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_2010.csv")
data
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	I
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	I
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	I
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.410
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.670
...	...	...	...	...	...	...	...	...	...	...	
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999	I
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN	51.259
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000	I
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999	I
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998	47.150

209448 rows × 17 columns

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

Out[3]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	NaN	
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	NaN	
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	NaN	
3	2010-03-01 01:00:00	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000	NaN	72.970001	19.410000	7
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.670000	22
5	2010-03-01 01:00:00	0.56	NaN	0.58	NaN	NaN	21.370001	25.870001	NaN	NaN	NaN	
6	2010-03-01 01:00:00	NaN	NaN	NaN	NaN	NaN	16.660000	25.230000	NaN	NaN	39.799999	
7	2010-03-01 01:00:00	NaN	0.23	NaN	NaN	NaN	17.799999	21.639999	NaN	55.880001	NaN	
8	2010-03-01 01:00:00	NaN	NaN	NaN	NaN	NaN	12.050000	14.870000	NaN	57.369999	NaN	
9	2010-03-01 01:00:00	1.48	0.18	0.51	NaN	NaN	16.780001	21.680000	NaN	78.660004	21.969999	

In [4]:

```
data.tail(20)
```

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
209428	2010-08-01 00:00:00	NaN	0.67	NaN	NaN	NaN	113.000000	147.699997	NaN	25.450001	I
209429	2010-08-01 00:00:00	1.34	0.49	1.18	NaN	0.16	115.099998	124.900002	NaN	31.950001	61.060
209430	2010-08-01 00:00:00	0.50	NaN	0.55	NaN	NaN	48.430000	53.509998	NaN	NaN	52.959
209431	2010-08-01 00:00:00	2.08	NaN	0.96	NaN	NaN	48.509998	53.470001	NaN	NaN	I
209432	2010-08-01 00:00:00	NaN	0.41	NaN	NaN	NaN	NaN	NaN	NaN	55.490002	I
209433	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	75.620003	83.419998	NaN	45.580002	I
209434	2010-08-01 00:00:00	0.18	0.35	0.25	NaN	NaN	53.290001	58.099998	NaN	76.870003	36.419
209435	2010-08-01 00:00:00	NaN	0.46	NaN	NaN	NaN	99.550003	108.099998	NaN	NaN	35.299
209436	2010-08-01 00:00:00	0.87	0.25	1.17	NaN	0.16	38.369999	39.790001	NaN	82.809998	40.180
209437	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.29	79.970001	83.709999	NaN	60.939999	I
209438	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	89.279999	90.250000	NaN	NaN	46.230
209439	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	89.220001	97.540001	NaN	NaN	45.950
209440	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	55.310001	58.119999	NaN	49.270000	I
209441	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	66.470001	67.860001	NaN	56.090000	I
209442	2010-08-01 00:00:00	0.57	NaN	0.16	NaN	0.38	113.800003	132.699997	NaN	NaN	54.970
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999	I
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN	51.259

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PI
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000	I
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999	I
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998	47.150

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

Out[5]:

	BEN	CO	EBE	MXY	NMHC	NO_2	
count	60268.000000	94982.000000	60253.000000	6750.000000	51727.000000	208219.000000	208:
mean	0.773250	0.357195	1.075482	0.839812	0.209493	44.273986	
std	0.816808	0.214328	1.168859	0.382826	0.088662	30.612227	
min	0.100000	0.060000	0.100000	0.110000	0.000000	0.600000	
25%	0.220000	0.230000	0.270000	0.590000	0.150000	21.250000	
50%	0.490000	0.300000	0.730000	1.000000	0.210000	37.259998	
75%	0.970000	0.410000	1.320000	1.000000	0.250000	60.470001	
max	13.850000	4.190000	19.980000	6.810000	1.500000	435.399994	1.

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

Out[6]:
```
(209448, 17)
```

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

Out[7]:
```
3560616
```

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

Out[8]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	S
0	False	True	False	True	True	True	False	False	True	False	True	True	True	F
1	False	True	False	True	True	True	False	False	True	True	True	True	True	F
2	False	True	False	True	True	True	False	False	True	False	True	True	True	-
3	False	False	False	False	True	False	False	False	True	False	False	False	True	F
4	False	False	True	False	True	True	False	False	True	True	False	False	True	F
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
209443	False	True	False	True	True	True	False	False	True	False	True	True	True	-
209444	False	True	False	True	True	True	False	False	True	True	False	True	True	F
209445	False	True	True	True	True	False	False	False	True	False	True	True	True	-
209446	False	True	True	True	True	True	False	False	True	False	True	True	True	-
209447	False	False	False	False	True	False	False	False	True	False	False	False	True	F

