Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing Datasets

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\madrid_2004.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	
0	01-08-2004 01:00	NaN	0.66	NaN	NaN	NaN	89.550003	118.900002	NaN	40.020000	39.990002	25.8
1	01-08-2004 01:00	2.66	0.54	2.99	6.08	0.18	51.799999	53.860001	3.28	51.689999	22.950001	
2	01-08-2004 01:00	NaN	1.02	NaN	NaN	NaN	93.389999	138.600006	NaN	20.860001	49.480000	
3	01-08-2004 01:00	NaN	0.53	NaN	NaN	NaN	87.290001	105.000000	NaN	36.730000	31.070000	
4	01-08-2004 01:00	NaN	0.17	NaN	NaN	NaN	34.910000	35.349998	NaN	86.269997	54.080002	
...	...	...	...	...	...	...	...	...	...	...	...	...
245491	01-06-2004 00:00	0.75	0.21	0.85	1.55	0.07	59.580002	64.389999	0.66	33.029999	30.900000	14.8
245492	01-06-2004 00:00	2.49	0.75	2.44	4.57	NaN	97.139999	146.899994	2.34	7.740000	37.689999	
245493	01-06-2004 00:00	NaN	NaN	NaN	NaN	0.13	102.699997	132.600006	NaN	17.809999	22.840000	12.0

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	
245494	01-06-2004 00:00	NaN	NaN	NaN	NaN	0.09	82.599998	102.599998	NaN	NaN	45.630001	
245495	01-06-2004 00:00	3.01	0.67	2.78	5.12	0.20	92.550003	141.000000	2.60	11.460000	24.389999	17.9

245496 rows × 17 columns

Data Cleaning and Data Preprocessing

In [3]:
```
df=df.dropna()
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19397 entries, 5 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     19397 non-null  object 
 1   BEN      19397 non-null  float64
 2   CO       19397 non-null  float64
 3   EBE      19397 non-null  float64
 4   MXY      19397 non-null  float64
 5   NMHC     19397 non-null  float64
 6   NO_2     19397 non-null  float64
 7   NOx      19397 non-null  float64
 8   OXY      19397 non-null  float64
 9   O_3      19397 non-null  float64
 10  PM10     19397 non-null  float64
 11  PM25     19397 non-null  float64
 12  PXY      19397 non-null  float64
 13  SO_2     19397 non-null  float64
 14  TCH      19397 non-null  float64
 15  TOL      19397 non-null  float64
 16  station  19397 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 2.7+ MB
```

In [6]:
```
data=df[['CO' ,'station']]
data
```

Out[6]:

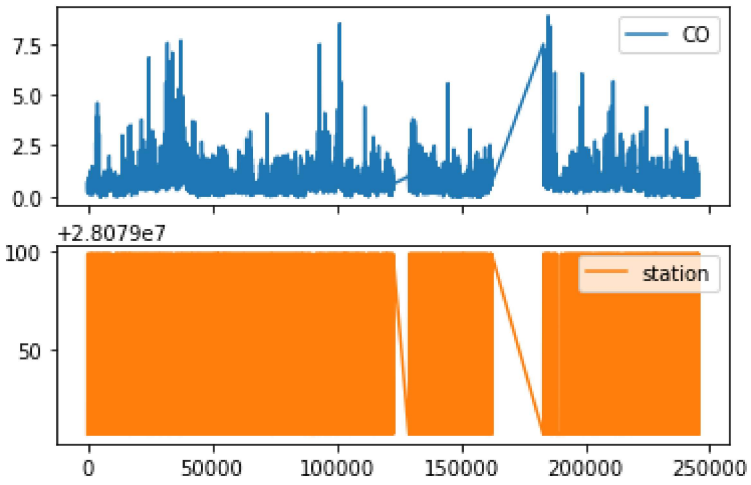
	CO	station
5	0.63	28079006
22	0.36	28079024
26	0.46	28079099
32	0.67	28079006
49	0.30	28079024
...	...	...
245463	0.08	28079024
245467	0.67	28079099
245473	1.12	28079006
245491	0.21	28079024
245495	0.67	28079099

19397 rows × 2 columns

Line chart

In [7]:
```
data.plot.line(subplots=True)
```

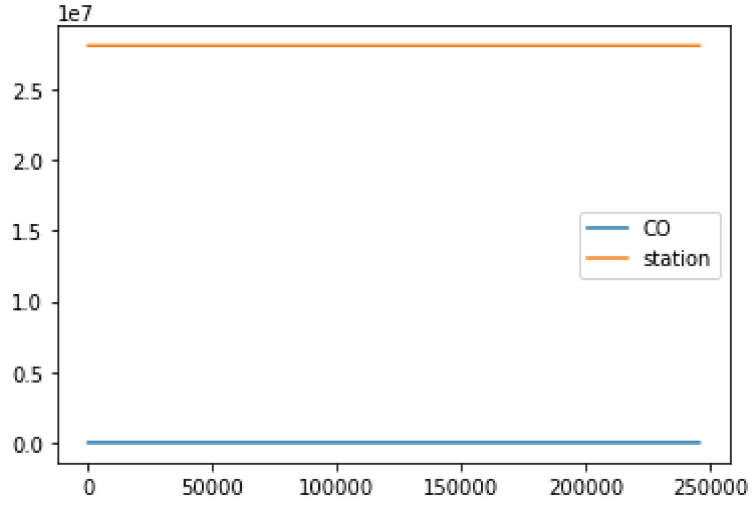
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

In [8]:
```
data.plot.line()
```

Out[8]:
```
<AxesSubplot:>
```

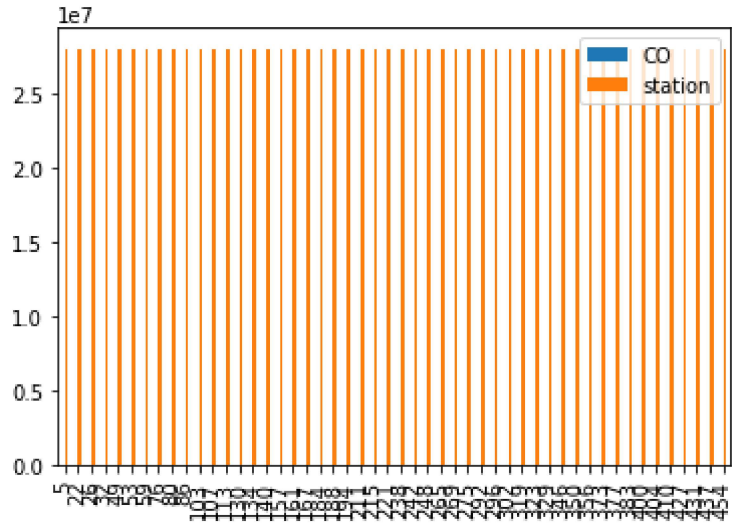


Bar chart

In [9]:
```
b=data[0:50]
```

In [10]:
```
b.plot.bar()
```

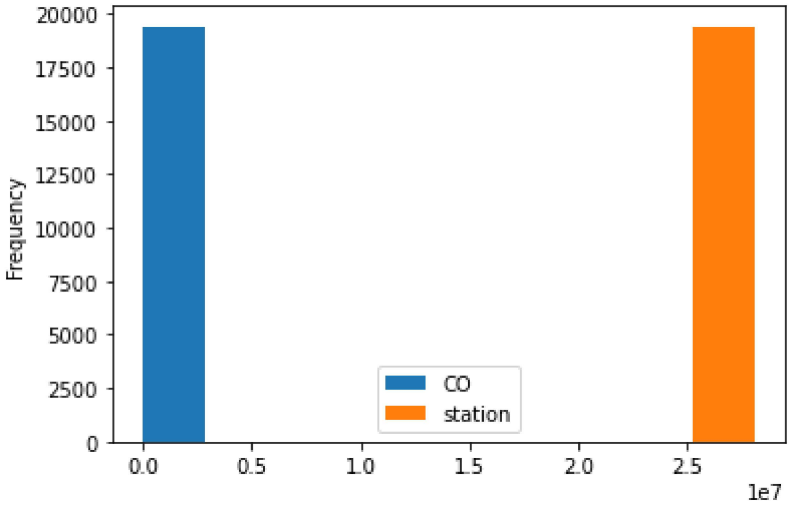
Out[10]: <AxesSubplot:>



Histogram

In [11]:
```
data.plot.hist()
```

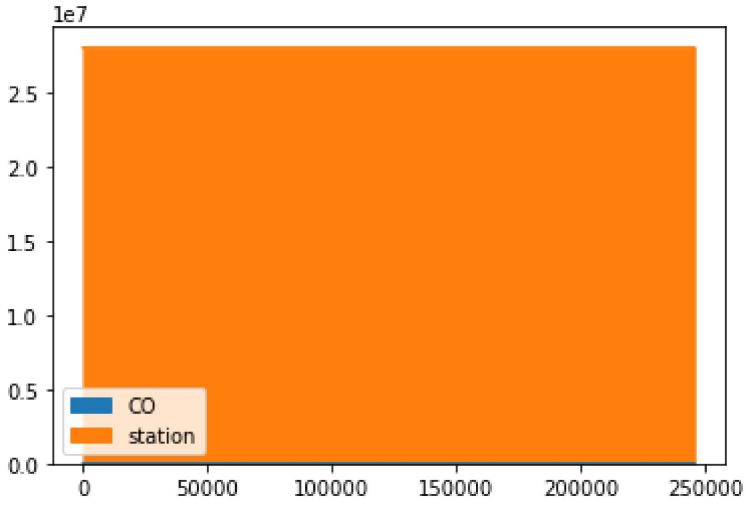
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

In [12]:
```
data.plot.area()
```

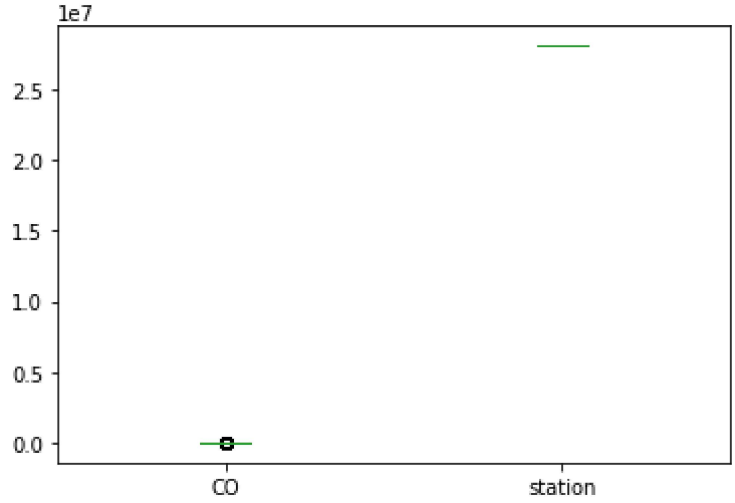
Out[12]: <AxesSubplot:>



Box chart

In [13]:
```
data.plot.box()
```

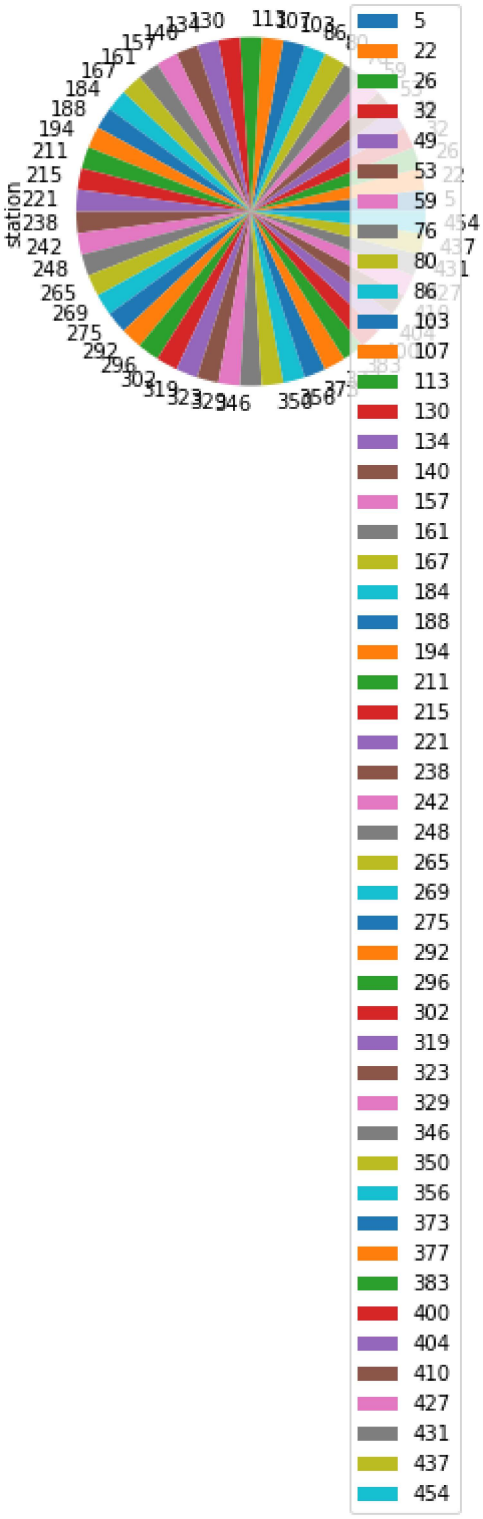
Out[13]: <AxesSubplot:>



Pie chart

In [14]:
```
b.plot.pie(y='station' )
```

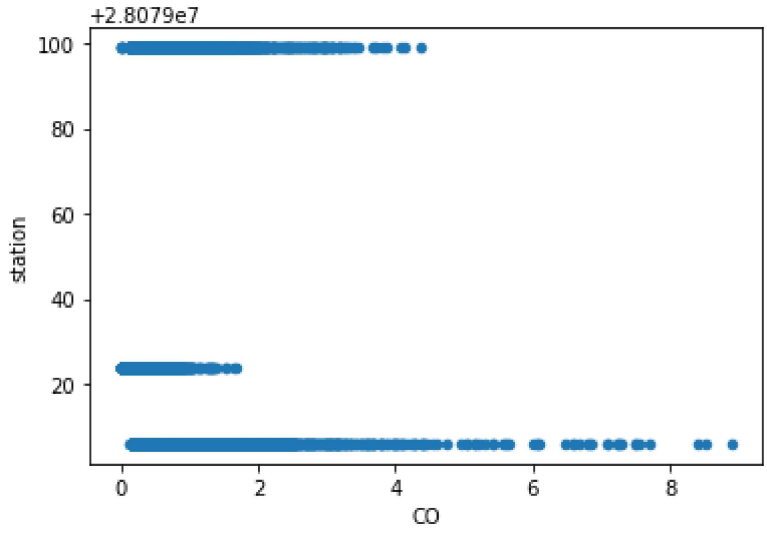
Out[14]:
```
<AxesSubplot:ylabel='station'>
```



Scatter chart

In [15]:
```
data.plot.scatter(x='CO' ,y='station')
```

Out[15]:
```
<AxesSubplot:xlabel='CO', ylabel='station'>
```



In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19397 entries, 5 to 245495
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     19397 non-null  object 
 1   BEN      19397 non-null  float64
 2   CO       19397 non-null  float64
 3   EBE      19397 non-null  float64
 4   MXY      19397 non-null  float64
 5   NMHC     19397 non-null  float64
 6   NO_2     19397 non-null  float64
 7   NOx      19397 non-null  float64
 8   OXY      19397 non-null  float64
 9   O_3      19397 non-null  float64
 10  PM10     19397 non-null  float64
 11  PM25     19397 non-null  float64
 12  PXY      19397 non-null  float64
 13  SO_2     19397 non-null  float64
 14  TCH      19397 non-null  float64
 15  TOL      19397 non-null  float64
 16  station  19397 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 2.7+ MB
```

In [17]:
```
df.describe()
```

Out[17]:

	BEN	CO	EBE	MXY	NMHC	NO_2	NO
count	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000	19397.000000	19397.00000
mean	2.250781	0.675347	2.775913	5.424809	0.151024	62.887023	128.55402
std	2.184724	0.591026	2.729622	5.554358	0.158603	37.952255	127.91241
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.090000	2.41000
25%	0.870000	0.320000	1.020000	1.780000	0.060000	35.150002	45.20999
50%	1.620000	0.520000	1.970000	3.800000	0.110000	58.310001	93.22000

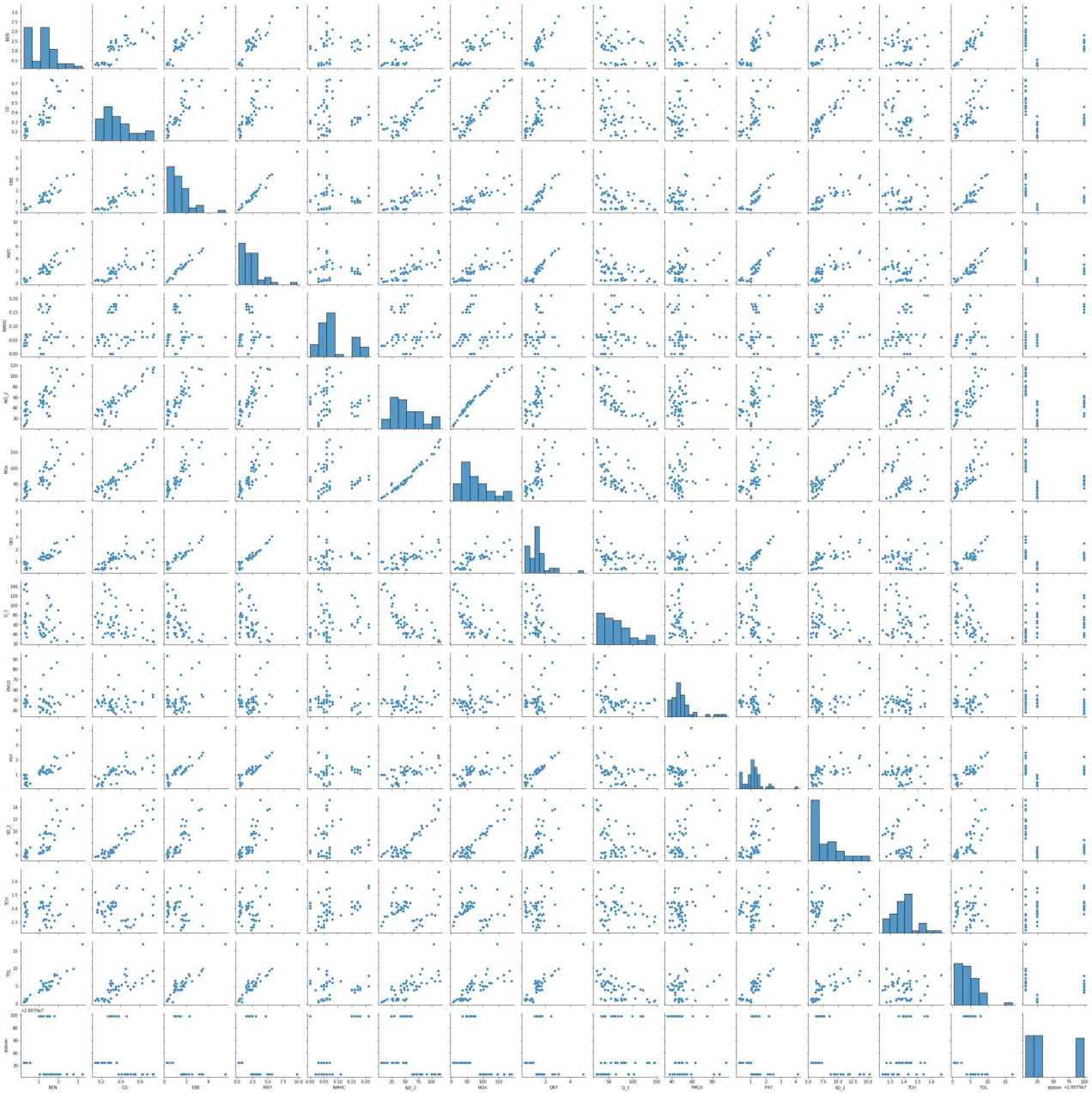
	BEN	CO	EBE	MXY	NMHC	NO_2	NO
75%	2.910000	0.860000	3.580000	7.260000	0.200000	85.730003	174.30000
max	34.180000	8.900000	41.880001	91.599998	4.810000	355.100006	1700.00000

In [18]:
```
df1=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

EDA AND VISUALIZATION

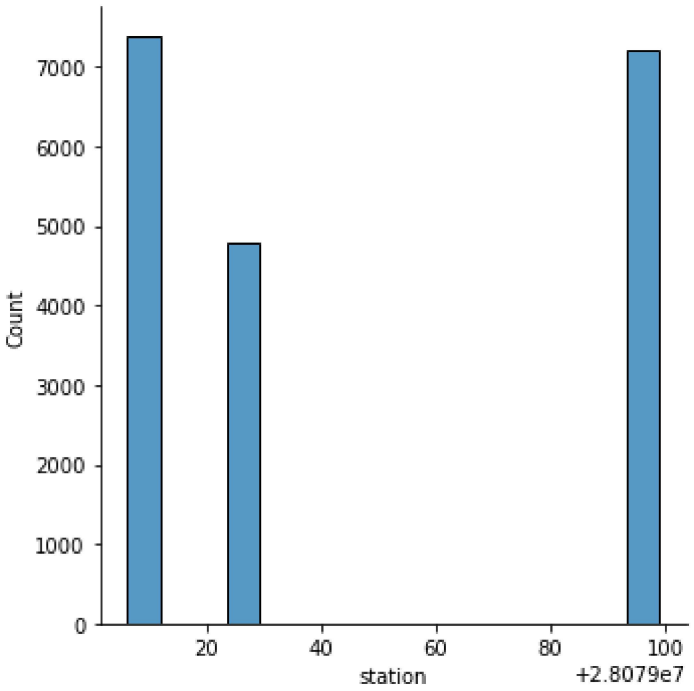
In [19]:
```
sns.pairplot(df1[0:50])
```

Out[19]: <seaborn.axisgrid.PairGrid at 0x268391bf7c0>



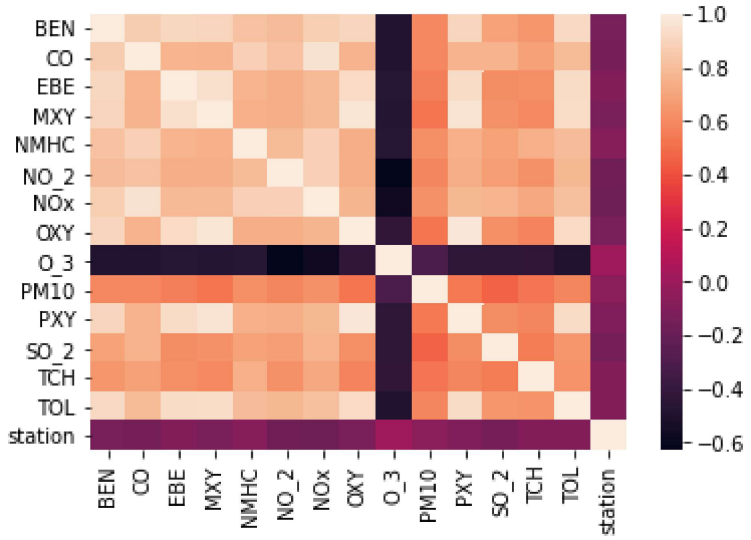
In [20]:
```
sns.displot(df1['station'])
```

Out[20]: <seaborn.axisgrid.FacetGrid at 0x2683d21d0d0>



In [21]:
```
sns.heatmap(df1.corr())
```

Out[21]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [22]:
```
x=df[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [23]:
```
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)
```

Linear Regression

In [24]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[24]:
```
LinearRegression()
```

In [25]:
```
lr.intercept_
```

Out[25]:
```
28079075.8640171
```

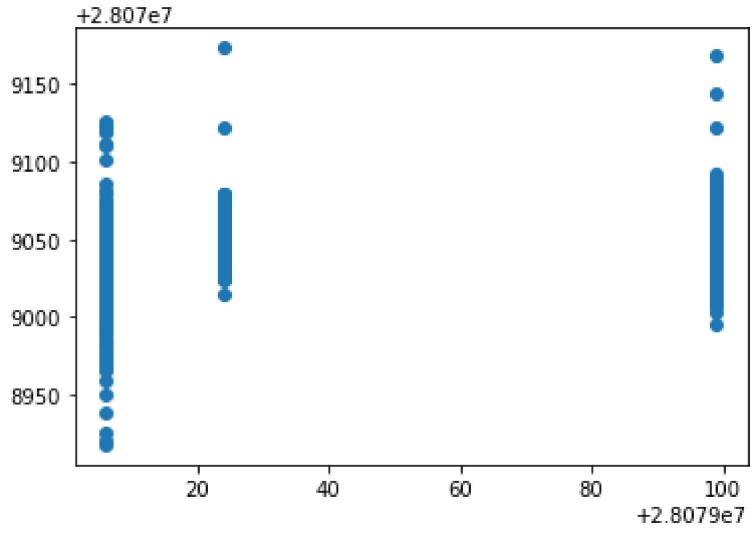
In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

	Co-efficient
BEN	-4.196410
CO	26.854857
EBE	4.172731
MXY	-3.413079
NMHC	81.467188
NO_2	-0.147838
NOx	-0.251817
OXY	-2.217535
O_3	-0.295535
PM10	0.098381
PXY	5.946268
SO_2	-0.193742
TCH	-7.546275
TOL	0.996183

In [27]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[27]:
```
<matplotlib.collections.PathCollection at 0x26847c305b0>
```



ACCURACY

In [28]:
```
lr.score(x_test,y_test)
```

Out[28]: 0.10990354193388185

In [29]:
```
lr.score(x_train,y_train)
```

Out[29]: 0.10464045071918138

Ridge and Lasso

In [30]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [31]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[31]: Ridge(alpha=10)

Accuracy(Ridge)

In [32]:
```
rr.score(x_test,y_test)
```

Out[32]: 0.10806621766961122

In [33]:
```
rr.score(x_train,y_train)
```

Out[33]: 0.1043003233084504

In [34]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[34]:
```
Lasso(alpha=10)
```

In [35]:
```
la.score(x_train,y_train)
```

Out[35]:
```
0.05340672044325423
```

Accuracy(Lasso)

In [36]:
```
la.score(x_test,y_test)
```

Out[36]:
```
0.0512306701375993
```

Elastic Net regression

In [37]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[37]:
```
ElasticNet()
```

In [38]:
```
en.coef_
```

Out[38]:
```
array([-0.        ,  0.40070785,  1.57620087, -1.81505073,  0.        ,
       -0.1652734 , -0.08802389, -0.        , -0.22674954,  0.12357185,
        0.37590709, -0.11768332,  0.        ,  1.02601709])
```

In [39]:
```
en.intercept_
```

Out[39]:
```
28079066.396113858
```

In [40]:
```
prediction=en.predict(x_test)
```

In [41]:
```
en.score(x_test,y_test)
```

Out[41]:
```
0.06655870100813144
```

Evaluation Metrics

In [42]:
```
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)))
```

```
38.79840024825301
1679.0972889522834
40.97678963696745
```

Logistic Regression

In [43]:
```
from sklearn.linear_model import LogisticRegression
```

In [44]:
```
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 [45]:
```
feature_matrix.shape
```

Out[45]: (19397, 14)

In [46]:
```
target_vector.shape
```

Out[46]: (19397,)

In [47]:
```
from sklearn.preprocessing import StandardScaler
```

In [48]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [49]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[49]:
```
LogisticRegression(max_iter=10000)
```

In [50]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14]]
```

In [51]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079006]
```

In [52]:
```
logr.classes_
```

Out[52]: array([28079006, 28079024, 28079099], dtype=int64)

In [53]:
```
logr.score(fs,target_vector)
```

Out[53]:
```
0.7359901015620972
```

In [54]:
```
logr.predict_proba(observation)[0][0]
```

Out[54]:
```
0.9999978295301757
```

In [55]:
```
logr.predict_proba(observation)
```

Out[55]:
```
array([[9.99997830e-01, 7.73941535e-20, 2.17046982e-06]])
```

Random Forest

In [56]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [57]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[57]:
```
RandomForestClassifier()
```

In [58]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [59]:
```
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[59]:
```
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 [60]:
```
grid_search.best_score_
```

Out[60]:
```
0.7830149585107387
```

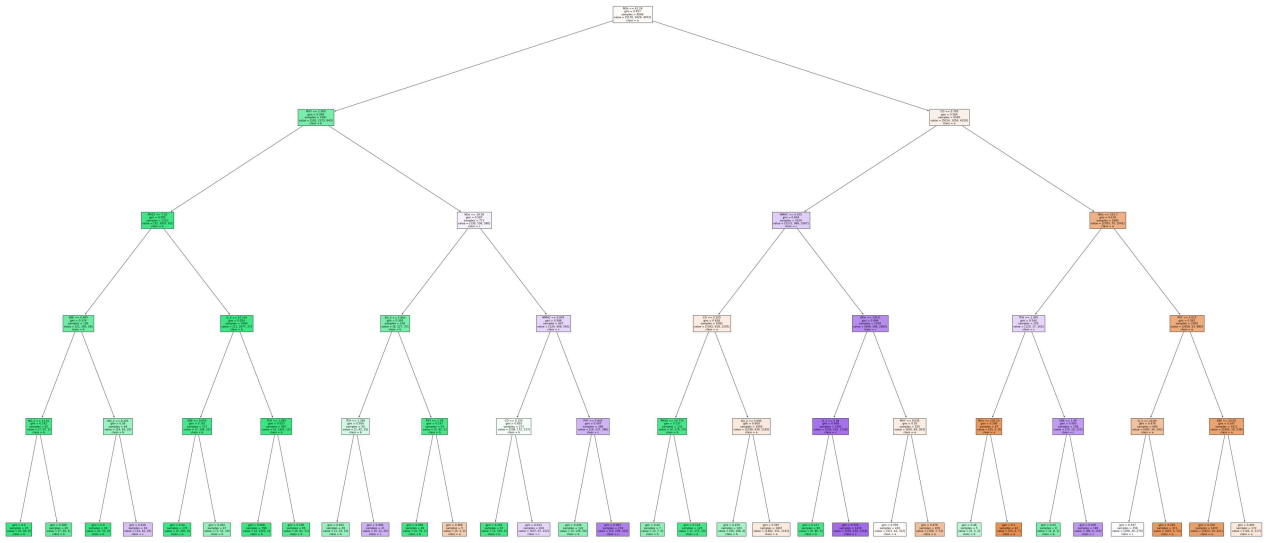
In [61]:
```
rfc_best=grid_search.best_estimator_
```

In [62]:
```
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[62]:
```
[Text(2232.0, 1993.2, 'NOx <= 41.24\ngini = 0.657\nsamples = 8566\nvalue = [5176, 3429,
4972]\nclass = a'),
 Text(1116.0, 1630.8000000000002, 'MXY <= 1.065\ngini = 0.399\nsamples = 1981\nvalue =
[162, 2373, 643]\nclass = b'),
 Text(558.0, 1268.4, 'PM10 <= 7.02\ngini = 0.095\nsamples = 1210\nvalue = [32, 1837, 63]
\nclass = b'),
 Text(279.0, 906.0, 'EBE <= 0.485\ngini = 0.376\nsamples = 126\nvalue = [21, 160, 26]\nc
lass = b'),
 Text(139.5, 543.5999999999999, 'NO_2 <= 13.01\ngini = 0.142\nsamples = 62\nvalue = [7,
97, 1]\nclass = b'),
 Text(69.75, 181.19999999999982, 'gini = 0.0\nsamples = 33\nvalue = [0, 58, 0]\nclass =
b'),
 Text(209.25, 181.19999999999982, 'gini = 0.289\nsamples = 29\nvalue = [7, 39, 1]\nclass
= b'),
 Text(418.5, 543.5999999999999, 'NO_2 <= 9.205\ngini = 0.54\nsamples = 64\nvalue = [14,
63, 25]\nclass = b'),
 Text(348.75, 181.19999999999982, 'gini = 0.0\nsamples = 30\nvalue = [0, 53, 0]\nclass =
b'),
 Text(488.25, 181.19999999999982, 'gini = 0.616\nsamples = 34\nvalue = [14, 10, 25]\ncla
ss = c'),
 Text(837.0, 906.0, 'O_3 <= 57.235\ngini = 0.054\nsamples = 1084\nvalue = [11, 1677, 37]
\nclass = b'),
 Text(697.5, 543.5999999999999, 'EBE <= 0.655\ngini = 0.161\nsamples = 217\nvalue = [7,
336, 25]\nclass = b'),
 Text(627.75, 181.19999999999982, 'gini = 0.04\nsamples = 170\nvalue = [0, 285, 6]\nclas
s = b'),
 Text(767.25, 181.19999999999982, 'gini = 0.492\nsamples = 47\nvalue = [7, 51, 19]\nclas
s = b'),
 Text(976.5, 543.5999999999999, 'TCH <= 1.295\ngini = 0.023\nsamples = 867\nvalue = [4,
1341, 12]\nclass = b'),
 Text(906.75, 181.19999999999982, 'gini = 0.008\nsamples = 798\nvalue = [4, 1250, 1]\ncl
ass = b'),
 Text(1046.25, 181.19999999999982, 'gini = 0.192\nsamples = 69\nvalue = [0, 91, 11]\ncla
ss = b'),
 Text(1674.0, 1268.4, 'NOx <= 19.36\ngini = 0.587\nsamples = 771\nvalue = [130, 536, 58
0]\nclass = c'),
 Text(1395.0, 906.0, 'SO_2 <= 5.805\ngini = 0.393\nsamples = 104\nvalue = [6, 127, 37]\n
class = b'),
 Text(1255.5, 543.5999999999999, 'TCH <= 1.285\ngini = 0.504\nsamples = 50\nvalue = [1,
45, 35]\nclass = b'),
 Text(1185.75, 181.19999999999982, 'gini = 0.452\nsamples = 29\nvalue = [1, 33, 15]\ncla
ss = b'),
 Text(1325.25, 181.19999999999982, 'gini = 0.469\nsamples = 21\nvalue = [0, 12, 20]\ncla
ss = c'),
 Text(1534.5, 543.5999999999999, 'PXY <= 1.38\ngini = 0.147\nsamples = 54\nvalue = [5, 8
2, 2]\nclass = b'),
 Text(1464.75, 181.19999999999982, 'gini = 0.048\nsamples = 49\nvalue = [0, 79, 2]\nclas
s = b'),
 Text(1604.25, 181.19999999999982, 'gini = 0.469\nsamples = 5\nvalue = [5, 3, 0]\nclass
= a'),
 Text(1953.0, 906.0, 'NMHC <= 0.045\ngini = 0.588\nsamples = 667\nvalue = [124, 409, 54
3]\nclass = c'),
 Text(1813.5, 543.5999999999999, 'CO <= 0.155\ngini = 0.655\nsamples = 271\nvalue = [10
8, 172, 157]\nclass = b'),
 Text(1743.75, 181.19999999999982, 'gini = 0.103\nsamples = 67\nvalue = [1, 105, 5]\ncla
ss = b'),
 Text(1883.25, 181.19999999999982, 'gini = 0.633\nsamples = 204\nvalue = [107, 67, 152]
\nclass = c'),
 Text(2092.5, 543.5999999999999, 'PXY <= 0.645\ngini = 0.497\nsamples = 396\nvalue = [1
6, 237, 386]\nclass = c'),
 Text(2022.75, 181.19999999999982, 'gini = 0.436\nsamples = 122\nvalue = [3, 128, 54]\nc
lass = b'),
 Text(2162.25, 181.19999999999982, 'gini = 0.407\nsamples = 274\nvalue = [13, 109, 332]
\nclass = c'),
 Text(3348.0, 1630.8000000000002, 'CO <= 0.765\ngini = 0.584\nsamples = 6585\nvalue = [5
```

```
014, 1056, 4329]\nclass = a'),
 Text(2790.0, 1268.4, 'NMHC <= 0.105\ngini = 0.604\nsamples = 4100\nvalue = [2231, 986,
3287]\nclass = c'),
 Text(2511.0, 906.0, 'CO <= 0.205\ngini = 0.628\nsamples = 2082\nvalue = [1542, 618, 120
5]\nclass = a'),
 Text(2371.5, 543.5999999999999, 'PM10 <= 10.775\ngini = 0.137\nsamples = 132\nvalue =
[4, 179, 10]\nclass = b'),
 Text(2301.75, 181.19999999999982, 'gini = 0.42\nsamples = 7\nvalue = [3, 7, 0]\nclass =
b'),
 Text(2441.25, 181.19999999999982, 'gini = 0.114\nsamples = 125\nvalue = [1, 172, 10]\nc
lass = b'),
 Text(2650.5, 543.5999999999999, 'SO_2 <= 5.095\ngini = 0.604\nsamples = 1950\nvalue =
[1538, 439, 1195]\nclass = a'),
 Text(2580.75, 181.19999999999982, 'gini = 0.475\nsamples = 103\nvalue = [56, 108, 4]\nc
lass = b'),
 Text(2720.25, 181.19999999999982, 'gini = 0.587\nsamples = 1847\nvalue = [1482, 331, 11
91]\nclass = a'),
 Text(3069.0, 906.0, 'NOx <= 130.6\ngini = 0.498\nsamples = 2018\nvalue = [689, 368, 208
2]\nclass = c'),
 Text(2929.5, 543.5999999999999, 'O_3 <= 5.38\ngini = 0.408\nsamples = 1493\nvalue = [25
9, 319, 1719]\nclass = c'),
 Text(2859.75, 181.19999999999982, 'gini = 0.111\nsamples = 60\nvalue = [0, 80, 5]\nclas
s = b'),
 Text(2999.25, 181.19999999999982, 'gini = 0.374\nsamples = 1433\nvalue = [259, 239, 171
4]\nclass = c'),
 Text(3208.5, 543.5999999999999, 'MXY <= 9.035\ngini = 0.55\nsamples = 525\nvalue = [43
0, 49, 363]\nclass = a'),
 Text(3138.75, 181.19999999999982, 'gini = 0.556\nsamples = 416\nvalue = [321, 42, 312]
\nclass = a'),
 Text(3278.25, 181.19999999999982, 'gini = 0.479\nsamples = 109\nvalue = [109, 7, 51]\nc
lass = a'),
 Text(3906.0, 1268.4, 'NOx <= 153.7\ngini = 0.418\nsamples = 2485\nvalue = [2783, 70, 10
42]\nclass = a'),
 Text(3627.0, 906.0, 'TCH <= 1.365\ngini = 0.544\nsamples = 205\nvalue = [125, 17, 162]
\nclass = c'),
 Text(3487.5, 543.5999999999999, 'NOx <= 118.15\ngini = 0.306\nsamples = 47\nvalue = [5
5, 3, 9]\nclass = a'),
 Text(3417.75, 181.19999999999982, 'gini = 0.48\nsamples = 5\nvalue = [0, 3, 2]\nclass =
b'),
 Text(3557.25, 181.19999999999982, 'gini = 0.2\nsamples = 42\nvalue = [55, 0, 7]\nclass
= a'),
 Text(3766.5, 543.5999999999999, 'EBE <= 1.49\ngini = 0.493\nsamples = 158\nvalue = [70,
14, 153]\nclass = c'),
 Text(3696.75, 181.19999999999982, 'gini = 0.43\nsamples = 9\nvalue = [2, 8, 1]\nclass =
b'),
 Text(3836.25, 181.19999999999982, 'gini = 0.456\nsamples = 149\nvalue = [68, 6, 152]\nc
lass = c'),
 Text(4185.0, 906.0, 'PXY <= 3.015\ngini = 0.392\nsamples = 2280\nvalue = [2658, 53, 88
0]\nclass = a'),
 Text(4045.5, 543.5999999999999, 'O_3 <= 10.85\ngini = 0.476\nsamples = 669\nvalue = [69
3, 34, 342]\nclass = a'),
 Text(3975.75, 181.19999999999982, 'gini = 0.547\nsamples = 358\nvalue = [268, 29, 270]
\nclass = c'),
 Text(4115.25, 181.19999999999982, 'gini = 0.263\nsamples = 311\nvalue = [425, 5, 72]\nc
lass = a'),
 Text(4324.5, 543.5999999999999, 'EBE <= 10.05\ngini = 0.347\nsamples = 1611\nvalue = [1
965, 19, 538]\nclass = a'),
 Text(4254.75, 181.19999999999982, 'gini = 0.316\nsamples = 1439\nvalue = [1823, 19, 42
1]\nclass = a'),
 Text(4394.25, 181.19999999999982, 'gini = 0.495\nsamples = 172\nvalue = [142, 0, 117]\n
class = a')]
```



Conclusion

Accuracy

Linear Regression:0.10990354193388185

Ridge Regression:0.10806621766961122

Lasso Regression:0.0512306701375993

ElasticNet Regression:0.06655870100813144

Logistic Regression:0.7359901015620972

Random Forest:0.7830149585107387

Accuracy for Random forest is higher so it is the best fit model