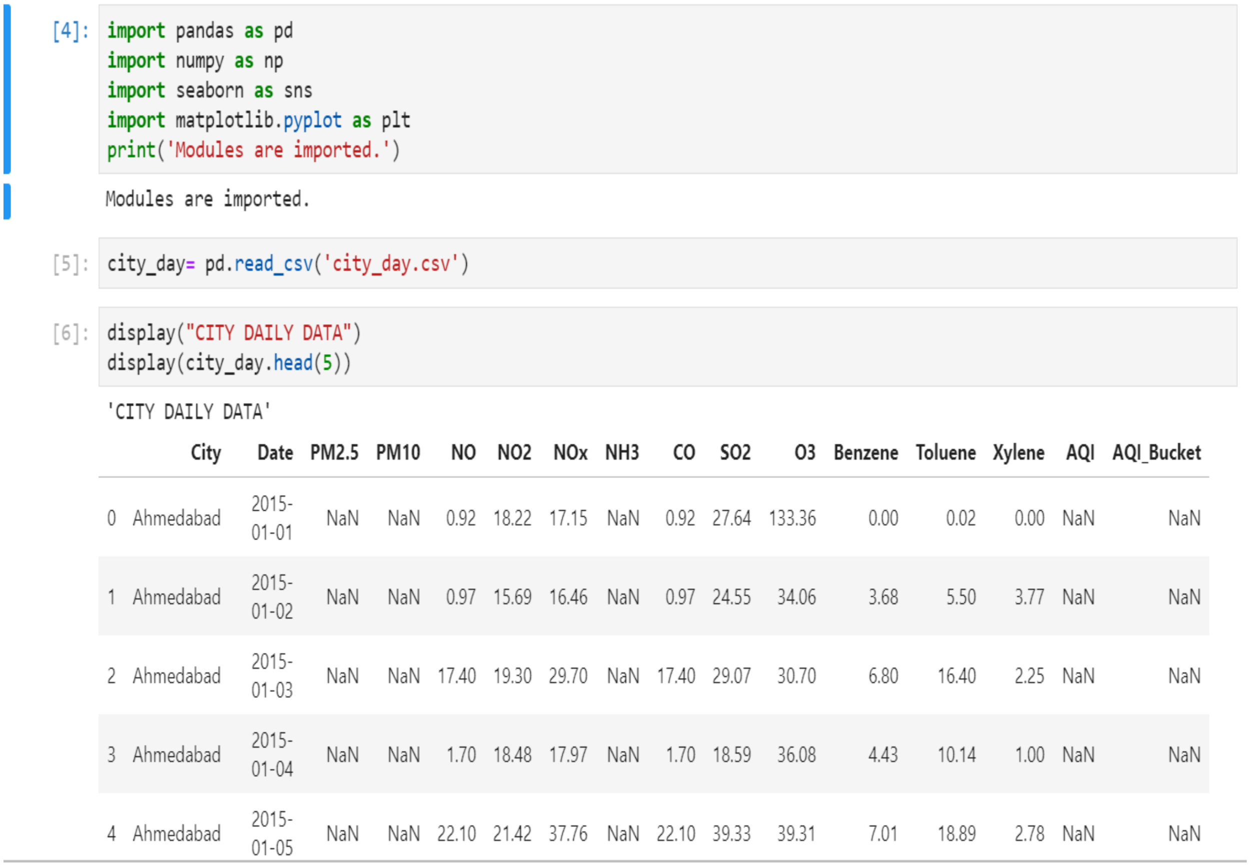
Analysis of Pollution Level in various cities during COVID-19 Lockdown

Importing modules



```
[4]: import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
     print('Modules are imported.')
```

```
Modules are imported.
```

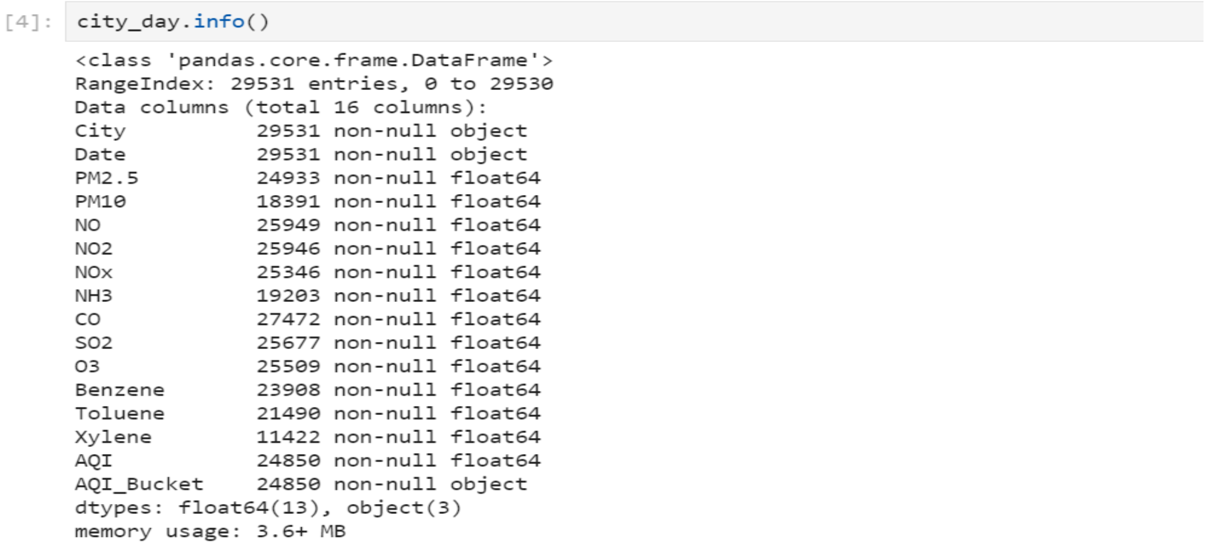
```
[5]: city_day= pd.read_csv('city_day.csv')
```

```
[6]: display("CITY DAILY DATA")
     display(city_day.head(5))
```

'CITY DAILY DATA'

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
0	Ahmedabad	2015-01-01	NaN	NaN	0.92	18.22	17.15	NaN	0.92	27.64	133.36	0.00	0.02	0.00	NaN	NaN
1	Ahmedabad	2015-01-02	NaN	NaN	0.97	15.69	16.46	NaN	0.97	24.55	34.06	3.68	5.50	3.77	NaN	NaN
2	Ahmedabad	2015-01-03	NaN	NaN	17.40	19.30	29.70	NaN	17.40	29.07	30.70	6.80	16.40	2.25	NaN	NaN
3	Ahmedabad	2015-01-04	NaN	NaN	1.70	18.48	17.97	NaN	1.70	18.59	36.08	4.43	10.14	1.00	NaN	NaN
4	Ahmedabad	2015-01-05	NaN	NaN	22.10	21.42	37.76	NaN	22.10	39.33	39.31	7.01	18.89	2.78	NaN	NaN

