20104016

DEENA

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_2009.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2009-10-01 01:00:00	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002	NaN	50.680000	18.2
1	2009-10-01 01:00:00	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000	NaN	55.880001	10.5
2	2009-10-01 01:00:00	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001	NaN	49.060001	25.1
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.5
4	2009-10-01 01:00:00	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002	NaN	38.090000	23.7
...	...	...	...	...	...	...	...	...	...	...	
215683	2009-06-01 00:00:00	0.50	0.22	0.39	0.75	0.09	22.000000	24.510000	1.00	82.239998	10.8
215684	2009-06-01 00:00:00	NaN	0.31	NaN	NaN	NaN	76.110001	101.099998	NaN	41.220001	9.9
215685	2009-06-01 00:00:00	0.13	NaN	0.86	NaN	0.23	81.050003	99.849998	NaN	24.830000	12.4
215686	2009-06-01 00:00:00	0.21	NaN	2.96	NaN	0.10	72.419998	82.959999	NaN	NaN	13.0
215687	2009-06-01 00:00:00	0.37	0.32	0.99	1.36	0.14	54.290001	64.480003	1.06	56.919998	15.3

215688 rows × 17 columns

Data Cleaning and Data Preprocessing

In [3]:

In [4]:

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]:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24717 entries, 3 to 215687
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     24717 non-null  object 
 1   BEN      24717 non-null  float64
 2   CO       24717 non-null  float64
 3   EBE      24717 non-null  float64
 4   MXY      24717 non-null  float64
 5   NMHC     24717 non-null  float64
 6   NO_2     24717 non-null  float64
 7   NOx      24717 non-null  float64
 8   OXY      24717 non-null  float64
 9   O_3      24717 non-null  float64
 10  PM10     24717 non-null  float64
 11  PM25     24717 non-null  float64
 12  PXY      24717 non-null  float64
 13  SO_2     24717 non-null  float64
 14  TCH      24717 non-null  float64
 15  TOL      24717 non-null  float64
 16  station  24717 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.4+ MB
```

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

Out[6]:

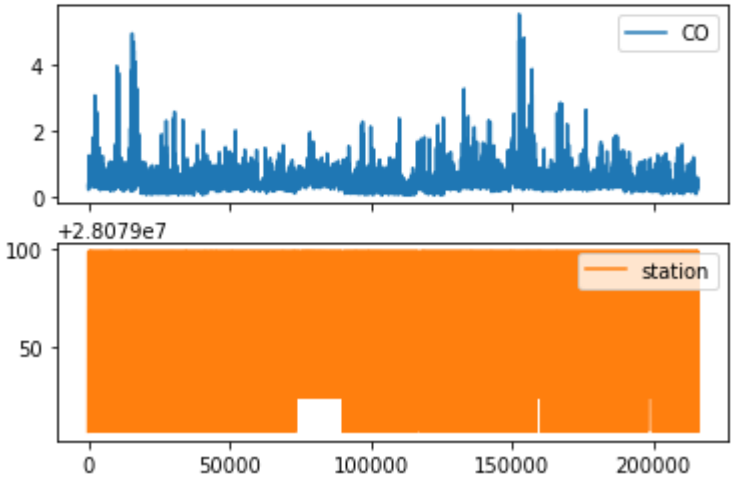
	CO	station
3	0.33	28079006
20	0.32	28079024
24	0.24	28079099
28	0.21	28079006
45	0.30	28079024
...	...	...
215659	0.27	28079024
215663	0.35	28079099
215667	0.29	28079006
215683	0.22	28079024
215687	0.32	28079099

24717 rows × 2 columns

Line chart

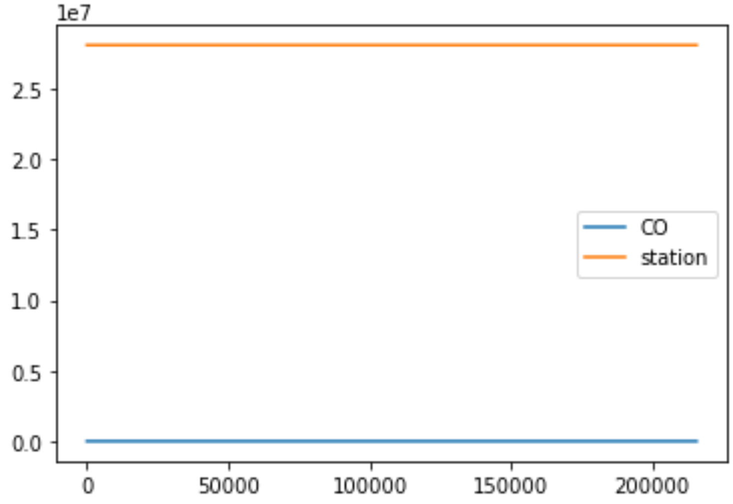
In [7]:

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



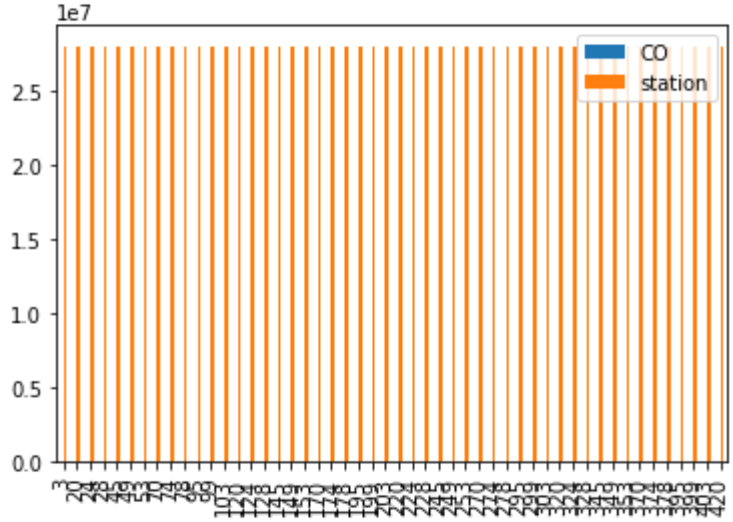
Line chart

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

Out[8]: <AxesSubplot:>
```



Bar chart

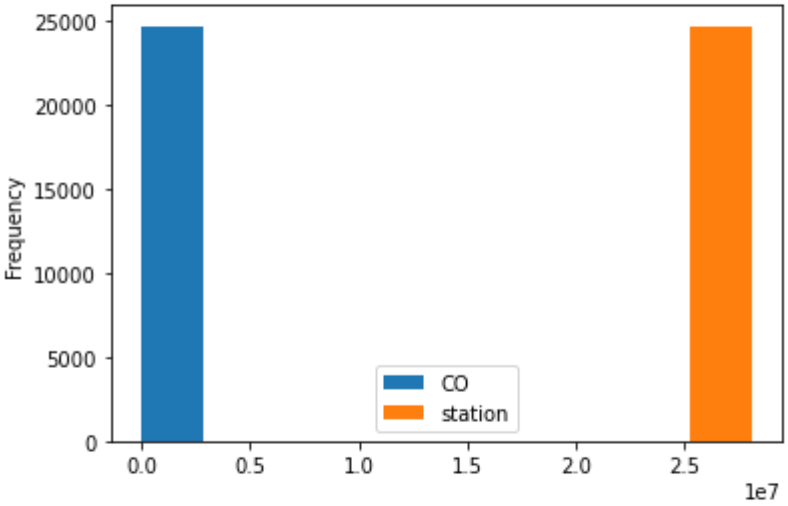
```
In [9]: b=data[0:50]

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

Out[10]: <AxesSubplot:>
```



Histogram

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

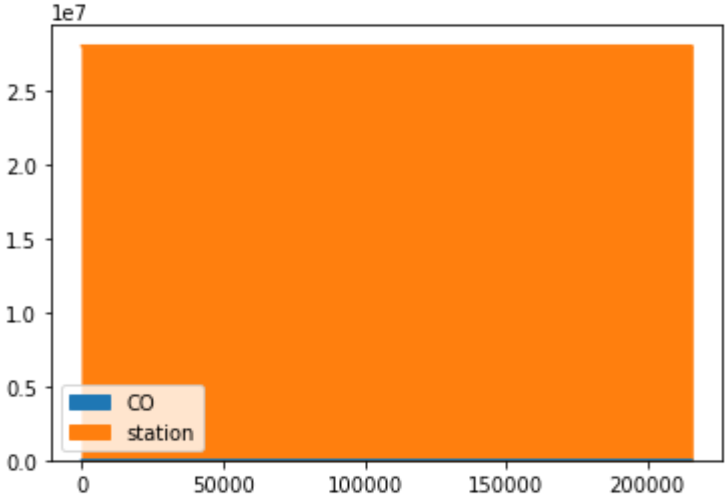
Out[11]:
```
<AxesSubplot:ylabel='Frequency'>
```



Area chart

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

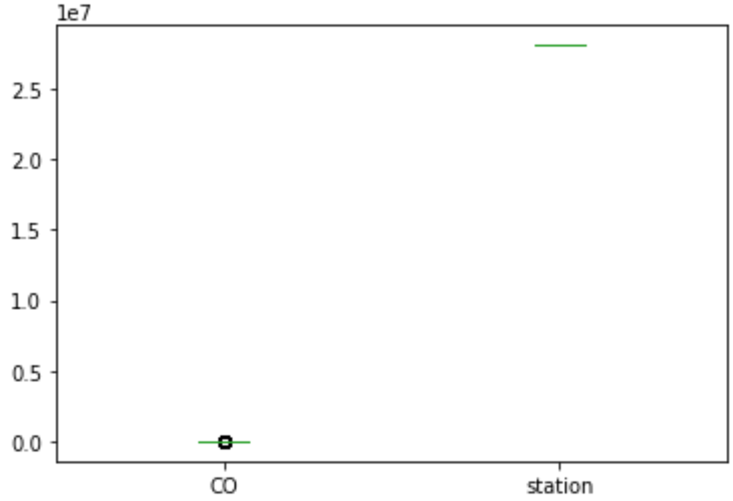
Out[12]:
```
<AxesSubplot:>
```



Box chart

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

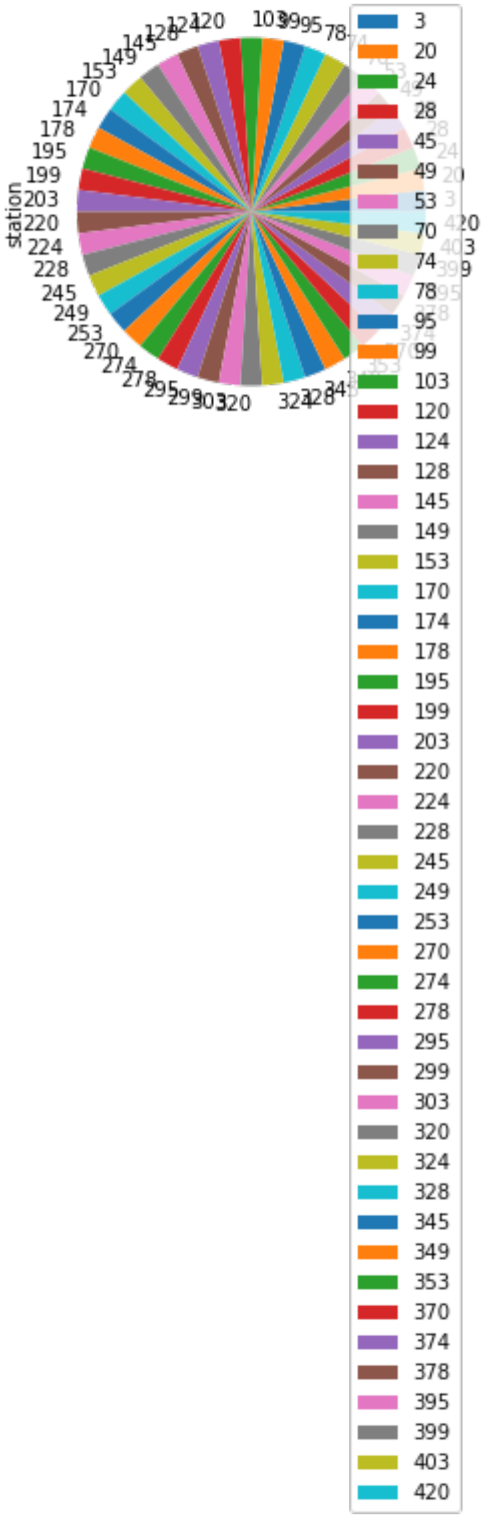
Out[13]:
```
<AxesSubplot:>
```



Pie chart

In [14]: [illegible]

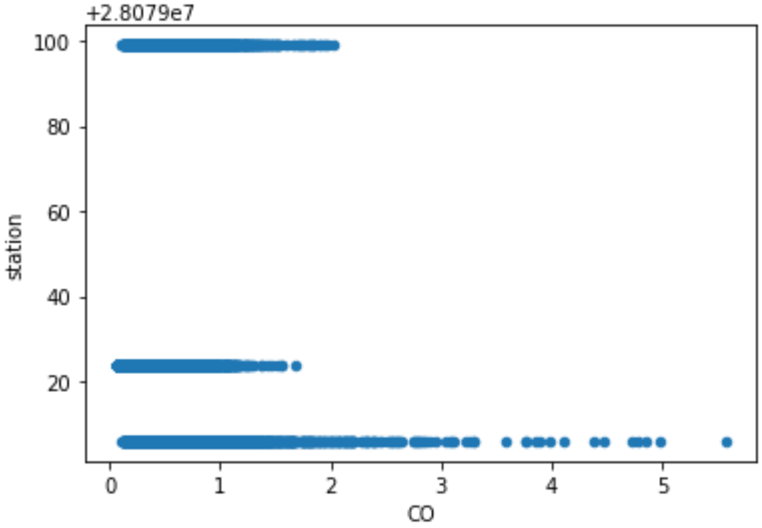
```
Out[14]: <AxesSubplot:ylabel='station'>
```



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: 24717 entries, 3 to 215687
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     24717 non-null  object 
 1   BEN      24717 non-null  float64
 2   CO       24717 non-null  float64
 3   EBE      24717 non-null  float64
 4   MXY      24717 non-null  float64
 5   NMHC     24717 non-null  float64
 6   NO_2     24717 non-null  float64
 7   NOx      24717 non-null  float64
 8   OXY      24717 non-null  float64
 9   O_3      24717 non-null  float64
 10  PM10     24717 non-null  float64
 11  PM25     24717 non-null  float64
 12  PXY      24717 non-null  float64
 13  SO_2     24717 non-null  float64
 14  TCH      24717 non-null  float64
```

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

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	
count	24717.000000	24717.000000	24717.000000	24717.000000	24717.000000	24717.000000	247
mean	1.010583	0.448056	1.262430	2.244469	0.219582	55.563929	
std	1.007345	0.291706	1.074768	2.242214	0.141661	38.911677	
min	0.170000	0.060000	0.250000	0.240000	0.000000	0.600000	
25%	0.460000	0.270000	0.720000	0.990000	0.140000	26.510000	
50%	0.670000	0.370000	1.000000	1.490000	0.190000	47.930000	
75%	1.180000	0.570000	1.430000	2.820000	0.260000	76.269997	1
max	22.379999	5.570000	47.669998	56.500000	2.580000	477.399994	14

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

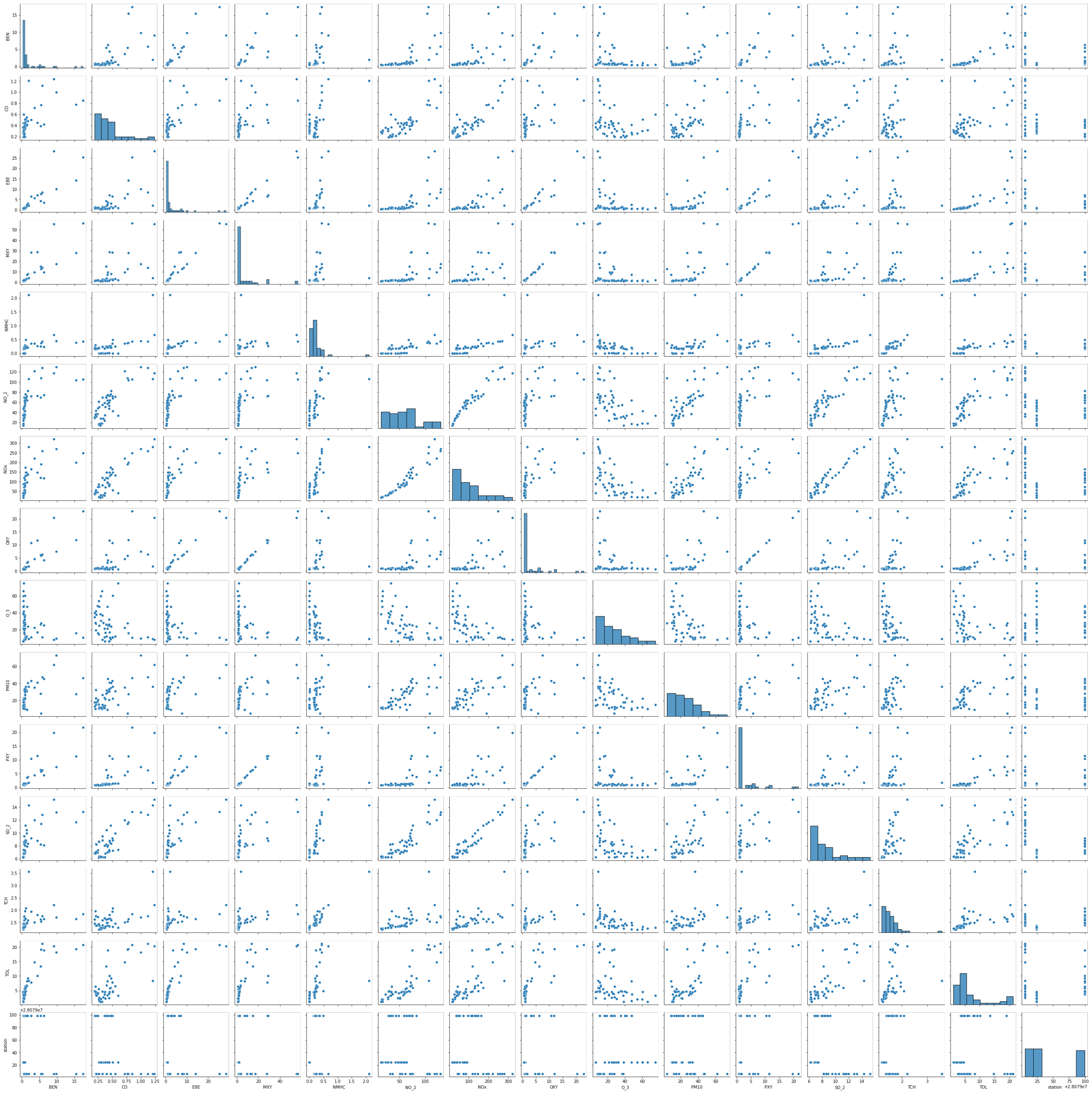
EDA AND VISUALIZATION

In [19]:

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

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x19b08992280>
```



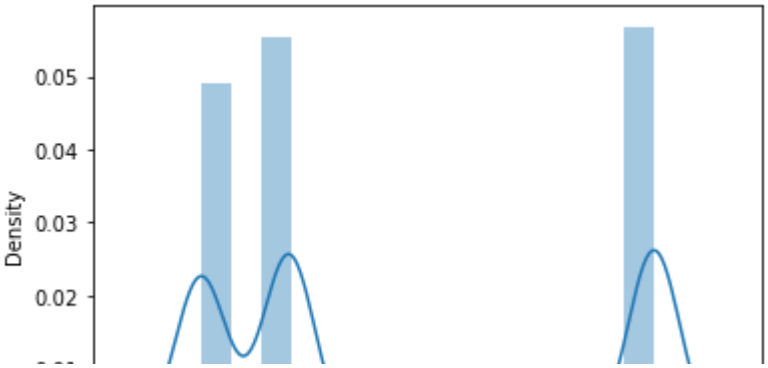
In [20]:

```
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'>
```

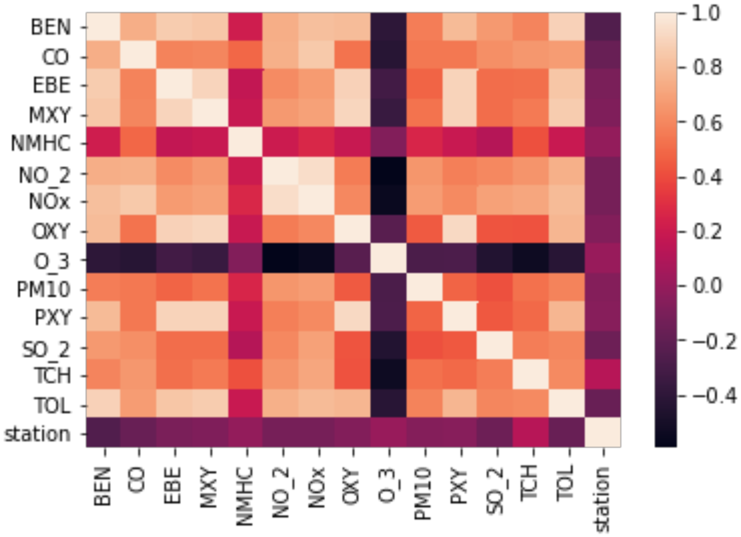


In [21]:

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

Out[21]:

```
<AxesSubplot:>
```



TO TRAIN THE MODEL AND MODEL BULDING

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]:
```
28078910.857585404
```

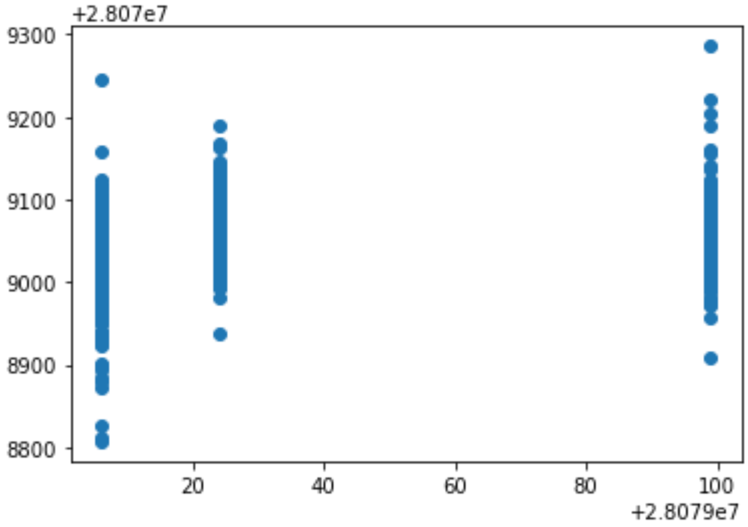
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-36.203443
CO	-30.648171
EBE	6.488422
MXY	-0.905931
NMHC	-16.617460
NO_2	-0.170997
NOx	0.213958
OXY	13.118534
O_3	0.010307
PM10	-0.053681
PXY	3.870440
SO_2	-0.332929
TCH	111.370403
TOL	-1.288304

```
In [27]: prediction =lr.predict(x_test)
```

```
Out[27]: <matplotlib.collections.PathCollection at 0x19b127da580>
```



ACCURACY

```
In [28]:
```

```
Out[28]: 0.29852823000409645
```

```
In [29]:
```

```
Out[29]: 0.2819204013446853
```

Ridge and Lasso

```
In [30]:
```

```
In [31]: rr=Ridge(alpha=10)
```

```
Out[31]: Ridge(alpha=10)
```

Accuracy(Ridge)

```
In [32]:
```

```
Out[32]: 0.296716101588l199
```

```
In [33]:
```

```
Out[33]: 0.2816595851212129
```

```
In [34]: la=Lasso(alpha=10)

Out[34]: Lasso(alpha=10)

In [35]:

Out[35]: 0.03737806206447625
```

Accuracy(Lasso)

```
In [36]:

Out[36]: 0.03407271577982274

In [37]: from sklearn.linear_model import ElasticNet
         en=ElasticNet()

Out[37]: ElasticNet()

In [38]:

Out[38]: array([-6.99357656, -0.70564139,  0.3459839 ,  2.25400319, -0.        ,
                -0.20897486,  0.11926181,  1.21581674, -0.14525936,  0.07281443,
                 1.9480131 , -0.77200717,  1.39621687, -2.06358417])

In [39]:

Out[39]: 28079063.86754315

In [40]:

In [41]:

Out[41]: 0.1051779453553251
```

Evaluation Metrics

```
In [42]: from sklearn import metrics
         print(metrics.mean_absolute_error(y_test,prediction))
         print(metrics.mean_squared_error(y_test,prediction))

         35.91039339833247
         1470.1634656890203
         38.342710724321776
```

Logistic Regression

```
In [43]:
```

```
In [44]: feature_matrix=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O
                'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
```

```
In [45]:

Out[45]: (24717, 14)
```

```
In [46]:

Out[46]: (24717,)
```

```
In [47]:
```

```
In [48]:
```

```
In [49]: logr=LogisticRegression(max_iter=10000)

Out[49]: LogisticRegression(max_iter=10000)
```

```
In [50]:
```

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

         [28079099]
```

```
In [52]:

Out[52]: array([28079006, 28079024, 28079099], dtype=int64)
```

```
In [53]:

Out[53]: 0.8951733624630821
```

```
In [54]:

Out[54]: 5.447205522232353e-13
```

```
In [55]:

Out[55]: array([[5.44720552e-13, 8.28692830e-44, 1.00000000e+00]])
```

Random Forest

```
In [56]:
```

```
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
```

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')
```

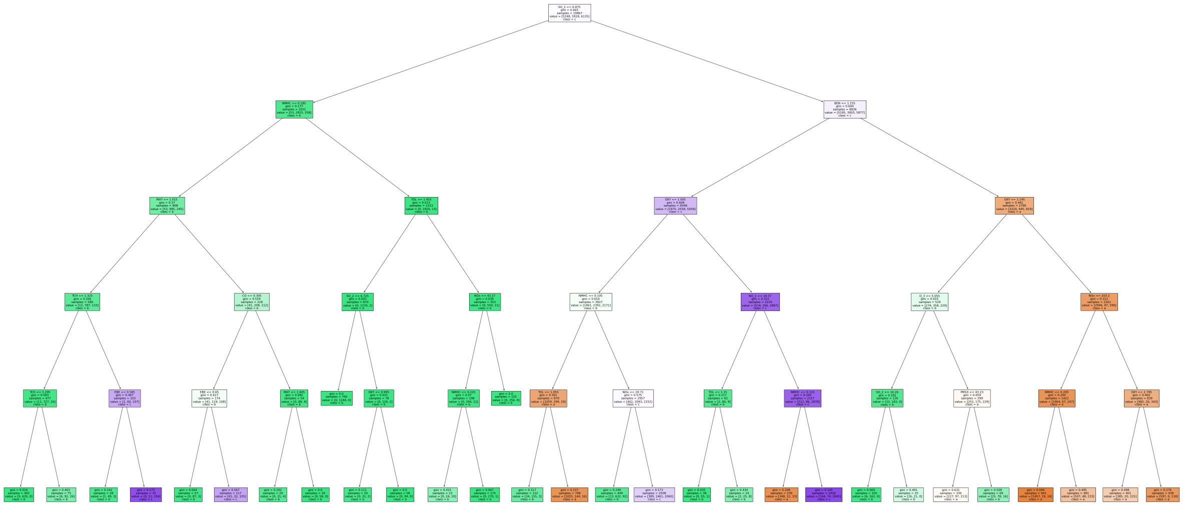
In [60]:

Out[60]:
```
0.8949199430985626
```

In [61]:

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

plt.figure(figsize=(80,40))
```

Out[62]:
```
[Text(2152.285714285714, 1993.2, 'SO_2 <= 6.875\ngini = 0.665\nsamples = 1086
7\nvalue = [5248, 5918, 6135]\nclass = c'),
 Text(1116.0, 1630.8000000000002, 'NMHC <= 0.185\ngini = 0.177\nsamples = 203
1\nvalue = [53, 2915, 258]\nclass = b'),
 Text(637.7142857142857, 1268.4, 'MXY <= 1.015\ngini = 0.37\nsamples = 808\nv
alue = [53, 995, 245]\nclass = b'),
 Text(318.85714285714283, 906.0, 'TCH <= 1.325\ngini = 0.266\nsamples = 580\n
value = [12, 787, 133]\nclass = b'),
 Text(159.42857142857142, 543.5999999999999, 'TCH <= 1.295\ngini = 0.093\nsam
ples = 477\nvalue = [11, 727, 26]\nclass = b'),
 Text(79.71428571428571, 181.19999999999982, 'gini = 0.016\nsamples = 402\nva
lue = [5, 635, 0]\nclass = b'),
 Text(239.1428571428571, 181.19999999999982, 'gini = 0.403\nsamples = 75\nval
ue = [6, 92, 26]\nclass = b'),
 Text(478.2857142857142, 543.5999999999999, 'EBE <= 0.585\ngini = 0.467\nsamp
les = 103\nvalue = [1, 60, 107]\nclass = c'),
 Text(398.57142857142856, 181.19999999999982, 'gini = 0.142\nsamples = 28\nva
lue = [1, 49, 3]\nclass = b'),
 Text(558.0, 181.19999999999982, 'gini = 0.173\nsamples = 75\nvalue = [0, 11,
104]\nclass = c'),
 Text(956.5714285714284, 906.0, 'CO <= 0.305\ngini = 0.559\nsamples = 228\nva
lue = [41, 208, 112]\nclass = b'),
 Text(797.1428571428571, 543.5999999999999, 'EBE <= 0.65\ngini = 0.617\nsampl
es = 174\nvalue = [41, 119, 108]\nclass = b'),
 Text(717.4285714285713, 181.19999999999982, 'gini = 0.064\nsamples = 57\nval
ue = [0, 87, 3]\nclass = b'),
 Text(876.8571428571428, 181.19999999999982, 'gini = 0.567\nsamples = 117\nva
lue = [41, 32, 105]\nclass = c'),
 Text(1116.0, 543.5999999999999, 'MXY <= 1.605\ngini = 0.082\nsamples = 54\nv
alue = [0, 89, 4]\nclass = b'),
 Text(1036.2857142857142, 181.19999999999982, 'gini = 0.202\nsamples = 20\nva
lue = [0, 31, 4]\nclass = b'),
 Text(1195.7142857142856, 181.19999999999982, 'gini = 0.0\nsamples = 34\nvalu
e = [0, 58, 0]\nclass = b'),
 Text(1594.2857142857142, 1268.4, 'TOL <= 1.425\ngini = 0.013\nsamples = 122
3\nvalue = [0, 1920, 13]\nclass = b'),
 Text(1355.142857142857, 906.0, 'SO_2 <= 6.725\ngini = 0.003\nsamples = 870\n
value = [0, 1370, 2]\nclass = b'),
 Text(1275.4285714285713, 543.5999999999999, 'gini = 0.0\nsamples = 792\nvalu
e = [0, 1244, 0]\nclass = b'),
 Text(1434.8571428571427, 543.5999999999999, 'OXY <= 0.995\ngini = 0.031\nsam
ples = 78\nvalue = [0, 126, 2]\nclass = b'),
 Text(1355.142857142857, 181.19999999999982, 'gini = 0.111\nsamples = 20\nval
ue = [0, 32, 2]\nclass = b'),
 Text(1514.5714285714284, 181.19999999999982, 'gini = 0.0\nsamples = 58\nvalu
e = [0, 94, 0]\nclass = b'),
 Text(1833.4285714285713, 906.0, 'NOx <= 43.27\ngini = 0.038\nsamples = 353\n
value = [0, 550, 11]\nclass = b'),
 Text(1753.7142857142856, 543.5999999999999, 'NMHC <= 0.225\ngini = 0.07\nsam
ples = 198\nvalue = [0, 294, 11]\nclass = b'),
 Text(1673.9999999999998, 181.19999999999982, 'gini = 0.415\nsamples = 23\nva
lue = [0, 24, 10]\nclass = b'),
 Text(1833.4285714285713, 181.19999999999982, 'gini = 0.007\nsamples = 175\nv
alue = [0, 270, 1]\nclass = b'),
 Text(1913.1428571428569, 543.5999999999999, 'gini = 0.0\nsamples = 155\nvalu
e = [0, 256, 0]\nclass = b'),
 Text(3188.5714285714284, 1630.8000000000002, 'BEN <= 1.155\ngini = 0.644\nsa
mples = 8836\nvalue = [5195, 3003, 5877]\nclass = c'),
 Text(2550.8571428571427, 1268.4, 'OXY <= 1.005\ngini = 0.608\nsamples = 604
6\nvalue = [1975, 2558, 5058]\nclass = c'),
 Text(2232.0, 906.0, 'NMHC <= 0.105\ngini = 0.654\nsamples = 3827\nvalue = [1
461, 2392, 2171]\nclass = b'),
 Text(2072.5714285714284, 543.5999999999999, 'TOL <= 1.055\ngini = 0.361\nsam
ples = 870\nvalue = [1059, 299, 19]\nclass = a'),
 Text(1992.8571428571427, 181.19999999999982, 'gini = 0.317\nsamples = 122\nv
alue = [34, 155, 3]\nclass = b'),
 Text(2152.285714285714, 181.19999999999982, 'gini = 0.237\nsamples = 748\nva
lue = [1025, 144, 16]\nclass = a'),
 Text(2391.428571428571, 543.5999999999999, 'NOx <= 20.71\ngini = 0.575\nsamp
les = 2957\nvalue = [402, 2093, 2152]\nclass = c'),
 Text(2311.7142857142853, 181.19999999999982, 'gini = 0.249\nsamples = 449\nv
alue = [13, 632, 92]\nclass = b'),
 Text(2471.142857142857, 181.19999999999982, 'gini = 0.573\nsamples = 2508\nv
alue = [389, 1461, 2060]\nclass = c'),
 Text(2869.7142857142853, 906.0, 'NO_2 <= 18.37\ngini = 0.322\nsamples = 221
9\nvalue = [514, 166, 2887]\nclass = c'),
 Text(2710.285714285714, 543.5999999999999, 'TOL <= 1.15\ngini = 0.217\nsampl
es = 62\nvalue = [2, 80, 9]\nclass = b'),
 Text(2630.5714285714284, 181.19999999999982, 'gini = 0.035\nsamples = 38\nva
lue = [0, 55, 1]\nclass = b'),
 Text(2790.0, 181.19999999999982, 'gini = 0.434\nsamples = 24\nvalue = [2, 2
5, 8]\nclass = b'),
 Text(3029.142857142857, 543.5999999999999, 'NMHC <= 0.115\ngini = 0.292\nsam
ples = 2157\nvalue = [512, 86, 2878]\nclass = c'),
 Text(2949.428571428571, 181.19999999999982, 'gini = 0.208\nsamples = 239\nva
lue = [348, 12, 33]\nclass = a'),
 Text(3108.8571428571427, 181.19999999999982, 'gini = 0.145\nsamples = 1918\n
value = [164, 74, 2845]\nclass = c'),
 Text(3826.2857142857138, 1268.4, 'OXY <= 1.145\ngini = 0.441\nsamples = 279
0\nvalue = [3220, 445, 819]\nclass = a'),
 Text(3507.428571428571, 906.0, 'O_3 <= 5.055\ngini = 0.655\nsamples = 529\nv
alue = [274, 358, 229]\nclass = b'),
 Text(3347.9999999999995, 543.5999999999999, 'SO_2 <= 16.26\ngini = 0.192\nsa
mples = 130\nvalue = [22, 183, 0]\nclass = b'),
 Text(3268.285714285714, 181.19999999999982, 'gini = 0.069\nsamples = 105\nva
lue = [6, 162, 0]\nclass = b'),
 Text(3427.7142857142853, 181.19999999999982, 'gini = 0.491\nsamples = 25\nva
lue = [16, 21, 0]\nclass = b'),
 Text(3666.8571428571427, 543.5999999999999, 'PM10 <= 43.23\ngini = 0.659\nsa
mples = 399\nvalue = [252, 175, 229]\nclass = a'),
 Text(3587.142857142857, 181.19999999999982, 'gini = 0.631\nsamples = 330\nva
lue = [227, 97, 213]\nclass = a'),
 Text(3746.5714285714284, 181.19999999999982, 'gini = 0.508\nsamples = 69\nva
lue = [25, 78, 16]\nclass = b'),
 Text(4145.142857142857, 906.0, 'NOx <= 203.2\ngini = 0.312\nsamples = 2261\n
value = [2946, 87, 590]\nclass = a'),
 Text(3985.7142857142853, 543.5999999999999, 'NMHC <= 0.205\ngini = 0.244\nsa
mples = 1422\nvalue = [1964, 67, 247]\nclass = a'),
 Text(3905.9999999999995, 181.19999999999982, 'gini = 0.044\nsamples = 941\nv
alue = [1457, 19, 14]\nclass = a'),
 Text(4065.428571428571, 181.19999999999982, 'gini = 0.495\nsamples = 481\nva
lue = [507, 48, 233]\nclass = a'),
 Text(4304.571428571428, 543.5999999999999, 'OXY <= 2.795\ngini = 0.402\nsamp
les = 839\nvalue = [982, 20, 343]\nclass = a'),
 Text(4224.857142857142, 181.19999999999982, 'gini = 0.498\nsamples = 401\nva
lue = [385, 20, 225]\nclass = a'),
 Text(4384.285714285714, 181.19999999999982, 'gini = 0.276\nsamples = 438\nva
lue = [597, 0, 118]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression :0.2819204013446853

Ridge Regression : 0.03737806206447625

Lasso Regression : 0.03407271577982274

ElasticNet Regression :0.1051779453553251

Logistic Regression :0.8951733624630821

Random Forest :0.8949199430985626

Random Forest is suitable for this dataset