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"E:\154\C10_air\csvs_per_year\csvs_per_year\madrid_2011.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	NMHC	NO	NO_2	O_3	PM10	PM25	SO_2	TCH	TOL	
0	2011-11-01 01:00:00	NaN	1.0	NaN	NaN	154.0	84.0	NaN	NaN	NaN	6.0	NaN	NaN	2
1	2011-11-01 01:00:00	2.5	0.4	3.5	0.26	68.0	92.0	3.0	40.0	24.0	9.0	1.54	8.7	2
2	2011-11-01 01:00:00	2.9	NaN	3.8	NaN	96.0	99.0	NaN	NaN	NaN	NaN	NaN	7.2	2
3	2011-11-01 01:00:00	NaN	0.6	NaN	NaN	60.0	83.0	2.0	NaN	NaN	NaN	NaN	NaN	2
4	2011-11-01 01:00:00	NaN	NaN	NaN	NaN	44.0	62.0	3.0	NaN	NaN	3.0	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209923	2011-09-01 00:00:00	NaN	0.2	NaN	NaN	5.0	19.0	44.0	NaN	NaN	NaN	NaN	NaN	2
209924	2011-09-01 00:00:00	NaN	0.1	NaN	NaN	6.0	29.0	NaN	11.0	NaN	7.0	NaN	NaN	2
209925	2011-09-01 00:00:00	NaN	NaN	NaN	0.23	1.0	21.0	28.0	NaN	NaN	NaN	1.44	NaN	2
209926	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	3.0	15.0	48.0	NaN	NaN	NaN	NaN	NaN	2
209927	2011-09-01 00:00:00	NaN	NaN	NaN	NaN	4.0	33.0	38.0	13.0	NaN	NaN	NaN	NaN	2