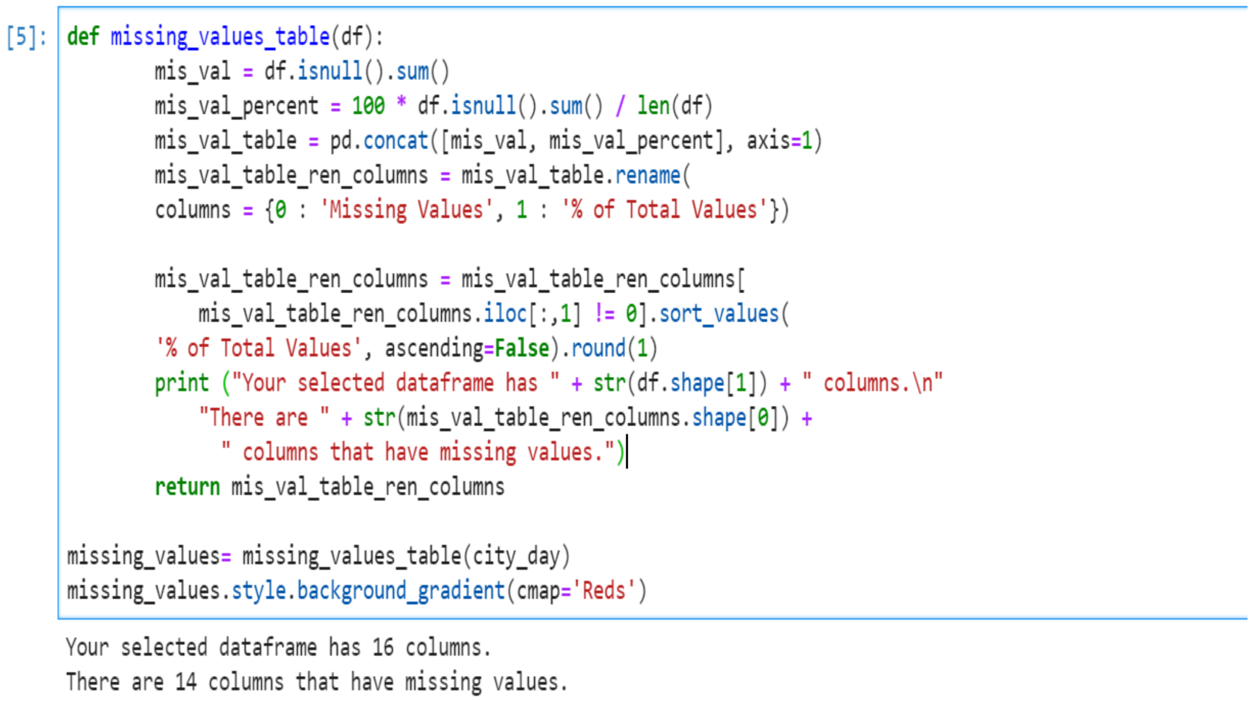
Enlisting datatypes



```
[4]: city_day.info()
```

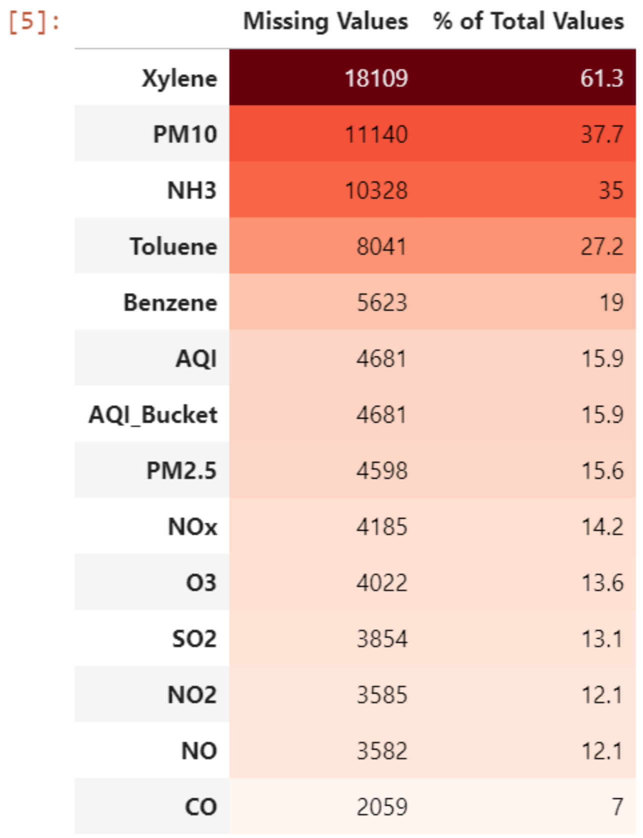
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29531 entries, 0 to 29530
Data columns (total 16 columns):
City          29531 non-null object
Date          29531 non-null object
PM2.5         24933 non-null float64
PM10          18391 non-null float64
NO            25949 non-null float64
NO2           25946 non-null float64
NOx           25346 non-null float64
NH3           19203 non-null float64
CO            27472 non-null float64
SO2           25677 non-null float64
O3            25509 non-null float64
Benzene       23908 non-null float64
Toluene       21490 non-null float64
Xylene        11422 non-null float64
AQI           24850 non-null float64
AQI_Bucket    24850 non-null object
dtypes: float64(13), object(3)
memory usage: 3.6+ MB
```

Missing value Treatment



