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_2010.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.410
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.670
...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999	
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN	51.259
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000	
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999	
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998	47.150

209448 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: 6666 entries, 11 to 191927
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     6666 non-null   object 
 1   BEN      6666 non-null   float64
 2   CO       6666 non-null   float64
 3   EBE      6666 non-null   float64
 4   MXY      6666 non-null   float64
 5   NMHC     6666 non-null   float64
 6   NO_2     6666 non-null   float64
 7   NOx      6666 non-null   float64
 8   OXY      6666 non-null   float64
 9   O_3      6666 non-null   float64
 10  PM10     6666 non-null   float64
 11  PM25     6666 non-null   float64
 12  PXY      6666 non-null   float64
 13  SO_2     6666 non-null   float64
 14  TCH      6666 non-null   float64
 15  TOL      6666 non-null   float64
 16  station  6666 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 937.4+ KB
```

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

Out[6]:

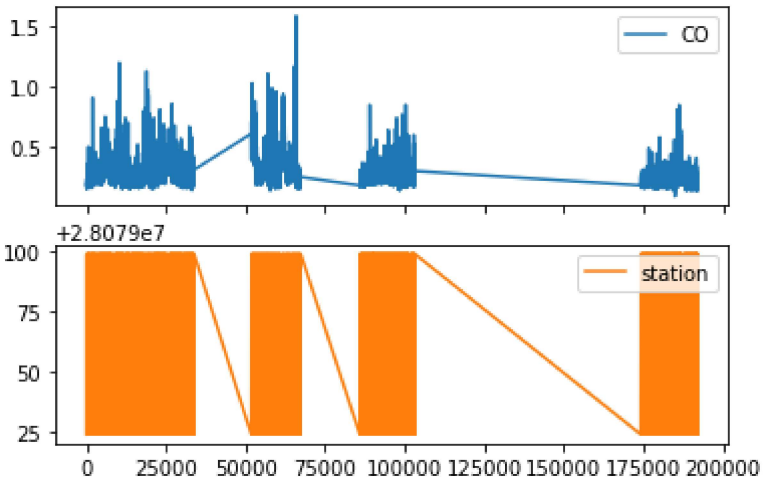
	CO	station
11	0.18	28079024
23	0.23	28079099
35	0.17	28079024
47	0.21	28079099
59	0.16	28079024
...	...	...
191879	0.26	28079099
191891	0.16	28079024
191903	0.28	28079099
191915	0.16	28079024
191927	0.25	28079099

6666 rows × 2 columns

Line chart

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

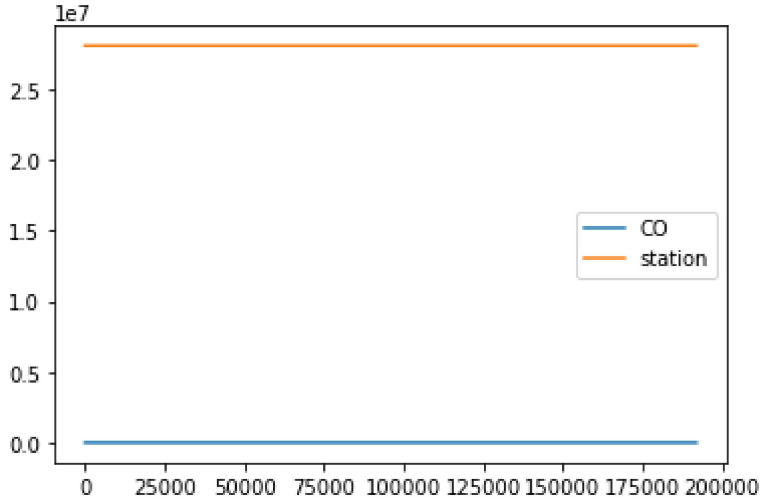
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

Out[8]: <AxesSubplot:>

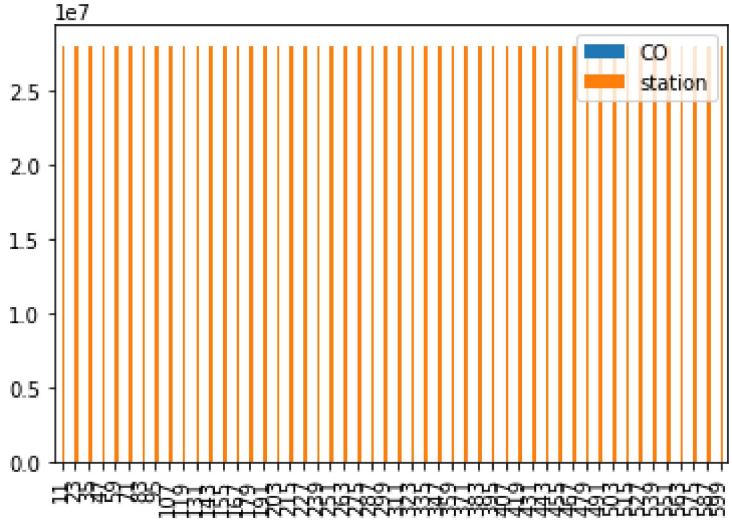


Bar chart

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

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

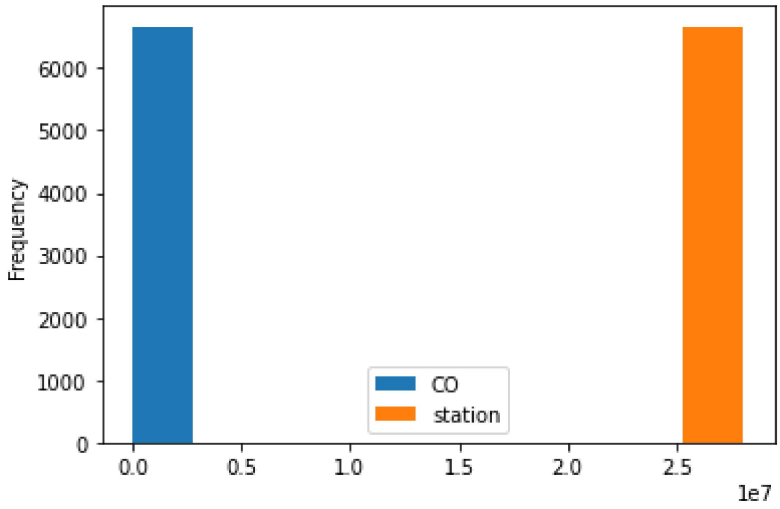
Out[10]: <AxesSubplot:>



Histogram

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

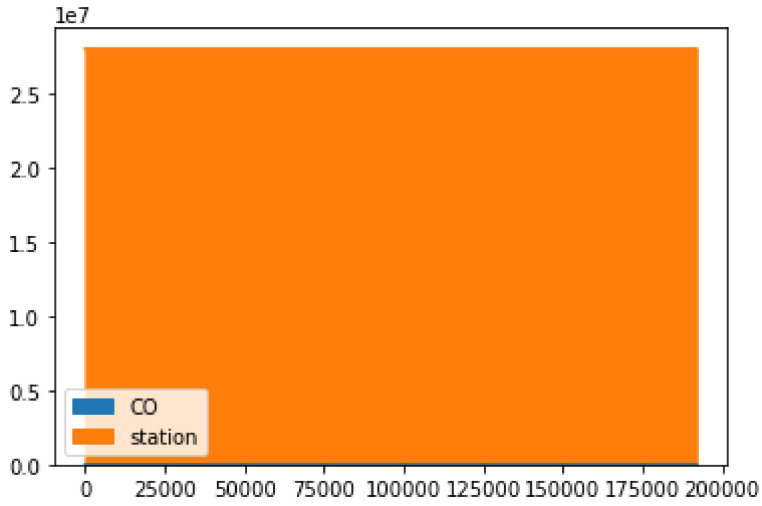
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

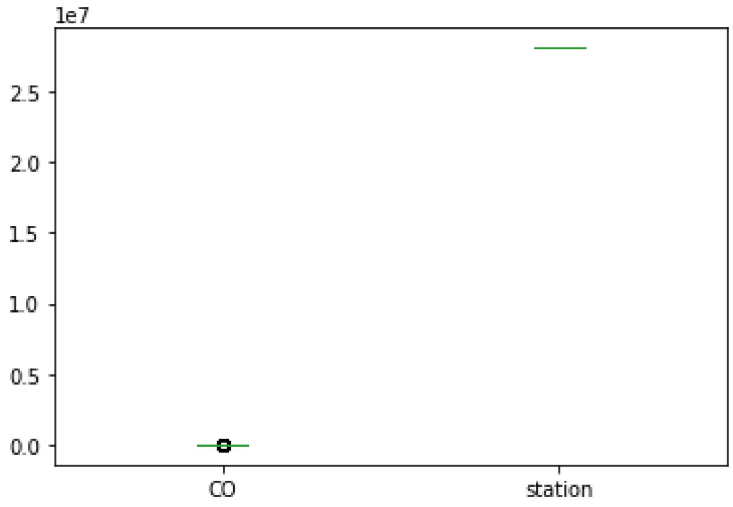
Out[12]: <AxesSubplot:>



Box chart

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

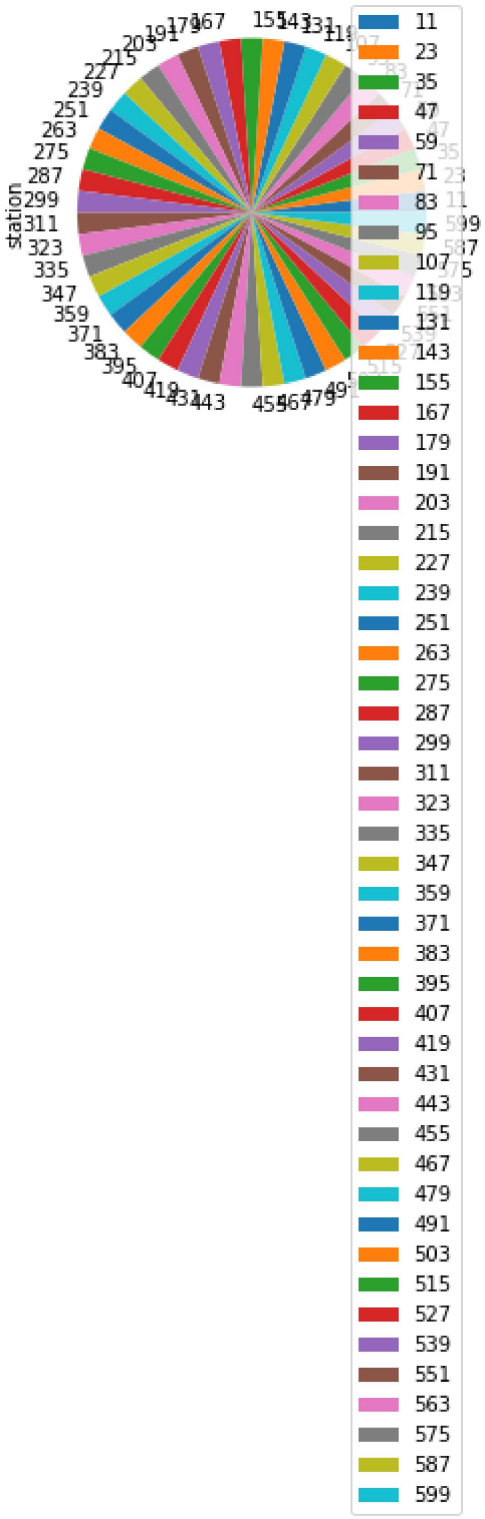
Out[13]: <AxesSubplot:>



Pie chart

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

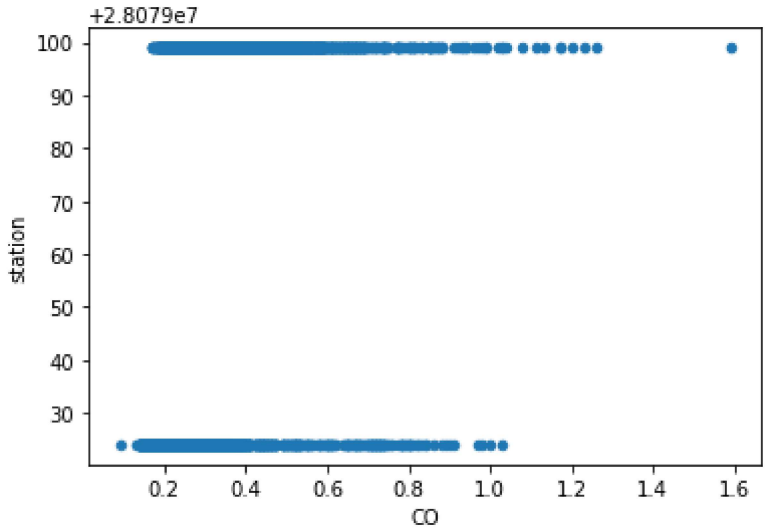
Out[14]: <AxesSubplot:ylabel='station'>



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: 6666 entries, 11 to 191927
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     6666 non-null   object 
 1   BEN      6666 non-null   float64
 2   CO       6666 non-null   float64
 3   EBE      6666 non-null   float64
 4   MXY      6666 non-null   float64
 5   NMHC     6666 non-null   float64
 6   NO_2     6666 non-null   float64
 7   NOx      6666 non-null   float64
 8   OXY      6666 non-null   float64
 9   O_3      6666 non-null   float64
 10  PM10     6666 non-null   float64
 11  PM25     6666 non-null   float64
 12  PXY      6666 non-null   float64
 13  SO_2     6666 non-null   float64
```

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

Out[17]:

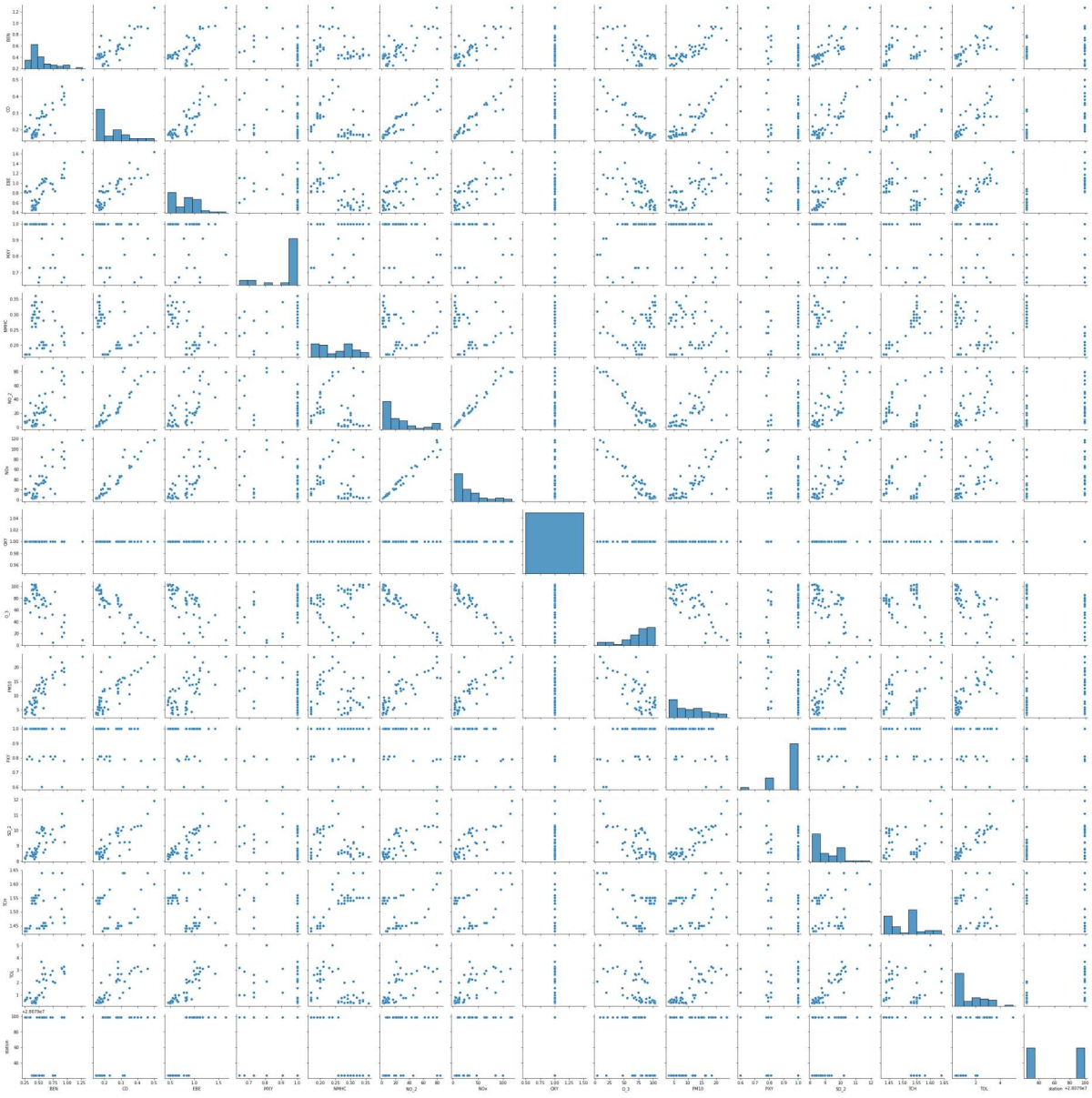
	BEN	CO	EBE	MXY	NMHC	NO_2	NO
count	6666.000000	6666.000000	6666.000000	6666.000000	6666.000000	6666.000000	6666.00000
mean	0.648425	0.296280	0.840585	0.839959	0.243378	33.888744	47.54061
std	0.395346	0.133296	0.508031	0.382263	0.115730	23.465169	41.23057
min	0.170000	0.090000	0.140000	0.110000	0.000000	1.290000	2.76000
25%	0.380000	0.200000	0.470000	0.590000	0.180000	15.752500	19.44250
50%	0.540000	0.260000	0.755000	1.000000	0.220000	29.320000	36.77000
75%	0.810000	0.340000	1.000000	1.000000	0.280000	47.657500	62.10250
max	5.110000	1.590000	5.190000	6.810000	0.930000	133.399994	409.29998

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])
```

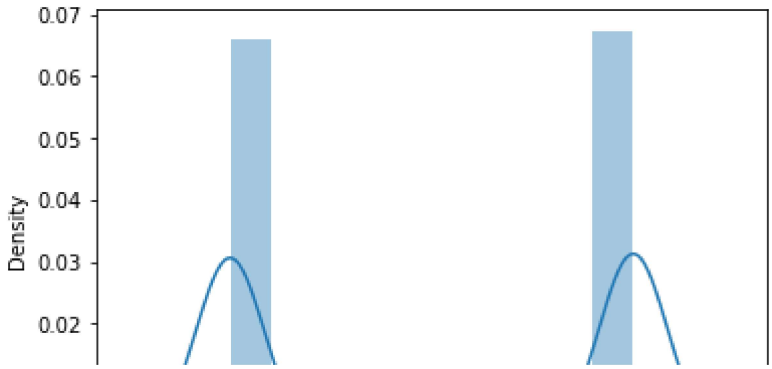
Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x155903d4910>
```



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

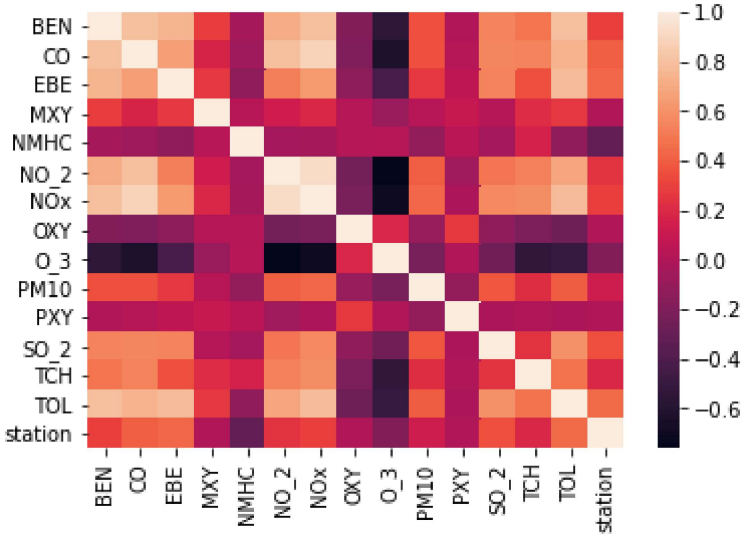
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
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]:
```
28078935.783117577
```

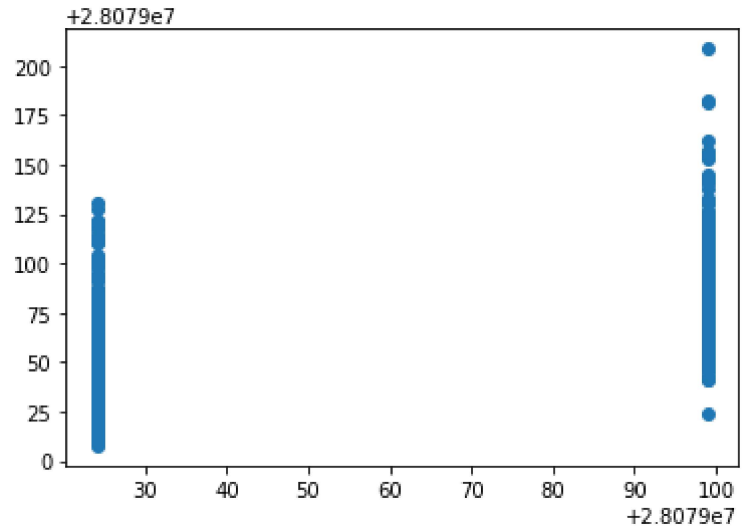
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-33.951146
CO	157.670196
EBE	16.975462
MXY	-9.952246
NMHC	-70.539657
NO_2	0.366696
NOx	-0.627102
OXY	31.374439
O_3	0.070589
PM10	-0.189305
PXY	-5.862070
SO_2	1.806854
TCH	46.117775
TOL	9.630995

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

Out[27]:
```
<matplotlib.collections.PathCollection at 0x1559e42a490>
```



ACCURACY

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

Out[28]:
```
0.433198509620901
```

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

Out[29]:
```
0.42446893633496463
```

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]:
```
0.41149988785696723
```

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

Out[33]:
```
0.4137335200197093
```

Accuracy(Lasso)

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]:
```
0.1824852605949907
```

ElasticNet

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

Out[36]:
```
0.17148732597413485
```

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([-0.        ,  0.16533918,  2.79654101, -1.31204197, -1.23138   ,
        0.12266477, -0.15169913,  0.44222465, -0.03083657, -0.14900935,
       -0.        ,  2.53825425,  0.        ,  7.01605851])
```

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

Out[39]:
```
28079026.405199617
```

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

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

Out[41]:
```
0.2270730230246586
```

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)))
```

```
31.05208047340624
1086.8894319433643
32.96800618695896
```

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]:
```
(6666, 14)
```

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

Out[46]:
```
(6666,)
```

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([28079024, 28079099], dtype=int64)
```

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

Out[53]:
```
0.8660366036603661
```

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

Out[54]:
```
0.0
```

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

Out[55]:
```
array([[0., 1.]])
```

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="acc
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]:
```
0.9316330904414916
```

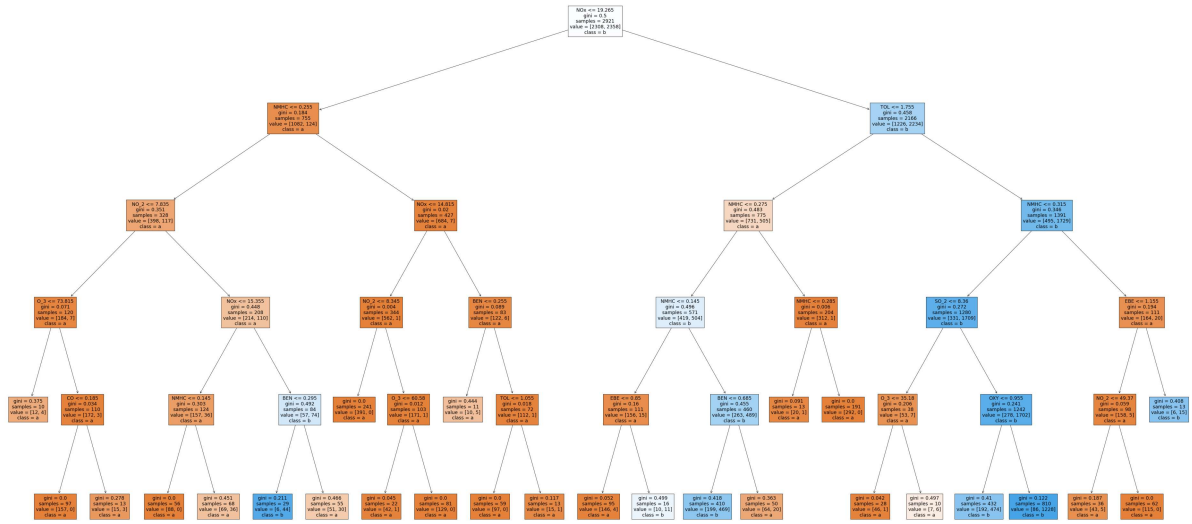
In [61]:
```
rfc_best=grid_search.best_estimator_
```

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(2219.318181818182, 1993.2, 'NOx <= 19.265\ngini = 0.5\nsamples = 2921\n
value = [2308, 2358]\nclass = b'),
 Text(1090.6363636363635, 1630.8000000000002, 'NMHC <= 0.255\ngini = 0.184\ns
amples = 755\nvalue = [1082, 124]\nclass = a'),
 Text(558.0, 1268.4, 'NO_2 <= 7.835\ngini = 0.351\nsamples = 328\nvalue = [39
8, 117]\nclass = a'),
 Text(202.9090909090909, 906.0, 'O_3 <= 73.815\ngini = 0.071\nsamples = 120\n
value = [184, 7]\nclass = a'),
 Text(101.45454545454545, 543.5999999999999, 'gini = 0.375\nsamples = 10\nval
ue = [12, 4]\nclass = a'),
 Text(304.3636363636364, 543.5999999999999, 'CO <= 0.185\ngini = 0.034\nsampl
es = 110\nvalue = [172, 3]\nclass = a'),
 Text(202.9090909090909, 181.19999999999982, 'gini = 0.0\nsamples = 97\nvalue
= [157, 0]\nclass = a'),
 Text(405.8181818181818, 181.19999999999982, 'gini = 0.278\nsamples = 13\nval
ue = [15, 3]\nclass = a'),
 Text(913.0909090909091, 906.0, 'NOx <= 15.355\ngini = 0.448\nsamples = 208\n
value = [214, 110]\nclass = a'),
 Text(710.1818181818181, 543.5999999999999, 'NMHC <= 0.145\ngini = 0.303\nsam
ples = 124\nvalue = [157, 36]\nclass = a'),
 Text(608.7272727272727, 181.19999999999982, 'gini = 0.0\nsamples = 56\nvalue
= [88, 0]\nclass = a'),
 Text(811.6363636363636, 181.19999999999982, 'gini = 0.451\nsamples = 68\nval
ue = [69, 36]\nclass = a'),
 Text(1116.0, 543.5999999999999, 'BEN <= 0.295\ngini = 0.492\nsamples = 84\nv
alue = [57, 74]\nclass = b'),
 Text(1014.5454545454545, 181.19999999999982, 'gini = 0.211\nsamples = 29\nva
lue = [6, 44]\nclass = b'),
 Text(1217.4545454545455, 181.19999999999982, 'gini = 0.466\nsamples = 55\nva
lue = [51, 30]\nclass = a'),
 Text(1623.2727272727273, 1268.4, 'NOx <= 14.815\ngini = 0.02\nsamples = 427
\nvalue = [684, 7]\nclass = a'),
 Text(1420.3636363636363, 906.0, 'NO_2 <= 8.345\ngini = 0.004\nsamples = 344
\nvalue = [562, 1]\nclass = a'),
 Text(1318.909090909091, 543.5999999999999, 'gini = 0.0\nsamples = 241\nvalue
= [391, 0]\nclass = a'),
 Text(1521.8181818181818, 543.5999999999999, 'O_3 <= 60.58\ngini = 0.012\nsam
ples = 103\nvalue = [171, 1]\nclass = a'),
 Text(1420.3636363636363, 181.19999999999982, 'gini = 0.045\nsamples = 22\nva
lue = [42, 1]\nclass = a'),
 Text(1623.2727272727273, 181.19999999999982, 'gini = 0.0\nsamples = 81\nvalu
e = [129, 0]\nclass = a'),
 Text(1826.1818181818182, 906.0, 'BEN <= 0.255\ngini = 0.089\nsamples = 83\nv
alue = [122, 6]\nclass = a'),
 Text(1724.7272727272727, 543.5999999999999, 'gini = 0.444\nsamples = 11\nval
ue = [10, 5]\nclass = a'),
 Text(1927.6363636363635, 543.5999999999999, 'TOL <= 1.055\ngini = 0.018\nsam
ples = 72\nvalue = [112, 1]\nclass = a'),
 Text(1826.1818181818182, 181.19999999999982, 'gini = 0.0\nsamples = 59\nvalu
e = [97, 0]\nclass = a'),
 Text(2029.090909090909, 181.19999999999982, 'gini = 0.117\nsamples = 13\nval
ue = [15, 1]\nclass = a'),
 Text(3348.0, 1630.8000000000002, 'TOL <= 1.755\ngini = 0.458\nsamples = 2166
\nvalue = [1226, 2234]\nclass = b'),
 Text(2790.0, 1268.4, 'NMHC <= 0.275\ngini = 0.483\nsamples = 775\nvalue = [7
31, 505]\nclass = a'),
 Text(2536.3636363636365, 906.0, 'NMHC <= 0.145\ngini = 0.496\nsamples = 571
```

```
\nvalue = [419, 504]\nclass = b'),
 Text(2333.4545454545455, 543.5999999999999, 'EBE <= 0.85\ngini = 0.16\nsampl
es = 111\nvalue = [156, 15]\nclass = a'),
 Text(2232.0, 181.19999999999982, 'gini = 0.052\nsamples = 95\nvalue = [146,
4]\nclass = a'),
 Text(2434.909090909091, 181.19999999999982, 'gini = 0.499\nsamples = 16\nval
ue = [10, 11]\nclass = b'),
 Text(2739.272727272727, 543.5999999999999, 'BEN <= 0.685\ngini = 0.455\nsamp
les = 460\nvalue = [263, 489]\nclass = b'),
 Text(2637.818181818182, 181.19999999999982, 'gini = 0.418\nsamples = 410\nva
lue = [199, 469]\nclass = b'),
 Text(2840.7272727272725, 181.19999999999982, 'gini = 0.363\nsamples = 50\nva
lue = [64, 20]\nclass = a'),
 Text(3043.6363636363635, 906.0, 'NMHC <= 0.285\ngini = 0.006\nsamples = 204
\nvalue = [312, 1]\nclass = a'),
 Text(2942.181818181818, 543.5999999999999, 'gini = 0.091\nsamples = 13\nvalu
e = [20, 1]\nclass = a'),
 Text(3145.090909090909, 543.5999999999999, 'gini = 0.0\nsamples = 191\nvalue
= [292, 0]\nclass = a'),
 Text(3906.0, 1268.4, 'NMHC <= 0.315\ngini = 0.346\nsamples = 1391\nvalue =
[495, 1729]\nclass = b'),
 Text(3550.909090909091, 906.0, 'SO_2 <= 8.36\ngini = 0.272\nsamples = 1280\n
value = [331, 1709]\nclass = b'),
 Text(3348.0, 543.5999999999999, 'O_3 <= 35.18\ngini = 0.206\nsamples = 38\nv
alue = [53, 7]\nclass = a'),
 Text(3246.5454545454545, 181.19999999999982, 'gini = 0.042\nsamples = 28\nva
lue = [46, 1]\nclass = a'),
 Text(3449.4545454545455, 181.19999999999982, 'gini = 0.497\nsamples = 10\nva
lue = [7, 6]\nclass = a'),
 Text(3753.818181818182, 543.5999999999999, 'OXY <= 0.955\ngini = 0.241\nsamp
les = 1242\nvalue = [278, 1702]\nclass = b'),
 Text(3652.3636363636365, 181.19999999999982, 'gini = 0.41\nsamples = 432\nva
lue = [192, 474]\nclass = b'),
 Text(3855.272727272727, 181.19999999999982, 'gini = 0.122\nsamples = 810\nva
lue = [86, 1228]\nclass = b'),
 Text(4261.090909090909, 906.0, 'EBE <= 1.155\ngini = 0.194\nsamples = 111\nv
alue = [164, 20]\nclass = a'),
 Text(4159.636363636364, 543.5999999999999, 'NO_2 <= 49.37\ngini = 0.059\nsam
ples = 98\nvalue = [158, 5]\nclass = a'),
 Text(4058.181818181818, 181.19999999999982, 'gini = 0.187\nsamples = 36\nval
ue = [43, 5]\nclass = a'),
 Text(4261.090909090909, 181.19999999999982, 'gini = 0.0\nsamples = 62\nvalue
= [115, 0]\nclass = a'),
 Text(4362.545454545454, 543.5999999999999, 'gini = 0.408\nsamples = 13\nvalu
e = [6, 15]\nclass = b')]
```



Conclusion

Accuracy

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

Out[63]: 0.42446893633496463

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

Out[64]: 0.4137335200197093

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

Out[65]: 0.1824852605949907

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

Out[66]: 0.2270730230246586

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

Out[67]: 0.8660366036603661

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

Out[68]: 0.9316330904414916

Random is suitable for this dataset