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

In [155]:
```
df=pd.read_csv(r"C:\Users\Admin\Downloads\csvs_per_year\csvs_per_year\madrid_2005.csv")
df
```

Out[155]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 | PM25 | PXY | SO_ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 | 4.6 |
| **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 | 7.8 |
| **2** | 2005-11-01 01:00:00 | NaN | 0.40 | NaN | NaN | NaN | 46.119999 | 53.000000 | NaN | 30.469999 | 14.60 | NaN | NaN | 5.7 |
| **3** | 2005-11-01 01:00:00 | NaN | 0.42 | NaN | NaN | NaN | 37.220001 | 52.009998 | NaN | 21.379999 | 15.16 | NaN | NaN | 6.6 |
| **4** | 2005-11-01 01:00:00 | NaN | 0.57 | NaN | NaN | NaN | 32.160000 | 36.680000 | NaN | 33.410000 | 5.00 | NaN | NaN | 3.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **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 | 6.6 |
| **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 | 7.0 |
| **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 | 13.2 |
| **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 | 5.8 |
| **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 | 11.0 |

237000 rows × 17 columns

In [156]:

```
df1 = df.fillna(0)
df1
```

Out[156]:

| | date | BEN | CO | EBE | MXY | NMHC | NO_2 | NOx | OXY | O_3 | PM10 | PM25 | PXY | SO_ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2005-11-01 01:00:00 | 0.00 | 0.77 | 0.00 | 0.00 | 0.00 | 57.130001 | 128.699997 | 0.00 | 14.720000 | 14.91 | 10.65 | 0.00 | 4.6 |
| **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 | 0.00 | 1.59 | 7.8 |
| **2** | 2005-11-01 01:00:00 | 0.00 | 0.40 | 0.00 | 0.00 | 0.00 | 46.119999 | 53.000000 | 0.00 | 30.469999 | 14.60 | 0.00 | 0.00 | 5.7 |
| **3** | 2005-11-01 01:00:00 | 0.00 | 0.42 | 0.00 | 0.00 | 0.00 | 37.220001 | 52.009998 | 0.00 | 21.379999 | 15.16 | 0.00 | 0.00 | 6.6 |
| **4** | 2005-11-01 01:00:00 | 0.00 | 0.57 | 0.00 | 0.00 | 0.00 | 32.160000 | 36.680000 | 0.00 | 33.410000 | 5.00 | 0.00 | 0.00 | 3.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | .. |
| **236995** | 2006-01-01 00:00:00 | 1.08 | 0.36 | 1.01 | 0.00 | 0.11 | 21.990000 | 23.610001 | 0.00 | 43.349998 | 5.00 | 0.00 | 0.00 | 6.6 |
| **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 | 7.0 |
| **236997** | 2006-01-01 00:00:00 | 0.19 | 0.00 | 0.26 | 0.00 | 0.08 | 26.730000 | 30.809999 | 0.00 | 43.840000 | 4.31 | 2.93 | 0.00 | 13.2 |
| **236998** | 2006-01-01 00:00:00 | 0.14 | 0.00 | 1.00 | 0.00 | 0.06 | 13.770000 | 17.770000 | 0.00 | 0.000000 | 5.00 | 0.00 | 0.00 | 5.8 |
| **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 | 11.0 |

237000 rows × 17 columns

In [157]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237000 entries, 0 to 236999
Data columns (total 17 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     237000 non-null  object 
 1   BEN      70370 non-null   float64
 2   CO       217656 non-null  float64
 3   EBE      68955 non-null   float64
 4   MXY      32549 non-null   float64
 5   NMHC     92854 non-null   float64
 6   NO_2     235022 non-null  float64
 7   NOx      235049 non-null  float64
 8   OXY      32555 non-null   float64
 9   O_3      223162 non-null  float64
 10  PM10     232142 non-null  float64
 11  PM25     69407 non-null   float64
 12  PXY      32549 non-null   float64
 13  SO_2     235277 non-null  float64
 14  TCH      93076 non-null   float64
 15  TOL      70255 non-null   float64
 16  station  237000 non-null  int64  
dtypes: float64(15), int64(1), object(1)
memory usage: 30.7+ MB
```

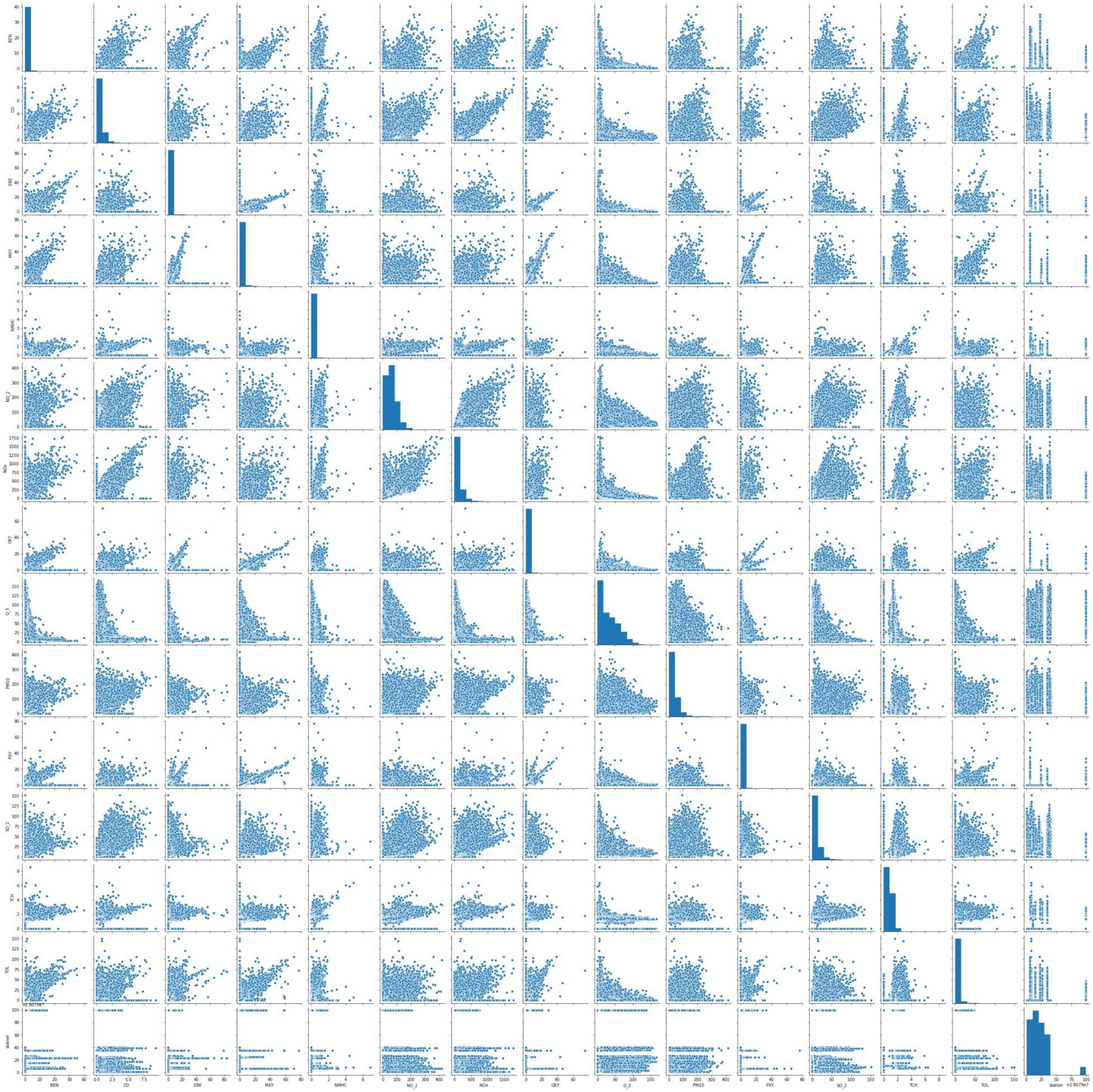
In [158]:
```
df.columns
```

Out[158]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PM25', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [159]:
```
df2=df1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

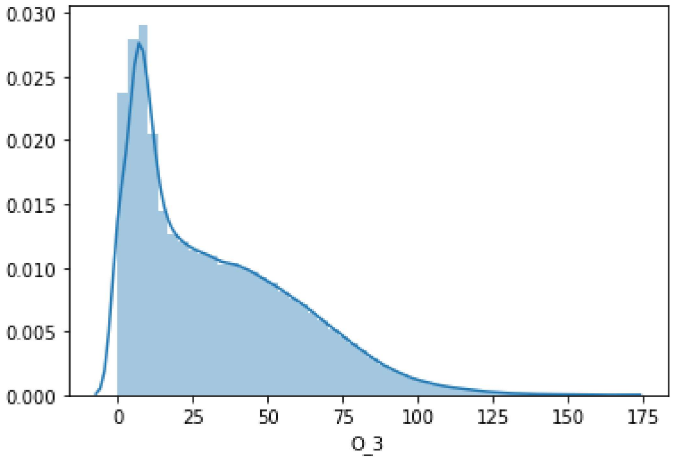
```
In [160]: sns.pairplot(df2)
```

```
Out[160]: <seaborn.axisgrid.PairGrid at 0x23344056f70>
```



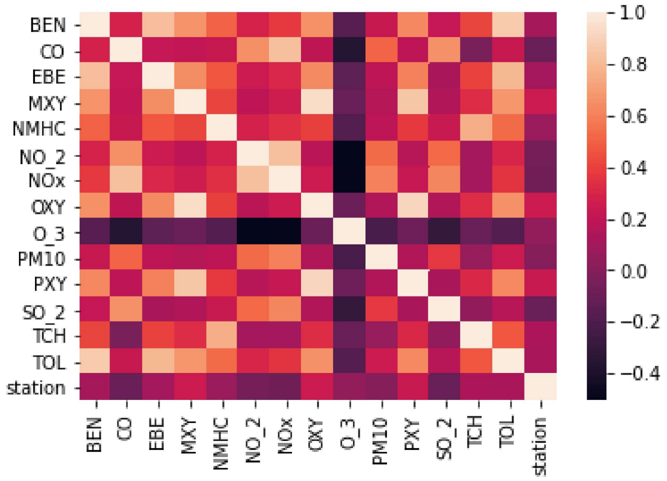
In [161]:
```
sns.distplot(df2['O_3'])
```

Out[161]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2343e148fa0>
```



In [162]:
```
sns.heatmap(df2.corr())
```

Out[162]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x234566f54c0>
```



# Linear Regression

In [163]:
```
x = df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y = df2['O_3']
```

In [164]:
```
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 [165]:
```
from sklearn.linear_model import LinearRegression

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

Out[165]:
```
LinearRegression()
```

In [166]:
```
print(lr.intercept_)
```

```
51.42294040923451
```

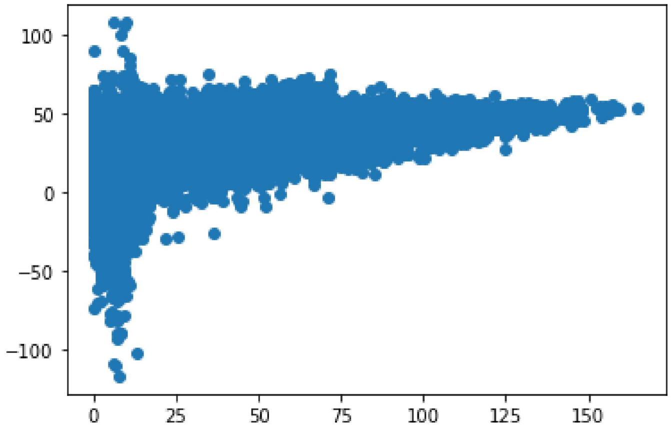
In [167]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-effecient'])
coeff
```

Out[167]:

| | Co-effecient |
|---|---|
| **BEN** | 1.552194 |
| **CO** | 4.035896 |
| **EBE** | -0.026706 |
| **MXY** | -0.382310 |
| **NMHC** | 7.055252 |
| **NO_2** | -0.225619 |
| **NOx** | -0.085837 |
| **OXY** | 0.856665 |
| **PM10** | 0.113306 |
| **PXY** | 0.538582 |
| **SO_2** | -0.032458 |
| **TCH** | -2.506577 |
| **TOL** | -0.277806 |

In [168]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[168]:
```
<matplotlib.collections.PathCollection at 0x2344f9a4b50>
```



In [169]:
```
print(lr.score(x_test,y_test))
```

```
0.28903819112398743
```

In [170]:
```
lr.score(x_train,y_train)
```

Out[170]:
```
0.28783184644650417
```

# Ridge Lasso

```
In [171]: from sklearn.linear_model import Ridge,Lasso
```

```
In [172]: rr=Ridge(alpha=10)
          rr.fit(x_train,y_train)
          rr.score(x_test,y_test)

Out[172]: 0.28903742721761205
```

```
In [173]: predict2=(rr.predict(x_test))
```

```
In [174]: la=Lasso(alpha=10)
          la.fit(x_train,y_train)

Out[174]: Lasso(alpha=10)
```

```
In [175]: la.score(x_test,y_test)

Out[175]: 0.28085040815849505
```

# Elastic Net regression

```
In [176]: from sklearn.linear_model import ElasticNet
          en=ElasticNet()
          en.fit(x_train,y_train)

Out[176]: ElasticNet()
```

```
In [177]: print(en.coef_)

          [ 0.00774178  0.          0.          0.14495589 -0.         -0.23862087
           -0.06696927  0.          0.11308583  0.0986593   0.03145642 -0.33690942
           -0.0502186 ]
```

```
In [178]: print(en.intercept_)

          51.13160182619739
```

```
In [179]: print(en.score(x_test,y_test))

          0.28285699603945946
```

```
In [180]: print(en.score(x_train,y_train))

          0.28216854923305
```

# Logistic Regression

```
In [181]: from sklearn.linear_model import LogisticRegression
```

```
In [182]: feature_matrix=df2.iloc[:,0:5]
          target_vector=df2.iloc[:,-1]
```

```
In [183]: from sklearn.preprocessing import StandardScaler
In [184]: fs=StandardScaler().fit_transform(feature_matrix)
```

```
In [185]: logr=LogisticRegression()
          logr.fit(fs,target_vector)

          C:\Users\Admin\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: Convergen
          ceWarning: 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://scikit-learn.org/st
          able/modules/preprocessing.html)
          Please also refer to the documentation for alternative solver options:
              https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://
          scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
            n_iter_i = _check_optimize_result(

Out[185]: LogisticRegression()
```

```
In [186]: df2.shape

Out[186]: (237000, 15)
```

```
In [187]: observation=[[1,2,3,4,5]]
          predication = logr.predict(observation)
```

```
In [188]: print(predication)

          [28079099]
```

```
In [189]: logr.classes_

Out[189]: array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
                 28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
                 28079017, 28079018, 28079019, 28079021, 28079022, 28079023,
                 28079024, 28079025, 28079026, 28079027, 28079035, 28079036,
                 28079038, 28079039, 28079040, 28079099], dtype=int64)
```

```
In [190]: from sklearn.model_selection import train_test_split

          x_train,x_test,y_train,y_test = train_test_split(feature_matrix,target_vector,test_size=0.30
```

```
In [191]: print(logr.score(x_test,y_test))

          0.10585091420534458
```

```
In [192]: print(logr.score(x_train,y_train))

          0.10491259795057263
```

# Conclusion

Linear Regression is bestfit model

The Score x_test,y_test is 0.28903819112398743 and x_train,y_train score is 0.28783184644650417

```
In [ ]:
```

```
In [ ]:
```