mk 3/08/2023

In [75]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [264]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2005.csv")
df
```

Out[264]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	TCH	TOL
0	2005-11-01 01:00:00	NaN	0.77	NaN	NaN	NaN	57.130001	128.699997	NaN	14.720000	14.91	10.65	NaN	4.62	NaN	NaN
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	NaN	1.59	7.80	1.35	7.98
2	2005-11-01 01:00:00	NaN	0.40	NaN	NaN	NaN	46.119999	53.000000	NaN	30.469999	14.60	NaN	NaN	5.76	NaN	NaN
3	2005-11-01 01:00:00	NaN	0.42	NaN	NaN	NaN	37.220001	52.009998	NaN	21.379999	15.16	NaN	NaN	6.60	NaN	NaN
4	2005-11-01 01:00:00	NaN	0.57	NaN	NaN	NaN	32.160000	36.680000	NaN	33.410000	5.00	NaN	NaN	3.00	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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	NaN	NaN	6.68	1.37	2.75
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	1.49	1.00	7.06	1.28	0.38
236997	2006-01-01 00:00:00	0.19	NaN	0.26	NaN	0.08	26.730000	30.809999	NaN	43.840000	4.31	2.93	NaN	13.20	1.28	0.56
236998	2006-01-01 00:00:00	0.14	NaN	1.00	NaN	0.06	13.770000	17.770000	NaN	NaN	5.00	NaN	NaN	5.81	1.25	0.22
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	2.11	1.09	11.07	1.30	1.29

237000 rows × 17 columns

In [265]:
```
df=df.dropna()
```

In [266]:
```
df.columns

```

Out[266]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [267]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20070 entries, 5 to 236999
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     20070 non-null  object 
 1   BEN      20070 non-null  float64
 2   CO       20070 non-null  float64
 3   EBE      20070 non-null  float64
 4   MXY      20070 non-null  float64
 5   NMHC     20070 non-null  float64
 6   NO_2     20070 non-null  float64
 7   NOx      20070 non-null  float64
 8   OXY      20070 non-null  float64
 9   O_3      20070 non-null  float64
 10  PM10     20070 non-null  float64
 11  PM25     20070 non-null  float64
 12  PXY      20070 non-null  float64
 13  SO_2     20070 non-null  float64
 14  TCH      20070 non-null  float64
 15  TOL      20070 non-null  float64
 16  station  20070 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 2.8+ MB
```

In [268]:
```
1 data=df[['BEN', 'CO','station']]
2 data
3
```

Out[268]:

	BEN	CO	station
5	1.92	0.88	28079006
22	0.30	0.22	28079024
25	0.67	0.49	28079099
31	3.10	0.84	28079006
48	0.39	0.20	28079024
...	...	...	...
236970	0.37	0.39	28079024
236973	0.92	0.45	28079099
236979	1.00	0.38	28079006
236996	0.39	0.54	28079024
236999	0.50	0.40	28079099

20070 rows × 3 columns

In [270]:
```
df=df.head(10000)
df
```

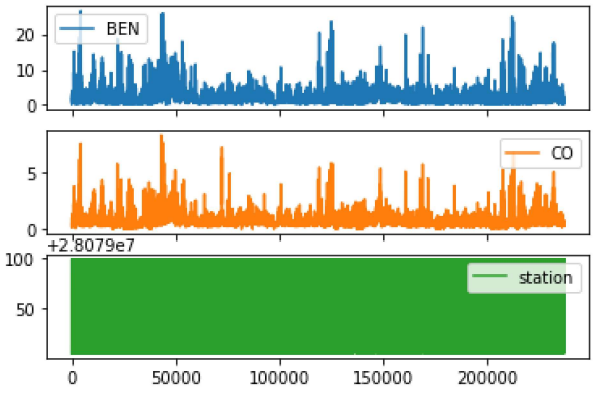
Out[270]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PM25	PXY	SO_2	TCH
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	17.600000	2.44	10.39	1.38
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	9.880000	6.020000	1.00	6.94	1.29
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	13.880000	10.260000	1.68	6.20	1.45
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	28.740000	21.870001	3.26	10.60	1.38
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	8.780000	5.350000	0.61	6.89	1.29
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9668	2005-11-16 00:00:00	0.84	0.57	0.59	1.44	0.15	44.279999	54.630001	0.67	11.660000	8.170000	6.250000	0.46	7.76	1.31
9671	2005-11-16 00:00:00	1.38	0.82	1.31	4.28	0.21	75.790001	155.100006	2.32	7.960000	13.750000	12.510000	1.90	10.33	1.42
9677	2005-11-16 01:00:00	2.33	0.87	2.84	6.20	0.20	91.400002	221.600006	3.50	9.030000	22.440001	13.710000	2.99	12.22	1.35
9695	2005-11-16 01:00:00	0.64	0.54	0.40	1.00	0.15	26.139999	28.799999	0.48	19.430000	5.910000	4.660000	0.33	7.57	1.32
9698	2005-11-16 01:00:00	1.11	0.74	1.19	3.54	0.21	68.389999	134.600006	1.92	9.110000	11.330000	10.620000	1.56	9.53	1.42

1000 rows × 17 columns

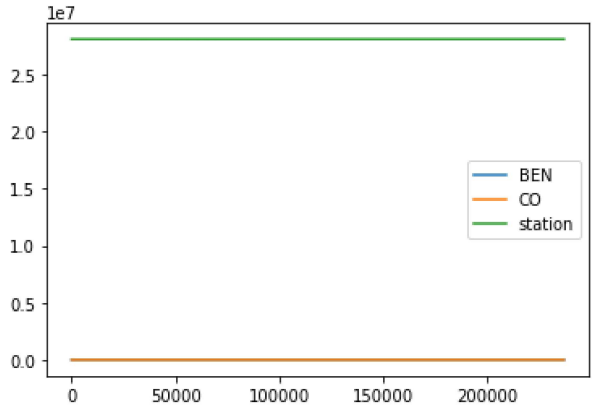
In [271]:
```
data.plot.line(subplots=True)
```

Out[271]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [272]:
```
1 data.plot.line()
2
```

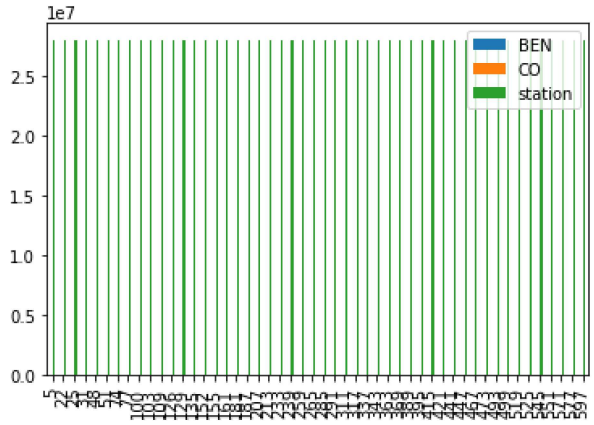
Out[272]: <AxesSubplot:>



In [273]:
```
1 b=data[0:50]
2
```

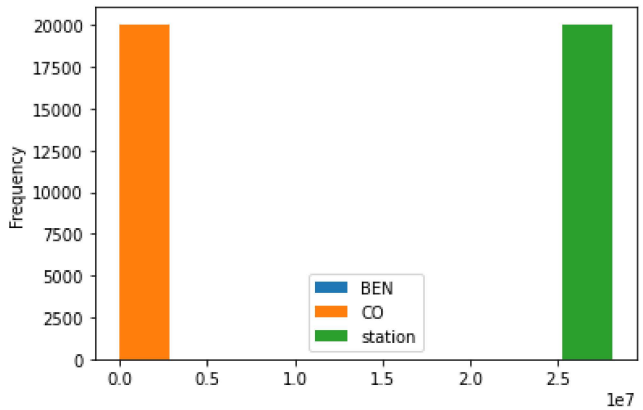
In [274]:
```
1 b.plot.bar()
```

Out[274]: <AxesSubplot:>



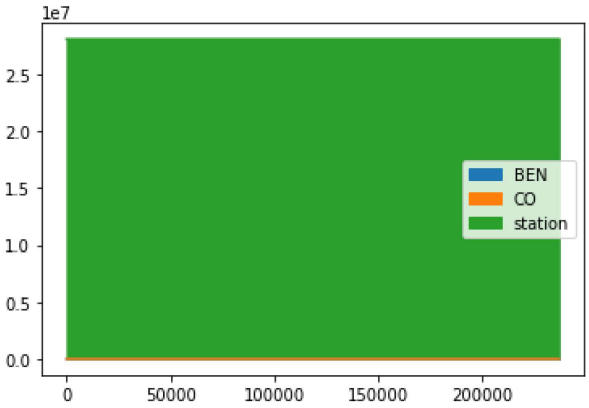
In [275]:
```
1 data.plot.hist()
2
```

Out[275]: <AxesSubplot:ylabel='Frequency'>



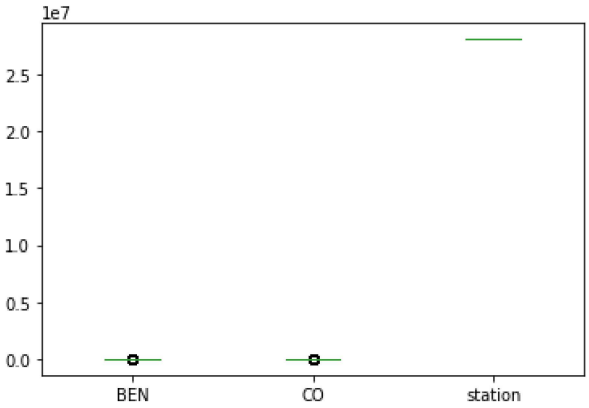
In [276]:
```
1 data.plot.area()
```

Out[276]: <AxesSubplot:>



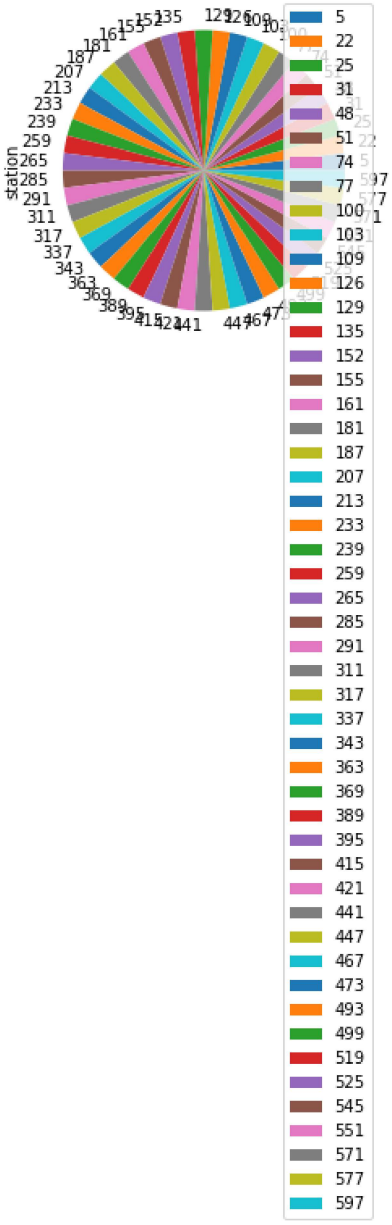
In [277]:
```
1 data.plot.box()
```

Out[277]: <AxesSubplot:>



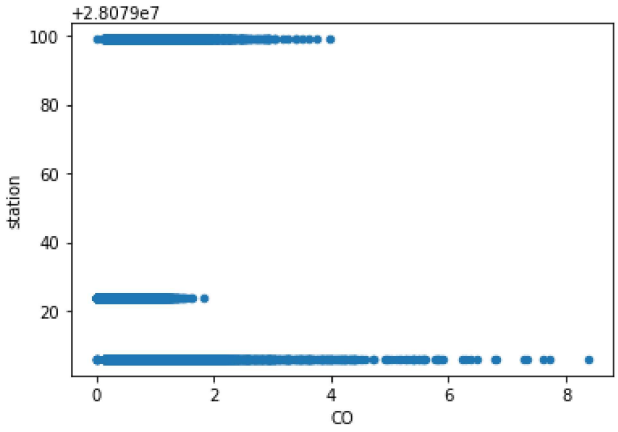
In [278]:
```
1 b.plot.pie(y='station' )
```

Out[278]: <AxesSubplot:ylabel='station'>



In [279]:
```
1 data.plot.scatter(x='CO' ,y='station')
2
```

Out[279]: <AxesSubplot:xlabel='CO', ylabel='station'>



In [280]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 5 to 9698
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     1000 non-null   object
 1   BEN      1000 non-null   float64
 2   CO       1000 non-null   float64
 3   EBE      1000 non-null   float64
 4   MXY      1000 non-null   float64
 5   NMHC     1000 non-null   float64
 6   NO_2     1000 non-null   float64
 7   NOx      1000 non-null   float64
 8   OXY      1000 non-null   float64
 9   O_3      1000 non-null   float64
 10  PM10     1000 non-null   float64
 11  PM25     1000 non-null   float64
 12  PXY      1000 non-null   float64
 13  SO_2     1000 non-null   float64
 14  TCH      1000 non-null   float64
 15  TOL      1000 non-null   float64
 16  station  1000 non-null   int64
dtypes: float64(15), int64(1), object(1)
memory usage: 140.6+ KB
```

In [281]:
```
1 df.describe()
2
```

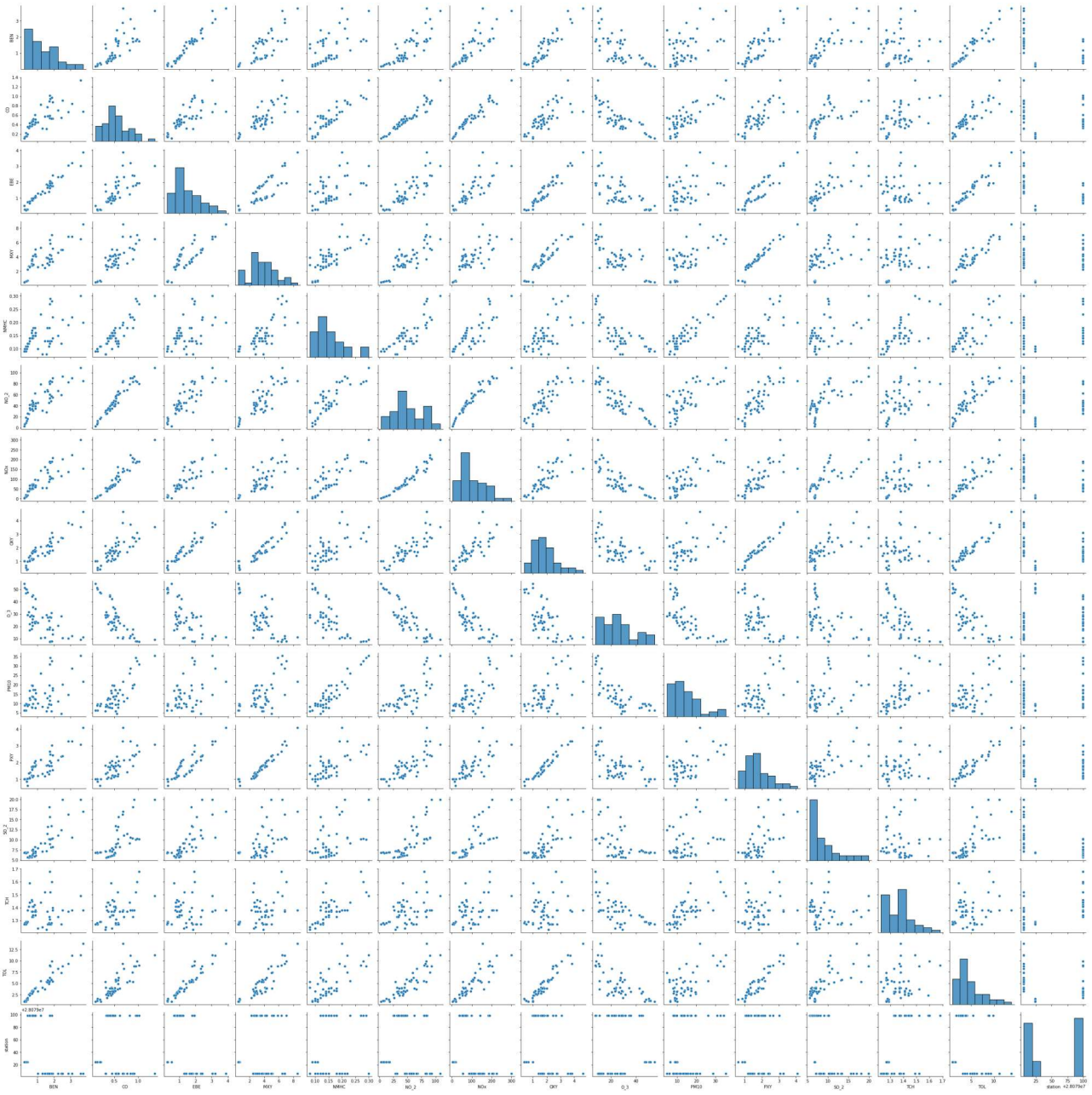
Out[281]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	1(
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	
mean	1.939710	0.779930	2.300050	5.811830	0.233450	69.242240	156.315120	2.940270	21.709360	
std	2.373978	0.636424	2.740280	6.793915	0.170952	44.166492	165.443460	3.494354	16.199118	
min	0.180000	0.060000	0.210000	0.480000	0.060000	3.110000	3.270000	0.350000	2.210000	
25%	0.660000	0.450000	0.860000	1.697500	0.150000	39.735000	55.092501	1.000000	9.277500	
50%	1.210000	0.610000	1.400000	3.705000	0.190000	63.324999	114.350002	1.780000	15.550000	
75%	2.352500	0.890000	2.680000	7.242500	0.252500	90.380001	203.075001	3.522500	29.315000	
max	26.570000	7.620000	29.870001	71.050003	1.870000	419.500000	1740.000000	38.680000	72.900002	2

In [282]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [283]:
```
sns.pairplot(df1[0:50])
```

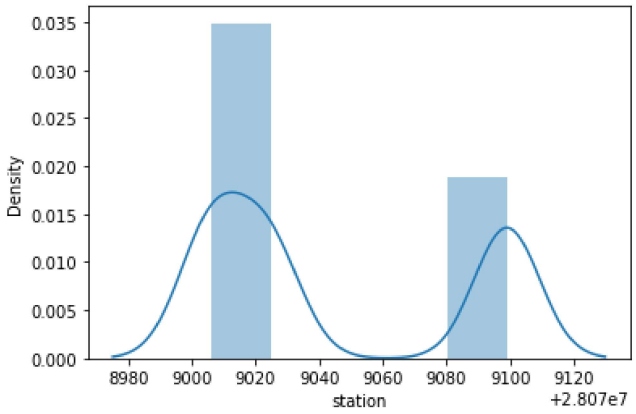
Out[283]: <seaborn.axisgrid.PairGrid at 0x21008e93df0>



In [284]:
```
1 sns.distplot(df1['station'])
2
```

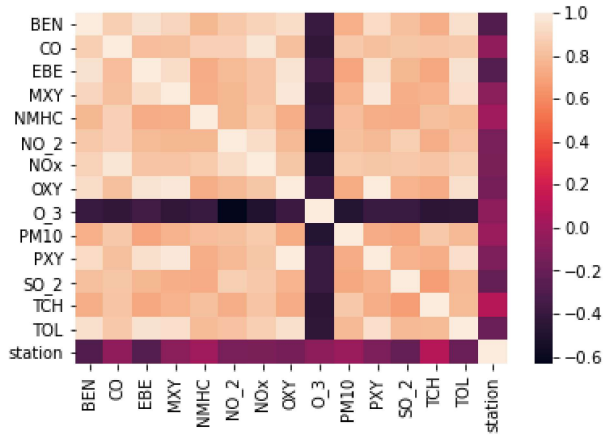
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use either `di
splot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[284]: <AxesSubplot:xlabel='station', ylabel='Density'>



In [285]:
```
1 sns.heatmap(df1.corr())
```

Out[285]: <AxesSubplot:>



In [286]:
```
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 [287]:
```
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 [288]:
```
1 from sklearn.linear_model import LinearRegression
2 lr=LinearRegression()
3 lr.fit(x_train,y_train)
4
```

Out[288]: LinearRegression()

In [289]:
```
1 lr.intercept_
```

Out[289]: 28078920.867494144

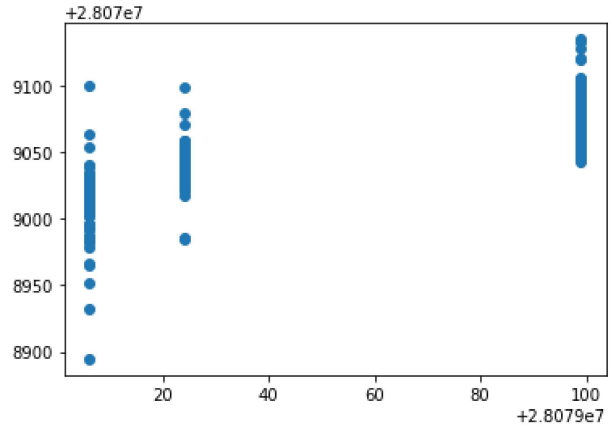
In [290]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[290]:

	Co-efficient
BEN	-21.960909
CO	71.643723
EBE	-9.064967
MXY	3.809100
NMHC	40.305719
NO_2	0.539838
NOx	-0.256901
OXY	-3.009760
O_3	0.001258
PM10	-0.325015
PXY	10.198070
SO_2	-1.954650
TCH	83.571412
TOL	-0.208193

In [291]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[291]: <matplotlib.collections.PathCollection at 0x21018fc0040>



In [292]:
```
lr.score(x_test,y_test)

```

Out[292]: 0.5429650479461272

In [293]:
```
lr.score(x_train,y_train)

```

Out[293]: 0.5819946640140975

In [294]:
```
from sklearn.linear_model import Ridge,Lasso

```

In [295]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)

```

Out[295]: Ridge(alpha=10)

In [296]:
```
rr.score(x_test,y_test)

```

Out[296]: 0.5269255342981909

In [297]:
```
rr.score(x_train,y_train)

```

Out[297]: 0.5537845022072946

In [298]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[298]: Lasso(alpha=10)

In [299]:
```
la.score(x_train,y_train)

```

Out[299]: 0.24608905949779336

In [300]:
```
la.score(x_test,y_test)

```

Out[300]: 0.17880146932264307

In [301]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)

```

Out[301]: ElasticNet()

In [302]:
```
en.coef_

```

Out[302]:
```
array([-8.68710556,  2.70231234, -7.91444656,  5.16926415,  0.39661327,
       -0.01251074,  0.05217137, -0.53896504, -0.17689445,  0.40428938,
        0.66086421, -1.2263197 ,  1.37066842, -1.15697118])
```

In [303]:
```
en.intercept_

```

Out[303]: 28079056.084456548

In [304]:
```
prediction=en.predict(x_test)

```

In [305]:
```
en.score(x_test,y_test)
```

Out[305]: 0.3492472465016302

In [306]:
```
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)))

```

```
30.013051416737337
1114.027803531482
33.37705504581676
```

In [307]:
```
from sklearn.linear_model import LogisticRegression

```

In [308]:
```
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 [309]:
```
feature_matrix.shape

```

Out[309]: (1000, 14)

In [310]:
```
target_vector.shape

```

Out[310]: (1000,)

In [311]:
```
from sklearn.preprocessing import StandardScaler
```

In [312]:
```
fs=StandardScaler().fit_transform(feature_matrix)

```

In [313]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[313]: LogisticRegression(max_iter=10000)

In [314]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]

```

In [315]:
```
prediction=logr.predict(observation)
print(prediction)

```

```
[28079006]
```

In [316]:
```
logr.classes_

```

Out[316]: array([28079006, 28079024, 28079099], dtype=int64)

In [317]:
```
logr.score(fs,target_vector)

```

Out[317]: 0.963

In [318]:
```
logr.predict_proba(observation)[0][0]

```

Out[318]: 0.9936367371114279

In [319]:
```
logr.predict_proba(observation)

```

Out[319]: array([[9.93636737e-01, 6.10138252e-28, 6.36326289e-03]])

In [320]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [321]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[321]: RandomForestClassifier()

In [322]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [323]:
```
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[323]:
```
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 [324]:
```
grid_search.best_score_
```

Out[324]:
```
0.8928571428571428
```

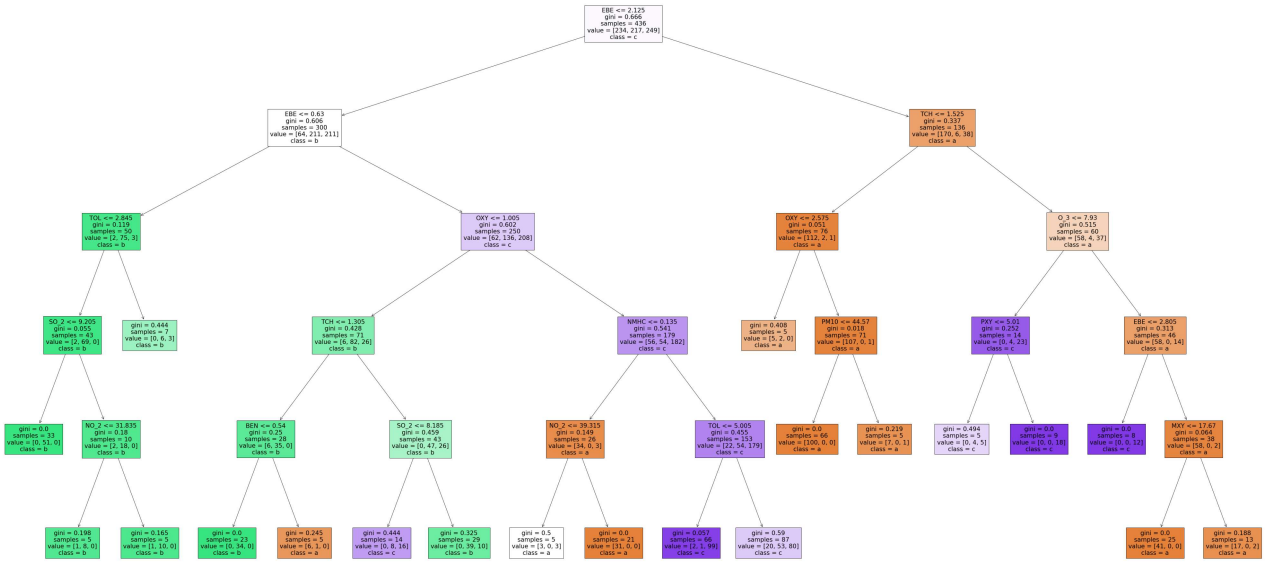
In [325]:
```
rfc_best=grid_search.best_estimator_
```

In [326]:

```
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[326]:
```
[Text(2198.1818181818185, 1993.2, 'EBE <= 2.125\ngini = 0.666\nsamples = 436\nvalue = [234, 217, 249]\nclass = c'),
 Text(1082.1818181818182, 1630.8000000000002, 'EBE <= 0.63\ngini = 0.606\nsamples = 300\nvalue = [64, 211, 211]\nclass = b'),
 Text(405.81818181818187, 1268.4, 'TOL <= 2.845\ngini = 0.119\nsamples = 50\nvalue = [2, 75, 3]\nclass = b'),
 Text(270.54545454545456, 906.0, 'SO_2 <= 9.205\ngini = 0.055\nsamples = 43\nvalue = [2, 69, 0]\nclass = b'),
 Text(135.27272727272728, 543.5999999999999, 'gini = 0.0\nsamples = 33\nvalue = [0, 51, 0]\nclass = b'),
 Text(405.81818181818187, 543.5999999999999, 'NO_2 <= 31.835\ngini = 0.18\nsamples = 10\nvalue = [2, 18, 0]\nclass = b'),
 Text(270.54545454545456, 181.19999999999982, 'gini = 0.198\nsamples = 5\nvalue = [1, 8, 0]\nclass = b'),
 Text(541.0909090909091, 181.19999999999982, 'gini = 0.165\nsamples = 5\nvalue = [1, 10, 0]\nclass = b'),
 Text(541.0909090909091, 906.0, 'gini = 0.444\nsamples = 7\nvalue = [0, 6, 3]\nclass = b'),
 Text(1758.5454545454547, 1268.4, 'OXY <= 1.005\ngini = 0.602\nsamples = 250\nvalue = [62, 136, 208]\nclass = c'),
 Text(1217.4545454545455, 906.0, 'TCH <= 1.305\ngini = 0.428\nsamples = 71\nvalue = [6, 82, 26]\nclass = b'),
 Text(946.909090909091, 543.5999999999999, 'BEN <= 0.54\ngini = 0.25\nsamples = 28\nvalue = [6, 35, 0]\nclass = b'),
 Text(811.6363636363637, 181.19999999999982, 'gini = 0.0\nsamples = 23\nvalue = [0, 34, 0]\nclass = b'),
 Text(1082.1818181818182, 181.19999999999982, 'gini = 0.245\nsamples = 5\nvalue = [6, 1, 0]\nclass = a'),
 Text(1488.0, 543.5999999999999, 'SO_2 <= 8.185\ngini = 0.459\nsamples = 43\nvalue = [0, 47, 26]\nclass = b'),
 Text(1352.7272727272727, 181.19999999999982, 'gini = 0.444\nsamples = 14\nvalue = [0, 8, 16]\nclass = c'),
 Text(1623.2727272727275, 181.19999999999982, 'gini = 0.325\nsamples = 29\nvalue = [0, 39, 10]\nclass = b'),
 Text(2299.636363636364, 906.0, 'NMHC <= 0.135\ngini = 0.541\nsamples = 179\nvalue = [56, 54, 182]\nclass = c'),
 Text(2029.0909090909092, 543.5999999999999, 'NO_2 <= 39.315\ngini = 0.149\nsamples = 26\nvalue = [34, 0, 3]\nclass = a'),
 Text(1893.818181818182, 181.19999999999982, 'gini = 0.5\nsamples = 5\nvalue = [3, 0, 3]\nclass = a'),
 Text(2164.3636363636365, 181.19999999999982, 'gini = 0.0\nsamples = 21\nvalue = [31, 0, 0]\nclass = a'),
 Text(2570.1818181818185, 543.5999999999999, 'TOL <= 5.005\ngini = 0.455\nsamples = 153\nvalue = [22, 54, 179]\nclass = c'),
 Text(2434.909090909091, 181.19999999999982, 'gini = 0.057\nsamples = 66\nvalue = [2, 1, 99]\nclass = c'),
 Text(2705.4545454545455, 181.19999999999982, 'gini = 0.59\nsamples = 87\nvalue = [20, 53, 80]\nclass = c'),
 Text(3314.1818181818185, 1630.8000000000002, 'TCH <= 1.525\ngini = 0.337\nsamples = 136\nvalue = [170, 6, 38]\nclass = a'),
 Text(2840.727272727273, 1268.4, 'OXY <= 2.575\ngini = 0.051\nsamples = 76\nvalue = [112, 2, 1]\nclass = a'),
 Text(2705.4545454545455, 906.0, 'gini = 0.408\nsamples = 5\nvalue = [5, 2, 0]\nclass = a'),
 Text(2976.0, 906.0, 'PM10 <= 44.57\ngini = 0.018\nsamples = 71\nvalue = [107, 0, 1]\nclass = a'),
 Text(2840.727272727273, 543.5999999999999, 'gini = 0.0\nsamples = 66\nvalue = [100, 0, 0]\nclass = a'),
 Text(3111.2727272727275, 543.5999999999999, 'gini = 0.219\nsamples = 5\nvalue = [7, 0, 1]\nclass = a'),
 Text(3787.636363636364, 1268.4, 'O_3 <= 7.93\ngini = 0.515\nsamples = 60\nvalue = [58, 4, 37]\nclass = a'),
 Text(3517.0909090909095, 906.0, 'PXY <= 5.01\ngini = 0.252\nsamples = 14\nvalue = [0, 4, 23]\nclass = c'),
 Text(3381.818181818182, 543.5999999999999, 'gini = 0.494\nsamples = 5\nvalue = [0, 4, 5]\nclass = c'),
 Text(3652.3636363636365, 543.5999999999999, 'gini = 0.0\nsamples = 9\nvalue = [0, 0, 18]\nclass = c'),
 Text(4058.1818181818185, 906.0, 'EBE <= 2.805\ngini = 0.313\nsamples = 46\nvalue = [58, 0, 14]\nclass = a'),
 Text(3922.909090909091, 543.5999999999999, 'gini = 0.0\nsamples = 8\nvalue = [0, 0, 12]\nclass = c'),
 Text(4193.454545454546, 543.5999999999999, 'MXY <= 17.67\ngini = 0.064\nsamples = 38\nvalue = [58, 0, 2]\nclass = a'),
 Text(4058.1818181818185, 181.19999999999982, 'gini = 0.0\nsamples = 25\nvalue = [41, 0, 0]\nclass = a'),
```

```
 Text(4328.727272727273, 181.19999999999982, 'gini = 0.188\nsamples = 13\nvalue = [17, 0, 2]\nclass =
a')]
```



Conclusion

Linear Regression=0.5819946640140975

Ridge Regression=0.5537845022072946

Lasso Regression=0.24608905949779336

ElasticNet Regression=0.3492472465016302

Logistic Regression=0.963

Random Forest=0.8928571428571428

Logistic Regression Is Suitable for this Dataset

In [ ]:
```
1
```