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_2002.
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	P
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.99(
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	20.98(
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.44(
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.18(
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.33(
...	...	...	...	...	...	...	...	...	...	...	
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.75(
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.38(
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	9.64(
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	I
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	29.24(

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 [11]:

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

Out[11]:

	BEN	TOL	PXY
1	1.93	10.98	2.53
5	3.19	15.60	2.98
22	2.02	7.32	1.48
24	3.02	11.42	2.18
26	2.02	10.60	2.45
...	...	...	...
217269	1.24	4.45	0.94
217271	3.13	15.10	3.40
217273	2.50	16.65	3.60
217293	1.37	4.33	0.94
217295	3.11	15.51	3.35

32381 rows × 3 columns

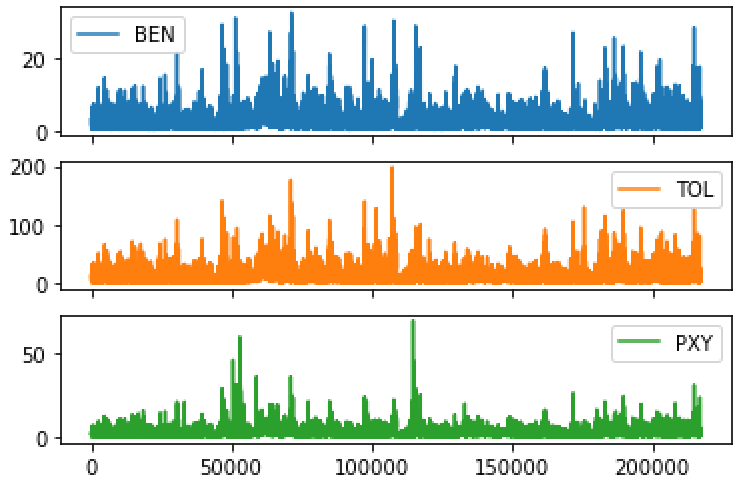
Line chart

In [12]:

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

Out[12]:

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



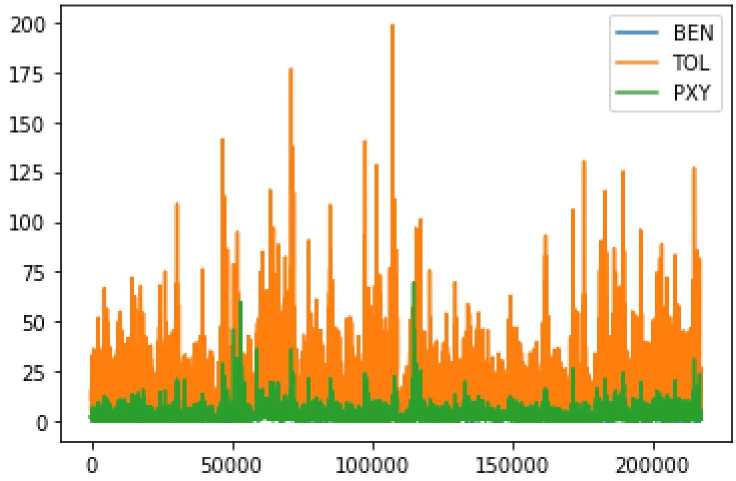
Line chart

In [13]:

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

Out[13]:

```
<AxesSubplot:>
```



Bar chart

In [14]:

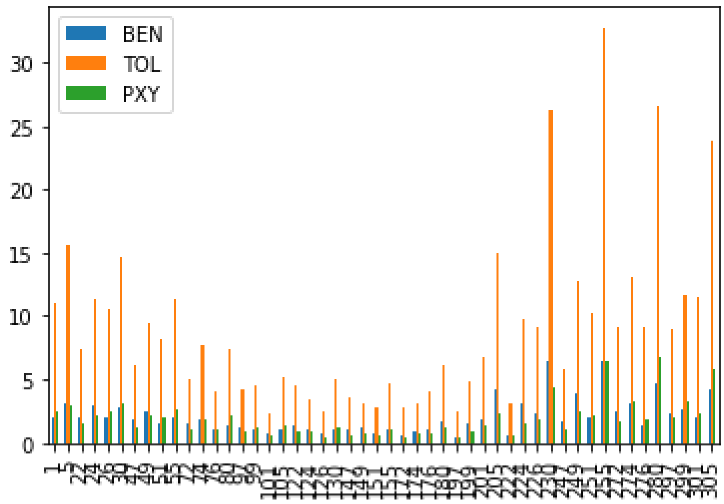
```
b=data[0:50]
```

In [15]:

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

Out[15]:

```
<AxesSubplot:>
```



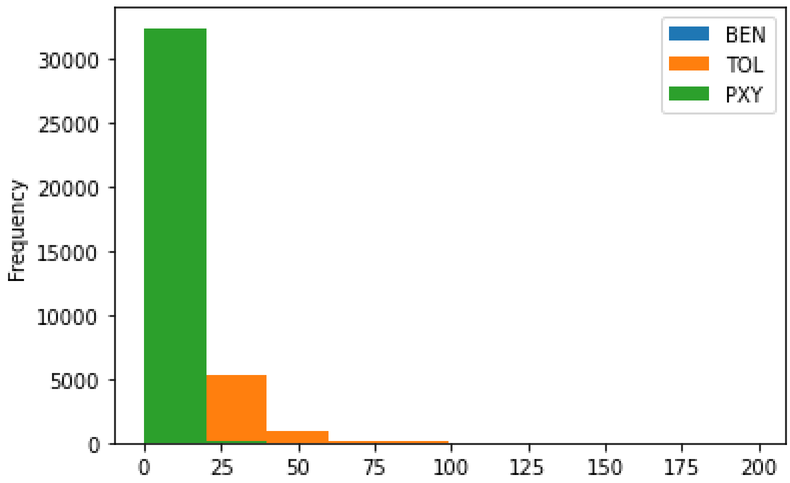
Histogram

In [16]:

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

Out[16]:

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



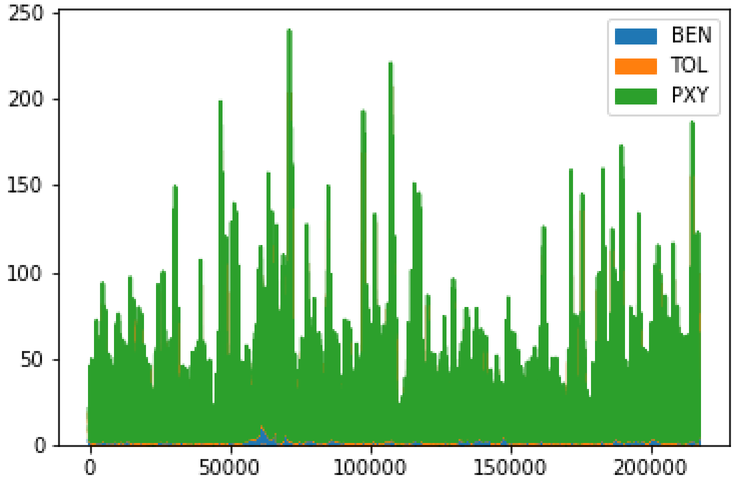
Area chart

In [17]:

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

Out[17]:

```
<AxesSubplot:>
```



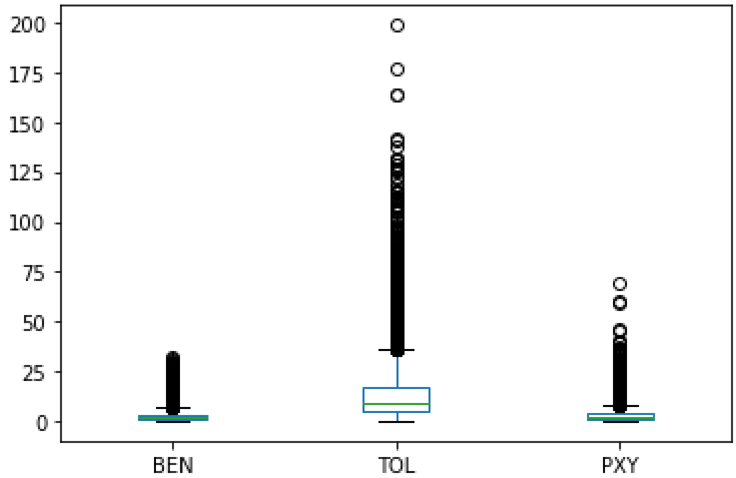
Box chart

In [18]:

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

Out[18]:

<AxesSubplot:>



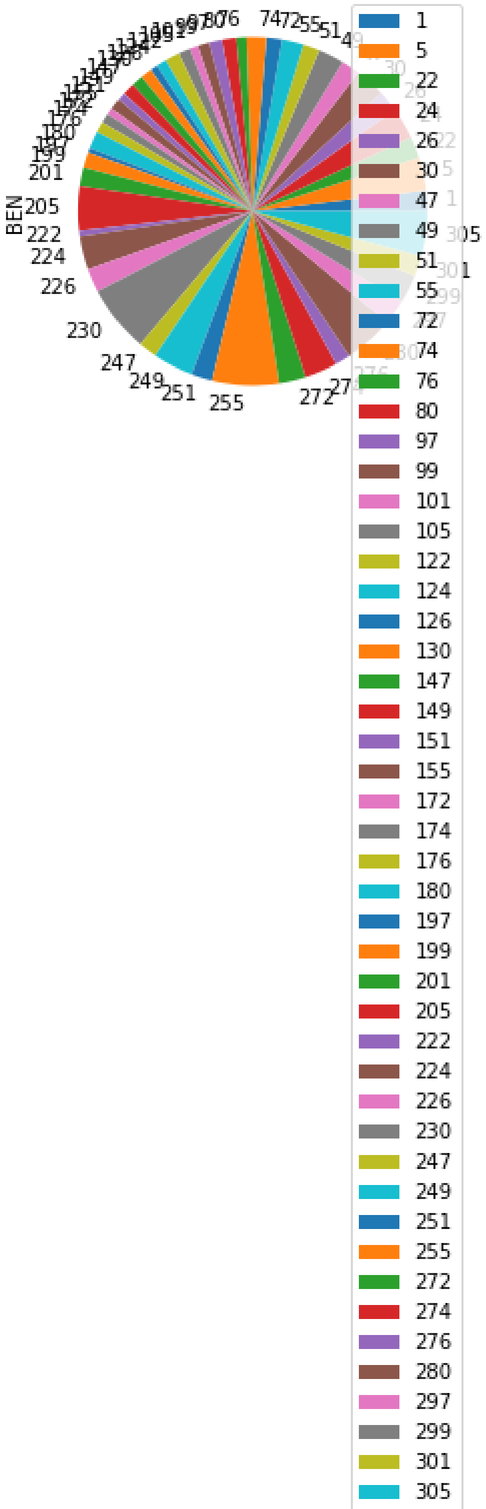
Pie chart

In [20]:

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

Out[20]:

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



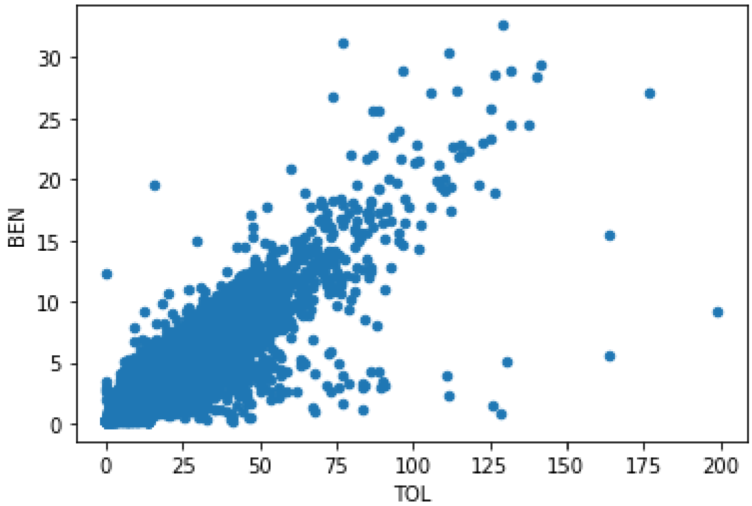
Scatter chart

In [22]:

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

Out[22]:

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



In [23]:

```
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 [24]:

```
df.describe()
```

Out[24]:

	BEN	CO	EBE	MXY	NMHC	NO_2
count	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000	32381.000000
mean	2.479155	0.787323	2.914004	7.013636	0.155827	58.936796
std	2.280959	0.610810	2.667881	6.774365	0.135731	31.472733
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
max	32.660000	8.460000	41.740002	99.879997	2.700000	263.600006

In [25]:

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

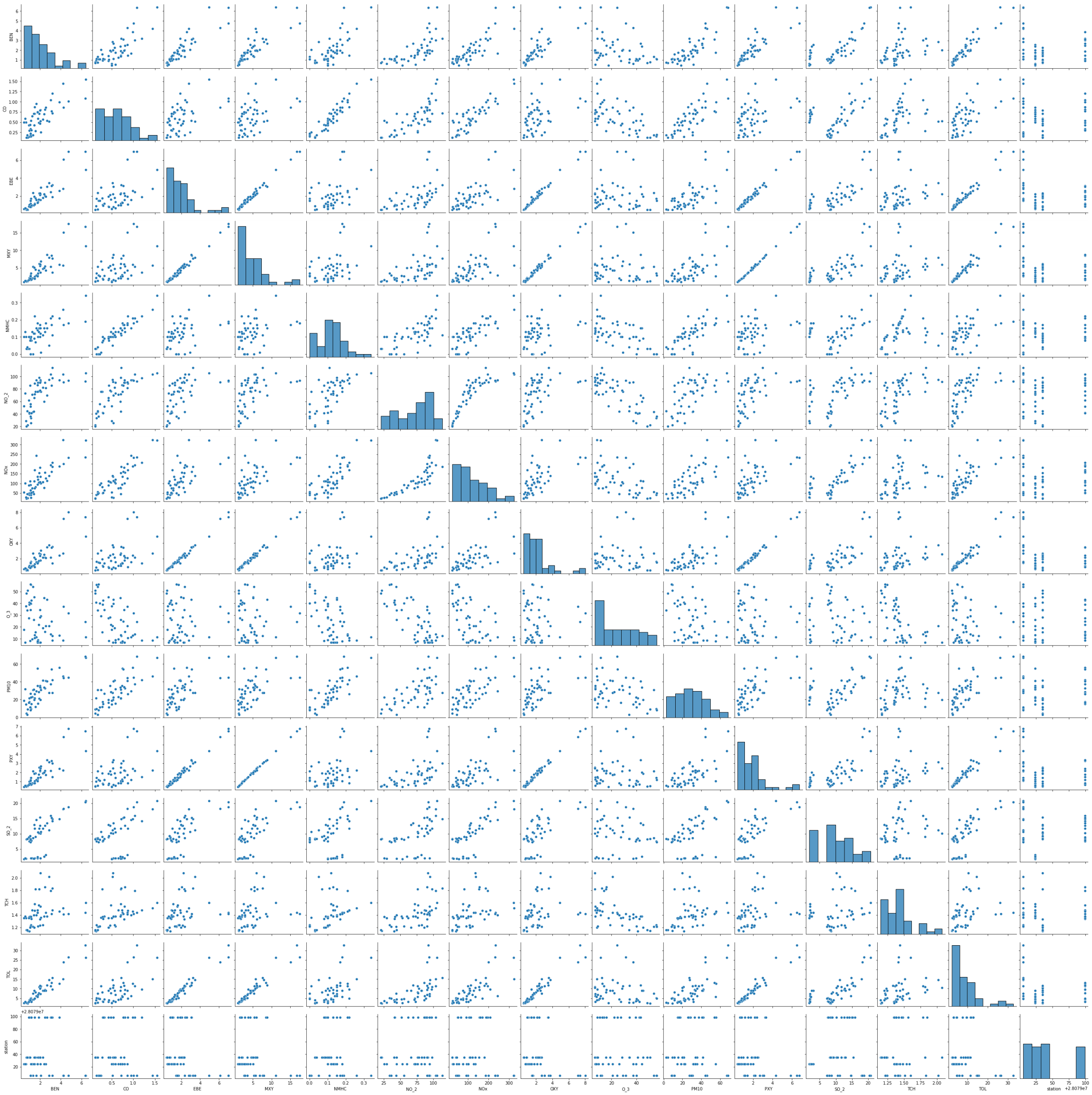
EDA AND VISUALIZATION

In [26]:

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

Out[26]:

<seaborn.axisgrid.PairGrid at 0x11456bcdd30>



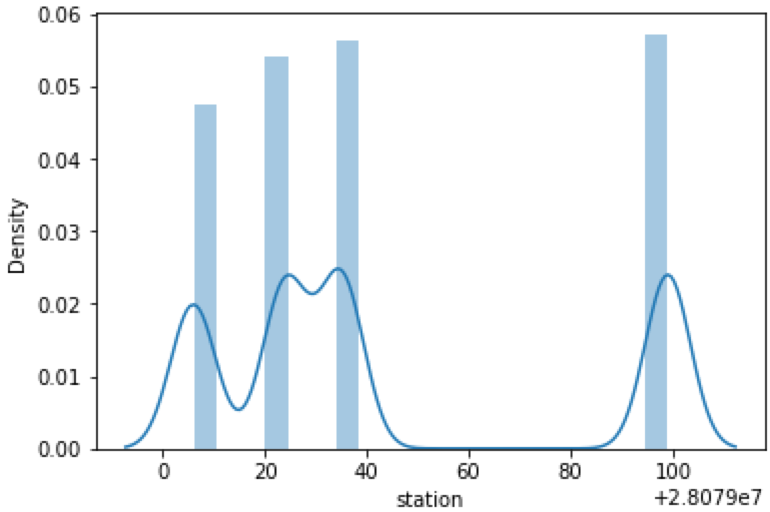
In [27]:

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

```
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[27]:

```
<AxesSubplot:xlabel='station', ylabel='Density'>
```

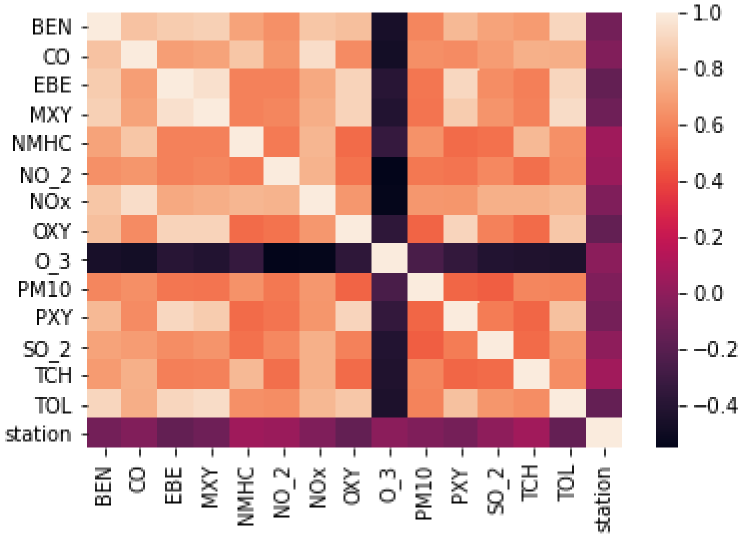


In [28]:

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

Out[28]:

```
<AxesSubplot:>
```



TO TRAIN THE MODEL AND MODEL BULDING

In [29]:

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

In [30]:

```
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 [31]:

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

Out[31]:

```
LinearRegression()
```

In [32]:

```
lr.intercept_
```

Out[32]:

```
28078991.39595005
```

In [33]:

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

Out[33]:

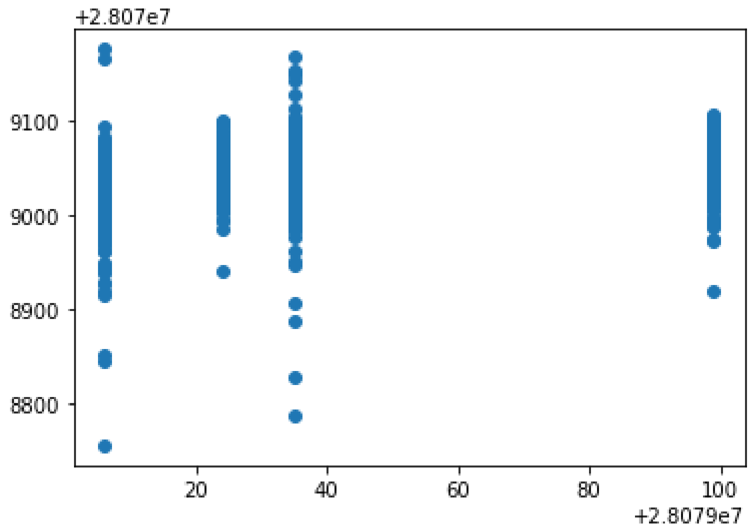
	Co-efficient
BEN	1.974642
CO	-13.144500
EBE	-12.590009
MXY	4.183719
NMHC	80.220883
NO_2	0.254409
NOx	-0.094810
OXY	-4.793770
O_3	-0.039461
PM10	-0.114979
PXY	8.165716
SO_2	0.551567
TCH	41.115474
TOL	-1.451087

In [34]:

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

Out[34]:

```
<matplotlib.collections.PathCollection at 0x11467f38940>
```



ACCURACY

In [35]:

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

Out[35]:

```
0.19727401561681268
```

In [36]:

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

Out[36]:

```
0.19864042035800789
```

Ridge and Lasso

In [37]:

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

In [38]:

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

Out[38]:

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

Accuracy(Ridge)

In [39]:

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

Out[39]:

```
0.1961181197383136
```

In [40]:

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

Out[40]:

```
0.1984498616765833
```

In [41]:

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

Out[41]:

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

In [42]:

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

Out[42]:

```
0.059286006322965545
```

Accuracy(Lasso)

In [43]:

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

Out[43]:

```
0.05399322398318607
```

Accuracy(Elastic Net)

In [44]:

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

Out[44]:

```
ElasticNet()
```

In [45]:

```
en.coef_
```

Out[45]:

```
array([ 0.90807009,  0.        , -3.03778728,  1.50214494,  0.18520451,
        0.23397266, -0.0264939 , -2.25820265, -0.03191645,  0.00659687,
        2.32046868,  0.36749735,  1.03444336, -1.15987874])
```

In [46]:

```
en.intercept_
```

Out[46]:

```
28079038.682628848
```

In [47]:

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

In [48]:

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

Out[48]:

```
0.09587022840432957
```

Evaluation Metrics

In [49]:

```
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)))
```

```
28.716055972336175
1131.179666418981
33.633014530650996
```

Logistic Regression

In [50]:

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

In [51]:

```
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 [52]:

```
feature_matrix.shape
```

Out[52]:

```
(32381, 14)
```

In [53]:

```
target_vector.shape
```

Out[53]:

```
(32381,)
```

In [54]:

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

In [55]:

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

In [56]:

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

Out[56]:

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

In [57]:

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

In [58]:

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

```
[28079035]
```

In [59]:

```
logr.classes_
```

Out[59]:

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

In [60]:

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

Out[60]:

```
0.8480899292795158
```

In [61]:

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

Out[61]:

```
2.5638972732451705e-10
```

In [62]:

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

Out[62]:

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

Random Forest

In [63]:

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

In [64]:

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

Out[64]:

```
RandomForestClassifier()
```

In [65]:

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

In [66]:

```
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[66]:

```
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 [67]:

```
grid_search.best_score_
```

Out[67]:

```
0.7753022147710227
```

In [68]:

```
rfc_best=grid_search.best_estimator_
```

In [69]:

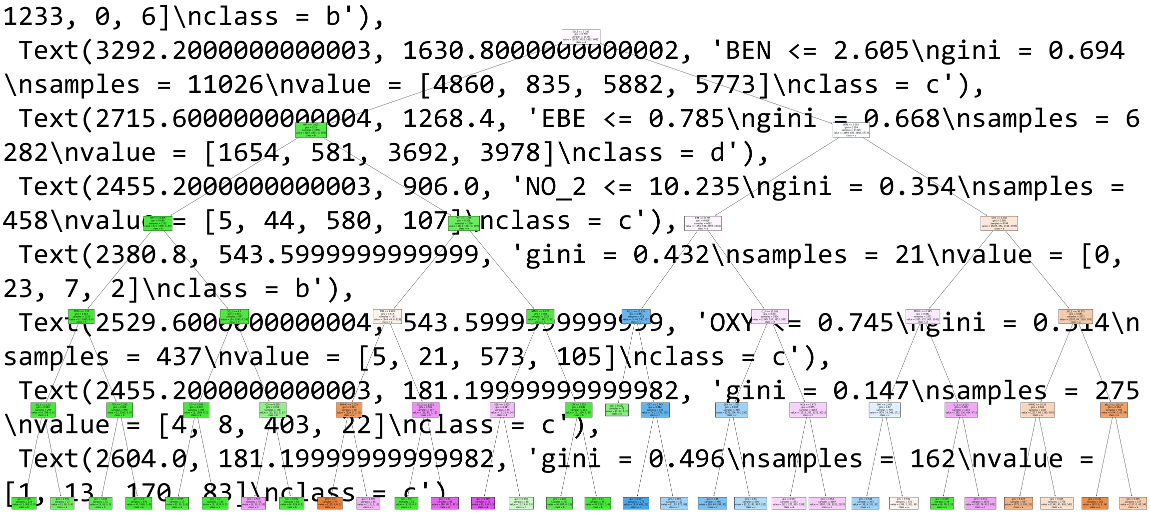
```
from sklearn.tree import plot_tree

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

Out[69]:

```
[Text(2241.3, 1993.2, 'SO_2 <= 5.785\ngini = 0.749\nsamples = 14384\nvalue
= [5027, 5726, 5882, 6031]\nclass = d'),
 Text(1190.4, 1630.8000000000002, 'OXY <= 1.005\ngini = 0.15\nsamples = 33
58\nvalue = [167, 4891, 0, 258]\nclass = b'),
 Text(595.2, 1268.4, 'OXY <= 0.805\ngini = 0.046\nsamples = 2132\nvalue =
[21, 3300, 0, 59]\nclass = b'),
 Text(297.6, 906.0, 'PM10 <= 7.23\ngini = 0.013\nsamples = 1235\nvalue =
[7, 1960, 0, 6]\nclass = b'),
 Text(148.8, 543.5999999999999, 'SO_2 <= 4.52\ngini = 0.048\nsamples = 250
\nvalue = [4, 398, 0, 6]\nclass = b'),
 Text(74.4, 181.19999999999982, 'gini = 0.005\nsamples = 223\nvalue = [1,
362, 0, 0]\nclass = b'),
 Text(223.20000000000002, 181.19999999999982, 'gini = 0.338\nsamples = 27
\nvalue = [3, 36, 0, 6]\nclass = b'),
 Text(446.40000000000003, 543.5999999999999, 'TCH <= 1.205\ngini = 0.004\n
samples = 985\nvalue = [3, 1562, 0, 0]\nclass = b'),
 Text(372.0, 181.19999999999982, 'gini = 0.185\nsamples = 15\nvalue = [3,
26, 0, 0]\nclass = b'),
 Text(520.8000000000001, 181.19999999999982, 'gini = 0.0\nsamples = 970\nv
alue = [0, 1536, 0, 0]\nclass = b'),
 Text(892.8000000000001, 906.0, 'SO_2 <= 4.7\ngini = 0.091\nsamples = 897
\nvalue = [14, 1340, 0, 53]\nclass = b'),
 Text(744.0, 543.5999999999999, 'CO <= 0.095\ngini = 0.002\nsamples = 761
\nvalue = [1, 1189, 0, 0]\nclass = b'),
 Text(669.6, 181.19999999999982, 'gini = 0.095\nsamples = 15\nvalue = [1,
19, 0, 0]\nclass = b'),
 Text(818.4000000000001, 181.19999999999982, 'gini = 0.0\nsamples = 746\nv
alue = [0, 1170, 0, 0]\nclass = b'),
 Text(1041.6000000000001, 543.5999999999999, 'CO <= 0.295\ngini = 0.453\ns
amples = 136\nvalue = [13, 151, 0, 53]\nclass = b'),
 Text(967.2, 181.19999999999982, 'gini = 0.525\nsamples = 54\nvalue = [9,
22, 0, 52]\nclass = d'),
 Text(1116.0, 181.19999999999982, 'gini = 0.072\nsamples = 82\nvalue = [4,
129, 0, 1]\nclass = b'),
 Text(1785.6000000000001, 1268.4, 'TCH <= 1.255\ngini = 0.308\nsamples = 1
226\nvalue = [146, 1591, 0, 199]\nclass = b'),
 Text(1488.0, 906.0, 'TCH <= 1.205\ngini = 0.617\nsamples = 197\nvalue =
[144, 49, 0, 126]\nclass = a'),
 Text(1339.2, 543.5999999999999, 'NMHC <= 0.015\ngini = 0.281\nsamples = 9
0\nvalue = [126, 9, 0, 15]\nclass = a'),
 Text(1264.8000000000002, 181.19999999999982, 'gini = 0.0\nsamples = 75\nv
alue = [125, 0, 0, 0]\nclass = a'),
 Text(1413.6000000000001, 181.19999999999982, 'gini = 0.509\nsamples = 15
\nvalue = [1, 9, 0, 15]\nclass = d'),
 Text(1636.8000000000002, 543.5999999999999, 'SO_2 <= 4.255\ngini = 0.501
\nsamples = 107\nvalue = [18, 40, 0, 111]\nclass = d'),
 Text(1562.4, 181.19999999999982, 'gini = 0.0\nsamples = 16\nvalue = [0, 3
0, 0, 0]\nclass = b'),
 Text(1711.2, 181.19999999999982, 'gini = 0.34\nsamples = 91\nvalue = [18,
10, 0, 111]\nclass = d'),
 Text(2083.2000000000003, 906.0, 'NMHC <= 0.075\ngini = 0.089\nsamples = 1
029\nvalue = [2, 1542, 0, 73]\nclass = b'),
 Text(1934.4, 543.5999999999999, 'EBE <= 1.095\ngini = 0.511\nsamples = 35
\nvalue = [2, 22, 0, 34]\nclass = d'),
 Text(1860.0000000000002, 181.19999999999982, 'gini = 0.219\nsamples = 17
\nvalue = [0, 3, 0, 21]\nclass = d'),
 Text(2008.8000000000002, 181.19999999999982, 'gini = 0.538\nsamples = 18
\nvalue = [2, 19, 0, 13]\nclass = b'),
 Text(2232.0, 543.5999999999999, 'CO <= 0.395\ngini = 0.049\nsamples = 994
\nvalue = [0, 1520, 0, 39]\nclass = b'),
 Text(2157.6000000000004, 181.19999999999982, 'gini = 0.185\nsamples = 203
```

```
\nvalue = [0, 287, 0, 33]\nclass = b'),
 Text(2306.4, 181.19999999999982, 'gini = 0.01\nsamples = 791\nvalue = [0,
1233, 0, 6]\nclass = b'),
 Text(3292.2000000000003, 1630.8000000000002, 'BEN <= 2.605\ngini = 0.694
\nsamples = 11026\nvalue = [4860, 835, 5882, 5773]\nclass = c'),
 Text(2715.6000000000004, 1268.4, 'EBE <= 0.785\ngini = 0.668\nsamples = 6
282\nvalue = [1654, 581, 3692, 3978]\nclass = d'),
 Text(2455.2000000000003, 906.0, 'NO_2 <= 10.235\ngini = 0.354\nsamples =
458\nvalue = [5, 44, 580, 107]\nclass = c'),
 Text(2380.8, 543.5999999999999, 'gini = 0.432\nsamples = 21\nvalue = [0,
23, 7, 2]\nclass = b'),
 Text(2529.6000000000004, 543.5999999999999, 'OXY <= 0.745\ngini = 0.314\n
samples = 437\nvalue = [5, 21, 573, 105]\nclass = c'),
 Text(2455.2000000000003, 181.19999999999982, 'gini = 0.147\nsamples = 275
\nvalue = [4, 8, 403, 22]\nclass = c'),
 Text(2604.0, 181.19999999999982, 'gini = 0.496\nsamples = 162\nvalue =
[1, 13, 170, 83]\nclass = c'),
 Text(2976.0, 906.0, 'O_3 <= 10.365\ngini = 0.671\nsamples = 5824\nvalue =
[1649, 537, 3112, 3871]\nclass = d'),
 Text(2827.2000000000003, 543.5999999999999, 'BEN <= 1.565\ngini = 0.606\n
samples = 866\nvalue = [131, 206, 791, 250]\nclass = c'),
 Text(2752.8, 181.19999999999982, 'gini = 0.46\nsamples = 266\nvalue = [2
0, 64, 294, 35]\nclass = c'),
 Text(2901.6000000000004, 181.19999999999982, 'gini = 0.65\nsamples = 600
\nvalue = [111, 142, 497, 215]\nclass = c'),
 Text(3124.8, 543.5999999999999, 'MXY <= 3.675\ngini = 0.655\nsamples = 49
58\nvalue = [1518, 331, 2321, 3621]\nclass = d'),
 Text(3050.4, 181.19999999999982, 'gini = 0.642\nsamples = 1851\nvalue =
[321, 225, 995, 1389]\nclass = d'),
 Text(3199.2000000000003, 181.19999999999982, 'gini = 0.654\nsamples = 310
7\nvalue = [1197, 106, 1326, 2252]\nclass = d'),
 Text(3868.8, 1268.4, 'OXY <= 4.465\ngini = 0.669\nsamples = 4744\nvalue =
[3206, 254, 2190, 1795]\nclass = a'),
 Text(3571.2000000000003, 906.0, 'NMHC <= 0.165\ngini = 0.684\nsamples = 1
872\nvalue = [615, 170, 960, 1142]\nclass = d'),
 Text(3422.4, 543.5999999999999, 'OXY <= 2.925\ngini = 0.62\nsamples = 739
\nvalue = [430, 14, 546, 171]\nclass = c'),
 Text(3348.0000000000005, 181.19999999999982, 'gini = 0.546\nsamples = 203
\nvalue = [31, 9, 193, 82]\nclass = c'),
 Text(3496.8, 181.19999999999982, 'gini = 0.592\nsamples = 536\nvalue = [3
99, 5, 353, 89]\nclass = a'),
 Text(3720.0000000000005, 543.5999999999999, 'O_3 <= 4.525\ngini = 0.606\n
samples = 1133\nvalue = [185, 156, 414, 971]\nclass = d'),
 Text(3645.6000000000004, 181.19999999999982, 'gini = 0.049\nsamples = 59
\nvalue = [0, 78, 2, 0]\nclass = b'),
 Text(3794.4, 181.19999999999982, 'gini = 0.574\nsamples = 1074\nvalue =
[185, 78, 412, 971]\nclass = d'),
 Text(4166.400000000001, 906.0, 'SO_2 <= 36.375\ngini = 0.583\nsamples = 2
872\nvalue = [2591, 84, 1230, 653]\nclass = a'),
 Text(4017.6000000000004, 543.5999999999999, 'NMHC <= 0.155\ngini = 0.602
\nsamples = 2473\nvalue = [2117, 84, 1185, 565]\nclass = a'),
 Text(3943.2000000000003, 181.19999999999982, 'gini = 0.425\nsamples = 697
```



Conclusion

Accuracy

Linear Regression:0.19864042035800789

Ridge Regression:0.059286006322965545

Lasso Regression:0.05399322398318607

ElasticNet Regression:0.09587022840432957

Logistic Regression:0.8480899292795158

Random Forest:0.7753022147710227

Logistic Regression is suitable for this dataset