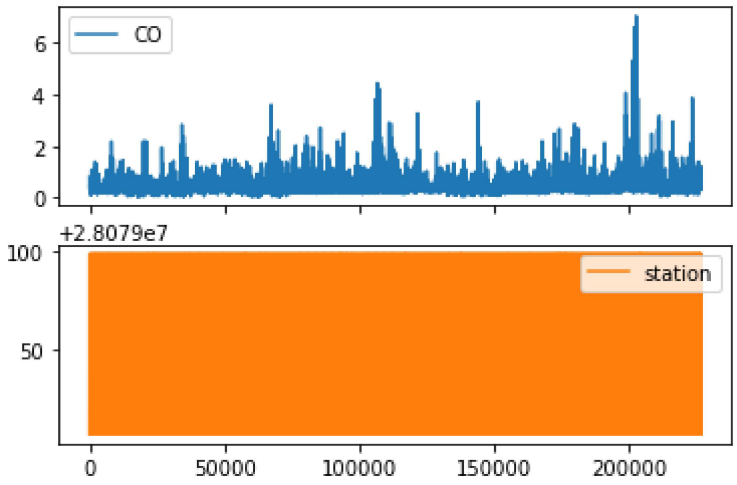
| | CO | station |
|---|---|---|
| **4** | 0.80 | 28079006 |
| **21** | 0.37 | 28079024 |
| **25** | 0.39 | 28079099 |
| **30** | 0.51 | 28079006 |
| **47** | 0.39 | 28079024 |
| **...** | ... | ... |
| **226362** | 0.35 | 28079024 |
| **226366** | 0.46 | 28079099 |
| **226371** | 0.53 | 28079006 |
| **226387** | 0.30 | 28079024 |
| **226391** | 0.36 | 28079099 |

25631 rows × 2 columns

# Line chart

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

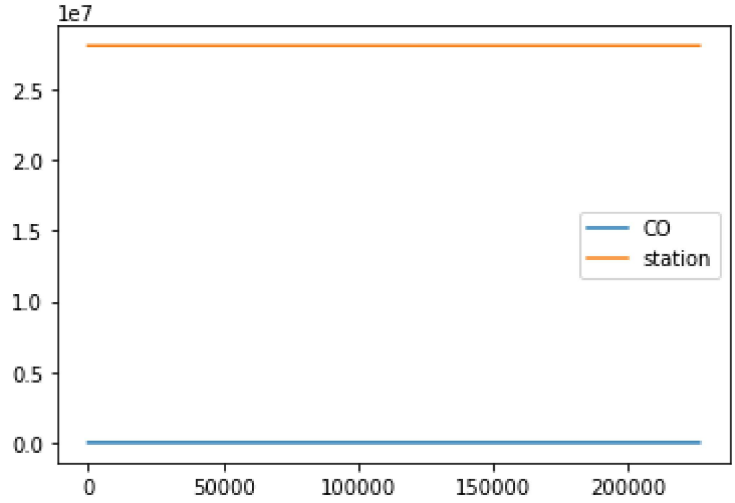
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```




# Line chart

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

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

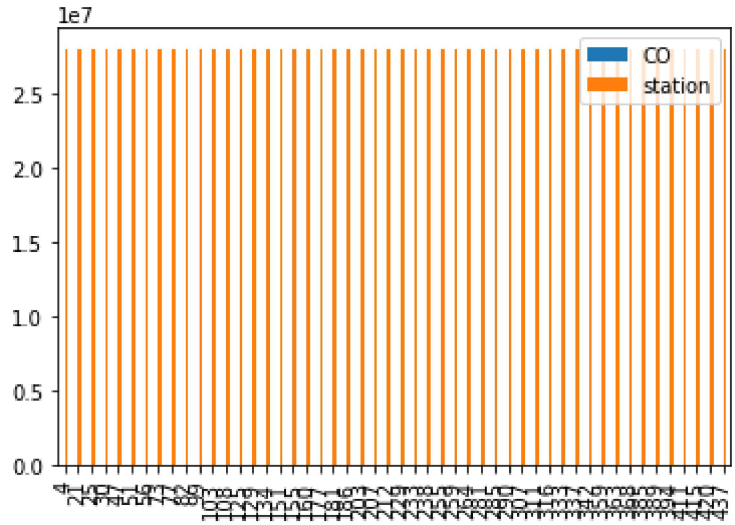



# Bar chart

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

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

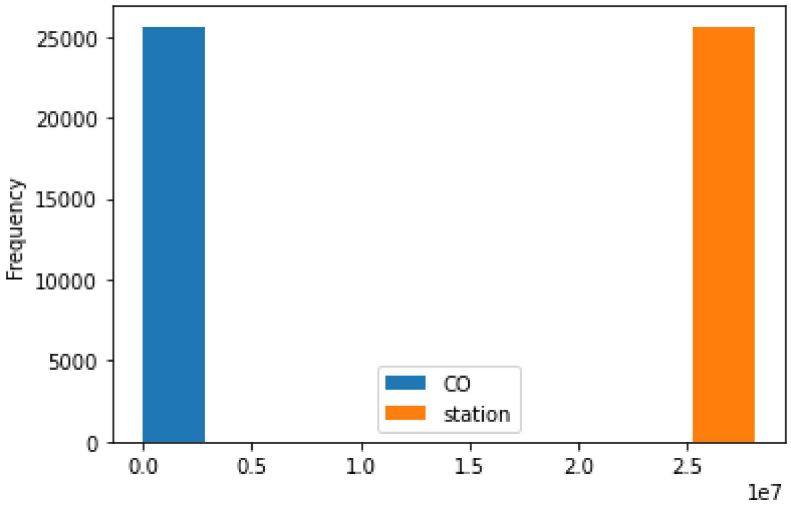
Out[10]: <AxesSubplot:>




# Histogram

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

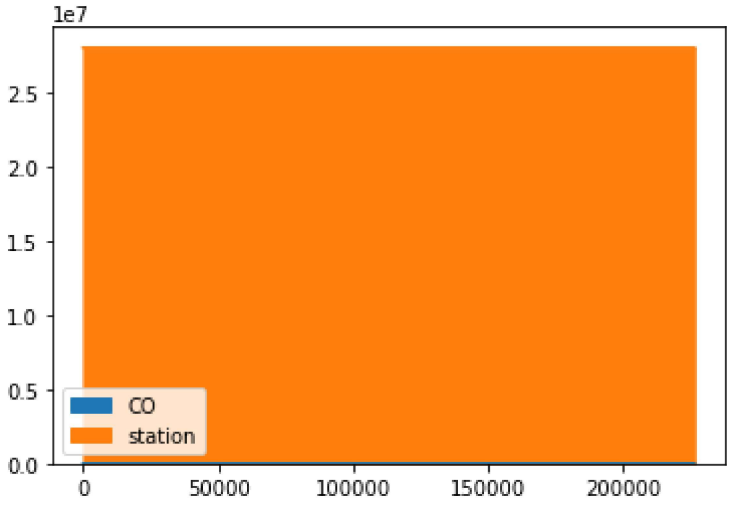
Out[11]: <AxesSubplot:ylabel='Frequency'>




# Area chart

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

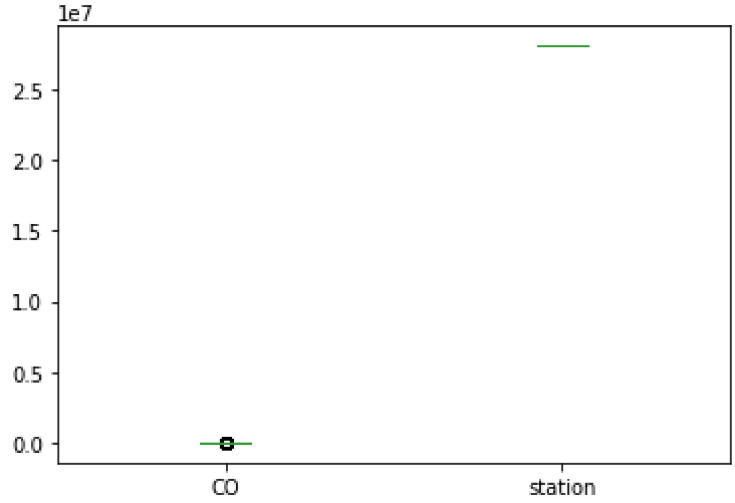
Out[12]: <AxesSubplot:>




# Box chart

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

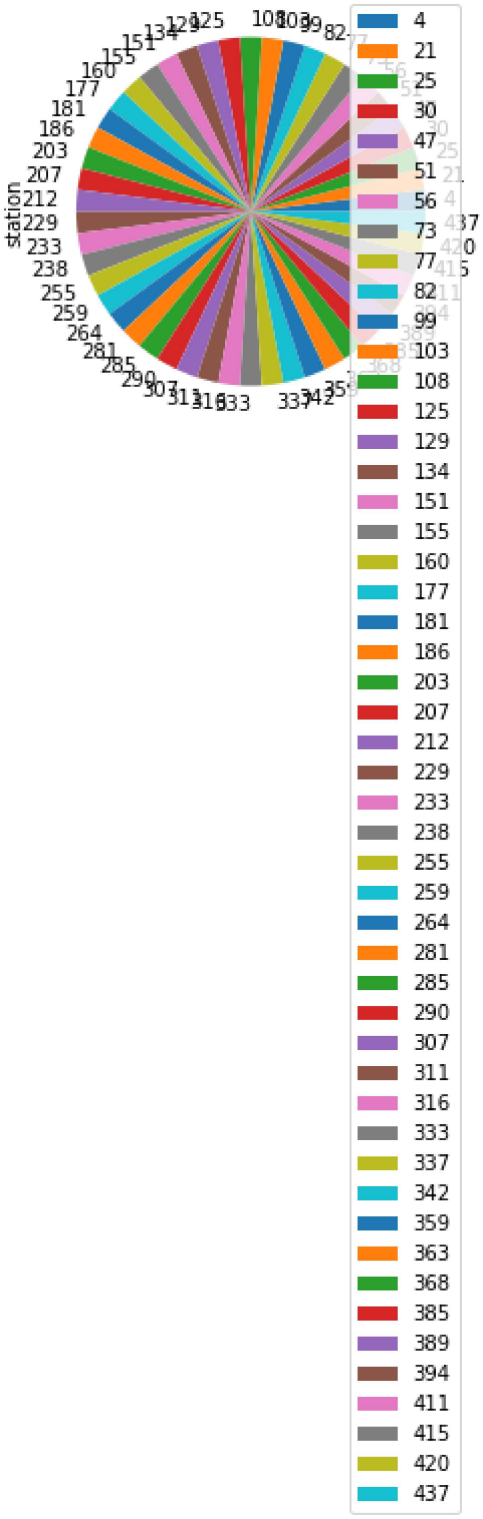
Out[13]: <AxesSubplot:>




# Pie chart

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

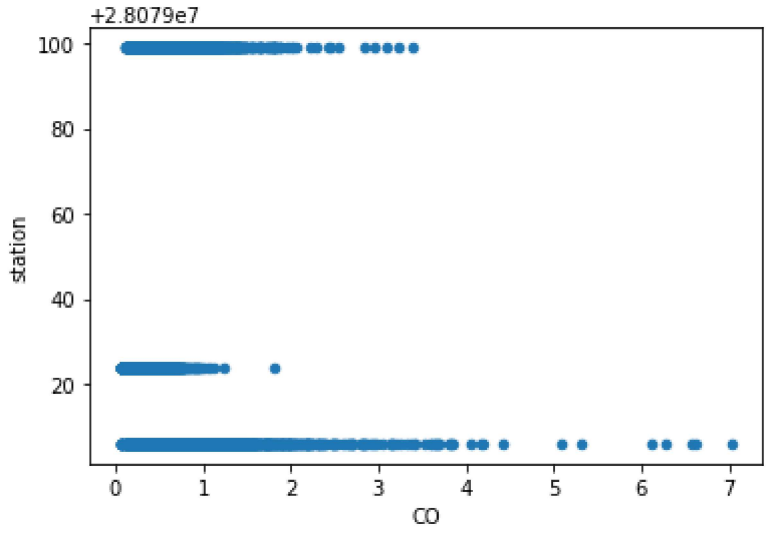
Out[14]:
```
<AxesSubplot:ylabel='station'>
```




# 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: 25631 entries, 4 to 226391
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     25631 non-null  object 
 1   BEN      25631 non-null  float64
 2   CO       25631 non-null  float64
 3   EBE      25631 non-null  float64
 4   MXY      25631 non-null  float64
 5   NMHC     25631 non-null  float64
 6   NO_2     25631 non-null  float64
 7   NOx      25631 non-null  float64
 8   OXY      25631 non-null  float64
 9   O_3      25631 non-null  float64
 10  PM10     25631 non-null  float64
 11  PM25     25631 non-null  float64
 12  PXY      25631 non-null  float64
 13  SO_2     25631 non-null  float64
 14  TCH      25631 non-null  float64
 15  TOL      25631 non-null  float64
 16  station  25631 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.5+ MB
```

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

Out[17]:

| | BEN | CO | EBE | MXY | NMHC | NO_2 | NO |
|---|---|---|---|---|---|---|---|
| **count** | 25631.000000 | 25631.000000 | 25631.000000 | 25631.000000 | 25631.000000 | 25631.000000 | 25631.00000 |
| **mean** | 1.090541 | 0.440632 | 1.352355 | 2.446045 | 0.213323 | 54.225261 | 98.00773 |
| **std** | 1.146461 | 0.317853 | 1.118191 | 2.390023 | 0.123409 | 38.164647 | 101.44823 |
| **min** | 0.100000 | 0.060000 | 0.170000 | 0.240000 | 0.000000 | 0.240000 | 2.11000 |
| **25%** | 0.430000 | 0.260000 | 0.740000 | 1.000000 | 0.130000 | 25.719999 | 32.63500 |
| **50%** | 0.750000 | 0.350000 | 1.000000 | 1.620000 | 0.190000 | 48.000000 | 71.11000 |

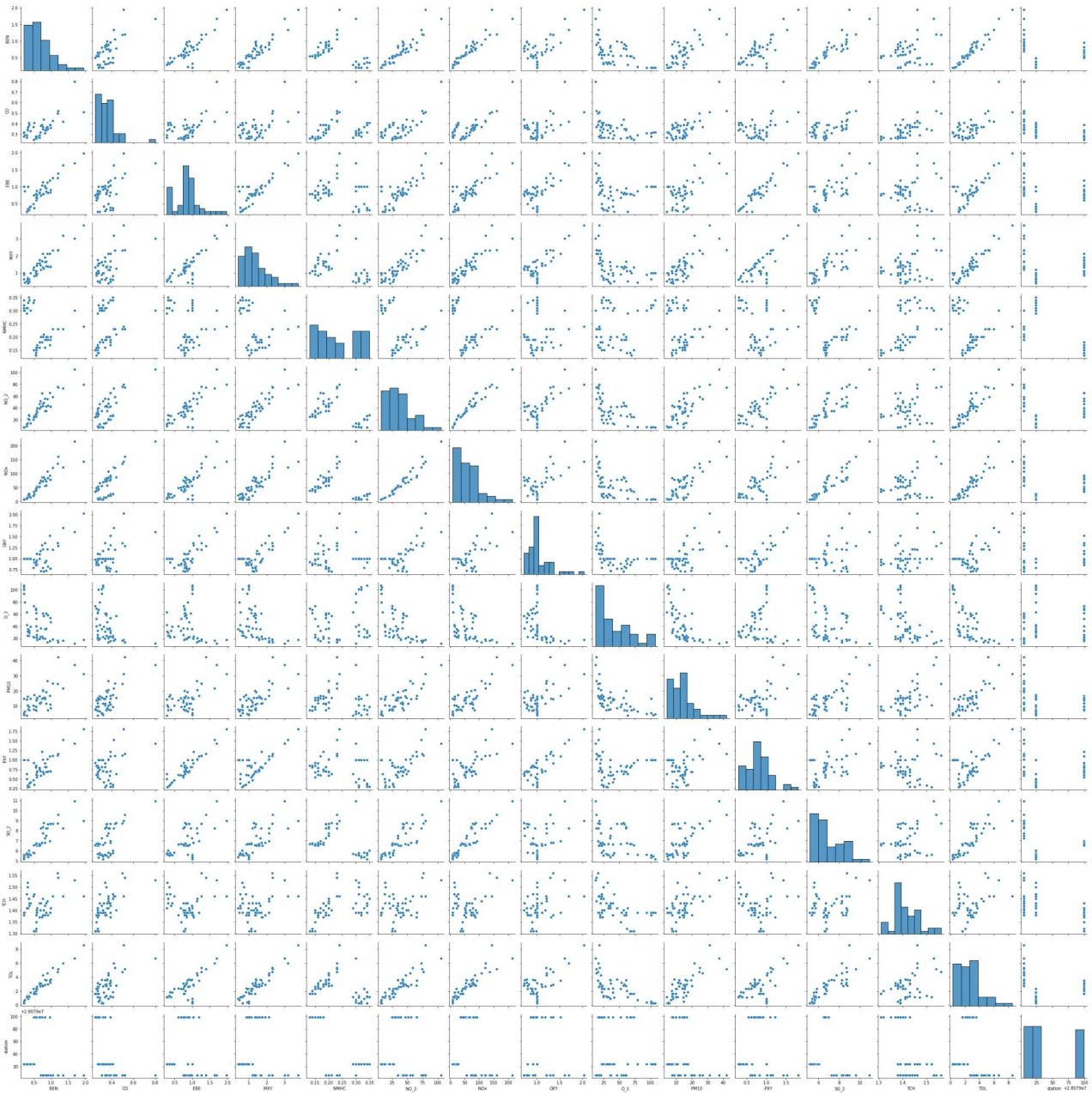
| | BEN | CO | EBE | MXY | NMHC | NO_2 | NO |
|---|---|---|---|---|---|---|---|
| **75%** | 1.320000 | 0.510000 | 1.580000 | 3.105000 | 0.270000 | 74.924999 | 131.55000 |
| **max** | 27.230000 | 7.030000 | 26.740000 | 55.889999 | 1.760000 | 554.900024 | 2004.00000 |

In [18]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

# EDA AND VISUALIZATION

In [19]:
```
sns.pairplot(df1[0:50])
```

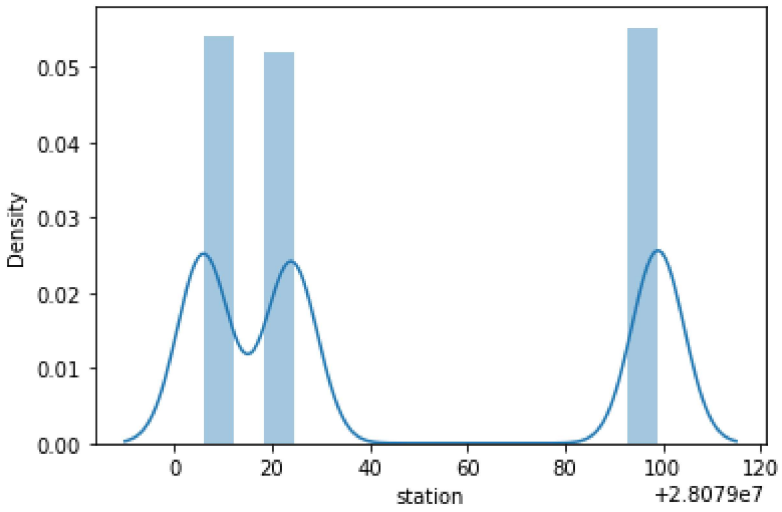
Out[19]: <seaborn.axisgrid.PairGrid at 0x2ded38d36d0>



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

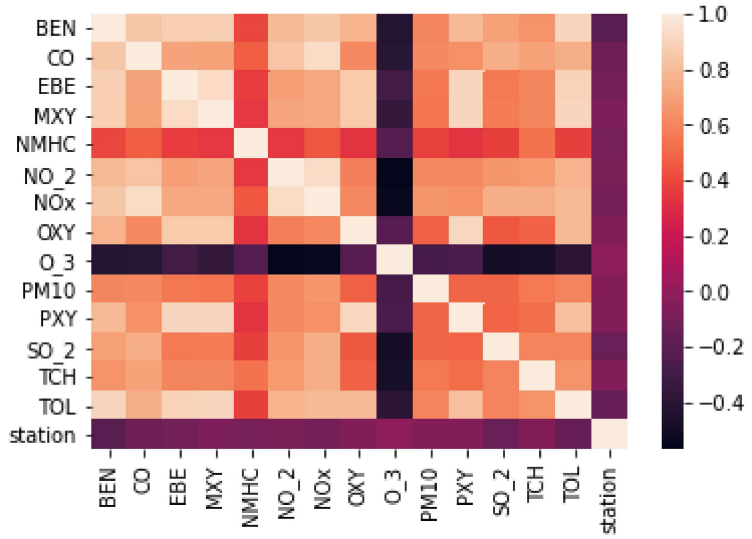
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function 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', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', '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]:
```
28079036.397904124
```

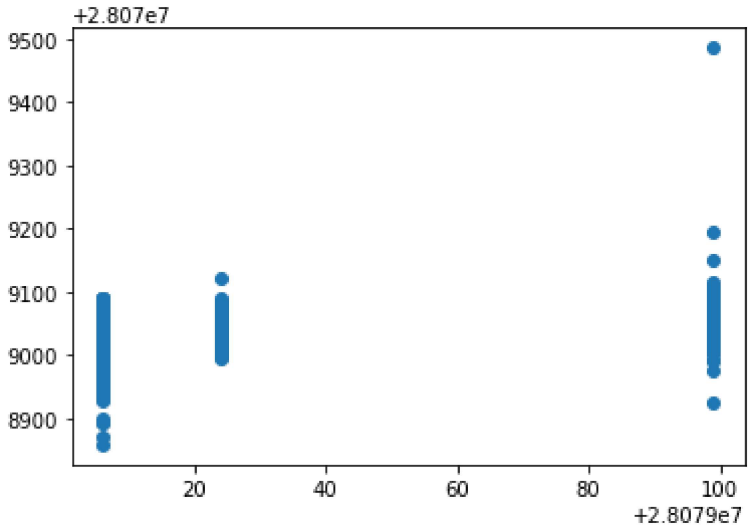
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

| | Co-efficient |
|---|---|
| **BEN** | -26.371935 |
| **CO** | -1.853514 |
| **EBE** | -2.407173 |
| **MXY** | 7.499778 |
| **NMHC** | -25.948440 |
| **NO_2** | -0.054696 |
| **NOx** | 0.140469 |
| **OXY** | 4.161625 |
| **O_3** | -0.137308 |
| **PM10** | 0.132925 |
| **PXY** | 4.062823 |
| **SO_2** | -0.566902 |
| **TCH** | 15.972244 |
| **TOL** | -1.881657 |

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

Out[27]:
```
<matplotlib.collections.PathCollection at 0x2dee228e850>
```




# ACCURACY

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

Out[28]: 0.13796631568894835

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

Out[29]: 0.14563020243366387

# 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]: 0.13788369114445032

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

Out[33]: 0.14560850699579742

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]:
```
0.040880310514757046
```

# Accuracy(Lasso)

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

Out[36]:
```
0.041939499058839425
```

# Elastic Net regression

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([-4.59605114, -0.        , -0.        ,  3.36119425, -0.        ,
        0.04586469,  0.03105838,  1.57283069, -0.15599659,  0.13430945,
        1.68310687, -0.95294493,  0.        , -2.59604792])
```

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

Out[39]:
```
28079057.189837225
```

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

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

Out[41]:
```
0.09581264928877309
```

# 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)))
```

```
35.7801596720135
1494.585665856179
38.65987151887832
```

# Logistic Regression

In [43]:
```
from sklearn.linear_model import LogisticRegression
```

In [44]:
```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

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

Out[45]: (25631, 14)

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

Out[46]: (25631,)

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=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [51]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079099]
```

In [52]:
```
logr.classes_
```

Out[52]: array([28079006, 28079024, 28079099], dtype=int64)

In [53]:
```
logr.score(fs,target_vector)
```

Out[53]:
```
0.7942335453162186
```

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

Out[54]:
```
8.280697732723258e-09
```

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

Out[55]:
```
array([[8.28069773e-09, 1.19826346e-13, 9.99999992e-01]])
```

# 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="accuracy")
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]:
```
0.8554711657916186
```

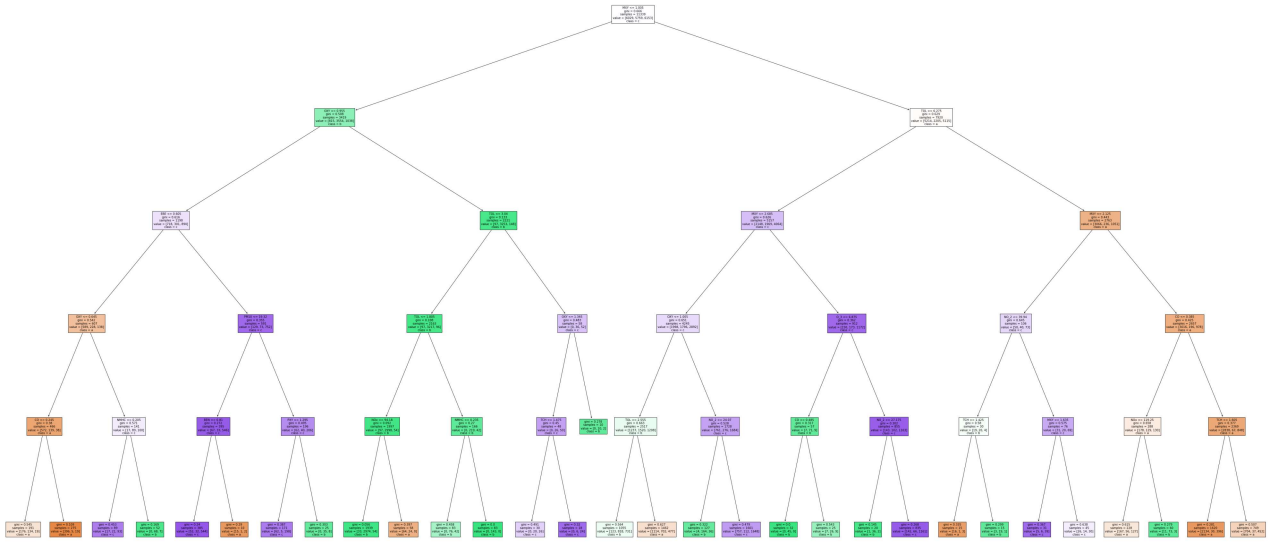
In [61]:
```
rfc_best=grid_search.best_estimator_
```

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','c','d']
```

Out[62]:
```
[Text(2222.7000000000003, 1993.2, 'MXY <= 1.005\ngini = 0.666\nsamples = 11339\nvalue =
[6029, 5759, 6153]\nclass = c'),
 Text(1171.8000000000002, 1630.8000000000002, 'OXY <= 0.955\ngini = 0.508\nsamples = 341
9\nvalue = [815, 3554, 1038]\nclass = b'),
 Text(595.2, 1268.4, 'EBE <= 0.605\ngini = 0.616\nsamples = 1198\nvalue = [718, 301, 89
0]\nclass = c'),
 Text(297.6, 906.0, 'OXY <= 0.645\ngini = 0.542\nsamples = 607\nvalue = [589, 228, 138]
\nclass = a'),
 Text(148.8, 543.5999999999999, 'CO <= 0.245\ngini = 0.38\nsamples = 466\nvalue = [572,
139, 38]\nclass = a'),
 Text(74.4, 181.19999999999982, 'gini = 0.545\nsamples = 191\nvalue = [176, 134, 19]\ncl
ass = a'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.109\nsamples = 275\nvalue = [39
6, 5, 19]\nclass = a'),
 Text(446.40000000000003, 543.5999999999999, 'NMHC <= 0.205\ngini = 0.571\nsamples = 141
\nvalue = [17, 89, 100]\nclass = c'),
 Text(372.0, 181.19999999999982, 'gini = 0.453\nsamples = 89\nvalue = [17, 21, 93]\nclas
s = c'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.169\nsamples = 52\nvalue = [0, 6
8, 7]\nclass = b'),
 Text(892.8000000000001, 906.0, 'PM10 <= 19.32\ngini = 0.355\nsamples = 591\nvalue = [12
9, 73, 752]\nclass = c'),
 Text(744.0, 543.5999999999999, 'BEN <= 0.81\ngini = 0.272\nsamples = 395\nvalue = [67,
33, 546]\nclass = c'),
 Text(669.6, 181.19999999999982, 'gini = 0.24\nsamples = 385\nvalue = [52, 32, 544]\ncla
ss = c'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.29\nsamples = 10\nvalue = [15, 1,
2]\nclass = a'),
 Text(1041.6000000000001, 543.5999999999999, 'PXY <= 1.295\ngini = 0.495\nsamples = 196
\nvalue = [62, 40, 206]\nclass = c'),
 Text(967.2, 181.19999999999982, 'gini = 0.387\nsamples = 171\nvalue = [62, 5, 198]\ncla
ss = c'),
 Text(1116.0, 181.19999999999982, 'gini = 0.303\nsamples = 25\nvalue = [0, 35, 8]\nclass
= b'),
 Text(1748.4, 1268.4, 'TOL <= 3.04\ngini = 0.133\nsamples = 2221\nvalue = [97, 3253, 14
8]\nclass = b'),
 Text(1488.0, 906.0, 'TOL <= 1.805\ngini = 0.108\nsamples = 2163\nvalue = [97, 3217, 96]
\nclass = b'),
 Text(1339.2, 543.5999999999999, 'NOx <= 94.18\ngini = 0.092\nsamples = 1997\nvalue = [9
7, 2998, 54]\nclass = b'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.056\nsamples = 1939\nvalue = [3
3, 2974, 54]\nclass = b'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.397\nsamples = 58\nvalue = [64,
24, 0]\nclass = a'),
 Text(1636.8000000000002, 543.5999999999999, 'NMHC <= 0.235\ngini = 0.27\nsamples = 166
\nvalue = [0, 219, 42]\nclass = b'),
 Text(1562.4, 181.19999999999982, 'gini = 0.458\nsamples = 83\nvalue = [0, 76, 42]\nclas
s = b'),
 Text(1711.2, 181.19999999999982, 'gini = 0.0\nsamples = 83\nvalue = [0, 143, 0]\nclass
= b'),
 Text(2008.8000000000002, 906.0, 'OXY <= 1.345\ngini = 0.483\nsamples = 58\nvalue = [0,
36, 52]\nclass = c'),
 Text(1934.4, 543.5999999999999, 'TCH <= 1.475\ngini = 0.45\nsamples = 48\nvalue = [0, 2
6, 50]\nclass = c'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.491\nsamples = 30\nvalue = [0, 2
0, 26]\nclass = c'),
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.32\nsamples = 18\nvalue = [0, 6,
24]\nclass = c'),
 Text(2083.2000000000003, 543.5999999999999, 'gini = 0.278\nsamples = 10\nvalue = [0, 1
0, 2]\nclass = b'),
 Text(3273.6000000000004, 1630.8000000000002, 'TOL <= 6.275\ngini = 0.629\nsamples = 792
0\nvalue = [5214, 2205, 5115]\nclass = a'),
 Text(2678.4, 1268.4, 'MXY <= 2.685\ngini = 0.626\nsamples = 5157\nvalue = [2148, 1969,
4064]\nclass = c'),
 Text(2380.8, 906.0, 'OXY <= 1.005\ngini = 0.651\nsamples = 4245\nvalue = [1998, 1796, 2
```

```
892]\nclass = c'),
 Text(2232.0, 543.5999999999999, 'TOL <= 2.555\ngini = 0.663\nsamples = 2517\nvalue = [1
237, 1520, 1208]\nclass = b'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.564\nsamples = 1055\nvalue = [12
3, 818, 731]\nclass = b'),
 Text(2306.4, 181.19999999999982, 'gini = 0.627\nsamples = 1462\nvalue = [1114, 702, 47
7]\nclass = a'),
 Text(2529.6000000000004, 543.5999999999999, 'NO_2 <= 24.07\ngini = 0.528\nsamples = 172
8\nvalue = [761, 276, 1684]\nclass = c'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.322\nsamples = 127\nvalue = [4,
164, 36]\nclass = b'),
 Text(2604.0, 181.19999999999982, 'gini = 0.479\nsamples = 1601\nvalue = [757, 112, 164
8]\nclass = c'),
 Text(2976.0, 906.0, 'O_3 <= 6.875\ngini = 0.362\nsamples = 912\nvalue = [150, 173, 117
2]\nclass = c'),
 Text(2827.2000000000003, 543.5999999999999, 'CO <= 0.485\ngini = 0.317\nsamples = 57\nv
alue = [7, 71, 9]\nclass = b'),
 Text(2752.8, 181.19999999999982, 'gini = 0.0\nsamples = 32\nvalue = [0, 45, 0]\nclass =
b'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.543\nsamples = 25\nvalue = [7, 2
6, 9]\nclass = b'),
 Text(3124.8, 543.5999999999999, 'NO_2 <= 27.175\ngini = 0.302\nsamples = 855\nvalue =
[143, 102, 1163]\nclass = c'),
 Text(3050.4, 181.19999999999982, 'gini = 0.145\nsamples = 20\nvalue = [1, 36, 2]\nclass
= b'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.268\nsamples = 835\nvalue = [14
2, 66, 1161]\nclass = c'),
 Text(3868.8, 1268.4, 'MXY <= 2.125\ngini = 0.443\nsamples = 2763\nvalue = [3066, 236, 1
051]\nclass = a'),
 Text(3571.2000000000003, 906.0, 'NO_2 <= 39.94\ngini = 0.645\nsamples = 106\nvalue = [5
0, 40, 73]\nclass = c'),
 Text(3422.4, 543.5999999999999, 'TCH <= 1.425\ngini = 0.58\nsamples = 30\nvalue = [19,
20, 4]\nclass = b'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.335\nsamples = 15\nvalue = [16,
1, 3]\nclass = a'),
 Text(3496.8, 181.19999999999982, 'gini = 0.299\nsamples = 15\nvalue = [3, 19, 1]\nclass
= b'),
 Text(3720.0000000000005, 543.5999999999999, 'MXY <= 1.635\ngini = 0.575\nsamples = 76\n
value = [31, 20, 69]\nclass = c'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.367\nsamples = 31\nvalue = [5,
6, 39]\nclass = c'),
 Text(3794.4, 181.19999999999982, 'gini = 0.638\nsamples = 45\nvalue = [26, 14, 30]\ncla
ss = c'),
 Text(4166.400000000001, 906.0, 'CO <= 0.385\ngini = 0.425\nsamples = 2657\nvalue = [301
6, 196, 978]\nclass = a'),
 Text(4017.6000000000004, 543.5999999999999, 'NOx <= 119.25\ngini = 0.658\nsamples = 288
\nvalue = [178, 129, 130]\nclass = a'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.615\nsamples = 228\nvalue = [16
7, 56, 127]\nclass = a'),
 Text(4092.0000000000005, 181.19999999999982, 'gini = 0.279\nsamples = 60\nvalue = [11,
73, 3]\nclass = b'),
 Text(4315.200000000001, 543.5999999999999, 'TCH <= 1.605\ngini = 0.377\nsamples = 2369
\nvalue = [2838, 67, 848]\nclass = a'),
 Text(4240.8, 181.19999999999982, 'gini = 0.281\nsamples = 1620\nvalue = [2134, 30, 396]
\nclass = a'),
 Text(4389.6, 181.19999999999982, 'gini = 0.507\nsamples = 749\nvalue = [704, 37, 452]\n
class = a')]
```



# Conclusion

## Accuracy

# linear regression

In [63]:
```
lr.score(x_test,y_test)
```

Out[63]: 0.13796631568894835

# Ridge regression

In [64]:
```
rr.score(x_test,y_test)
```

Out[64]: 0.13788369114445032

# Lasso regression

In [65]:
```
la.score(x_test,y_test)
```

Out[65]: 0.041939499058839425

# Elastic net regression

In [66]:
```
en.score(x_test,y_test)
```

Out[66]: 0.09581264928877309

# Logistic regression

In [67]:
```
logr.score(fs,target_vector)
```

Out[67]: 0.7942335453162186

# Random forest

In [68]:
```
grid_search.best_score_
```

Out[68]: 0.8554711657916186

## Accuracy for random forest is higher so it is the best fit model