209928 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', 'PM2
5',
       'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16460 entries, 1 to 209910
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16460 non-null  object 
 1   BEN      16460 non-null  float64
 2   CO       16460 non-null  float64
 3   EBE      16460 non-null  float64
 4   NMHC     16460 non-null  float64
 5   NO       16460 non-null  float64
 6   NO_2     16460 non-null  float64
 7   O_3      16460 non-null  float64
 8   PM10     16460 non-null  float64
 9   PM25     16460 non-null  float64
 10  SO_2     16460 non-null  float64
 11  TCH      16460 non-null  float64
 12  TOL      16460 non-null  float64
 13  station  16460 non-null  int64  
dtypes: float64(12), int64(1), object(1)
memory usage: 1.9+ MB
```

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

Out[6]:

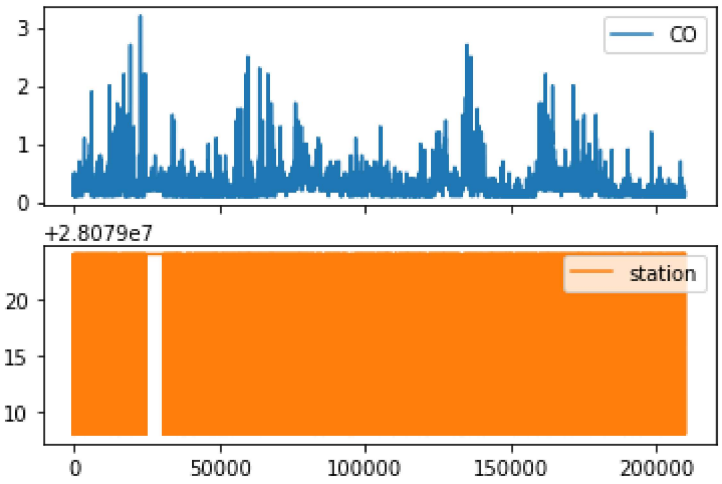
	CO	station
1	0.4	28079008
6	0.3	28079024
25	0.3	28079008
30	0.4	28079024
49	0.2	28079008
...	...	...
209862	0.1	28079024
209881	0.1	28079008
209886	0.1	28079024
209905	0.1	28079008
209910	0.1	28079024

16460 rows × 2 columns

Line chart

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

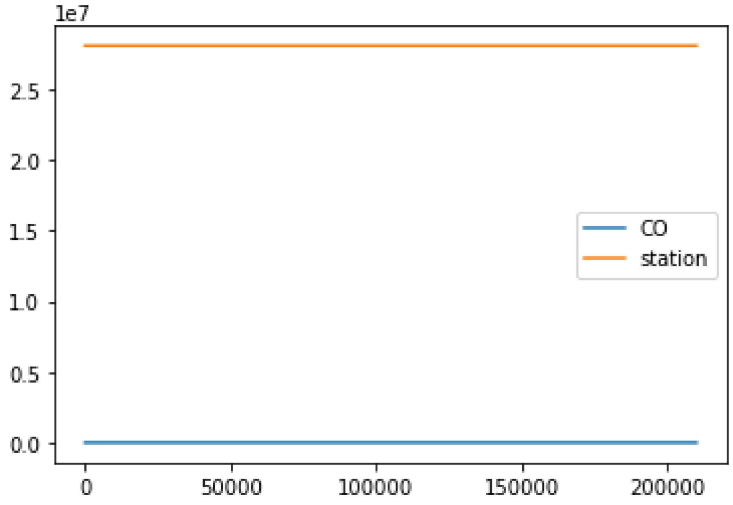
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

Out[8]: <AxesSubplot:>

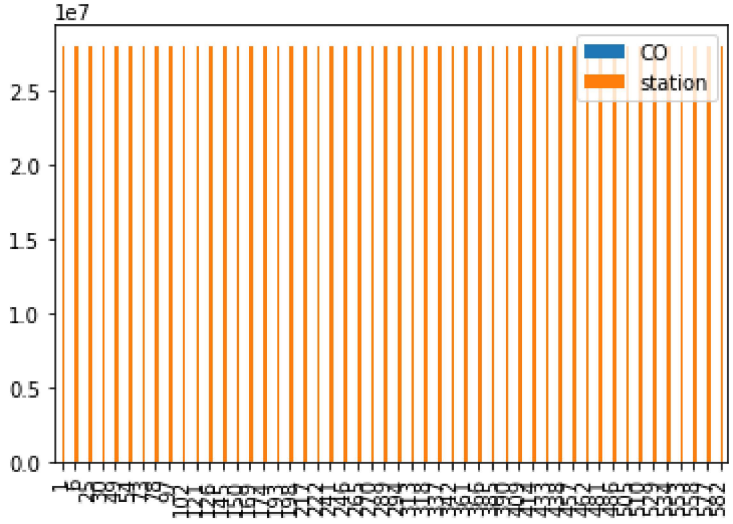


Bar chart

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

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

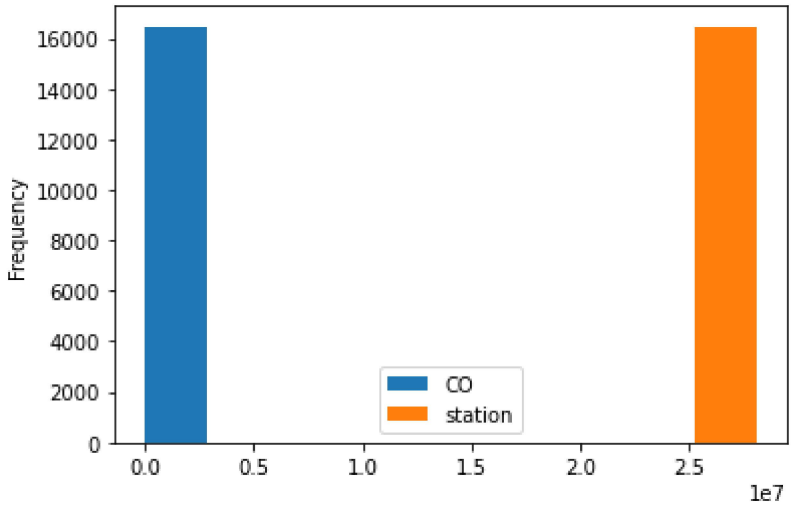
Out[10]: <AxesSubplot:>



Histogram

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

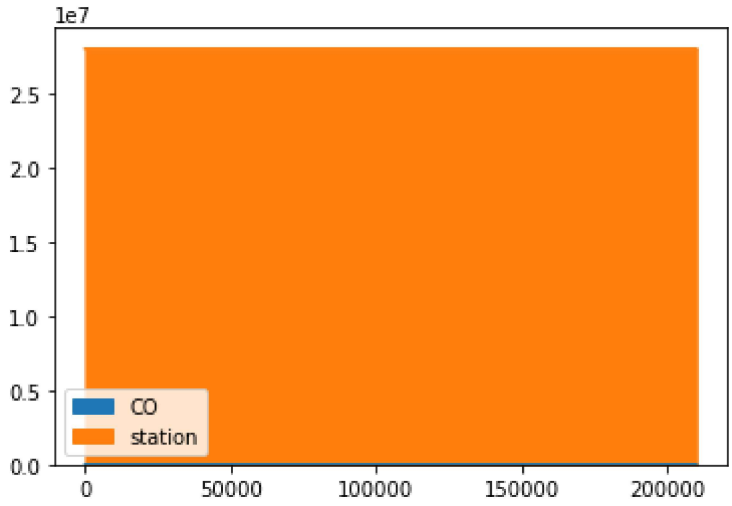
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

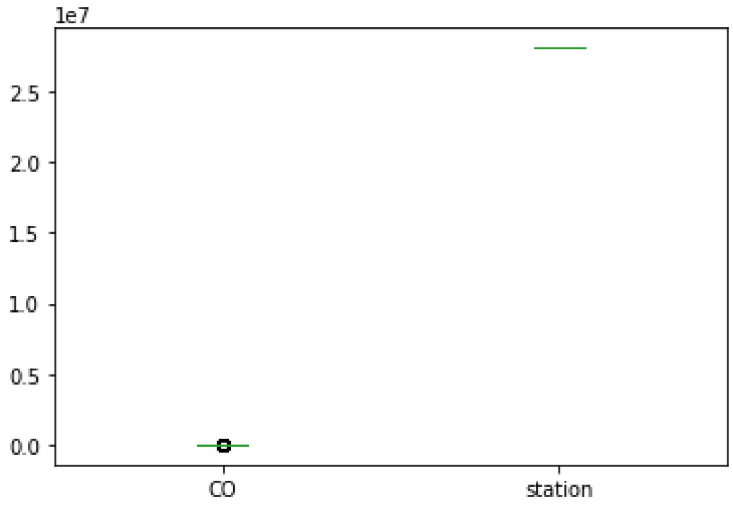
Out[12]: <AxesSubplot:>



Box chart

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

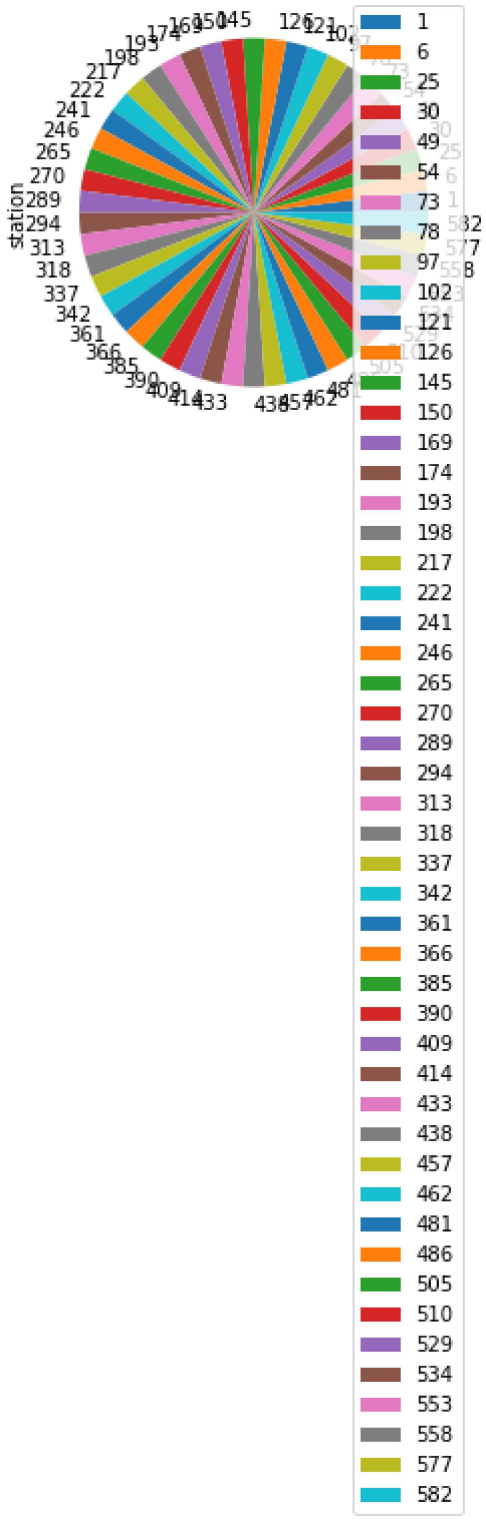
Out[13]: <AxesSubplot:>



Pie chart

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

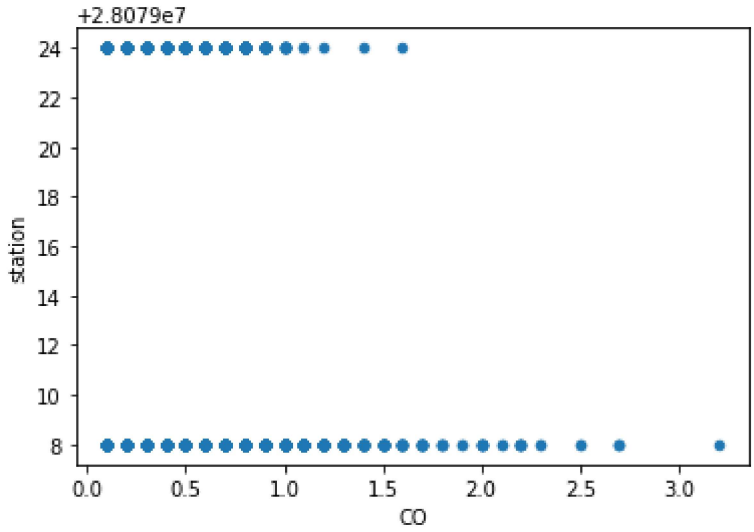
Out[14]: <AxesSubplot:ylabel='station'>



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: 16460 entries, 1 to 209910
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     16460 non-null  object 
 1   BEN      16460 non-null  float64
 2   CO       16460 non-null  float64
 3   EBE      16460 non-null  float64
 4   NMHC     16460 non-null  float64
 5   NO       16460 non-null  float64
 6   NO_2     16460 non-null  float64
 7   O_3      16460 non-null  float64
 8   PM10     16460 non-null  float64
 9   PM25     16460 non-null  float64
 10  SO_2     16460 non-null  float64
 11  TCH      16460 non-null  float64
 12  TOL      16460 non-null  float64
 13  station  16460 non-null  int64
dtypes: float64(12), int64(1), object(1)
```

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

Out[17]:

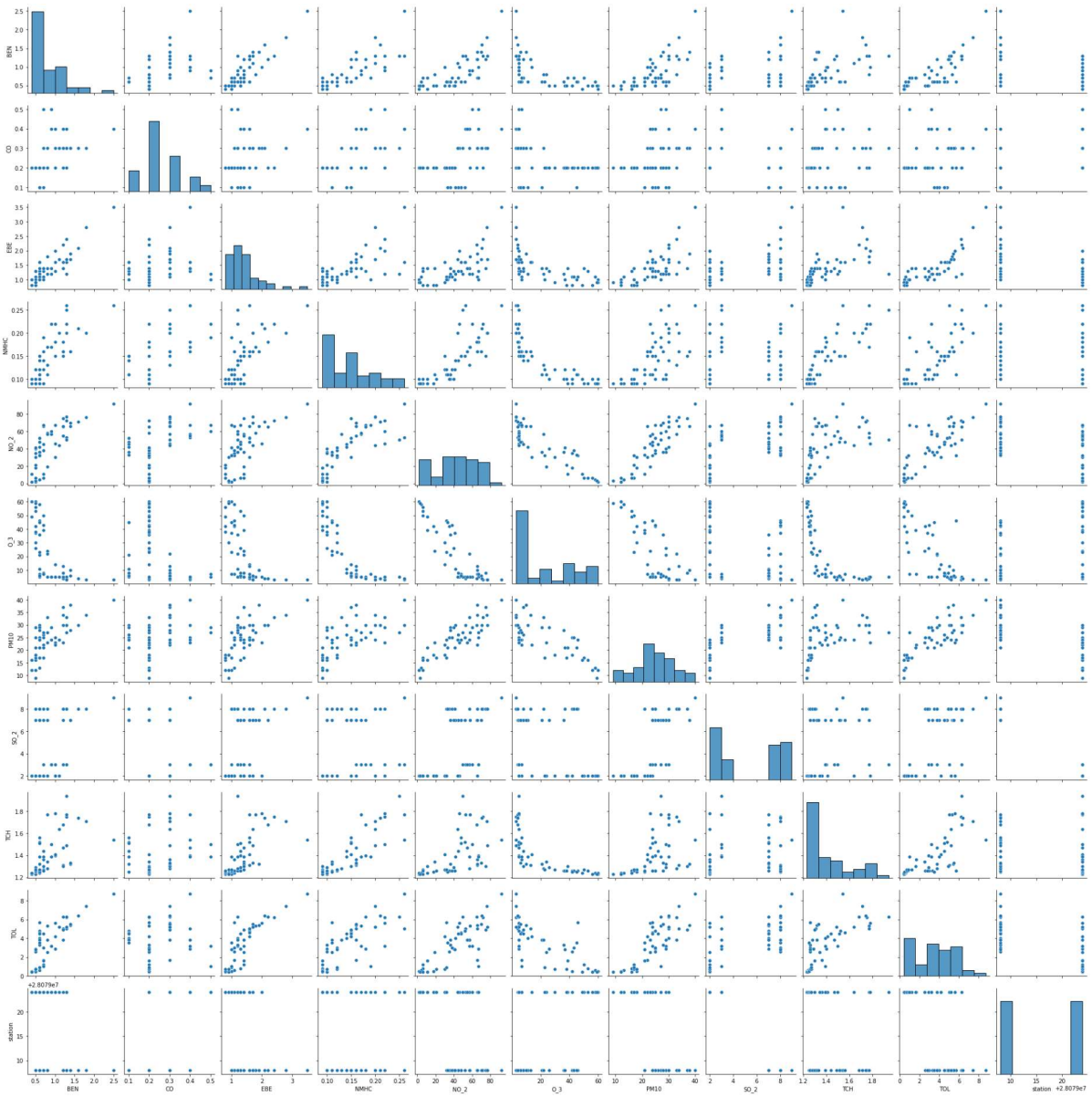
	BEN	CO	EBE	NMHC	NO	NO_2	
count	16460.000000	16460.000000	16460.000000	16460.000000	16460.000000	16460.000000	164
mean	0.900680	0.277758	1.471871	0.167043	23.671810	44.583961	
std	0.768892	0.206143	1.051004	0.075068	44.362859	31.569185	
min	0.100000	0.100000	0.200000	0.010000	1.000000	1.000000	
25%	0.500000	0.200000	0.800000	0.120000	2.000000	19.000000	
50%	0.700000	0.200000	1.200000	0.160000	7.000000	40.000000	
75%	1.100000	0.300000	1.700000	0.200000	25.000000	63.000000	
max	9.500000	3.200000	12.800000	0.840000	615.000000	289.000000	1

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

EDA AND VISUALIZATION

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

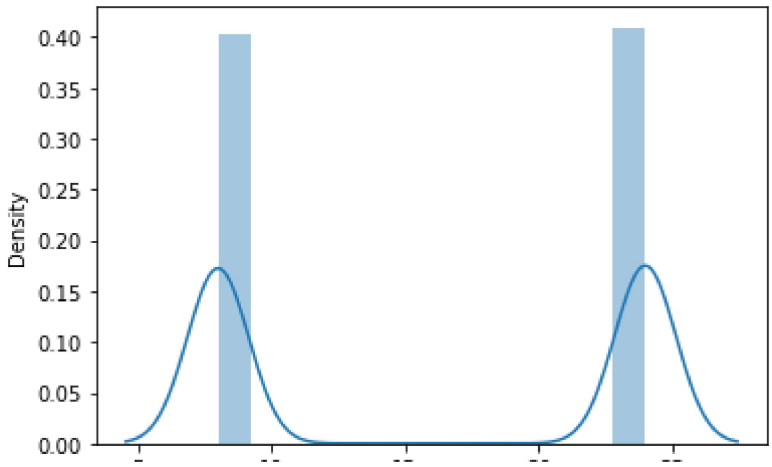
Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x1b4c1143af0>
```



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

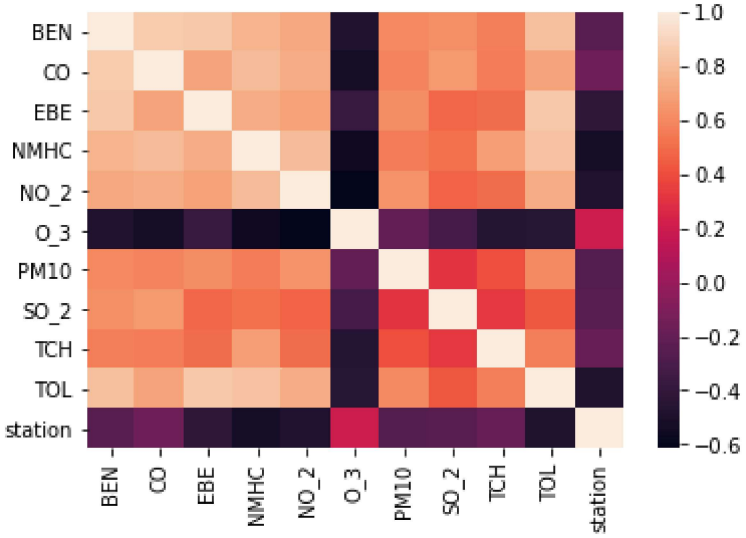
```
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
```

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','NMHC', 'NO_2','O_3',
      'PM10', '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]:
```
28079018.379894726
```

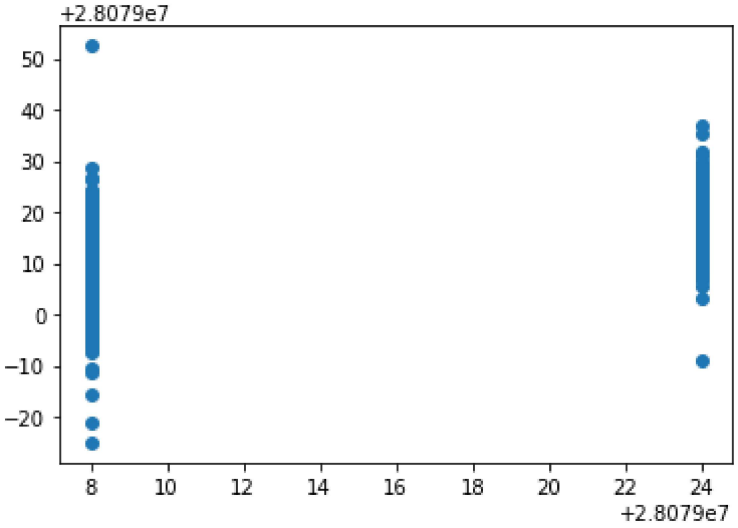
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	3.957996
CO	32.795678
EBE	-2.128572
NMHC	-95.653807
NO_2	-0.081529
O_3	-0.014847
PM10	0.000658
SO_2	-0.524357
TCH	9.546366
TOL	-0.447424

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

Out[27]: <matplotlib.collections.PathCollection at 0x1b4c9708ca0>



ACCURACY

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

Out[28]: 0.6169329869461591

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

Out[29]: 0.6236818673887985

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]: 0.5860635585849482

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

Out[33]: 0.5895649062638599

In [34]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[34]: Lasso(alpha=10)

Accuracy(Lasso)

In [35]:
```
la.score(x_train,y_train)
```

Out[35]: 0.2293062526678089

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

Out[36]: 0.24597881956998158

ElasticNet

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([ 6.19490205e-01,  0.00000000e+00, -0.00000000e+00, -0.00000000e+00,
       -1.28854861e-01, -5.36514789e-02,  7.10066717e-02, -4.63639394e-04,
        0.00000000e+00, -7.31046739e-01])
```

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

Out[39]: 28079024.273564596

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

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

Out[41]: 0.3321355459243095

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)))
```

```
5.826072893511698
42.729750322206954
6.536799700327903
```

Logistic Regression

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

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

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

Out[45]:
```
(16460, 10)
```

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

Out[46]:
```
(16460,)
```

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]]
```

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

```
[28079008]
```

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

Out[52]:
```
array([28079008, 28079024], dtype=int64)
```

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

Out[53]:
```
0.9237545565006076
```

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

Out[54]:
```
0.9999999999999966
```

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

Out[55]:
```
array([[1.00000000e+00, 3.47334507e-15]])
```

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]:
```
0.9346467627148065
```

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

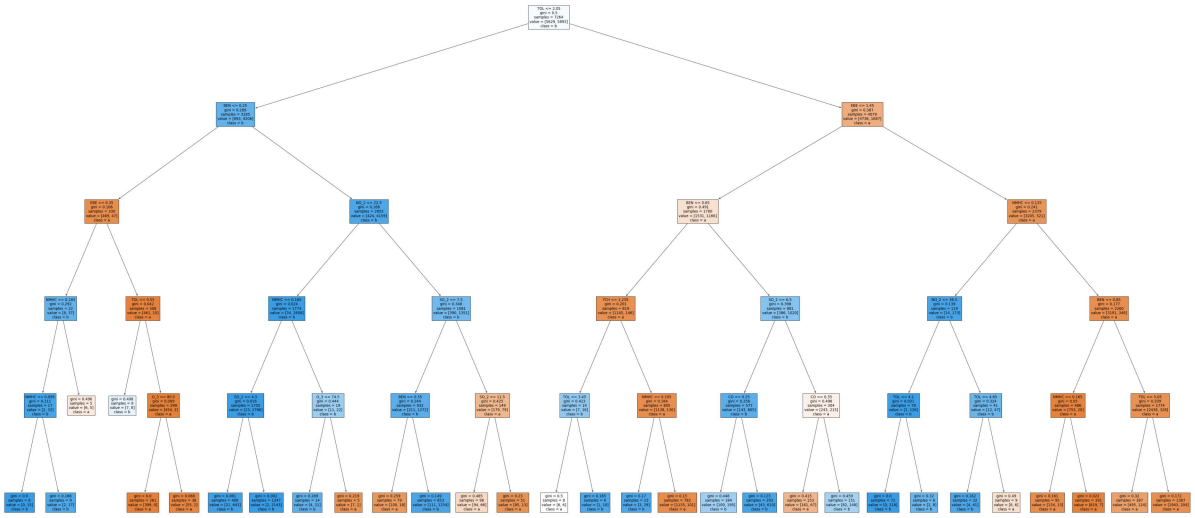
In [62]:
```
from sklearn.tree import plot_tree

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

Out[62]: [Text(2058.8275862068967, 1993.2, 'TOL <= 2.05\ngini = 0.5\nsamples = 7264\nvalue = [5629, 5893]\nclass = b'),
 Text(885.1034482758621, 1630.8000000000002, 'BEN <= 0.25\ngini = 0.289\nsamples = 3185\nvalue = [893, 4206]\nclass = b'),
 Text(384.82758620689657, 1268.4, 'EBE <= 0.35\ngini = 0.166\nsamples = 330\nvalue = [469, 47]\nclass = a'),
 Text(230.89655172413796, 906.0, 'NMHC <= 0.185\ngini = 0.292\nsamples = 22\nvalue = [8, 37]\nclass = b'),
 Text(153.93103448275863, 543.5999999999999, 'NMHC <= 0.095\ngini = 0.111\nsamples = 17\nvalue = [2, 32]\nclass = b'),
 Text(76.96551724137932, 181.19999999999982, 'gini = 0.0\nsamples = 8\nvalue = [0, 15]\nclass = b'),
 Text(230.89655172413796, 181.19999999999982, 'gini = 0.188\nsamples = 9\nvalue = [2, 17]\nclass = b'),
 Text(307.86206896551727, 543.5999999999999, 'gini = 0.496\nsamples = 5\nvalue = [6, 5]\nclass = a'),
 Text(538.7586206896552, 906.0, 'TOL <= 0.55\ngini = 0.042\nsamples = 308\nvalue = [461, 10]\nclass = a'),
 Text(461.79310344827593, 543.5999999999999, 'gini = 0.498\nsamples = 9\nvalue = [7, 8]\nclass = b'),
 Text(615.7241379310345, 543.5999999999999, 'O_3 <= 80.5\ngini = 0.009\nsamples = 299\nvalue = [454, 2]\nclass = a'),
 Text(538.7586206896552, 181.19999999999982, 'gini = 0.0\nsamples = 261\nvalue = [399, 0]\nclass = a'),
 Text(692.6896551724138, 181.19999999999982, 'gini = 0.068\nsamples = 38\nvalue = [55, 2]\nclass = a'),
 Text(1385.3793103448277, 1268.4, 'NO_2 <= 22.5\ngini = 0.168\nsamples = 2855\nvalue = [424, 4159]\nclass = b'),
 Text(1077.5172413793105, 906.0, 'NMHC <= 0.165\ngini = 0.024\nsamples = 1774\nvalue = [34, 2808]\nclass = b'),
 Text(923.5862068965519, 543.5999999999999, 'SO_2 <= 4.5\ngini = 0.016\nsamples = 1755\nvalue = [23, 2786]\nclass = b'),
 Text(846.6206896551724, 181.19999999999982, 'gini = 0.061\nsamples = 408\nvalue = [21, 643]\nclass = b'),
 Text(1000.5517241379312, 181.19999999999982, 'gini = 0.002\nsamples = 1347\nvalue = [2, 2143]\nclass = b'),
 Text(1231.448275862069, 543.5999999999999, 'O_3 <= 74.5\ngini = 0.444\nsamples = 19\nvalue = [11, 22]\nclass = b'),
 Text(1154.4827586206898, 181.19999999999982, 'gini = 0.269\nsamples = 14\nvalue = [4, 21]\nclass = b'),
 Text(1308.4137931034484, 181.19999999999982, 'gini = 0.219\nsamples = 5\nvalue = [7, 1]\nclass = a'),
 Text(1693.2413793103449, 906.0, 'SO_2 <= 7.5\ngini = 0.348\nsamples = 1081\nvalue = [390, 1351]\nclass = b'),
 Text(1539.3103448275863, 543.5999999999999, 'BEN <= 0.35\ngini = 0.244\nsamples = 932\nvalue = [211, 1272]\nclass = b'),
 Text(1462.344827586207, 181.19999999999982, 'gini = 0.259\nsamples = 79\nvalue = [100, 18]\nclass = a'),
 Text(1616.2758620689656, 181.19999999999982, 'gini = 0.149\nsamples = 853\nvalue = [111, 1254]\nclass = b'),
 Text(1847.1724137931037, 543.5999999999999, 'SO_2 <= 11.5\ngini = 0.425\nsamples = 149\nvalue = [179, 79]\nclass = a'),
 Text(1770.2068965517242, 181.19999999999982, 'gini = 0.485\nsamples = 98\nvalue = [94, 66]\nclass = a'),
 Text(1924.137931034483, 181.19999999999982, 'gini = 0.23\nsamples = 51\nvalue = [85, 13]\nclass = a'),
 Text(3232.551724137931, 1630.8000000000002, 'EBE <= 1.45\ngini = 0.387\nsamp

```
les = 4079\nvalue = [4736, 1687]\nclass = a'),
 Text(2616.8275862068967, 1268.4, 'BEN <= 0.65\ngini = 0.491\nsamples = 1700
\nvalue = [1531, 1166]\nclass = a'),
 Text(2308.9655172413795, 906.0, 'TCH <= 1.235\ngini = 0.201\nsamples = 819\n
value = [1145, 146]\nclass = a'),
 Text(2155.034482758621, 543.5999999999999, 'TOL <= 3.45\ngini = 0.423\nsampl
es = 14\nvalue = [7, 16]\nclass = b'),
 Text(2078.0689655172414, 181.19999999999982, 'gini = 0.5\nsamples = 8\nvalue
= [6, 6]\nclass = a'),
 Text(2232.0, 181.19999999999982, 'gini = 0.165\nsamples = 6\nvalue = [1, 10]
\nclass = b'),
 Text(2462.896551724138, 543.5999999999999, 'NMHC <= 0.105\ngini = 0.184\nsam
ples = 805\nvalue = [1138, 130]\nclass = a'),
 Text(2385.9310344827586, 181.19999999999982, 'gini = 0.17\nsamples = 23\nval
ue = [3, 29]\nclass = b'),
 Text(2539.8620689655177, 181.19999999999982, 'gini = 0.15\nsamples = 782\nva
lue = [1135, 101]\nclass = a'),
 Text(2924.689655172414, 906.0, 'SO_2 <= 6.5\ngini = 0.398\nsamples = 881\nva
lue = [386, 1020]\nclass = b'),
 Text(2770.7586206896553, 543.5999999999999, 'CO <= 0.25\ngini = 0.256\nsampl
es = 577\nvalue = [143, 805]\nclass = b'),
 Text(2693.7931034482763, 181.19999999999982, 'gini = 0.448\nsamples = 184\nv
alue = [100, 195]\nclass = b'),
 Text(2847.724137931035, 181.19999999999982, 'gini = 0.123\nsamples = 393\nva
lue = [43, 610]\nclass = b'),
 Text(3078.6206896551726, 543.5999999999999, 'CO <= 0.35\ngini = 0.498\nsampl
es = 304\nvalue = [243, 215]\nclass = a'),
 Text(3001.6551724137935, 181.19999999999982, 'gini = 0.415\nsamples = 153\nv
alue = [161, 67]\nclass = a'),
 Text(3155.586206896552, 181.19999999999982, 'gini = 0.459\nsamples = 151\nva
lue = [82, 148]\nclass = b'),
 Text(3848.275862068966, 1268.4, 'NMHC <= 0.135\ngini = 0.241\nsamples = 2379
\nvalue = [3205, 521]\nclass = a'),
 Text(3540.4137931034484, 906.0, 'NO_2 <= 36.5\ngini = 0.139\nsamples = 119\n
value = [14, 173]\nclass = b'),
 Text(3386.4827586206898, 543.5999999999999, 'TOL <= 4.1\ngini = 0.031\nsampl
es = 78\nvalue = [2, 126]\nclass = b'),
 Text(3309.5172413793107, 181.19999999999982, 'gini = 0.0\nsamples = 72\nvalu
e = [0, 118]\nclass = b'),
 Text(3463.4482758620693, 181.19999999999982, 'gini = 0.32\nsamples = 6\nvalu
e = [2, 8]\nclass = b'),
 Text(3694.3448275862074, 543.5999999999999, 'TOL <= 4.65\ngini = 0.324\nsamp
les = 41\nvalue = [12, 47]\nclass = b'),
 Text(3617.379310344828, 181.19999999999982, 'gini = 0.162\nsamples = 32\nval
ue = [4, 41]\nclass = b'),
 Text(3771.3103448275865, 181.19999999999982, 'gini = 0.49\nsamples = 9\nvalu
e = [8, 6]\nclass = a'),
 Text(4156.137931034483, 906.0, 'BEN <= 0.85\ngini = 0.177\nsamples = 2260\nv
alue = [3191, 348]\nclass = a'),
 Text(4002.2068965517246, 543.5999999999999, 'NMHC <= 0.165\ngini = 0.05\nsam
ples = 486\nvalue = [753, 20]\nclass = a'),
 Text(3925.241379310345, 181.19999999999982, 'gini = 0.161\nsamples = 95\nval
ue = [134, 13]\nclass = a'),
 Text(4079.1724137931037, 181.19999999999982, 'gini = 0.022\nsamples = 391\nv
alue = [619, 7]\nclass = a'),
 Text(4310.068965517242, 543.5999999999999, 'TOL <= 5.05\ngini = 0.209\nsampl
es = 1774\nvalue = [2438, 328]\nclass = a'),
```

```
 Text(4233.103448275862, 181.19999999999982, 'gini = 0.32\nsamples = 387\nval
ue = [495, 124]\nclass = a'),
 Text(4387.034482758621, 181.19999999999982, 'gini = 0.172\nsamples = 1387\nv
alue = [1943, 204]\nclass = a')]
```



Accuracy

Linear Regression

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

Out[63]: 0.6236818673887985

Ridge Regression

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

Out[64]: 0.5895649062638599

Lasso Regression

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

Out[65]: 0.24597881956998158

ElasticNet Regression

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

Out[66]:
```
0.3321355459243095
```

Logistic Regression

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

Out[67]:
```
0.9237545565006076
```

Random Forest

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

Out[68]:
```
0.9346467627148065
```

Conclusion

RandomForest for this dataset