# 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_2002.csv")
df
```

Out[2]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2002-04-01 01:00:00 | NaN | 1.39 | NaN | NaN | NaN | 145.100006 | 352.100006 | NaN | 6.54 | 41.990002 |
| **1** | 2002-04-01 01:00:00 | 1.93 | 0.71 | 2.33 | 6.20 | 0.15 | 98.150002 | 153.399994 | 2.67 | 6.85 | 20.980000 |
| **2** | 2002-04-01 01:00:00 | NaN | 0.80 | NaN | NaN | NaN | 103.699997 | 134.000000 | NaN | 13.01 | 28.440001 |
| **3** | 2002-04-01 01:00:00 | NaN | 1.61 | NaN | NaN | NaN | 97.599998 | 268.000000 | NaN | 5.12 | 42.180000 |
| **4** | 2002-04-01 01:00:00 | NaN | 1.90 | NaN | NaN | NaN | 92.089996 | 237.199997 | NaN | 7.28 | 76.330002 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **217291** | 2002-11-01 00:00:00 | 4.16 | 1.14 | NaN | NaN | NaN | 81.080002 | 265.700012 | NaN | 7.21 | 36.750000 |
| **217292** | 2002-11-01 00:00:00 | 3.67 | 1.73 | 2.89 | NaN | 0.38 | 113.900002 | 373.100006 | NaN | 5.66 | 63.389999 |
| **217293** | 2002-11-01 00:00:00 | 1.37 | 0.58 | 1.17 | 2.37 | 0.15 | 65.389999 | 107.699997 | 1.30 | 9.11 | 9.640000 |
| **217294** | 2002-11-01 00:00:00 | 4.51 | 0.91 | 4.83 | 10.99 | NaN | 149.800003 | 202.199997 | 1.00 | 5.75 | NaN |
| **217295** | 2002-11-01 00:00:00 | 3.11 | 1.17 | 3.00 | 7.77 | 0.26 | 80.110001 | 180.300003 | 2.25 | 7.38 | 29.240000 |

217296 rows × 16 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', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32381 entries, 1 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     32381 non-null  object 
 1   BEN      32381 non-null  float64
 2   CO       32381 non-null  float64
 3   EBE      32381 non-null  float64
 4   MXY      32381 non-null  float64
 5   NMHC     32381 non-null  float64
 6   NO_2     32381 non-null  float64
 7   NOx      32381 non-null  float64
 8   OXY      32381 non-null  float64
 9   O_3      32381 non-null  float64
 10  PM10     32381 non-null  float64
 11  PXY      32381 non-null  float64
 12  SO_2     32381 non-null  float64
 13  TCH      32381 non-null  float64
 14  TOL      32381 non-null  float64
 15  station  32381 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 4.2+ MB
```

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

Out[6]:

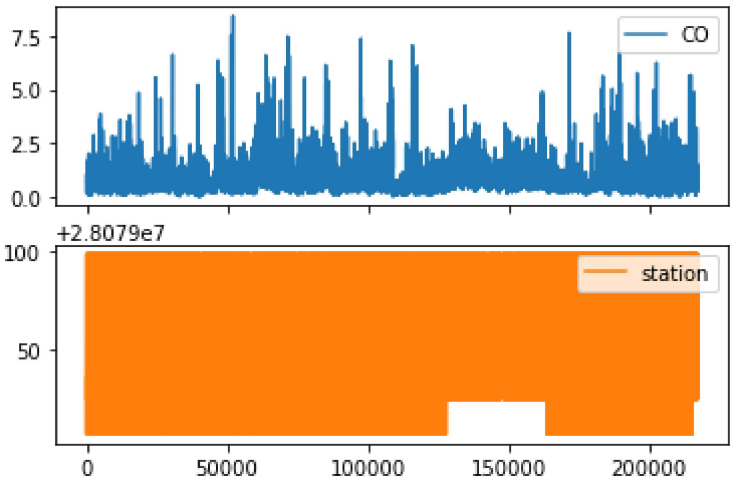
| | CO | station |
|---|---|---|
| **1** | 0.71 | 28079035 |
| **5** | 0.72 | 28079006 |
| **22** | 0.80 | 28079024 |
| **24** | 1.04 | 28079099 |
| **26** | 0.53 | 28079035 |
| **...** | ... | ... |
| **217269** | 0.28 | 28079024 |
| **217271** | 1.30 | 28079099 |
| **217273** | 0.97 | 28079035 |
| **217293** | 0.58 | 28079024 |
| **217295** | 1.17 | 28079099 |

32381 rows × 2 columns

# Line chart

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

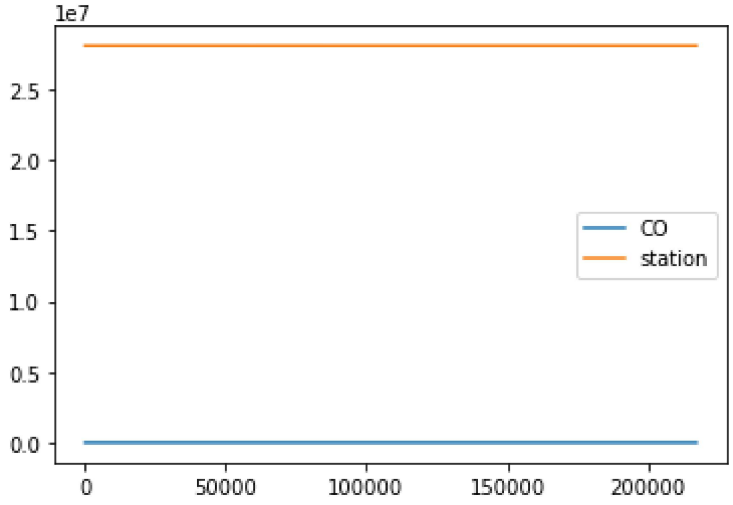
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```




# Line chart

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

Out[8]: <AxesSubplot:>

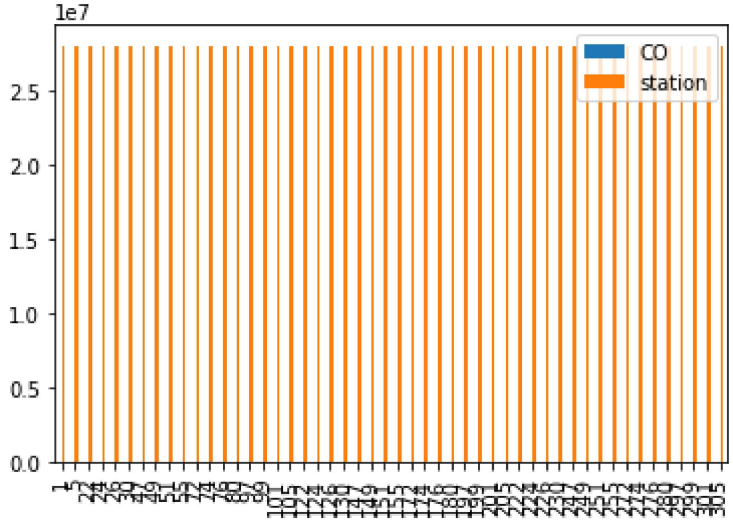



# Bar chart

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

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

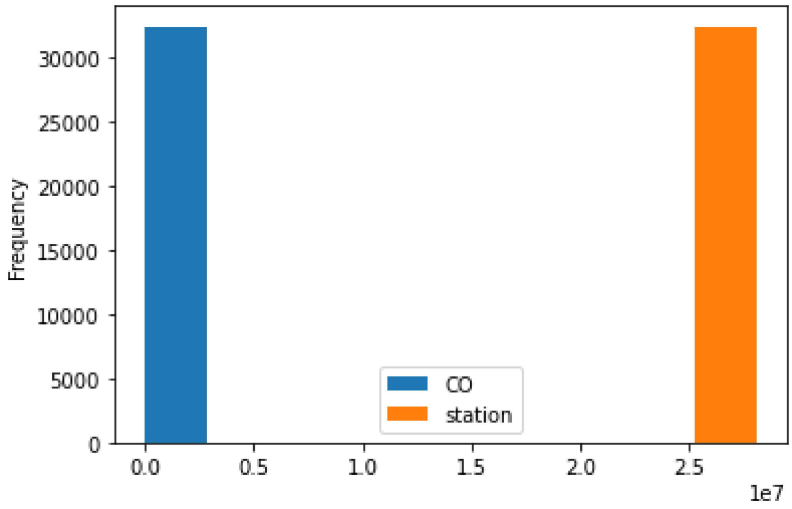
Out[10]: <AxesSubplot:>




# Histogram

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

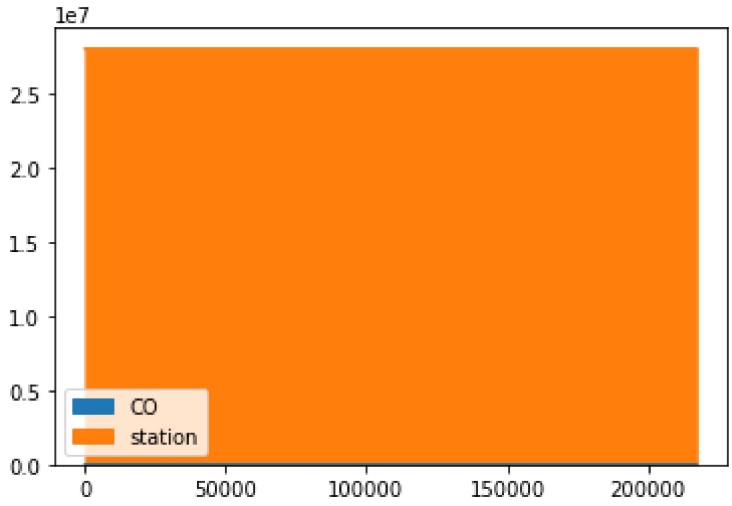
Out[11]:
```
<AxesSubplot:ylabel='Frequency'>
```




# Area chart

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

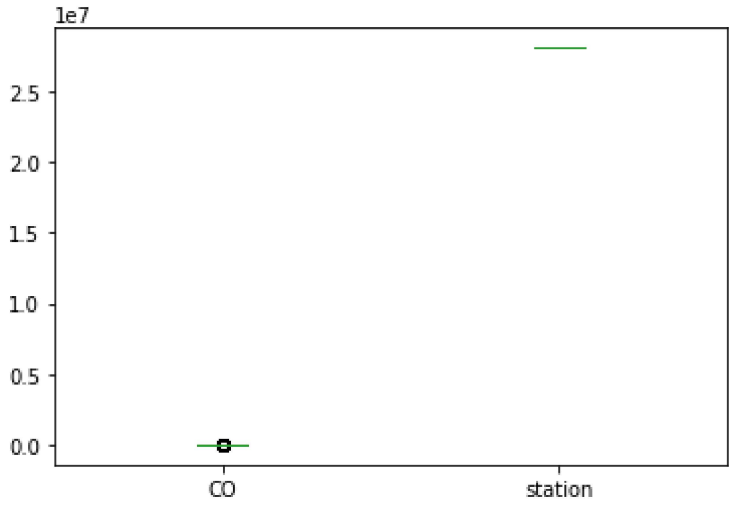
Out[12]:
```
<AxesSubplot:>
```




# Box chart

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

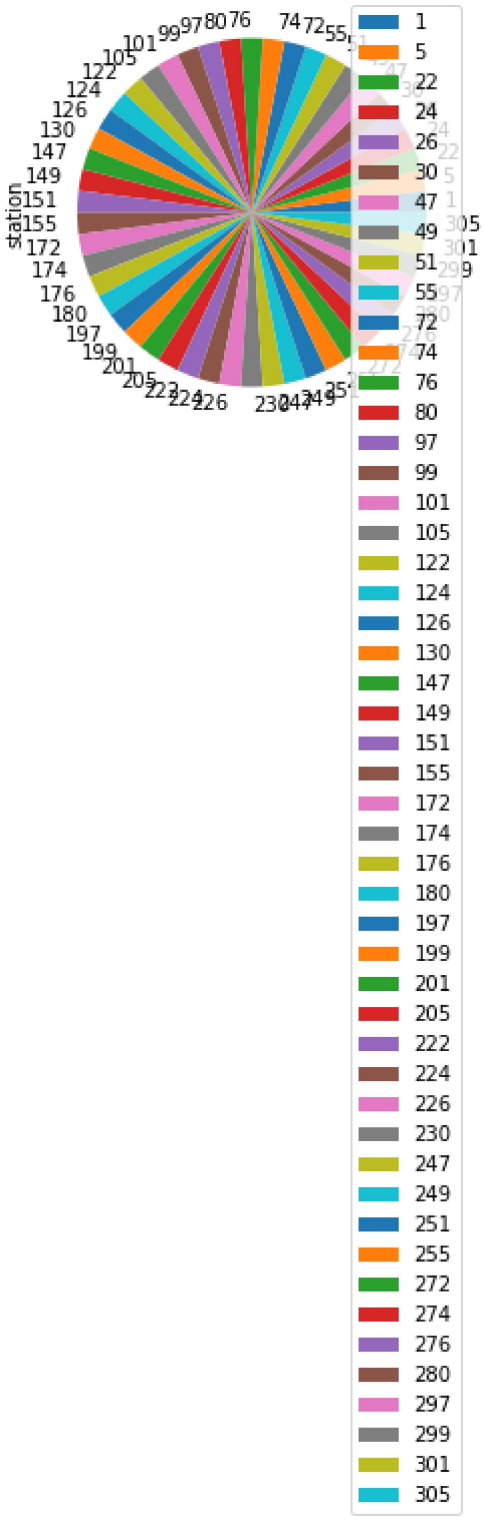
Out[13]: <AxesSubplot:>




# Pie chart

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

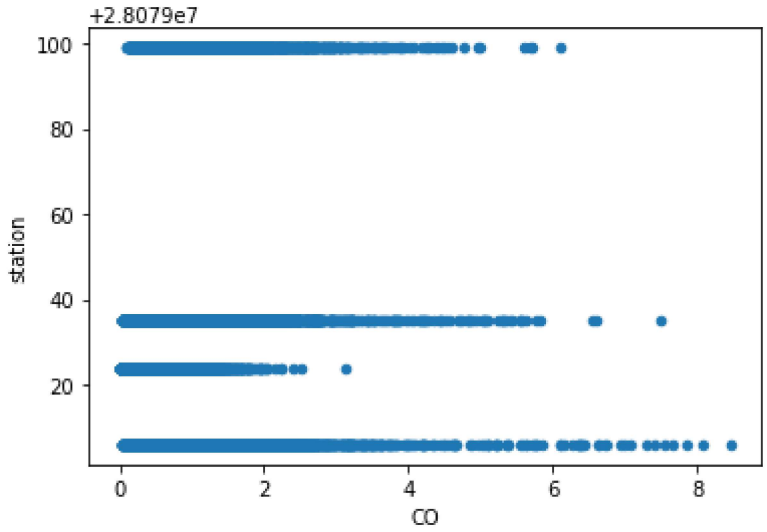
Out[14]: <AxesSubplot:ylabel='station'>




# 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: 32381 entries, 1 to 217295
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     32381 non-null  object 
 1   BEN      32381 non-null  float64
 2   CO       32381 non-null  float64
 3   EBE      32381 non-null  float64
 4   MXY      32381 non-null  float64
 5   NMHC     32381 non-null  float64
 6   NO_2     32381 non-null  float64
 7   NOx      32381 non-null  float64
 8   OXY      32381 non-null  float64
 9   O_3      32381 non-null  float64
 10  PM10     32381 non-null  float64
 11  PXY      32381 non-null  float64
 12  SO_2     32381 non-null  float64
 13  TCH      32381 non-null  float64
```

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

Out[17]:

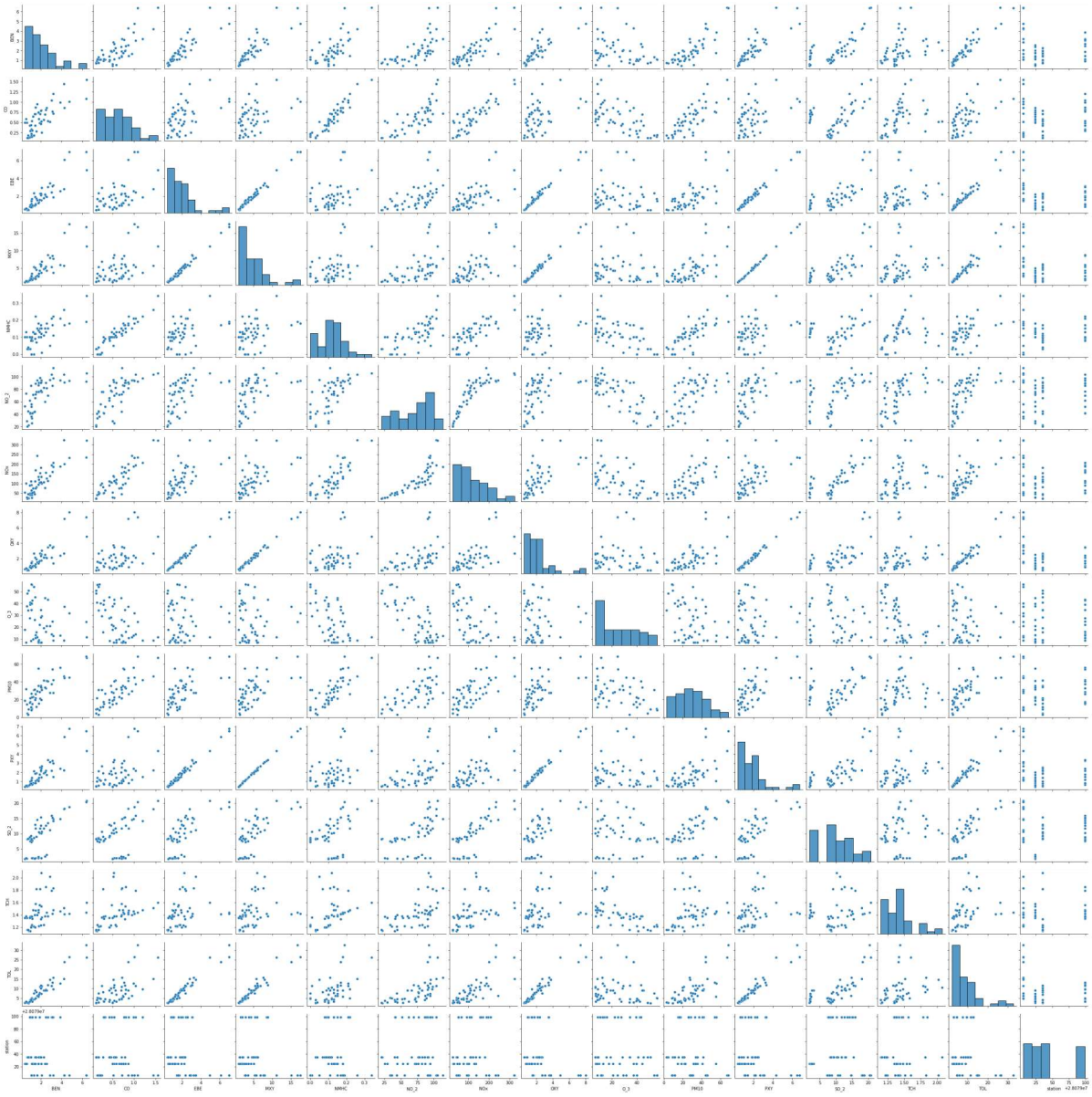
| | BEN | CO | EBE | MXY | NMHC | NO_2 | |
|---|---|---|---|---|---|---|---|
| **count** | 32381.000000 | 32381.000000 | 32381.000000 | 32381.000000 | 32381.000000 | 32381.000000 | 323 |
| **mean** | 2.479155 | 0.787323 | 2.914004 | 7.013636 | 0.155827 | 58.936796 | 1 |
| **std** | 2.280959 | 0.610810 | 2.667881 | 6.774365 | 0.135731 | 31.472733 | 1 |
| **min** | 0.180000 | 0.000000 | 0.180000 | 0.190000 | 0.000000 | 0.890000 | |
| **25%** | 0.970000 | 0.420000 | 1.140000 | 2.420000 | 0.080000 | 35.660000 | |
| **50%** | 1.840000 | 0.620000 | 2.130000 | 5.140000 | 0.130000 | 57.160000 | |
| **75%** | 3.250000 | 0.980000 | 3.830000 | 9.420000 | 0.200000 | 78.769997 | 1 |
| **max** | 32.660000 | 8.460000 | 41.740002 | 99.879997 | 2.700000 | 263.600006 | 13 |

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])
```

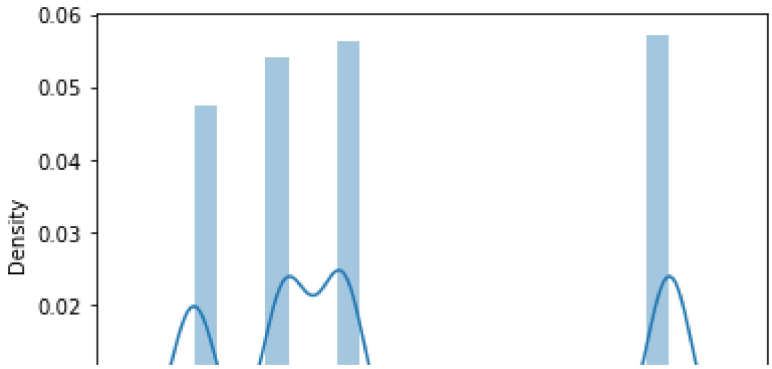
Out[19]: <seaborn.axisgrid.PairGrid at 0x1c1d56f0460>



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

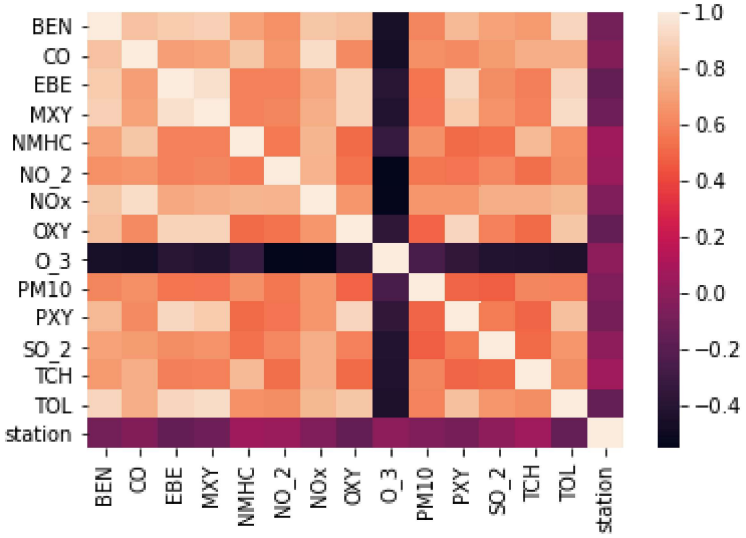
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
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]:
```
28078987.901386347
```

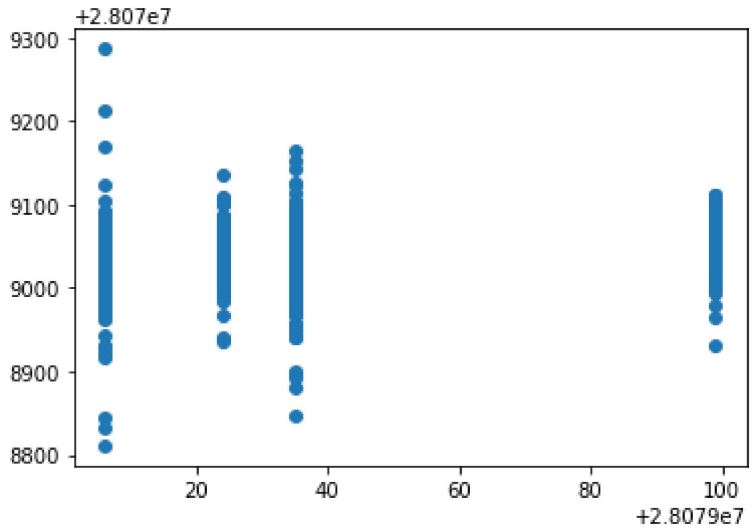
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

| | Co-efficient |
|---|---|
| **BEN** | 2.125454 |
| **CO** | -13.766248 |
| **EBE** | -11.419364 |
| **MXY** | 4.140084 |
| **NMHC** | 83.042808 |
| **NO_2** | 0.255394 |
| **NOx** | -0.098849 |
| **OXY** | -5.100131 |
| **O_3** | -0.033571 |
| **PM10** | -0.115078 |
| **PXY** | 7.991349 |
| **SO_2** | 0.671541 |
| **TCH** | 43.578380 |
| **TOL** | -1.661425 |

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

Out[27]: <matplotlib.collections.PathCollection at 0x1c1e4f26580>




# ACCURACY

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

Out[28]: 0.1837216607188462

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

Out[29]: 0.20410773820018646

# 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]: 0.18324514844316653

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

Out[33]:
```
0.20390330994003047
```

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]:
```
0.061940553181437985
```

# Accuracy(Lasso)

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

Out[36]:
```
0.05426985291531594
```

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([ 1.11180324,  0.        , -2.79863616,  1.53404939,  0.23062826,
        0.22914989, -0.02762068, -2.37311702, -0.02536318,  0.01281824,
        2.32523214,  0.47914468,  1.11459539, -1.30459123])
```

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

Out[39]:
```
28079038.287570737
```

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

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

Out[41]:
```
0.08795788332481069
```

# 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)))
```

```
28.481753575637345
1109.382765069125
33.30739805312215
```

# 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]:
```
(32381, 14)
```

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

Out[46]:
```
(32381,)
```

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)
```

```
[28079035]
```

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

Out[52]:
```
array([28079006, 28079024, 28079035, 28079099], dtype=int64)
```

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

Out[53]:
```
0.8480899292795158
```

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

Out[54]:
```
2.5638972732451705e-10
```

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

Out[55]:
```
array([[2.56389727e-10, 3.44199742e-71, 1.00000000e+00, 1.43898646e-13]])
```

# 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]:
```
0.772919791758581
```

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

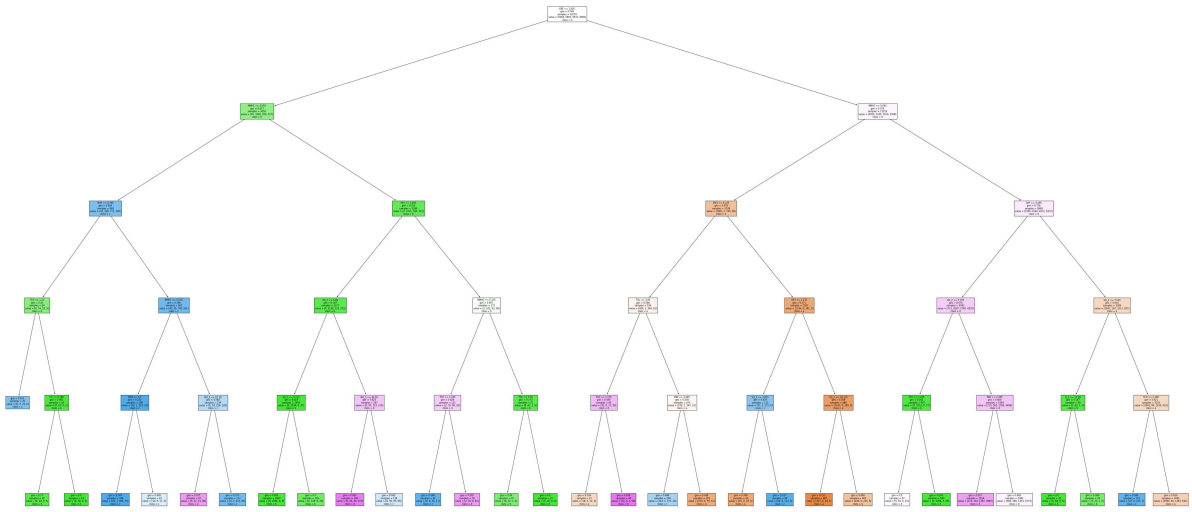
In [62]:
```
from sklearn.tree import plot_tree

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

Out[62]:
```
[Text(2131.377049180328, 1993.2, 'EBE <= 1.005\ngini = 0.749\nsamples = 14276
\nvalue = [5004, 5605, 5972, 6085]\nclass = d'),
 Text(969.639344262295, 1630.8000000000002, 'NMHC <= 0.055\ngini = 0.477\nsam
ples = 3058\nvalue = [45, 3340, 956, 517]\nclass = b'),
 Text(402.4918032786885, 1268.4, 'MXY <= 0.765\ngini = 0.464\nsamples = 669\n
value = [43, 109, 772, 166]\nclass = c'),
 Text(146.36065573770492, 906.0, 'TCH <= 1.23\ngini = 0.42\nsamples = 64\nval
ue = [0, 74, 23, 5]\nclass = b'),
 Text(73.18032786885246, 543.5999999999999, 'gini = 0.432\nsamples = 20\nvalu
e = [0, 7, 23, 2]\nclass = c'),
 Text(219.54098360655738, 543.5999999999999, 'CO <= 0.285\ngini = 0.082\nsamp
les = 44\nvalue = [0, 67, 0, 3]\nclass = b'),
 Text(146.36065573770492, 181.19999999999982, 'gini = 0.17\nsamples = 20\nval
ue = [0, 29, 0, 3]\nclass = b'),
 Text(292.72131147540983, 181.19999999999982, 'gini = 0.0\nsamples = 24\nvalu
e = [0, 38, 0, 0]\nclass = b'),
 Text(658.6229508196722, 906.0, 'NMHC <= 0.035\ngini = 0.396\nsamples = 605\n
value = [43, 35, 749, 161]\nclass = c'),
 Text(512.2622950819672, 543.5999999999999, 'BEN <= 1.3\ngini = 0.232\nsample
s = 366\nvalue = [43, 1, 523, 33]\nclass = c'),
 Text(439.08196721311475, 181.19999999999982, 'gini = 0.203\nsamples = 346\nv
alue = [29, 1, 506, 33]\nclass = c'),
 Text(585.4426229508197, 181.19999999999982, 'gini = 0.495\nsamples = 20\nval
ue = [14, 0, 17, 0]\nclass = c'),
 Text(804.983606557377, 543.5999999999999, 'NO_2 <= 21.19\ngini = 0.544\nsamp
les = 239\nvalue = [0, 34, 226, 128]\nclass = c'),
 Text(731.8032786885246, 181.19999999999982, 'gini = 0.527\nsamples = 67\nval
ue = [0, 27, 13, 68]\nclass = d'),
 Text(878.1639344262295, 181.19999999999982, 'gini = 0.375\nsamples = 172\nva
lue = [0, 7, 213, 60]\nclass = c'),
 Text(1536.7868852459017, 1268.4, 'PXY <= 1.005\ngini = 0.254\nsamples = 2389
\nvalue = [2, 3231, 184, 351]\nclass = b'),
 Text(1244.0655737704917, 906.0, 'SO_2 <= 5.89\ngini = 0.197\nsamples = 2217
\nvalue = [0, 3129, 123, 253]\nclass = b'),
 Text(1097.704918032787, 543.5999999999999, 'SO_2 <= 5.115\ngini = 0.024\nsam
ples = 1950\nvalue = [0, 3034, 0, 37]\nclass = b'),
 Text(1024.5245901639344, 181.19999999999982, 'gini = 0.005\nsamples = 1847\n
value = [0, 2905, 0, 8]\nclass = b'),
 Text(1170.8852459016393, 181.19999999999982, 'gini = 0.3\nsamples = 103\nval
ue = [0, 129, 0, 29]\nclass = b'),
 Text(1390.4262295081967, 543.5999999999999, 'NO_2 <= 44.05\ngini = 0.624\nsa
mples = 267\nvalue = [0, 95, 123, 216]\nclass = d'),
 Text(1317.2459016393443, 181.19999999999982, 'gini = 0.463\nsamples = 149\nv
alue = [0, 39, 28, 157]\nclass = d'),
 Text(1463.6065573770493, 181.19999999999982, 'gini = 0.645\nsamples = 118\nv
alue = [0, 56, 95, 59]\nclass = c'),
 Text(1829.5081967213114, 906.0, 'NMHC <= 0.125\ngini = 0.657\nsamples = 172
\nvalue = [2, 102, 61, 98]\nclass = b'),
 Text(1683.1475409836066, 543.5999999999999, 'TCH <= 1.245\ngini = 0.628\nsam
ples = 121\nvalue = [2, 35, 60, 92]\nclass = d'),
 Text(1609.967213114754, 181.19999999999982, 'gini = 0.288\nsamples = 42\nval
ue = [0, 0, 52, 11]\nclass = c'),
 Text(1756.327868852459, 181.19999999999982, 'gini = 0.505\nsamples = 79\nval
ue = [2, 35, 8, 81]\nclass = d'),
 Text(1975.8688524590164, 543.5999999999999, 'TOL <= 7.25\ngini = 0.173\nsamp
les = 51\nvalue = [0, 67, 1, 6]\nclass = b'),
 Text(1902.688524590164, 181.19999999999982, 'gini = 0.38\nsamples = 20\nvalu
```

```
e = [0, 22, 1, 6]\nclass = b'),
 Text(2049.0491803278687, 181.19999999999982, 'gini = 0.0\nsamples = 31\nvalu
e = [0, 45, 0, 0]\nclass = b'),
 Text(3293.1147540983607, 1630.8000000000002, 'NMHC <= 0.045\ngini = 0.729\ns
amples = 11218\nvalue = [4959, 2265, 5016, 5568]\nclass = d'),
 Text(2707.6721311475408, 1268.4, 'MXY <= 4.145\ngini = 0.476\nsamples = 1538
\nvalue = [1603, 1, 745, 96]\nclass = a'),
 Text(2414.9508196721313, 906.0, 'TOL <= 3.45\ngini = 0.588\nsamples = 536\nv
alue = [409, 1, 364, 91]\nclass = a'),
 Text(2268.590163934426, 543.5999999999999, 'TCH <= 1.175\ngini = 0.589\nsamp
les = 65\nvalue = [33, 0, 17, 60]\nclass = d'),
 Text(2195.409836065574, 181.19999999999982, 'gini = 0.539\nsamples = 21\nval
ue = [18, 0, 12, 2]\nclass = a'),
 Text(2341.7704918032787, 181.19999999999982, 'gini = 0.406\nsamples = 44\nva
lue = [15, 0, 5, 58]\nclass = d'),
 Text(2561.311475409836, 543.5999999999999, 'EBE <= 1.545\ngini = 0.539\nsamp
les = 471\nvalue = [376, 1, 347, 31]\nclass = a'),
 Text(2488.1311475409834, 181.19999999999982, 'gini = 0.498\nsamples = 269\nv
alue = [143, 1, 277, 20]\nclass = c'),
 Text(2634.4918032786886, 181.19999999999982, 'gini = 0.398\nsamples = 202\nv
alue = [233, 0, 70, 11]\nclass = a'),
 Text(3000.3934426229507, 906.0, 'OXY <= 2.215\ngini = 0.371\nsamples = 1002
\nvalue = [1194, 0, 381, 5]\nclass = a'),
 Text(2854.032786885246, 543.5999999999999, 'SO_2 <= 7.855\ngini = 0.416\nsam
ples = 114\nvalue = [51, 0, 122, 0]\nclass = c'),
 Text(2780.8524590163934, 181.19999999999982, 'gini = 0.369\nsamples = 24\nva
lue = [31, 0, 10, 0]\nclass = a'),
 Text(2927.2131147540986, 181.19999999999982, 'gini = 0.257\nsamples = 90\nva
lue = [20, 0, 112, 0]\nclass = c'),
 Text(3146.754098360656, 543.5999999999999, 'SO_2 <= 10.125\ngini = 0.306\nsa
mples = 888\nvalue = [1143, 0, 259, 5]\nclass = a'),
 Text(3073.5737704918033, 181.19999999999982, 'gini = 0.121\nsamples = 487\nv
alue = [717, 0, 44, 5]\nclass = a'),
 Text(3219.934426229508, 181.19999999999982, 'gini = 0.446\nsamples = 401\nva
lue = [426, 0, 215, 0]\nclass = a'),
 Text(3878.55737704918, 1268.4, 'OXY <= 4.095\ngini = 0.726\nsamples = 9680\n
value = [3356, 2264, 4271, 5472]\nclass = d'),
 Text(3585.8360655737706, 906.0, 'SO_2 <= 5.765\ngini = 0.676\nsamples = 6391
\nvalue = [713, 2097, 2760, 4535]\nclass = d'),
 Text(3439.4754098360654, 543.5999999999999, 'TCH <= 1.265\ngini = 0.092\nsam
ples = 997\nvalue = [0, 1514, 0, 77]\nclass = b'),
 Text(3366.2950819672133, 181.19999999999982, 'gini = 0.5\nsamples = 67\nvalu
e = [0, 50, 0, 51]\nclass = d'),
 Text(3512.655737704918, 181.19999999999982, 'gini = 0.034\nsamples = 930\nva
lue = [0, 1464, 0, 26]\nclass = b'),
 Text(3732.1967213114754, 543.5999999999999, 'EBE <= 2.285\ngini = 0.609\nsam
ples = 5394\nvalue = [713, 583, 2760, 4458]\nclass = d'),
 Text(3659.0163934426228, 181.19999999999982, 'gini = 0.527\nsamples = 2914\n
value = [133, 303, 1307, 2885]\nclass = d'),
 Text(3805.377049180328, 181.19999999999982, 'gini = 0.669\nsamples = 2480\nv
alue = [580, 280, 1453, 1573]\nclass = d'),
 Text(4171.2786885245905, 906.0, 'SO_2 <= 7.125\ngini = 0.632\nsamples = 3289
\nvalue = [2643, 167, 1511, 937]\nclass = a'),
 Text(4024.9180327868853, 543.5999999999999, 'O_3 <= 11.99\ngini = 0.147\nsam
ples = 56\nvalue = [2, 83, 1, 4]\nclass = b'),
 Text(3951.7377049180327, 181.19999999999982, 'gini = 0.0\nsamples = 36\nvalu
e = [0, 60, 0, 0]\nclass = b'),
```

```
 Text(4098.098360655737, 181.19999999999982, 'gini = 0.389\nsamples = 20\nval
ue = [2, 23, 1, 4]\nclass = b'),
 Text(4317.639344262295, 543.5999999999999, 'TCH <= 1.285\ngini = 0.621\nsamp
les = 3233\nvalue = [2641, 84, 1510, 933]\nclass = a'),
 Text(4244.459016393443, 181.19999999999982, 'gini = 0.289\nsamples = 172\nva
lue = [47, 0, 227, 1]\nclass = c'),
 Text(4390.819672131148, 181.19999999999982, 'gini = 0.614\nsamples = 3061\nv
alue = [2594, 84, 1283, 932]\nclass = a')]
```



# Conclusion

## Accuracy

***Linear Regression:0.20410773820018646***

***Ridge Regression:0.20390330994003047***

***Lasso Regression:0.061940553181437985***

***ElasticNet Regression:0.08795788332481069***

***Logistic Regression:0.8480899292795158***

***Random Forest:0.772919791758581***

## Logistic Regression is suitable for this dataset