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_2005.csv")
data
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
0	2005-11-01 01:00:00	NaN	0.77	NaN	NaN	NaN	57.130001	128.699997	NaN	14.720000	14.91
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.93
2	2005-11-01 01:00:00	NaN	0.40	NaN	NaN	NaN	46.119999	53.000000	NaN	30.469999	14.60
3	2005-11-01 01:00:00	NaN	0.42	NaN	NaN	NaN	37.220001	52.009998	NaN	21.379999	15.16
4	2005-11-01 01:00:00	NaN	0.57	NaN	NaN	NaN	32.160000	36.680000	NaN	33.410000	5.00
...	...	...	...	...	...	...	...	...	...	...	...
236995	2006-01-01 00:00:00	1.08	0.36	1.01	NaN	0.11	21.990000	23.610001	NaN	43.349998	5.00
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999	4.95
236997	2006-01-01 00:00:00	0.19	NaN	0.26	NaN	0.08	26.730000	30.809999	NaN	43.840000	4.31
236998	2006-01-01 00:00:00	0.14	NaN	1.00	NaN	0.06	13.770000	17.770000	NaN	NaN	5.00
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998	5.67

237000 rows × 17 columns

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

Out[3]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	P
0	2005-11-01 01:00:00	NaN	0.77	NaN	NaN	NaN	57.130001	128.699997	NaN	14.720000	14.910000	1
1	2005-11-01 01:00:00	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997	1.27	11.680000	30.930000	
2	2005-11-01 01:00:00	NaN	0.40	NaN	NaN	NaN	46.119999	53.000000	NaN	30.469999	14.600000	
3	2005-11-01 01:00:00	NaN	0.42	NaN	NaN	NaN	37.220001	52.009998	NaN	21.379999	15.160000	
4	2005-11-01 01:00:00	NaN	0.57	NaN	NaN	NaN	32.160000	36.680000	NaN	33.410000	5.000000	
5	2005-11-01 01:00:00	1.92	0.88	2.44	5.14	0.22	90.309998	207.699997	2.78	13.760000	18.070000	1
6	2005-11-01 01:00:00	NaN	0.55	NaN	NaN	0.27	50.279999	77.209999	NaN	19.120001	18.209999	
7	2005-11-01 01:00:00	0.20	0.38	1.00	NaN	0.27	51.759998	72.989998	NaN	14.810000	16.430000	
8	2005-11-01 01:00:00	NaN	0.70	NaN	NaN	NaN	39.040001	43.860001	NaN	25.379999	16.139999	
9	2005-11-01 01:00:00	NaN	0.56	NaN	NaN	NaN	41.820000	51.869999	NaN	24.290001	7.130000	

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

Out[4]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	P
236980	2006-01-01 00:00:00	NaN	0.32	NaN	NaN	0.21	23.610001	32.470001	NaN	37.020000	4.33	
236981	2006-01-01 00:00:00	0.20	0.30	0.12	NaN	0.13	33.740002	42.869999	NaN	40.849998	3.79	
236982	2006-01-01 00:00:00	NaN	0.44	NaN	NaN	NaN	38.139999	43.439999	NaN	48.009998	5.00	
236983	2006-01-01 00:00:00	NaN	0.56	NaN	NaN	NaN	16.010000	18.160000	NaN	53.700001	5.00	
236984	2006-01-01 00:00:00	NaN	0.62	NaN	NaN	0.27	25.559999	56.130001	NaN	54.959999	4.70	
236985	2006-01-01 00:00:00	NaN	0.52	NaN	NaN	NaN	30.240000	35.919998	NaN	43.029999	4.69	
236986	2006-01-01 00:00:00	NaN	0.33	NaN	NaN	NaN	15.470000	20.559999	NaN	46.200001	5.91	
236987	2006-01-01 00:00:00	NaN	0.30	NaN	NaN	NaN	21.219999	27.860001	NaN	NaN	5.00	
236988	2006-01-01 00:00:00	0.37	0.29	0.43	NaN	0.10	21.040001	28.280001	NaN	43.430000	5.00	
236989	2006-01-01 00:00:00	NaN	0.43	NaN	NaN	NaN	19.170000	20.420000	NaN	49.470001	5.00	
236990	2006-01-01 00:00:00	NaN	0.61	NaN	NaN	NaN	14.920000	16.200001	NaN	52.910000	5.00	
236991	2006-01-01 00:00:00	NaN	0.23	NaN	NaN	NaN	13.470000	14.850000	NaN	55.730000	5.00	
236992	2006-01-01 00:00:00	NaN	0.25	NaN	NaN	NaN	21.570000	23.450001	NaN	50.650002	5.00	
236993	2006-01-01 00:00:00	NaN	0.64	NaN	NaN	NaN	8.640000	10.900000	NaN	53.970001	5.00	
236994	2006-01-01 00:00:00	NaN	0.31	NaN	NaN	NaN	7.290000	7.970000	NaN	51.709999	5.00	
236995	2006-01-01 00:00:00	1.08	0.36	1.01	NaN	0.11	21.990000	23.610001	NaN	43.349998	5.00	
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999	4.95	

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	P
236997	2006-01-01 00:00:00	0.19	NaN	0.26	NaN	0.08	26.730000	30.809999	NaN	43.840000	4.31	
236998	2006-01-01 00:00:00	0.14	NaN	1.00	NaN	0.06	13.770000	17.770000	NaN	NaN	5.00	
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998	5.67	

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

Out[5]:

	BEN	CO	EBE	MXY	NMHC	NO_2	
count	70370.000000	217656.000000	68955.000000	32549.000000	92854.000000	235022.000000	2
mean	1.294665	0.652278	1.633530	4.862915	0.176943	61.829514	
std	1.845280	0.527821	2.524665	5.093012	0.153792	37.404725	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.240000	0.330000	0.550000	1.580000	0.080000	33.619999	
50%	0.720000	0.510000	1.000000	3.300000	0.140000	56.080002	
75%	1.550000	0.790000	1.710000	6.270000	0.220000	82.779999	
max	40.090000	9.350000	84.279999	77.379997	6.860000	419.500000	

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

Out[6]:
```
(237000, 17)
```

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

Out[7]:
```
4029000
```

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	False	True
1	False	False	False	False	False	False	False	False	False	False	False	True	False
2	False	True	False	True	True	True	False	False	True	False	False	True	True
3	False	True	False	True	True	True	False	False	True	False	False	True	True
4	False	True	False	True	True	True	False	False	True	False	False	True	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...
236995	False	False	False	False	True	False	False	False	True	False	False	True	True
236996	False	False	False	False	False	False	False	False	False	False	False	False	False
236997	False	False	True	False	True	False	False	False	True	False	False	False	True
236998	False	False	True	False	True	False	False	False	True	True	False	True	True
236999	False	False	False	False	False	False	False	False	False	False	False	False	False

237000 rows × 17 columns

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

Out[9]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10
5	2005-11-01 01:00:00	1.92	0.88	2.44	5.14	0.22	90.309998	207.699997	2.78	13.760000	18.07
22	2005-11-01 01:00:00	0.30	0.22	0.25	0.59	0.11	18.540001	19.020000	0.67	46.799999	9.88
25	2005-11-01 01:00:00	0.67	0.49	0.94	3.44	0.17	48.740002	74.349998	1.57	23.430000	13.88
31	2005-11-01 02:00:00	3.10	0.84	3.21	6.82	0.22	89.919998	224.199997	3.72	12.390000	28.74
48	2005-11-01 02:00:00	0.39	0.20	0.29	0.68	0.11	16.639999	17.080000	0.40	47.689999	8.78
...	...	...	...	...	...	...	...	...	...	...	...
236970	2005-12-31 23:00:00	0.37	0.39	1.00	1.00	0.10	4.500000	5.550000	1.00	57.779999	8.26
236973	2005-12-31 23:00:00	0.92	0.45	1.26	3.42	0.14	37.250000	49.060001	2.57	31.889999	19.73
236979	2006-01-01 00:00:00	1.00	0.38	1.11	2.35	0.04	35.919998	59.480000	1.39	35.810001	4.22
236996	2006-01-01 00:00:00	0.39	0.54	1.00	1.00	0.11	2.200000	4.220000	1.00	69.639999	4.95
236999	2006-01-01 00:00:00	0.50	0.40	0.73	1.84	0.13	20.940001	26.950001	1.49	48.259998	5.67

20070 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']]
```

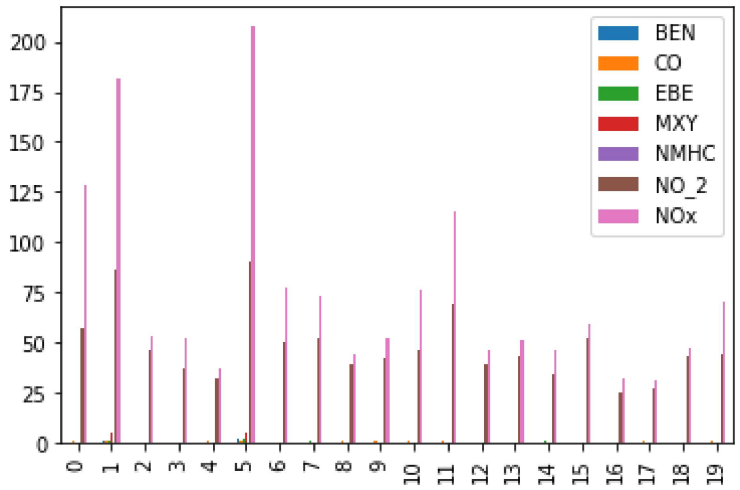
In [12]:
```
dd=sd.head(20)
dd
```

Out[12]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx
0	NaN	0.77	NaN	NaN	NaN	57.130001	128.699997
1	1.52	0.65	1.49	4.57	0.25	86.559998	181.699997
2	NaN	0.40	NaN	NaN	NaN	46.119999	53.000000
3	NaN	0.42	NaN	NaN	NaN	37.220001	52.009998
4	NaN	0.57	NaN	NaN	NaN	32.160000	36.680000
5	1.92	0.88	2.44	5.14	0.22	90.309998	207.699997
6	NaN	0.55	NaN	NaN	0.27	50.279999	77.209999
7	0.20	0.38	1.00	NaN	0.27	51.759998	72.989998
8	NaN	0.70	NaN	NaN	NaN	39.040001	43.860001