209448 rows × 17 columns

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

Out[9]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM1
11	2010-03-01 01:00:00	0.78	0.18	0.84	0.73	0.28	10.420000	11.900000	1.0	90.309998	18.37000
23	2010-03-01 01:00:00	0.70	0.23	1.00	0.73	0.18	17.820000	22.290001	1.0	70.550003	23.63999
35	2010-03-01 02:00:00	0.58	0.17	0.84	0.73	0.28	3.500000	4.950000	1.0	68.849998	5.60000
47	2010-03-01 02:00:00	0.33	0.21	0.84	0.73	0.17	10.810000	14.900000	1.0	74.750000	7.89000
59	2010-03-01 03:00:00	0.38	0.16	0.64	1.00	0.26	2.750000	4.200000	1.0	93.629997	5.13000
...	...	...	...	...	...	...	...	...	...	...	.
191879	2010-05-31 22:00:00	0.60	0.26	0.82	0.13	0.16	33.360001	43.779999	1.0	38.459999	20.34000
191891	2010-05-31 23:00:00	0.41	0.16	0.71	0.19	0.10	24.299999	26.059999	1.0	50.290001	14.38000
191903	2010-05-31 23:00:00	0.57	0.28	0.64	0.19	0.18	35.540001	44.590000	1.0	34.020000	22.84000
191915	2010-06-01 00:00:00	0.34	0.16	0.69	0.22	0.10	23.559999	25.209999	1.0	45.930000	10.77000
191927	2010-06-01 00:00:00	0.43	0.25	0.79	0.22	0.18	34.910000	42.369999	1.0	29.540001	15.35000

6666 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']]
```

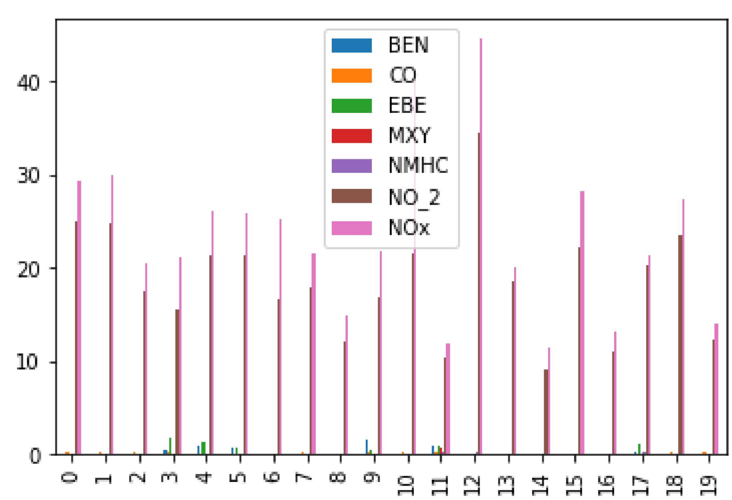
In [12]:
```
dd=sd.head(20)
dd
```

Out[12]:

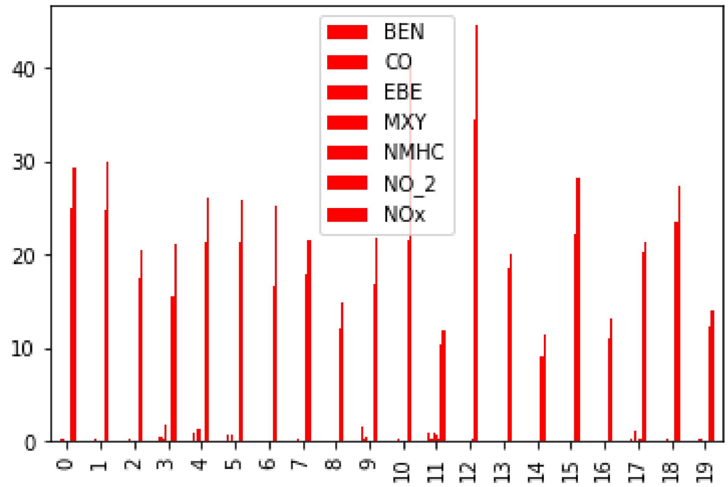
	BEN	CO	EBE	MXY	NMHC	NO_2	NOx
0	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999
1	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001
2	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001
3	0.38	0.24	1.74	NaN	0.05	15.610000	21.080000
4	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000
5	0.56	NaN	0.58	NaN	NaN	21.370001	25.870001
6	NaN	NaN	NaN	NaN	NaN	16.660000	25.230000
7	NaN	0.23	NaN	NaN	NaN	17.799999	21.639999
8	NaN	NaN	NaN	NaN	NaN	12.050000	14.870000
9	1.48	0.18	0.51	NaN	NaN	16.780001	21.680000
10	NaN	0.22	NaN	NaN	NaN	21.450001	40.029999
11	0.78	0.18	0.84	0.73	0.28	10.420000	11.900000
12	NaN	NaN	NaN	NaN	0.14	34.369999	44.590000
13	NaN	NaN	NaN	NaN	NaN	18.620001	20.139999
14	NaN	NaN	NaN	NaN	NaN	9.040000	11.360000
15	NaN	NaN	NaN	NaN	NaN	22.150000	28.299999
16	NaN	NaN	NaN	NaN	NaN	11.070000	13.140000
17	0.21	NaN	1.00	NaN	0.22	20.290001	21.240000
18	NaN	0.20	NaN	NaN	NaN	23.410000	27.379999
19	NaN	0.20	NaN	NaN	NaN	12.230000	14.010000

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

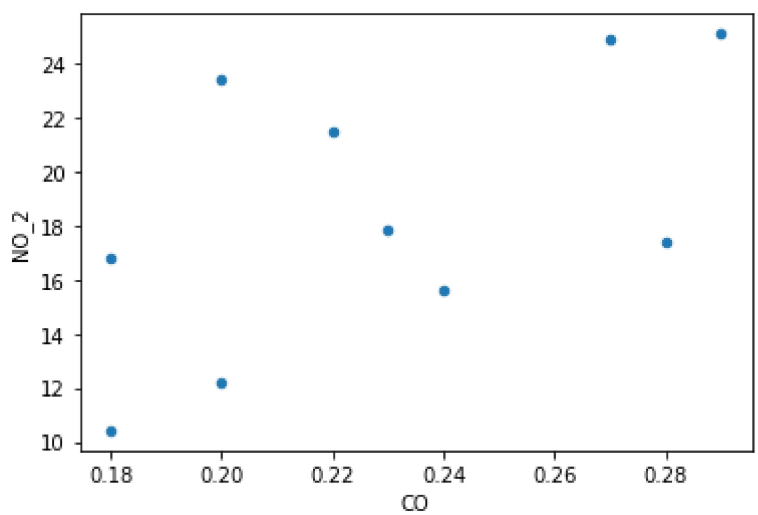
Out[13]: <AxesSubplot:>



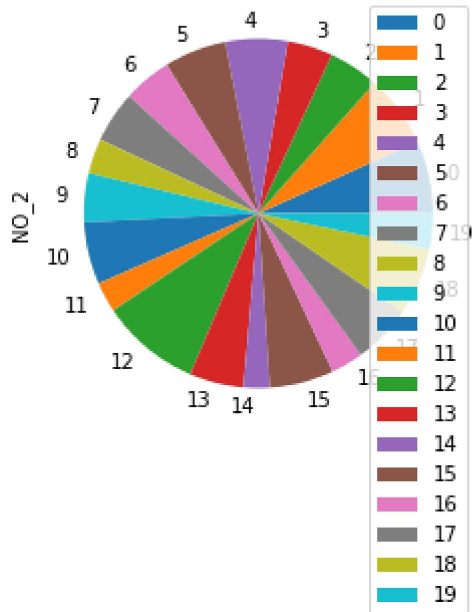
In [14]:
```
dd.plot.bar(color='r')
```
Out[14]: <AxesSubplot:>



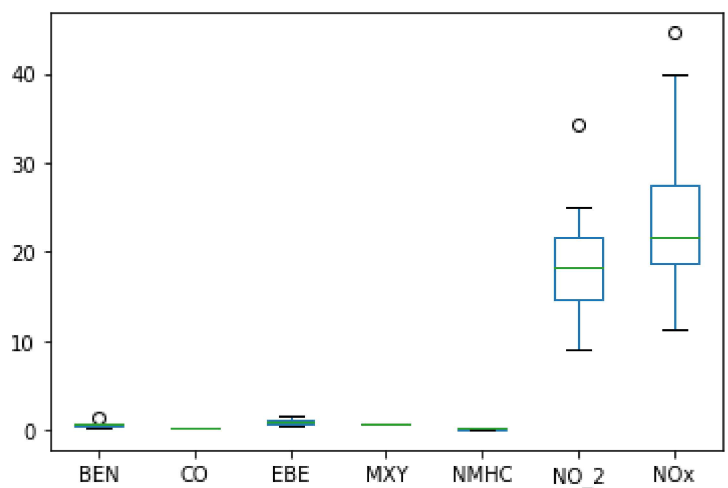
In [15]:
```
dd.plot.scatter(x='CO',y='NO_2')
```
Out[15]: <AxesSubplot:xlabel='CO', ylabel='NO_2'>



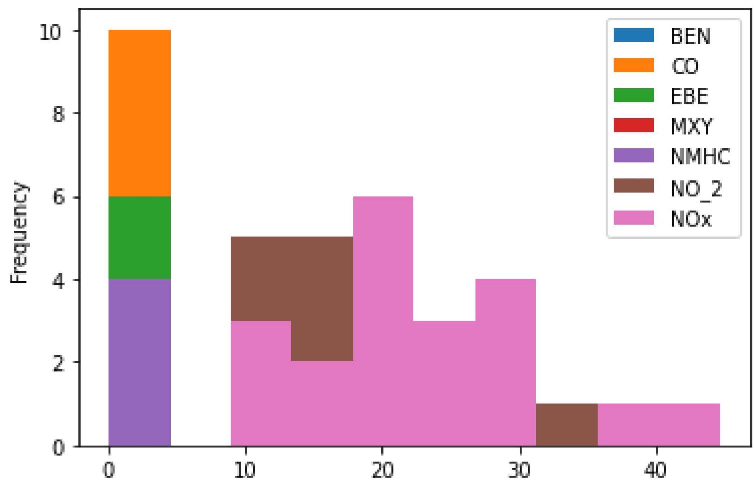
In [16]:
```
dd.plot.pie(y='NO_2')
```
Out[16]: <AxesSubplot:ylabel='NO_2'>



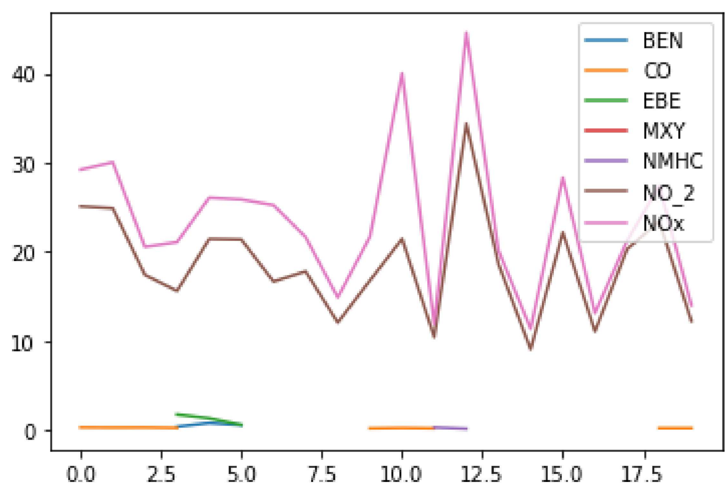
In [17]:
```
dd.plot.box()
```
Out[17]: <AxesSubplot:>



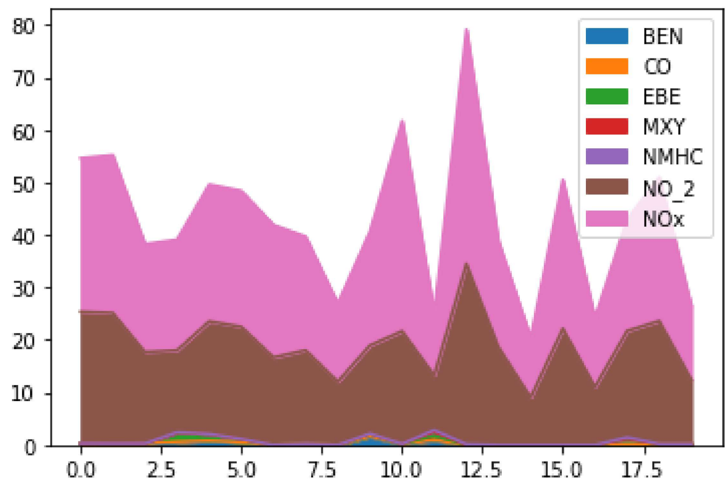
In [18]:
```
dd.plot.hist()
```
Out[18]: <AxesSubplot:ylabel='Frequency'>



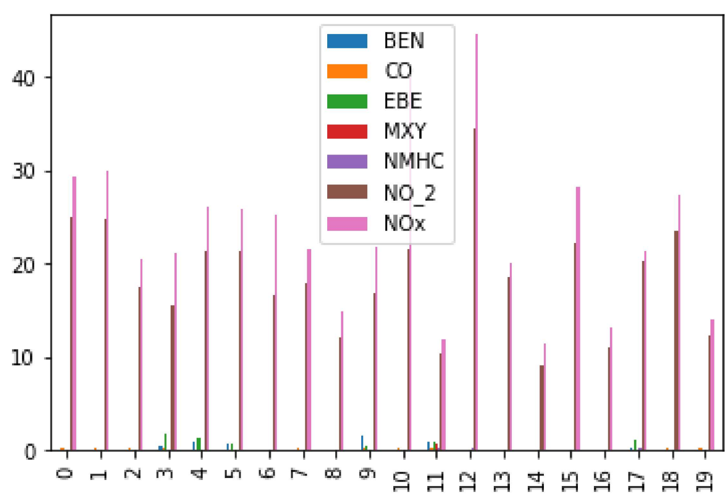
In [19]:
```
dd.plot.line()
```
Out[19]: <AxesSubplot:>



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

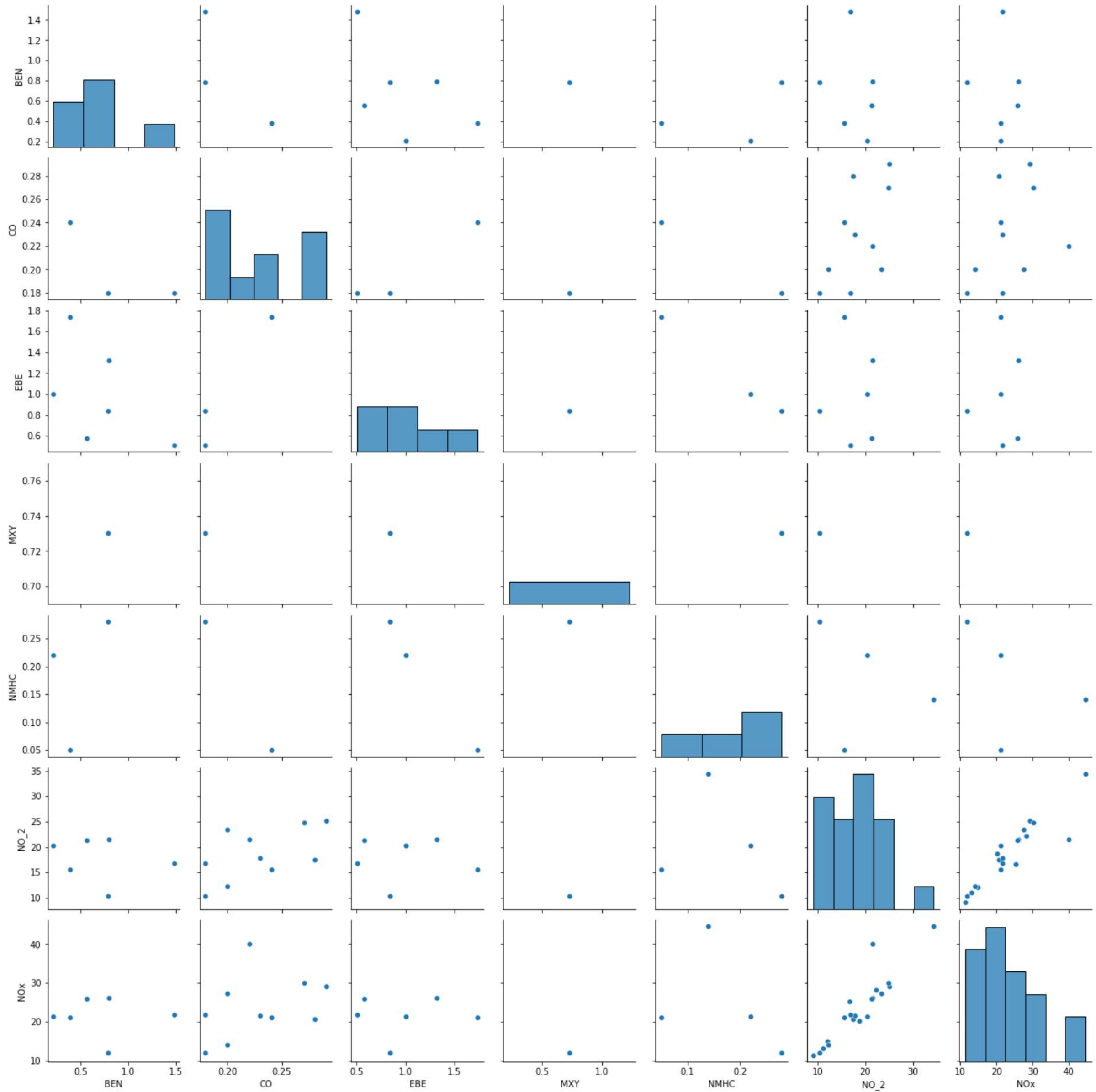


In [21]:
```
dd.plot.bar()
```
Out[21]: <AxesSubplot:>



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

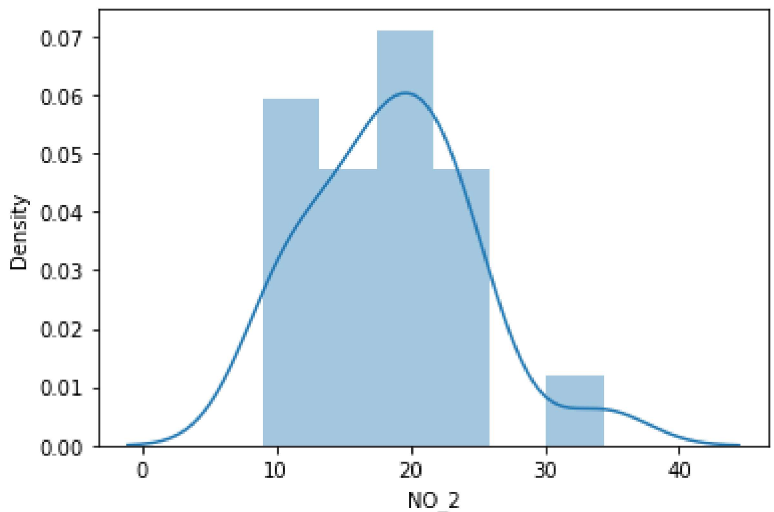
Out[22]: <seaborn.axisgrid.PairGrid at 0x1c11f5c7f70>



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

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



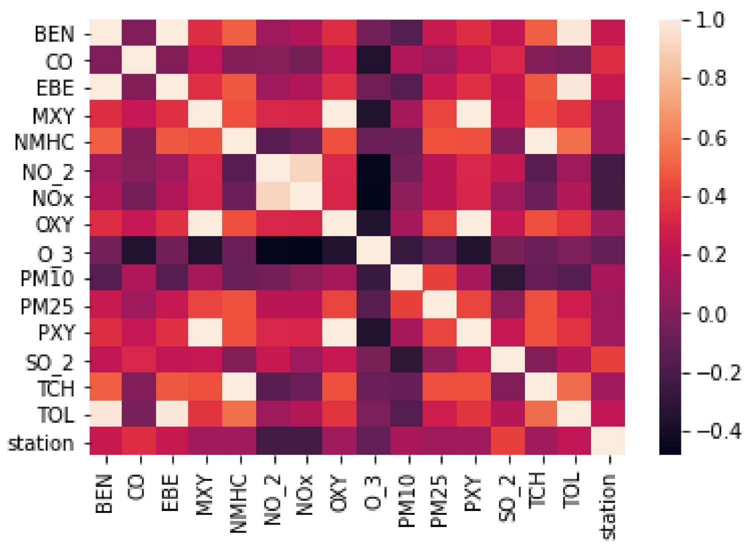
In [24]:
```
ds=data.fillna(20)
```

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_)
```

28079031.6903269

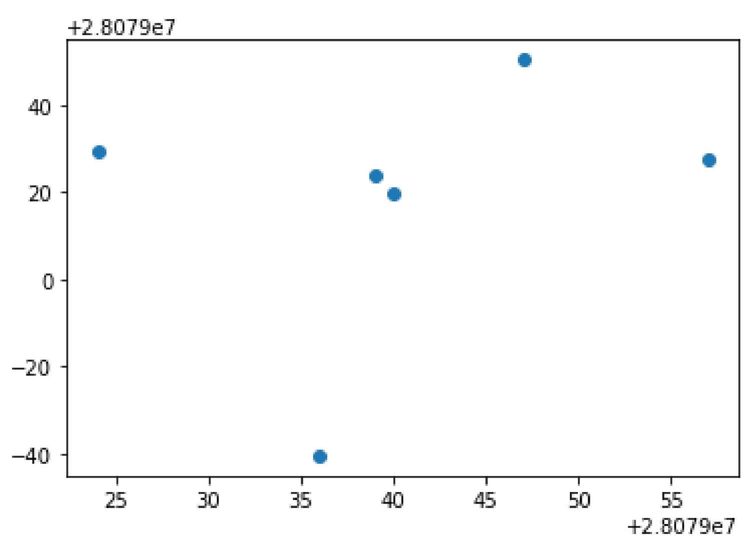
In [32]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[32]:

	Co-efficient
BEN	4.339156e+00
CO	9.668564e-01
EBE	-3.890120e+00
MXY	-8.881784e-16
NMHC	-5.065505e-01
NO_2	4.642079e+00
NOx	-4.267292e+00

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

Out[33]: <matplotlib.collections.PathCollection at 0x1c1261f2d30>



In [34]:
```
print(lr.score(x_test,y_test))
```
```
-11.130051414973929
```

In [35]:
```
lr.score(x_test,y_test)
```
Out[35]:
```
-11.130051414973929
```

In [36]:
```
lr.score(x_train,y_train)
```
Out[36]:
```
0.3889425669802936
```

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]:
```
-5.093006568470381
```

In [40]:
```
dr.score(x_train,y_train)
```
Out[40]:
```
0.36768285485339114
```

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]:
```
-2.2092382739113536
```

In [43]:
```
la.score(x_train,y_train)
```
Out[43]:
```
0.29764902585864284
```

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_)
```

```
[ 0.41982734  0.95495587  0.          0.         -0.29013345  2.21989258
 -2.27402603]
```

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

```
28079028.05308662
```

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

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

```
-5.050765386084293
```

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)
```

```
[28079050]
```

In [59]:
```
logr.classes_
```
Out[59]:
```
array([28079003, 28079004, 28079008, 28079011, 28079016, 28079017,
       28079018, 28079024, 28079027, 28079036, 28079038, 28079039,
       28079040, 28079047, 28079049, 28079050, 28079054, 28079055,
       28079056, 28079057], dtype=int64)
```
In [60]:
```
logr.predict_proba(observation)[0][0]
```
Out[60]:
```
0.04466678403915628
```
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]:
```
print(len(x))
print(len(y))
```
```
100
100
```
In [66]:
```
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 [67]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```
Out[67]:
```
RandomForestClassifier()
```
In [68]:
```
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 [69]:
```
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[69]:
```
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')
```

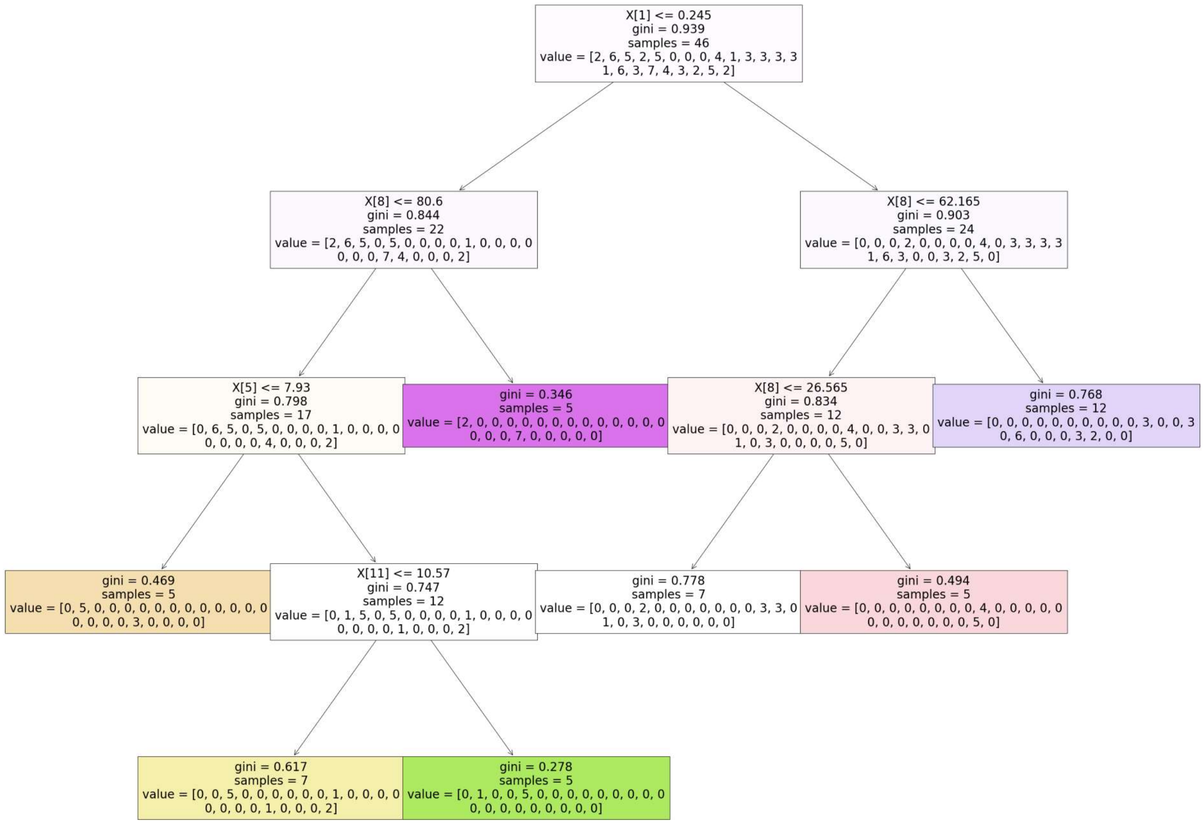
In [70]:
```
grid_search.best_score_
```

Out[70]:
```
0.48571428571428565
```

In [71]:
```
rfc_best=grid_search.best_estimator_
```

In [72]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(50,40))
plot_tree(rfc_best.estimators_[5],filled=True)
```

Out[72]:
```
[Text(1550.0, 1956.96, 'X[1] <= 0.245\ngini = 0.939\nsamples = 46\nvalue =
[2, 6, 5, 2, 5, 0, 0, 0, 4, 1, 3, 3, 3, 3\n1, 6, 3, 7, 4, 3, 2, 5, 2]'),
 Text(930.0, 1522.0800000000002, 'X[8] <= 80.6\ngini = 0.844\nsamples = 22\nv
alue = [2, 6, 5, 0, 5, 0, 0, 0, 0, 1, 0, 0, 0, 0\n0, 0, 0, 7, 4, 0, 0, 0,
2]'),
 Text(620.0, 1087.2, 'X[5] <= 7.93\ngini = 0.798\nsamples = 17\nvalue = [0,
6, 5, 0, 5, 0, 0, 0, 0, 1, 0, 0, 0, 0\n0, 0, 0, 0, 4, 0, 0, 0, 2]'),
 Text(310.0, 652.3200000000002, 'gini = 0.469\nsamples = 5\nvalue = [0, 5, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 3, 0, 0, 0, 0]'),
 Text(930.0, 652.3200000000002, 'X[11] <= 10.57\ngini = 0.747\nsamples = 12\n
value = [0, 1, 5, 0, 5, 0, 0, 0, 0, 1, 0, 0, 0, 0\n0, 0, 0, 0, 1, 0, 0, 0,
2]'),
 Text(620.0, 217.44000000000005, 'gini = 0.617\nsamples = 7\nvalue = [0, 0,
5, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0\n0, 0, 0, 0, 1, 0, 0, 0, 2]'),
 Text(1240.0, 217.44000000000005, 'gini = 0.278\nsamples = 5\nvalue = [0, 1,
0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 0, 0]'),
 Text(1240.0, 1087.2, 'gini = 0.346\nsamples = 5\nvalue = [2, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 7, 0, 0, 0, 0, 0]'),
 Text(2170.0, 1522.0800000000002, 'X[8] <= 62.165\ngini = 0.903\nsamples = 24
\nvalue = [0, 0, 0, 2, 0, 0, 0, 0, 4, 0, 3, 3, 3, 3\n1, 6, 3, 0, 0, 3, 2, 5,
0]'),
 Text(1860.0, 1087.2, 'X[8] <= 26.565\ngini = 0.834\nsamples = 12\nvalue =
[0, 0, 0, 2, 0, 0, 0, 0, 4, 0, 0, 3, 3, 0\n1, 0, 3, 0, 0, 0, 0, 5, 0]'),
 Text(1550.0, 652.3200000000002, 'gini = 0.778\nsamples = 7\nvalue = [0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 3, 3, 0\n1, 0, 3, 0, 0, 0, 0, 0, 0]'),
 Text(2170.0, 652.3200000000002, 'gini = 0.494\nsamples = 5\nvalue = [0, 0,
0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 0, 0, 5, 0]'),
 Text(2480.0, 1087.2, 'gini = 0.768\nsamples = 12\nvalue = [0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 3, 0, 0, 3\n0, 6, 0, 0, 0, 3, 2, 0, 0]')]
```



Conclusion : RandomForestClassifier() 0.48571428571428565 HIGH RANGE

In [ ]: