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_2003.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
0	2003-03-01 01:00:00	NaN	1.72	NaN	NaN	NaN	73.900002	316.299988	NaN	10.550000	55.209
1	2003-03-01 01:00:00	NaN	1.45	NaN	NaN	0.26	72.110001	250.000000	0.73	6.720000	52.389
2	2003-03-01 01:00:00	NaN	1.57	NaN	NaN	NaN	80.559998	224.199997	NaN	21.049999	63.240
3	2003-03-01 01:00:00	NaN	2.45	NaN	NaN	NaN	78.370003	450.399994	NaN	4.220000	67.839
4	2003-03-01 01:00:00	NaN	3.26	NaN	NaN	NaN	96.250000	479.100006	NaN	8.460000	95.779
...	...	...	...	...	...	...	...	...	...	...	
243979	2003-10-01 00:00:00	0.20	0.16	2.01	3.17	0.02	31.799999	32.299999	1.68	34.049999	7.380
243980	2003-10-01 00:00:00	0.32	0.08	0.36	0.72	NaN	10.450000	14.760000	1.00	34.610001	7.400
243981	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	34.639999	50.810001	NaN	32.160000	16.830
243982	2003-10-01 00:00:00	NaN	NaN	NaN	NaN	0.07	32.580002	41.020000	NaN	NaN	13.570
243983	2003-10-01 00:00:00	1.00	0.29	2.15	6.41	0.07	37.150002	56.849998	2.28	21.480000	12.350

243984 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: 33010 entries, 5 to 243983
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     33010 non-null  object 
 1   BEN      33010 non-null  float64
 2   CO       33010 non-null  float64
 3   EBE      33010 non-null  float64
 4   MXY      33010 non-null  float64
 5   NMHC     33010 non-null  float64
 6   NO_2     33010 non-null  float64
 7   NOx      33010 non-null  float64
 8   OXY      33010 non-null  float64
 9   O_3      33010 non-null  float64
 10  PM10     33010 non-null  float64
 11  PXY      33010 non-null  float64
 12  SO_2     33010 non-null  float64
 13  TCH      33010 non-null  float64
 14  TOL      33010 non-null  float64
 15  station  33010 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 4.3+ MB
```

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

Out[6]:

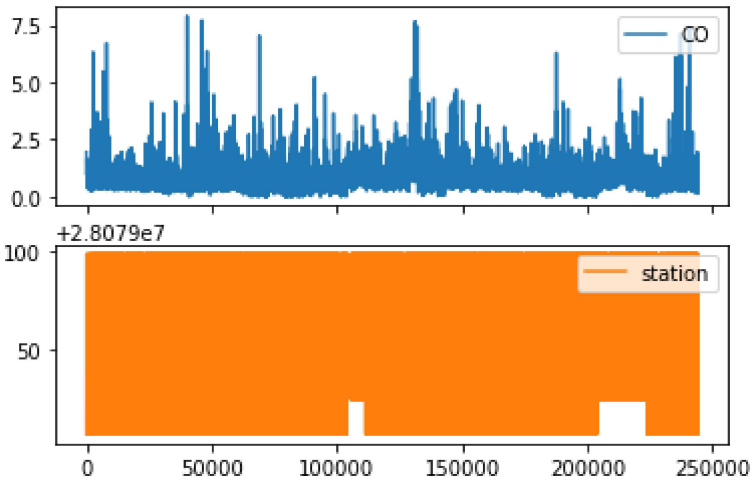
	CO	station
5	1.94	28079006
23	1.27	28079024
27	1.79	28079099
33	1.47	28079006
51	1.29	28079024
...	...	...
243955	0.41	28079099
243957	0.60	28079035
243961	0.82	28079006
243979	0.16	28079024
243983	0.29	28079099

33010 rows × 2 columns

Line chart

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

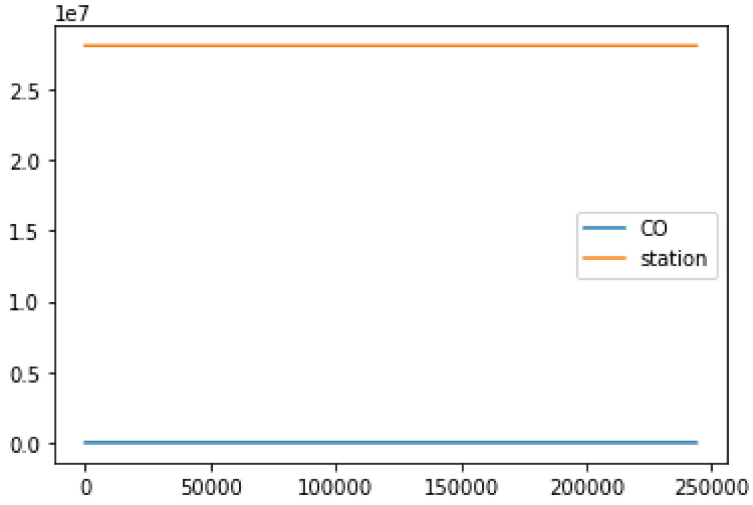
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

Out[8]: <AxesSubplot:>

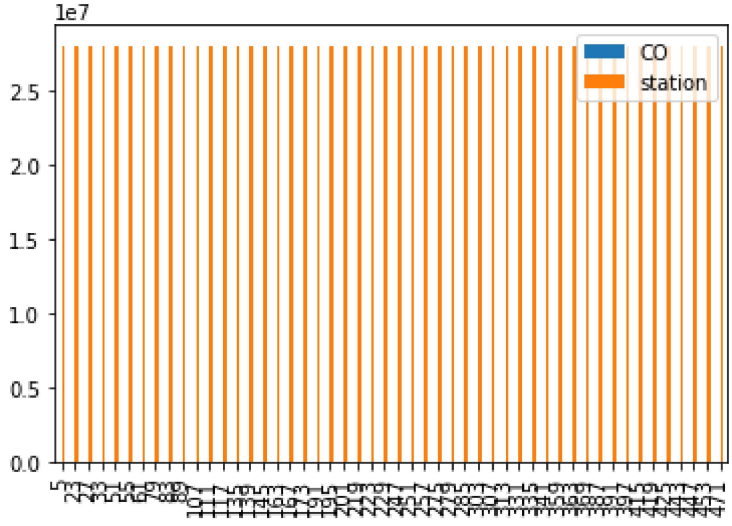


Bar chart

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

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

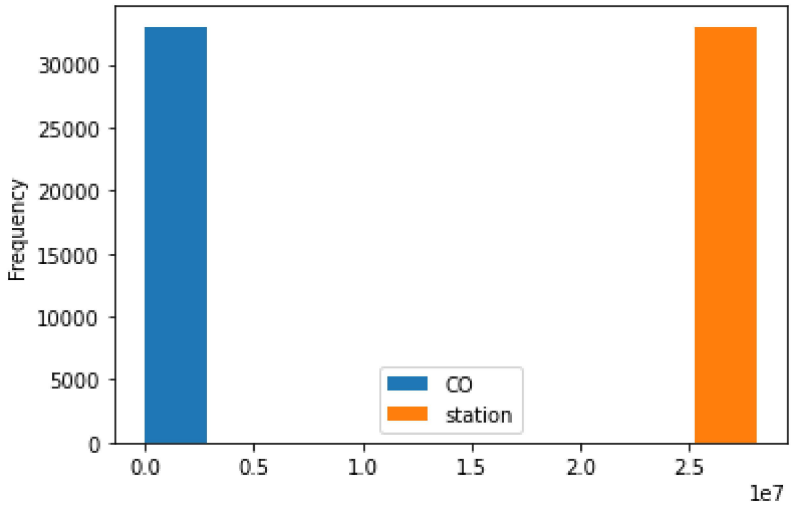
Out[10]: <AxesSubplot:>



Histogram

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

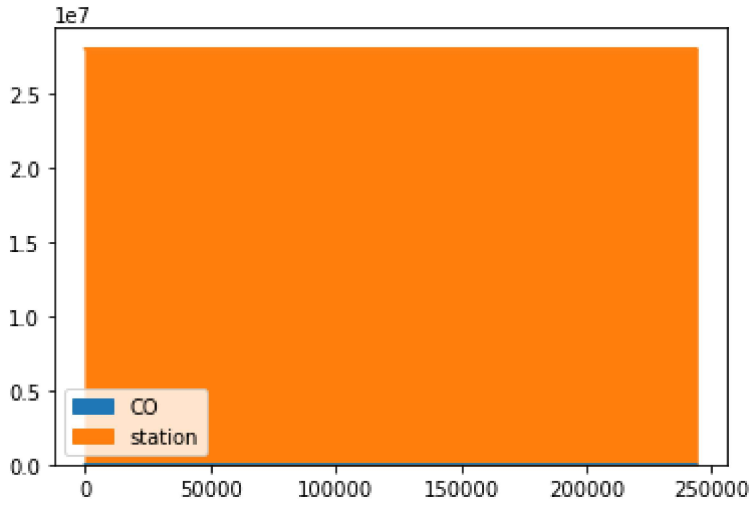
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

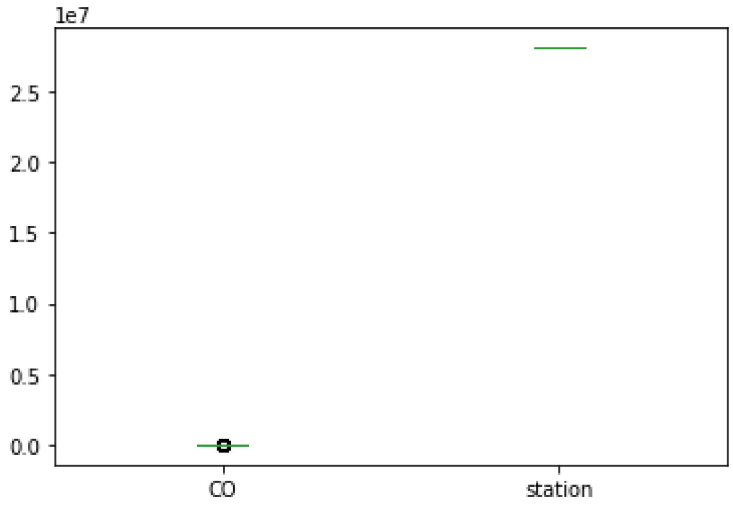
Out[12]: <AxesSubplot:>



Box chart

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

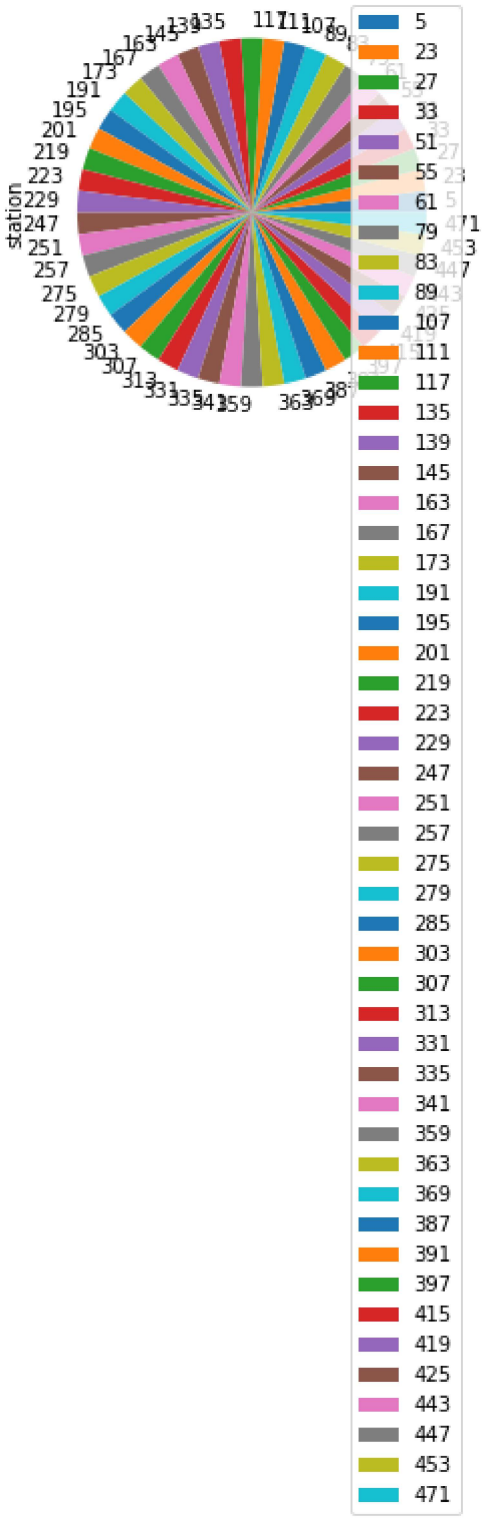
Out[13]: <AxesSubplot:>



Pie chart

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

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

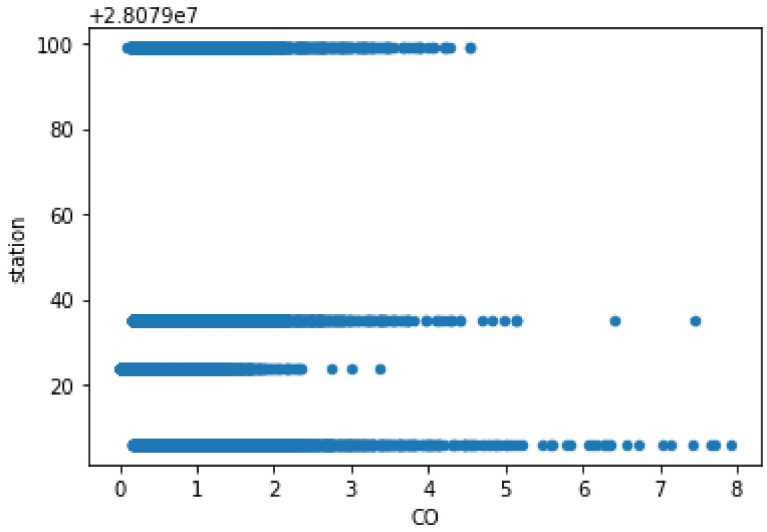


Scatter chart

Loading [MathJax]/extensions/Safe.js

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: 33010 entries, 5 to 243983
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     33010 non-null  object 
 1   BEN      33010 non-null  float64
 2   CO       33010 non-null  float64
 3   EBE      33010 non-null  float64
 4   MXY      33010 non-null  float64
 5   NMHC     33010 non-null  float64
 6   NO_2     33010 non-null  float64
 7   NOx      33010 non-null  float64
 8   OXY      33010 non-null  float64
 9   O_3      33010 non-null  float64
 10  PM10     33010 non-null  float64
 11  PXY      33010 non-null  float64
 12  SO_2     33010 non-null  float64
 13  TCH      33010 non-null  float64
```

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

Out[17]:

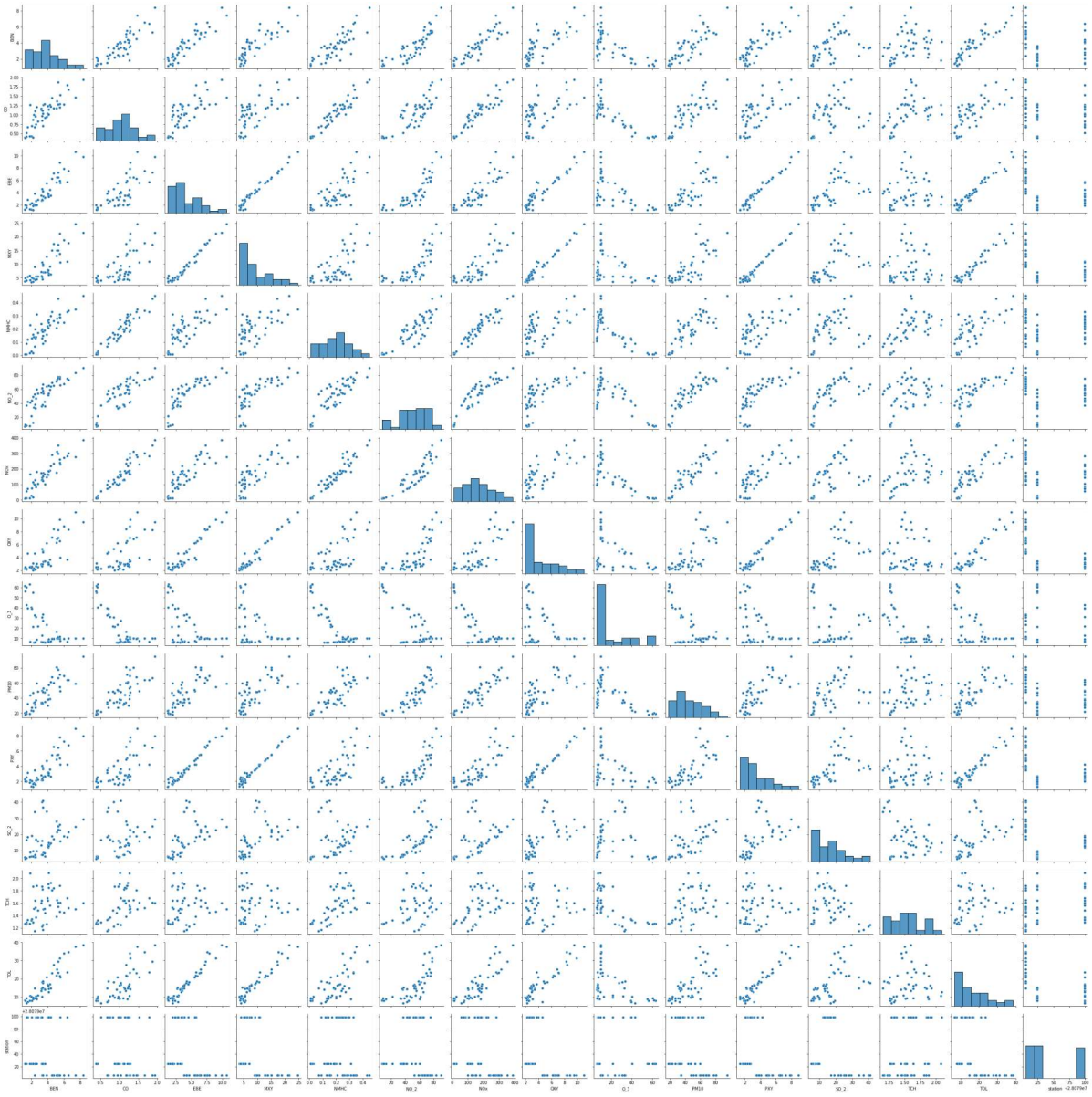
	BEN	CO	EBE	MXY	NMHC	NO_2	
count	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000	33010.000000	330
mean	2.192633	0.759868	2.639726	5.838414	0.137177	57.328049	1
std	2.064160	0.545999	2.825194	6.267296	0.127863	31.811082	1
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.900000	0.430000	1.010000	1.880000	0.060000	34.529999	
50%	1.610000	0.620000	1.890000	4.070000	0.110000	55.105000	
75%	2.810000	0.930000	3.300000	7.530000	0.170000	76.160004	1
max	66.389999	7.920000	92.589996	177.600006	2.180000	342.700012	12

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 0x2415e6ef0a0>
```

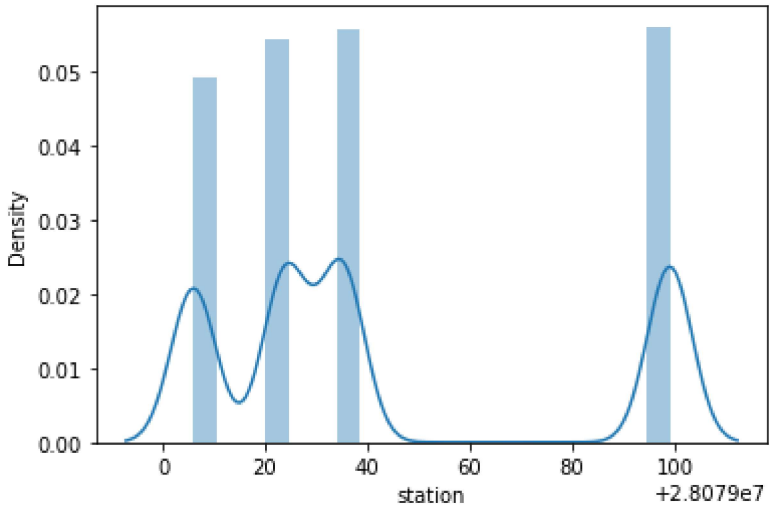


Loading [MathJax]/extensions/Safe.js

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

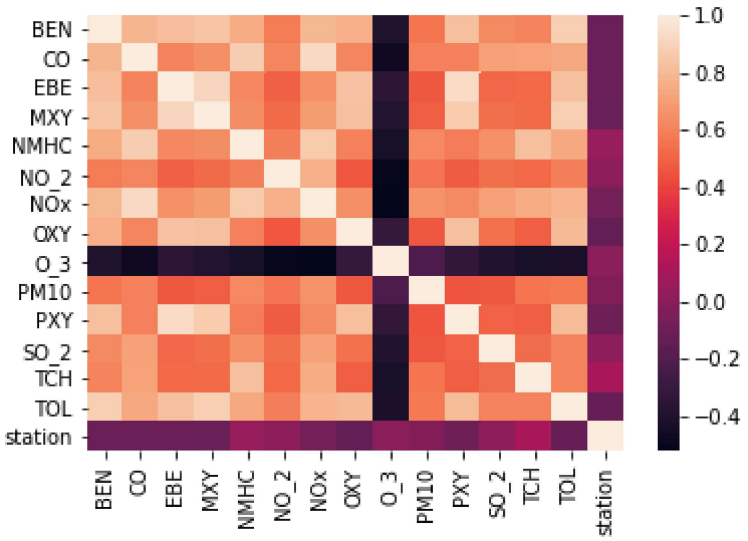
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  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

Loading [MathJax]/extensions/Safe.js

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]:
```
28079002.727508757
```

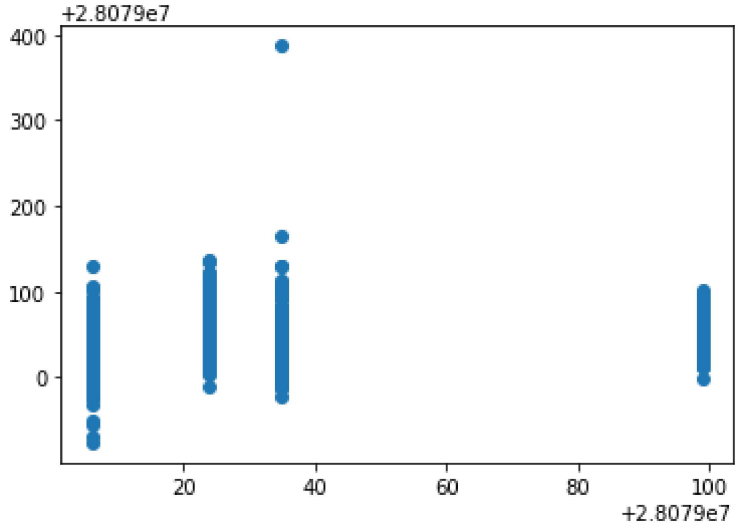
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	0.993103
CO	-40.119423
EBE	-1.445670
MXY	0.061327
NMHC	165.101240
NO_2	0.152386
NOx	-0.065513
OXY	-1.031079
O_3	-0.005881
PM10	-0.067257
PXY	1.534801
SO_2	0.826953
TCH	34.410525
TOL	-0.789745

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

Out[27]:
```
<matplotlib.collections.PathCollection at 0x2416e3ef670>
```



ACCURACY

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

Out[28]:
```
0.17022220278913502
```

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

Out[29]:
```
0.17786700200945338
```

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]:
```
0.17247606759373646
```

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

Out[33]:
```
0.17660041942965687
```

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]:
```
0.0327001321291992
```

Accuracy(Lasso)

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

Out[36]:
```
0.039278137420029724
```

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.        , -0.32390743,  0.15547184, -0.0599649 ,  0.13260659,
        0.14811645, -0.06713054, -1.04018259, -0.03276775,  0.06930004,
        0.12188199,  0.74093692,  1.59531052, -0.42632165])
```

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

Out[39]:
```
28079037.111688923
```

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

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

Out[41]:
```
0.05692164529101229
```

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)))
```

```
28.958528695753504
1166.5218235272396
34.15438220093052
```

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]:
```
(33010, 14)
```

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

Out[46]:
```
(33010,)
```

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]:
```
0.7584974250227204
```

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

Out[54]:
```
2.3306153265290618e-23
```

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

Out[55]:
```
array([[2.33061533e-23, 1.44436075e-55, 1.00000000e+00, 6.68457491e-16]])
```

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]:
```
0.7315966052757945
```

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

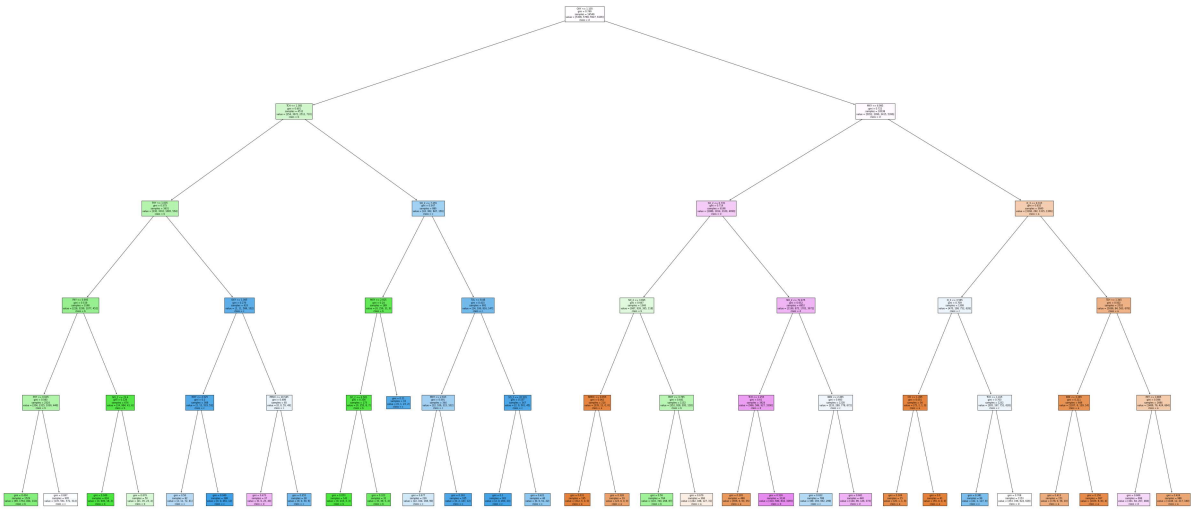
In [62]:
```
from sklearn.tree import plot_tree

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

Out[62]:
```
[Text(2185.5, 1993.2, 'OXY <= 1.105\ngini = 0.749\nsamples = 14549\nvalue =
[5306, 5768, 5927, 6106]\nclass = d'),
 Text(1097.4, 1630.8000000000002, 'TCH <= 1.385\ngini = 0.601\nsamples = 4511
\nvalue = [254, 3672, 2512, 710]\nclass = b'),
 Text(595.2, 1268.4, 'PXY <= 1.005\ngini = 0.575\nsamples = 3631\nvalue = [23
0, 3310, 1665, 554]\nclass = b'),
 Text(297.6, 906.0, 'PXY <= 0.995\ngini = 0.519\nsamples = 3198\nvalue = [22
8, 3299, 1077, 452]\nclass = b'),
 Text(148.8, 543.5999999999999, 'PXY <= 0.635\ngini = 0.583\nsamples = 2524\n
value = [204, 2315, 1036, 446]\nclass = b'),
 Text(74.4, 181.19999999999982, 'gini = 0.434\nsamples = 1529\nvalue = [69, 1
754, 464, 132]\nclass = b'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.697\nsamples = 995\nv
alue = [135, 561, 572, 314]\nclass = c'),
 Text(446.40000000000003, 543.5999999999999, 'NO_2 <= 29.4\ngini = 0.128\nsam
ples = 674\nvalue = [24, 984, 41, 6]\nclass = b'),
 Text(372.0, 181.19999999999982, 'gini = 0.046\nsamples = 624\nvalue = [3, 94
9, 18, 2]\nclass = b'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.679\nsamples = 50\nval
ue = [21, 35, 23, 4]\nclass = b'),
 Text(892.8000000000001, 906.0, 'OXY <= 1.005\ngini = 0.279\nsamples = 433\nv
alue = [2, 11, 588, 102]\nclass = c'),
 Text(744.0, 543.5999999999999, 'MXY <= 2.525\ngini = 0.2\nsamples = 368\nval
ue = [2, 11, 533, 53]\nclass = c'),
 Text(669.6, 181.19999999999982, 'gini = 0.56\nsamples = 82\nvalue = [2, 11,
72, 41]\nclass = c'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.049\nsamples = 286\nva
lue = [0, 0, 461, 12]\nclass = c'),
 Text(1041.6000000000001, 543.5999999999999, 'PM10 <= 20.545\ngini = 0.498\ns
amples = 65\nvalue = [0, 0, 55, 49]\nclass = c'),
 Text(967.2, 181.19999999999982, 'gini = 0.473\nsamples = 37\nvalue = [0, 0,
25, 40]\nclass = d'),
 Text(1116.0, 181.19999999999982, 'gini = 0.355\nsamples = 28\nvalue = [0, 0,
30, 9]\nclass = c'),
 Text(1599.6000000000001, 1268.4, 'SO_2 <= 7.255\ngini = 0.547\nsamples = 880
\nvalue = [24, 362, 847, 156]\nclass = c'),
 Text(1413.6000000000001, 906.0, 'MXY <= 2.925\ngini = 0.24\nsamples = 189\nv
alue = [0, 256, 31, 9]\nclass = b'),
 Text(1339.2, 543.5999999999999, 'SO_2 <= 6.585\ngini = 0.106\nsamples = 173
\nvalue = [0, 255, 8, 7]\nclass = b'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.053\nsamples = 142\nv
alue = [0, 216, 3, 3]\nclass = b'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.322\nsamples = 31\nva
lue = [0, 39, 5, 4]\nclass = b'),
 Text(1488.0, 543.5999999999999, 'gini = 0.21\nsamples = 16\nvalue = [0, 1, 2
3, 2]\nclass = c'),
 Text(1785.6000000000001, 906.0, 'TOL <= 8.48\ngini = 0.415\nsamples = 691\nv
alue = [24, 106, 816, 147]\nclass = c'),
 Text(1636.8000000000002, 543.5999999999999, 'MXY <= 2.615\ngini = 0.591\nsam
ples = 344\nvalue = [22, 106, 315, 102]\nclass = c'),
 Text(1562.4, 181.19999999999982, 'gini = 0.677\nsamples = 239\nvalue = [22,
104, 168, 90]\nclass = c'),
 Text(1711.2, 181.19999999999982, 'gini = 0.161\nsamples = 105\nvalue = [0,
2, 147, 12]\nclass = c'),
 Text(1934.4, 543.5999999999999, 'SO_2 <= 20.505\ngini = 0.157\nsamples = 347
\nvalue = [2, 0, 501, 45]\nclass = c'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.1\nsamples = 301\nval
```

```
ue = [2, 0, 450, 23]\nclass = c'),
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.421\nsamples = 46\nva
lue = [0, 0, 51, 22]\nclass = c'),
 Text(3273.6000000000004, 1630.8000000000002, 'MXY <= 6.985\ngini = 0.722\nsa
mples = 10038\nvalue = [5052, 2096, 3415, 5396]\nclass = d'),
 Text(2678.4, 1268.4, 'SO_2 <= 6.735\ngini = 0.714\nsamples = 6198\nvalue =
[1886, 1814, 2100, 4090]\nclass = d'),
 Text(2380.8, 906.0, 'SO_2 <= 3.695\ngini = 0.667\nsamples = 1346\nvalue = [6
87, 939, 395, 118]\nclass = b'),
 Text(2232.0, 543.5999999999999, 'NMHC <= 0.055\ngini = 0.062\nsamples = 214
\nvalue = [335, 11, 0, 0]\nclass = a'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.031\nsamples = 195\nv
alue = [312, 5, 0, 0]\nclass = a'),
 Text(2306.4, 181.19999999999982, 'gini = 0.328\nsamples = 19\nvalue = [23,
6, 0, 0]\nclass = a'),
 Text(2529.6000000000004, 543.5999999999999, 'MXY <= 3.765\ngini = 0.641\nsam
ples = 1132\nvalue = [352, 928, 395, 118]\nclass = b'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.56\nsamples = 764\nva
lue = [110, 740, 268, 87]\nclass = b'),
 Text(2604.0, 181.19999999999982, 'gini = 0.679\nsamples = 368\nvalue = [242,
188, 127, 31]\nclass = a'),
 Text(2976.0, 906.0, 'NO_2 <= 72.475\ngini = 0.652\nsamples = 4852\nvalue =
[1199, 875, 1705, 3972]\nclass = d'),
 Text(2827.2000000000003, 543.5999999999999, 'TCH <= 1.255\ngini = 0.61\nsamp
les = 3624\nvalue = [966, 586, 927, 3300]\nclass = d'),
 Text(2752.8, 181.19999999999982, 'gini = 0.295\nsamples = 488\nvalue = [656,
6, 93, 35]\nclass = a'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.526\nsamples = 3136\n
value = [310, 580, 834, 3265]\nclass = d'),
 Text(3124.8, 543.5999999999999, 'BEN <= 2.465\ngini = 0.693\nsamples = 1228
\nvalue = [233, 289, 778, 672]\nclass = c'),
 Text(3050.4, 181.19999999999982, 'gini = 0.632\nsamples = 788\nvalue = [89,
193, 652, 299]\nclass = c'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.661\nsamples = 440\nv
alue = [144, 96, 126, 373]\nclass = d'),
 Text(3868.8, 1268.4, 'O_3 <= 9.315\ngini = 0.632\nsamples = 3840\nvalue = [3
166, 282, 1315, 1306]\nclass = a'),
 Text(3571.2000000000003, 906.0, 'O_3 <= 3.565\ngini = 0.709\nsamples = 1308
\nvalue = [470, 198, 752, 628]\nclass = c'),
 Text(3422.4, 543.5999999999999, 'CO <= 1.495\ngini = 0.051\nsamples = 56\nva
lue = [75, 1, 1, 0]\nclass = a'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.169\nsamples = 15\nva
lue = [20, 1, 1, 0]\nclass = a'),
 Text(3496.8, 181.19999999999982, 'gini = 0.0\nsamples = 41\nvalue = [55, 0,
0, 0]\nclass = a'),
 Text(3720.0000000000005, 543.5999999999999, 'TCH <= 1.415\ngini = 0.703\nsam
ples = 1252\nvalue = [395, 197, 751, 628]\nclass = c'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.381\nsamples = 99\nva
lue = [42, 1, 127, 0]\nclass = c'),
 Text(3794.4, 181.19999999999982, 'gini = 0.708\nsamples = 1153\nvalue = [35
3, 196, 624, 628]\nclass = d'),
 Text(4166.400000000001, 906.0, 'TCH <= 1.345\ngini = 0.502\nsamples = 2532\n
value = [2696, 84, 563, 678]\nclass = a'),
 Text(4017.6000000000004, 543.5999999999999, 'EBE <= 3.285\ngini = 0.211\nsam
ples = 848\nvalue = [1207, 8, 139, 14]\nclass = a'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.413\nsamples = 151\nv
alue = [178, 0, 56, 10]\nclass = a'),
```

```
 Text(4092.0000000000005, 181.19999999999982, 'gini = 0.156\nsamples = 697\nv
alue = [1029, 8, 83, 4]\nclass = a'),
 Text(4315.200000000001, 543.5999999999999, 'PXY <= 3.865\ngini = 0.596\nsamp
les = 1684\nvalue = [1489, 76, 424, 664]\nclass = a'),
 Text(4240.8, 181.19999999999982, 'gini = 0.669\nsamples = 698\nvalue = [341,
64, 207, 484]\nclass = d'),
 Text(4389.6, 181.19999999999982, 'gini = 0.424\nsamples = 986\nvalue = [114
8, 12, 217, 180]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression:0.17786700200945338

Ridge Regression:0.17660041942965687

Lasso Regression:0.0327001321291992

ElasticNet Regression:0.05692164529101229

Logistic Regression:0.7584974250227204

Random Forest:0.7315966052757945

Logistic Regression is suitable for this dataset