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_2018.csv")
df
```

Out[2]:

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL
0	01-03-2018 01:00	NaN	NaN	0.3	NaN	NaN	1.0	29.0	31.0	NaN	NaN	NaN	2.0	NaN	NaN
1	01-03-2018 01:00	0.5	1.39	0.3	0.2	0.02	6.0	40.0	49.0	52.0	5.0	4.0	3.0	1.41	0.8
2	01-03-2018 01:00	0.4	NaN	NaN	0.2	NaN	4.0	41.0	47.0	NaN	NaN	NaN	NaN	NaN	1.1
3	01-03-2018 01:00	NaN	NaN	0.3	NaN	NaN	1.0	35.0	37.0	54.0	NaN	NaN	NaN	NaN	NaN
4	01-03-2018 01:00	NaN	NaN	NaN	NaN	NaN	1.0	27.0	29.0	49.0	NaN	NaN	3.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
69091	01-02-2018 00:00	NaN	NaN	0.5	NaN	NaN	66.0	91.0	192.0	1.0	35.0	22.0	NaN	NaN	NaN
69092	01-02-2018 00:00	NaN	NaN	0.7	NaN	NaN	87.0	107.0	241.0	NaN	29.0	NaN	15.0	NaN	NaN
69093	01-02-2018 00:00	NaN	NaN	NaN	NaN	NaN	28.0	48.0	91.0	2.0	NaN	NaN	NaN	NaN	NaN

	date	BEN	CH4	CO	EBE	NMHC	NO	NO_2	NOx	O_3	PM10	PM25	SO_2	TCH	TOL
69094	01-02-2018 00:00	NaN	NaN	NaN	NaN	NaN	141.0	103.0	320.0	2.0	NaN	NaN	NaN	NaN	NaN
69095	01-02-2018 00:00	NaN	NaN	NaN	NaN	NaN	69.0	96.0	202.0	3.0	26.0	NaN	NaN	NaN	NaN

69096 rows × 16 columns

Data Cleaning and Data Preprocessing

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

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

Out[4]:
```
Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4562 entries, 1 to 69078
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     4562 non-null   object 
 1   BEN      4562 non-null   float64
 2   CH4      4562 non-null   float64
 3   CO       4562 non-null   float64
 4   EBE      4562 non-null   float64
 5   NMHC     4562 non-null   float64
 6   NO       4562 non-null   float64
 7   NO_2     4562 non-null   float64
 8   NOx      4562 non-null   float64
 9   O_3      4562 non-null   float64
 10  PM10     4562 non-null   float64
 11  PM25     4562 non-null   float64
 12  SO_2     4562 non-null   float64
 13  TCH      4562 non-null   float64
 14  TOL      4562 non-null   float64
 15  station  4562 non-null   int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 605.9+ KB
```

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

Out[6]:

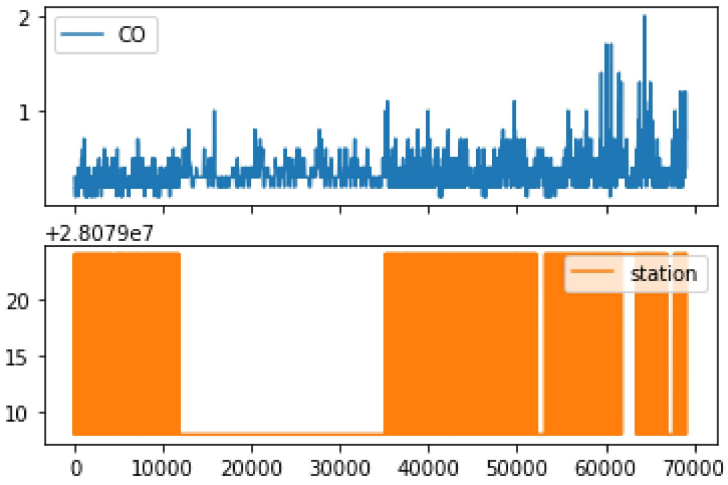
	CO	station
1	0.3	28079008
6	0.2	28079024
25	0.2	28079008
30	0.2	28079024
49	0.2	28079008
...	...	...
69030	0.7	28079024
69049	1.2	28079008
69054	0.6	28079024
69073	1.0	28079008
69078	0.4	28079024

4562 rows × 2 columns

Line chart

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

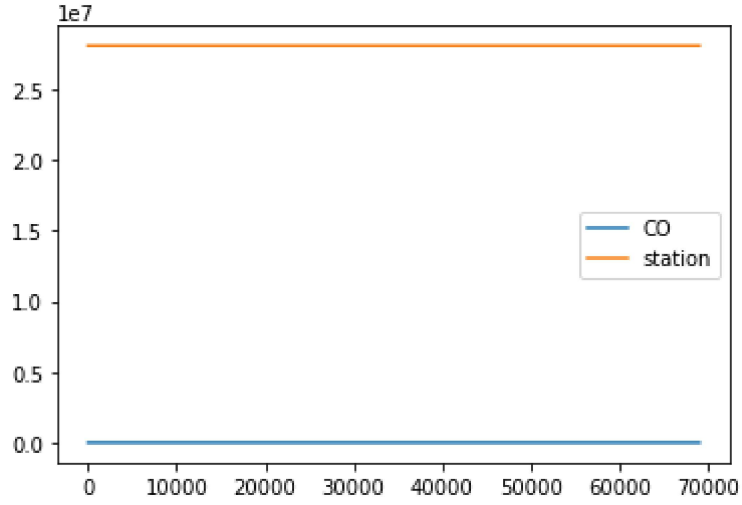
Out[7]:
```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```



Line chart

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

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

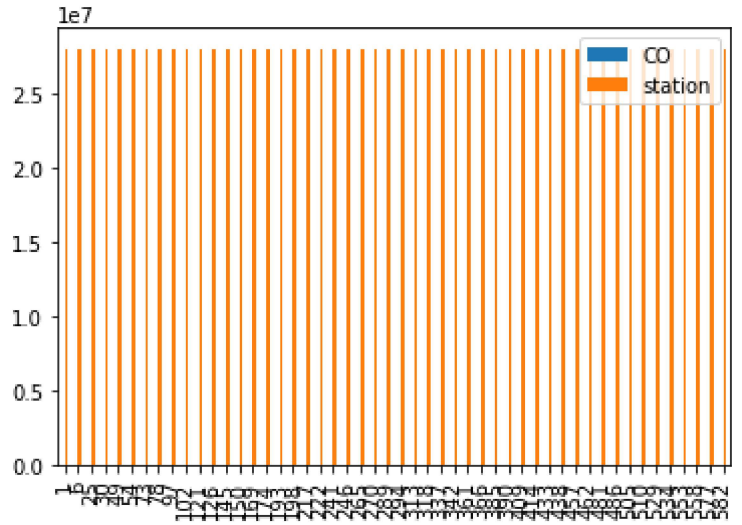


Bar chart

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

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

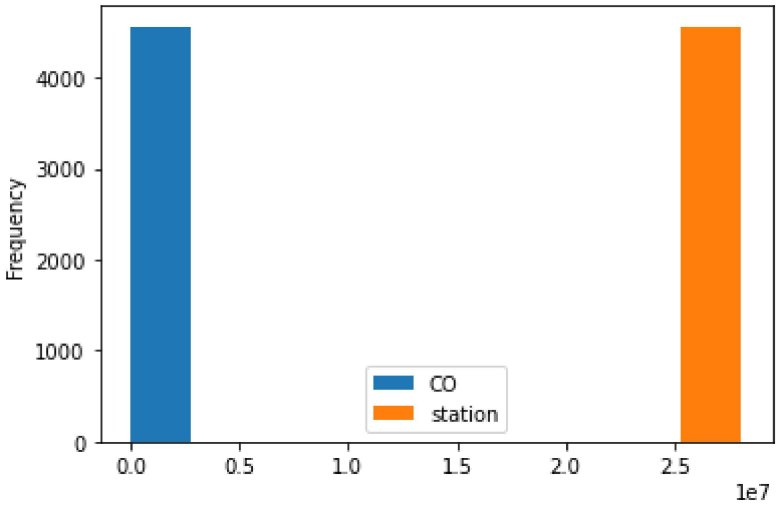
Out[10]: <AxesSubplot:>



Histogram

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

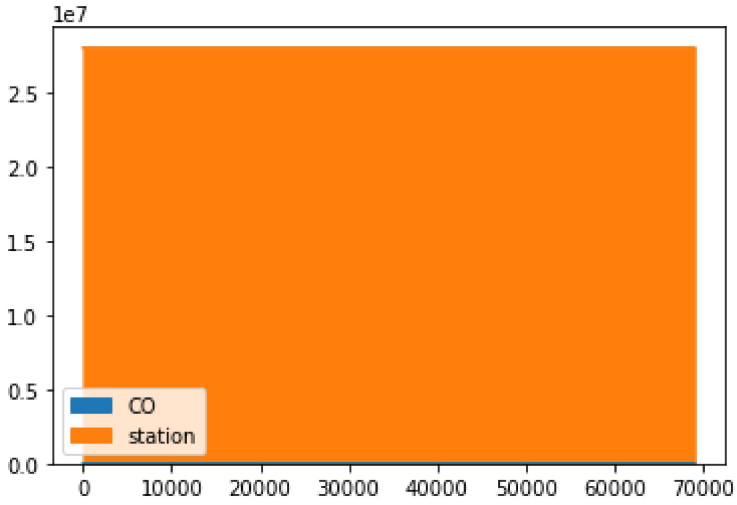
Out[11]: <AxesSubplot:ylabel='Frequency'>



Area chart

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

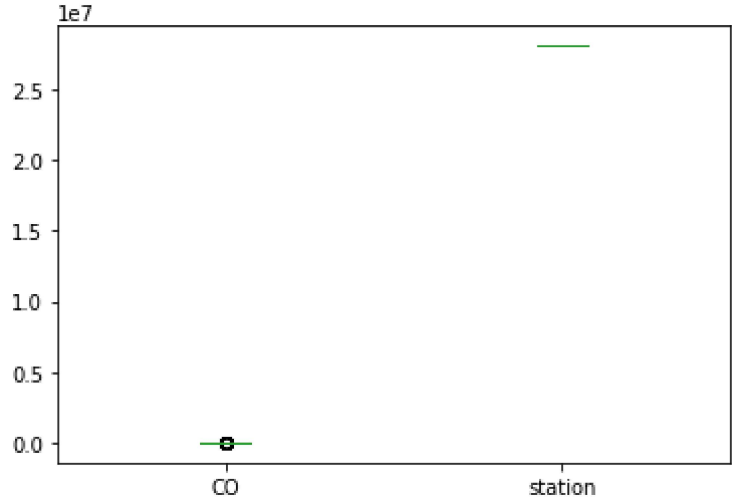
Out[12]: <AxesSubplot:>



Box chart

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

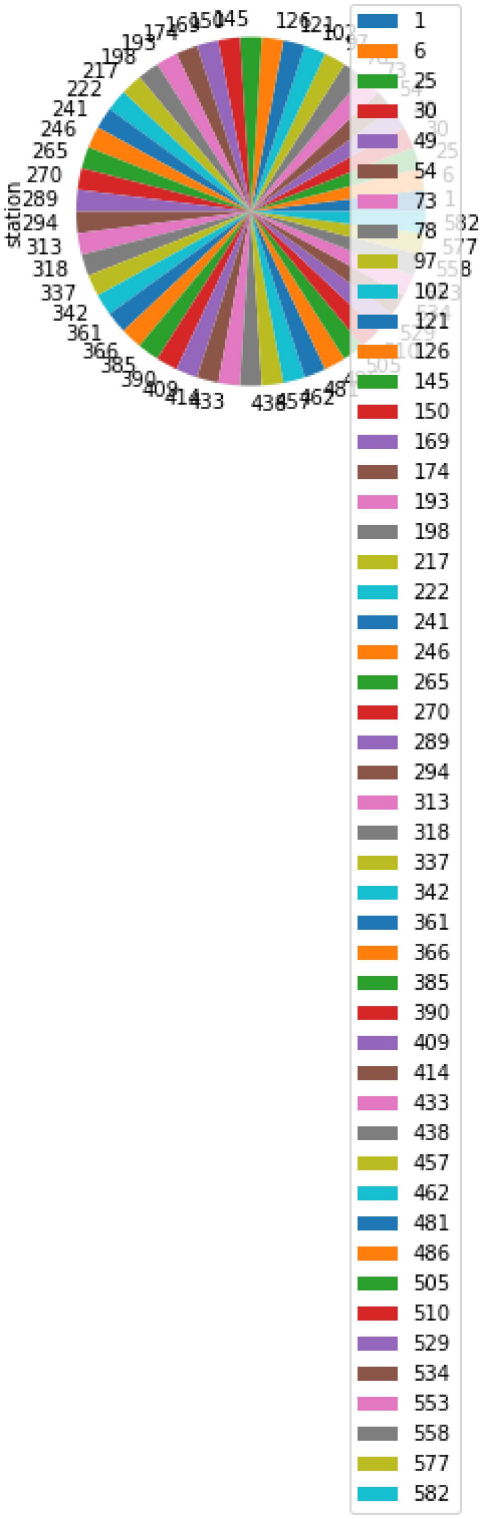
Out[13]: <AxesSubplot:>



Pie chart

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

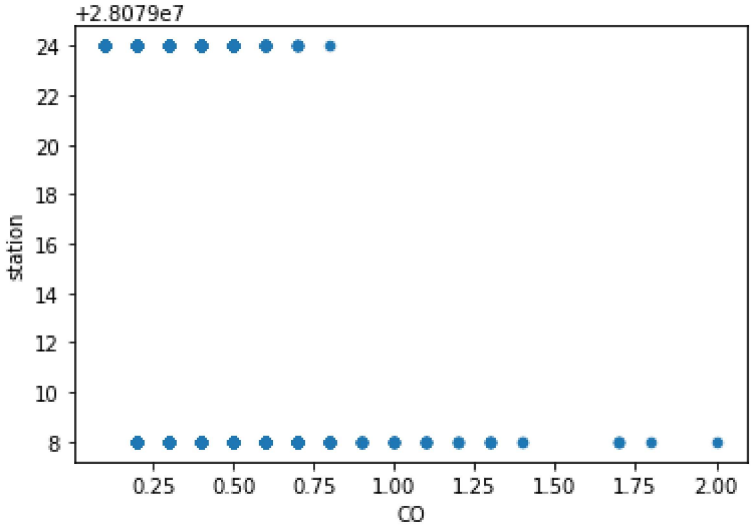
Out[14]:
```
<AxesSubplot:ylabel='station'>
```



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: 4562 entries, 1 to 69078
Data columns (total 16 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   date     4562 non-null   object 
 1   BEN      4562 non-null   float64
 2   CH4      4562 non-null   float64
 3   CO       4562 non-null   float64
 4   EBE      4562 non-null   float64
 5   NMHC     4562 non-null   float64
 6   NO       4562 non-null   float64
 7   NO_2     4562 non-null   float64
 8   NOx      4562 non-null   float64
 9   O_3      4562 non-null   float64
 10  PM10     4562 non-null   float64
 11  PM25     4562 non-null   float64
 12  SO_2     4562 non-null   float64
 13  TCH      4562 non-null   float64
 14  TOL      4562 non-null   float64
 15  station  4562 non-null   int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 605.9+ KB
```

In [17]:
```
df.columns
```

Out[17]:
```
Index(['date', 'BEN', 'CH4', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'NOx', 'O_3',
       'PM10', 'PM25', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [18]:
```
df.describe()
```

Out[18]:

	BEN	CH4	CO	EBE	NMHC	NO	NO_2	
count	4562.00000	4562.000000	4562.000000	4562.000000	4562.000000	4562.000000	4562.000000	4562.0
mean	0.69349	1.329163	0.330579	0.286782	0.056773	21.742218	44.152126	77.4
std	0.46832	0.214399	0.161489	0.354442	0.037711	35.539531	30.234015	79.2

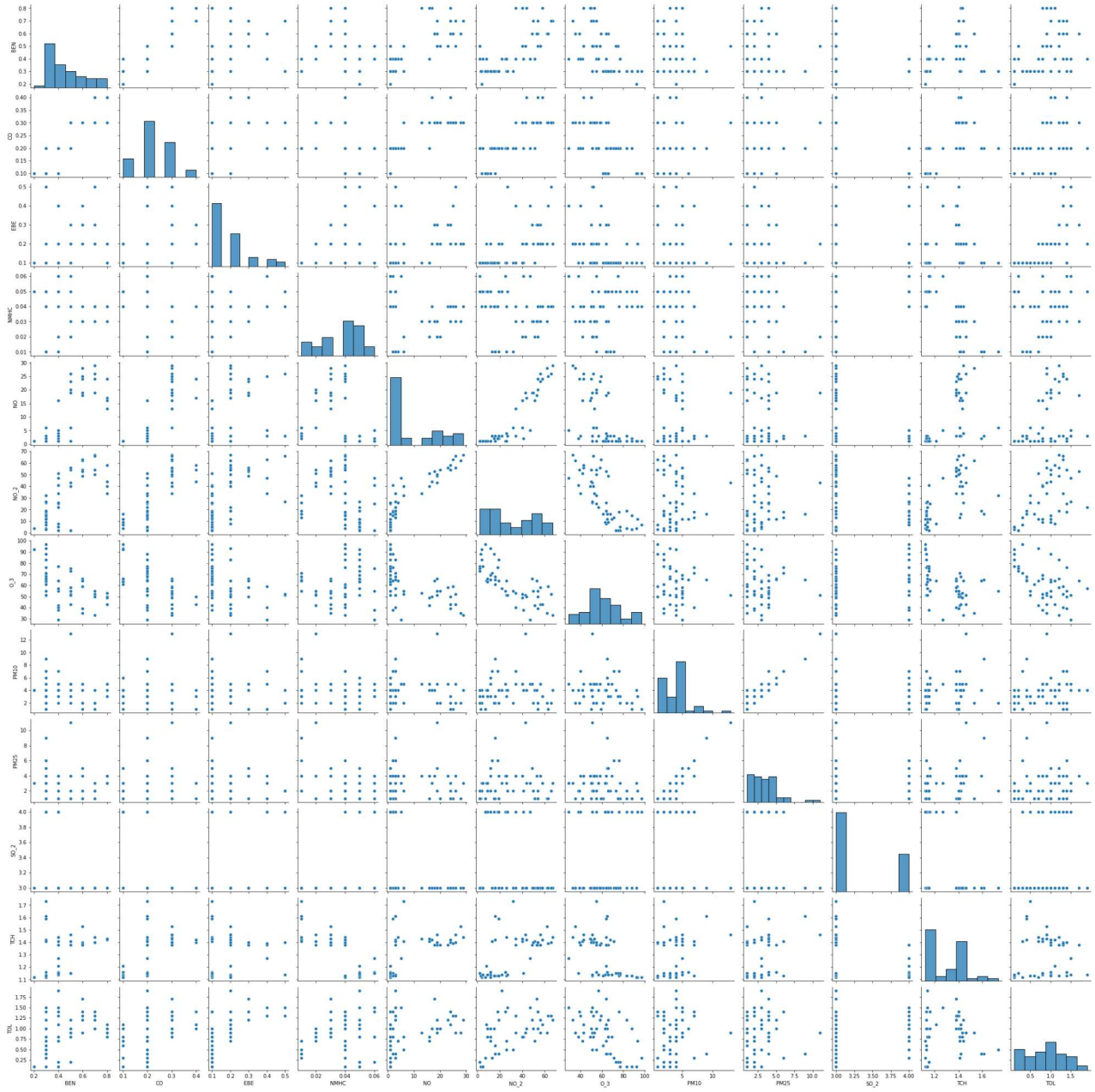
	BEN	CH4	CO	EBE	NMHC	NO	NO_2	
min	0.10000	0.020000	0.100000	0.100000	0.000000	1.000000	1.000000	2.0
25%	0.40000	1.120000	0.200000	0.100000	0.030000	1.000000	20.000000	24.0
50%	0.60000	1.390000	0.300000	0.200000	0.050000	9.000000	41.000000	56.0
75%	0.90000	1.420000	0.400000	0.300000	0.070000	27.000000	64.000000	106.0
max	6.60000	3.920000	2.000000	7.400000	0.490000	431.000000	184.000000	844.0

In [19]:
```
df1=df[['date', 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
```

EDA AND VISUALIZATION

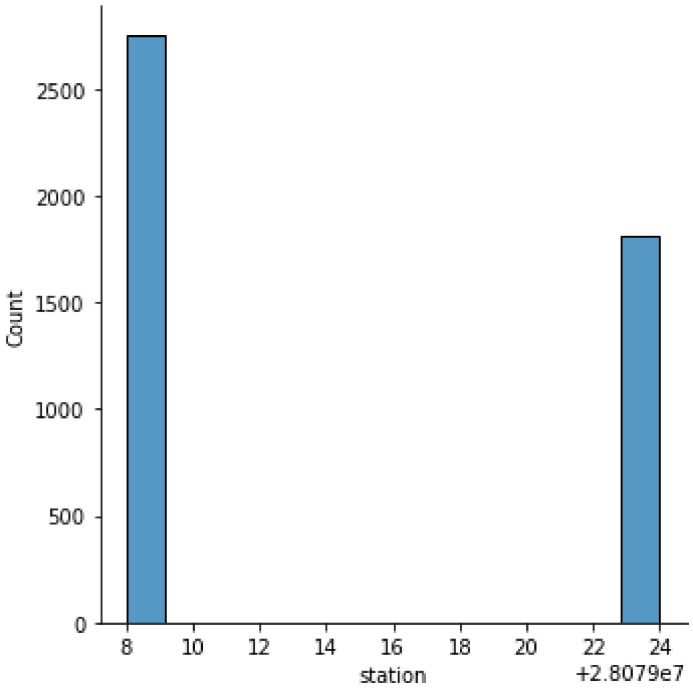
In [20]:
```
sns.pairplot(df1[0:50])
```

Out[20]: <seaborn.axisgrid.PairGrid at 0x226ce510550>



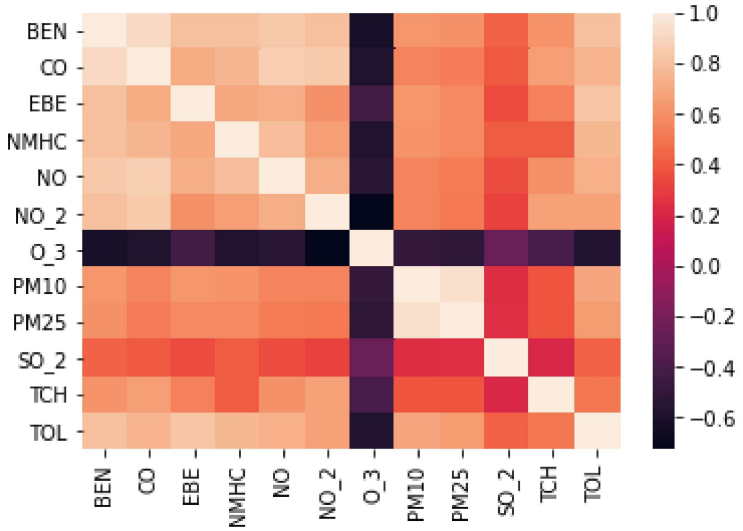
In [21]:
```
sns.displot(df['station'])
```

Out[21]:
```
<seaborn.axisgrid.FacetGrid at 0x226ce4e5700>
```



In [22]:
```
sns.heatmap(df1.corr())
```

Out[22]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [23]:
```
x=df[[ 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
y=df['station']
```

In [24]:
```
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 [25]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[25]:
```
LinearRegression()
```

In [26]:
```
lr.intercept_
```

Out[26]:
```
28079045.149262898
```

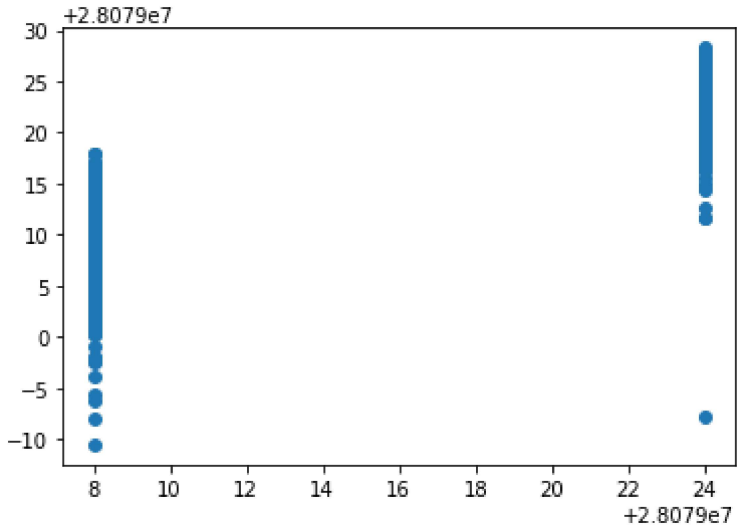
In [27]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[27]:

	Co-efficient
BEN	-1.236215
CO	-21.087221
EBE	0.617714
NMHC	143.711323
NO	0.032049
NO_2	-0.154326
O_3	-0.084711
PM10	0.086061
PM25	0.047837
SO_2	-0.033933
TCH	-16.286402
TOL	-0.143127

In [28]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[28]:
```
<matplotlib.collections.PathCollection at 0x226d67113a0>
```



ACCURACY

In [29]:
```
lr.score(x_test,y_test)
```

Out[29]: 0.8174389328742364

In [30]:
```
lr.score(x_train,y_train)
```

Out[30]: 0.8067119657607368

Ridge and Lasso

In [31]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [32]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[32]: Ridge(alpha=10)

Accuracy(Ridge)

In [33]:
```
rr.score(x_test,y_test)
```

Out[33]: 0.7090233455872081

In [34]:
```
rr.score(x_train,y_train)
```

Out[34]: 0.7047722419366638

In [35]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[35]:
```
Lasso(alpha=10)
```

In [36]:
```
la.score(x_train,y_train)
```

Out[36]:
```
0.4233834020085827
```

Accuracy(Lasso)

In [37]:
```
la.score(x_test,y_test)
```

Out[37]:
```
0.42711197070225293
```

Elastic Net regression

In [38]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[38]:
```
ElasticNet()
```

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

Out[39]:
```
array([-0.        , -0.        , -0.        ,  0.        ,  0.03503798,
       -0.29388983, -0.14913886,  0.29100896, -0.10448349,  0.05485943,
       -0.10273661,  0.        ])
```

In [40]:
```
en.intercept_
```

Out[40]:
```
28079030.227815483
```

In [41]:
```
prediction=en.predict(x_test)
```

In [42]:
```
en.score(x_test,y_test)
```

Out[42]:
```
0.4677676147550599
```

Evaluation Metrics

In [43]:
```
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)))
```

```
4.8040206322162105
32.58660843320153
5.708468133676629
```

Logistic Regression

In [44]:
```
from sklearn.linear_model import LogisticRegression
```

In [45]:
```
feature_matrix=df[[ 'BEN', 'CO', 'EBE', 'NMHC', 'NO', 'NO_2', 'O_3', 'PM10', 'PM25',
       'SO_2', 'TCH', 'TOL']]
target_vector=df[ 'station']
```

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

Out[46]: (4562, 12)

In [47]:
```
target_vector.shape
```

Out[47]: (4562,)

In [48]:
```
from sklearn.preprocessing import StandardScaler
```

In [49]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [50]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[50]: LogisticRegression(max_iter=10000)

In [51]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12]]
```

In [52]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079008]
```

In [53]:
```
logr.classes_
```

Out[53]: array([28079008, 28079024], dtype=int64)

In [54]:
```
logr.score(fs,target_vector)
```

Out[54]:
```
0.9890398947829899
```

In [55]:
```
logr.predict_proba(observation)[0][0]
```

Out[55]:
```
1.0
```

In [56]:
```
logr.predict_proba(observation)
```

Out[56]:
```
array([[1.00000000e+00, 1.09190928e-22]])
```

Random Forest

In [57]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [58]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[58]:
```
RandomForestClassifier()
```

In [59]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [60]:
```
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[60]:
```
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 [61]:
```
grid_search.best_score_
```

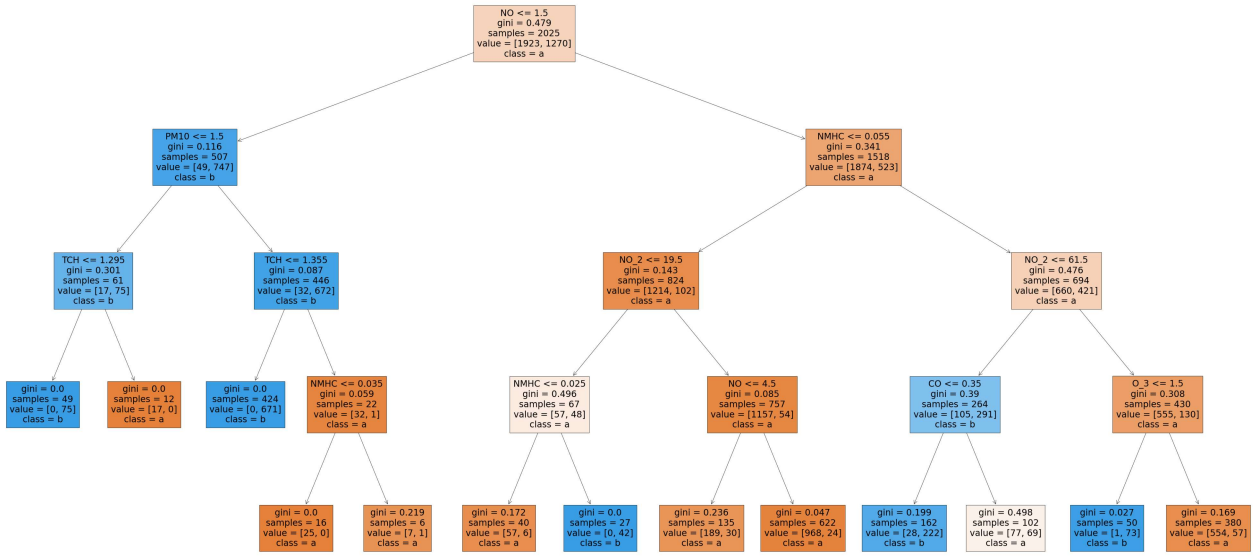
Out[61]:
```
0.9921700776675565
```

In [62]:
```
rfc_best=grid_search.best_estimator_
```

In [63]:
```
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[63]:
```
[Text(1874.88, 1956.96, 'NO <= 1.5\ngini = 0.479\nsamples = 2025\nvalue = [1923, 1270]\n
class = a'),
 Text(714.24, 1522.0800000000002, 'PM10 <= 1.5\ngini = 0.116\nsamples = 507\nvalue = [4
9, 747]\nclass = b'),
 Text(357.12, 1087.2, 'TCH <= 1.295\ngini = 0.301\nsamples = 61\nvalue = [17, 75]\nclass
= b'),
 Text(178.56, 652.3200000000002, 'gini = 0.0\nsamples = 49\nvalue = [0, 75]\nclass =
b'),
 Text(535.6800000000001, 652.3200000000002, 'gini = 0.0\nsamples = 12\nvalue = [17, 0]\n
class = a'),
 Text(1071.3600000000001, 1087.2, 'TCH <= 1.355\ngini = 0.087\nsamples = 446\nvalue = [3
2, 672]\nclass = b'),
 Text(892.8, 652.3200000000002, 'gini = 0.0\nsamples = 424\nvalue = [0, 671]\nclass =
b'),
 Text(1249.92, 652.3200000000002, 'NMHC <= 0.035\ngini = 0.059\nsamples = 22\nvalue = [3
2, 1]\nclass = a'),
 Text(1071.3600000000001, 217.44000000000005, 'gini = 0.0\nsamples = 16\nvalue = [25, 0]
\nclass = a'),
 Text(1428.48, 217.44000000000005, 'gini = 0.219\nsamples = 6\nvalue = [7, 1]\nclass =
a'),
 Text(3035.52, 1522.0800000000002, 'NMHC <= 0.055\ngini = 0.341\nsamples = 1518\nvalue =
[1874, 523]\nclass = a'),
 Text(2321.28, 1087.2, 'NO_2 <= 19.5\ngini = 0.143\nsamples = 824\nvalue = [1214, 102]\n
class = a'),
 Text(1964.16, 652.3200000000002, 'NMHC <= 0.025\ngini = 0.496\nsamples = 67\nvalue = [5
7, 48]\nclass = a'),
 Text(1785.6, 217.44000000000005, 'gini = 0.172\nsamples = 40\nvalue = [57, 6]\nclass =
a'),
 Text(2142.7200000000003, 217.44000000000005, 'gini = 0.0\nsamples = 27\nvalue = [0, 42]
\nclass = b'),
 Text(2678.4, 652.3200000000002, 'NO <= 4.5\ngini = 0.085\nsamples = 757\nvalue = [1157,
54]\nclass = a'),
 Text(2499.84, 217.44000000000005, 'gini = 0.236\nsamples = 135\nvalue = [189, 30]\nclas
s = a'),
 Text(2856.96, 217.44000000000005, 'gini = 0.047\nsamples = 622\nvalue = [968, 24]\nclas
s = a'),
 Text(3749.76, 1087.2, 'NO_2 <= 61.5\ngini = 0.476\nsamples = 694\nvalue = [660, 421]\nc
lass = a'),
 Text(3392.64, 652.3200000000002, 'CO <= 0.35\ngini = 0.39\nsamples = 264\nvalue = [105,
291]\nclass = b'),
 Text(3214.08, 217.44000000000005, 'gini = 0.199\nsamples = 162\nvalue = [28, 222]\nclas
s = b'),
 Text(3571.2, 217.44000000000005, 'gini = 0.498\nsamples = 102\nvalue = [77, 69]\nclass
= a'),
 Text(4106.88, 652.3200000000002, 'O_3 <= 1.5\ngini = 0.308\nsamples = 430\nvalue = [55
5, 130]\nclass = a'),
 Text(3928.32, 217.44000000000005, 'gini = 0.027\nsamples = 50\nvalue = [1, 73]\nclass =
b'),
 Text(4285.4400000000005, 217.44000000000005, 'gini = 0.169\nsamples = 380\nvalue = [55
4, 57]\nclass = a')]
```



Conclusion

Accuracy

linear regression

In [64]:
```
lr.score(x_test,y_test)
```

Out[64]: 0.8174389328742364

Ridge regression

In [65]:
```
rr.score(x_test,y_test)
```

Out[65]: 0.7090233455872081

Lasso regression

In [66]:
```
la.score(x_test,y_test)
```

Out[66]: 0.42711197070225293

Elastic net regression

In [67]:
```
en.score(x_test,y_test)
```

Out[67]: 0.4677676147550599

Logistic regression

In [68]:
```
logr.score(fs,target_vector)
```

Out[68]: 0.9890398947829899

Random forest

In [69]:
```
grid_search.best_score_
```

Out[69]: 0.9921700776675565

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