9	NaN	0.56	NaN	NaN	NaN	41.820000	51.869999
10	NaN	0.65	NaN	NaN	0.18	46.040001	76.610001
11	NaN	0.68	NaN	NaN	NaN	69.150002	115.199997
12	NaN	0.29	NaN	NaN	NaN	38.689999	45.790001
13	NaN	0.26	NaN	NaN	NaN	43.509998	50.910000
14	0.41	0.33	0.56	NaN	0.09	34.220001	46.529999
15	NaN	0.41	NaN	NaN	NaN	51.950001	59.450001
16	NaN	0.28	NaN	NaN	NaN	25.600000	32.610001
17	NaN	0.56	NaN	NaN	NaN	26.709999	30.969999
18	NaN	0.29	NaN	NaN	NaN	43.310001	47.650002
19	NaN	0.72	NaN	NaN	NaN	44.360001	70.089996

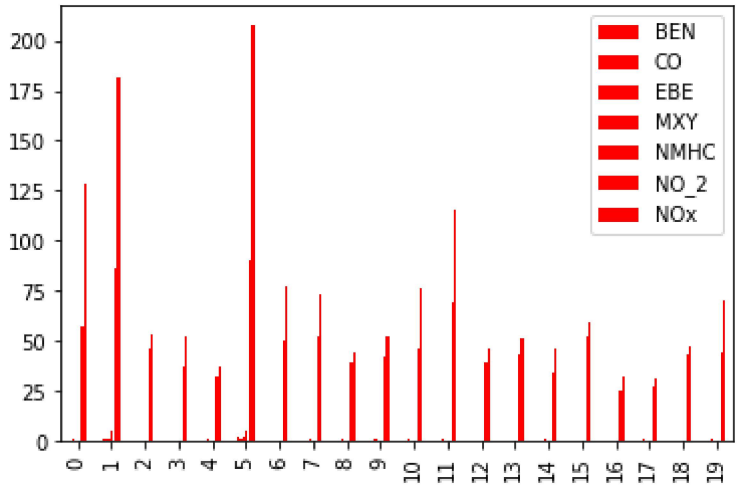
In [13]:
```
dd.plot.bar()
```

Out[13]: <AxesSubplot:>



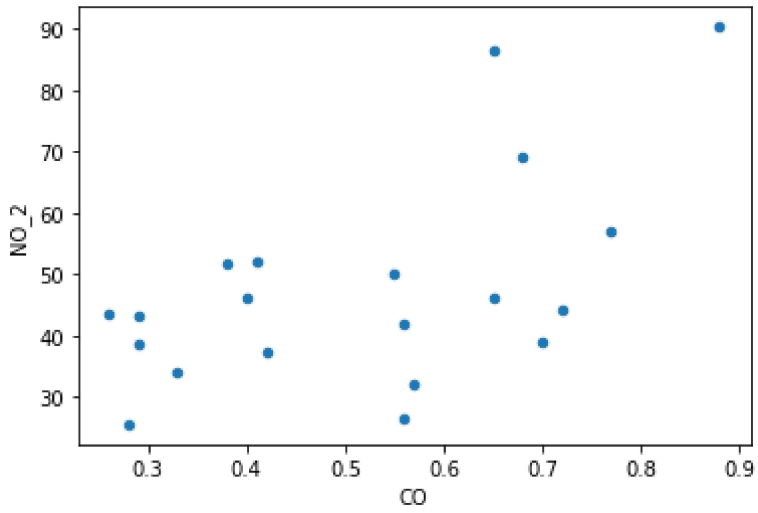
In [14]:
```
dd.plot.bar(color='r')
```

Out[14]: <AxesSubplot:>



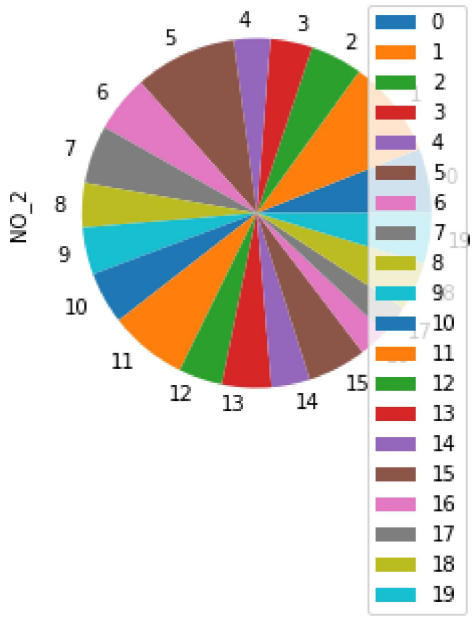
In [15]:
```
dd.plot.scatter(x='CO',y='NO_2')
```

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



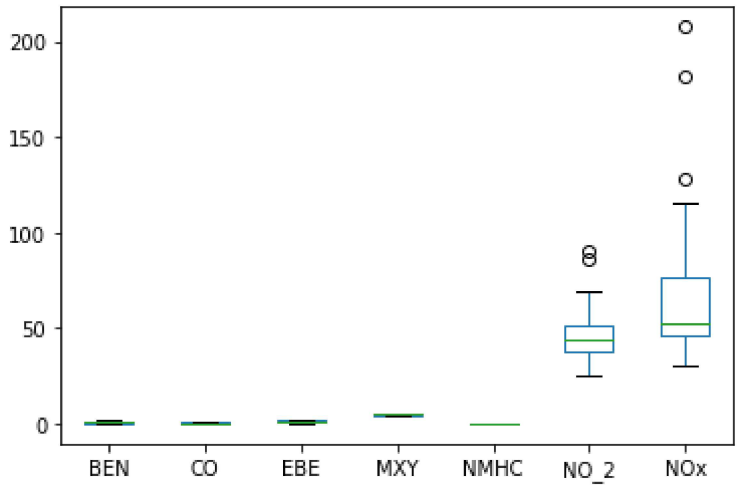
In [16]:
```
dd.plot.pie(y='NO_2')
```

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



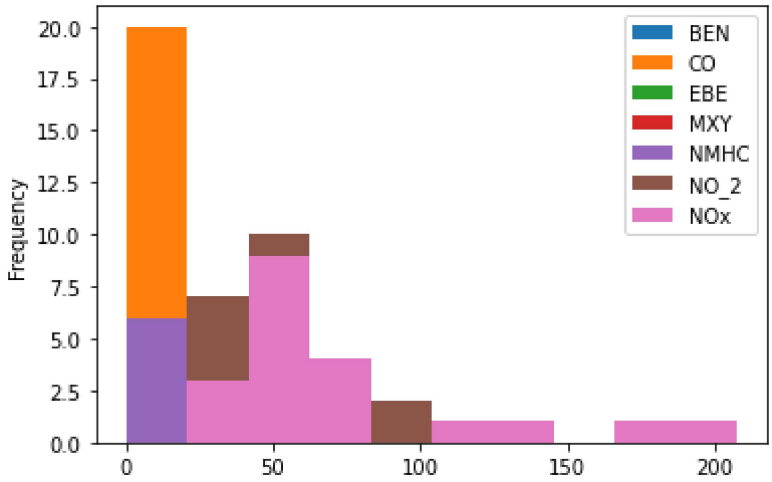
In [17]:
```
dd.plot.box()
```

Out[17]: <AxesSubplot:>



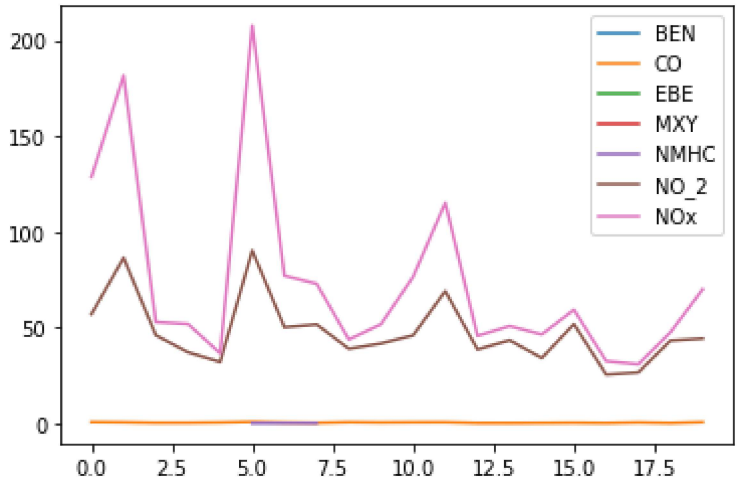
In [18]:
```
dd.plot.hist()
