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

In [2]:
```
df=pd.read_csv(r"C:\Users\Admin\Downloads\csvs_per_year\csvs_per_year\madrid_2001.csv")
df
```

Out[2]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	
0	2001-08-01 01:00:00	NaN	0.37	NaN	NaN	NaN	58.400002	87.150002	NaN	34.529999	105.000000	NaN	6.34
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.73	8.11
2	2001-08-01 01:00:00	NaN	0.28	NaN	NaN	NaN	50.660000	61.380001	NaN	46.310001	100.099998	NaN	7.85
3	2001-08-01 01:00:00	NaN	0.47	NaN	NaN	NaN	69.790001	73.449997	NaN	40.650002	69.779999	NaN	6.46
4	2001-08-01 01:00:00	NaN	0.39	NaN	NaN	NaN	22.830000	24.799999	NaN	66.309998	75.180000	NaN	8.80
...	...	...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	NaN	NaN	NaN	73.000000	264.399994	NaN	5.200000	47.880001	NaN	39.91
217868	2001-04-01 00:00:00	5.20	0.69	4.56	NaN	0.13	71.080002	129.300003	NaN	13.460000	26.809999	NaN	13.45
217869	2001-04-01 00:00:00	0.49	1.09	NaN	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.770000	0.61	14.70
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	NaN	80.019997	197.000000	2.58	5.840000	37.889999	4.31	39.91
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.369999	4.95	27.34

217872 rows × 16 columns

In [3]:
```
df1 = df.fillna(0)
df1
```

Out[3]:

	date	BEN	CO	EBE	MXY	NMHC	NO_2	NOx	OXY	O_3	PM10	PXY	S
0	2001-08-01 01:00:00	0.00	0.37	0.00	0.00	0.00	58.400002	87.150002	0.00	34.529999	105.000000	0.00	6.34
1	2001-08-01 01:00:00	1.50	0.34	1.49	4.10	0.07	56.250000	75.169998	2.11	42.160000	100.599998	1.73	8.11
2	2001-08-01 01:00:00	0.00	0.28	0.00	0.00	0.00	50.660000	61.380001	0.00	46.310001	100.099998	0.00	7.85
3	2001-08-01 01:00:00	0.00	0.47	0.00	0.00	0.00	69.790001	73.449997	0.00	40.650002	69.779999	0.00	6.46
4	2001-08-01 01:00:00	0.00	0.39	0.00	0.00	0.00	22.830000	24.799999	0.00	66.309998	75.180000	0.00	8.80
...	...	...	...	...	...	...	...	...	...	...	...	...	
217867	2001-04-01 00:00:00	10.45	1.81	0.00	0.00	0.00	73.000000	264.399994	0.00	5.200000	47.880001	0.00	39.91
217868	2001-04-01 00:00:00	5.20	0.69	4.56	0.00	0.13	71.080002	129.300003	0.00	13.460000	26.809999	0.00	13.45
217869	2001-04-01 00:00:00	0.49	1.09	0.00	1.00	0.19	76.279999	128.399994	0.35	5.020000	40.770000	0.61	14.70
217870	2001-04-01 00:00:00	5.62	1.01	5.04	11.38	0.00	80.019997	197.000000	2.58	5.840000	37.889999	4.31	39.91
217871	2001-04-01 00:00:00	8.09	1.62	6.66	13.04	0.18	76.809998	206.300003	5.20	8.340000	35.369999	4.95	27.34

217872 rows × 16 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217872 entries, 0 to 217871
Data columns (total 16 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   date     217872 non-null  object 
 1   BEN      70389 non-null   float64
 2   CO       216341 non-null  float64
 3   EBE      57752 non-null   float64
 4   MXY      42753 non-null   float64
 5   NMHC     85719 non-null   float64
 6   NO_2     216331 non-null  float64
 7   NOx      216318 non-null  float64
 8   OXY      42856 non-null   float64
 9   O_3      216514 non-null  float64
 10  PM10     207776 non-null  float64
 11  PXY      42845 non-null   float64
 12  SO_2     216403 non-null  float64
 13  TCH      85797 non-null   float64
 14  TOL      70196 non-null   float64
 15  station  217872 non-null  int64  
dtypes: float64(14), int64(1), object(1)
memory usage: 26.6+ MB
```

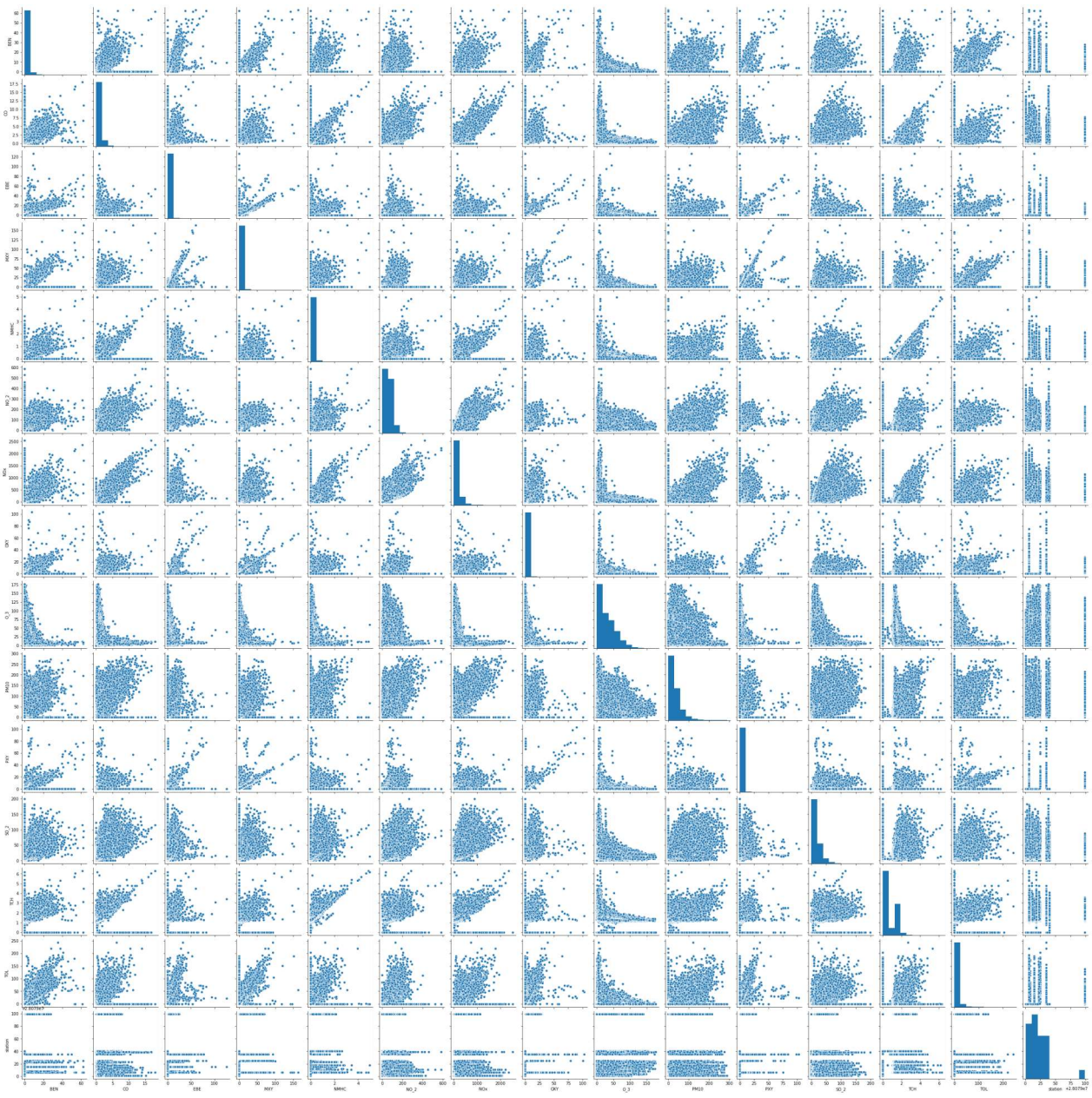
In [5]:
```
df.columns
```

Out[5]:
```
Index(['date', 'BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station'],
      dtype='object')
```

In [6]:
```
df2=df1[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY', 'O_3',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL', 'station']]
```

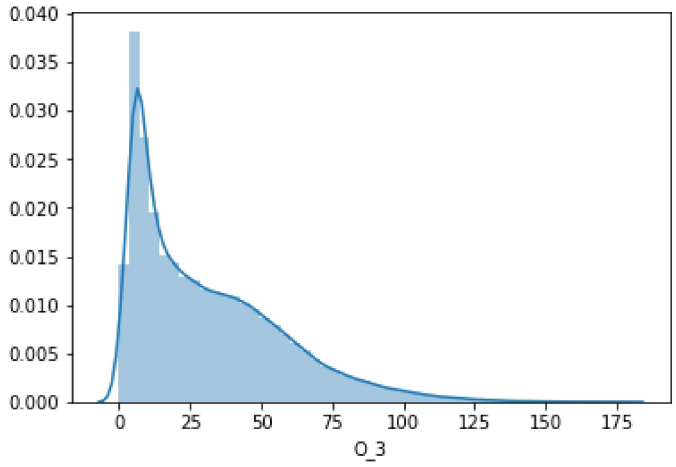
In [7]:
```
sns.pairplot(df2)
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x233784fbd30>
```



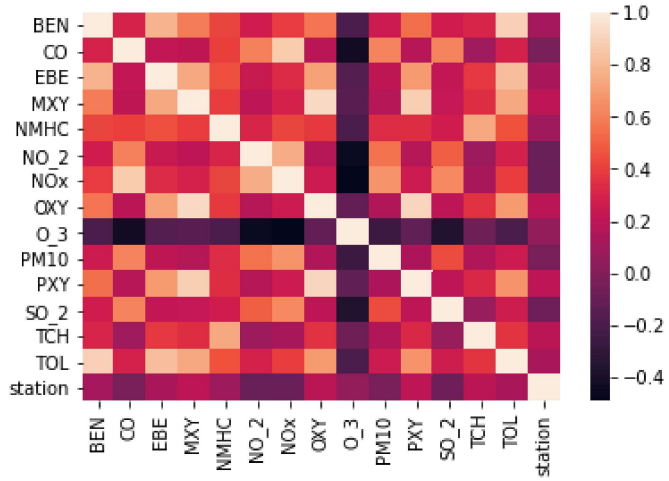
In [8]:
```
sns.distplot(df2['O_3'])
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x233163c0760>
```



In [9]:
```
sns.heatmap(df2.corr())
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x233165e1490>
```



Linear Regression

In [10]:
```
x = df2[['BEN', 'CO', 'EBE', 'MXY', 'NMHC', 'NO_2', 'NOx', 'OXY',
       'PM10', 'PXY', 'SO_2', 'TCH', 'TOL']]
y = df2['O_3']
```

In [11]:
```
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 [12]:
```
from sklearn.linear_model import LinearRegression

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

Out[12]:
```
LinearRegression()
```

In [13]:
```
print(lr.intercept_)
```

```
49.412324245490396
```

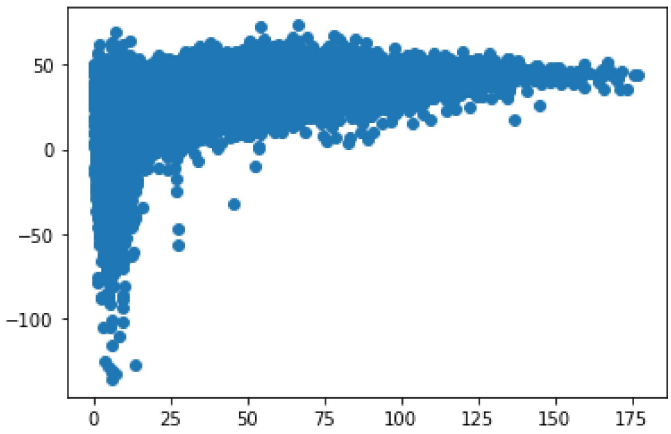
In [14]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-effecient'])
coeff
```

Out[14]:

	Co-effecient
BEN	-0.012127
CO	-0.506842
EBE	0.045555
MXY	0.087719
NMHC	10.574531
NO_2	-0.141696
NOx	-0.072500
OXY	0.440141
PM10	0.136818
PXY	-0.107741
SO_2	-0.138993
TCH	-2.495309
TOL	-0.082440

In [15]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x23316cf0a30>
```



In [16]:
```
print(lr.score(x_test,y_test))
```

```
0.2740199492493419
```

In [17]:
```
lr.score(x_train,y_train)
```

Out[17]:
```
0.27323510751160207
```

Ridge Lasso

In [18]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [19]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[19]:
```
0.274019071467575
```

In [20]:
```
predict2=(rr.predict(x_test))
```

In [21]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[21]:
```
Lasso(alpha=10)
```

In [22]:
```
la.score(x_test,y_test)
```

Out[22]:
```
0.26983570008735613
```

Elastic Net regression

In [23]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[23]:
```
ElasticNet()
```

In [25]:
```
print(en.coef_)
```

```
[-0.         -0.          0.          0.13582955  0.         -0.13885115
 -0.07246158  0.          0.13302468  0.         -0.13332664 -0.
 -0.04934335]
```

In [26]:
```
print(en.intercept_)
```

```
48.17709854091179
```

In [27]:
```
print(en.score(x_test,y_test))
```

```
0.27169261624993957
```

In [28]:
```
print(en.score(x_train,y_train))
```

```
0.27106254972196187
```

Logistic Regression

In [29]:
```
from sklearn.linear_model import LogisticRegression
```

In [30]:
```
feature_matrix=df2.iloc[:,0:5]
target_vector=df2.iloc[:,-1]
```

In [31]:
```
from sklearn.preprocessing import StandardScaler
```

In [32]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [33]:
```
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[33]:
```
LogisticRegression()
```

In [34]:
```
df2.shape
```

Out[34]:
```
(217872, 15)
```

In [35]:
```
observation=[[1,2,3,4,5]]
predication = logr.predict(observation)
```

In [36]:
```
print(predication)
```

```
[28079006]
```

In [37]:
```
logr.classes_
```

Out[37]:
```
array([28079001, 28079003, 28079004, 28079006, 28079007, 28079008,
       28079009, 28079011, 28079012, 28079014, 28079015, 28079016,
       28079017, 28079018, 28079019, 28079021, 28079022, 28079023,
       28079024, 28079025, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079099], dtype=int64)
```

In [38]:
```
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 [39]:
```
print(logr.score(x_test,y_test))
```

```
0.09228603775894251
```

In [40]:
```
print(logr.score(x_train,y_train))
```

```
0.09245951085174743
```

Conclusion

Linear Regression is bestfit model

Linear Regression is bestfit model for dataset madrid_2001. The Score x_test,y_test is 0.2740199492493419 and x_train,y_train score is 0.27323510751160207

In [ ]: