# Vijay(P7) 3/08/2023

In [327]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [328]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C10_air\csvs_per_year\csvs_per_year\madrid_2007.csv")
df
```

Out[328]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 | PM25 | PXY | SO_2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2007-12-01 01:00:00 | NaN | 2.86 | NaN | NaN | NaN | 282.200012 | 1054.000000 | NaN | 4.030000 | 156.199997 | 97.43 | NaN | 64.519997 |
| **1** | 2007-12-01 01:00:00 | NaN | 1.82 | NaN | NaN | NaN | 86.419998 | 354.600006 | NaN | 3.260000 | 80.809998 | NaN | NaN | 35.419998 |
| **2** | 2007-12-01 01:00:00 | NaN | 1.47 | NaN | NaN | NaN | 94.639999 | 319.000000 | NaN | 5.310000 | 53.099998 | NaN | NaN | 19.080000 |
| **3** | 2007-12-01 01:00:00 | NaN | 1.64 | NaN | NaN | NaN | 127.900002 | 476.700012 | NaN | 4.500000 | 105.300003 | NaN | NaN | 17.670000 |
| **4** | 2007-12-01 01:00:00 | 4.64 | 1.86 | 4.26 | 7.98 | 0.57 | 145.100006 | 573.900024 | 3.49 | 52.689999 | 106.500000 | 15.90 | 3.56 | 40.230000 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **225115** | 2007-03-01 00:00:00 | 0.30 | 0.45 | 1.00 | 0.30 | 0.26 | 8.690000 | 11.690000 | 1.00 | 42.209999 | 6.760000 | 5.14 | 1.00 | 7.420000 |
| **225116** | 2007-03-01 00:00:00 | NaN | 0.16 | NaN | NaN | NaN | 46.820000 | 51.480000 | NaN | 22.150000 | 5.700000 | NaN | NaN | 7.130000 |
| **225117** | 2007-03-01 00:00:00 | 0.24 | NaN | 0.20 | NaN | 0.09 | 51.259998 | 66.809998 | NaN | 18.540001 | 13.010000 | 6.95 | NaN | 8.740000 |
| **225118** | 2007-03-01 00:00:00 | 0.11 | NaN | 1.00 | NaN | 0.05 | 24.240000 | 36.930000 | NaN | NaN | 6.610000 | NaN | NaN | 9.890000 |
| **225119** | 2007-03-01 00:00:00 | 0.53 | 0.40 | 1.00 | 1.70 | 0.12 | 32.360001 | 47.860001 | 1.37 | 24.150000 | 10.260000 | 7.08 | 1.23 | 9.890000 |

225120 rows × 17 columns

In [329]:
```
df=df.dropna()
```

In [330]:
```
df.columns

```

Out[330]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [331]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25443 entries, 4 to 225119
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     25443 non-null  object 
 1   BEN      25443 non-null  float64
 2   CO       25443 non-null  float64
 3   EBE      25443 non-null  float64
 4   MXY      25443 non-null  float64
 5   NMHC     25443 non-null  float64
 6   NO_2     25443 non-null  float64
 7   NOx      25443 non-null  float64
 8   OXY      25443 non-null  float64
 9   O_3      25443 non-null  float64
 10  PM10     25443 non-null  float64
 11  PM25     25443 non-null  float64
 12  PXY      25443 non-null  float64
 13  SO_2     25443 non-null  float64
 14  TCH      25443 non-null  float64
 15  TOL      25443 non-null  float64
 16  station  25443 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 3.5+ MB
```

In [332]:
```
1 data=df[['BEN', 'CO','station']]
2 data
3
```

Out[332]:

| | BEN | CO | station |
|---|---|---|---|
| **4** | 4.64 | 1.86 | 28079006 |
| **21** | 1.98 | 0.31 | 28079024 |
| **25** | 2.82 | 1.42 | 28079099 |
| **30** | 4.65 | 1.89 | 28079006 |
| **47** | 1.97 | 0.30 | 28079024 |
| **...** | ... | ... | ... |
| **225073** | 2.12 | 0.47 | 28079006 |
| **225094** | 0.87 | 0.45 | 28079099 |
| **225098** | 0.95 | 0.41 | 28079006 |
| **225115** | 0.30 | 0.45 | 28079024 |
| **225119** | 0.53 | 0.40 | 28079099 |

25443 rows × 3 columns

In [333]:
```
df=df.head(10000)
df
```

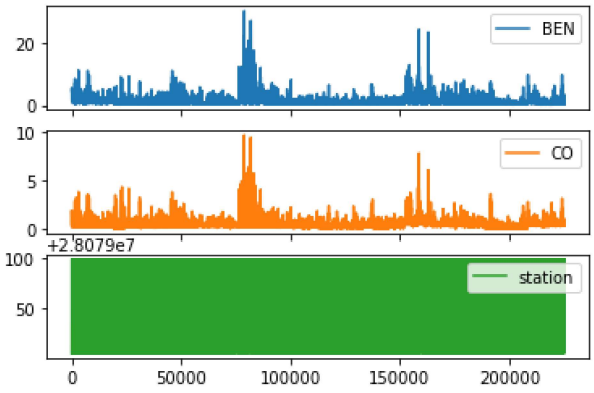
Out[333]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 | PM25 | PXY | SO_2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **4** | 2007-12-01 01:00:00 | 4.64 | 1.86 | 4.26 | 7.98 | 0.57 | 145.100006 | 573.900024 | 3.49 | 52.689999 | 106.500000 | 15.900000 | 3.56 | 40.230000 |
| **21** | 2007-12-01 01:00:00 | 1.98 | 0.31 | 2.56 | 6.06 | 0.35 | 76.059998 | 208.899994 | 1.70 | 1.000000 | 37.799999 | 25.580000 | 1.78 | 11.310000 |
| **25** | 2007-12-01 01:00:00 | 2.82 | 1.42 | 3.15 | 7.02 | 0.49 | 123.099998 | 402.399994 | 2.60 | 7.160000 | 70.809998 | 37.009998 | 2.67 | 25.670000 |
| **30** | 2007-12-01 02:00:00 | 4.65 | 1.89 | 4.41 | 8.21 | 0.65 | 151.000000 | 622.700012 | 3.55 | 58.080002 | 117.099998 | 17.049999 | 3.57 | 36.459999 |
| **47** | 2007-12-01 02:00:00 | 1.97 | 0.30 | 2.15 | 5.08 | 0.33 | 78.760002 | 189.800003 | 1.62 | 1.000000 | 34.740002 | 24.730000 | 1.59 | 10.500000 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **89387** | 2007-11-22 20:00:00 | 0.49 | 0.23 | 0.69 | 1.69 | 0.21 | 47.919998 | 52.080002 | 0.47 | 20.950001 | 8.260000 | 5.890000 | 0.41 | 7.670000 |
| **89391** | 2007-11-22 20:00:00 | 1.22 | 0.61 | 1.63 | 3.12 | 0.26 | 83.550003 | 149.300003 | 1.25 | 8.220000 | 32.700001 | 19.360001 | 1.23 | 14.150000 |
| **89396** | 2007-11-22 21:00:00 | 3.47 | 1.03 | 2.51 | 4.89 | 0.31 | 74.779999 | 192.000000 | 2.25 | 4.390000 | 34.529999 | 12.030000 | 2.18 | 18.150000 |
| **89413** | 2007-11-22 21:00:00 | 0.68 | 0.25 | 0.66 | 1.64 | 0.24 | 75.330002 | 95.309998 | 0.46 | 4.140000 | 15.440000 | 11.040000 | 0.40 | 9.050000 |
| **89417** | 2007-11-22 21:00:00 | 1.34 | 0.60 | 1.86 | 3.27 | 0.24 | 80.660004 | 141.699997 | 1.36 | 8.110000 | 18.049999 | 12.630000 | 1.29 | 15.120000 |

10000 rows × 17 columns

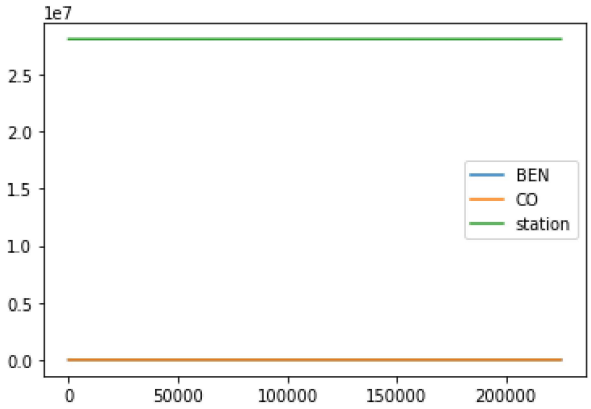
In [334]:
```
data.plot.line(subplots=True)
```

Out[334]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



In [335]:
```
1 data.plot.line()
2
```

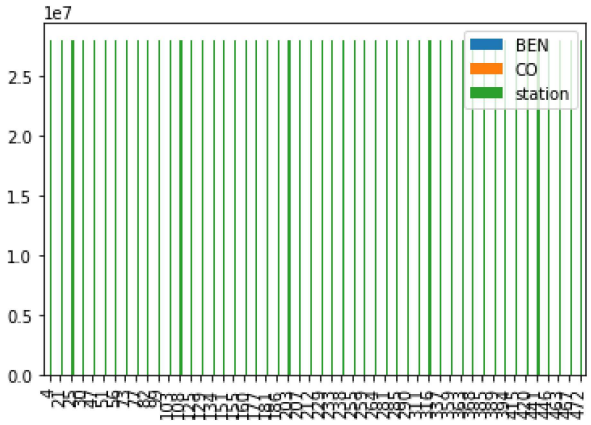
Out[335]: <AxesSubplot:>




In [336]:
```
1 b=data[0:50]
2
```

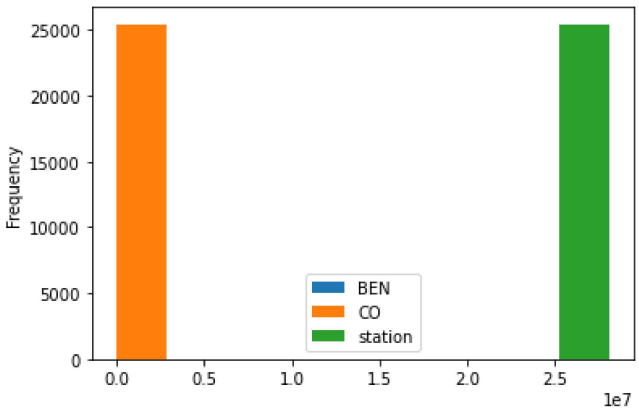
In [337]:
```
1 b.plot.bar()
```

Out[337]: <AxesSubplot:>



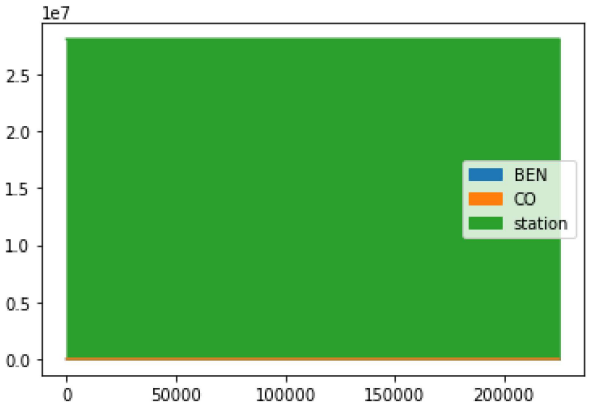

In [338]:
```
1 data.plot.hist()
2
```

Out[338]: <AxesSubplot:ylabel='Frequency'>



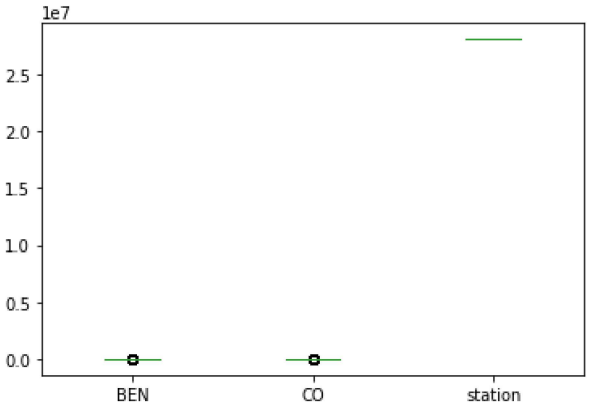
In [339]:
```
1 data.plot.area()
```

Out[339]: <AxesSubplot:>




In [340]:
```
1 data.plot.box()
```

Out[340]: <AxesSubplot:>

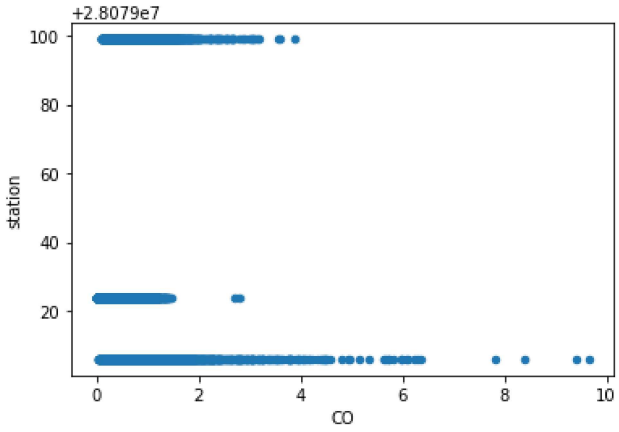


In [341]:
```
1 b.plot.pie(y='station' )
```

Out[341]: <AxesSubplot:ylabel='station'>

In [342]:
```
1 data.plot.scatter(x='CO' ,y='station')
2
```

Out[342]: <AxesSubplot:xlabel='CO', ylabel='station'>




In [343]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 4 to 89417
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     10000 non-null  object
 1   BEN      10000 non-null  float64
 2   CO       10000 non-null  float64
 3   EBE      10000 non-null  float64
 4   MXY      10000 non-null  float64
 5   NMHC     10000 non-null  float64
 6   NO_2     10000 non-null  float64
 7   NOx      10000 non-null  float64
 8   OXY      10000 non-null  float64
 9   O_3      10000 non-null  float64
 10  PM10     10000 non-null  float64
 11  PM25     10000 non-null  float64
 12  PXY      10000 non-null  float64
 13  SO_2     10000 non-null  float64
 14  TCH      10000 non-null  float64
 15  TOL      10000 non-null  float64
 16  station  10000 non-null  int64
dtypes: float64(15), int64(1), object(1)
memory usage: 1.4+ MB
```

In [344]:
```
1 df.describe()
2
```

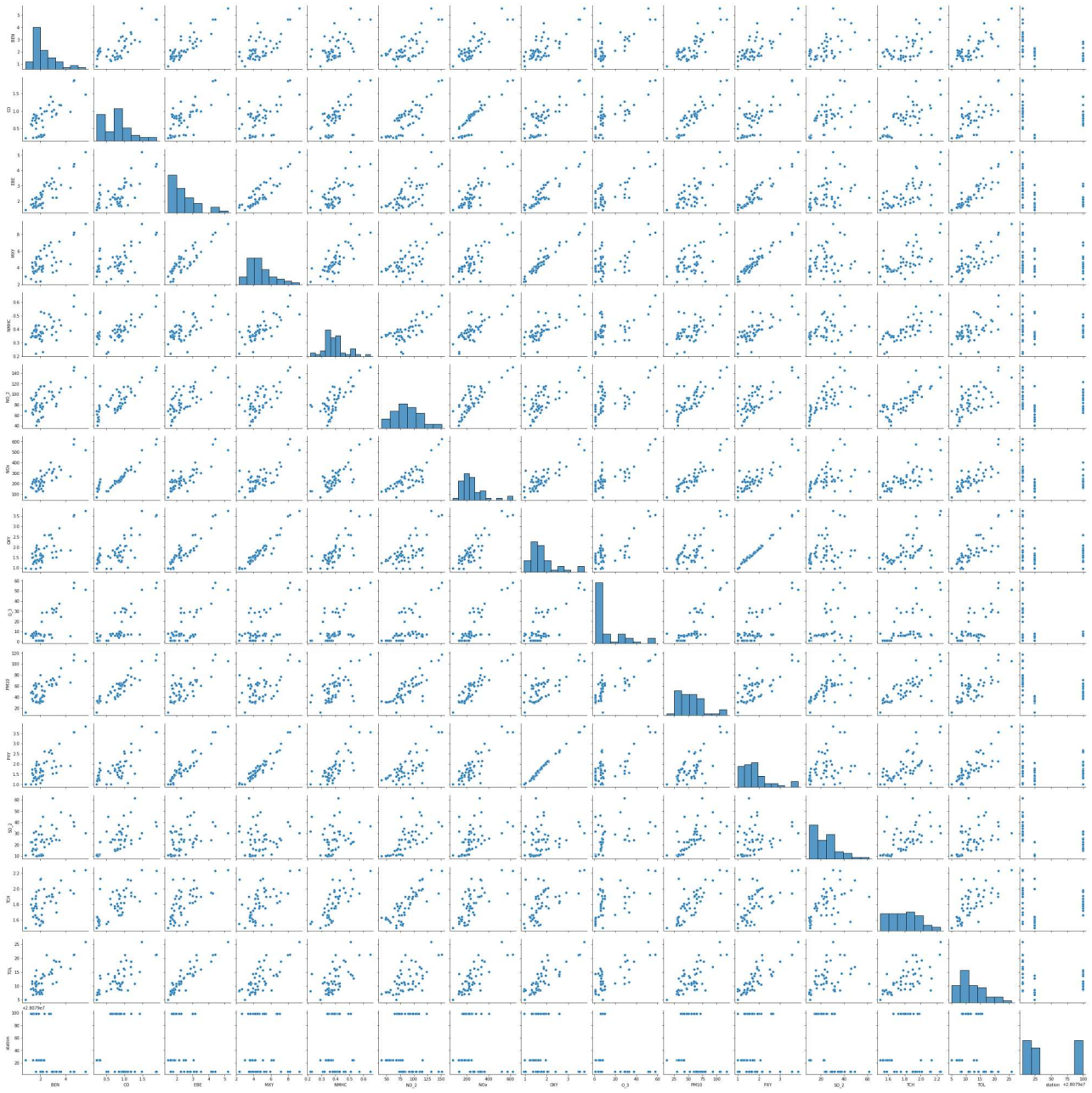
Out[344]:

| | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000. |
| **mean** | 1.337737 | 0.520786 | 1.508041 | 2.654044 | 0.315455 | 65.391101 | 129.601638 | 1.303739 | 29. |
| **std** | 1.465736 | 0.491611 | 1.292714 | 2.677514 | 0.162820 | 38.943317 | 129.665140 | 1.129495 | 24. |
| **min** | 0.150000 | 0.010000 | 0.190000 | 0.170000 | 0.000000 | 2.510000 | 2.620000 | 0.110000 | 0. |
| **25%** | 0.560000 | 0.250000 | 0.790000 | 1.000000 | 0.210000 | 37.919998 | 49.892501 | 0.730000 | 7. |
| **50%** | 0.930000 | 0.375000 | 1.140000 | 1.850000 | 0.280000 | 60.525000 | 95.309998 | 1.000000 | 22. |
| **75%** | 1.620000 | 0.630000 | 1.830000 | 3.362500 | 0.400000 | 86.162502 | 168.325001 | 1.510000 | 44. |
| **max** | 30.139999 | 9.660000 | 23.379999 | 46.730000 | 2.570000 | 430.299988 | 1847.000000 | 32.540001 | 140. |

In [345]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [346]:
```
sns.pairplot(df1[0:50])
```

Out[346]: <seaborn.axisgrid.PairGrid at 0x21019f38280>

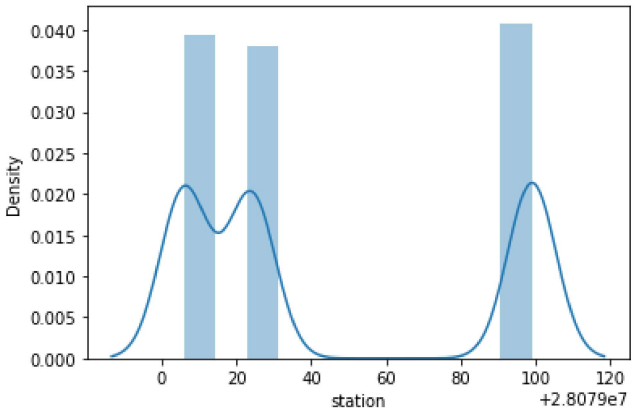


In [347]:
```
sns.distplot(df1['station'])

```

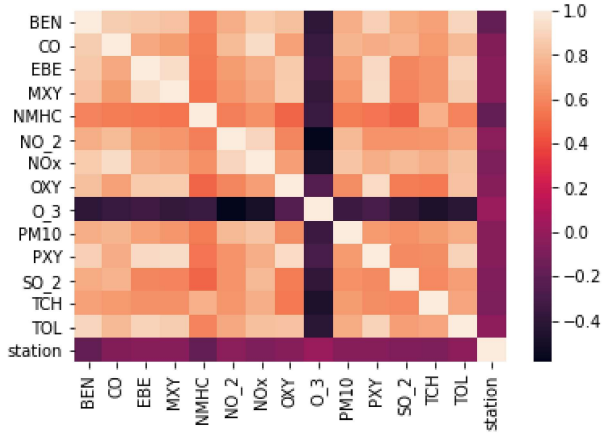
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[347]: <AxesSubplot:xlabel='station', ylabel='Density'>




In [348]:
```
sns.heatmap(df1.corr())
```

Out[348]: <AxesSubplot:>




In [349]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
 'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']

```

In [350]:
```
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 [351]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)

```

Out[351]: LinearRegression()

In [352]:
```
lr.intercept_
```

Out[352]: 28079024.11153279

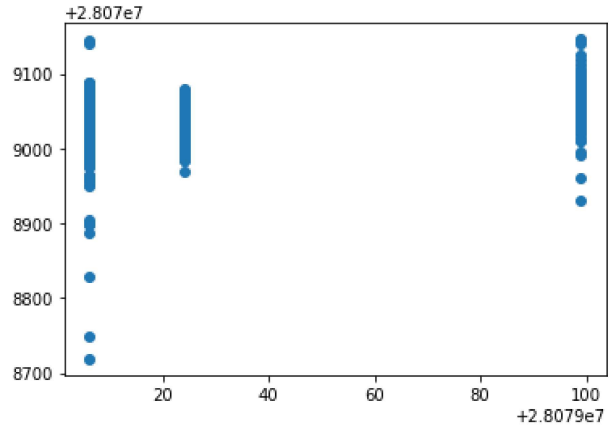
In [353]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[353]:

| | Co-efficient |
|---|---|
| **BEN** | -39.715434 |
| **CO** | 38.232571 |
| **EBE** | 1.649810 |
| **MXY** | -1.727348 |
| **NMHC** | -67.799240 |
| **NO_2** | 0.119086 |
| **NOx** | -0.048293 |
| **OXY** | 0.569914 |
| **O_3** | -0.094167 |
| **PM10** | 0.010241 |
| **PXY** | 14.261016 |
| **SO_2** | 0.261422 |
| **TCH** | 21.922055 |
| **TOL** | 3.920577 |

In [354]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[354]: <matplotlib.collections.PathCollection at 0x2102b1257f0>




In [355]:
```
lr.score(x_test,y_test)

```

Out[355]: 0.18832072724652216

In [356]:
```
lr.score(x_train,y_train)

```

Out[356]: 0.24039322600259538

In [357]:
```
from sklearn.linear_model import Ridge,Lasso

```

In [358]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)

```

Out[358]: Ridge(alpha=10)

In [359]:
```
rr.score(x_test,y_test)
```

Out[359]: 0.18956540856303727

In [360]:
```
rr.score(x_train,y_train)
```

Out[360]: 0.23975318959885694

In [361]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[361]: Lasso(alpha=10)

In [362]:
```
la.score(x_train,y_train)
```

Out[362]: 0.013157253655736056

In [363]:
```
la.score(x_test,y_test)
```

Out[363]: 0.008479502920505033

In [364]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[364]: ElasticNet()

In [365]:
```
en.coef_
```

Out[365]:
```
array([-9.58857946e+00,  0.00000000e+00, -0.00000000e+00, -1.36724997e-01,
       -8.21327427e-01,  1.48081172e-01, -5.05636798e-02,  0.00000000e+00,
        2.06137262e-02, -2.32329912e-03,  0.00000000e+00,  2.98477369e-02,
       -0.00000000e+00,  1.90310519e+00])
```

In [366]:
```
en.intercept_
```

Out[366]: 28079041.88418573

In [367]:
```
prediction=en.predict(x_test)
```

In [368]:
```
en.score(x_test,y_test)
```

Out[368]: 0.08889326711786572

In [369]:
```
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)))
```

```
36.24640511702622
1512.82603329351
38.89506438217464
```

In [370]:
```
from sklearn.linear_model import LogisticRegression
```

In [371]:
```
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 [372]:
```
feature_matrix.shape

```

Out[372]: (10000, 14)

In [373]:
```
target_vector.shape

```

Out[373]: (10000,)

In [374]:
```
from sklearn.preprocessing import StandardScaler
```

In [375]:
```
fs=StandardScaler().fit_transform(feature_matrix)

```

In [376]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[376]: LogisticRegression(max_iter=10000)

In [377]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]

```

In [378]:
```
prediction=logr.predict(observation)
print(prediction)

```

```
[28079099]
```

In [379]:
```
logr.classes_

```

Out[379]: array([28079006, 28079024, 28079099], dtype=int64)

In [380]:
```
logr.score(fs,target_vector)

```

Out[380]: 0.8101

In [381]:
```
logr.predict_proba(observation)[0][0]

```

Out[381]: 1.2204737088913592e-28

In [382]:
```
logr.predict_proba(observation)

```

Out[382]: array([[1.22047371e-28, 4.19009794e-17, 1.00000000e+00]])

In [383]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [384]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[384]: RandomForestClassifier()

In [385]:
```
parameters={'max_depth':[1,2,3,4,5],
 'min_samples_leaf':[5,10,15,20,25],
 'n_estimators':[10,20,30,40,50]
}
```

In [386]:
```
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[386]:
```
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 [387]:
```
grid_search.best_score_
```

Out[387]:
```
0.8162857142857143
```

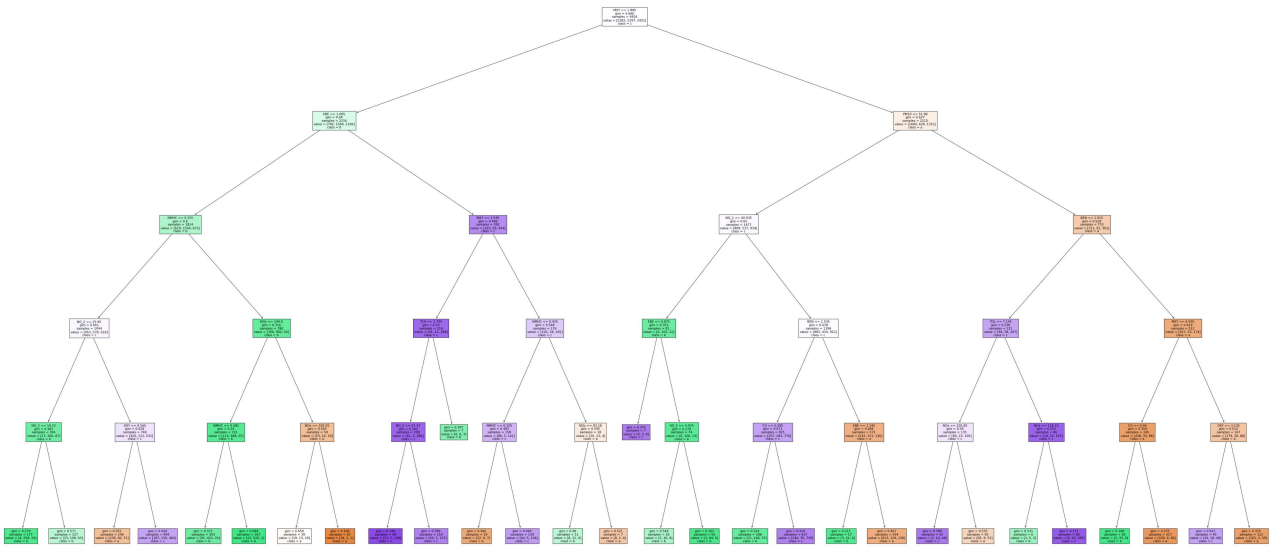
In [388]:
```
rfc_best=grid_search.best_estimator_
```

In [389]:

```
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[389]:
```
[Text(2192.142857142857, 1993.2, 'MXY <= 1.885\ngini = 0.666\nsamples = 4426\nvalue = [2382, 2197, 2421]\nclass = c'),
 Text(1175.7857142857142, 1630.8000000000002, 'EBE <= 1.005\ngini = 0.64\nsamples = 2216\nvalue = [782, 1569, 1106]\nclass = b'),
 Text(637.7142857142857, 1268.4, 'NMHC <= 0.255\ngini = 0.6\nsamples = 1824\nvalue = [619, 1540, 672]\nclass = b'),
 Text(318.85714285714283, 906.0, 'NO_2 <= 25.95\ngini = 0.661\nsamples = 1044\nvalue = [453, 578, 618]\nclass = c'),
 Text(159.42857142857142, 543.5999999999999, 'NO_2 <= 19.22\ngini = 0.383\nsamples = 304\nvalue = [27, 366, 87]\nclass = b'),
 Text(79.71428571428571, 181.19999999999982, 'gini = 0.179\nsamples = 177\nvalue = [4, 258, 24]\nclass = b'),
 Text(239.1428571428571, 181.19999999999982, 'gini = 0.571\nsamples = 127\nvalue = [23, 108, 63]\nclass = b'),
 Text(478.2857142857142, 543.5999999999999, 'OXY <= 0.565\ngini = 0.628\nsamples = 740\nvalue = [426, 212, 531]\nclass = c'),
 Text(398.57142857142856, 181.19999999999982, 'gini = 0.552\nsamples = 246\nvalue = [239, 82, 71]\nclass = a'),
 Text(558.0, 181.19999999999982, 'gini = 0.564\nsamples = 494\nvalue = [187, 130, 460]\nclass = c'),
 Text(956.5714285714284, 906.0, 'NOx <= 144.0\ngini = 0.316\nsamples = 780\nvalue = [166, 962, 54]\nclass = b'),
 Text(797.1428571428571, 543.5999999999999, 'NMHC <= 0.385\ngini = 0.24\nsamples = 721\nvalue = [113, 948, 35]\nclass = b'),
 Text(717.4285714285713, 181.19999999999982, 'gini = 0.371\nsamples = 354\nvalue = [91, 422, 33]\nclass = b'),
 Text(876.8571428571428, 181.19999999999982, 'gini = 0.084\nsamples = 367\nvalue = [22, 526, 2]\nclass = b'),
 Text(1116.0, 543.5999999999999, 'NOx <= 192.55\ngini = 0.545\nsamples = 59\nvalue = [53, 14, 19]\nclass = a'),
 Text(1036.2857142857142, 181.19999999999982, 'gini = 0.658\nsamples = 39\nvalue = [19, 13, 18]\nclass = a'),
 Text(1195.7142857142856, 181.19999999999982, 'gini = 0.106\nsamples = 20\nvalue = [34, 1, 1]\nclass = a'),
 Text(1713.8571428571427, 1268.4, 'MXY <= 1.545\ngini = 0.449\nsamples = 392\nvalue = [163, 29, 434]\nclass = c'),
 Text(1514.5714285714284, 906.0, 'TCH <= 1.705\ngini = 0.33\nsamples = 216\nvalue = [61, 11, 289]\nclass = c'),
 Text(1434.8571428571427, 543.5999999999999, 'NO_2 <= 51.17\ngini = 0.302\nsamples = 209\nvalue = [61, 3, 286]\nclass = c'),
 Text(1355.142857142857, 181.19999999999982, 'gini = 0.149\nsamples = 99\nvalue = [11, 2, 149]\nclass = c'),
 Text(1514.5714285714284, 181.19999999999982, 'gini = 0.398\nsamples = 110\nvalue = [50, 1, 137]\nclass = c'),
 Text(1594.2857142857142, 543.5999999999999, 'gini = 0.397\nsamples = 7\nvalue = [0, 8, 3]\nclass = b'),
 Text(1913.1428571428569, 906.0, 'NMHC <= 0.425\ngini = 0.548\nsamples = 176\nvalue = [102, 18, 145]\nclass = c'),
 Text(1753.7142857142856, 543.5999999999999, 'NMHC <= 0.155\ngini = 0.493\nsamples = 158\nvalue = [86, 5, 141]\nclass = c'),
 Text(1673.9999999999998, 181.19999999999982, 'gini = 0.366\nsamples = 19\nvalue = [22, 0, 7]\nclass = a'),
 Text(1833.4285714285713, 181.19999999999982, 'gini = 0.464\nsamples = 139\nvalue = [64, 5, 134]\nclass = c'),
 Text(2072.5714285714284, 543.5999999999999, 'NOx <= 93.18\ngini = 0.595\nsamples = 18\nvalue = [16, 13, 4]\nclass = a'),
 Text(1992.8571428571427, 181.19999999999982, 'gini = 0.48\nsamples = 11\nvalue = [8, 12, 0]\nclass = b'),
 Text(2152.285714285714, 181.19999999999982, 'gini = 0.521\nsamples = 7\nvalue = [8, 1, 4]\nclass = a'),
 Text(3208.4999999999995, 1630.8000000000002, 'PM10 <= 51.98\ngini = 0.627\nsamples = 2210\nvalue = [1600, 628, 1315]\nclass = a'),
 Text(2590.7142857142853, 1268.4, 'NO_2 <= 40.035\ngini = 0.65\nsamples = 1477\nvalue = [889, 537, 934]\nclass = c'),
 Text(2311.7142857142853, 906.0, 'EBE <= 0.875\ngini = 0.351\nsamples = 81\nvalue = [6, 103, 22]\nclass = b'),
 Text(2232.0, 543.5999999999999, 'gini = 0.375\nsamples = 7\nvalue = [0, 3, 9]\nclass = c'),
 Text(2391.428571428571, 543.5999999999999, 'SO_2 <= 5.975\ngini = 0.279\nsamples = 74\nvalue = [6, 100, 13]\nclass = b'),
 Text(2311.7142857142853, 181.19999999999982, 'gini = 0.549\nsamples = 18\nvalue = [3, 16, 8]\nclass = b'),
 Text(2471.142857142857, 181.19999999999982, 'gini = 0.162\nsamples = 56\nvalue = [3, 84, 5]\nclass =
```

```
b'),
 Text(2869.7142857142853, 906.0, 'BEN <= 1.535\ngini = 0.638\nsamples = 1396\nvalue = [883, 434, 912]\n
class = c'),
 Text(2710.285714285714, 543.5999999999999, 'CO <= 0.295\ngini = 0.571\nsamples = 825\nvalue = [267, 28
3, 776]\nclass = c'),
 Text(2630.5714285714284, 181.19999999999982, 'gini = 0.324\nsamples = 188\nvalue = [21, 248, 37]\nclas
s = b'),
 Text(2790.0, 181.19999999999982, 'gini = 0.416\nsamples = 637\nvalue = [246, 35, 739]\nclass = c'),
 Text(3029.142857142857, 543.5999999999999, 'EBE <= 1.145\ngini = 0.484\nsamples = 571\nvalue = [616, 1
51, 136]\nclass = a'),
 Text(2949.428571428571, 181.19999999999982, 'gini = 0.211\nsamples = 17\nvalue = [3, 22, 0]\nclass =
b'),
 Text(3108.8571428571427, 181.19999999999982, 'gini = 0.467\nsamples = 554\nvalue = [613, 129, 136]\ncl
ass = a'),
 Text(3826.2857142857138, 1268.4, 'BEN <= 1.915\ngini = 0.529\nsamples = 733\nvalue = [711, 91, 381]\nc
lass = a'),
 Text(3507.428571428571, 906.0, 'TOL <= 7.145\ngini = 0.538\nsamples = 221\nvalue = [94, 38, 207]\nclas
s = c'),
 Text(3347.9999999999995, 543.5999999999999, 'NOx <= 105.85\ngini = 0.59\nsamples = 135\nvalue = [84, 2
3, 100]\nclass = c'),
 Text(3268.285714285714, 181.19999999999982, 'gini = 0.389\nsamples = 42\nvalue = [2, 14, 48]\nclass =
c'),
 Text(3427.7142857142853, 181.19999999999982, 'gini = 0.535\nsamples = 93\nvalue = [82, 9, 52]\nclass =
a'),
 Text(3666.8571428571427, 543.5999999999999, 'NOx <= 116.15\ngini = 0.324\nsamples = 86\nvalue = [10, 1
5, 107]\nclass = c'),
 Text(3587.142857142857, 181.19999999999982, 'gini = 0.531\nsamples = 6\nvalue = [1, 5, 2]\nclass =
b'),
 Text(3746.5714285714284, 181.19999999999982, 'gini = 0.271\nsamples = 80\nvalue = [9, 10, 105]\nclass
= c'),
 Text(4145.142857142857, 906.0, 'MXY <= 6.935\ngini = 0.419\nsamples = 512\nvalue = [617, 53, 174]\ncla
ss = a'),
 Text(3985.7142857142853, 543.5999999999999, 'CO <= 0.46\ngini = 0.354\nsamples = 345\nvalue = [438, 3
3, 86]\nclass = a'),
 Text(3905.9999999999995, 181.19999999999982, 'gini = 0.108\nsamples = 28\nvalue = [2, 33, 0]\nclass =
b'),
 Text(4065.428571428571, 181.19999999999982, 'gini = 0.275\nsamples = 317\nvalue = [436, 0, 86]\nclass
= a'),
 Text(4304.571428571428, 543.5999999999999, 'OXY <= 3.135\ngini = 0.512\nsamples = 167\nvalue = [179, 2
0, 88]\nclass = a'),
 Text(4224.857142857142, 181.19999999999982, 'gini = 0.567\nsamples = 45\nvalue = [16, 18, 49]\nclass =
c'),
 Text(4384.285714285714, 181.19999999999982, 'gini = 0.325\nsamples = 122\nvalue = [163, 2, 39]\nclass
= a')]
```



# Conclusion

Linear Regression=0.24039322600259538

Ridge Regression=0.23975318959885694

Lasso Regression=0.013157253655736056

ElasticNet Regression=0.08889326711786572

Logistic Regression=0.8101

Random Forest=0.8162857142857143

Random Forest is Suitable for this Dataset