Vijay(P9) 3/08/2023

In [67]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [133]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_200
df
```

Out[133]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY
0	2009-10-01 01:00:00	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002	NaN	50.680000	18.260000	NaN	NaN
1	2009-10-01 01:00:00	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000	NaN	55.880001	10.580000	NaN	NaN
2	2009-10-01 01:00:00	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001	NaN	49.060001	25.190001	NaN	NaN
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	6.82	1.30
4	2009-10-01 01:00:00	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002	NaN	38.090000	23.760000	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
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	10.830000	7.15	0.74
215684	2009-06-01 00:00:00	NaN	0.31	NaN	NaN	NaN	76.110001	101.099998	NaN	41.220001	9.920000	NaN	NaN
215685	2009-06-01 00:00:00	0.13	NaN	0.86	NaN	0.23	81.050003	99.849998	NaN	24.830000	12.460000	6.77	NaN
215686	2009-06-01 00:00:00	0.21	NaN	2.96	NaN	0.10	72.419998	82.959999	NaN	NaN	13.030000	NaN	NaN
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	15.360000	11.61	0.83

215688 rows × 17 columns

In [134]:
```
df=df.dropna()
```

In [135]:
```
df.columns

```

Out[135]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [136]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24717 entries, 3 to 215687
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     24717 non-null  object 
 1   BEN      24717 non-null  float64
 2   CO       24717 non-null  float64
 3   EBE      24717 non-null  float64
 4   MXY      24717 non-null  float64
 5   NMHC     24717 non-null  float64
 6   NO_2     24717 non-null  float64
 7   NOx      24717 non-null  float64
 8   OXY      24717 non-null  float64
 9   O_3      24717 non-null  float64
 10  PM10     24717 non-null  float64
 11  PM25     24717 non-null  float64
 12  PXY      24717 non-null  float64
 13  SO_2     24717 non-null  float64
 14  TCH      24717 non-null  float64
 15  TOL      24717 non-null  float64
 16  station  24717 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.4+ MB
```

In [137]:
```
1 data=df[['BEN', 'CO','station']]
2 data
3 
```

Out[137]:

	BEN	CO	station
3	0.95	0.33	28079006
20	0.38	0.32	28079024
24	0.55	0.24	28079099
28	0.65	0.21	28079006
45	0.38	0.30	28079024
...	...	...	...
215659	0.54	0.27	28079024
215663	0.74	0.35	28079099
215667	0.78	0.29	28079006
215683	0.50	0.22	28079024
215687	0.37	0.32	28079099

24717 rows × 3 columns

In [138]:
```
df=df.head(15000)
df
```

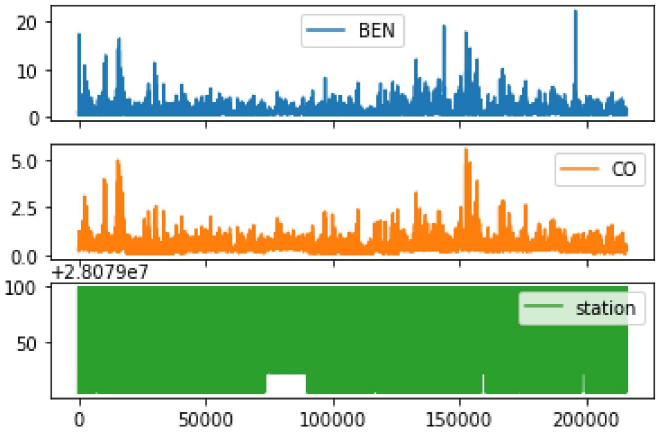
Out[138]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY
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	6.82	1.30
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	10.520000	7.01	0.84
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	19.150000	9.33	1.07
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	32.200001	6.94	1.04
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	12.260000	8.46	0.88
...	...	...	...	...	...	...	...	...	...	...	...	...	...
133346	2009-02-12 23:00:00	2.17	0.69	1.91	3.75	0.24	93.320000	190.199997	2.14	7.150000	20.379999	11.89	1.77
133362	2009-02-12 23:00:00	0.56	0.33	0.45	1.17	0.17	41.470001	44.639999	1.00	34.520000	8.750000	5.13	0.61
133366	2009-02-12 23:00:00	0.72	0.55	0.96	2.46	0.19	80.019997	134.199997	1.57	13.970000	17.650000	9.23	1.19
133371	2009-02-13 00:00:00	2.00	1.04	1.70	3.32	0.31	125.800003	302.399994	1.88	2.180000	32.790001	20.27	1.56
133387	2009-02-13 00:00:00	0.64	0.55	0.45	1.16	0.18	67.540001	81.190002	1.00	9.880000	12.260000	7.16	0.60

15000 rows × 17 columns

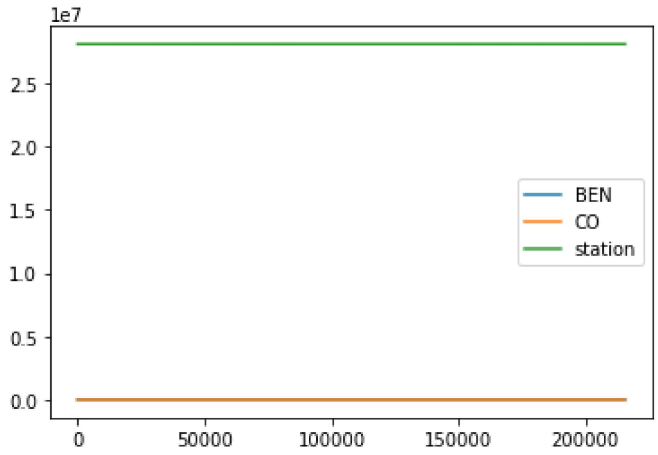
In [139]:
```
data.plot.line(subplots=True)
```

Out[139]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [140]:
```
1 data.plot.line()
2
```

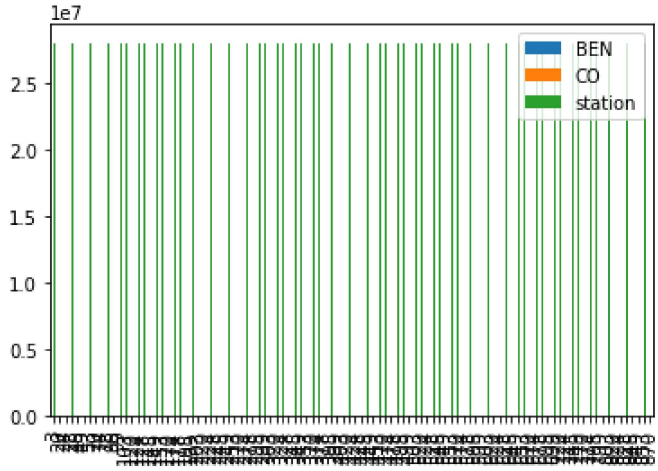
Out[140]: <AxesSubplot:>



In [143]:
```
1 b=data[0:100]
2
```

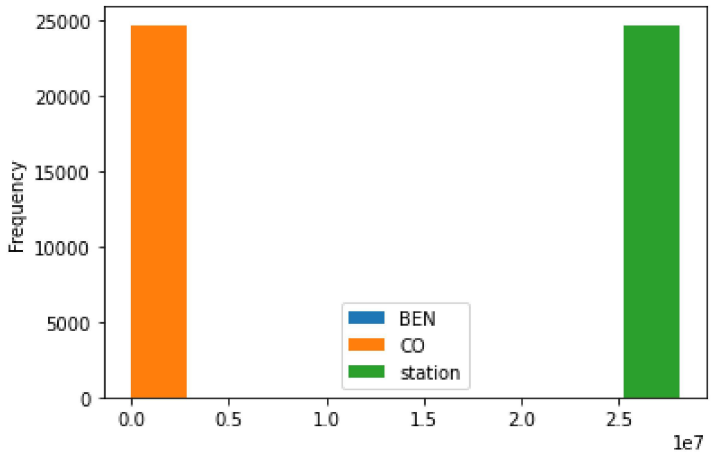
In [144]:
```
1 b.plot.bar()
```

Out[144]: <AxesSubplot:>



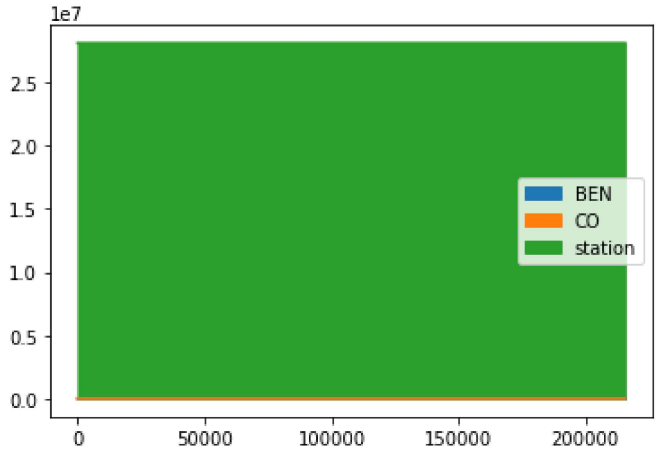
In [145]:
```
1 data.plot.hist()
2
```

Out[145]: <AxesSubplot:ylabel='Frequency'>



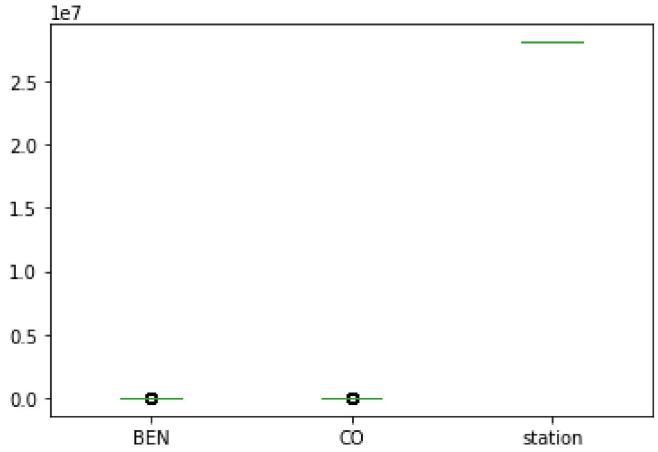
In [146]:
```
1 data.plot.area()
```

Out[146]: <AxesSubplot:>



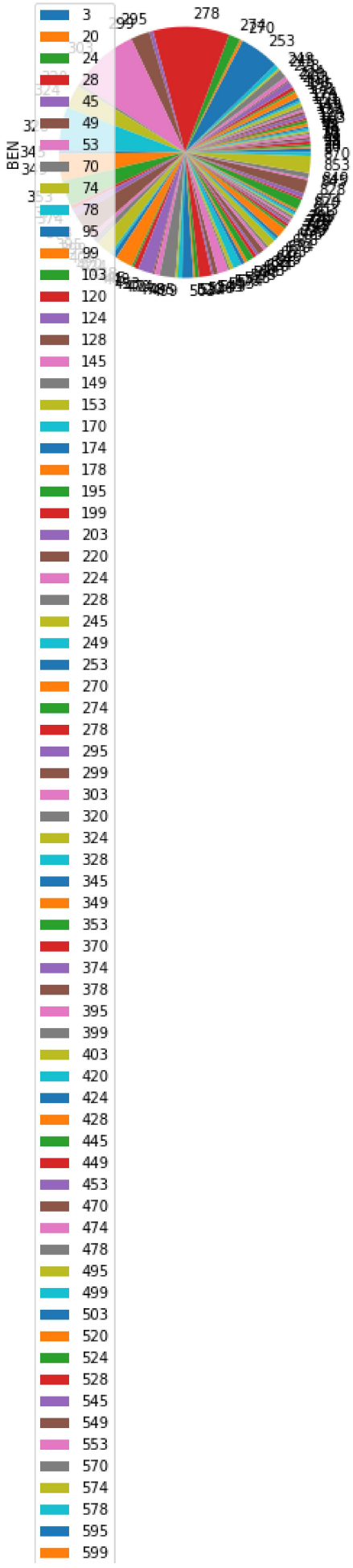
In [147]:
```
1 data.plot.box()
```

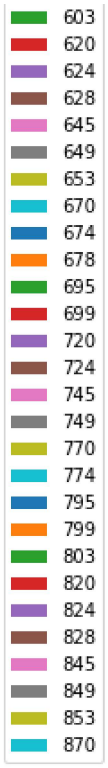
Out[147]: <AxesSubplot:>



In [148]:
```
1 b.plot.pie(y='BEN' )
```

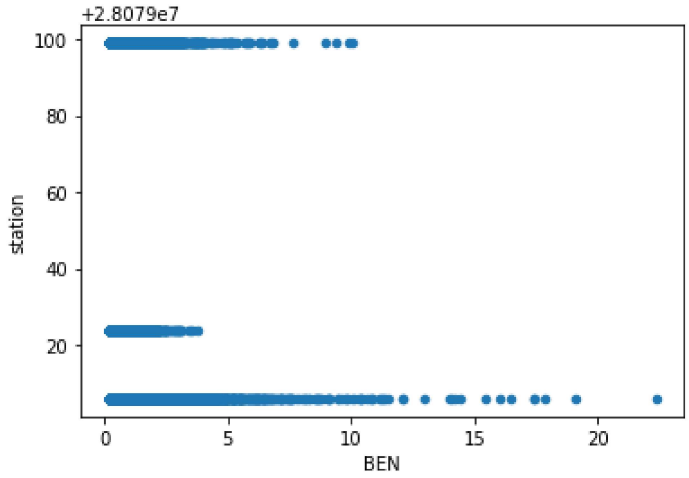
Out[148]:
```
<AxesSubplot:ylabel='BEN'>
```


BEN
278
253
3
20
24
28
45
49
53
70
74
78
95
99
103
120
124
128
145
149
153
170
174
178
195
199
203
220
224
228
245
249
253
270
274
278
295
299
303
320
324
328
345
349
353
370
374
378
395
399
403
420
424
428
445
449
453
470
474
478
495
499
503
520
524
528
545
549
553
570
574
578
595
599



In [149]:
```
1 data.plot.scatter(x='BEN' ,y='station')
2
```

Out[149]: <AxesSubplot:xlabel='BEN', ylabel='station'>



In [150]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15000 entries, 3 to 133387
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     15000 non-null  object 
 1   BEN      15000 non-null  float64
 2   CO       15000 non-null  float64
 3   EBE      15000 non-null  float64
 4   MXY      15000 non-null  float64
 5   NMHC     15000 non-null  float64
 6   NO_2     15000 non-null  float64
 7   NOx      15000 non-null  float64
 8   OXY      15000 non-null  float64
 9   O_3      15000 non-null  float64
 10  PM10     15000 non-null  float64
 11  PM25     15000 non-null  float64
 12  PXY      15000 non-null  float64
 13  SO_2     15000 non-null  float64
 14  TCH      15000 non-null  float64
 15  TOL      15000 non-null  float64
 16  station  15000 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 2.1+ MB
```

In [151]:
```
1 df.describe()
2
```

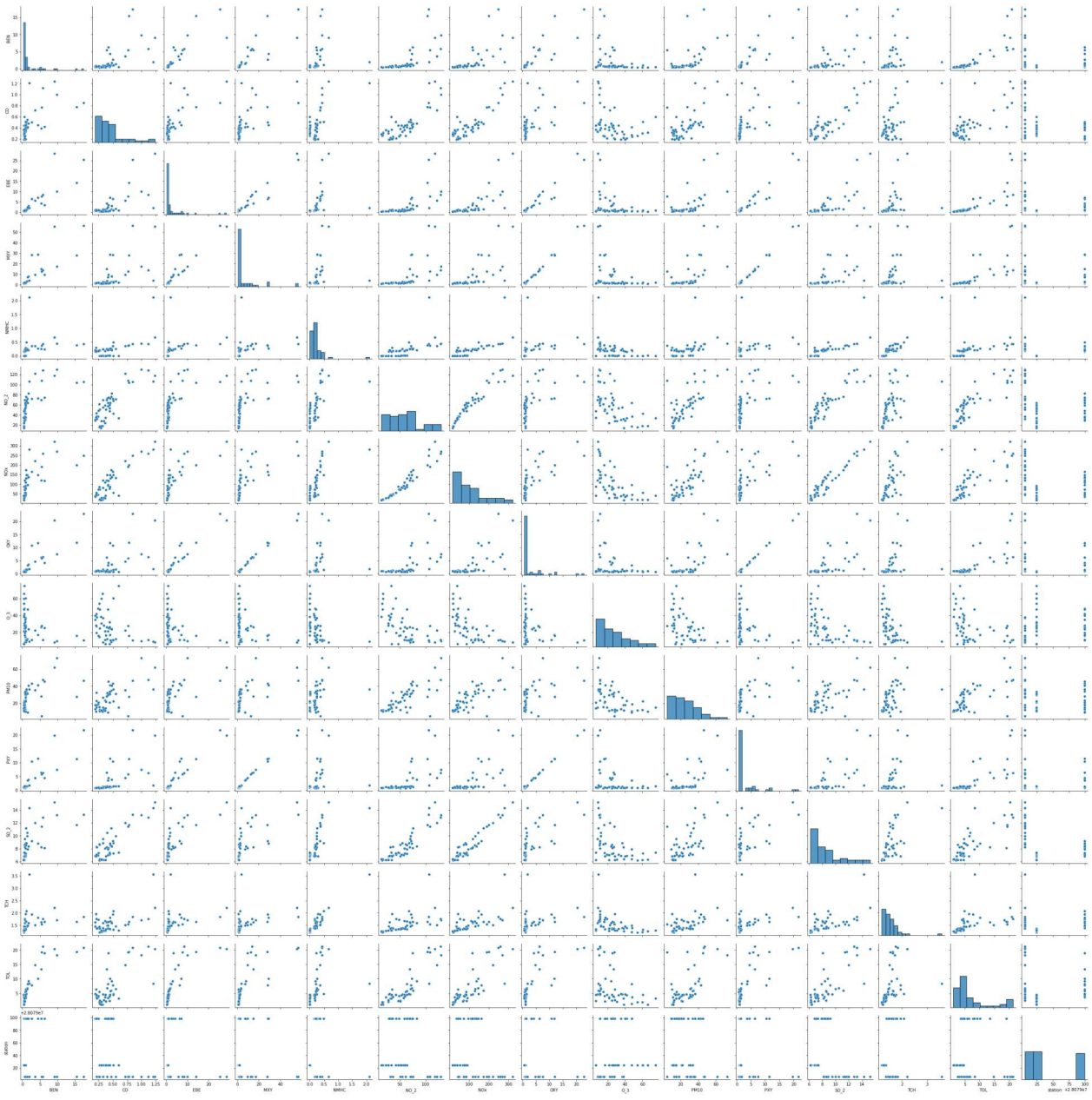
Out[151]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	
count	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000	15000
mean	0.930545	0.424697	1.209391	2.115665	0.221291	54.027933	86.398438	1
std	0.937072	0.284678	0.942650	2.151884	0.152719	39.598022	87.947826	0
min	0.170000	0.060000	0.250000	0.240000	0.000000	0.860000	2.640000	0
25%	0.440000	0.250000	0.720000	0.940000	0.140000	24.700001	30.510000	0
50%	0.610000	0.350000	1.000000	1.350000	0.190000	45.184999	62.094999	1
75%	1.060000	0.560000	1.370000	2.680000	0.250000	74.222502	115.300003	1
max	17.400000	4.980000	28.410000	56.500000	2.110000	477.399994	1438.000000	22

In [152]:
```
1 df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
2  'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [153]:
```
1 sns.pairplot(df1[0:50])
```

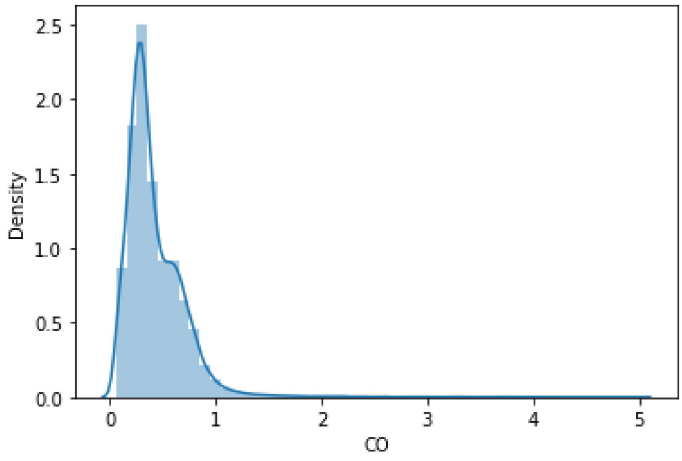
Out[153]: <seaborn.axisgrid.PairGrid at 0x27d329b75b0>



In [154]:
```
1 sns.distplot(df1['CO'])
2
```

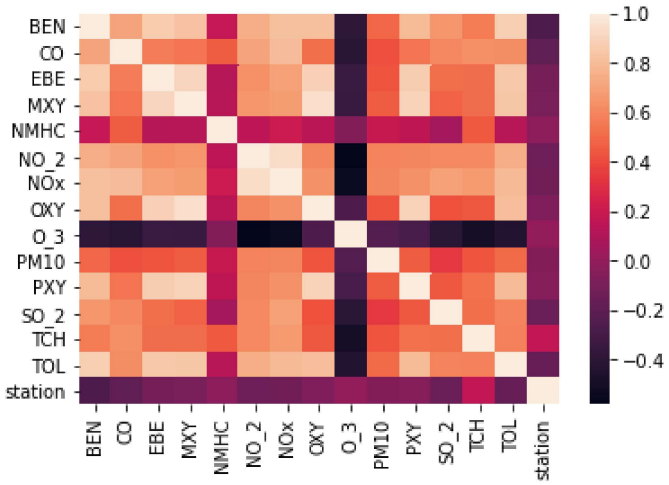
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[154]: <AxesSubplot:xlabel='CO', ylabel='Density'>



In [155]:
```
1 sns.heatmap(df1.corr())
```

Out[155]: <AxesSubplot:>



In [156]:
```
1 x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
2  'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
3 y=df['station']
4
```

In [157]:
```
1 from sklearn.model_selection import train_test_split
2 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
3
```

In [158]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)

```

Out[158]:
```
LinearRegression()
```

In [159]:
```
lr.intercept_
```

Out[159]:
```
28078835.04228549
```

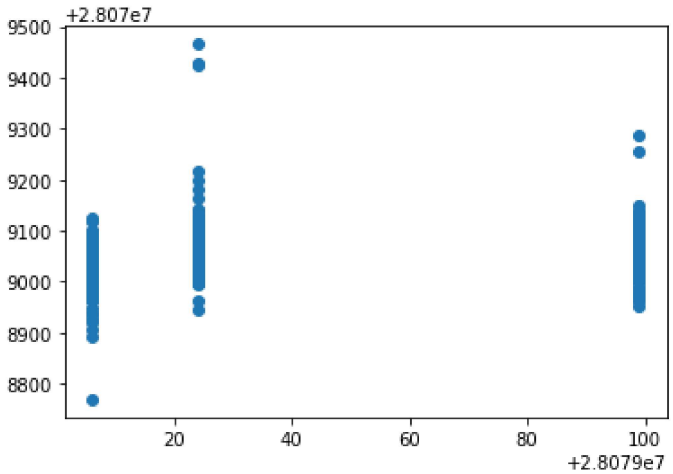
In [160]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[160]:

	Co-efficient
BEN	-32.629401
CO	-26.427462
EBE	4.549725
MXY	-4.091074
NMHC	-46.142858
NO_2	-0.317408
NOx	0.201844
OXY	19.935779
O_3	0.005785
PM10	-0.000663
PXY	2.293015
SO_2	-0.759300
TCH	176.605802
TOL	-1.164715

In [161]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[161]:
```
<matplotlib.collections.PathCollection at 0x27d44a40c40>
```



In [162]:
```
lr.score(x_test,y_test)

```

Out[162]: 0.2714591646166624

In [163]:
```
lr.score(x_train,y_train)

```

Out[163]: 0.32874346114212316

In [164]:
```
from sklearn.linear_model import Ridge,Lasso

```

In [165]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)

```

Out[165]: Ridge(alpha=10)

In [166]:
```
rr.score(x_test,y_test)

```

Out[166]: 0.28198376549370974

In [167]:
```
rr.score(x_train,y_train)

```

Out[167]: 0.3265312248018226

In [168]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[168]: Lasso(alpha=10)

In [169]:
```
la.score(x_train,y_train)

```

Out[169]: 0.03594355902967106

In [170]:
```
la.score(x_test,y_test)

```

Out[170]: 0.02864461246144845

In [171]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)

```

Out[171]: ElasticNet()

In [172]:
```
en.coef_

```

Out[172]:
```
array([-6.21414103, -0.88172107,  0.        ,  1.58857296, -0.        ,
       -0.32394856,  0.13688405,  1.00189763, -0.16854009,  0.12559179,
        2.40741098, -0.96157604,  1.75509927, -1.66205067])
```

In [173]:
```
en.intercept_

```

Out[173]: 28079068.769639283

In [174]:
```
prediction=en.predict(x_test)

```

In [175]:
```
en.score(x_test,y_test)
```

Out[175]: 0.09100787558661116

In [176]:
```
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)))

```

```
36.393475561408536
1502.6854920865546
38.76448751223928
```

In [177]:
```
from sklearn.linear_model import LogisticRegression

```

In [178]:
```
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 [179]:
```
feature_matrix.shape

```

Out[179]: (15000, 14)

In [180]:
```
target_vector.shape

```

Out[180]: (15000,)

In [181]:
```
from sklearn.preprocessing import StandardScaler
```

In [182]:
```
fs=StandardScaler().fit_transform(feature_matrix)

```

In [183]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[183]: LogisticRegression(max_iter=10000)

In [184]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]

```

In [185]:
```
prediction=logr.predict(observation)
print(prediction)

```

```
[28079099]
```

In [186]:
```
logr.classes_

```

Out[186]:
```
array([28079006, 28079024, 28079099], dtype=int64)
```

In [187]:
```
logr.score(fs,target_vector)

```

Out[187]:
```
0.9266
```

In [188]:
```
logr.predict_proba(observation)[0][0]

```

Out[188]:
```
1.741833318862e-13
```

In [189]:
```
logr.predict_proba(observation)

```

Out[189]:
```
array([[1.74183332e-13, 2.94598015e-42, 1.00000000e+00]])
```

In [190]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [191]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[191]:
```
RandomForestClassifier()
```

In [192]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [193]:
```
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[193]:
```
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 [194]:
```
grid_search.best_score_
```

Out[194]:
```
0.9172380952380952
```

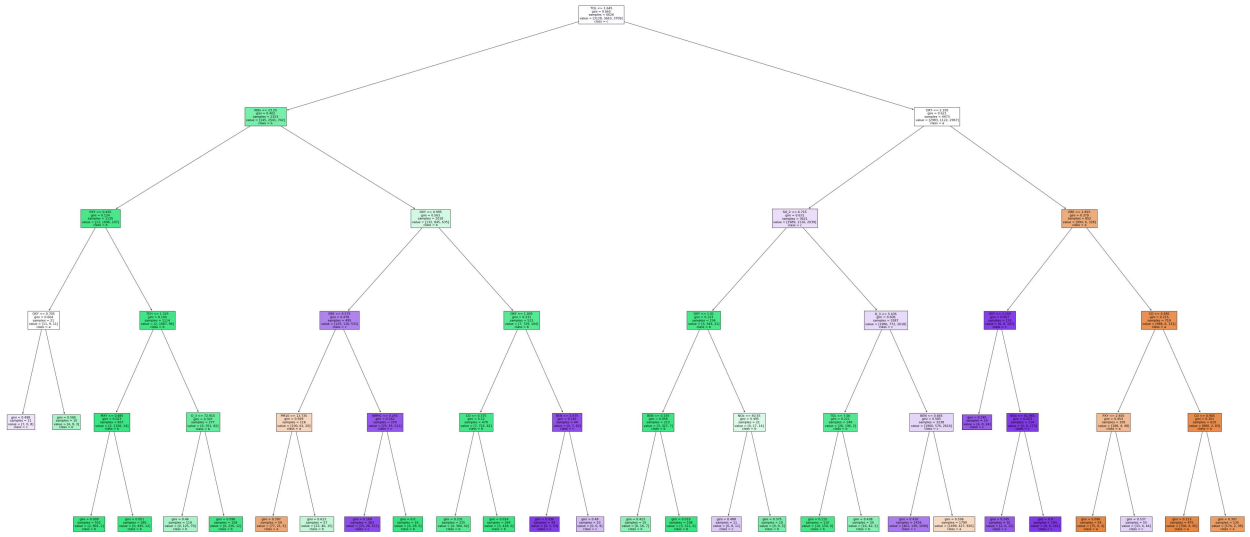
In [195]:
```
rfc_best=grid_search.best_estimator_
```

In [196]:

```
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']
```

Out[196]:
```
[Text(2150.8363636363633, 1993.2, 'TOL <= 1.645\ngini = 0.665\nsamples = 6626\nvalue = [3128, 3663, 3709]\nclass = c'),
 Text(953.6727272727272, 1630.8000000000002, 'NOx <= 23.25\ngini = 0.402\nsamples = 2153\nvalue = [145, 2541, 742]\nclass = b'),
 Text(365.2363636363636, 1268.4, 'PXY <= 0.425\ngini = 0.124\nsamples = 1135\nvalue = [13, 1696, 107]\nclass = b'),
 Text(162.3272727272727, 906.0, 'OXY <= 0.705\ngini = 0.664\nsamples = 21\nvalue = [11, 9, 11]\nclass = a'),
 Text(81.16363636363636, 543.5999999999999, 'gini = 0.498\nsamples = 11\nvalue = [7, 0, 8]\nclass = c'),
 Text(243.49090909090907, 543.5999999999999, 'gini = 0.586\nsamples = 10\nvalue = [4, 9, 3]\nclass = b'),
 Text(568.1454545454545, 906.0, 'TCH <= 1.325\ngini = 0.104\nsamples = 1114\nvalue = [2, 1687, 96]\nclass = b'),
 Text(405.81818181818176, 543.5999999999999, 'MXY <= 0.885\ngini = 0.023\nsamples = 837\nvalue = [2, 1336, 14]\nclass = b'),
 Text(324.6545454545454, 181.19999999999982, 'gini = 0.009\nsamples = 552\nvalue = [2, 891, 2]\nclass = b'),
 Text(486.98181818181814, 181.19999999999982, 'gini = 0.051\nsamples = 285\nvalue = [0, 445, 12]\nclass = b'),
 Text(730.4727272727272, 543.5999999999999, 'O_3 <= 72.915\ngini = 0.307\nsamples = 277\nvalue = [0, 351, 82]\nclass = b'),
 Text(649.3090909090909, 181.19999999999982, 'gini = 0.46\nsamples = 119\nvalue = [0, 125, 70]\nclass = b'),
 Text(811.6363636363635, 181.19999999999982, 'gini = 0.096\nsamples = 158\nvalue = [0, 226, 12]\nclass = b'),
 Text(1542.1090909090908, 1268.4, 'OXY <= 0.995\ngini = 0.563\nsamples = 1018\nvalue = [132, 845, 635]\nclass = b'),
 Text(1217.4545454545453, 906.0, 'EBE <= 0.575\ngini = 0.478\nsamples = 495\nvalue = [125, 116, 531]\nclass = c'),
 Text(1055.1272727272726, 543.5999999999999, 'PM10 <= 13.735\ngini = 0.569\nsamples = 116\nvalue = [100, 61, 20]\nclass = a'),
 Text(973.9636363636363, 181.19999999999982, 'gini = 0.397\nsamples = 59\nvalue = [77, 21, 5]\nclass = a'),
 Text(1136.290909090909, 181.19999999999982, 'gini = 0.613\nsamples = 57\nvalue = [23, 40, 15]\nclass = b'),
 Text(1379.7818181818182, 543.5999999999999, 'NMHC <= 0.245\ngini = 0.242\nsamples = 379\nvalue = [25, 55, 511]\nclass = c'),
 Text(1298.6181818181817, 181.19999999999982, 'gini = 0.169\nsamples = 363\nvalue = [25, 26, 511]\nclass = c'),
 Text(1460.9454545454544, 181.19999999999982, 'gini = 0.0\nsamples = 16\nvalue = [0, 29, 0]\nclass = b'),
 Text(1866.7636363636361, 906.0, 'OXY <= 1.005\ngini = 0.231\nsamples = 523\nvalue = [7, 729, 104]\nclass = b'),
 Text(1704.4363636363635, 543.5999999999999, 'CO <= 0.375\ngini = 0.12\nsamples = 479\nvalue = [7, 722, 42]\nclass = b'),
 Text(1623.272727272727, 181.19999999999982, 'gini = 0.231\nsamples = 215\nvalue = [4, 304, 42]\nclass = b'),
 Text(1785.6, 181.19999999999982, 'gini = 0.014\nsamples = 264\nvalue = [3, 418, 0]\nclass = b'),
 Text(2029.090909090909, 543.5999999999999, 'BEN <= 0.475\ngini = 0.182\nsamples = 44\nvalue = [0, 7, 62]\nclass = c'),
 Text(1947.9272727272726, 181.19999999999982, 'gini = 0.036\nsamples = 34\nvalue = [0, 1, 53]\nclass = c'),
 Text(2110.254545454545, 181.19999999999982, 'gini = 0.48\nsamples = 10\nvalue = [0, 6, 9]\nclass = c'),
 Text(3347.9999999999995, 1630.8000000000002, 'OXY <= 2.105\ngini = 0.621\nsamples = 4473\nvalue = [2983, 1122, 2967]\nclass = a'),
 Text(2840.7272727272725, 1268.4, 'SO_2 <= 6.715\ngini = 0.631\nsamples = 3621\nvalue = [1989, 1116, 2639]\nclass = c'),
 Text(2516.072727272727, 906.0, 'OXY <= 1.01\ngini = 0.123\nsamples = 234\nvalue = [3, 344, 21]\nclass = b'),
 Text(2353.7454545454543, 543.5999999999999, 'BEN <= 0.335\ngini = 0.058\nsamples = 213\nv
```

```
alue = [3, 327, 7]\nclass = b'),
 Text(2272.581818181818, 181.19999999999982, 'gini = 0.423\nsamples = 15\nvalue = [0, 16,
7]\nclass = b'),
 Text(2434.9090909090905, 181.19999999999982, 'gini = 0.019\nsamples = 198\nvalue = [3, 31
1, 0]\nclass = b'),
 Text(2678.3999999999996, 543.5999999999999, 'NOx <= 40.55\ngini = 0.495\nsamples = 21\nva
lue = [0, 17, 14]\nclass = b'),
 Text(2597.2363636363634, 181.19999999999982, 'gini = 0.488\nsamples = 11\nvalue = [0, 8,
11]\nclass = c'),
 Text(2759.5636363636363, 181.19999999999982, 'gini = 0.375\nsamples = 10\nvalue = [0, 9,
3]\nclass = b'),
 Text(3165.381818181818, 906.0, 'O_3 <= 5.435\ngini = 0.606\nsamples = 3387\nvalue = [198
6, 772, 2618]\nclass = c'),
 Text(3003.0545454545454, 543.5999999999999, 'TOL <= 7.06\ngini = 0.221\nsamples = 149\nva
lue = [26, 196, 2]\nclass = b'),
 Text(2921.8909090909087, 181.19999999999982, 'gini = 0.115\nsamples = 110\nvalue = [10, 1
54, 0]\nclass = b'),
 Text(3084.2181818181816, 181.19999999999982, 'gini = 0.438\nsamples = 39\nvalue = [16, 4
2, 2]\nclass = b'),
 Text(3327.7090909090907, 543.5999999999999, 'BEN <= 0.665\ngini = 0.585\nsamples = 3238\n
value = [1960, 576, 2616]\nclass = c'),
 Text(3246.545454545454, 181.19999999999982, 'gini = 0.416\nsamples = 1439\nvalue = [461,
149, 1690]\nclass = c'),
 Text(3408.872727272727, 181.19999999999982, 'gini = 0.596\nsamples = 1799\nvalue = [1499,
427, 926]\nclass = a'),
 Text(3855.272727272727, 1268.4, 'EBE <= 1.815\ngini = 0.379\nsamples = 852\nvalue = [994,
6, 328]\nclass = a'),
 Text(3571.2, 906.0, 'OXY <= 2.155\ngini = 0.057\nsamples = 133\nvalue = [6, 0, 197]\nclas
s = c'),
 Text(3490.036363636363, 543.5999999999999, 'gini = 0.245\nsamples = 19\nvalue = [4, 0, 2
4]\nclass = c'),
 Text(3652.363636363636, 543.5999999999999, 'NOx <= 61.965\ngini = 0.023\nsamples = 114\nv
alue = [2, 0, 173]\nclass = c'),
 Text(3571.2, 181.19999999999982, 'gini = 0.245\nsamples = 10\nvalue = [2, 0, 12]\nclass =
c'),
 Text(3733.5272727272722, 181.19999999999982, 'gini = 0.0\nsamples = 104\nvalue = [0, 0, 1
61]\nclass = c'),
 Text(4139.345454545454, 906.0, 'CO <= 0.485\ngini = 0.215\nsamples = 719\nvalue = [988,
6, 131]\nclass = a'),
 Text(3977.0181818181813, 543.5999999999999, 'PXY <= 2.405\ngini = 0.454\nsamples = 109\nv
alue = [108, 4, 48]\nclass = a'),
 Text(3895.854545454545, 181.19999999999982, 'gini = 0.096\nsamples = 54\nvalue = [75, 0,
4]\nclass = a'),
 Text(4058.181818181818, 181.19999999999982, 'gini = 0.537\nsamples = 55\nvalue = [33, 4,
44]\nclass = c'),
 Text(4301.672727272727, 543.5999999999999, 'CO <= 0.905\ngini = 0.161\nsamples = 610\nval
ue = [880, 2, 83]\nclass = a'),
 Text(4220.50909090909, 181.19999999999982, 'gini = 0.113\nsamples = 475\nvalue = [706, 0,
45]\nclass = a'),
 Text(4382.836363636363, 181.19999999999982, 'gini = 0.307\nsamples = 135\nvalue = [174,
2, 38]\nclass = a')]
```



Conclusion

Linear Regression=0.32874346114212316

Ridge Regression=0.3265312248018226

Lasso Regression=0.03594355902967106

ElasticNet Regression=0.09100787558661116

Logistic Regression=0.9266

Random Forest=0.9172380952380952

Logistic Regression is Suitable for this Dataset