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

In [2]:
```
data=pd.read_csv(r"C:\Users\user\Desktop\DINESH\C10_air\madrid_2002.csv")
data
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.54	41.990002
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	20.980000
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.01	28.440001
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.12	42.180000
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.28	76.330002
...	...	...	...	...	...	...	...	...	...	...	...
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750000
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.389999
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	9.640000
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	NaN
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	29.240000

217296 rows × 16 columns

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

Out[3]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2002-04-01 01:00:00	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006	NaN	6.540000	41.990002
1	2002-04-01 01:00:00	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994	2.67	6.850000	20.980000
2	2002-04-01 01:00:00	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000	NaN	13.010000	28.440001
3	2002-04-01 01:00:00	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000	NaN	5.120000	42.180000
4	2002-04-01 01:00:00	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997	NaN	7.280000	76.330002
5	2002-04-01 01:00:00	3.19	0.72	3.23	7.65	0.11	113.699997	187.000000	3.53	12.370000	27.450001
6	2002-04-01 01:00:00	NaN	0.78	NaN	NaN	0.09	101.000000	119.300003	NaN	20.549999	23.950001
7	2002-04-01 01:00:00	NaN	1.06	NaN	NaN	NaN	127.300003	204.100006	NaN	3.150000	49.639999
8	2002-04-01 01:00:00	NaN	1.21	NaN	NaN	NaN	106.300003	126.599998	NaN	22.389999	32.090000
9	2002-04-01 01:00:00	NaN	0.61	NaN	NaN	0.14	95.540001	110.699997	NaN	27.770000	24.610001

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

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
217276	2002-11-01 00:00:00	NaN	1.20	NaN	NaN	NaN	81.919998	184.699997	NaN	5.29	21.110001
217277	2002-11-01 00:00:00	NaN	1.16	NaN	NaN	0.29	81.620003	165.699997	NaN	10.97	25.250000
217278	2002-11-01 00:00:00	NaN	1.25	NaN	NaN	NaN	83.660004	166.300003	NaN	1.97	16.389999
217279	2002-11-01 00:00:00	NaN	1.61	NaN	NaN	NaN	67.910004	120.900002	NaN	8.23	25.299999
217280	2002-11-01 00:00:00	NaN	1.02	NaN	NaN	0.27	95.120003	198.500000	NaN	8.92	31.930000
217281	2002-11-01 00:00:00	NaN	1.08	NaN	NaN	NaN	75.209999	104.699997	NaN	7.52	22.240000
217282	2002-11-01 00:00:00	NaN	1.01	NaN	NaN	0.25	69.610001	116.000000	NaN	6.96	27.860001
217283	2002-11-01 00:00:00	NaN	2.25	NaN	NaN	NaN	67.750000	162.199997	NaN	12.20	38.080002
217284	2002-11-01 00:00:00	2.54	1.12	2.48	NaN	0.23	69.739998	214.699997	NaN	7.48	39.340000
217285	2002-11-01 00:00:00	NaN	1.24	NaN	NaN	NaN	80.160004	182.800003	NaN	8.53	23.290001
217286	2002-11-01 00:00:00	NaN	1.16	NaN	NaN	NaN	63.959999	184.399994	NaN	6.13	32.040001
217287	2002-11-01 00:00:00	NaN	1.09	NaN	NaN	0.32	92.970001	179.899994	NaN	6.28	43.709999
217288	2002-11-01 00:00:00	NaN	1.18	NaN	NaN	NaN	76.480003	287.299988	NaN	3.03	8.540000
217289	2002-11-01 00:00:00	NaN	1.09	NaN	NaN	NaN	70.580002	149.500000	NaN	2.67	37.770000
217290	2002-11-01 00:00:00	NaN	0.69	NaN	NaN	NaN	62.490002	127.099998	NaN	6.96	23.610001
217291	2002-11-01 00:00:00	4.16	1.14	NaN	NaN	NaN	81.080002	265.700012	NaN	7.21	36.750000
217292	2002-11-01 00:00:00	3.67	1.73	2.89	NaN	0.38	113.900002	373.100006	NaN	5.66	63.389999

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
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	9.640000
217294	2002-11-01 00:00:00	4.51	0.91	4.83	10.99	NaN	149.800003	202.199997	1.00	5.75	NaN
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	29.240000

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

Out[5]:

	BEN	CO	EBE	MXY	NMHC	NO_2	
count	66747.000000	216637.000000	58547.000000	41255.000000	87045.000000	216439.000000	2
mean	2.344545	0.760415	2.528299	6.485316	0.175150	61.659558	
std	2.547749	0.722964	2.637413	6.685265	0.155831	31.826456	
min	0.160000	0.000000	0.150000	0.190000	0.000000	0.890000	
25%	0.790000	0.340000	0.980000	2.030000	0.080000	37.759998	
50%	1.570000	0.560000	1.680000	4.470000	0.140000	58.520000	
75%	3.010000	0.940000	3.190000	8.610000	0.220000	81.110001	
max	46.369999	13.920000	62.090000	102.599998	2.930000	407.200012	

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

Out[6]:
```
