Vijay(P8) 3/08/2023

In [452]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [453]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2008.csv")
df
```

Out[453]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	TCH	T
0	2008-06-01 01:00:00	NaN	0.47	NaN	NaN	NaN	83.089996	120.699997	NaN	16.990000	16.889999	10.40	NaN	8.98	NaN	I
1	2008-06-01 01:00:00	NaN	0.59	NaN	NaN	NaN	94.820000	130.399994	NaN	17.469999	19.040001	NaN	NaN	5.85	NaN	I
2	2008-06-01 01:00:00	NaN	0.55	NaN	NaN	NaN	75.919998	104.599998	NaN	13.470000	20.270000	NaN	NaN	6.95	NaN	I
3	2008-06-01 01:00:00	NaN	0.36	NaN	NaN	NaN	61.029999	66.559998	NaN	23.110001	10.850000	NaN	NaN	5.96	NaN	I
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.160000	21.90	1.43	10.92	1.53	(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002	5.450000	5.15	1.86	9.68	1.23	1
226388	2008-11-01 00:00:00	NaN	0.30	NaN	NaN	NaN	41.880001	48.500000	NaN	35.830002	15.020000	NaN	NaN	8.90	NaN	I
226389	2008-11-01 00:00:00	0.25	NaN	0.56	NaN	0.11	83.610001	102.199997	NaN	14.130000	17.540001	13.91	NaN	7.00	1.56	(
226390	2008-11-01 00:00:00	0.54	NaN	2.70	NaN	0.18	70.639999	81.860001	NaN	NaN	11.910000	NaN	NaN	8.02	1.57	1
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000	12.690000	11.42	1.98	8.74	1.43	1

226392 rows × 17 columns

In [454]:
```
df=df.dropna()
```

In [455]:
```
df.columns

```

Out[455]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [456]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25631 entries, 4 to 226391
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     25631 non-null  object 
 1   BEN      25631 non-null  float64
 2   CO       25631 non-null  float64
 3   EBE      25631 non-null  float64
 4   MXY      25631 non-null  float64
 5   NMHC     25631 non-null  float64
 6   NO_2     25631 non-null  float64
 7   NOx      25631 non-null  float64
 8   OXY      25631 non-null  float64
 9   O_3      25631 non-null  float64
 10  PM10     25631 non-null  float64
 11  PM25     25631 non-null  float64
 12  PXY      25631 non-null  float64
 13  SO_2     25631 non-null  float64
 14  TCH      25631 non-null  float64
 15  TOL      25631 non-null  float64
 16  station  25631 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.5+ MB
```

In [457]:
```
1 data=df[['BEN', 'CO','station']]
2 data
3
```

Out[457]:

	BEN	CO	station
4	1.68	0.80	28079006
21	0.32	0.37	28079024
25	0.73	0.39	28079099
30	1.95	0.51	28079006
47	0.36	0.39	28079024
...	...	...	...
226362	0.47	0.35	28079024
226366	0.92	0.46	28079099
226371	1.83	0.53	28079006
226387	0.48	0.30	28079024
226391	0.75	0.36	28079099

25631 rows × 3 columns

In [458]:
```
df=df.head(100000)
df
```

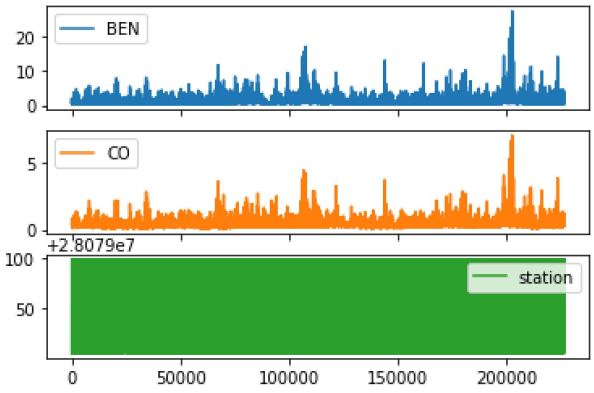
Out[458]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	TC
4	2008-06-01 01:00:00	1.68	0.80	1.70	3.01	0.30	105.199997	214.899994	1.61	12.120000	37.160000	21.900000	1.43	10.92	1.
21	2008-06-01 01:00:00	0.32	0.37	1.00	0.39	0.33	21.580000	22.180000	1.00	35.770000	7.900000	6.140000	1.00	5.39	1.
25	2008-06-01 01:00:00	0.73	0.39	1.04	1.70	0.18	64.839996	86.709999	1.31	23.379999	14.760000	9.840000	1.22	6.82	1.
30	2008-06-01 02:00:00	1.95	0.51	1.98	3.77	0.24	79.750000	143.399994	2.03	18.090000	31.139999	18.410000	1.81	8.97	1.
47	2008-06-01 02:00:00	0.36	0.39	0.39	0.50	0.34	26.790001	27.389999	1.00	33.029999	7.620000	6.250000	0.38	5.59	1.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
226362	2008-10-31 23:00:00	0.47	0.35	0.65	1.00	0.33	22.480000	25.020000	1.00	33.509998	10.200000	7.680000	1.84	9.47	1.
226366	2008-10-31 23:00:00	0.92	0.46	1.21	2.75	0.19	78.440002	106.199997	1.70	18.320000	14.140000	10.590000	1.98	9.66	1.
226371	2008-11-01 00:00:00	1.83	0.53	2.22	4.51	0.17	93.260002	158.399994	2.38	18.770000	20.750000	18.620001	2.10	12.27	1.
226387	2008-11-01 00:00:00	0.48	0.30	0.57	1.00	0.31	13.050000	14.160000	0.91	57.400002	5.450000	5.150000	1.86	9.68	1.
226391	2008-11-01 00:00:00	0.75	0.36	1.20	2.75	0.16	58.240002	74.239998	1.64	31.910000	12.690000	11.420000	1.98	8.74	1.

25631 rows × 17 columns

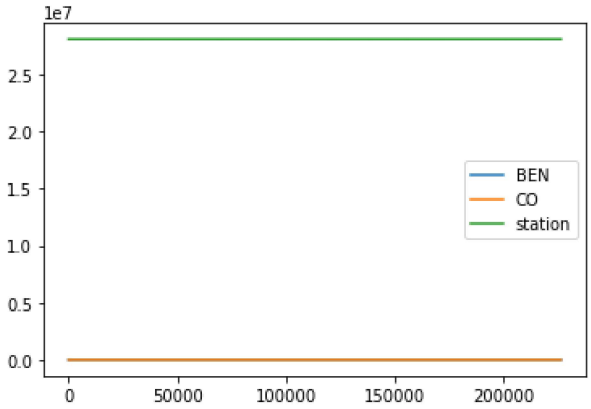
In [459]:
```
data.plot.line(subplots=True)
```

Out[459]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [460]:
```
1 data.plot.line()
2
```

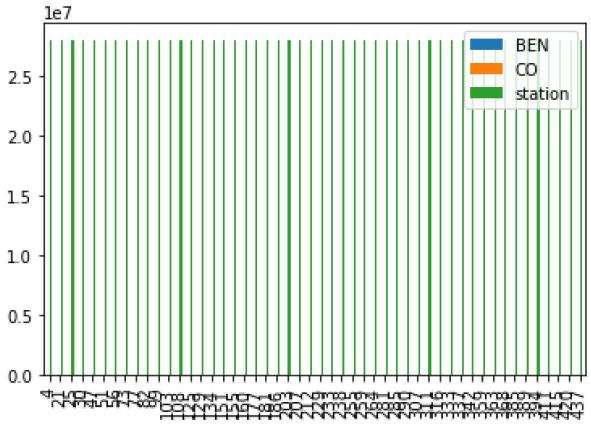
Out[460]: <AxesSubplot:>



In [461]:
```
1 b=data[0:50]
2
```

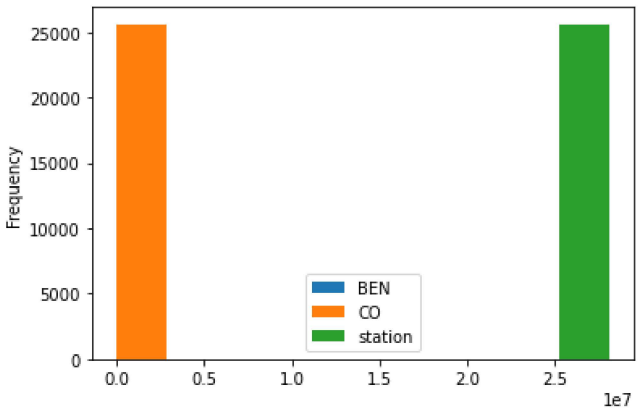
In [462]:
```
1 b.plot.bar()
```

Out[462]: <AxesSubplot:>



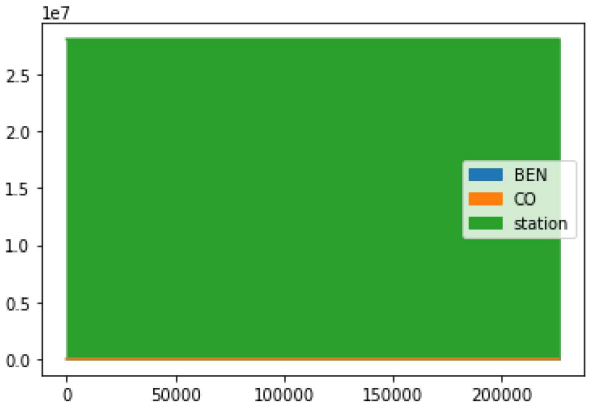
In [463]:
```
1 data.plot.hist()
2
```

Out[463]: <AxesSubplot:ylabel='Frequency'>



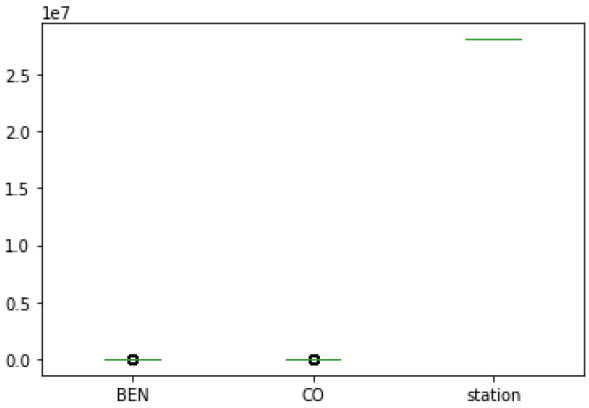
In [464]:
```
1 data.plot.area()
```

Out[464]: <AxesSubplot:>



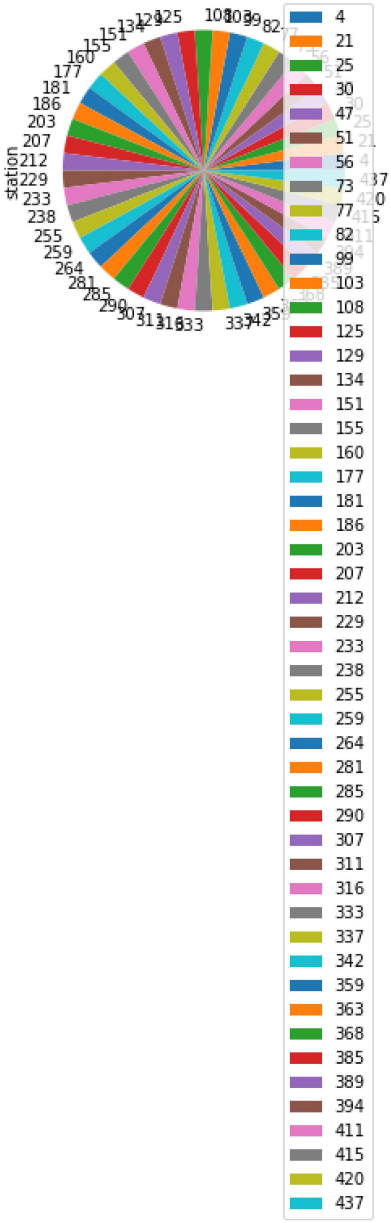
In [465]:
```
1 data.plot.box()
```

Out[465]: <AxesSubplot:>



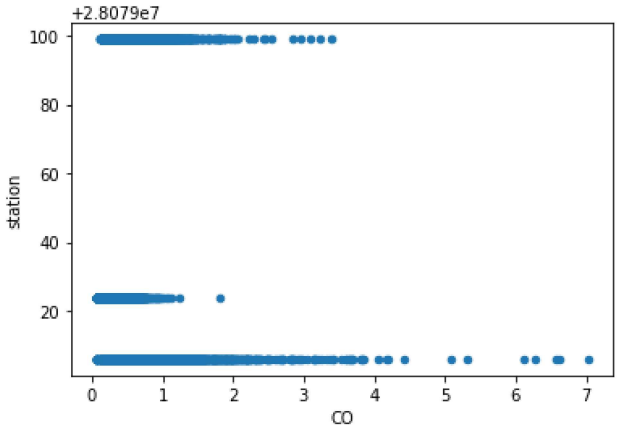
In [466]:
```
1 b.plot.pie(y='station' )
```

Out[466]: <AxesSubplot:ylabel='station'>



In [467]:
```
1 data.plot.scatter(x='CO' ,y='station')
2
```

Out[467]: <AxesSubplot:xlabel='CO', ylabel='station'>



In [468]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25631 entries, 4 to 226391
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     25631 non-null  object
 1   BEN      25631 non-null  float64
 2   CO       25631 non-null  float64
 3   EBE      25631 non-null  float64
 4   MXY      25631 non-null  float64
 5   NMHC     25631 non-null  float64
 6   NO_2     25631 non-null  float64
 7   NOx      25631 non-null  float64
 8   OXY      25631 non-null  float64
 9   O_3      25631 non-null  float64
 10  PM10     25631 non-null  float64
 11  PM25     25631 non-null  float64
 12  PXY      25631 non-null  float64
 13  SO_2     25631 non-null  float64
 14  TCH      25631 non-null  float64
 15  TOL      25631 non-null  float64
 16  station  25631 non-null  int64
dtypes: float64(15), int64(1), object(1)
memory usage: 3.5+ MB
```

In [469]:
```
1 df.describe()
2
```

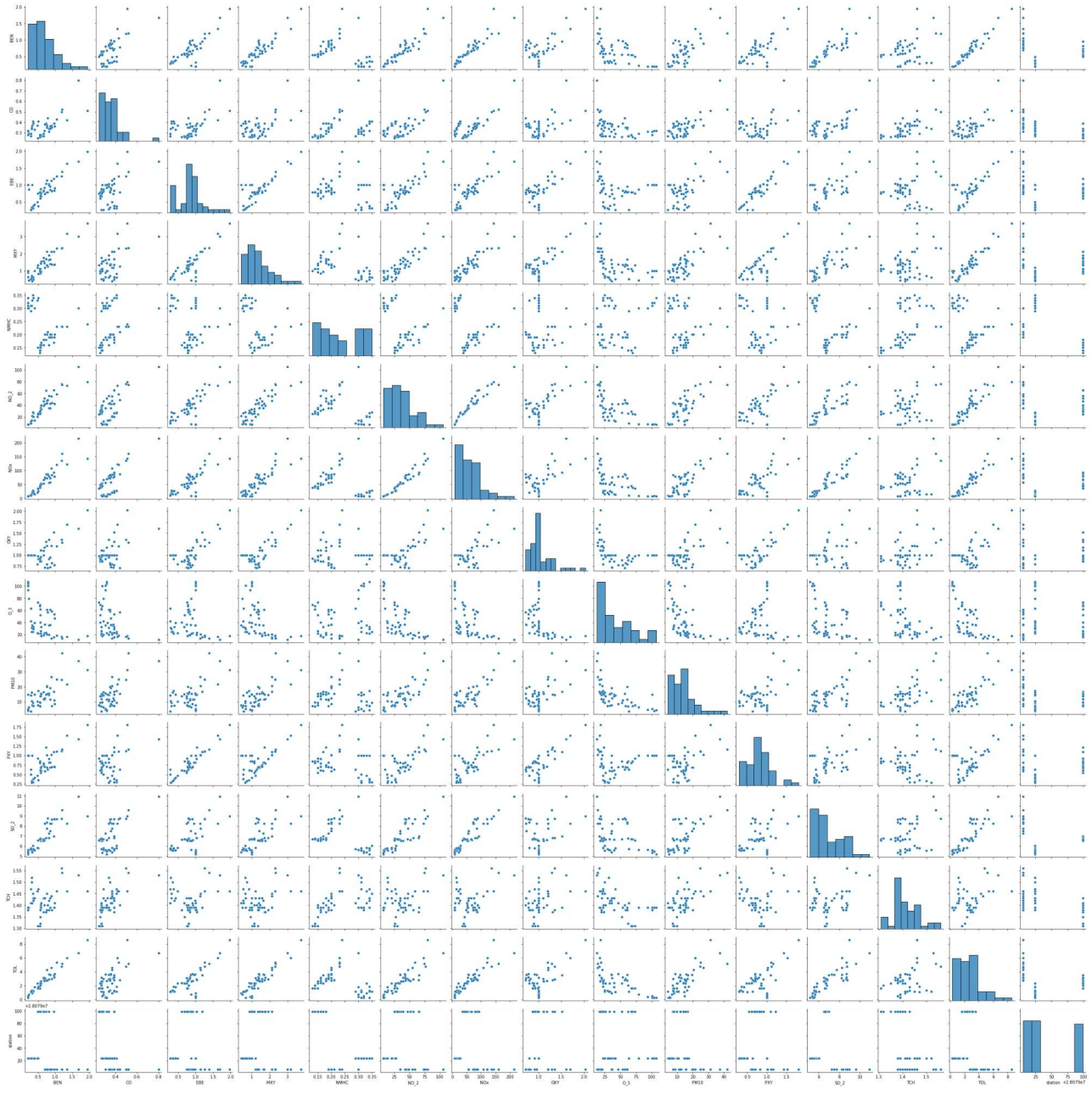
Out[469]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	
count	25631.000000	25631.000000	25631.000000	25631.000000	25631.000000	25631.000000	25631.000000	25631.000000	25631.
mean	1.090541	0.440632	1.352355	2.446045	0.213323	54.225261	98.007732	1.479964	41.
std	1.146461	0.317853	1.118191	2.390023	0.123409	38.164647	101.448238	1.258928	29.
min	0.100000	0.060000	0.170000	0.240000	0.000000	0.240000	2.110000	0.140000	1.
25%	0.430000	0.260000	0.740000	1.000000	0.130000	25.719999	32.635000	0.870000	16.
50%	0.750000	0.350000	1.000000	1.620000	0.190000	48.000000	71.110001	1.000000	37.
75%	1.320000	0.510000	1.580000	3.105000	0.270000	74.924999	131.550003	1.760000	60.
max	27.230000	7.030000	26.740000	55.889999	1.760000	554.900024	2004.000000	28.020000	201.

In [470]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [471]:
```
sns.pairplot(df1[0:50])
```

Out[471]: <seaborn.axisgrid.PairGrid at 0x210881342b0>

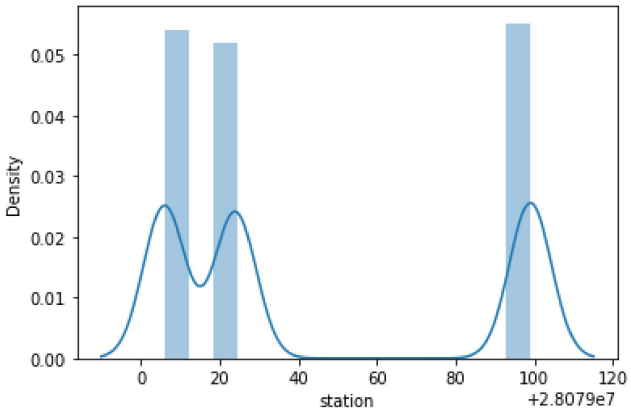


In [472]:
```
sns.distplot(df1['station'])

```

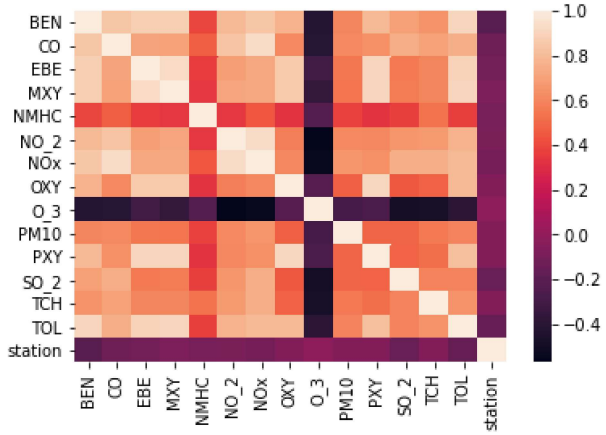
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `di
splot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[472]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [473]:
```
sns.heatmap(df1.corr())
```

Out[473]: <AxesSubplot:>



In [474]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']

```

In [475]:
```
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 [476]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)

```

Out[476]: LinearRegression()

In [477]:
```
lr.intercept_
```

Out[477]: 28079035.811439633

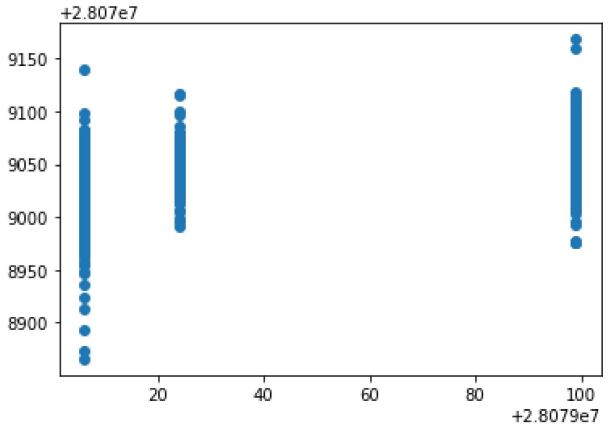
In [478]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[478]:

	Co-efficient
BEN	-25.190075
CO	-0.667850
EBE	-1.682079
MXY	7.494390
NMHC	-23.235397
NO_2	-0.034093
NOx	0.130631
OXY	4.430467
O_3	-0.130568
PM10	0.127106
PXY	1.685818
SO_2	-0.615968
TCH	16.179718
TOL	-1.878799

In [479]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[479]: <matplotlib.collections.PathCollection at 0x21086e03700>



In [480]:
```
lr.score(x_test,y_test)

```

Out[480]: 0.15407867665456487

In [481]:
```
lr.score(x_train,y_train)

```

Out[481]: 0.138750773798046

In [482]:
```
from sklearn.linear_model import Ridge,Lasso

```

In [483]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)

```

Out[483]: Ridge(alpha=10)

In [484]:
```
rr.score(x_test,y_test)

```

Out[484]: 0.15377494353774235

In [485]:
```
rr.score(x_train,y_train)

```

Out[485]: 0.13873120352817703

In [486]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[486]: Lasso(alpha=10)

In [487]:
```
la.score(x_train,y_train)

```

Out[487]: 0.04280450614345732

In [488]:
```
la.score(x_test,y_test)

```

Out[488]: 0.041193448520571385

In [489]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)

```

Out[489]: ElasticNet()

In [490]:
```
en.coef_

```

Out[490]:
```
array([-4.52952408, -0.        , -0.        ,  3.12910559, -0.        ,
        0.06483129,  0.02882664,  1.57323114, -0.14730816,  0.12449959,
        1.40133162, -0.9456563 ,  0.        , -2.52511834])
```

In [491]:
```
en.intercept_

```

Out[491]: 28079056.700558312

In [492]:
```
prediction=en.predict(x_test)

```

In [493]:
```
en.score(x_test,y_test)
```

Out[493]: 0.09545703741779221

In [494]:
```
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)))

```

```
35.72119451489043
1489.370230104188
38.5923597374427
```

In [495]:
```
from sklearn.linear_model import LogisticRegression

```

In [496]:
```
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 [497]:
```
feature_matrix.shape

```

Out[497]: (25631, 14)

In [498]:
```
target_vector.shape

```

Out[498]: (25631,)

In [499]:
```
from sklearn.preprocessing import StandardScaler
```

In [500]:
```
fs=StandardScaler().fit_transform(feature_matrix)

```

In [501]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[501]: LogisticRegression(max_iter=10000)

In [502]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]

```

In [503]:
```
prediction=logr.predict(observation)
print(prediction)

```

```
[28079099]
```

In [504]:
```
logr.classes_

```

Out[504]: array([28079006, 28079024, 28079099], dtype=int64)

In [505]:
```
logr.score(fs,target_vector)

```

Out[505]: 0.794194530061254

In [506]:
```
logr.predict_proba(observation)[0][0]

```

Out[506]: 8.321803242555043e-09

In [507]:
```
logr.predict_proba(observation)

```

Out[507]: array([[8.32180324e-09, 1.19114634e-13, 9.99999992e-01]])

In [508]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [509]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[509]: RandomForestClassifier()

In [510]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [511]:
```
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[511]:
```
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 [512]:
```
grid_search.best_score_
```

Out[512]:
```
0.8529067836197572
```

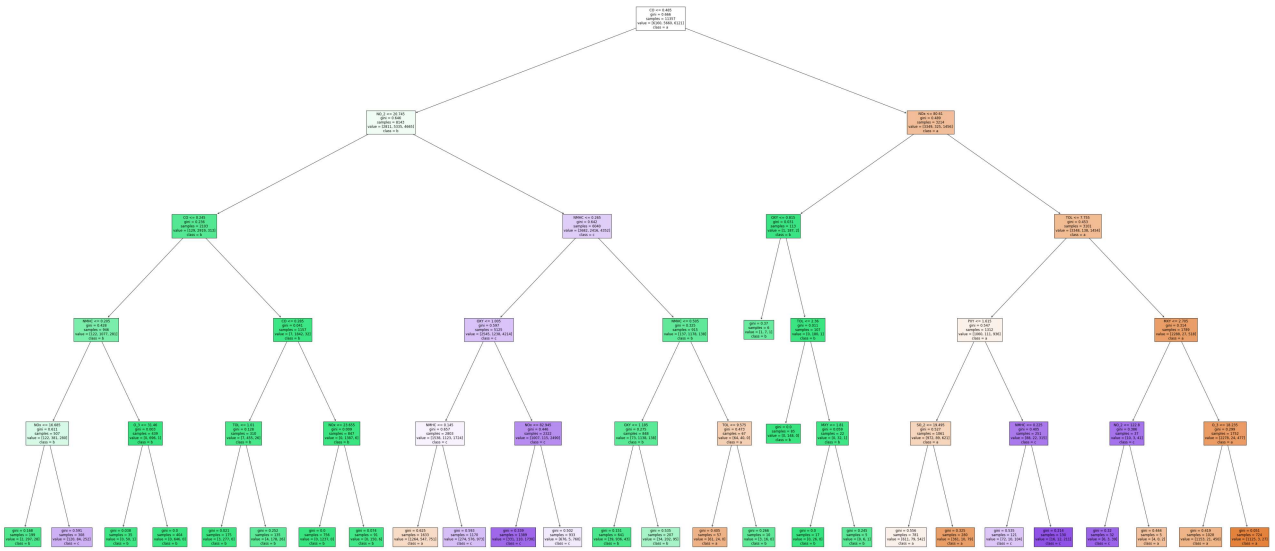
In [513]:
```
rfc_best=grid_search.best_estimator_
```

In [514]:

```
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'],filled=True
```

Out[514]:
```
[Text(2317.846153846154, 1993.2, 'CO <= 0.485\ngini = 0.666\nsamples = 11357\nvalue = [6160, 5660, 612
1]\nclass = a'),
 Text(1373.5384615384614, 1630.8000000000002, 'NO_2 <= 20.745\ngini = 0.646\nsamples = 8143\nvalue = [2
811, 5335, 4665]\nclass = b'),
 Text(686.7692307692307, 1268.4, 'CO <= 0.245\ngini = 0.236\nsamples = 2103\nvalue = [129, 2919, 313]\n
class = b'),
 Text(343.38461538461536, 906.0, 'NMHC <= 0.205\ngini = 0.428\nsamples = 946\nvalue = [122, 1077, 281]
\nclass = b'),
 Text(171.69230769230768, 543.5999999999999, 'NOx <= 16.685\ngini = 0.611\nsamples = 507\nvalue = [122,
381, 280]\nclass = b'),
 Text(85.84615384615384, 181.19999999999982, 'gini = 0.168\nsamples = 199\nvalue = [2, 297, 28]\nclass
= b'),
 Text(257.53846153846155, 181.19999999999982, 'gini = 0.591\nsamples = 308\nvalue = [120, 84, 252]\ncla
ss = c'),
 Text(515.0769230769231, 543.5999999999999, 'O_3 <= 31.46\ngini = 0.003\nsamples = 439\nvalue = [0, 69
6, 1]\nclass = b'),
 Text(429.23076923076917, 181.19999999999982, 'gini = 0.038\nsamples = 35\nvalue = [0, 50, 1]\nclass =
b'),
 Text(600.9230769230769, 181.19999999999982, 'gini = 0.0\nsamples = 404\nvalue = [0, 646, 0]\nclass =
b'),
 Text(1030.1538461538462, 906.0, 'CO <= 0.285\ngini = 0.041\nsamples = 1157\nvalue = [7, 1842, 32]\ncla
ss = b'),
 Text(858.4615384615383, 543.5999999999999, 'TOL <= 1.01\ngini = 0.128\nsamples = 310\nvalue = [7, 455,
26]\nclass = b'),
 Text(772.6153846153845, 181.19999999999982, 'gini = 0.021\nsamples = 175\nvalue = [3, 277, 0]\nclass =
b'),
 Text(944.3076923076923, 181.19999999999982, 'gini = 0.252\nsamples = 135\nvalue = [4, 178, 26]\nclass
= b'),
 Text(1201.8461538461538, 543.5999999999999, 'NOx <= 23.655\ngini = 0.009\nsamples = 847\nvalue = [0, 1
387, 6]\nclass = b'),
 Text(1116.0, 181.19999999999982, 'gini = 0.0\nsamples = 756\nvalue = [0, 1237, 0]\nclass = b'),
 Text(1287.6923076923076, 181.19999999999982, 'gini = 0.074\nsamples = 91\nvalue = [0, 150, 6]\nclass =
b'),
 Text(2060.3076923076924, 1268.4, 'NMHC <= 0.265\ngini = 0.642\nsamples = 6040\nvalue = [2682, 2416, 43
52]\nclass = c'),
 Text(1716.9230769230767, 906.0, 'OXY <= 1.005\ngini = 0.597\nsamples = 5125\nvalue = [2545, 1238, 421
4]\nclass = c'),
 Text(1545.230769230769, 543.5999999999999, 'NMHC <= 0.145\ngini = 0.657\nsamples = 2803\nvalue = [153
8, 1123, 1724]\nclass = c'),
 Text(1459.3846153846152, 181.19999999999982, 'gini = 0.625\nsamples = 1633\nvalue = [1264, 547, 751]\n
class = a'),
 Text(1631.0769230769229, 181.19999999999982, 'gini = 0.593\nsamples = 1170\nvalue = [274, 576, 973]\nc
lass = c'),
 Text(1888.6153846153845, 543.5999999999999, 'NOx <= 82.945\ngini = 0.446\nsamples = 2322\nvalue = [100
7, 115, 2490]\nclass = c'),
 Text(1802.7692307692307, 181.19999999999982, 'gini = 0.339\nsamples = 1389\nvalue = [331, 110, 1730]\n
class = c'),
 Text(1974.4615384615383, 181.19999999999982, 'gini = 0.502\nsamples = 933\nvalue = [676, 5, 760]\nclas
s = c'),
 Text(2403.6923076923076, 906.0, 'NMHC <= 0.505\ngini = 0.325\nsamples = 915\nvalue = [137, 1178, 138]
\nclass = b'),
 Text(2232.0, 543.5999999999999, 'OXY <= 1.185\ngini = 0.275\nsamples = 848\nvalue = [73, 1138, 138]\nc
lass = b'),
 Text(2146.153846153846, 181.19999999999982, 'gini = 0.151\nsamples = 641\nvalue = [39, 936, 43]\nclass
= b'),
 Text(2317.846153846154, 181.19999999999982, 'gini = 0.535\nsamples = 207\nvalue = [34, 202, 95]\nclass
= b'),
 Text(2575.3846153846152, 543.5999999999999, 'TOL <= 9.575\ngini = 0.473\nsamples = 67\nvalue = [64, 4
0, 0]\nclass = a'),
 Text(2489.5384615384614, 181.19999999999982, 'gini = 0.405\nsamples = 57\nvalue = [61, 24, 0]\nclass =
a'),
 Text(2661.230769230769, 181.19999999999982, 'gini = 0.266\nsamples = 10\nvalue = [3, 16, 0]\nclass =
b'),
 Text(3262.1538461538457, 1630.8000000000002, 'NOx <= 80.61\ngini = 0.489\nsamples = 3214\nvalue = [334
9, 325, 1456]\nclass = a'),
 Text(2747.076923076923, 1268.4, 'OXY <= 0.815\ngini = 0.031\nsamples = 113\nvalue = [1, 187, 2]\nclass
= b'),
 Text(2661.230769230769, 906.0, 'gini = 0.37\nsamples = 6\nvalue = [1, 7, 1]\nclass = b'),
 Text(2832.9230769230767, 906.0, 'TOL <= 2.36\ngini = 0.011\nsamples = 107\nvalue = [0, 180, 1]\nclass
= b'),
 Text(2747.076923076923, 543.5999999999999, 'gini = 0.0\nsamples = 85\nvalue = [0, 148, 0]\nclass =
```

```
b'),
 Text(2918.7692307692305, 543.5999999999999, 'MXY <= 1.81\ngini = 0.059\nsamples = 22\nvalue = [0, 32,
1]\nclass = b'),
 Text(2832.9230769230767, 181.19999999999982, 'gini = 0.0\nsamples = 17\nvalue = [0, 26, 0]\nclass =
b'),
 Text(3004.6153846153843, 181.19999999999982, 'gini = 0.245\nsamples = 5\nvalue = [0, 6, 1]\nclass =
b'),
 Text(3777.230769230769, 1268.4, 'TOL <= 7.755\ngini = 0.453\nsamples = 3101\nvalue = [3348, 138, 1454]
\nclass = a'),
 Text(3433.8461538461534, 906.0, 'PXY <= 1.615\ngini = 0.547\nsamples = 1312\nvalue = [1060, 111, 936]
\nclass = a'),
 Text(3262.1538461538457, 543.5999999999999, 'SO_2 <= 19.495\ngini = 0.527\nsamples = 1061\nvalue = [97
2, 89, 621]\nclass = a'),
 Text(3176.307692307692, 181.19999999999982, 'gini = 0.556\nsamples = 781\nvalue = [611, 79, 542]\nclas
s = a'),
 Text(3347.9999999999995, 181.19999999999982, 'gini = 0.325\nsamples = 280\nvalue = [361, 10, 79]\nclas
s = a'),
 Text(3605.5384615384614, 543.5999999999999, 'NMHC <= 0.225\ngini = 0.405\nsamples = 251\nvalue = [88,
22, 315]\nclass = c'),
 Text(3519.6923076923076, 181.19999999999982, 'gini = 0.535\nsamples = 121\nvalue = [72, 10, 104]\nclas
s = c'),
 Text(3691.3846153846152, 181.19999999999982, 'gini = 0.214\nsamples = 130\nvalue = [16, 12, 211]\nclas
s = c'),
 Text(4120.615384615385, 906.0, 'MXY <= 2.705\ngini = 0.314\nsamples = 1789\nvalue = [2288, 27, 518]\nc
lass = a'),
 Text(3948.9230769230767, 543.5999999999999, 'NO_2 <= 122.8\ngini = 0.386\nsamples = 37\nvalue = [10,
3, 41]\nclass = c'),
 Text(3863.076923076923, 181.19999999999982, 'gini = 0.32\nsamples = 32\nvalue = [6, 3, 39]\nclass =
c'),
 Text(4034.7692307692305, 181.19999999999982, 'gini = 0.444\nsamples = 5\nvalue = [4, 0, 2]\nclass =
a'),
 Text(4292.307692307692, 543.5999999999999, 'O_3 <= 18.235\ngini = 0.299\nsamples = 1752\nvalue = [227
8, 24, 477]\nclass = a'),
 Text(4206.461538461538, 181.19999999999982, 'gini = 0.419\nsamples = 1028\nvalue = [1153, 21, 450]\ncl
ass = a'),
 Text(4378.153846153846, 181.19999999999982, 'gini = 0.051\nsamples = 724\nvalue = [1125, 3, 27]\nclass
= a')]
```



Conclusion

Linear Regression=0.138750773798046

Ridge Regression=0.13873120352817703

Lasso Regression=0.04280450614345732

ElasticNet Regression=0.09545703741779221

Logistic Regression=0.794194530061254

Random Forest=0.8529067836197572

Random Forest is Suitable for this Dataset

In [ ]:
```
1
```