In [1]:
```python
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
data=pd.read_csv(r"C:\Users\user\Desktop\DINESH\C10_air\madrid_2009.csv")
data
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
0	2009-10-01 01:00:00	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002	NaN	50.680000	18.2600
1	2009-10-01 01:00:00	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000	NaN	55.880001	10.5800
2	2009-10-01 01:00:00	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001	NaN	49.060001	25.1900
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	26.5300
4	2009-10-01 01:00:00	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002	NaN	38.090000	23.7600
...	...	...	...	...	...	...	...	...	...	...	
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	10.8300
215684	2009-06-01 00:00:00	NaN	0.31	NaN	NaN	NaN	76.110001	101.099998	NaN	41.220001	9.9200
215685	2009-06-01 00:00:00	0.13	NaN	0.86	NaN	0.23	81.050003	99.849998	NaN	24.830000	12.4600
215686	2009-06-01 00:00:00	0.21	NaN	2.96	NaN	0.10	72.419998	82.959999	NaN	NaN	13.0300
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	15.3600

215688 rows × 17 columns

In [3]:
```
data.head(10)
```

Out[3]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM
0	2009-10-01 01:00:00	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002	NaN	50.680000	18.260000	N
1	2009-10-01 01:00:00	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000	NaN	55.880001	10.580000	N
2	2009-10-01 01:00:00	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001	NaN	49.060001	25.190001	N
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	26.530001	6
4	2009-10-01 01:00:00	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002	NaN	38.090000	23.760000	N
5	2009-10-01 01:00:00	NaN	0.29	NaN	NaN	NaN	43.200001	50.080002	NaN	35.840000	21.870001	N
6	2009-10-01 01:00:00	NaN	0.20	NaN	NaN	NaN	35.430000	38.520000	NaN	33.549999	17.350000	N
7	2009-10-01 01:00:00	NaN	0.15	NaN	NaN	NaN	27.309999	33.150002	NaN	53.549999	16.520000	11
8	2009-10-01 01:00:00	NaN	0.21	NaN	NaN	0.39	33.889999	40.799999	NaN	58.549999	16.650000	N
9	2009-10-01 01:00:00	NaN	0.32	NaN	NaN	NaN	46.349998	60.540001	NaN	45.340000	15.160000	N

In [4]:
```
data.tail(20)
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
215668	2009-06-01 00:00:00	NaN	0.21	NaN	NaN	0.13	44.130001	46.730000	NaN	75.720001	6.1000
215669	2009-06-01 00:00:00	0.15	0.31	0.39	NaN	0.18	49.000000	55.669998	NaN	57.200001	18.0300
215670	2009-06-01 00:00:00	NaN	0.34	NaN	NaN	NaN	56.799999	74.120003	NaN	22.860001	15.7300
215671	2009-06-01 00:00:00	NaN	0.37	NaN	NaN	NaN	62.450001	81.910004	NaN	55.360001	35.5999
215672	2009-06-01 00:00:00	NaN	0.28	NaN	NaN	0.15	25.340000	28.260000	NaN	65.750000	12.0400
215673	2009-06-01 00:00:00	NaN	0.35	NaN	NaN	NaN	40.160000	42.959999	NaN	87.650002	7.5600
215674	2009-06-01 00:00:00	NaN	0.61	NaN	NaN	NaN	46.200001	48.880001	NaN	57.340000	24.2500
215675	2009-06-01 00:00:00	NaN	0.33	NaN	NaN	NaN	75.980003	96.919998	NaN	43.139999	16.3400
215676	2009-06-01 00:00:00	NaN	0.35	NaN	NaN	NaN	40.799999	43.430000	NaN	71.209999	23.3899
215677	2009-06-01 00:00:00	NaN	0.25	NaN	NaN	NaN	45.299999	51.400002	NaN	62.939999	22.6299
215678	2009-06-01 00:00:00	NaN	0.40	NaN	NaN	NaN	87.239998	100.099998	NaN	27.410000	13.1200
215679	2009-06-01 00:00:00	NaN	0.21	NaN	NaN	NaN	39.650002	41.270000	NaN	66.870003	11.2700
215680	2009-06-01 00:00:00	NaN	0.51	NaN	NaN	NaN	21.750000	24.480000	NaN	84.900002	3.0500
215681	2009-06-01 00:00:00	NaN	0.32	NaN	NaN	NaN	62.630001	77.580002	NaN	49.529999	25.2700
215682	2009-06-01 00:00:00	0.41	0.30	0.37	NaN	0.18	75.290001	89.139999	NaN	33.330002	N
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	10.8300
215684	2009-06-01 00:00:00	NaN	0.31	NaN	NaN	NaN	76.110001	101.099998	NaN	41.220001	9.9200