```

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



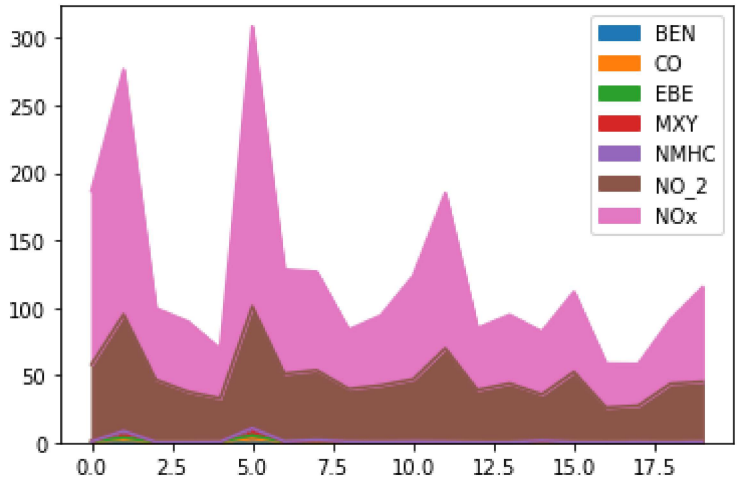
In [19]:
```
dd.plot.line()
```

Out[19]: <AxesSubplot:>



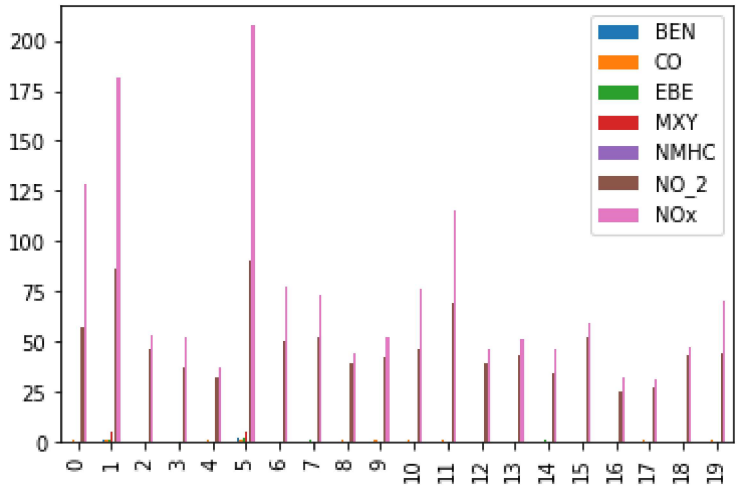
In [20]:
```
dd.plot.area()
```

Out[20]: <AxesSubplot:>



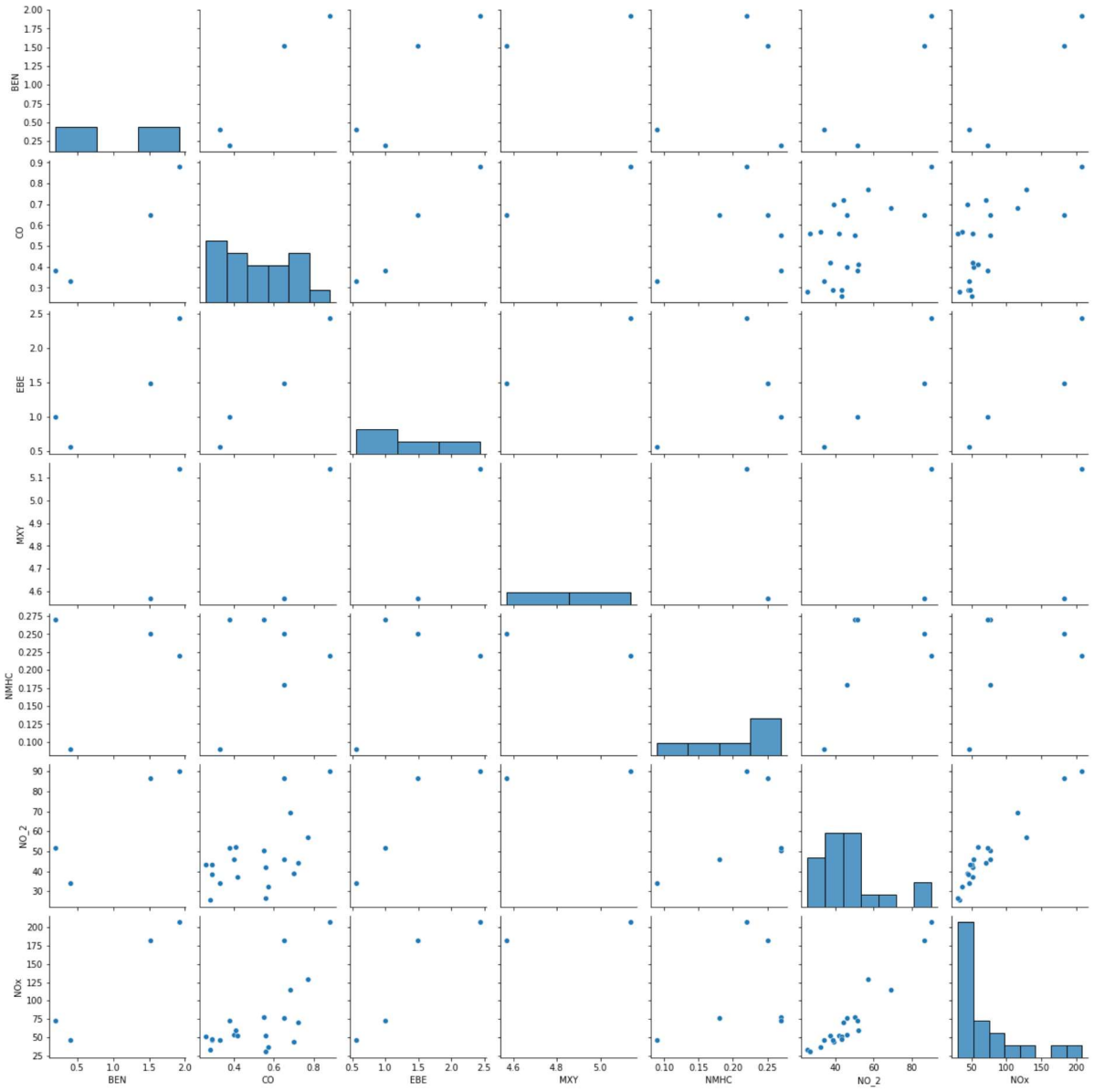
In [21]:
```
dd.plot.bar()
```

Out[21]: <AxesSubplot:>



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

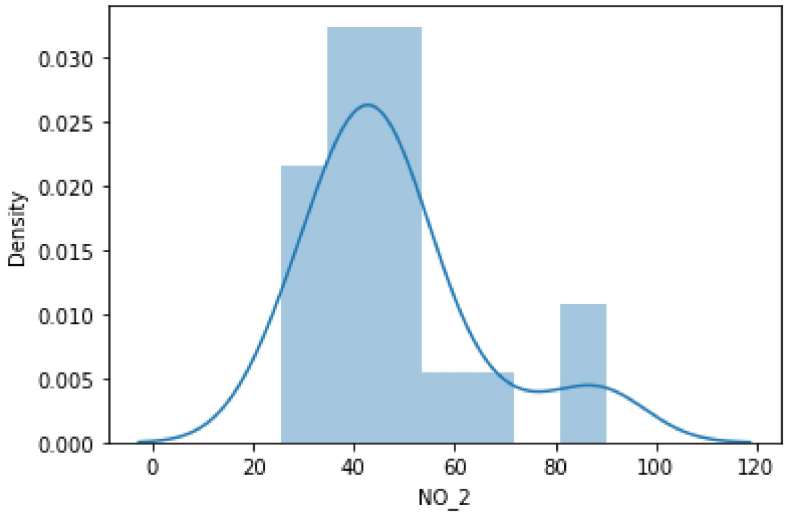
Out[22]:
```
<seaborn.axisgrid.PairGrid at 0x1bef62dc220>
```



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'>
```



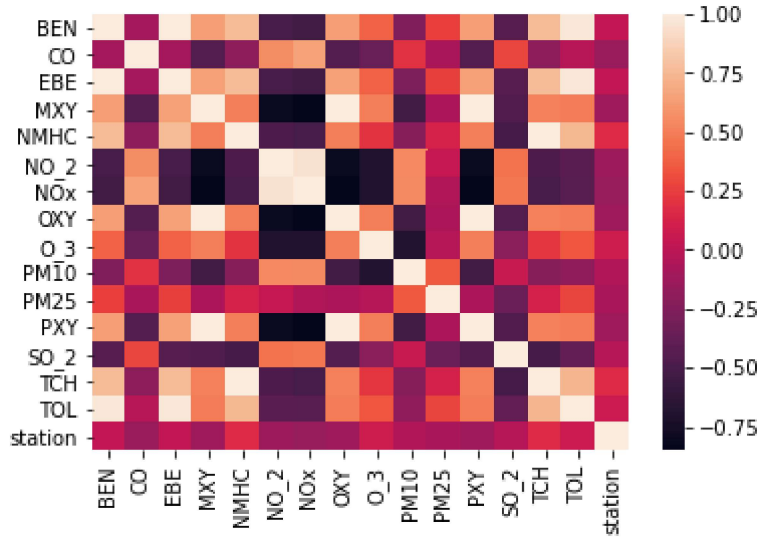
In [24]:
```
ds=data.fillna(20)
```

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_)
```

```
28079067.988365117
```

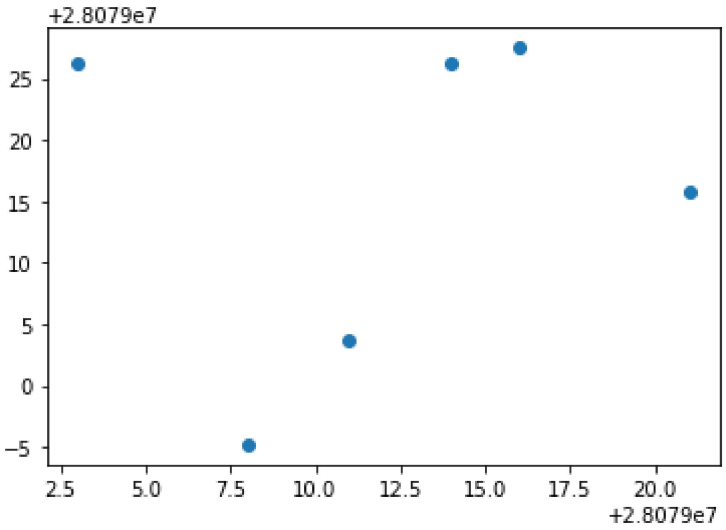
In [32]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[32]:

	Co-efficient
BEN	31.109126
CO	-5.797872
EBE	-31.594687
MXY	-2.499751
NMHC	0.548670
NO_2	0.701443
NOx	-0.434011

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

Out[33]:
```
<matplotlib.collections.PathCollection at 0x1befd73eb80>
```



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

```
-4.372096022320417
```

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

Out[35]:
```
-4.372096022320417
```

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

Out[36]:
```
0.4714946246429693
```

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]:
```
-4.189240646890433
```

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

Out[40]:
```
0.4065328952270928
```

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]:
```
-1.7258813737223608
```

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

Out[43]:
```
0.21841992749087602
```

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.         -0.         -0.32752493 -2.23503092  0.51857853  0.83219835
 -0.50412721]
```

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

```
28079055.84710282
```

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

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

```
-3.975899163340628
```

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([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079017, 28079018, 28079019, 28079021, 28079035, 28079038,
       28079039, 28079040], dtype=int64)
```

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

Out[60]:
```
0.20603948773364483
```

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')
```

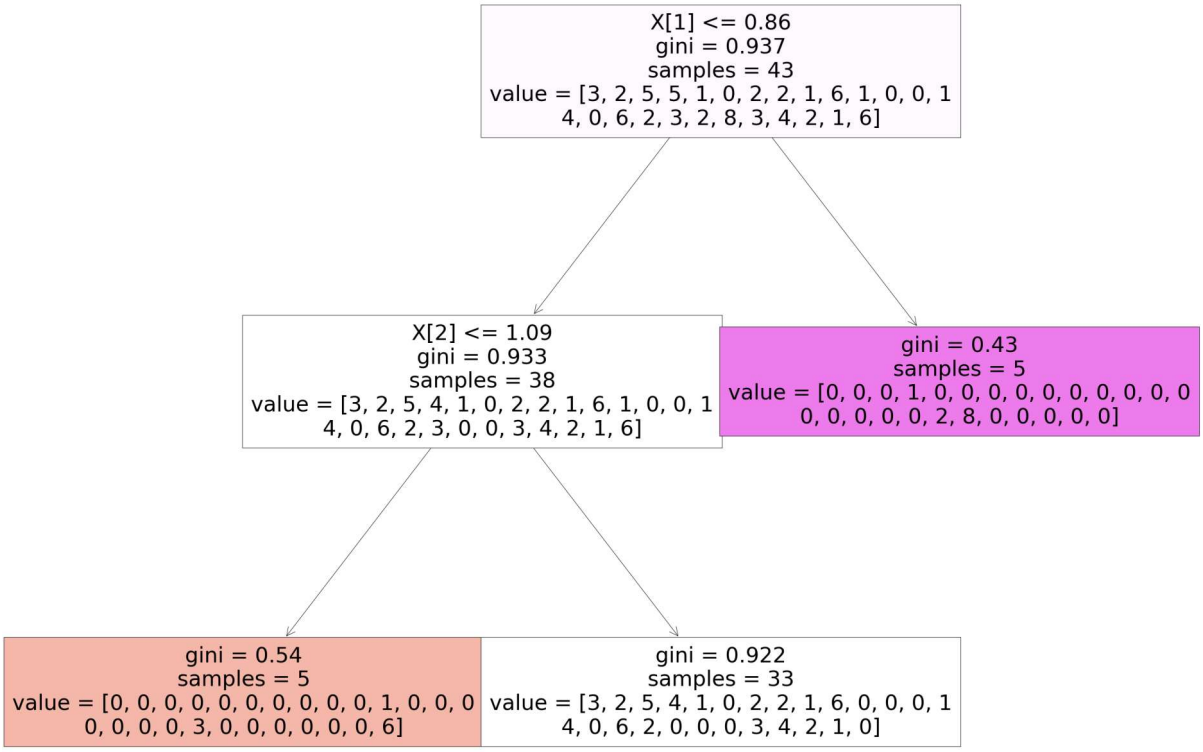
In [70]:
```
grid_search.best_score_
```

Out[70]:
```
0.45714285714285713
```

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[1] <= 0.86\ngini = 0.937\nsamples = 43\nvalue = [3,
2, 5, 5, 1, 0, 2, 2, 1, 6, 1, 0, 0, 1\n4, 0, 6, 2, 3, 2, 8, 3, 4, 2, 1, 6]'),
 Text(1116.0, 1087.2, 'X[2] <= 1.09\ngini = 0.933\nsamples = 38\nvalue = [3,
2, 5, 4, 1, 0, 2, 2, 1, 6, 1, 0, 0, 1\n4, 0, 6, 2, 3, 0, 0, 3, 4, 2, 1, 6]'),
 Text(558.0, 362.39999999999986, 'gini = 0.54\nsamples = 5\nvalue = [0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0\n0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 6]'),
 Text(1674.0, 362.39999999999986, 'gini = 0.922\nsamples = 33\nvalue = [3, 2,
5, 4, 1, 0, 2, 2, 1, 6, 0, 0, 0, 1\n4, 0, 6, 2, 0, 0, 0, 3, 4, 2, 1, 0]'),
 Text(2232.0, 1087.2, 'gini = 0.43\nsamples = 5\nvalue = [0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0\n0, 0, 0, 0, 0, 2, 8, 0, 0, 0, 0, 0]')]
```



Conclusion : LogisticRegression() [28079012] HIGH RANGE

In [ ]: