Vijay(P10) 3/08/2023

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

In [68]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_201(
df
```

Out[68]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25
0	2010-03-01 01:00:00	NaN	0.29	NaN	NaN	NaN	25.090000	29.219999	NaN	68.930000	NaN	NaN
1	2010-03-01 01:00:00	NaN	0.27	NaN	NaN	NaN	24.879999	30.040001	NaN	NaN	NaN	NaN
2	2010-03-01 01:00:00	NaN	0.28	NaN	NaN	NaN	17.410000	20.540001	NaN	72.120003	NaN	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	7.870000
4	2010-03-01 01:00:00	0.79	NaN	1.32	NaN	NaN	21.430000	26.070000	NaN	NaN	24.670000	22.030001
...	...	...	...	...	...	...	...	...	...	...	...	...
209443	2010-08-01 00:00:00	NaN	0.55	NaN	NaN	NaN	125.000000	219.899994	NaN	25.379999	NaN	NaN
209444	2010-08-01 00:00:00	NaN	0.27	NaN	NaN	NaN	45.709999	47.410000	NaN	NaN	51.259998	NaN
209445	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	0.24	46.560001	49.040001	NaN	46.250000	NaN	NaN
209446	2010-08-01 00:00:00	NaN	NaN	NaN	NaN	NaN	46.770000	50.119999	NaN	77.709999	NaN	NaN
209447	2010-08-01 00:00:00	0.92	0.43	0.71	NaN	0.25	76.330002	88.190002	NaN	52.259998	47.150002	26.860001

209448 rows × 17 columns

In [69]:
```
df=df.dropna()
```

In [70]:
```
df.columns

```

Out[70]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [71]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6666 entries, 11 to 191927
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     6666 non-null   object 
 1   BEN      6666 non-null   float64
 2   CO       6666 non-null   float64
 3   EBE      6666 non-null   float64
 4   MXY      6666 non-null   float64
 5   NMHC     6666 non-null   float64
 6   NO_2     6666 non-null   float64
 7   NOx      6666 non-null   float64
 8   OXY      6666 non-null   float64
 9   O_3      6666 non-null   float64
 10  PM10     6666 non-null   float64
 11  PM25     6666 non-null   float64
 12  PXY      6666 non-null   float64
 13  SO_2     6666 non-null   float64
 14  TCH      6666 non-null   float64
 15  TOL      6666 non-null   float64
 16  station  6666 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 937.4+ KB
```

In [72]:
```
1 data=df[['BEN', 'CO','station']]
2 data
3
```

Out[72]:

	BEN	CO	station
11	0.78	0.18	28079024
23	0.70	0.23	28079099
35	0.58	0.17	28079024
47	0.33	0.21	28079099
59	0.38	0.16	28079024
...	...	...	...
191879	0.60	0.26	28079099
191891	0.41	0.16	28079024
191903	0.57	0.28	28079099
191915	0.34	0.16	28079024
191927	0.43	0.25	28079099

6666 rows × 3 columns

In [75]:
```
df=df.head(5000)
df
```

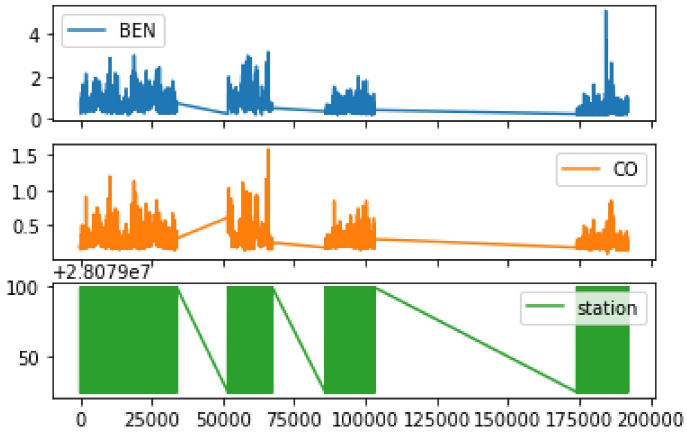
Out[75]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY
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	18.370001	11.300000	0.8
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	23.639999	13.150000	0.8
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	5.600000	5.250000	0.8
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	7.890000	5.540000	0.8
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	5.130000	4.900000	0.7
...	...	...	...	...	...	...	...	...	...	...	...	...	..
99647	2010-04-24 22:00:00	0.46	0.36	0.54	1.00	0.17	49.610001	62.250000	1.0	51.049999	23.629999	13.350000	0.8
99659	2010-04-24 23:00:00	0.31	0.22	0.51	1.00	0.11	39.750000	41.450001	1.0	48.740002	17.400000	9.090000	0.2
99671	2010-04-24 23:00:00	0.60	0.38	0.63	1.00	0.16	59.849998	73.209999	1.0	35.570000	27.610001	17.389999	0.2
99683	2010-04-25 00:00:00	0.36	0.23	0.63	1.00	0.15	55.000000	56.919998	1.0	29.950001	19.490000	9.430000	0.3
99695	2010-04-25 00:00:00	0.65	0.40	0.48	1.00	0.17	60.049999	71.610001	1.0	28.570000	27.120001	15.740000	0.3

5000 rows × 17 columns

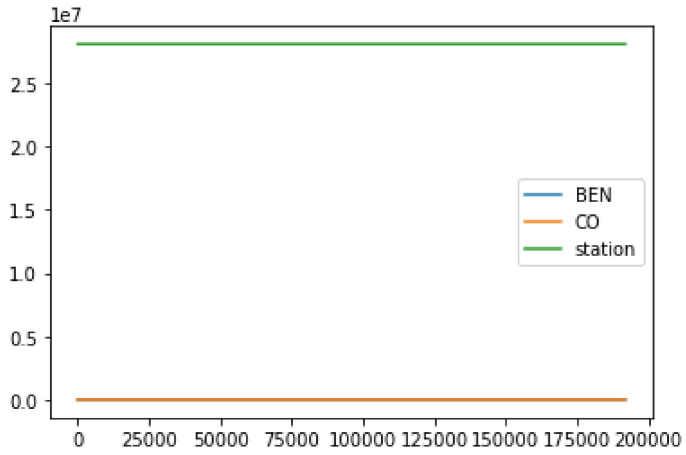
In [76]:
```
data.plot.line(subplots=True)
```

Out[76]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [77]:
```
1 data.plot.line()
2
```

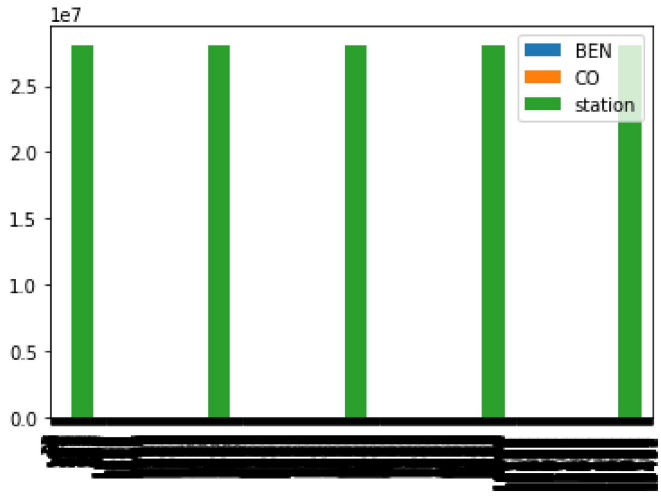
Out[77]: <AxesSubplot:>



In [79]:
```
1 b=data[0:1000]
2
```

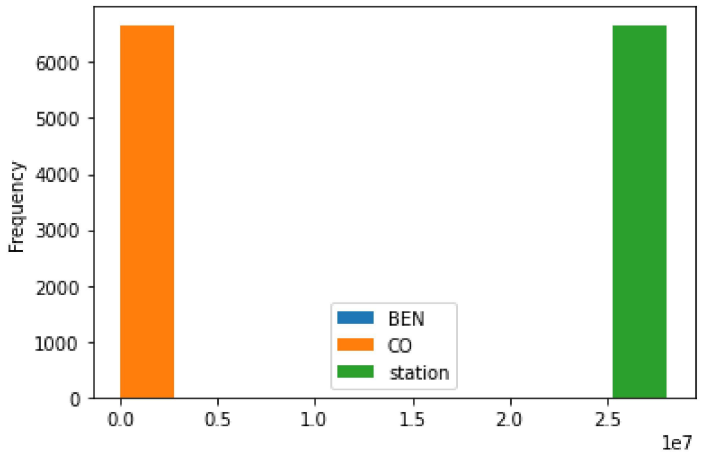
In [80]:
```
1 b.plot.bar()
```

Out[80]: <AxesSubplot:>



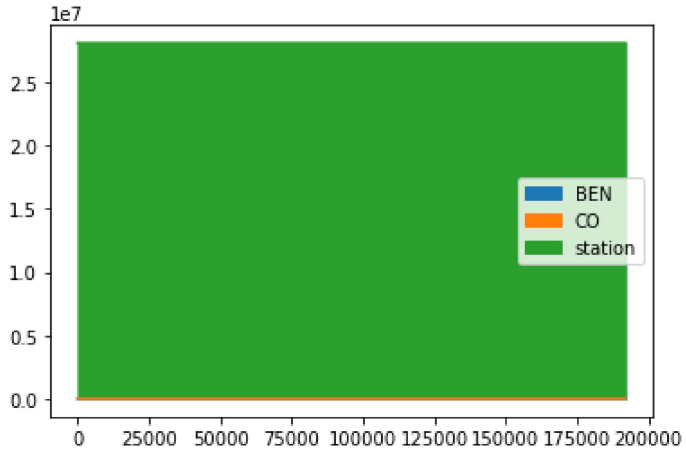
In [81]:
```
1 data.plot.hist()
2
```

Out[81]: <AxesSubplot:ylabel='Frequency'>



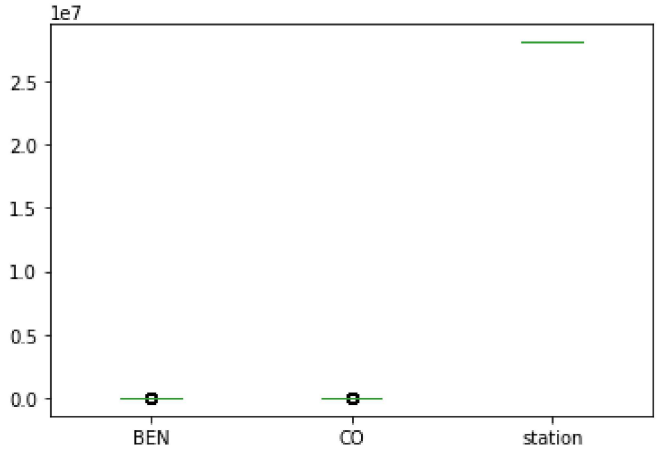
In [82]:
```
1 data.plot.area()
```

Out[82]: <AxesSubplot:>



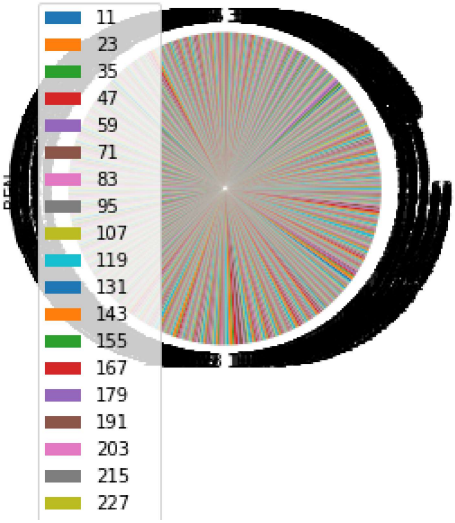
In [83]:
```
1 data.plot.box()
```

Out[83]: <AxesSubplot:>



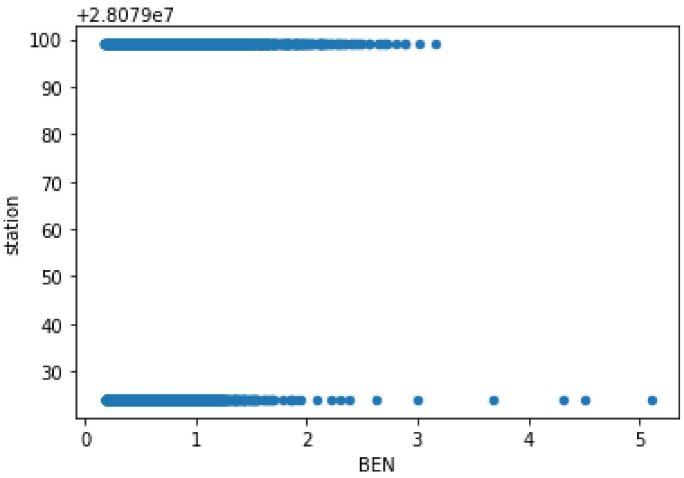
In [84]:
```
1 b.plot.pie(y='BEN' )
```

Out[84]: <AxesSubplot:ylabel='BEN'>



In [85]:
```
1 data.plot.scatter(x='BEN' ,y='station')
2
```

Out[85]: <AxesSubplot:xlabel='BEN', ylabel='station'>



In [86]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5000 entries, 11 to 99695
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     5000 non-null   object 
 1   BEN      5000 non-null   float64
 2   CO       5000 non-null   float64
 3   EBE      5000 non-null   float64
 4   MXY      5000 non-null   float64
 5   NMHC     5000 non-null   float64
 6   NO_2     5000 non-null   float64
 7   NOx      5000 non-null   float64
 8   OXY      5000 non-null   float64
 9   O_3      5000 non-null   float64
 10  PM10     5000 non-null   float64
 11  PM25     5000 non-null   float64
 12  PXY      5000 non-null   float64
 13  SO_2     5000 non-null   float64
 14  TCH      5000 non-null   float64
 15  TOL      5000 non-null   float64
 16  station  5000 non-null   int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 703.1+ KB
```

In [87]:
```
1 df.describe()
2 
```

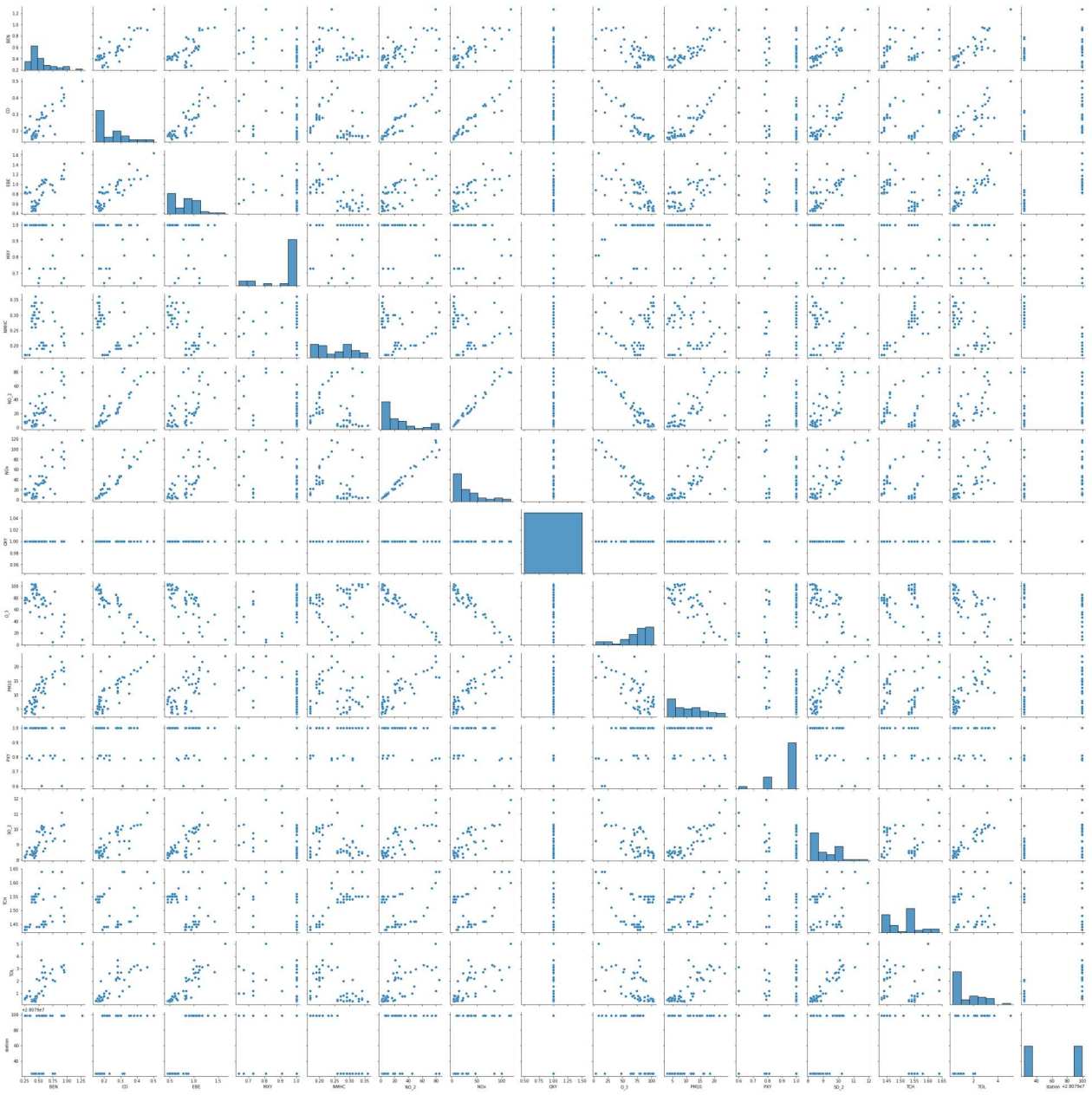
Out[87]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	0.718836	0.315760	0.921702	0.889666	0.236270	37.125532	52.664030	0.914976
std	0.394176	0.141225	0.537394	0.383965	0.100561	24.190899	44.025602	0.197877
min	0.200000	0.140000	0.140000	0.140000	0.000000	1.290000	2.780000	0.200000
25%	0.430000	0.220000	0.540000	0.660000	0.180000	18.370001	22.402500	1.000000
50%	0.610000	0.280000	0.880000	1.000000	0.220000	32.799999	41.045000	1.000000
75%	0.880000	0.360000	1.100000	1.000000	0.280000	52.320000	69.749998	1.000000
max	3.170000	1.590000	4.700000	6.810000	0.930000	133.399994	409.299988	2.380000

In [88]:
```
1 df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
2  'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [89]:
```
1 sns.pairplot(df1[0:50])
```

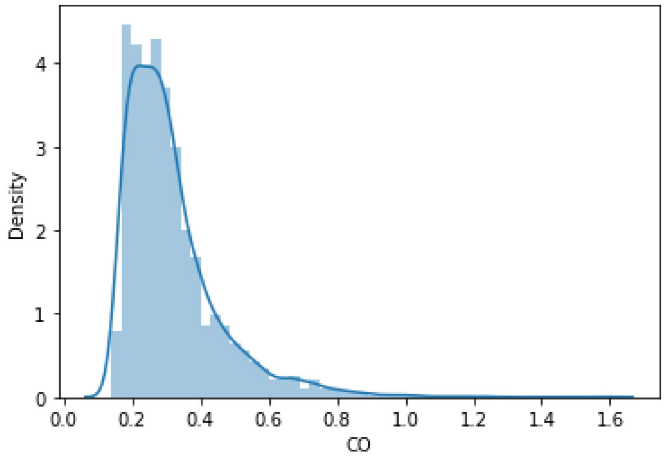
Out[89]: <seaborn.axisgrid.PairGrid at 0x27d231ea160>



In [90]:
```
1 sns.distplot(df1['CO'])
2
```

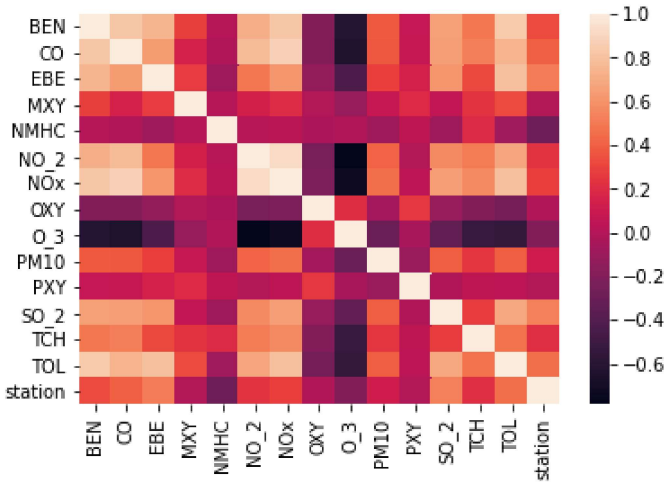
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `hi
stplot` (an axes-level function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

Out[90]: <AxesSubplot:xlabel='CO', ylabel='Density'>



In [91]:
```
1 sns.heatmap(df1.corr())
```

Out[91]: <AxesSubplot:>



In [92]:
```
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 [93]:
```
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 [94]:
```
1 from sklearn.linear_model import LinearRegression
2 lr=LinearRegression()
3 lr.fit(x_train,y_train)
4
```

Out[94]:
```
LinearRegression()
```

In [95]:
```
1 lr.intercept_
```

Out[95]:
```
28078903.631151836
```

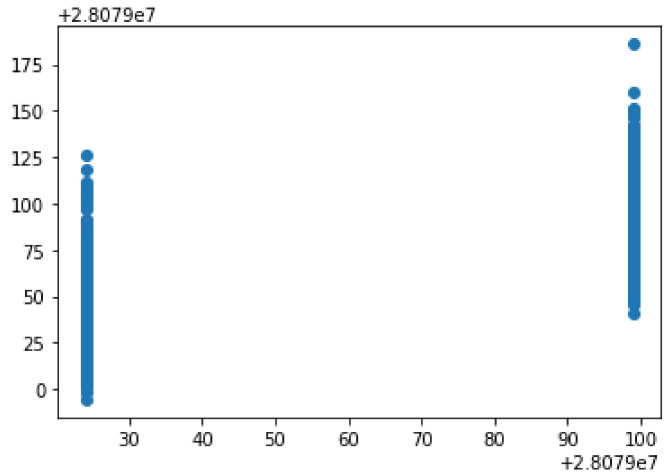
In [96]:
```
1 coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
2 coeff
```

Out[96]:

	Co-efficient
BEN	-40.594499
CO	127.933412
EBE	26.352137
MXY	-9.522224
NMHC	-81.076531
NO_2	0.238844
NOx	-0.591782
OXY	18.170288
O_3	-0.012530
PM10	-0.234078
PXY	-4.855637
SO_2	5.137631
TCH	68.469257
TOL	5.633101

In [97]:
```
1 prediction =lr.predict(x_test)
2 plt.scatter(y_test,prediction)
```

Out[97]:
```
<matplotlib.collections.PathCollection at 0x27d13973610>
```



In [98]:
```
lr.score(x_test,y_test)

```

Out[98]: 0.5238930319077906

In [99]:
```
lr.score(x_train,y_train)

```

Out[99]: 0.5309839926667481

In [100]:
```
from sklearn.linear_model import Ridge,Lasso

```

In [101]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)

```

Out[101]: Ridge(alpha=10)

In [102]:
```
rr.score(x_test,y_test)

```

Out[102]: 0.5238068359481874

In [103]:
```
rr.score(x_train,y_train)

```

Out[103]: 0.5178985565165815

In [104]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[104]: Lasso(alpha=10)

In [105]:
```
la.score(x_train,y_train)

```

Out[105]: 0.29152813192242955

In [106]:
```
la.score(x_test,y_test)

```

Out[106]: 0.3035732894314941

In [107]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)

```

Out[107]: ElasticNet()

In [108]:
```
en.coef_

```

Out[108]:
```
array([-0.        ,  0.02068565,  3.84225816, -1.11487808, -0.71550587,
       -0.08694377, -0.21974587,  0.        , -0.14437293, -0.18859655,
       -0.        ,  6.07894556,  0.        ,  5.2075267 ])
```

In [109]:
```
en.intercept_
```

Out[109]: 28079010.389548622

In [110]:
```
prediction=en.predict(x_test)
```

In [111]:
```
en.score(x_test,y_test)
```

Out[111]: 0.3673923100502474

In [112]:
```
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)))
```

```
27.294127269662916
889.59823791494
29.826133472425486
```

In [113]:
```
from sklearn.linear_model import LogisticRegression
```

In [114]:
```
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 [115]:
```
feature_matrix.shape
```

Out[115]: (5000, 14)

In [116]:
```
target_vector.shape
```

Out[116]: (5000,)

In [117]:
```
from sklearn.preprocessing import StandardScaler
```

In [118]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [119]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[119]: LogisticRegression(max_iter=10000)

In [120]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [121]:
```
prediction=logr.predict(observation)
print(prediction)

```

```
[28079099]
```

In [122]:
```
logr.classes_

```

Out[122]:
```
array([28079024, 28079099], dtype=int64)
```

In [123]:
```
logr.score(fs,target_vector)

```

Out[123]:
```
0.8946
```

In [124]:
```
logr.predict_proba(observation)[0][0]

```

Out[124]:
```
0.0
```

In [125]:
```
logr.predict_proba(observation)

```

Out[125]:
```
array([[0., 1.]])
```

In [126]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [127]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[127]:
```
RandomForestClassifier()
```

In [128]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [129]:
```
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[129]:
```
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 [130]:
```
grid_search.best_score_
```

Out[130]:
```
0.9437142857142857
```

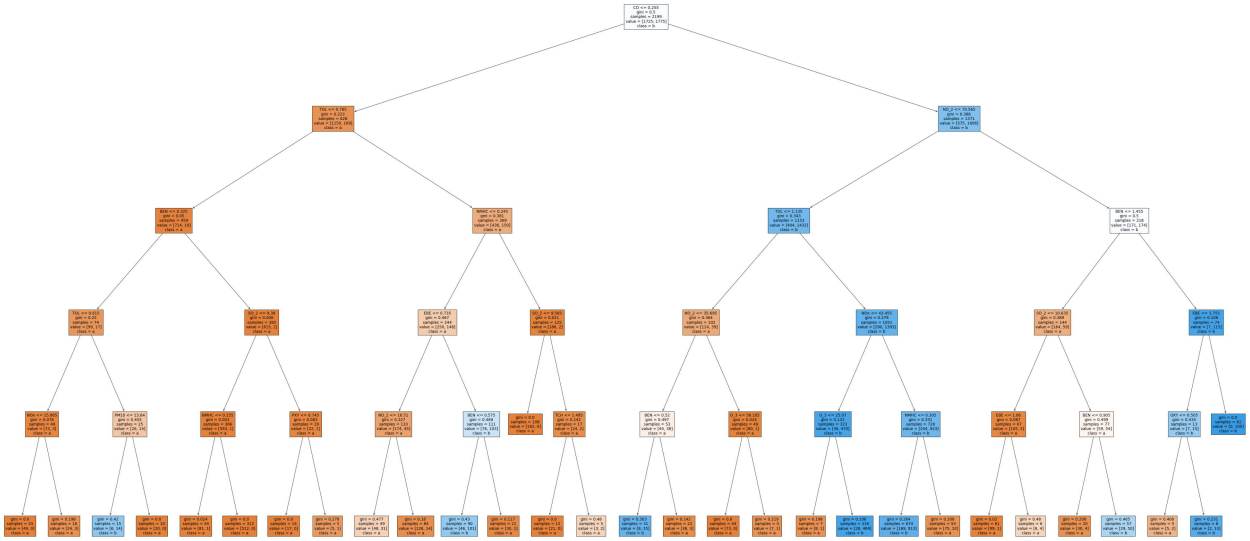
In [131]:
```
rfc_best=grid_search.best_estimator_
```

In [132]:

```
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[132]:
```
[Text(2310.315789473684, 1993.2, 'CO <= 0.255\ngini = 0.5\nsamples = 2199\nvalue = [1725,
1775]\nclass = b'),
 Text(1194.3157894736842, 1630.8000000000002, 'TOL <= 0.785\ngini = 0.223\nsamples = 828\n
value = [1150, 169]\nclass = a'),
 Text(626.5263157894736, 1268.4, 'BEN <= 0.325\ngini = 0.05\nsamples = 459\nvalue = [714,
19]\nclass = a'),
 Text(313.2631578947368, 906.0, 'TOL <= 0.615\ngini = 0.25\nsamples = 74\nvalue = [99, 17]
\nclass = a'),
 Text(156.6315789473684, 543.5999999999999, 'NOx <= 15.865\ngini = 0.076\nsamples = 49\nva
lue = [73, 3]\nclass = a'),
 Text(78.3157894736842, 181.19999999999982, 'gini = 0.0\nsamples = 33\nvalue = [49, 0]\ncl
ass = a'),
 Text(234.9473684210526, 181.19999999999982, 'gini = 0.198\nsamples = 16\nvalue = [24, 3]
\nclass = a'),
 Text(469.8947368421052, 543.5999999999999, 'PM10 <= 13.84\ngini = 0.455\nsamples = 25\nva
lue = [26, 14]\nclass = a'),
 Text(391.57894736842104, 181.19999999999982, 'gini = 0.42\nsamples = 15\nvalue = [6, 14]
\nclass = b'),
 Text(548.2105263157895, 181.19999999999982, 'gini = 0.0\nsamples = 10\nvalue = [20, 0]\nc
lass = a'),
 Text(939.7894736842104, 906.0, 'SO_2 <= 9.38\ngini = 0.006\nsamples = 385\nvalue = [615,
2]\nclass = a'),
 Text(783.1578947368421, 543.5999999999999, 'NMHC <= 0.155\ngini = 0.003\nsamples = 366\nv
alue = [593, 1]\nclass = a'),
 Text(704.8421052631578, 181.19999999999982, 'gini = 0.024\nsamples = 54\nvalue = [81, 1]
\nclass = a'),
 Text(861.4736842105262, 181.19999999999982, 'gini = 0.0\nsamples = 312\nvalue = [512, 0]
\nclass = a'),
 Text(1096.421052631579, 543.5999999999999, 'PXY <= 0.745\ngini = 0.083\nsamples = 19\nval
ue = [22, 1]\nclass = a'),
 Text(1018.1052631578947, 181.19999999999982, 'gini = 0.0\nsamples = 14\nvalue = [17, 0]\n
class = a'),
 Text(1174.7368421052631, 181.19999999999982, 'gini = 0.278\nsamples = 5\nvalue = [5, 1]\n
class = a'),
 Text(1762.1052631578946, 1268.4, 'NMHC <= 0.245\ngini = 0.381\nsamples = 369\nvalue = [43
6, 150]\nclass = a'),
 Text(1566.3157894736842, 906.0, 'EBE <= 0.735\ngini = 0.467\nsamples = 244\nvalue = [250,
148]\nclass = a'),
 Text(1409.6842105263156, 543.5999999999999, 'NO_2 <= 16.51\ngini = 0.327\nsamples = 133\n
value = [174, 45]\nclass = a'),
 Text(1331.3684210526314, 181.19999999999982, 'gini = 0.477\nsamples = 49\nvalue = [48, 3
1]\nclass = a'),
 Text(1488.0, 181.19999999999982, 'gini = 0.18\nsamples = 84\nvalue = [126, 14]\nclass =
a'),
 Text(1722.9473684210525, 543.5999999999999, 'BEN <= 0.575\ngini = 0.489\nsamples = 111\nv
alue = [76, 103]\nclass = b'),
 Text(1644.6315789473683, 181.19999999999982, 'gini = 0.43\nsamples = 90\nvalue = [46, 10
1]\nclass = b'),
 Text(1801.2631578947367, 181.19999999999982, 'gini = 0.117\nsamples = 21\nvalue = [30, 2]
\nclass = a'),
 Text(1957.8947368421052, 906.0, 'SO_2 <= 9.565\ngini = 0.021\nsamples = 125\nvalue = [18
6, 2]\nclass = a'),
 Text(1879.5789473684208, 543.5999999999999, 'gini = 0.0\nsamples = 108\nvalue = [162, 0]
\nclass = a'),
 Text(2036.2105263157894, 543.5999999999999, 'TCH <= 1.495\ngini = 0.142\nsamples = 17\nva
lue = [24, 2]\nclass = a'),
 Text(1957.8947368421052, 181.19999999999982, 'gini = 0.0\nsamples = 12\nvalue = [21, 0]\n
class = a'),
 Text(2114.5263157894738, 181.19999999999982, 'gini = 0.48\nsamples = 5\nvalue = [3, 2]\nc
lass = a'),
 Text(3426.315789473684, 1630.8000000000002, 'NO_2 <= 70.565\ngini = 0.388\nsamples = 1371
\nvalue = [575, 1606]\nclass = b'),
 Text(2819.368421052631, 1268.4, 'TOL <= 1.135\ngini = 0.343\nsamples = 1153\nvalue = [40
```

```
4, 1432]\nclass = b'),
 Text(2506.1052631578946, 906.0, 'NO_2 <= 35.695\ngini = 0.364\nsamples = 102\nvalue = [12
4, 39]\nclass = a'),
 Text(2349.4736842105262, 543.5999999999999, 'BEN <= 0.52\ngini = 0.497\nsamples = 53\nval
ue = [44, 38]\nclass = a'),
 Text(2271.157894736842, 181.19999999999982, 'gini = 0.303\nsamples = 31\nvalue = [8, 35]
\nclass = b'),
 Text(2427.7894736842104, 181.19999999999982, 'gini = 0.142\nsamples = 22\nvalue = [36, 3]
\nclass = a'),
 Text(2662.736842105263, 543.5999999999999, 'O_3 <= 56.105\ngini = 0.024\nsamples = 49\nva
lue = [80, 1]\nclass = a'),
 Text(2584.4210526315787, 181.19999999999982, 'gini = 0.0\nsamples = 44\nvalue = [73, 0]\n
class = a'),
 Text(2741.052631578947, 181.19999999999982, 'gini = 0.219\nsamples = 5\nvalue = [7, 1]\nc
lass = a'),
 Text(3132.6315789473683, 906.0, 'NOx <= 43.455\ngini = 0.279\nsamples = 1051\nvalue = [28
0, 1393]\nclass = b'),
 Text(2976.0, 543.5999999999999, 'O_3 <= 25.07\ngini = 0.132\nsamples = 323\nvalue = [36,
470]\nclass = b'),
 Text(2897.6842105263154, 181.19999999999982, 'gini = 0.198\nsamples = 7\nvalue = [8, 1]\n
class = a'),
 Text(3054.315789473684, 181.19999999999982, 'gini = 0.106\nsamples = 316\nvalue = [28, 46
9]\nclass = b'),
 Text(3289.2631578947367, 543.5999999999999, 'NMHC <= 0.305\ngini = 0.331\nsamples = 728\n
value = [244, 923]\nclass = b'),
 Text(3210.9473684210525, 181.19999999999982, 'gini = 0.264\nsamples = 674\nvalue = [169,
913]\nclass = b'),
 Text(3367.578947368421, 181.19999999999982, 'gini = 0.208\nsamples = 54\nvalue = [75, 10]
\nclass = a'),
 Text(4033.2631578947367, 1268.4, 'BEN <= 1.455\ngini = 0.5\nsamples = 218\nvalue = [171,
174]\nclass = b'),
 Text(3759.1578947368416, 906.0, 'SO_2 <= 10.635\ngini = 0.389\nsamples = 144\nvalue = [16
4, 59]\nclass = a'),
 Text(3602.5263157894733, 543.5999999999999, 'EBE <= 1.06\ngini = 0.087\nsamples = 67\nval
ue = [105, 5]\nclass = a'),
 Text(3524.210526315789, 181.19999999999982, 'gini = 0.02\nsamples = 61\nvalue = [99, 1]\n
class = a'),
 Text(3680.8421052631575, 181.19999999999982, 'gini = 0.48\nsamples = 6\nvalue = [6, 4]\nc
lass = a'),
 Text(3915.7894736842104, 543.5999999999999, 'BEN <= 0.905\ngini = 0.499\nsamples = 77\nva
lue = [59, 54]\nclass = a'),
 Text(3837.4736842105262, 181.19999999999982, 'gini = 0.208\nsamples = 20\nvalue = [30, 4]
\nclass = a'),
 Text(3994.1052631578946, 181.19999999999982, 'gini = 0.465\nsamples = 57\nvalue = [29, 5
0]\nclass = b'),
 Text(4307.368421052632, 906.0, 'EBE <= 1.755\ngini = 0.108\nsamples = 74\nvalue = [7, 11
5]\nclass = b'),
 Text(4229.0526315789475, 543.5999999999999, 'OXY <= 0.505\ngini = 0.434\nsamples = 13\nva
lue = [7, 15]\nclass = b'),
 Text(4150.736842105262, 181.19999999999982, 'gini = 0.408\nsamples = 5\nvalue = [5, 2]\nc
lass = a'),
 Text(4307.368421052632, 181.19999999999982, 'gini = 0.231\nsamples = 8\nvalue = [2, 13]\n
class = b'),
 Text(4385.684210526316, 543.5999999999999, 'gini = 0.0\nsamples = 61\nvalue = [0, 100]\nc
lass = b')]
```



Conclusion

Linear Regression=0.5309839926667481

Ridge Regression=0.5178985565165815

Lasso Regression=0.29152813192242955

ElasticNet Regression=0.3673923100502474

Logistic Regression=0.8946

Random Forest=0.9437142857142857

Random Forest is Suitable for this Dataset