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
215685	2009-06-01 00:00:00	0.13	NaN	0.86	NaN	0.23	81.050003	99.849998	NaN	24.830000	12.4600
215686	2009-06-01 00:00:00	0.21	NaN	2.96	NaN	0.10	72.419998	82.959999	NaN	NaN	13.0300
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	15.3600

In [5]:
```
data.describe()
```

Out[5]:

	BEN	CO	EBE	MXY	NMHC	NO_2	
count	60082.000000	190801.000000	60081.000000	24846.000000	74748.000000	214562.000000	2
mean	0.757749	0.393615	1.220672	2.248822	0.205894	54.345375	
std	1.011530	0.262863	1.266637	2.251823	0.124562	34.868690	
min	0.100000	0.060000	0.100000	0.240000	0.000000	0.600000	
25%	0.220000	0.240000	0.560000	0.990000	0.130000	28.379999	
50%	0.470000	0.320000	0.940000	1.490000	0.180000	47.599998	
75%	0.840000	0.470000	1.390000	2.830000	0.250000	72.339996	
max	37.720001	5.570000	81.480003	56.500000	4.330000	477.399994	

In [6]:
```
np.shape(data)
```

Out[6]: (215688, 17)

In [7]:
```
np.size(data)
```

Out[7]: 3666696

In [8]:
```
data.isna()
```

Out[8]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY
0	False	True	False	True	True	True	False	False	True	False	False	True	True
1	False	True	False	True	True	True	False	False	True	False	False	True	True
2	False	True	False	True	True	True	False	False	True	False	False	True	True
3	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	True	False	True	True	False	False	False	True	False	False	True	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...
215683	False	False	False	False	False	False	False	False	False	False	False	False	False
215684	False	True	False	True	True	True	False	False	True	False	False	True	True
215685	False	False	True	False	True	False	False	False	True	False	False	False	True
215686	False	False	True	False	True	False	False	False	True	True	False	True	True
215687	False	False	False	False	False	False	False	False	False	False	False	False	False

215688 rows × 17 columns

In [9]:
```
data.dropna()
```

Out[9]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM1
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	26.53000
20	2009-10-01 01:00:00	0.38	0.32	0.32	0.89	0.01	17.969999	19.240000	1.00	65.870003	10.52000
24	2009-10-01 01:00:00	0.55	0.24	0.65	1.79	0.18	36.619999	43.919998	1.28	48.070000	19.15000
28	2009-10-01 02:00:00	0.65	0.21	1.20	2.04	0.18	37.169998	48.869999	1.21	26.950001	32.20000
45	2009-10-01 02:00:00	0.38	0.30	0.50	1.15	0.00	17.889999	19.299999	1.00	60.009998	12.26000
...	...	...	...	...	...	...	...	...	...	...	.
215659	2009-05-31 23:00:00	0.54	0.27	1.00	0.69	0.09	28.280001	29.490000	0.86	78.750000	15.17000
215663	2009-05-31 23:00:00	0.74	0.35	1.13	1.65	0.15	56.410000	69.870003	1.26	56.799999	11.80000
215667	2009-06-01 00:00:00	0.78	0.29	0.99	1.96	0.04	64.870003	82.629997	1.13	58.000000	12.67000
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	10.83000
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	15.36000

24717 rows × 17 columns

In [10]:
```
data.columns
```

Out[10]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [11]:
```
sd=data[['BEN','CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx']]
```

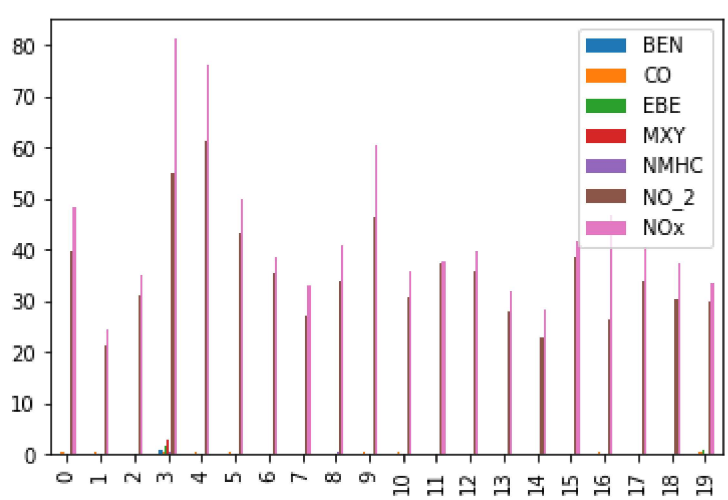
In [12]:
```
dd=sd.head(20)
dd
```

Out[12]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx
0	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002
1	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000
2	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001
3	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001
4	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002
5	NaN	0.29	NaN	NaN	NaN	43.200001	50.080002
6	NaN	0.20	NaN	NaN	NaN	35.430000	38.520000
7	NaN	0.15	NaN	NaN	NaN	27.309999	33.150002
8	NaN	0.21	NaN	NaN	0.39	33.889999	40.799999
9	NaN	0.32	NaN	NaN	NaN	46.349998	60.540001
10	NaN	0.24	NaN	NaN	NaN	30.860001	35.590000
11	NaN	0.18	NaN	NaN	NaN	37.230000	37.830002
12	NaN	0.19	NaN	NaN	NaN	35.680000	39.619999
13	NaN	NaN	NaN	NaN	NaN	28.000000	31.950001
14	NaN	0.17	NaN	NaN	NaN	22.629999	28.330000
15	NaN	0.21	NaN	NaN	NaN	38.340000	41.759998
16	NaN	0.26	NaN	NaN	NaN	26.209999	46.580002
17	NaN	0.21	NaN	NaN	NaN	33.759998	40.439999
18	NaN	0.21	NaN	NaN	NaN	30.180000	37.240002
19	0.20	0.25	0.62	NaN	0.13	29.930000	33.610001

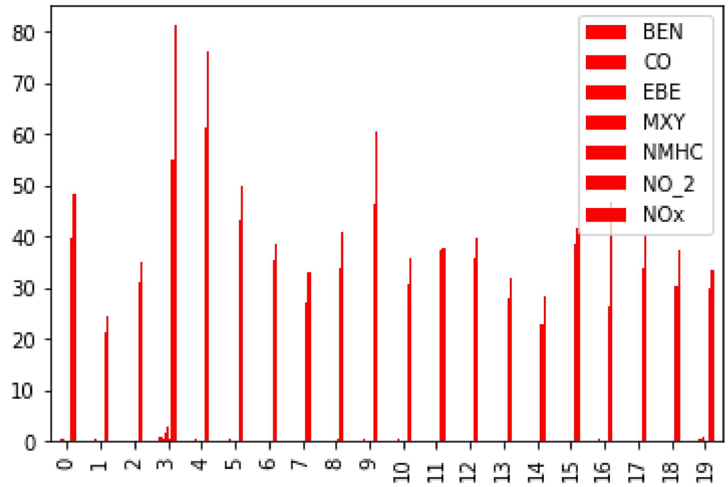
In [13]:
```
dd.plot.bar()
```

Out[13]: <AxesSubplot:>



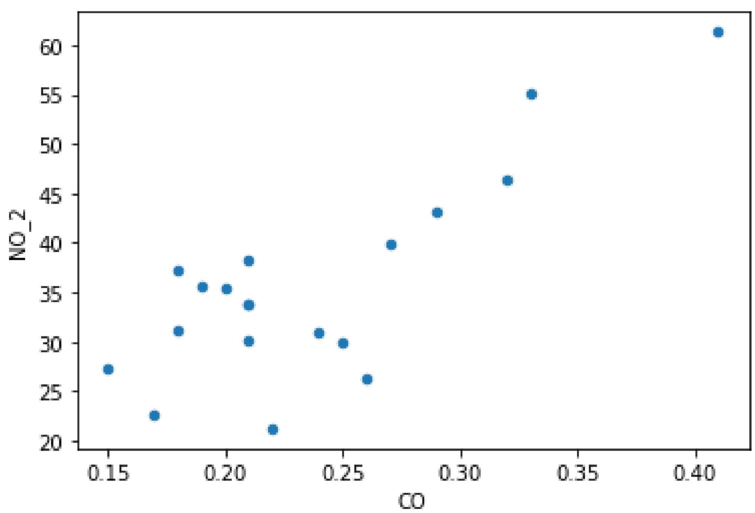
In [14]:
```
dd.plot.bar(color='r')
```

Out[14]: <AxesSubplot:>



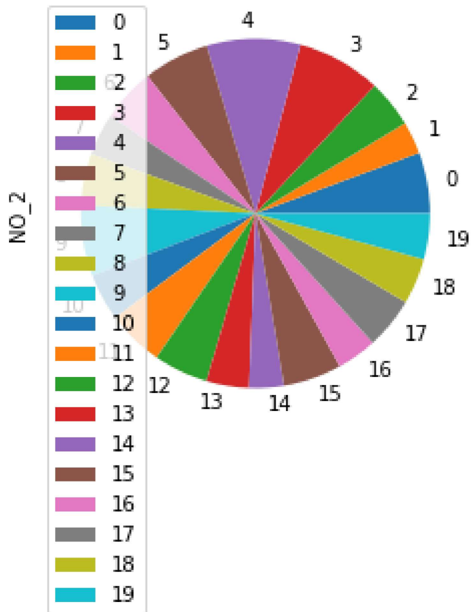
In [15]:
```
dd.plot.scatter(x='CO',y='NO_2')
```

Out[15]: <AxesSubplot:xlabel='CO', ylabel='NO_2'>



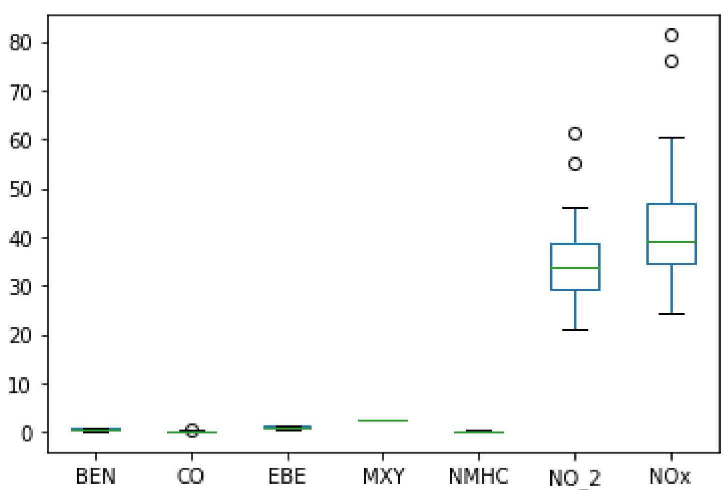
In [16]:
```
dd.plot.pie(y='NO_2')
```

Out[16]: <AxesSubplot:ylabel='NO_2'>



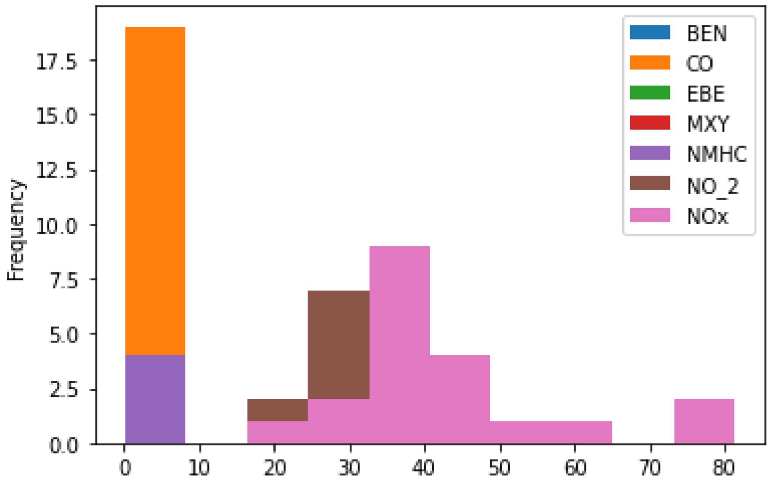
In [17]:
```
dd.plot.box()
```

Out[17]: <AxesSubplot:>



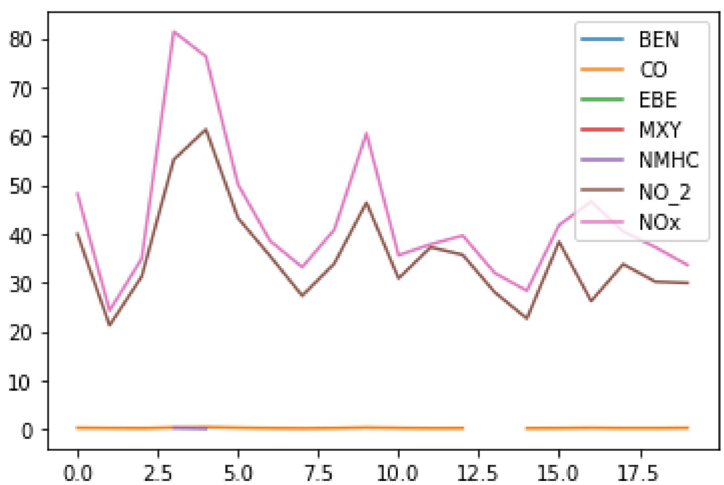
In [18]:
```
dd.plot.hist()
```

Out[18]: <AxesSubplot:ylabel='Frequency'>

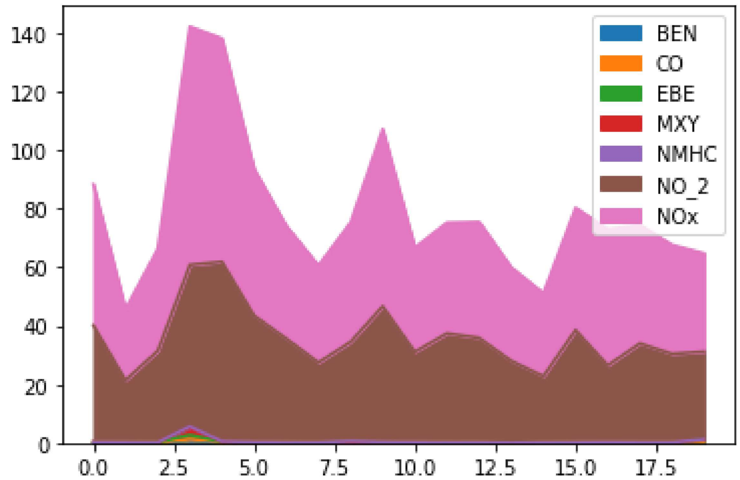


In [19]:
```
dd.plot.line()
```

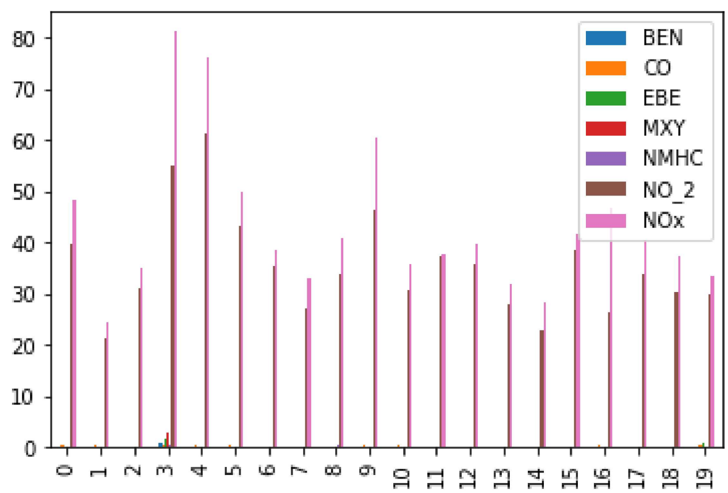
Out[19]: <AxesSubplot:>



In [20]:
```
dd.plot.area()
```
Out[20]: <AxesSubplot:>

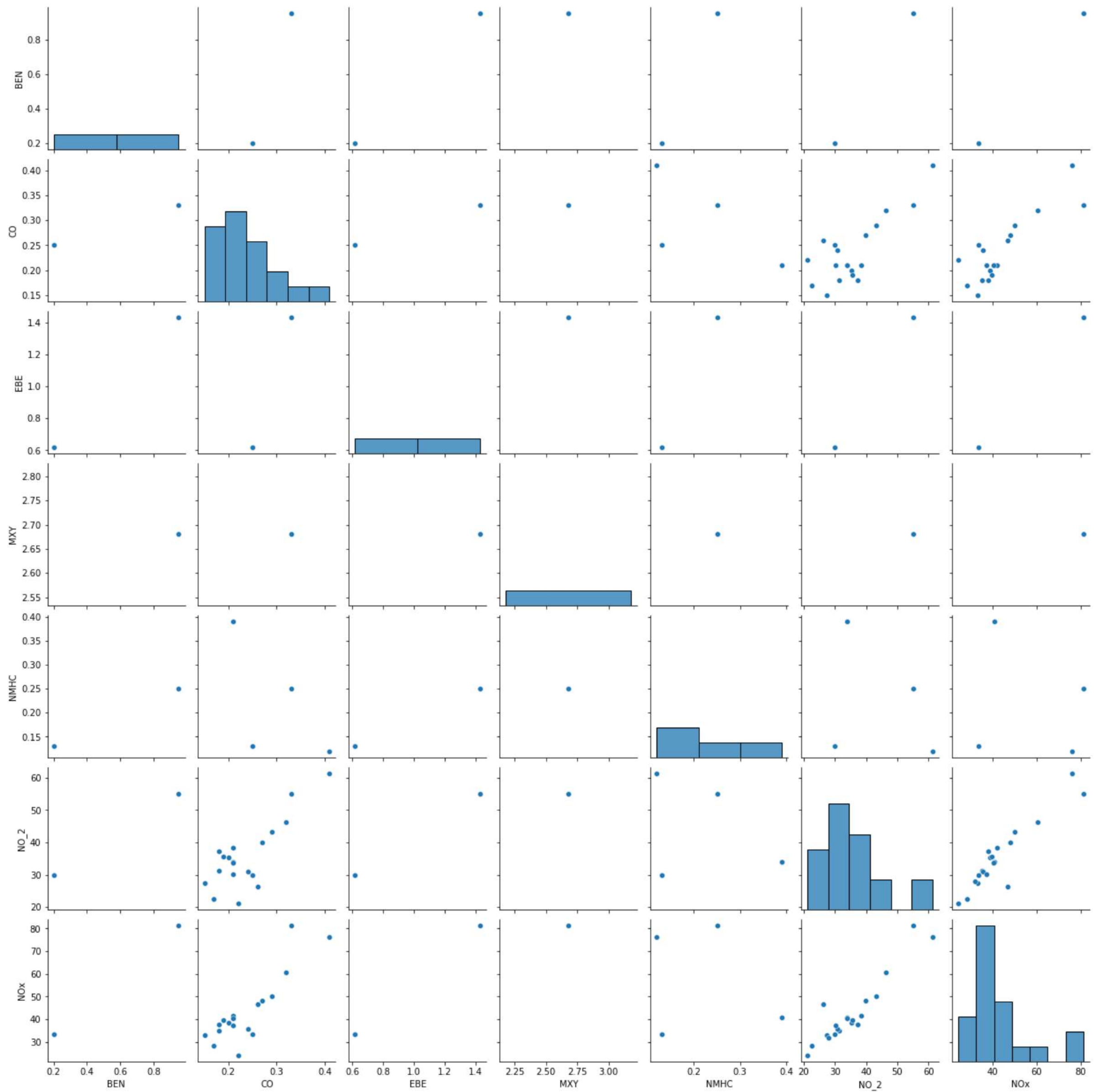


In [21]:
```
dd.plot.bar()
```
Out[21]: <AxesSubplot:>



In [22]:
```
sns.pairplot(dd)
```

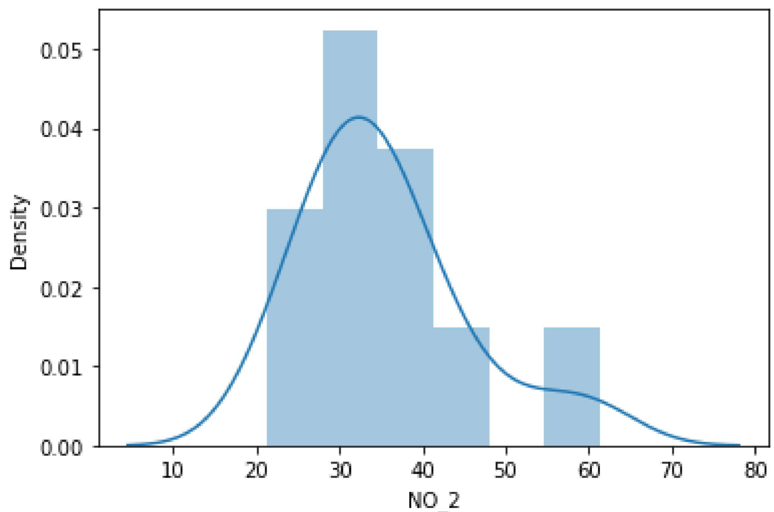
Out[22]: <seaborn.axisgrid.PairGrid at 0x1e10f1e0e80>



In [23]:
```
sns.distplot(dd['NO_2'])
```

```
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 function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[23]: <AxesSubplot:xlabel='NO_2', ylabel='Density'>



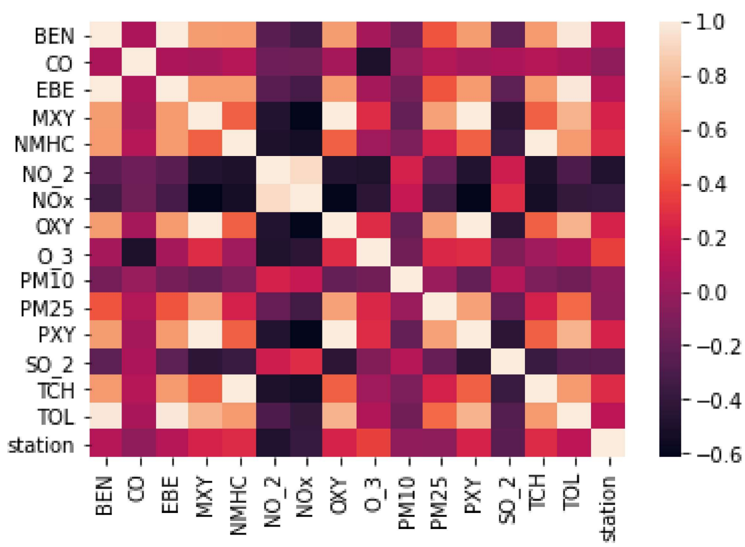
In [24]:
```
ds=data.fillna(20)
```

In [25]:
```
ssd=ds.head(20)
```

In [26]:
```
sd1=ssd[['BEN','CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx']]
```

In [27]:
```
sns.heatmap(ssd.corr())
```

Out[27]: <AxesSubplot:>



In [28]:
```
x= ssd[['BEN','CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx']]
y=ssd['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)
```

In [30]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[30]:
```
LinearRegression()
```

In [31]:
```
print(lr.intercept_)
```

```
28079032.65398972
```

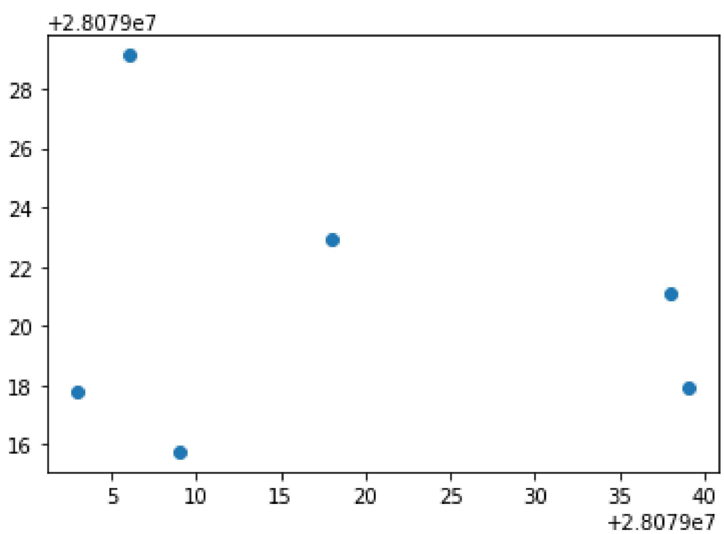
In [32]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[32]:

	Co-efficient
BEN	-0.316216
CO	-0.123839
EBE	-0.309508
MXY	0.000000
NMHC	0.386665
NO_2	-1.156499
NOx	0.749654

In [33]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[33]:
```
<matplotlib.collections.PathCollection at 0x1e115feb760>
```



In [34]:
```
print(lr.score(x_test,y_test))
```
```
-0.2086749635985623
```

In [35]:
```
lr.score(x_test,y_test)
```
Out[35]:
```
-0.2086749635985623
```

In [36]:
```
lr.score(x_train,y_train)
```
Out[36]:
```
0.346321238823563
```

In [37]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [38]:
```
dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```
Out[38]:
```
Ridge(alpha=10)
```

In [39]:
```
dr.score(x_test,y_test)
```
Out[39]:
```
-0.1668064167808725
```

In [40]:
```
dr.score(x_train,y_train)
```
Out[40]:
```
0.3453487068753818
```

In [41]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[41]:
```
Lasso(alpha=10)
```

In [42]:
```
la.score(x_test,y_test)
```
Out[42]:
```
0.21225155924918837
```

In [43]:
```
la.score(x_train,y_train)
```
Out[43]:
```
0.15037239677932068
```

ElasticNet

In [44]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[44]:
```
ElasticNet()
```

In [45]:
```
print(en.coef_)
```

```
[-0.29468731 -0.10936129 -0.26206997  0.          0.35004509 -1.052275
  0.65896394]
```

In [46]:
```
print(en.intercept_)
```

```
28079032.09424193
```

In [47]:
```
prediction=en.predict(x_test)
```

In [48]:
```
print(en.score(x_test,y_test))
```

```
-0.1318676264866061
```

In [49]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [50]:
```
from sklearn.linear_model import LogisticRegression
```

In [51]:
```
feature_matrix = ssd[['BEN','CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx']]
target_vector=ssd['station']
```

In [52]:
```
feature_matrix.shape
```

Out[52]:
```
(20, 7)
```

In [53]:
```
target_vector.shape
```

Out[53]:
```
(20,)
```

In [54]:
```
from sklearn.preprocessing import StandardScaler
```

In [55]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [56]:
```
logr= LogisticRegression()
logr.fit(fs,target_vector)
```

Out[56]:
```
LogisticRegression()
```

In [57]:
```
observation =[[1.2,2.3,3.3,4.3,5.3,6.3,7.3]]
```

In [58]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079012]
```

In [59]:
```
logr.classes_
```
Out[59]:
```
array([28079003, 28079004, 28079006, 28079007, 28079008, 28079009,
       28079011, 28079012, 28079014, 28079016, 28079017, 28079018,
       28079019, 28079021, 28079022, 28079023, 28079036, 28079038,
       28079039, 28079040], dtype=int64)
```

In [60]:
```
logr.predict_proba(observation)[0][0]
```
Out[60]:
```
0.0036763862551488853
```

In [61]:
```
ged=data[['BEN','CO','EBE','MXY','NMHC','NO_2','NOx','OXY','O_3','PM10','PXY','
```

In [62]:
```
d=ged.fillna(20)
```

In [63]:
```
dg=d.head(100)
```

In [64]:
```
x=dg[['BEN','CO','EBE','MXY','NMHC','NO_2','NOx','OXY','O_3','PM10','PXY','SO_2
y=dg['station']
```

In [65]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70)
```

In [66]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```
Out[66]:
```
RandomForestClassifier()
```

In [67]:
```
paramets = {'max_depth':[1,2,3,4,5,6,7],
              'min_samples_leaf':[5,10,15,20,25,30,35],
              'n_estimators':[10,20,30,40,50,60,70]}
```

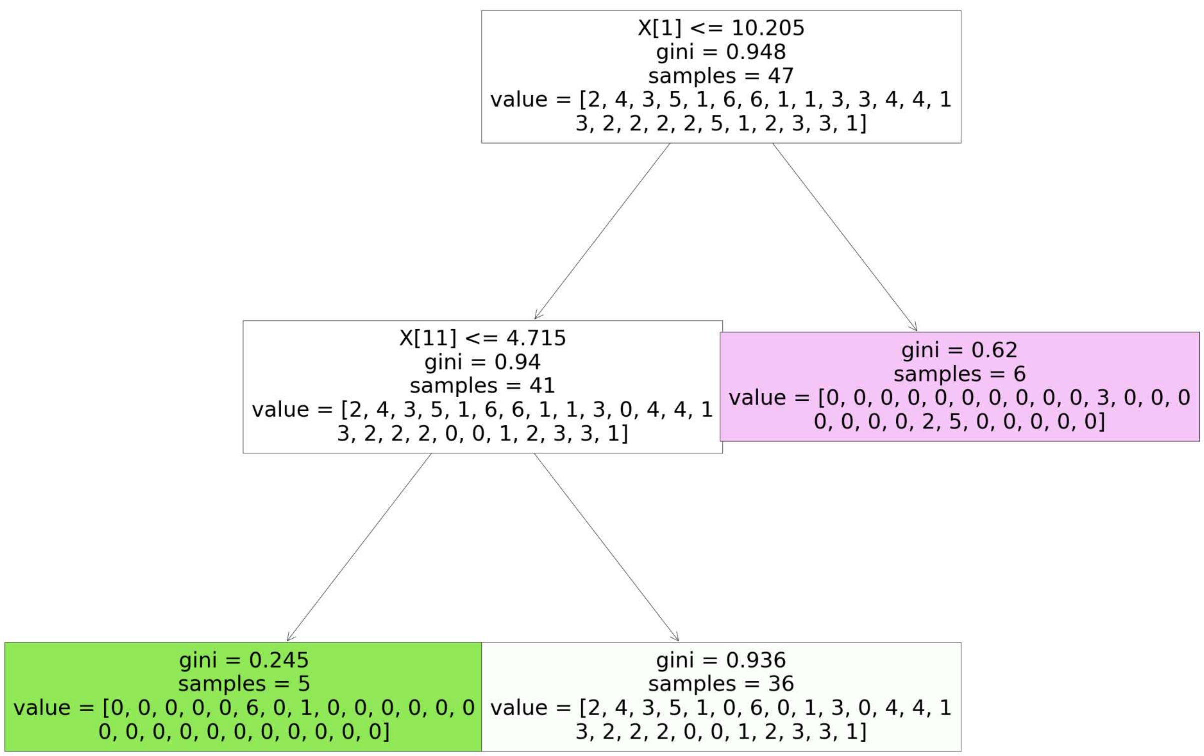
In [68]:
```
from sklearn.model_selection import GridSearchCV
grid_search= GridSearchCV(estimator = rfc,param_grid=paramets,cv=2,scoring="acc
grid_search.fit(x_train,y_train)
```
```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:
666: UserWarning: The least populated class in y has only 1 members, which is
less than n_splits=2.
  warnings.warn(("The least populated class in y has only %d"
```
Out[68]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5, 6, 7],
                         'min_samples_leaf': [5, 10, 15, 20, 25, 30, 35],
                         'n_estimators': [10, 20, 30, 40, 50, 60, 70]},
             scoring='accuracy')
```

In [69]:
```
grid_search.best_score_
```
Out[69]:
```
0.3
```

In [70]:
```
rfc_best=grid_search.best_estimator_
```

In [71]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(50,40))
plot_tree(rfc_best.estimators_[5],filled=True)
```

Out[71]:
```
[Text(1674.0, 1812.0, 'X[1] <= 10.205\ngini = 0.948\nsamples = 47\nvalue =
[2, 4, 3, 5, 1, 6, 6, 1, 1, 3, 3, 4, 4, 1\n3, 2, 2, 2, 2, 5, 1, 2, 3, 3,
1]'),
 Text(1116.0, 1087.2, 'X[11] <= 4.715\ngini = 0.94\nsamples = 41\nvalue = [2,
4, 3, 5, 1, 6, 6, 1, 1, 3, 0, 4, 4, 1\n3, 2, 2, 2, 0, 0, 1, 2, 3, 3, 1]'),
 Text(558.0, 362.39999999999986, 'gini = 0.245\nsamples = 5\nvalue = [0, 0,
0, 0, 0, 6, 0, 1, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(1674.0, 362.39999999999986, 'gini = 0.936\nsamples = 36\nvalue = [2, 4,
3, 5, 1, 0, 6, 0, 1, 3, 0, 4, 4, 1\n3, 2, 2, 2, 0, 0, 1, 2, 3, 3, 1]'),
 Text(2232.0, 1087.2, 'gini = 0.62\nsamples = 6\nvalue = [0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 3, 0, 0, 0\n0, 0, 0, 0, 2, 5, 0, 0, 0, 0, 0]')]
```



Conclusion : LinearRegression() 28079032.65398972 HIGH RANGE

In [ ]: