# 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_2007.csv")
df
```

Out[2]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2007-12-01 01:00:00 | NaN | 2.86 | NaN | NaN | NaN | 282.200012 | 1054.000000 | NaN | 4.030000 | 156.1 |
| **1** | 2007-12-01 01:00:00 | NaN | 1.82 | NaN | NaN | NaN | 86.419998 | 354.600006 | NaN | 3.260000 | 80.8 |
| **2** | 2007-12-01 01:00:00 | NaN | 1.47 | NaN | NaN | NaN | 94.639999 | 319.000000 | NaN | 5.310000 | 53.0 |
| **3** | 2007-12-01 01:00:00 | NaN | 1.64 | NaN | NaN | NaN | 127.900002 | 476.700012 | NaN | 4.500000 | 105.3 |
| **4** | 2007-12-01 01:00:00 | 4.64 | 1.86 | 4.26 | 7.98 | 0.57 | 145.100006 | 573.900024 | 3.49 | 52.689999 | 106.5 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **225115** | 2007-03-01 00:00:00 | 0.30 | 0.45 | 1.00 | 0.30 | 0.26 | 8.690000 | 11.690000 | 1.00 | 42.209999 | 6.7 |
| **225116** | 2007-03-01 00:00:00 | NaN | 0.16 | NaN | NaN | NaN | 46.820000 | 51.480000 | NaN | 22.150000 | 5.7 |
| **225117** | 2007-03-01 00:00:00 | 0.24 | NaN | 0.20 | NaN | 0.09 | 51.259998 | 66.809998 | NaN | 18.540001 | 13.0 |
| **225118** | 2007-03-01 00:00:00 | 0.11 | NaN | 1.00 | NaN | 0.05 | 24.240000 | 36.930000 | NaN | NaN | 6.6 |
| **225119** | 2007-03-01 00:00:00 | 0.53 | 0.40 | 1.00 | 1.70 | 0.12 | 32.360001 | 47.860001 | 1.37 | 24.150000 | 10.2 |

225120 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: 25443 entries, 4 to 225119
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     25443 non-null  object 
 1   BEN      25443 non-null  float64
 2   CO       25443 non-null  float64
 3   EBE      25443 non-null  float64
 4   MXY      25443 non-null  float64
 5   NMHC     25443 non-null  float64
 6   NO_2     25443 non-null  float64
 7   NOx      25443 non-null  float64
 8   OXY      25443 non-null  float64
 9   O_3      25443 non-null  float64
 10  PM10     25443 non-null  float64
 11  PM25     25443 non-null  float64
 12  PXY      25443 non-null  float64
 13  SO_2     25443 non-null  float64
 14  TCH      25443 non-null  float64
 15  TOL      25443 non-null  float64
 16  station  25443 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.5+ MB
```

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

Out[6]:

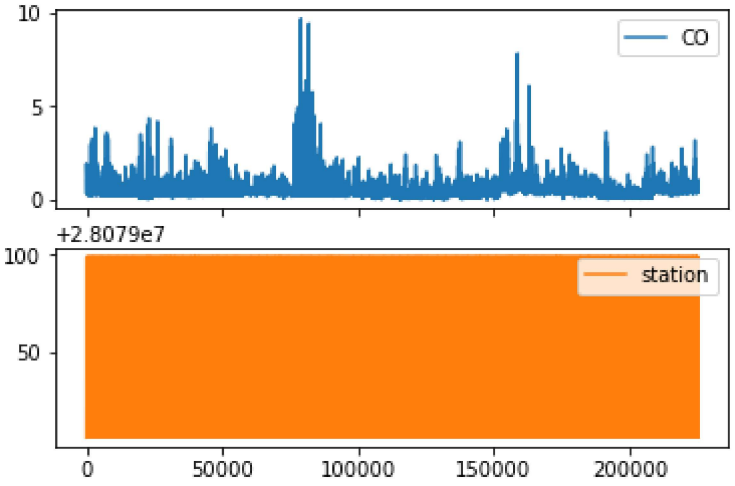
| | CO | station |
|---|---|---|
| **4** | 1.86 | 28079006 |
| **21** | 0.31 | 28079024 |
| **25** | 1.42 | 28079099 |
| **30** | 1.89 | 28079006 |
| **47** | 0.30 | 28079024 |
| **...** | ... | ... |
| **225073** | 0.47 | 28079006 |
| **225094** | 0.45 | 28079099 |
| **225098** | 0.41 | 28079006 |
| **225115** | 0.45 | 28079024 |
| **225119** | 0.40 | 28079099 |

25443 rows × 2 columns

# Line chart

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

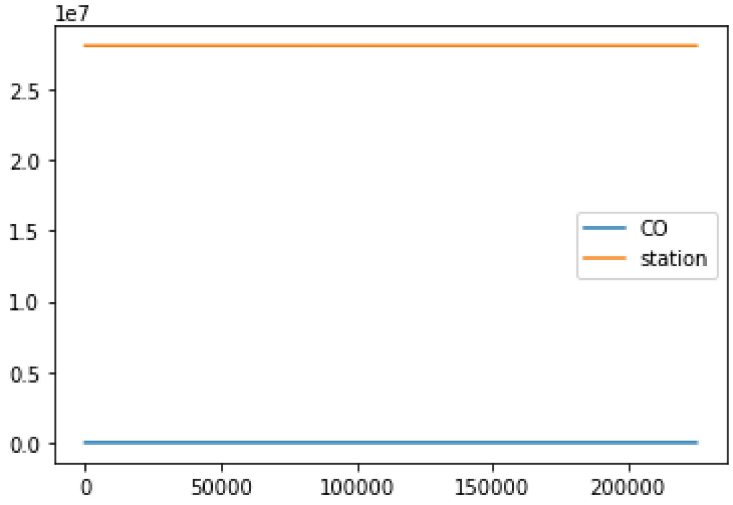
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```




# Line chart

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

Out[8]: <AxesSubplot:>

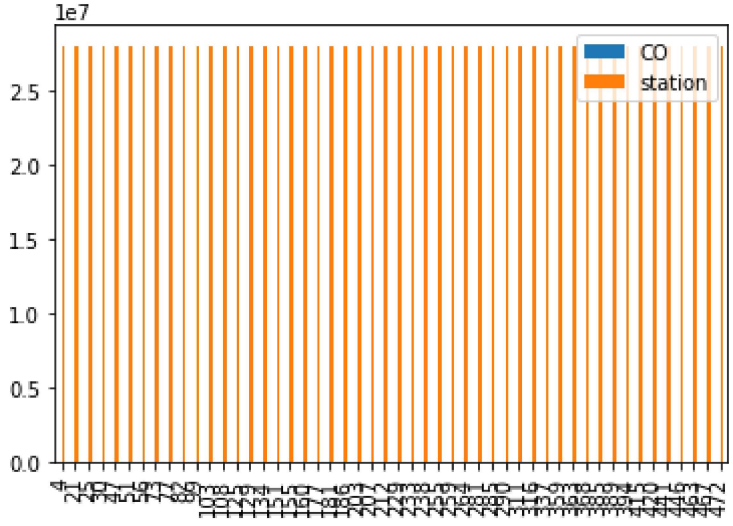



# Bar chart

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

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

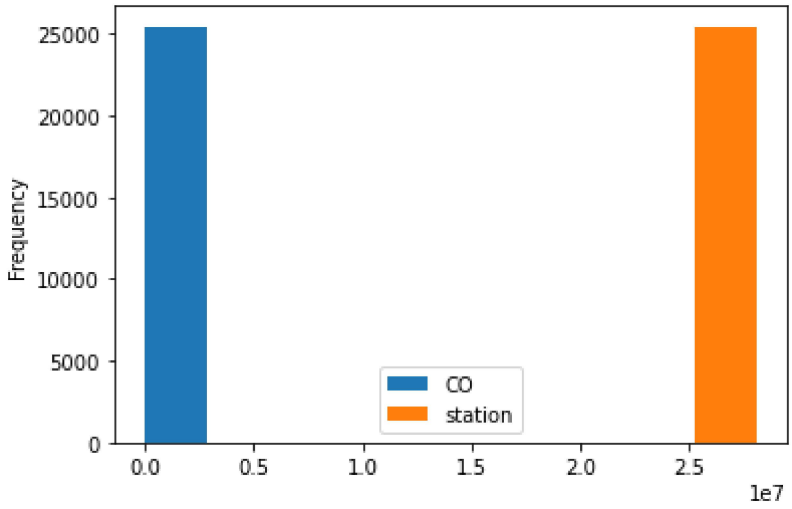
Out[10]: <AxesSubplot:>




# Histogram

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

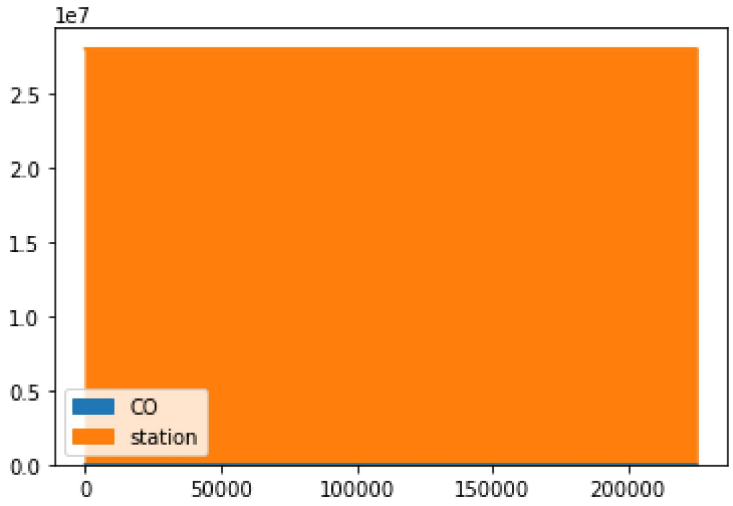
Out[11]: <AxesSubplot:ylabel='Frequency'>




# Area chart

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

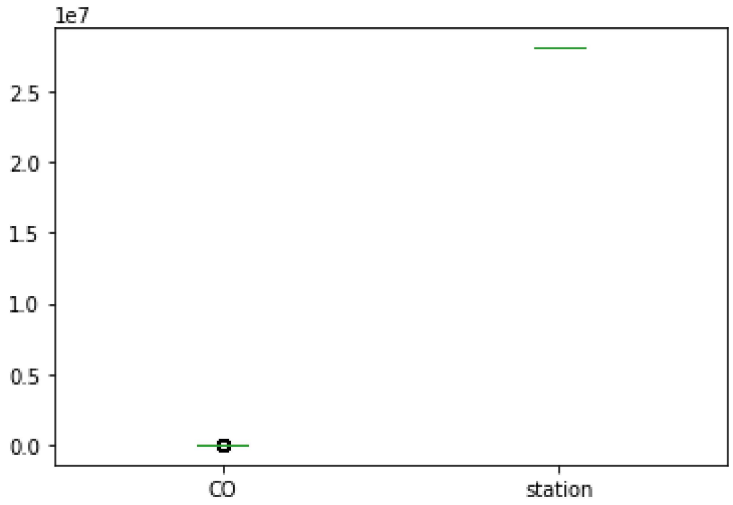
Out[12]: <AxesSubplot:>




# Box chart

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

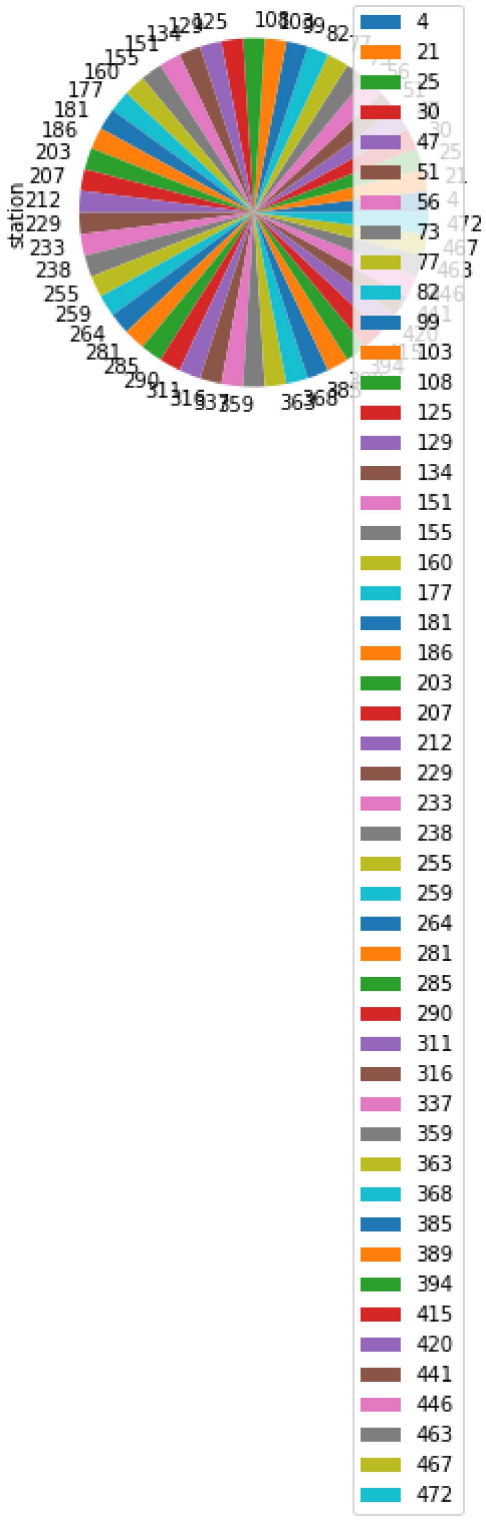
Out[13]:
```
<AxesSubplot:>
```




# Pie chart

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

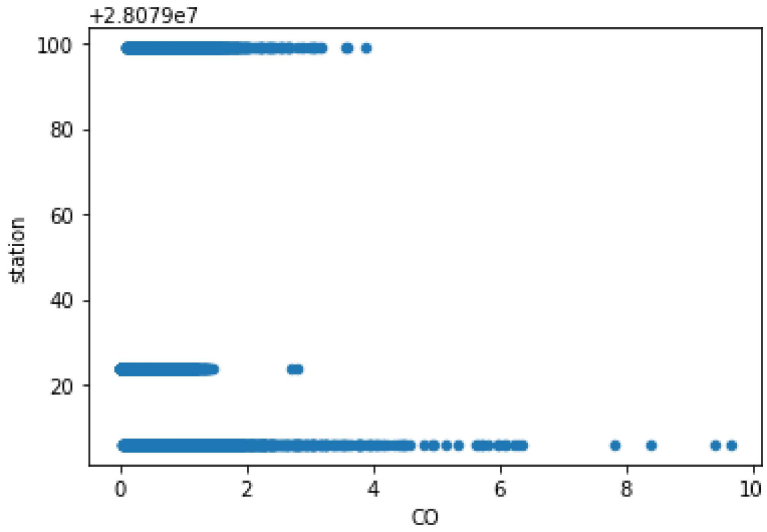
Out[14]: <AxesSubplot:ylabel='station'>




# 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: 25443 entries, 4 to 225119
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     25443 non-null  object
 1   BEN      25443 non-null  float64
 2   CO       25443 non-null  float64
 3   EBE      25443 non-null  float64
 4   MXY      25443 non-null  float64
 5   NMHC     25443 non-null  float64
 6   NO_2     25443 non-null  float64
 7   NOx      25443 non-null  float64
 8   OXY      25443 non-null  float64
 9   O_3      25443 non-null  float64
 10  PM10     25443 non-null  float64
 11  PM25     25443 non-null  float64
 12  PXY      25443 non-null  float64
 13  SO_2     25443 non-null  float64
```

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

Out[17]:

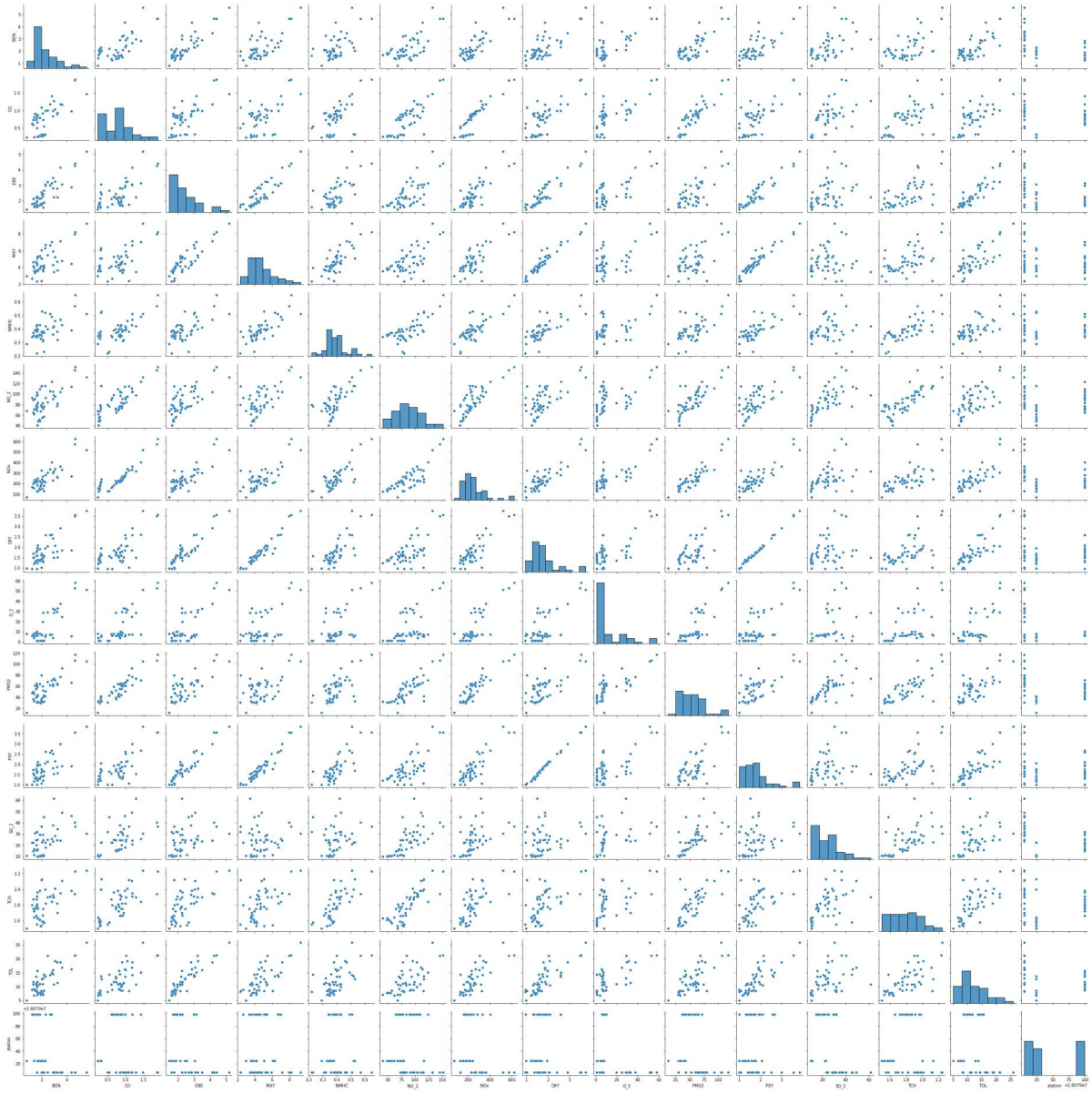
| | BEN | CO | EBE | MXY | NMHC | NO_2 | |
|---|---|---|---|---|---|---|---|
| **count** | 25443.000000 | 25443.000000 | 25443.000000 | 25443.000000 | 25443.000000 | 25443.000000 | 254 |
| **mean** | 1.146744 | 0.505120 | 1.394071 | 2.392008 | 0.249967 | 58.532683 | 1 |
| **std** | 1.278733 | 0.423231 | 1.268265 | 2.784302 | 0.142627 | 37.755029 | 1 |
| **min** | 0.130000 | 0.000000 | 0.120000 | 0.150000 | 0.000000 | 1.690000 | |
| **25%** | 0.450000 | 0.260000 | 0.780000 | 0.960000 | 0.160000 | 31.285001 | |
| **50%** | 0.770000 | 0.400000 | 1.000000 | 1.500000 | 0.220000 | 54.080002 | |
| **75%** | 1.390000 | 0.640000 | 1.580000 | 2.855000 | 0.300000 | 79.230003 | 1 |
| **max** | 30.139999 | 9.660000 | 31.680000 | 65.480003 | 2.570000 | 430.299988 | 18 |

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])
```

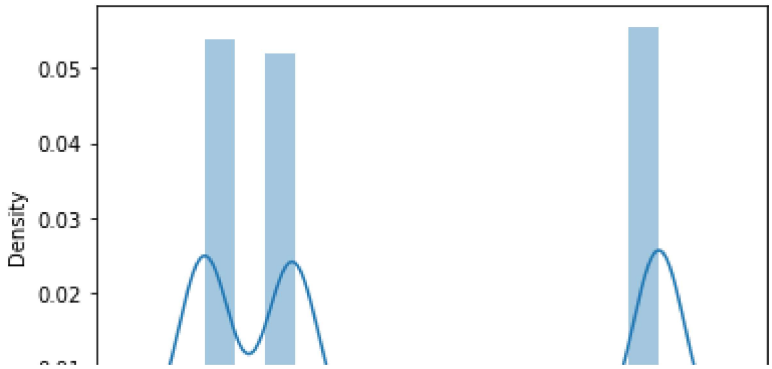
Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x25ef80ee4c0>
```



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

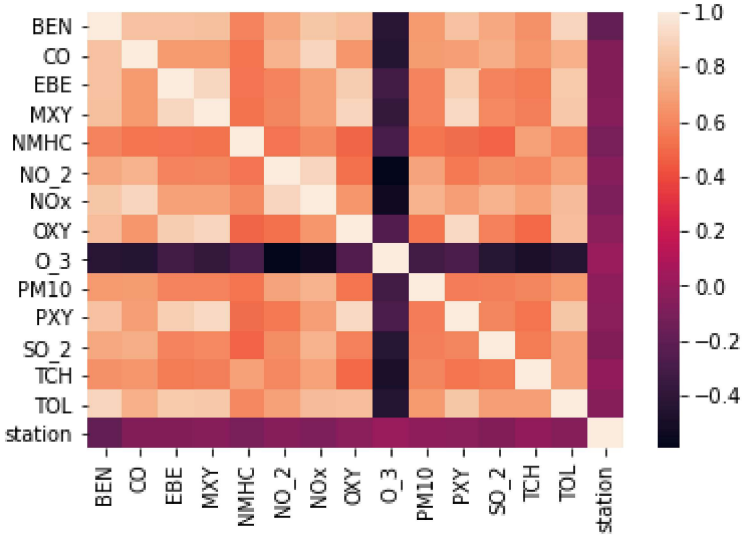
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
n 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]:
```
28079010.600033034
```

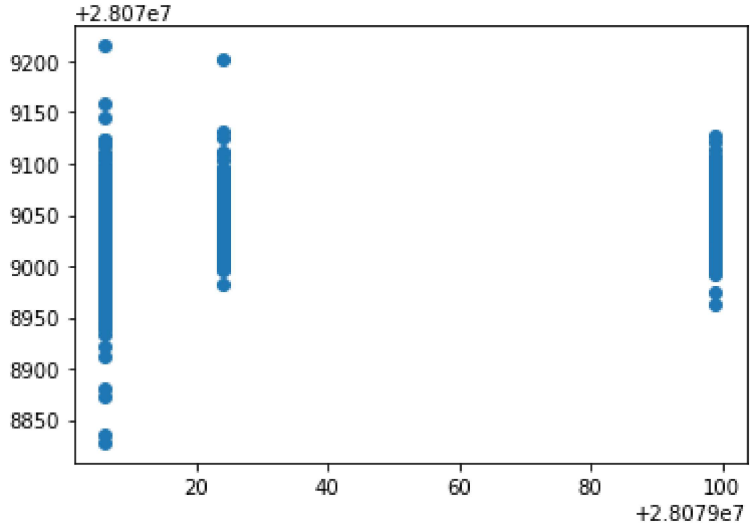
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

| | Co-efficient |
|---|---|
| **BEN** | -32.558438 |
| **CO** | 18.053937 |
| **EBE** | 0.165435 |
| **MXY** | -1.570164 |
| **NMHC** | -42.576573 |
| **NO_2** | 0.107104 |
| **NOx** | -0.049394 |
| **OXY** | 5.744534 |
| **O_3** | -0.047587 |
| **PM10** | 0.124554 |
| **PXY** | 7.117165 |
| **SO_2** | 0.225255 |
| **TCH** | 26.487376 |
| **TOL** | 3.431749 |

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

Out[27]: <matplotlib.collections.PathCollection at 0x25e87df0610>




# ACCURACY

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

Out[28]: 0.15928311043006327

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

Out[29]: 0.15812920950369025

# 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]: 0.15943434087340758

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

Out[33]:
```
0.15807304053939086
```

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]:
```
0.012186222056843055
```

# Accuracy(Lasso)

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

Out[36]:
```
0.013393235291537131
```

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([-7.99777272,  0.        , -0.        ,  0.        , -0.        ,
        0.03110117, -0.04998069,  0.69061266, -0.07494725,  0.1456773 ,
        0.79039579, -0.        ,  0.        ,  1.11616671])
```

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

Out[39]:
```
28079047.855154447
```

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

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

Out[41]:
```
0.06850581739933936
```

# 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)))
```

```
36.57406235306081
1520.3386583282227
38.99152033876369
```

# 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
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

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

Out[45]:
```
(25443, 14)
```

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

Out[46]:
```
(25443,)
```

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]:
```
0.8146838030106512
```

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

Out[54]:
```
1.082753977181323e-19
```

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

Out[55]:
```
array([[1.08275398e-19, 1.80383815e-19, 1.00000000e+00]])
```

# 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="ac
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]:
```
0.814654688377316
```

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

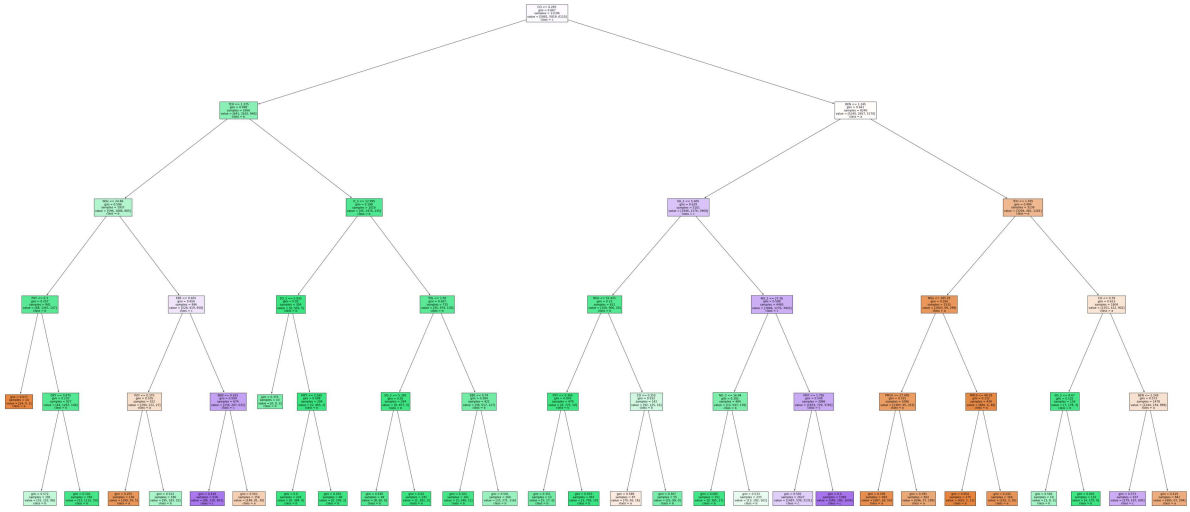
In [62]:
```
from sklearn.tree import plot_tree

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

Out[62]:
```
[Text(2055.7894736842104, 1993.2, 'CO <= 0.265\ngini = 0.667\nsamples = 11196
\nvalue = [5881, 5819, 6110]\nclass = c'),
 Text(900.6315789473683, 1630.8000000000002, 'TCH <= 1.375\ngini = 0.498\nsam
ples = 2956\nvalue = [641, 3162, 940]\nclass = b'),
 Text(430.7368421052631, 1268.4, 'NOx <= 24.86\ngini = 0.596\nsamples = 1937
\nvalue = [596, 1686, 805]\nclass = b'),
 Text(156.6315789473684, 906.0, 'PXY <= 0.3\ngini = 0.257\nsamples = 941\nval
ue = [68, 1267, 147]\nclass = b'),
 Text(78.3157894736842, 543.5999999999999, 'gini = 0.077\nsamples = 14\nvalue
= [24, 0, 1]\nclass = a'),
 Text(234.9473684210526, 543.5999999999999, 'OXY <= 0.875\ngini = 0.233\nsamp
les = 927\nvalue = [44, 1267, 146]\nclass = b'),
 Text(156.6315789473684, 181.19999999999982, 'gini = 0.572\nsamples = 181\nva
lue = [31, 152, 96]\nclass = b'),
 Text(313.2631578947368, 181.19999999999982, 'gini = 0.102\nsamples = 746\nva
lue = [13, 1115, 50]\nclass = b'),
 Text(704.8421052631578, 906.0, 'EBE <= 0.655\ngini = 0.656\nsamples = 996\nv
alue = [528, 419, 658]\nclass = c'),
 Text(548.2105263157895, 543.5999999999999, 'PXY <= 0.375\ngini = 0.535\nsamp
les = 322\nvalue = [294, 212, 27]\nclass = a'),
 Text(469.8947368421052, 181.19999999999982, 'gini = 0.255\nsamples = 136\nva
lue = [199, 29, 5]\nclass = a'),
 Text(626.5263157894736, 181.19999999999982, 'gini = 0.522\nsamples = 186\nva
lue = [95, 183, 22]\nclass = b'),
 Text(861.4736842105262, 543.5999999999999, 'BEN <= 0.625\ngini = 0.569\nsamp
les = 674\nvalue = [234, 207, 631]\nclass = c'),
 Text(783.1578947368421, 181.19999999999982, 'gini = 0.418\nsamples = 518\nva
lue = [86, 126, 601]\nclass = c'),
 Text(939.7894736842104, 181.19999999999982, 'gini = 0.562\nsamples = 156\nva
lue = [148, 81, 30]\nclass = a'),
 Text(1370.5263157894735, 1268.4, 'O_3 <= 12.995\ngini = 0.198\nsamples = 101
9\nvalue = [45, 1476, 135]\nclass = b'),
 Text(1096.421052631579, 906.0, 'SO_2 <= 5.935\ngini = 0.02\nsamples = 304\nv
alue = [0, 502, 5]\nclass = b'),
 Text(1018.1052631578947, 543.5999999999999, 'gini = 0.375\nsamples = 10\nval
ue = [0, 9, 3]\nclass = b'),
 Text(1174.7368421052631, 543.5999999999999, 'MXY <= 2.565\ngini = 0.008\nsam
ples = 294\nvalue = [0, 493, 2]\nclass = b'),
 Text(1096.421052631579, 181.19999999999982, 'gini = 0.0\nsamples = 226\nvalu
e = [0, 384, 0]\nclass = b'),
 Text(1253.0526315789473, 181.19999999999982, 'gini = 0.035\nsamples = 68\nva
lue = [0, 109, 2]\nclass = b'),
 Text(1644.6315789473683, 906.0, 'TOL <= 1.82\ngini = 0.267\nsamples = 715\nv
alue = [45, 974, 130]\nclass = b'),
 Text(1488.0, 543.5999999999999, 'SO_2 <= 5.305\ngini = 0.05\nsamples = 294\n
value = [9, 457, 3]\nclass = b'),
 Text(1409.6842105263156, 181.19999999999982, 'gini = 0.195\nsamples = 49\nva
lue = [8, 65, 0]\nclass = b'),
 Text(1566.3157894736842, 181.19999999999982, 'gini = 0.02\nsamples = 245\nva
lue = [1, 392, 3]\nclass = b'),
 Text(1801.2631578947367, 543.5999999999999, 'EBE <= 0.74\ngini = 0.384\nsamp
les = 421\nvalue = [36, 517, 127]\nclass = b'),
 Text(1722.9473684210525, 181.19999999999982, 'gini = 0.103\nsamples = 161\nv
alue = [3, 246, 11]\nclass = b'),
 Text(1879.5789473684208, 181.19999999999982, 'gini = 0.501\nsamples = 260\nv
alue = [33, 271, 116]\nclass = b'),
 Text(3210.9473684210525, 1630.8000000000002, 'BEN <= 1.245\ngini = 0.641\nsa
```

```
mples = 8240\nvalue = [5240, 2657, 5170]\nclass = a'),
 Text(2584.4210526315787, 1268.4, 'SO_2 <= 5.605\ngini = 0.629\nsamples = 510
1\nvalue = [1946, 2176, 3969]\nclass = c'),
 Text(2271.157894736842, 906.0, 'NOx <= 52.425\ngini = 0.22\nsamples = 621\nv
alue = [100, 900, 26]\nclass = b'),
 Text(2114.5263157894738, 543.5999999999999, 'PXY <= 0.365\ngini = 0.045\nsam
ples = 479\nvalue = [8, 775, 10]\nclass = b'),
 Text(2036.2105263157894, 181.19999999999982, 'gini = 0.351\nsamples = 12\nva
lue = [5, 17, 0]\nclass = b'),
 Text(2192.842105263158, 181.19999999999982, 'gini = 0.033\nsamples = 467\nva
lue = [3, 758, 10]\nclass = b'),
 Text(2427.7894736842104, 543.5999999999999, 'CO <= 0.355\ngini = 0.552\nsamp
les = 142\nvalue = [92, 125, 16]\nclass = b'),
 Text(2349.4736842105262, 181.19999999999982, 'gini = 0.589\nsamples = 87\nva
lue = [70, 56, 16]\nclass = a'),
 Text(2506.1052631578946, 181.19999999999982, 'gini = 0.367\nsamples = 55\nva
lue = [22, 69, 0]\nclass = b'),
 Text(2897.6842105263154, 906.0, 'NO_2 <= 27.36\ngini = 0.588\nsamples = 4480
\nvalue = [1846, 1276, 3943]\nclass = c'),
 Text(2741.052631578947, 543.5999999999999, 'NO_2 <= 16.94\ngini = 0.392\nsam
ples = 484\nvalue = [13, 547, 178]\nclass = b'),
 Text(2662.736842105263, 181.19999999999982, 'gini = 0.085\nsamples = 251\nva
lue = [0, 365, 17]\nclass = b'),
 Text(2819.368421052631, 181.19999999999982, 'gini = 0.533\nsamples = 233\nva
lue = [13, 182, 161]\nclass = b'),
 Text(3054.315789473684, 543.5999999999999, 'MXY <= 1.795\ngini = 0.549\nsamp
les = 3996\nvalue = [1833, 729, 3765]\nclass = c'),
 Text(2976.0, 181.19999999999982, 'gini = 0.592\nsamples = 2627\nvalue = [148
7, 539, 2131]\nclass = c'),
 Text(3132.6315789473683, 181.19999999999982, 'gini = 0.4\nsamples = 1369\nva
lue = [346, 190, 1634]\nclass = c'),
 Text(3837.4736842105262, 1268.4, 'TCH <= 1.505\ngini = 0.494\nsamples = 3139
\nvalue = [3294, 481, 1201]\nclass = a'),
 Text(3524.210526315789, 906.0, 'NOx <= 185.25\ngini = 0.266\nsamples = 1535
\nvalue = [2043, 69, 299]\nclass = a'),
 Text(3367.578947368421, 543.5999999999999, 'PM10 <= 27.495\ngini = 0.315\nsa
mples = 1096\nvalue = [1383, 65, 253]\nclass = a'),
 Text(3289.2631578947367, 181.19999999999982, 'gini = 0.196\nsamples = 494\nv
alue = [687, 28, 54]\nclass = a'),
 Text(3445.894736842105, 181.19999999999982, 'gini = 0.395\nsamples = 602\nva
lue = [696, 37, 199]\nclass = a'),
 Text(3680.8421052631575, 543.5999999999999, 'PM10 <= 48.19\ngini = 0.132\nsa
mples = 439\nvalue = [660, 4, 46]\nclass = a'),
 Text(3602.5263157894733, 181.19999999999982, 'gini = 0.054\nsamples = 275\nv
alue = [425, 1, 11]\nclass = a'),
 Text(3759.1578947368416, 181.19999999999982, 'gini = 0.242\nsamples = 164\nv
alue = [235, 3, 35]\nclass = a'),
 Text(4150.736842105262, 906.0, 'CO <= 0.39\ngini = 0.613\nsamples = 1604\nva
lue = [1251, 412, 902]\nclass = a'),
 Text(3994.1052631578946, 543.5999999999999, 'SO_2 <= 8.67\ngini = 0.102\nsam
ples = 126\nvalue = [7, 178, 3]\nclass = b'),
 Text(3915.7894736842104, 181.19999999999982, 'gini = 0.582\nsamples = 10\nva
lue = [3, 8, 3]\nclass = b'),
 Text(4072.4210526315787, 181.19999999999982, 'gini = 0.045\nsamples = 116\nv
alue = [4, 170, 0]\nclass = b'),
 Text(4307.368421052632, 543.5999999999999, 'BEN <= 2.245\ngini = 0.573\nsamp
les = 1478\nvalue = [1244, 234, 899]\nclass = a'),
```

```
 Text(4229.0526315789475, 181.19999999999982, 'gini = 0.573\nsamples = 637\nv
alue = [279, 167, 605]\nclass = c'),
 Text(4385.684210526316, 181.19999999999982, 'gini = 0.419\nsamples = 841\nva
lue = [965, 67, 294]\nclass = a')]
```



# Conclusion

## Accuracy

In [63]:
```
lr.score(x_train,y_train)
```

Out[63]: 0.15812920950369025

In [64]:
```
rr.score(x_train,y_train)
```

Out[64]: 0.15807304053939086

In [65]:
```
la.score(x_train,y_train)
```

Out[65]: 0.012186222056843055

In [67]:
```
en.score(x_test,y_test)
```

Out[67]: 0.06850581739933936

In [68]:
```
logr.score(fs,target_vector)
```

Out[68]: 0.8146838030106512

In [69]:
```
grid_search.best_score_
```

Out[69]: 0.814654688377316

# Logistic Regression is suitable for this dataset