(217296, 16)
```

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

Out[7]:
```
3476736
```

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

Out[8]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	SO_2
0	False	True	False	True	True	True	False	False	True	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	True	False	True	True	True	False	False	True	False	False	True	False
3	False	True	False	True	True	True	False	False	True	False	False	True	False
4	False	True	False	True	True	True	False	False	True	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
217291	False	False	False	True	True	True	False	False	True	False	False	True	False
217292	False	False	False	False	True	False	False	False	True	False	False	True	False
217293	False	False	False	False	False	False	False	False	False	False	False	False	False
217294	False	False	False	False	False	True	False	False	False	False	True	False	False
217295	False	False	False	False	False	False	False	False	False	False	False	False	False

217296 rows × 16 columns

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

Out[9]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
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	20.980000
5	2002-04-01 01:00:00	3.19	0.72	3.23	7.65	0.11	113.699997	187.000000	3.53	12.37	27.450001
22	2002-04-01 01:00:00	2.02	0.80	1.57	3.66	0.15	93.860001	101.300003	1.77	6.99	33.000000
24	2002-04-01 01:00:00	3.02	1.04	2.43	5.38	0.21	103.699997	195.399994	2.15	14.04	37.310001
26	2002-04-01 02:00:00	2.02	0.53	2.24	5.97	0.12	91.599998	136.199997	2.55	6.76	19.980000
...	...	...	...	...	...	...	...	...	...	...	...
217269	2002-10-31 23:00:00	1.24	0.28	1.26	2.64	0.11	60.080002	64.160004	1.23	15.64	13.910000
217271	2002-10-31 23:00:00	3.13	1.30	2.93	7.90	0.28	84.779999	184.000000	2.23	7.94	32.529999
217273	2002-11-01 00:00:00	2.50	0.97	3.63	9.95	0.19	61.759998	132.100006	4.46	5.45	29.500000
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	9.640000
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	29.240000

32381 rows × 16 columns

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

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

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

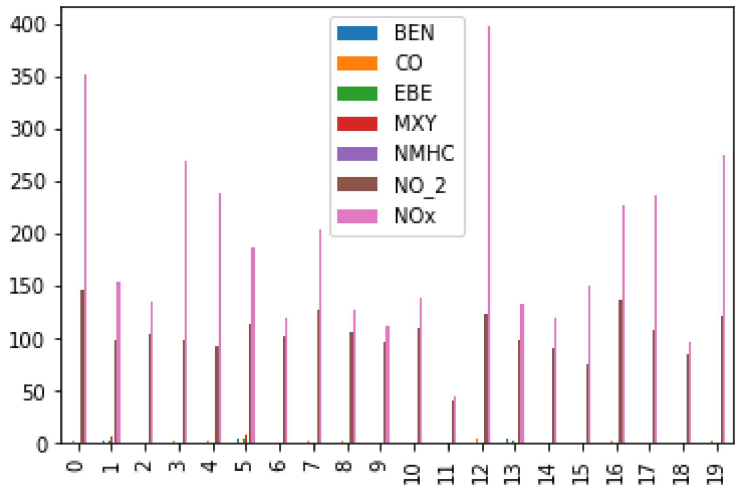
In [12]:
```
dd=sd.head(20)
dd
```

Out[12]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx
0	NaN	1.39	NaN	NaN	NaN	145.100006	352.100006
1	1.93	0.71	2.33	6.20	0.15	98.150002	153.399994
2	NaN	0.80	NaN	NaN	NaN	103.699997	134.000000
3	NaN	1.61	NaN	NaN	NaN	97.599998	268.000000
4	NaN	1.90	NaN	NaN	NaN	92.089996	237.199997
5	3.19	0.72	3.23	7.65	0.11	113.699997	187.000000
6	NaN	0.78	NaN	NaN	0.09	101.000000	119.300003
7	NaN	1.06	NaN	NaN	NaN	127.300003	204.100006
8	NaN	1.21	NaN	NaN	NaN	106.300003	126.599998
9	NaN	0.61	NaN	NaN	0.14	95.540001	110.699997
10	NaN	0.70	NaN	NaN	NaN	110.099998	138.600006
11	NaN	0.31	NaN	NaN	NaN	39.680000	43.619999
12	NaN	3.34	NaN	NaN	NaN	123.699997	397.500000
13	4.20	0.61	2.54	NaN	0.19	97.449997	133.000000
14	NaN	0.58	NaN	NaN	NaN	90.120003	119.500000
15	NaN	0.85	NaN	NaN	NaN	75.900002	149.399994
16	NaN	1.21	NaN	NaN	0.60	136.300003	225.800003
17	NaN	0.84	NaN	NaN	NaN	107.800003	235.500000
18	NaN	0.70	NaN	NaN	NaN	84.300003	95.629997
19	NaN	1.09	NaN	NaN	NaN	120.900002	275.200012

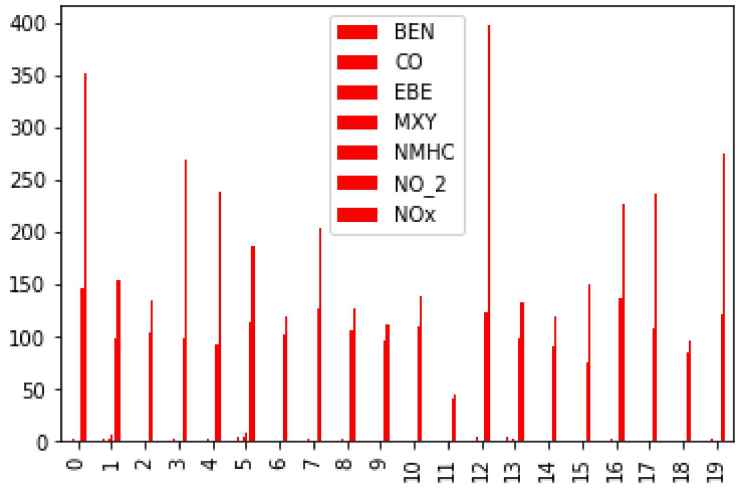
In [13]:
```
dd.plot.bar()
```

Out[13]: <AxesSubplot:>



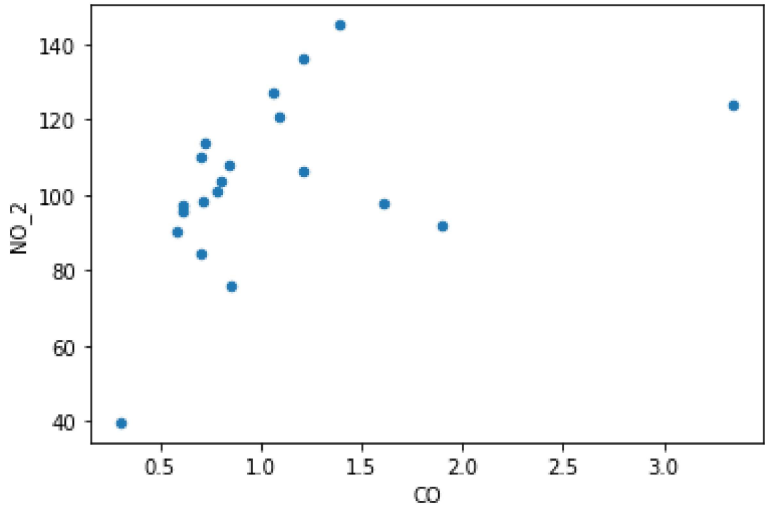
In [14]:
```
dd.plot.bar(color='r')
```

Out[14]: <AxesSubplot:>



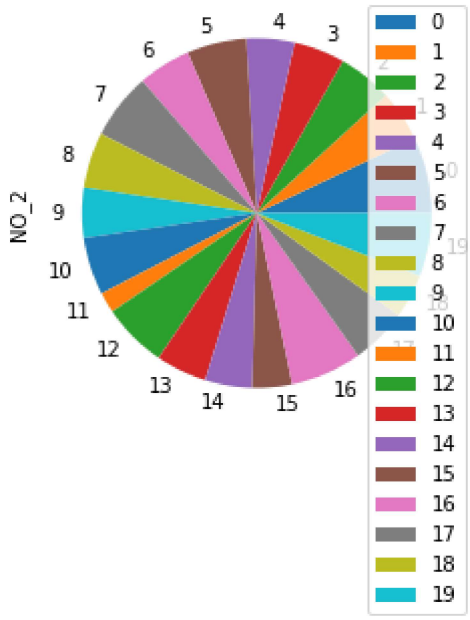
In [15]:
```
dd.plot.scatter(x='CO',y='NO_2')
```

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



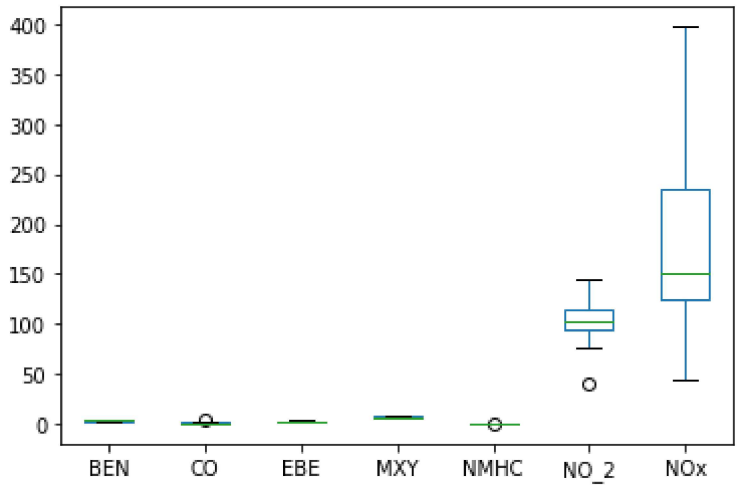
In [16]:
```
dd.plot.pie(y='NO_2')
```

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



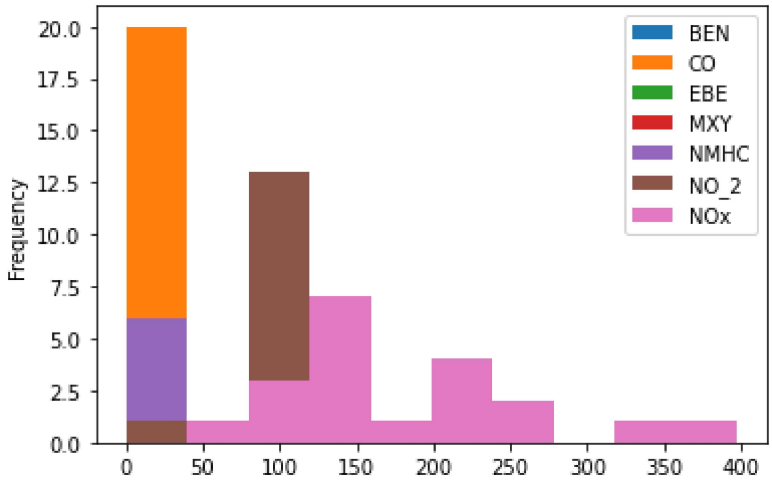
In [17]:
```
dd.plot.box()
```

Out[17]: <AxesSubplot:>



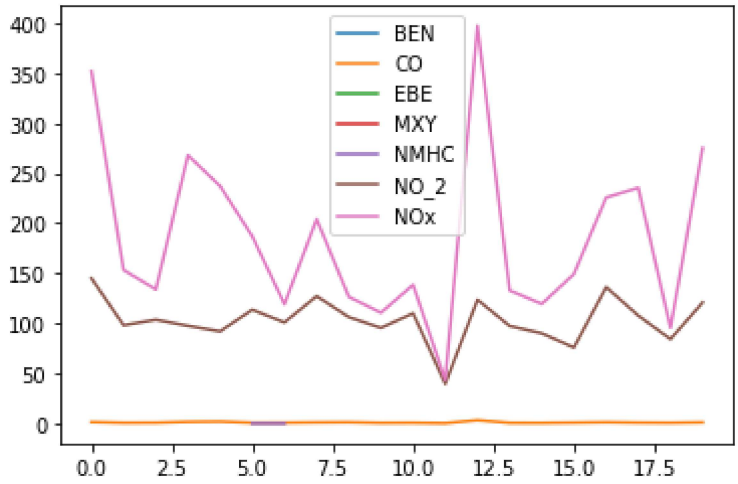
In [18]:
```
dd.plot.hist()
```

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



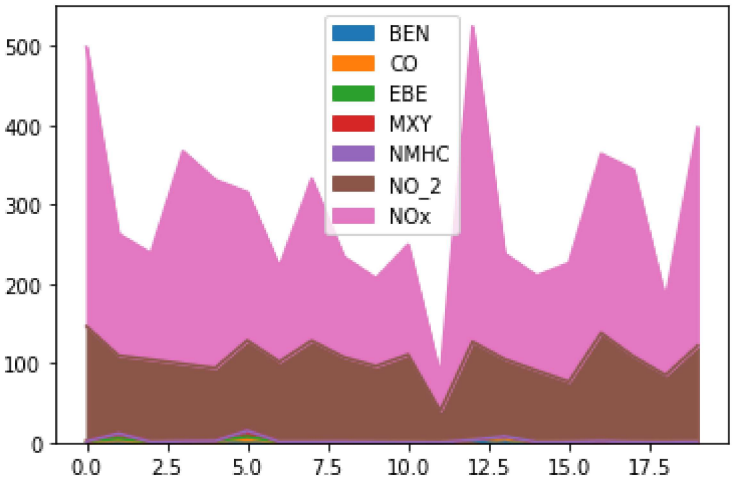
In [19]:
```
dd.plot.line()
```

Out[19]: <AxesSubplot:>



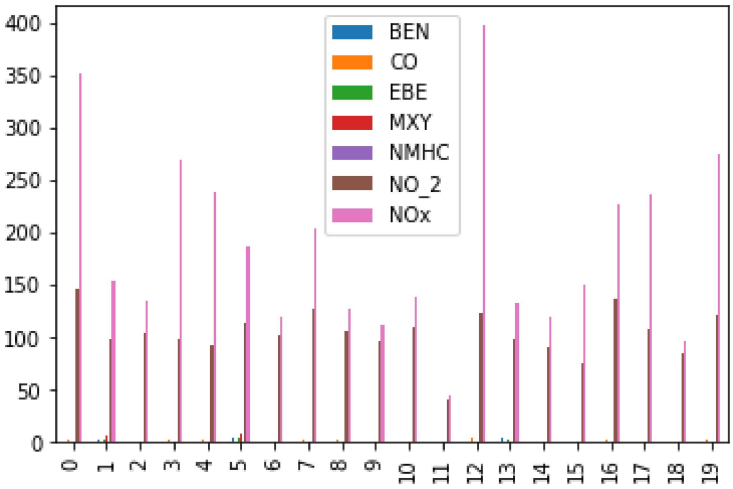
In [20]:
```
dd.plot.area()
```

Out[20]: <AxesSubplot:>



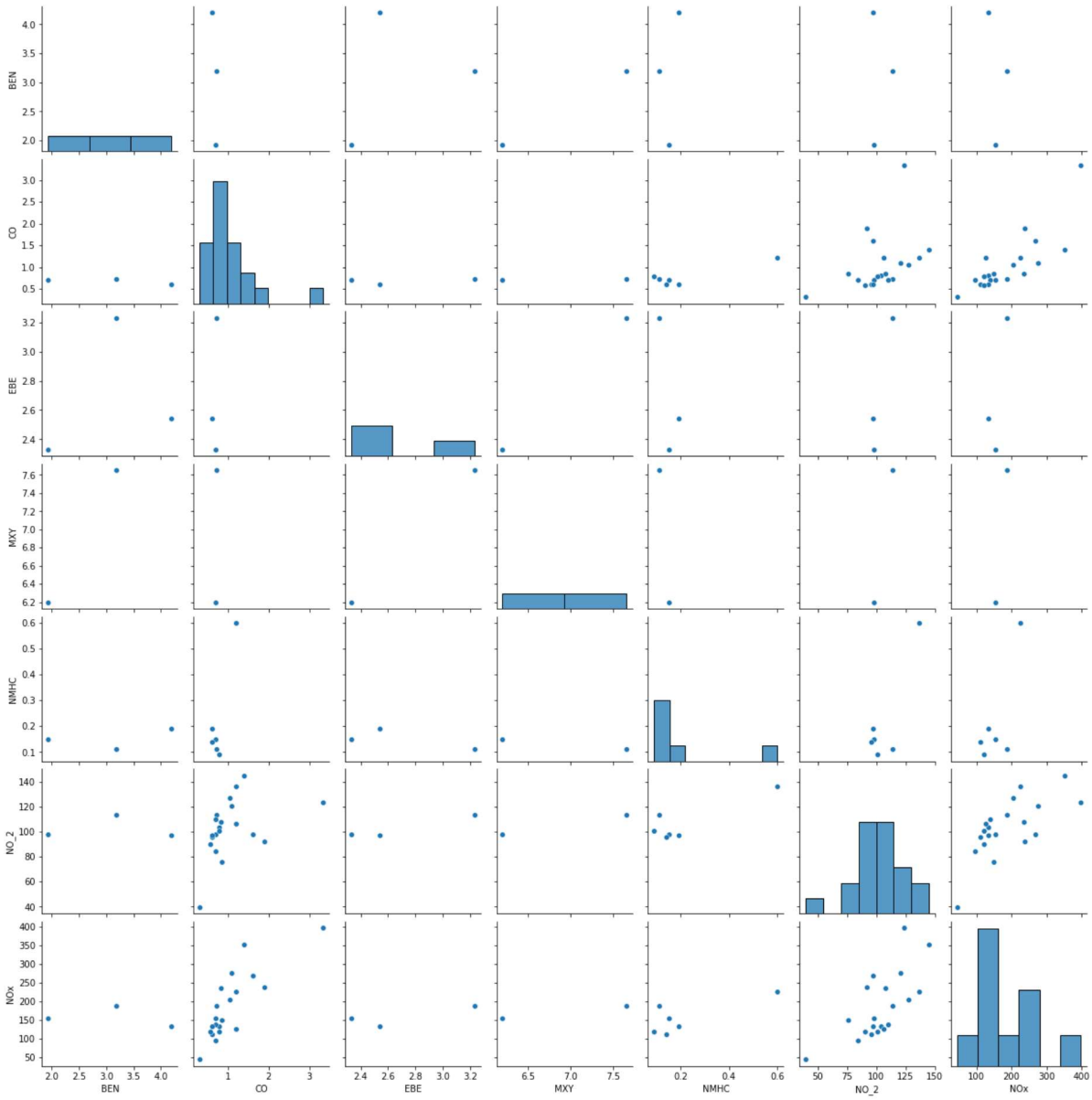
In [21]:
```
dd.plot.bar()
```

Out[21]: <AxesSubplot:>



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

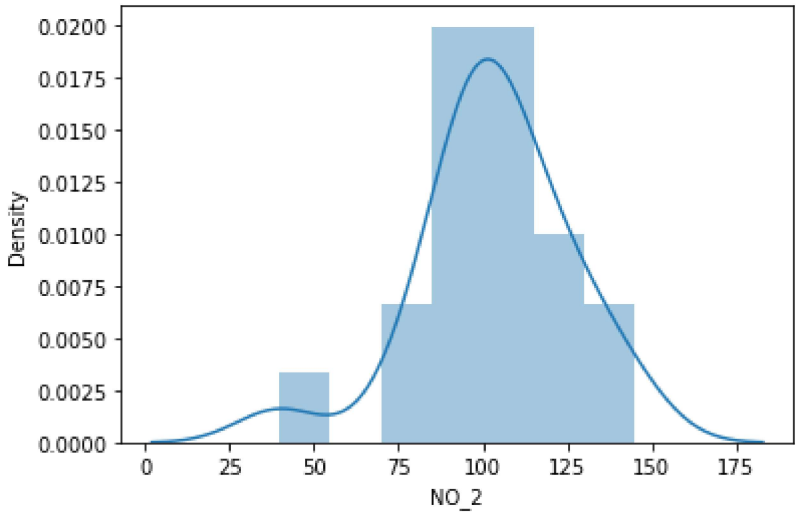
Out[22]:
```
<seaborn.axisgrid.PairGrid at 0x253700abf10>
```



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'>
```



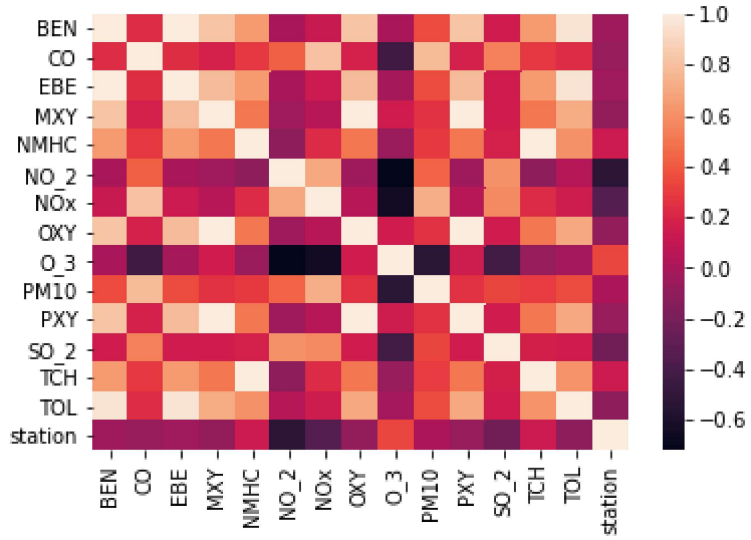
In [24]:
```
ds=data.fillna(20)
```

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:>
```



LinearRegression

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_)
```

```
28079028.136805344
```

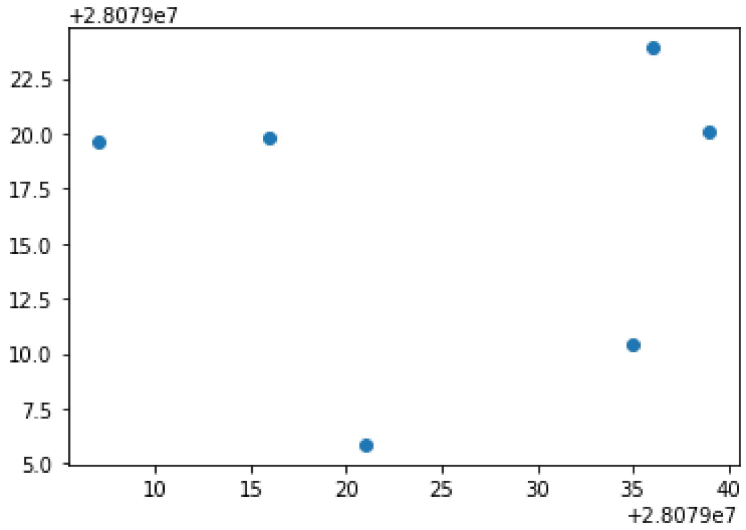
In [32]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[32]:

	Co-efficient
BEN	0.078471
CO	9.082512
EBE	0.052633
MXY	0.233055
NMHC	0.041289
NO_2	-0.111183
NOx	-0.097553

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

Out[33]: <matplotlib.collections.PathCollection at 0x25375c0e100>



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

```
-0.8076573148463491
```

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

Out[35]: -0.8076573148463491

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

Out[36]: 0.4565134952458446

Ridge,Lasso

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]: -0.89523477887986

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

Out[40]: 0.3958793072034211

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]:
```
-0.6132174850276557
```

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

Out[43]:
```
0.34165962692986984
```

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_)
```

```
[ 0.1469722   1.12204717  0.07428168  0.27151072  0.01907388 -0.16398196
 -0.03749015]
```

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

```
28079028.780283615
```

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

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

```
-0.8719386298371454
```

LogisticRegression()

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)
```

```
[28079001]
```

In [59]:
```
logr.classes_
```

Out[59]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079009,
       28079011, 28079012, 28079014, 28079015, 28079016, 28079017,
       28079018, 28079019, 28079021, 28079035, 28079036, 28079038,
       28079039, 28079040], dtype=int64)
```

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

Out[60]:
```
0.9778346330154625
```

RandomForestClassifier()

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)
```

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')
```

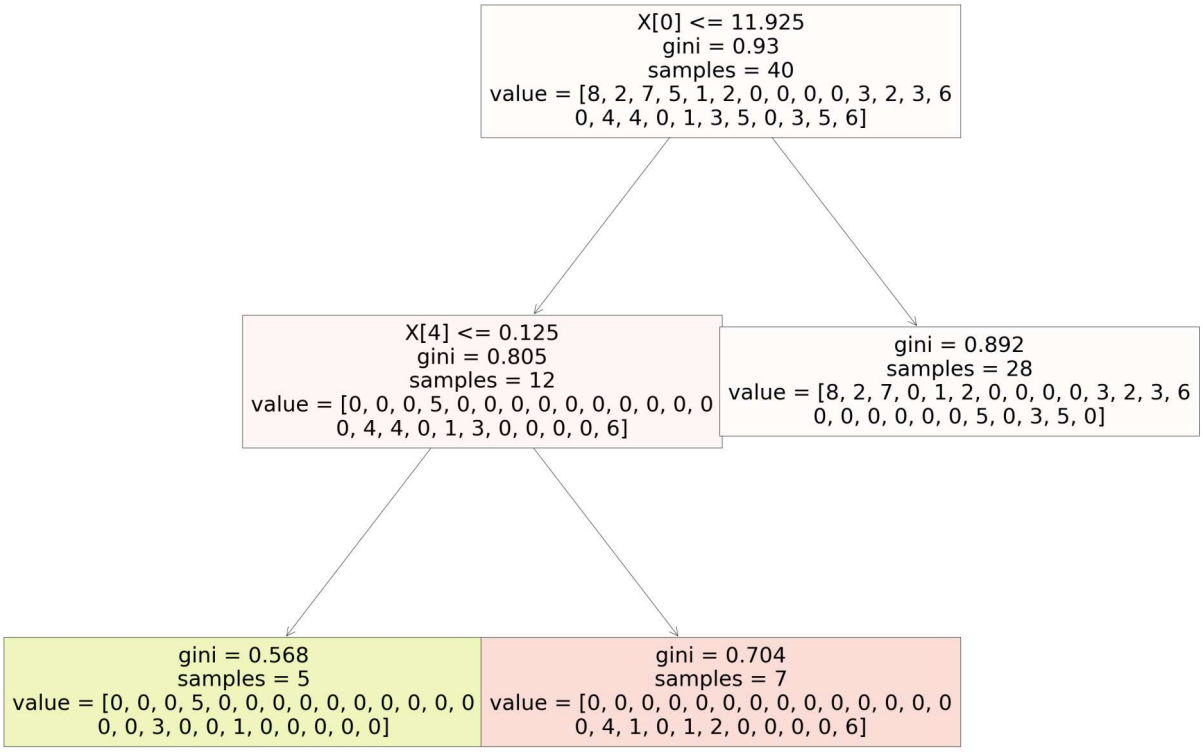
In [70]:
```
grid_search.best_score_
```

Out[70]:
```
0.2142857142857143
```

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(1674.0, 1812.0, 'X[0] <= 11.925\ngini = 0.93\nsamples = 40\nvalue = [8,
2, 7, 5, 1, 2, 0, 0, 0, 0, 3, 2, 3, 6\n0, 4, 4, 0, 1, 3, 5, 0, 3, 5, 6]'),
 Text(1116.0, 1087.2, 'X[4] <= 0.125\ngini = 0.805\nsamples = 12\nvalue = [0,
0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 4, 4, 0, 1, 3, 0, 0, 0, 0, 6]'),
 Text(558.0, 362.39999999999986, 'gini = 0.568\nsamples = 5\nvalue = [0, 0,
0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 3, 0, 0, 1, 0, 0, 0, 0, 0]'),
 Text(1674.0, 362.39999999999986, 'gini = 0.704\nsamples = 7\nvalue = [0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0, 4, 1, 0, 1, 2, 0, 0, 0, 0, 6]'),
 Text(2232.0, 1087.2, 'gini = 0.892\nsamples = 28\nvalue = [8, 2, 7, 0, 1, 2,
0, 0, 0, 0, 3, 2, 3, 6\n0, 0, 0, 0, 0, 0, 5, 0, 3, 5, 0]')]
```



Conclusion : RandomForest()
0.2142857142857143 HIGH RANGE

In [ ]:
```
```