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"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs(Dataset)\madrid_2010.
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
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
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24
...	...	...	...	...	...	...	...	...	...	...	
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
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

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 [11]:

```
data=df[['BEN', 'TOL', 'TCH']]
data
```

Out[11]:

	BEN	TOL	TCH
11	0.78	1.99	1.55
23	0.70	2.62	1.48
35	0.58	0.84	1.54
47	0.33	1.21	1.44
59	0.38	0.49	1.54
...	...	...	...
191879	0.60	2.94	1.34
191891	0.41	1.11	1.31
191903	0.57	2.95	1.36
191915	0.34	1.09	1.32
191927	0.43	2.80	1.38

6666 rows × 3 columns

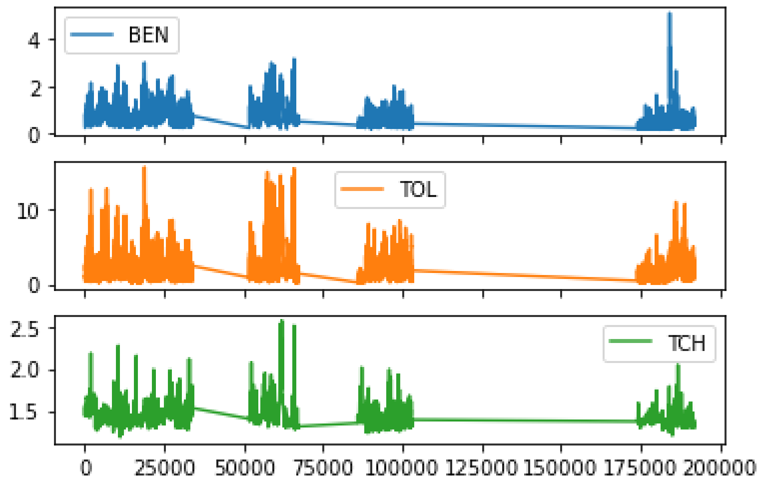
Line chart

In [12]:

```
data.plot.line(subplots=True)
```

Out[12]:

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



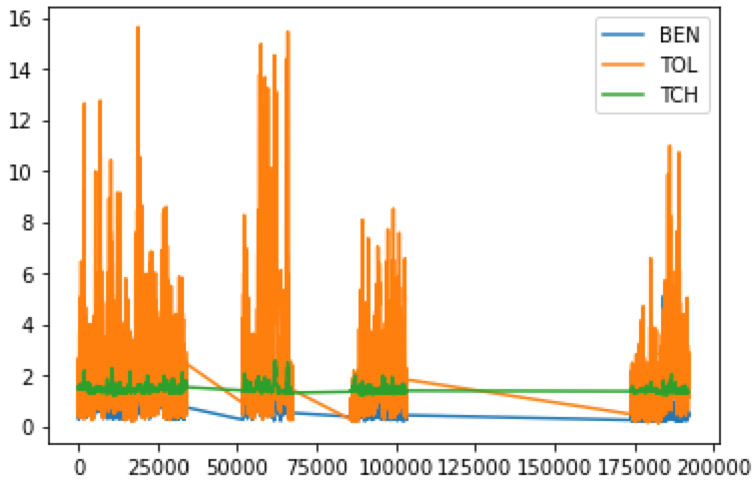
Line chart

In [13]:

```
data.plot.line()
```

Out[13]:

```
<AxesSubplot:>
```



Bar chart

In [14]:

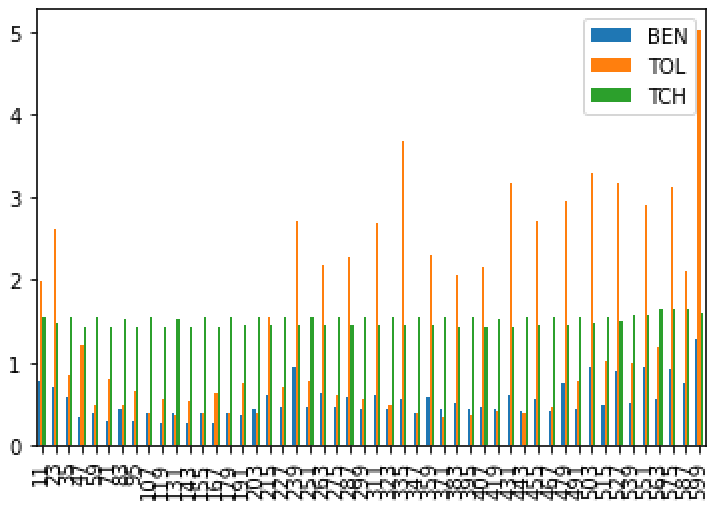
```
b=data[0:50]
```

In [15]:

```
b.plot.bar()
```

Out[15]:

```
<AxesSubplot:>
```



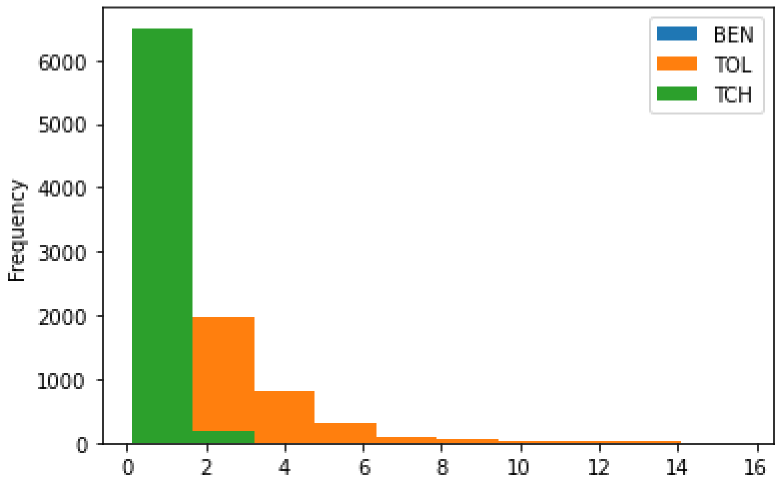
Histogram

In [16]:

```
data.plot.hist()
```

Out[16]:

<AxesSubplot:ylabel='Frequency'>



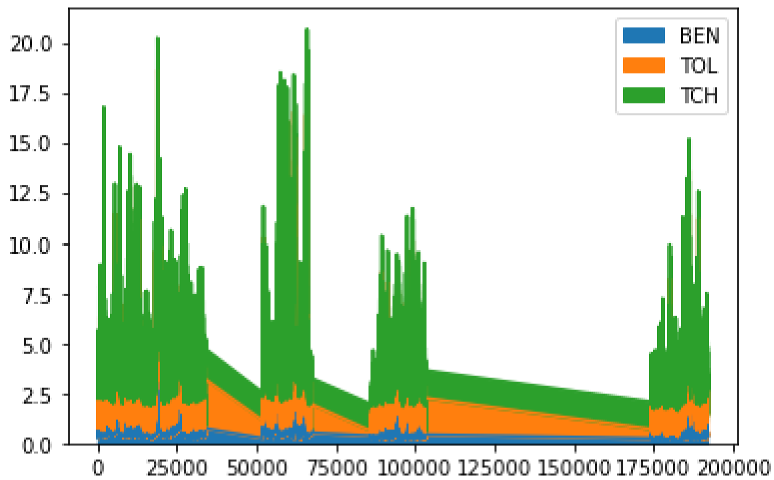
Area chart

In [17]:

```
data.plot.area()
```

Out[17]:

<AxesSubplot:>



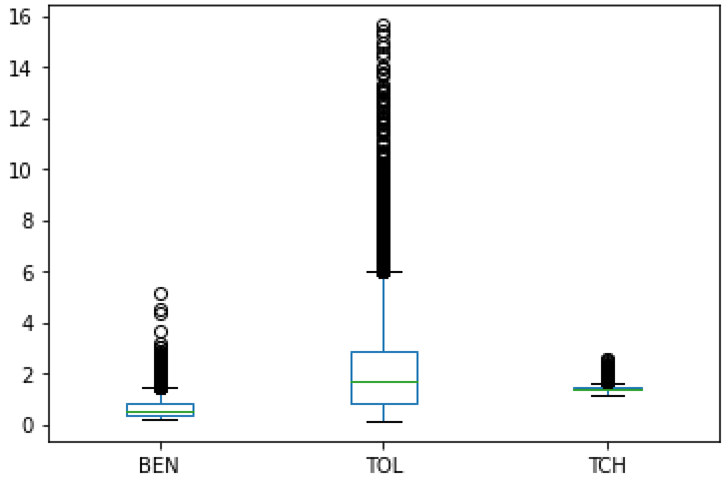
Box chart

In [18]:

```
data.plot.box()
```

Out[18]:

```
<AxesSubplot:>
```



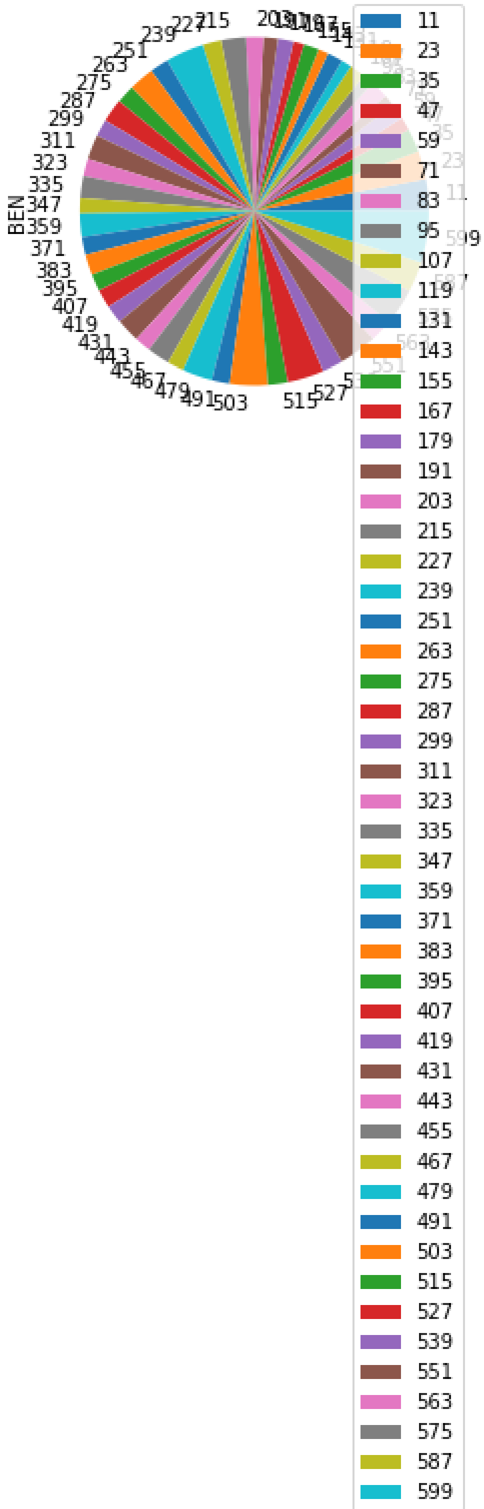
Pie chart

In [19]:

```
b.plot.pie(y='BEN' )
```

Out[19]:

```
<AxesSubplot:ylabel='BEN'>
```



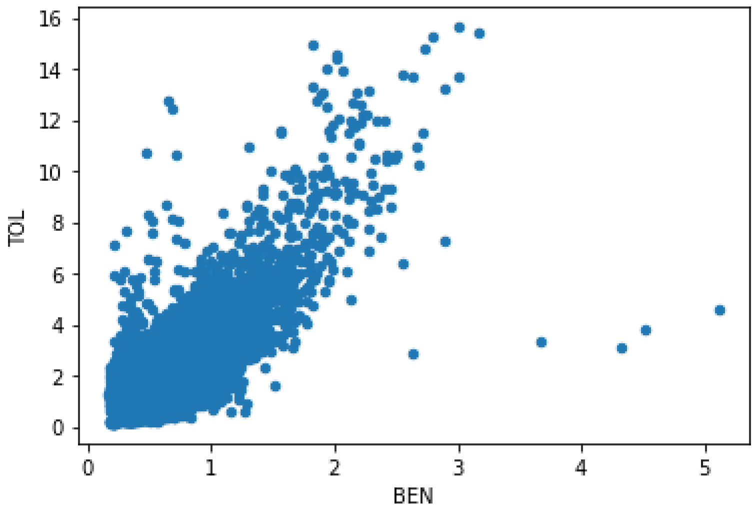
Scatter chart

In [20]:

```
data.plot.scatter(x='BEN' ,y='TOL')
```

Out[20]:

```
<AxesSubplot:xlabel='BEN', ylabel='TOL'>
```



In [21]:

```
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 [22]:

```
df.describe()
```

Out[22]:

	BEN	CO	EBE	MXY	NMHC	NO_2	
count	6666.000000	6666.000000	6666.000000	6666.000000	6666.000000	6666.000000	6666.0
mean	0.648425	0.296280	0.840585	0.839959	0.243378	33.888744	47.5
std	0.395346	0.133296	0.508031	0.382263	0.115730	23.465169	41.2
min	0.170000	0.090000	0.140000	0.110000	0.000000	1.290000	2.7
25%	0.380000	0.200000	0.470000	0.590000	0.180000	15.752500	19.4
50%	0.540000	0.260000	0.755000	1.000000	0.220000	29.320000	36.7
75%	0.810000	0.340000	1.000000	1.000000	0.280000	47.657500	62.1
max	5.110000	1.590000	5.190000	6.810000	0.930000	133.399994	409.2

In [23]:

```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

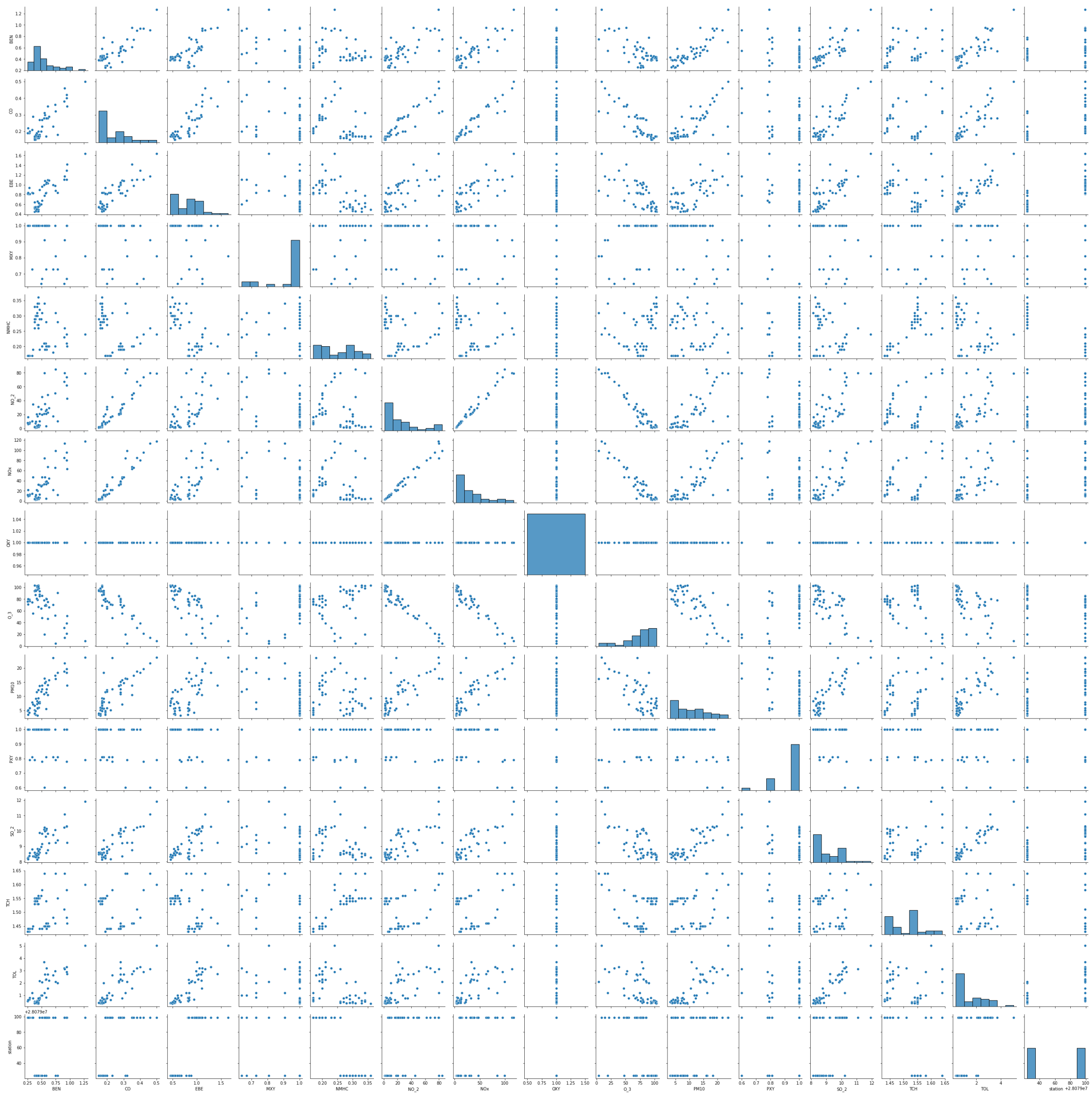
EDA AND VISUALIZATION

In [24]:

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

Out[24]:

```
<seaborn.axisgrid.PairGrid at 0x1e6db036850>
```



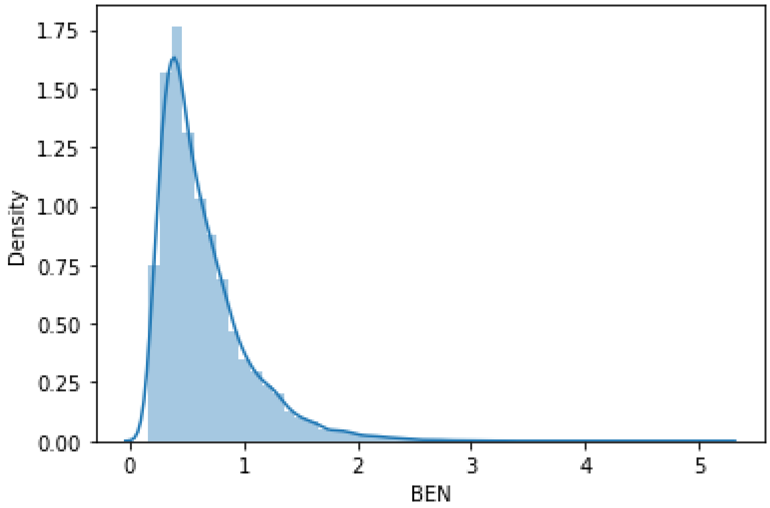
In [25]:

```
sns.distplot(df1['BEN'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[25]:

<AxesSubplot:xlabel='BEN', ylabel='Density'>

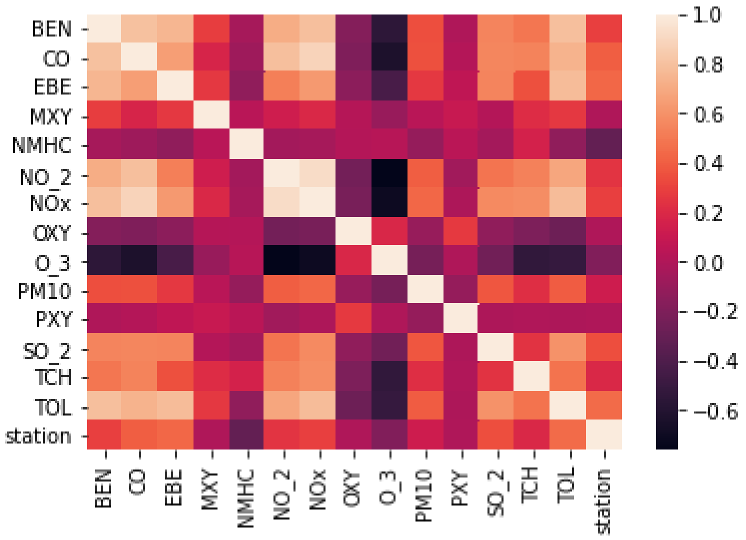


In [26]:

```
sns.heatmap(df1.corr())
```

Out[26]:

<AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [111]:

```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY','O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [112]:

```
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 [113]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[113]:

```
LinearRegression()
```

In [114]:

```
lr.intercept_
```

Out[114]:

```
28078940.864442065
```

In [115]:

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

Out[115]:

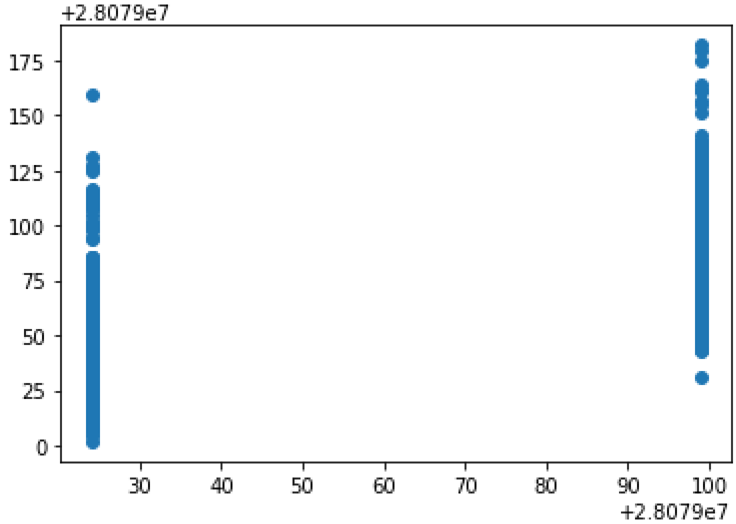
	Co-efficient
BEN	-34.991509
CO	169.287030
EBE	16.889120
MXY	-8.950140
NMHC	-75.485838
NO_2	0.312985
NOx	-0.626738
OXY	30.132175
O_3	0.076435
PM10	-0.162293
PXY	-4.783814
SO_2	1.781796
TCH	42.165548
TOL	9.666856

In [116]:

```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[116]:

<matplotlib.collections.PathCollection at 0x1e6ec61a4c0>



ACCURACY

In [117]:

```
lr.score(x_test,y_test)
```

Out[117]:

```
0.4179054524604561
```

In [118]:

```
lr.score(x_train,y_train)
```

Out[118]:

```
0.4315656379957823
```

Ridge and Lasso

In [119]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [120]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[120]:

```
Ridge(alpha=10)
```

Accuracy(Ridge)

In [121]:

```
rr.score(x_test,y_test)
```

Out[121]:

```
0.4056035229335304
```

In [122]:

```
rr.score(x_train,y_train)
```

Out[122]:

```
0.4193276233024624
```

In [123]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[123]:

```
Lasso(alpha=10)
```

In [124]:

```
la.score(x_test,y_test)
```

Out[124]:

```
0.18344759737444682
```

Accuracy(Lasso)

In [125]:

```
la.score(x_train,y_train)
```

Out[125]:

```
0.17829658999839737
```

Accuracy(Elastic Net)

In [126]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[126]:

```
ElasticNet()
```

In [127]:

```
en.coef_
```

Out[127]:

```
array([-0.        ,  0.19545881,  2.94342568, -1.19054764, -1.31923906,
        0.05983388, -0.12082628,  0.40926303, -0.01846025, -0.12404605,
       -0.        ,  2.57206502,  0.        ,  7.04094312])
```

In [128]:

```
en.intercept_
```

Out[128]:

```
28079025.73235495
```

In [129]:

```
prediction=en.predict(x_test)
```

In [130]:

```
en.score(x_test,y_test)
```

Out[130]:

```
0.23954371788037176
```

Evaluation Metrics

In [131]:

```
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)))
```

```
30.817089282844215
1069.284707566054
32.699919075833414
```

Logistic Regression

In [132]:

```
from sklearn.linear_model import LogisticRegression
```

In [133]:

```
feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

In [134]:

```
feature_matrix.shape
```

Out[134]:

```
(6666, 14)
```

In [135]:

```
target_vector.shape
```

Out[135]:

```
(6666,)
```

In [136]:

```
from sklearn.preprocessing import StandardScaler
```

In [137]:

```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [138]:

```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[138]:

```
LogisticRegression(max_iter=10000)
```

In [139]:

```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [140]:

```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079099]
```

In [141]:

```
logr.classes_
```

Out[141]:

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

In [142]:

```
logr.score(fs,target_vector)
```

Out[142]:

```
0.8660366036603661
```

In [143]:

```
logr.predict_proba(observation)[0][0]
```

Out[143]:

```
0.0
```

In [144]:

```
logr.predict_proba(observation)
```

Out[144]:

```
array([[0., 1.]])
```

Random Forest

In [145]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [146]:

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[146]:

```
RandomForestClassifier()
```

In [147]:

```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [148]:

```
from sklearn.model_selection import GridSearchCV
grid_search =GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[148]:

```
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 [149]:

```
grid_search.best_score_
```

Out[149]:

```
0.9303471924560651
```

In [150]:

```
rfc_best=grid_search.best_estimator_
```

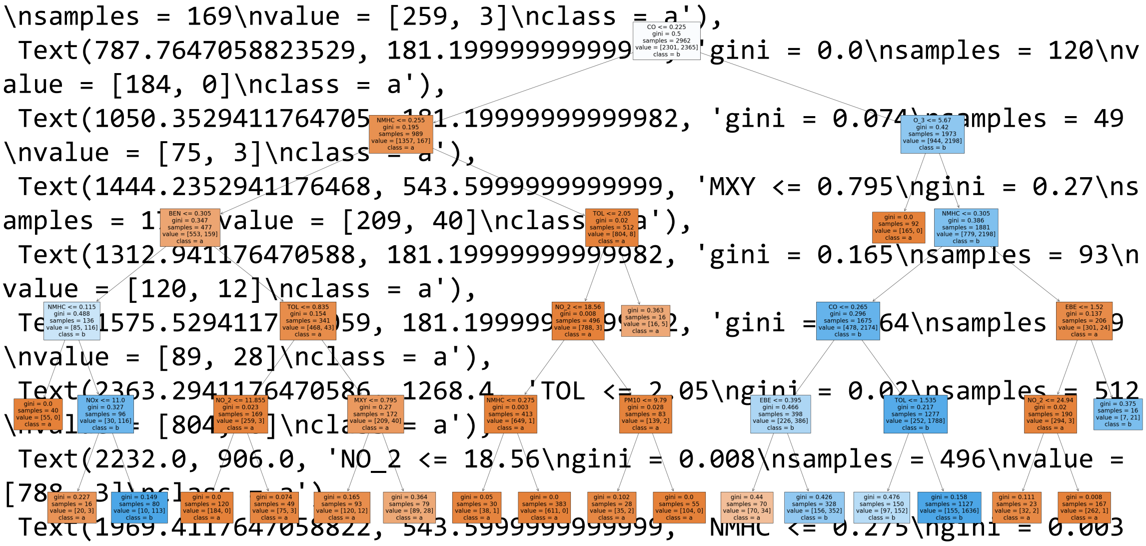
In [151]:

```
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','c','d'],f
```

Out[151]:

```
[Text(2576.6470588235293, 1993.2, 'CO <= 0.225\ngini = 0.5\nsamples = 2962
\nvalue = [2301, 2365]\nclass = b'),
 Text(1542.705882352941, 1630.8000000000002, 'NMHC <= 0.255\ngini = 0.195
\nsamples = 989\nvalue = [1357, 167]\nclass = a'),
 Text(722.1176470588234, 1268.4, 'BEN <= 0.305\ngini = 0.347\nsamples = 47
7\nvalue = [553, 159]\nclass = a'),
 Text(262.5882352941176, 906.0, 'NMHC <= 0.115\ngini = 0.488\nsamples = 13
6\nvalue = [85, 116]\nclass = b'),
 Text(131.2941176470588, 543.5999999999999, 'gini = 0.0\nsamples = 40\nval
ue = [55, 0]\nclass = a'),
 Text(393.88235294117646, 543.5999999999999, 'NOx <= 11.0\ngini = 0.327\ns
amples = 96\nvalue = [30, 116]\nclass = b'),
 Text(262.5882352941176, 181.19999999999982, 'gini = 0.227\nsamples = 16\n
value = [20, 3]\nclass = a'),
 Text(525.1764705882352, 181.19999999999982, 'gini = 0.149\nsamples = 80\n
value = [10, 113]\nclass = b'),
 Text(1181.6470588235293, 906.0, 'TOL <= 0.835\ngini = 0.154\nsamples = 34
1\nvalue = [468, 43]\nclass = a'),
 Text(919.0588235294117, 543.5999999999999, 'NO_2 <= 11.855\ngini = 0.023
\nsamples = 169\nvalue = [259, 3]\nclass = a'),
 Text(787.7647058823529, 181.19999999999982, 'gini = 0.0\nsamples = 120\nv
alue = [184, 0]\nclass = a'),
 Text(1050.3529411764705, 181.19999999999982, 'gini = 0.074\nsamples = 49
\nvalue = [75, 3]\nclass = a'),
 Text(1444.2352941176468, 543.5999999999999, 'MXY <= 0.795\ngini = 0.27\ns
amples = 172\nvalue = [209, 40]\nclass = a'),
 Text(1312.941176470588, 181.19999999999982, 'gini = 0.165\nsamples = 93\n
value = [120, 12]\nclass = a'),
 Text(1575.5294117647059, 181.19999999999982, 'gini = 0.364\nsamples = 79
\nvalue = [89, 28]\nclass = a'),
 Text(2363.2941176470586, 1268.4, 'TOL <= 2.05\ngini = 0.02\nsamples = 512
\nvalue = [804, 8]\nclass = a'),
 Text(2232.0, 906.0, 'NO_2 <= 18.56\ngini = 0.008\nsamples = 496\nvalue =
[788, 3]\nclass = a'),
 Text(1969.4117647058822, 543.5999999999999, 'NMHC <= 0.275\ngini = 0.003
\nsamples = 413\nvalue = [649, 1]\nclass = a'),
 Text(1838.1176470588234, 181.19999999999982, 'gini = 0.05\nsamples = 30\n
value = [38, 1]\nclass = a'),
 Text(2100.705882352941, 181.19999999999982, 'gini = 0.0\nsamples = 383\nv
alue = [611, 0]\nclass = a'),
 Text(2494.5882352941176, 543.5999999999999, 'PM10 <= 9.79\ngini = 0.028\n
samples = 83\nvalue = [139, 2]\nclass = a'),
 Text(2363.2941176470586, 181.19999999999982, 'gini = 0.102\nsamples = 28
\nvalue = [35, 2]\nclass = a'),
 Text(2625.882352941176, 181.19999999999982, 'gini = 0.0\nsamples = 55\nva
lue = [104, 0]\nclass = a'),
 Text(2494.5882352941176, 906.0, 'gini = 0.363\nsamples = 16\nvalue = [16,
5]\nclass = a'),
 Text(3610.588235294117, 1630.8000000000002, 'O_3 <= 5.67\ngini = 0.42\nsa
mples = 1973\nvalue = [944, 2198]\nclass = b'),
 Text(3479.2941176470586, 1268.4, 'gini = 0.0\nsamples = 92\nvalue = [165,
0]\nclass = a'),
 Text(3741.882352941176, 1268.4, 'NMHC <= 0.305\ngini = 0.386\nsamples = 1
881\nvalue = [779, 2198]\nclass = b'),
 Text(3282.3529411764703, 906.0, 'CO <= 0.265\ngini = 0.296\nsamples = 167
5\nvalue = [478, 2174]\nclass = b'),
 Text(3019.7647058823527, 543.5999999999999, 'EBE <= 0.395\ngini = 0.466\n
samples = 398\nvalue = [226, 386]\nclass = b'),
 Text(2888.4705882352937, 181.19999999999982, 'gini = 0.44\nsamples = 70\n
value = [70, 34]\nclass = a'),
 Text(3151.0588235294117, 181.19999999999982, 'gini = 0.426\nsamples = 328
```



Conclusion

Accuracy

Linear Regression:0.4315656379957823

Ridge Regression:0.4193276233024624

Lasso Regression:0.17829658999839737

ElasticNet Regression:0.23954371788037476

Logistic Regression:0.8660366036603661

Random Forest:0.9303471924560651

Random Forest is suitable for this dataset

```
\nvalue = [156, 352]\nclass = b'),
 Text(3544.941176470588, 543.5999999999999, 'TOL <= 1.535\ngini = 0.217\ns
amples = 1277\nvalue = [252, 1788]\nclass = b'),
 Text(3413.6470588235293, 181.19999999999982, 'gini = 0.476\nsamples = 150
\nvalue = [97, 152]\nclass = b'),
 Text(3676.235294117647, 181.19999999999982, 'gini = 0.158\nsamples = 1127
\nvalue = [155, 1636]\nclass = b'),
 Text(4201.411764705882, 906.0, 'EBE <= 1.52\ngini = 0.137\nsamples = 206
\nvalue = [301, 24]\nclass = a'),
 Text(4070.117647058823, 543.5999999999999, 'NO_2 <= 24.94\ngini = 0.02\ns
amples = 190\nvalue = [294, 3]\nclass = a'),
```