In [6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [7]:
```
df=pd.read_csv(r"C:\Users\Admin\Downloads\csvs_per_year\csvs_per_year\madrid_20
df
```

Out[7]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM
0	2009-10-01 01:00:00	NaN	0.27	NaN	NaN	NaN	39.889999	48.150002	NaN	50.680000	18.2600
1	2009-10-01 01:00:00	NaN	0.22	NaN	NaN	NaN	21.230000	24.260000	NaN	55.880001	10.5800
2	2009-10-01 01:00:00	NaN	0.18	NaN	NaN	NaN	31.230000	34.880001	NaN	49.060001	25.1900
3	2009-10-01 01:00:00	0.95	0.33	1.43	2.68	0.25	55.180000	81.360001	1.57	36.669998	26.5300
4	2009-10-01 01:00:00	NaN	0.41	NaN	NaN	0.12	61.349998	76.260002	NaN	38.090000	23.7600
...	...	...	...	...	...	...	...	...	...	...	
215683	2009-06-01 00:00:00	0.50	0.22	0.39	0.75	0.09	22.000000	24.510000	1.00	82.239998	10.8300
215684	2009-06-01 00:00:00	NaN	0.31	NaN	NaN	NaN	76.110001	101.099998	NaN	41.220001	9.9200
215685	2009-06-01 00:00:00	0.13	NaN	0.86	NaN	0.23	81.050003	99.849998	NaN	24.830000	12.4600
215686	2009-06-01 00:00:00	0.21	NaN	2.96	NaN	0.10	72.419998	82.959999	NaN	NaN	13.0300
215687	2009-06-01 00:00:00	0.37	0.32	0.99	1.36	0.14	54.290001	64.480003	1.06	56.919998	15.3600

215688 rows × 17 columns

In [8]:
```
df1 = df.fillna(0)
df1
```

Out[8]:
```
                  date  BEN   CO   EBE   MXY  NMHC       NO_2        NOx   OXY        O_3       PM
     0     2009-10-01  0.00  0.27  0.00  0.00  0.00  39.889999   48.150002  0.00  50.680000  18.2600
           01:00:00
     1     2009-10-01  0.00  0.22  0.00  0.00  0.00  21.230000   24.260000  0.00  55.880001  10.5800
           01:00:00
     2     2009-10-01  0.00  0.18  0.00  0.00  0.00  31.230000   34.880001  0.00  49.060001  25.1900
           01:00:00
     3     2009-10-01  0.95  0.33  1.43  2.68  0.25  55.180000   81.360001  1.57  36.669998  26.5300
           01:00:00
     4     2009-10-01  0.00  0.41  0.00  0.00  0.12  61.349998   76.260002  0.00  38.090000  23.7600
           01:00:00
   ...            ...   ...   ...   ...   ...   ...        ...         ...   ...        ...
215683     2009-06-01  0.50  0.22  0.39  0.75  0.09  22.000000   24.510000  1.00  82.239998  10.8300
           00:00:00
215684     2009-06-01  0.00  0.31  0.00  0.00  0.00  76.110001  101.099998  0.00  41.220001   9.9200
           00:00:00
215685     2009-06-01  0.13  0.00  0.86  0.00  0.23  81.050003   99.849998  0.00  24.830000  12.4600
           00:00:00
215686     2009-06-01  0.21  0.00  2.96  0.00  0.10  72.419998   82.959999  0.00   0.000000  13.0300
           00:00:00
215687     2009-06-01  0.37  0.32  0.99  1.36  0.14  54.290001   64.480003  1.06  56.919998  15.3600
           00:00:00

215688 rows × 17 columns
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215688 entries, 0 to 215687
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     215688 non-null  object 
 1   BEN      60082 non-null   float64
 2   CO       190801 non-null  float64
 3   EBE      60081 non-null   float64
 4   MXY      24846 non-null   float64
 5   NMHC     74748 non-null   float64
 6   NO_2     214562 non-null  float64
 7   NOx      214565 non-null  float64
 8   OXY      24854 non-null   float64
 9   O_3      204482 non-null  float64
 10  PM10     196331 non-null  float64
 11  PM25     55822 non-null   float64
 12  PXY      24854 non-null   float64
 13  SO_2     212671 non-null  float64
 14  TCH      75213 non-null   float64
 15  TOL      59920 non-null   float64
 16  station  215688 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 28.0+ MB
```

In [10]:

```
df.columns
```

Out[10]:

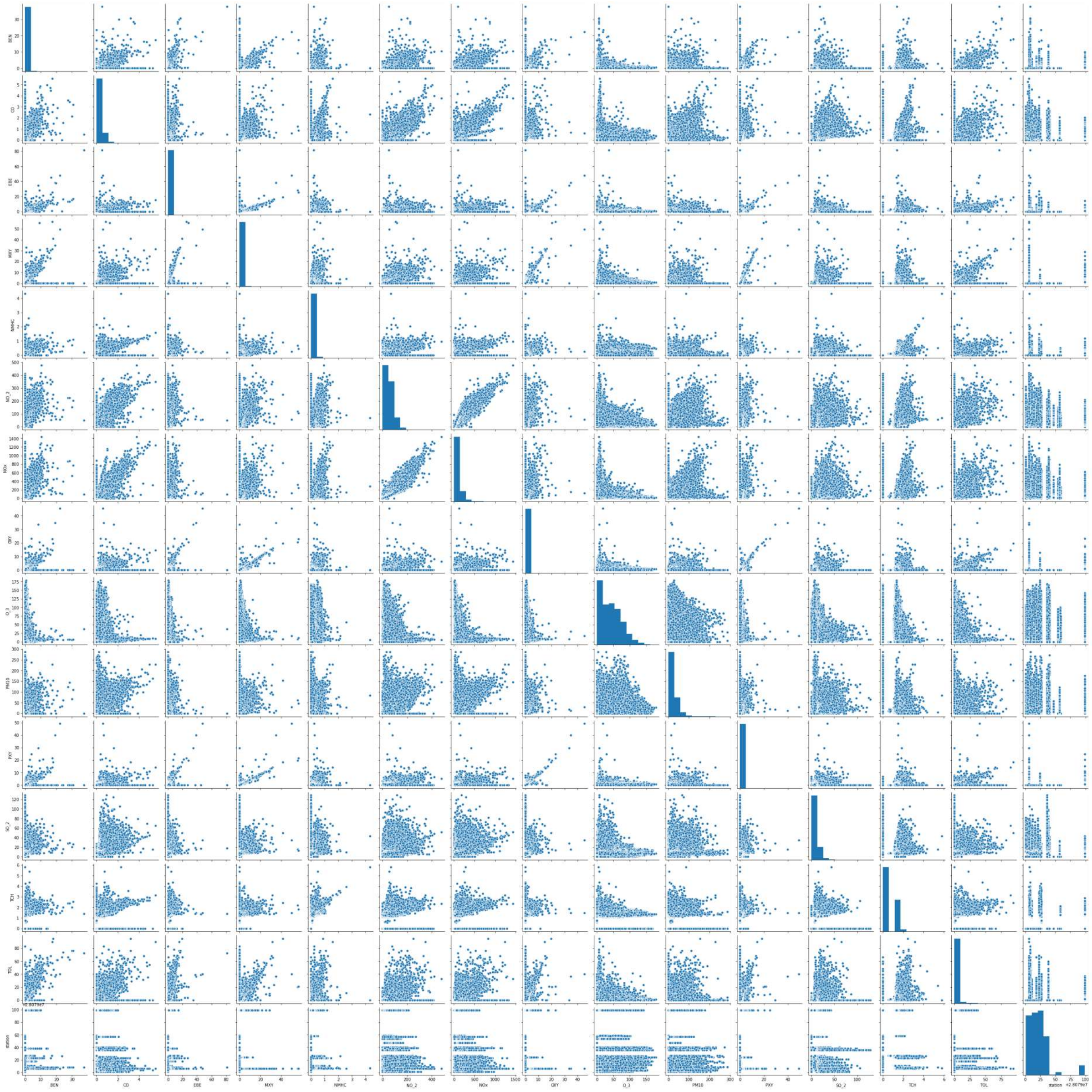
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_
3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [11]:

```
df2=df1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

In [12]:
```
sns.pairplot(df2)
```

Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x1ac70ed5f40>
```

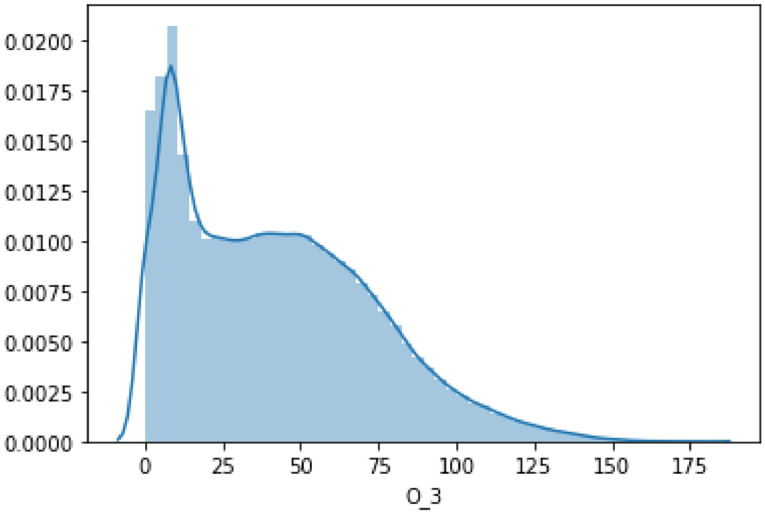


In [13]:

```
sns.distplot(df2['O_3'])
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ac2e33e7c0>
```

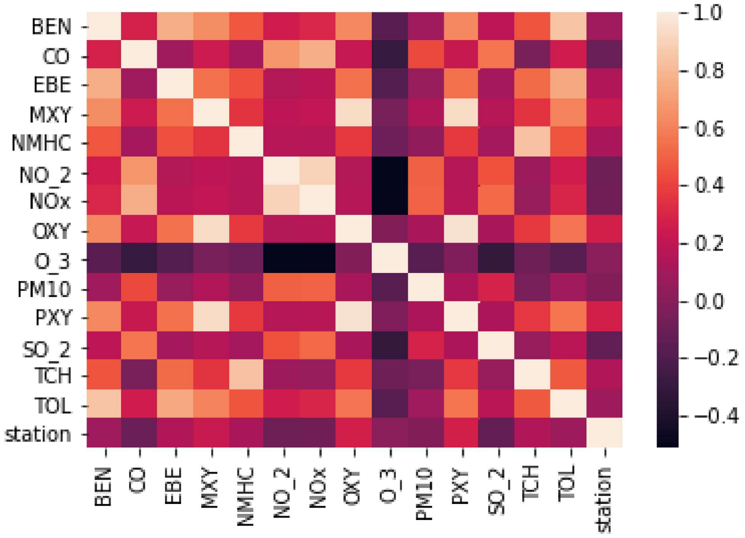


In [14]:

```
sns.heatmap(df2.corr())
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ac2e3ef160>
```



Linear Regression

In [15]:

```
x = df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY',
        'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y = df2['O_3']
```

In [16]:

```
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 [17]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[17]:
```
LinearRegression()
```

In [18]:
```
print(lr.intercept_)
```

```
66.95558974958072
```

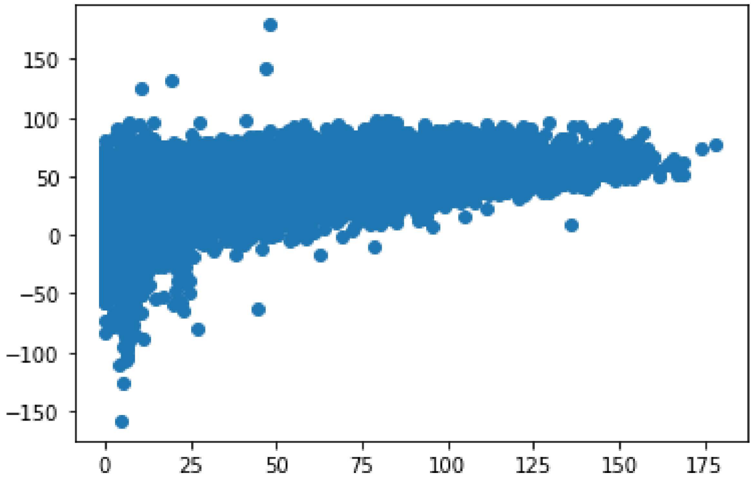
In [19]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-effecient'])
coeff
```

Out[19]:

	Co-effecient
BEN	3.544157
CO	17.404977
EBE	-7.545598
MXY	-2.135929
NMHC	29.425730
NO_2	-0.326915
NOx	-0.113924
OXY	10.360508
PM10	0.171713
PXY	-1.657101
SO_2	-0.583736
TCH	-4.477473
TOL	0.351467

In [20]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[20]:
```
<matplotlib.collections.PathCollection at 0x1ac2e702910>
```



In [21]:
```
print(lr.score(x_test,y_test))
```
```
0.3436569025600422
```

In [22]:
```
lr.score(x_train,y_train)
```
Out[22]:
```
0.3327286867692786
```

Ridge Lasso

In [23]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [24]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```
Out[24]:
```
0.3436540606390921
```

In [25]:
```
predict2=(rr.predict(x_test))
```

In [26]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[26]:
```
Lasso(alpha=10)
```

In [27]:
```
la.score(x_test,y_test)
```
Out[27]:
```
0.2891893530653368
```

Elastic Net regression

In [28]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[28]:
```
ElasticNet()
```

In [29]:
```
print(en.coef_)
```
```
[ 0.          0.3250155  -2.31668926  0.96889706  0.         -0.33518481
 -0.06882161  0.6987213   0.18557355  0.26107453 -0.3754829  -0.18163418
  0.05852214]
```

In [30]:
```
print(en.intercept_)
```
```
66.55187199777671
```

In [31]:
```
print(en.score(x_test,y_test))
```
```
0.3093740774829815
```

```
In [32]: print(en.score(x_train,y_train))
```

```
0.3011510507398939
```

Logistic Regression

```
In [33]: from sklearn.linear_model import LogisticRegression
```

```
In [34]: feature_matrix=df2.iloc[:,0:5]
         target_vector=df2.iloc[:,-1]
```

```
In [35]: from sklearn.preprocessing import StandardScaler
```

```
In [36]: fs=StandardScaler().fit_transform(feature_matrix)
```

```
In [37]: logr=LogisticRegression()
         logr.fit(fs,target_vector)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

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

```
In [38]: df2.shape
```

```
Out[38]: (215688, 15)
```

```
In [39]: observation=[[1,2,3,4,5]]
         predication = logr.predict(observation)
```

```
In [40]: print(predication)
```

```
[28079099]
```

```
In [41]: logr.classes_
```

```
Out[41]: array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
                28079009, 28079011, 28079012, 28079014, 28079016, 28079017,
                28079018, 28079019, 28079021, 28079022, 28079023, 28079024,
                28079025, 28079026, 28079027, 28079036, 28079038, 28079039,
                28079040, 28079047, 28079054, 28079057, 28079058, 28079059,
                28079099], dtype=int64)
```

```
In [42]: from sklearn.model_selection import train_test_split

         x_train,x_test,y_train,y_test = train_test_split(feature_matrix,target_vector,t
```

```
In [43]: print(logr.score(x_test,y_test))

         0.07248056624476486
```

```
In [44]: print(logr.score(x_train,y_train))

         0.0703797166530888
```

Conclusion

Ridge Regression is bestfit model

The Score x_test,y_test is 0.3436540606390921

```
In [ ]:
```

```
In [ ]:
```