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_2004.
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	
0	2004-08-01 01:00:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39
1	2004-08-01 01:00:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22
2	2004-08-01 01:00:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49
3	2004-08-01 01:00:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31
4	2004-08-01 01:00:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54
...	...	...	...	...	...	...	...	...	...	...	
245491	2004-06-01 00:00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30
245492	2004-06-01 00:00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37
245493	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22
245494	2004-06-01 00:00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45
245495	2004-06-01 00:00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24

245496 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: 19397 entries, 5 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     19397 non-null  object 
 1   BEN      19397 non-null  float64
 2   CO       19397 non-null  float64
 3   EBE      19397 non-null  float64
 4   MXY      19397 non-null  float64
 5   NMHC     19397 non-null  float64
 6   NO_2     19397 non-null  float64
 7   NOx      19397 non-null  float64
 8   OXY      19397 non-null  float64
 9   O_3      19397 non-null  float64
 10  PM10     19397 non-null  float64
 11  PM25     19397 non-null  float64
 12  PXY      19397 non-null  float64
 13  SO_2     19397 non-null  float64
 14  TCH      19397 non-null  float64
 15  TOL      19397 non-null  float64
 16  station  19397 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 2.7+ MB
```

In [8]:

```
data=df[['OXY', 'TCH', 'NOx']]
data
```

Out[8]:

	OXY	TCH	NOx
5	5.04	1.54	144.800003
22	0.50	1.55	32.799999
26	2.47	1.55	75.470001
32	2.56	1.67	165.800003
49	0.46	1.41	34.840000
...	...	...	...
245463	0.42	1.24	45.450001
245467	2.09	1.47	132.800003
245473	4.51	1.58	253.600006
245491	0.66	1.28	64.389999
245495	2.60	1.47	141.000000

19397 rows × 3 columns

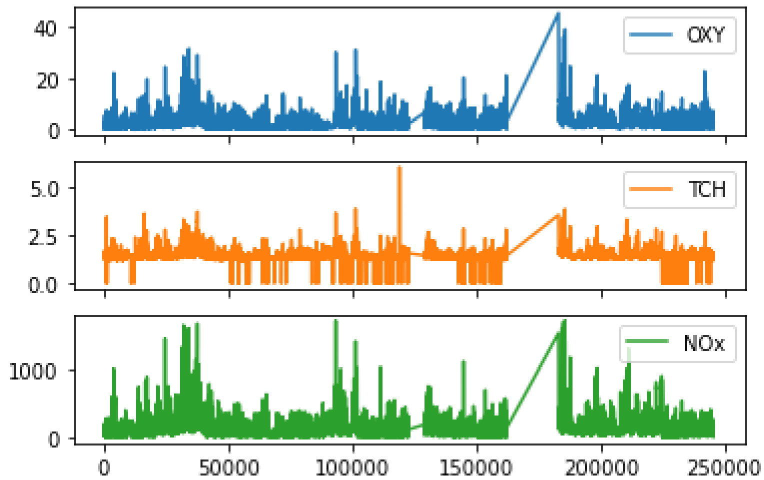
Line chart

In [9]:

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

Out[9]:

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



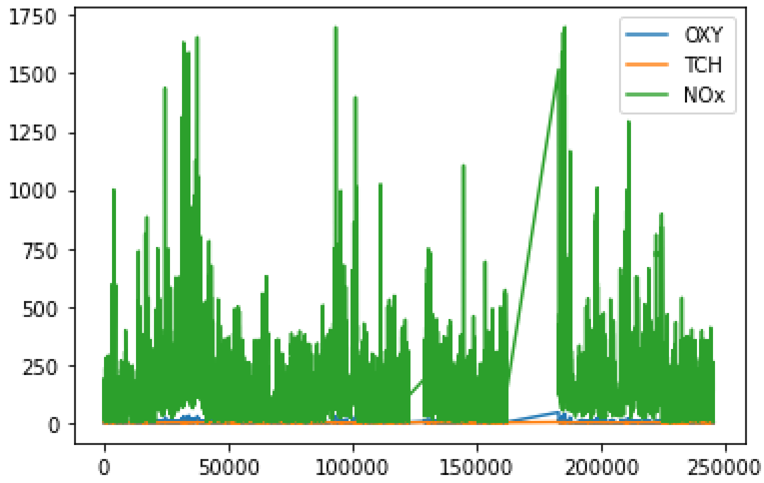
Line chart

In [10]:

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

Out[10]:

<AxesSubplot:>



Bar chart

In [11]:

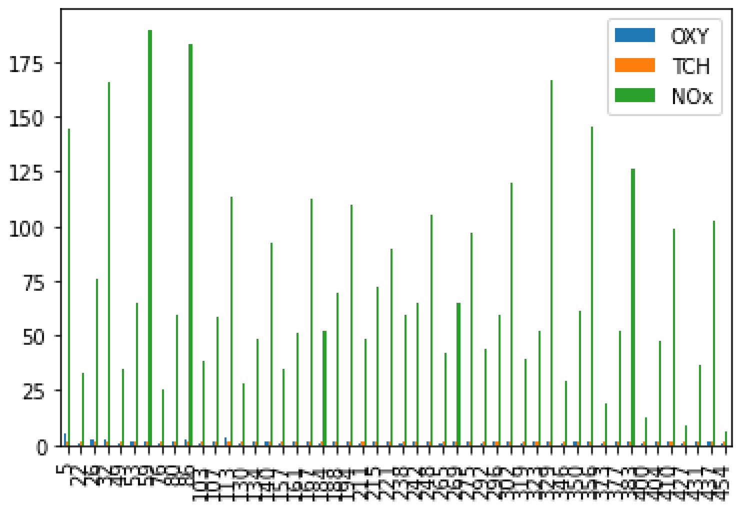
```
b=data[0:50]
```

In [12]:

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

Out[12]:

<AxesSubplot:>



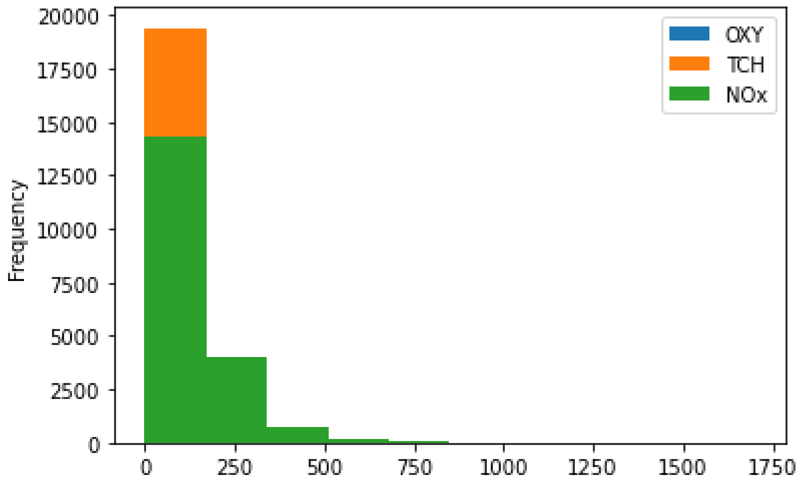
Histogram

In [13]:

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

Out[13]:

<AxesSubplot:ylabel='Frequency'>



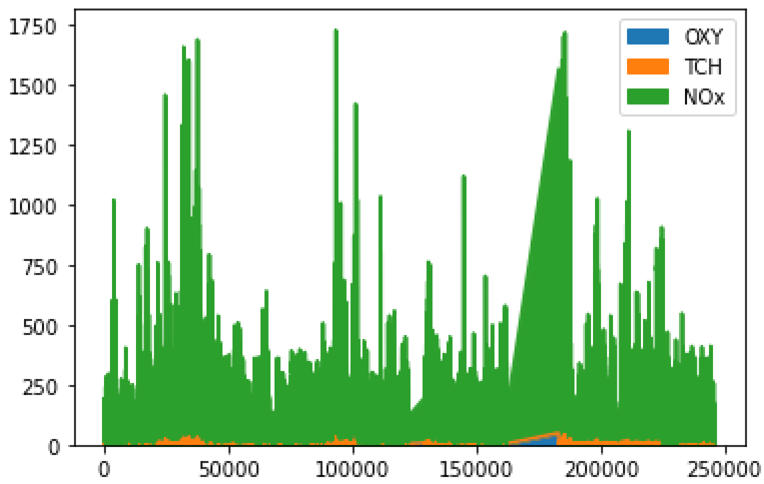
Area chart

In [14]:

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

Out[14]:

<AxesSubplot:>



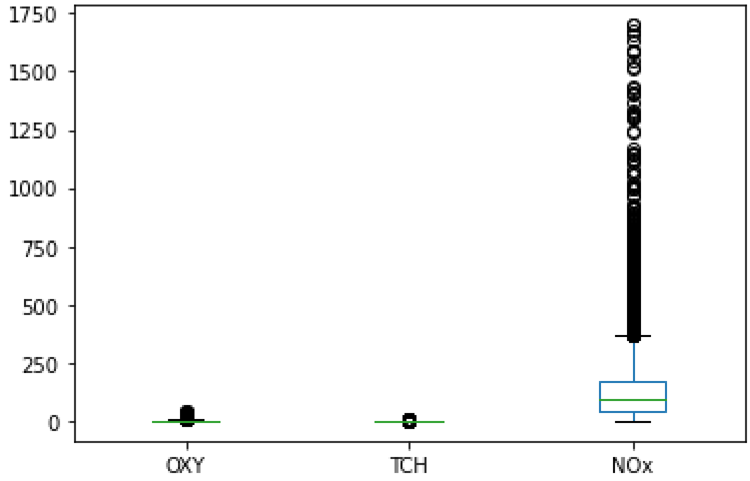
Box chart

In [15]:

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

Out[15]:

<AxesSubplot:>



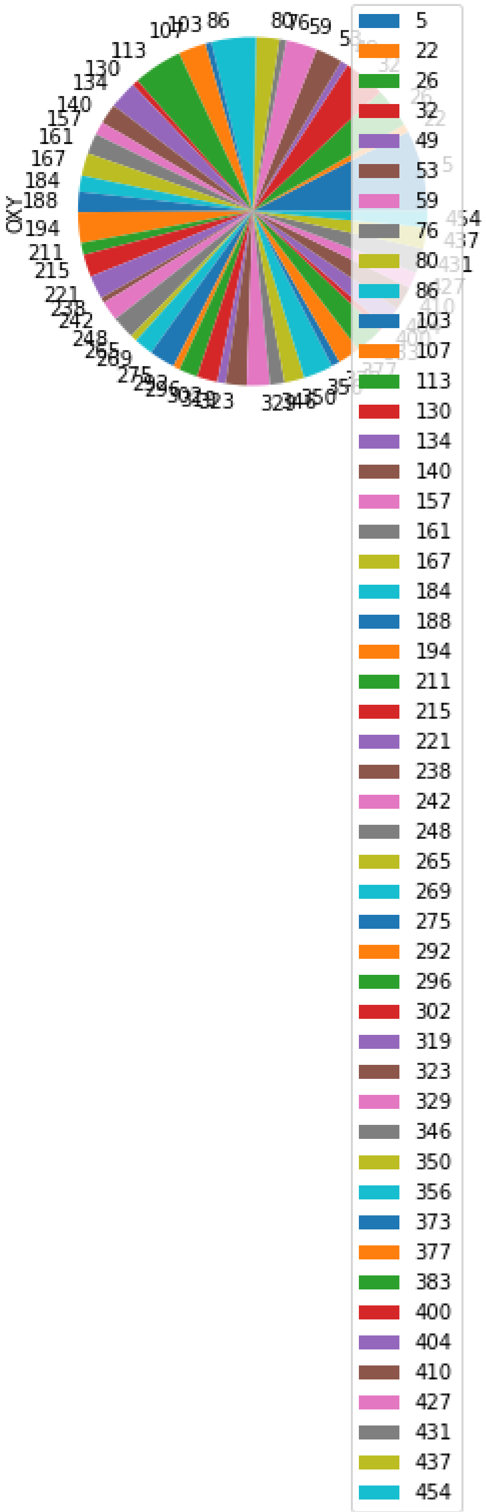
Pie chart

In [17]:

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

Out[17]:

<AxesSubplot:ylabel='OXY'>



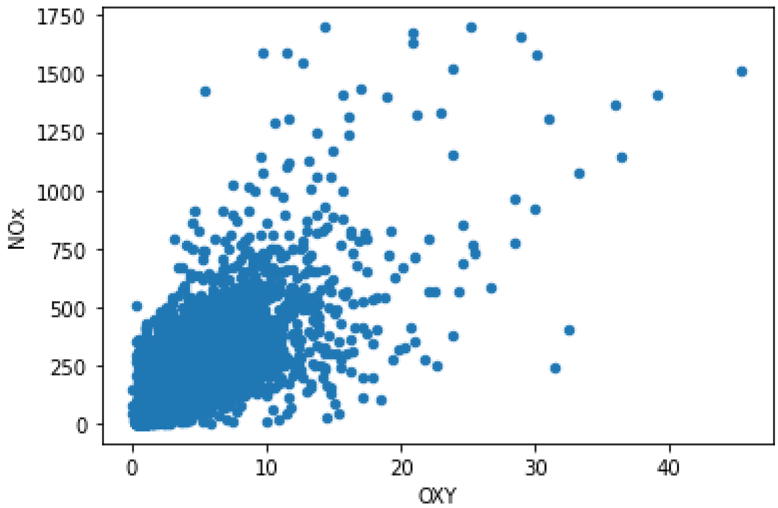
Scatter chart

In [18]:

```
data.plot.scatter(x='OXY' ,y='NOx')
```

Out[18]:

```
<AxesSubplot:xlabel='OXY', ylabel='NOx'>
```



In [19]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19397 entries, 5 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     19397 non-null  object 
 1   BEN      19397 non-null  float64
 2   CO       19397 non-null  float64
 3   EBE      19397 non-null  float64
 4   MXY      19397 non-null  float64
 5   NMHC     19397 non-null  float64
 6   NO_2     19397 non-null  float64
 7   NOx      19397 non-null  float64
 8   OXY      19397 non-null  float64
 9   O_3      19397 non-null  float64
 10  PM10     19397 non-null  float64
 11  PM25     19397 non-null  float64
 12  PXY      19397 non-null  float64
 13  SO_2     19397 non-null  float64
 14  TCH      19397 non-null  float64
 15  TOL      19397 non-null  float64
 16  station  19397 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 2.7+ MB
```

In [20]:

```
df.describe()
```

Out[20]:

	BEN	CO	EBE	MXY	NMHC	NO_2
count	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000
mean	2.250781	0.675347	2.775913	5.424809	0.151024	62.887023
std	2.184724	0.591026	2.729622	5.554358	0.158603	37.952255
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.090000
25%	0.870000	0.320000	1.020000	1.780000	0.060000	35.150002
50%	1.620000	0.520000	1.970000	3.800000	0.110000	58.310001
75%	2.910000	0.860000	3.580000	7.260000	0.200000	85.730003
max	34.180000	8.900000	41.880001	91.599998	4.810000	355.100006

In [21]:

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

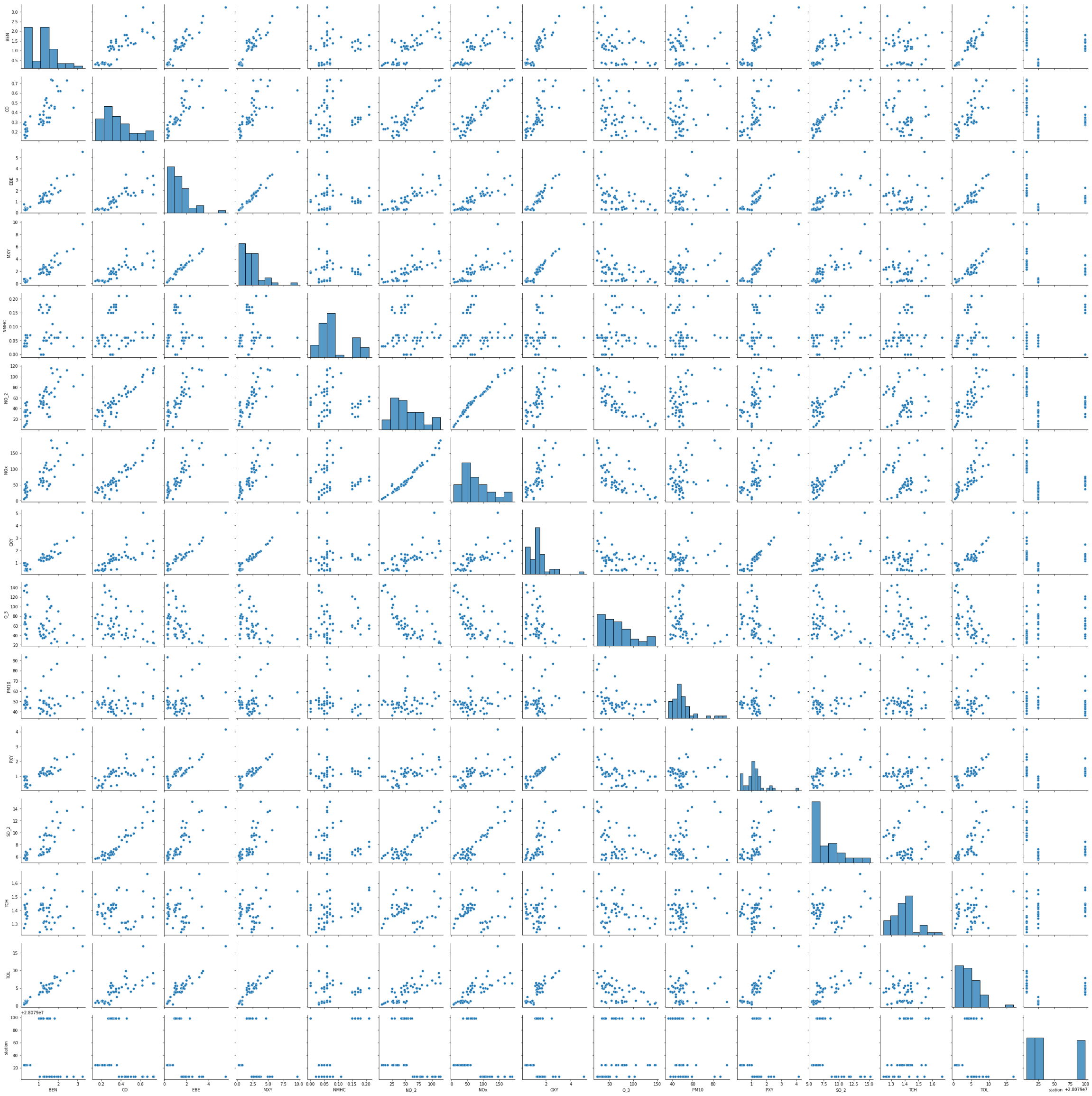
EDA AND VISUALIZATION

In [22]:

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

Out[22]:

<seaborn.axisgrid.PairGrid at 0x1f543b78100>



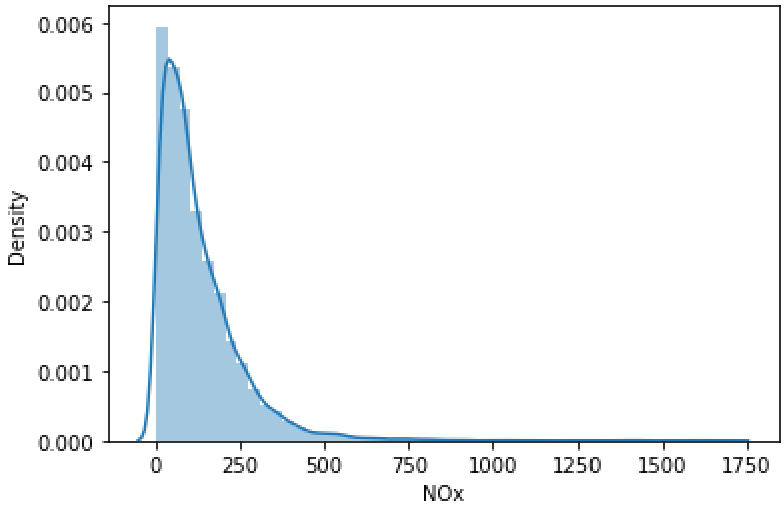
In [26]:

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

```
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[26]:

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

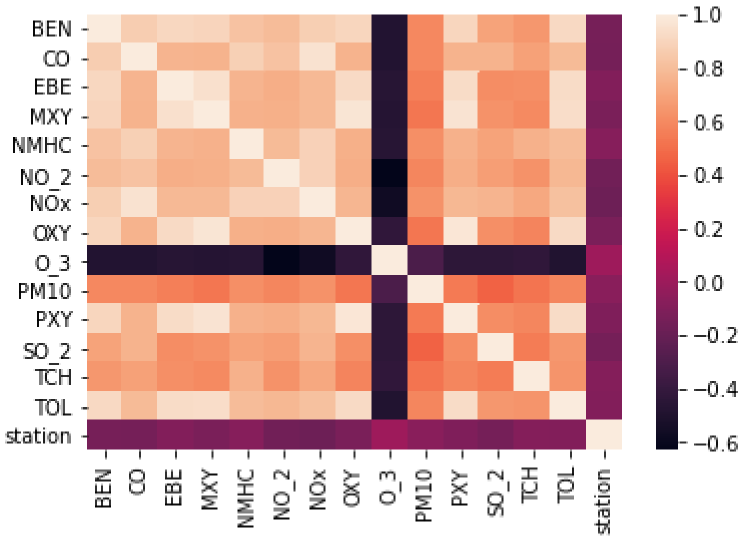


In [27]:

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

Out[27]:

```
<AxesSubplot:>
```



TO TRAIN THE MODEL AND MODEL BULDING

In [28]:

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

In [29]:

```
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 [30]:

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

Out[30]:

```
LinearRegression()
```

In [31]:

```
lr.intercept_
```

Out[31]:

```
28079074.466013305
```

In [32]:

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

Out[32]:

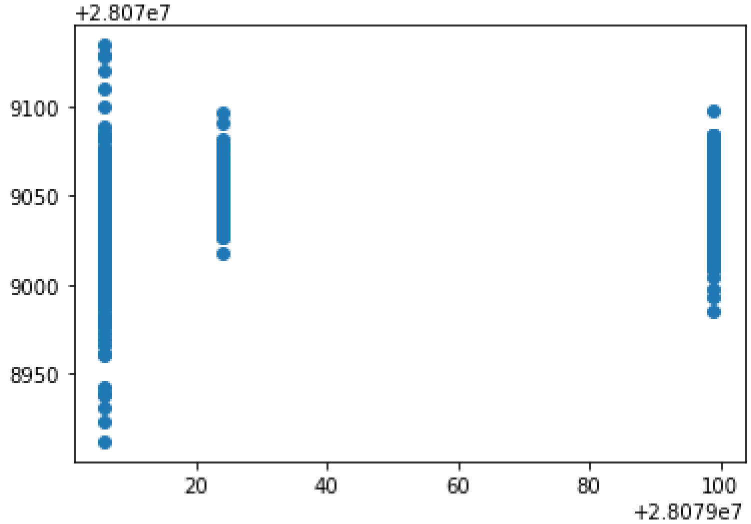
	Co-efficient
BEN	-3.428438
CO	23.720830
EBE	3.462263
MXY	-3.428359
NMHC	82.207293
NO_2	-0.142861
NOx	-0.245975
OXY	-2.022878
O_3	-0.271436
PM10	0.062757
PXY	6.011119
SO_2	-0.209234
TCH	-6.418532
TOL	1.149801

In [33]:

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

Out[33]:

<matplotlib.collections.PathCollection at 0x1f550850700>



ACCURACY

In [34]:

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

Out[34]:

```
0.11870348140040887
```

In [35]:

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

Out[35]:

```
0.10067971554067745
```

Ridge and Lasso

In [36]:

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

In [37]:

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

Out[37]:

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

Accuracy(Ridge)

In [38]:

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

Out[38]:

```
0.11664945877101263
```

In [39]:

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

Out[39]:

```
0.1003371173292632
```

In [40]:

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

Out[40]:

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

In [41]:

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

Out[41]:

```
0.0517813004708695 96
```

Accuracy(Lasso)

In [42]:

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

Out[42]:

```
0.05677379563780 2715
```

Accuracy(Elastic Net)

In [43]:

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

Out[43]:

```
ElasticNet()
```

In [44]:

```
en.coef_
```

Out[44]:

```
array([-0.        ,  0.33534742,  1.45923881, -1.87938948,  0.        ,
       -0.1537371 , -0.09301587, -0.        , -0.20672745,  0.09436363,
        0.5749052 , -0.13540571,  0.        ,  1.20582375])
```

In [45]:

```
en.intercept_
```

Out[45]:

```
28079066.002937175
```

In [46]:

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

In [47]:

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

Out[47]:

```
0.070338113054687731
```

Evaluation Metrics

In [48]:

```
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)))
```

```
38.45619093357262
1639.150228011821
40.4864202913992
```

Logistic Regression

In [49]:

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

In [50]:

```
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 [51]:

```
feature_matrix.shape
```

Out[51]:

```
(19397, 14)
```

In [52]:

```
target_vector.shape
```

Out[52]:

```
(19397,)
```

In [53]:

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

In [54]:

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

In [55]:

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

Out[55]:

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

In [56]:

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

In [57]:

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

```
[28079006]
```

In [58]:

```
logr.classes_
```

Out[58]:

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

In [59]:

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

Out[59]:

```
0.7360416559261741
```

In [60]:

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

Out[60]:

```
0.9999978255573396
```

In [61]:

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

Out[61]:

```
array([[9.99997826e-01, 7.75018107e-20, 2.17444266e-06]])
```

Random Forest

In [62]:

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

In [63]:

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

Out[63]:

```
RandomForestClassifier()
```

In [64]:

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

In [65]:

```
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[65]:

```
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 [66]:

```
grid_search.best_score_
```

Out[66]:

```
0.7720404784056986
```

In [67]:

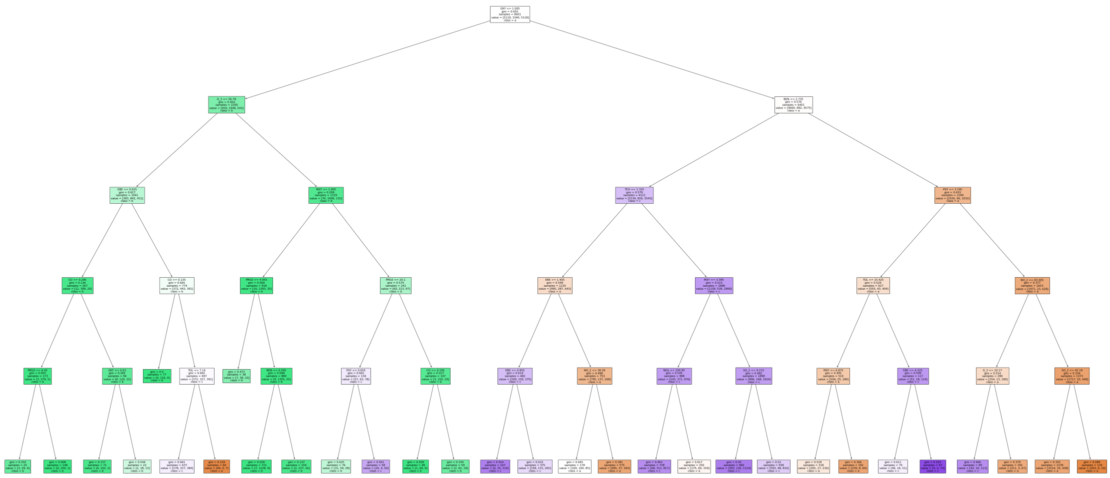
```
rfc_best=grid_search.best_estimator_
```

In [68]:

```
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
```

```
39\nvalue = [1514, 10, 438]\nclass = a'),
 Text(4384.285714285714, 181.19999999999982, 'gini = 0.089\nsamples = 13
4\nvalue = [203, 0, 10]\nclass = a')]
```



Conclusion

Accuracy

Linear Regression:0.10067971554067745

Ridge Regression:0.05178130047086959 6

Lasso Regression:0.056773795637802715

ElasticNet Regression:0.07033811305468773

Logistic Regression:0.7360416559261741

Random Forest:0.7720404784056986

Random Forest is suitable for this dataset