```
[5]: def missing_values_table(df):
             mis_val = df.isnull().sum()
             mis_val_percent = 100 * df.isnull().sum() / len(df)
             mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
             mis_val_table_ren_columns = mis_val_table.rename(
             columns = {0 : 'Missing Values', 1 : '% of Total Values'})

             mis_val_table_ren_columns = mis_val_table_ren_columns[
                 mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
             '% of Total Values', ascending=False).round(1)
             print ("Your selected dataframe has " + str(df.shape[1]) + " columns.\n"
                 "There are " + str(mis_val_table_ren_columns.shape[0]) +
                   " columns that have missing values.")
             return mis_val_table_ren_columns

     missing_values= missing_values_table(city_day)
     missing_values.style.background_gradient(cmap='Reds')
```

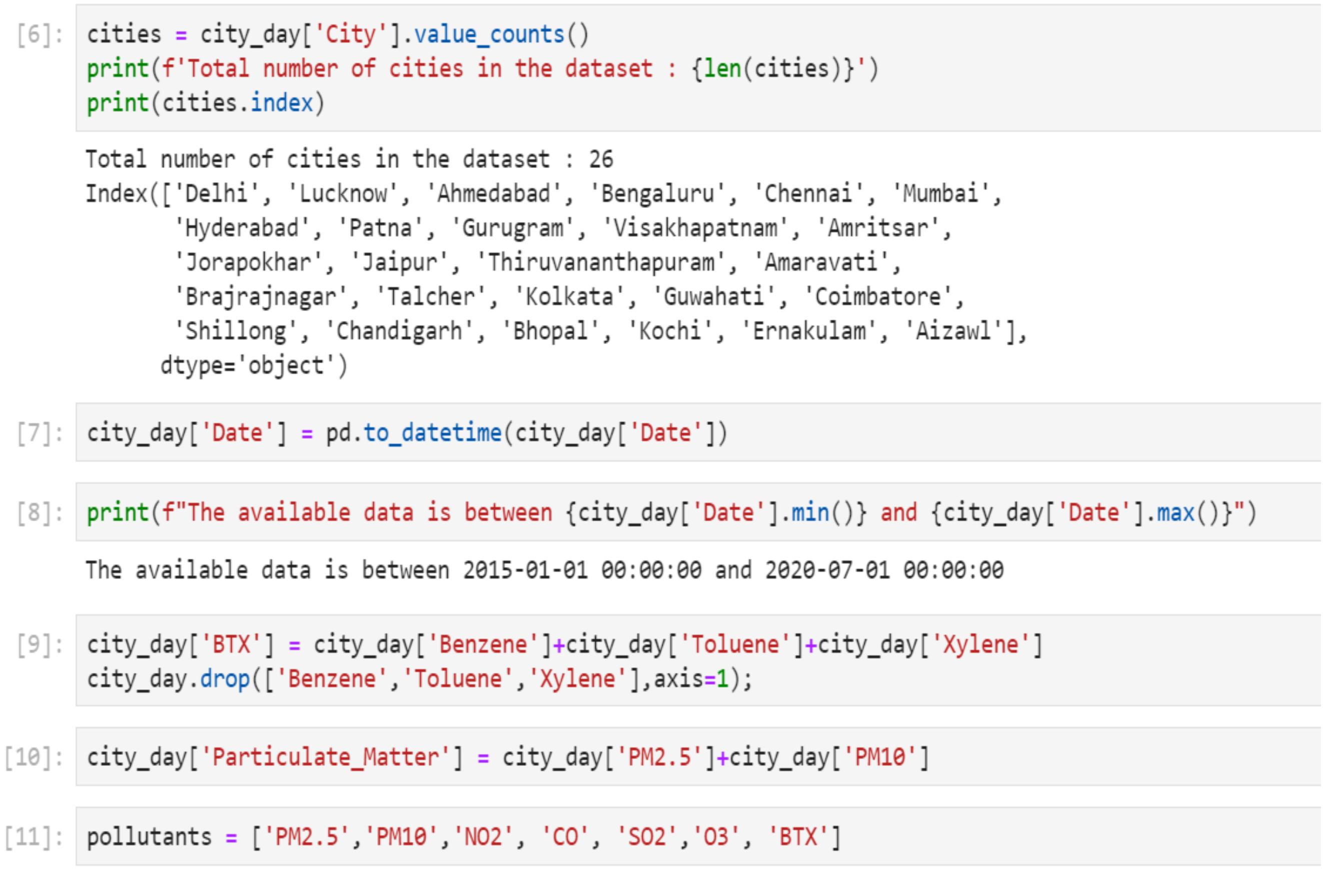
```
Your selected dataframe has 16 columns.
There are 14 columns that have missing values.
```



[5]:

	Missing Values	% of Total Values
Xylene	18109	61.3
PM10	11140	37.7
NH3	10328	35
Toluene	8041	27.2
Benzene	5623	19
AQI	4681	15.9
AQI_Bucket	4681	15.9
PM2.5	4598	15.6
NOx	4185	14.2
O3	4022	13.6
SO2	3854	13.1
NO2	3585	12.1
NO	3582	12.1
CO	2059	7

Displaying cities, range of dates and dropping redundant columns



```
[6]: cities = city_day['City'].value_counts()
     print(f'Total number of cities in the dataset : {len(cities)}')
     print(cities.index)
```

```
Total number of cities in the dataset : 26
Index(['Delhi', 'Lucknow', 'Ahmedabad', 'Bengaluru', 'Chennai', 'Mumbai',
       'Hyderabad', 'Patna', 'Gurugram', 'Visakhapatnam', 'Amritsar',
       'Jorapokhar', 'Jaipur', 'Thiruvananthapuram', 'Amaravati',
       'Brajrajnagar', 'Talcher', 'Kolkata', 'Guwahati', 'Coimbatore',
       'Shillong', 'Chandigarh', 'Bhopal', 'Kochi', 'Ernakulam', 'Aizawl'],
      dtype='object')
```

```
[7]: city_day['Date'] = pd.to_datetime(city_day['Date'])
```

```
[8]: print(f"The available data is between {city_day['Date'].min()} and {city_day['Date'].max()}")
```

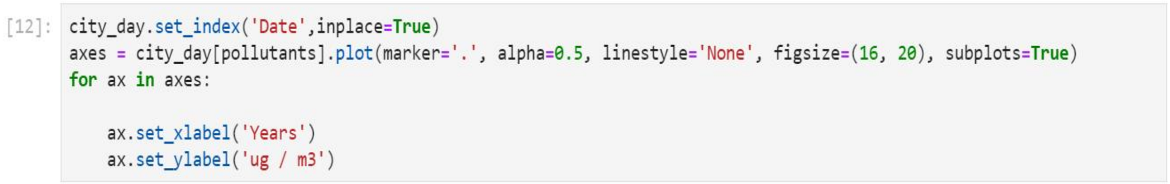
```
The available data is between 2015-01-01 00:00:00 and 2020-07-01 00:00:00
```

```
[9]: city_day['BTX'] = city_day['Benzene']+city_day['Toluene']+city_day['Xylene']
     city_day.drop(['Benzene','Toluene','Xylene'],axis=1);
```

```
[10]: city_day['Particulate_Matter'] = city_day['PM2.5']+city_day['PM10']
```

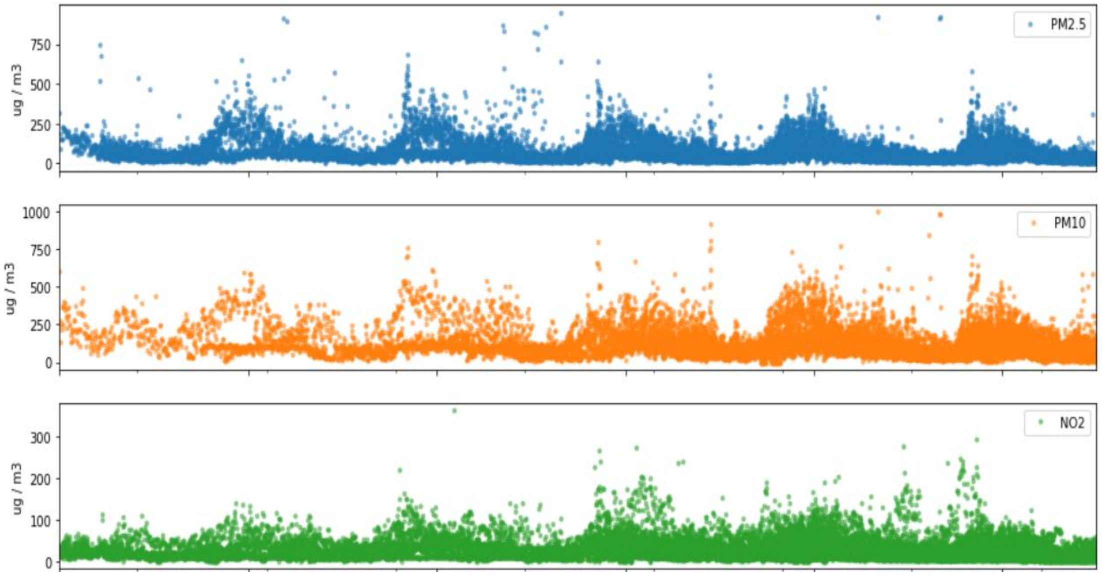
```
[11]: pollutants = ['PM2.5','PM10','NO2', 'CO', 'SO2','O3', 'BTX']
```

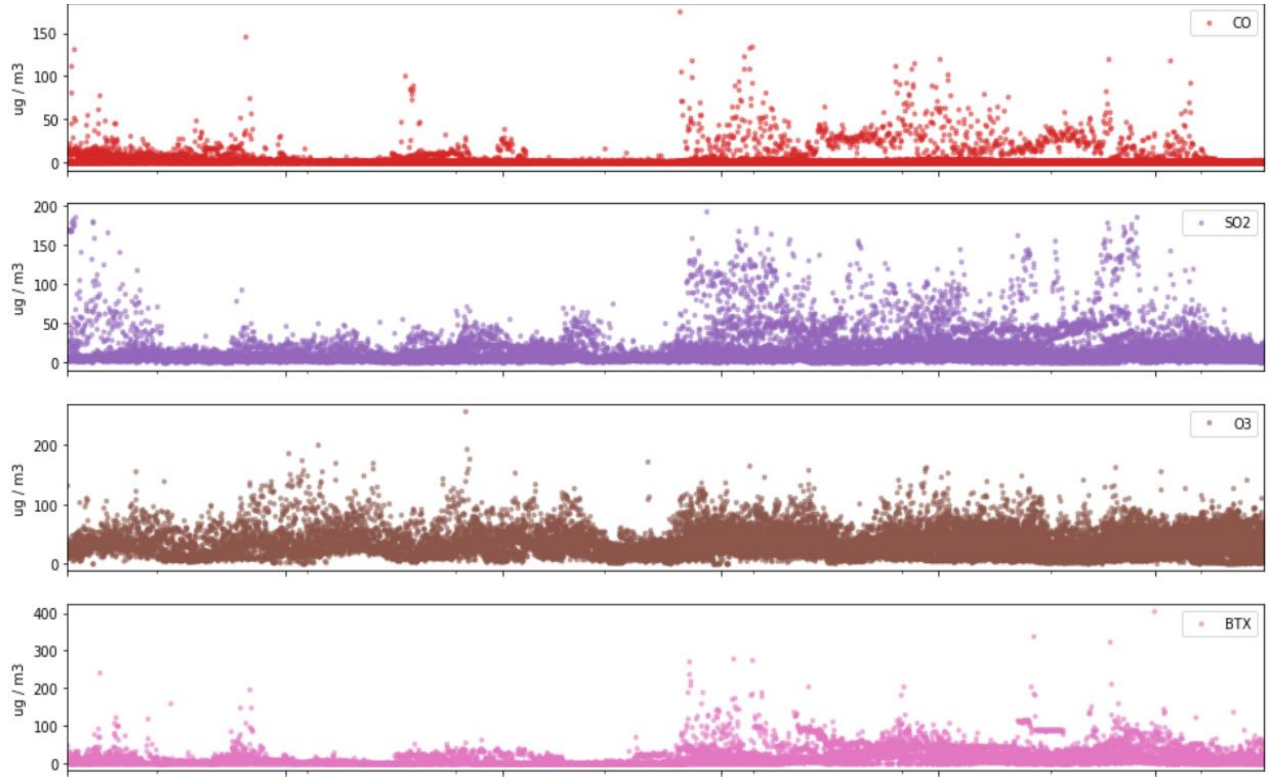
Displaying individual pollutant levels



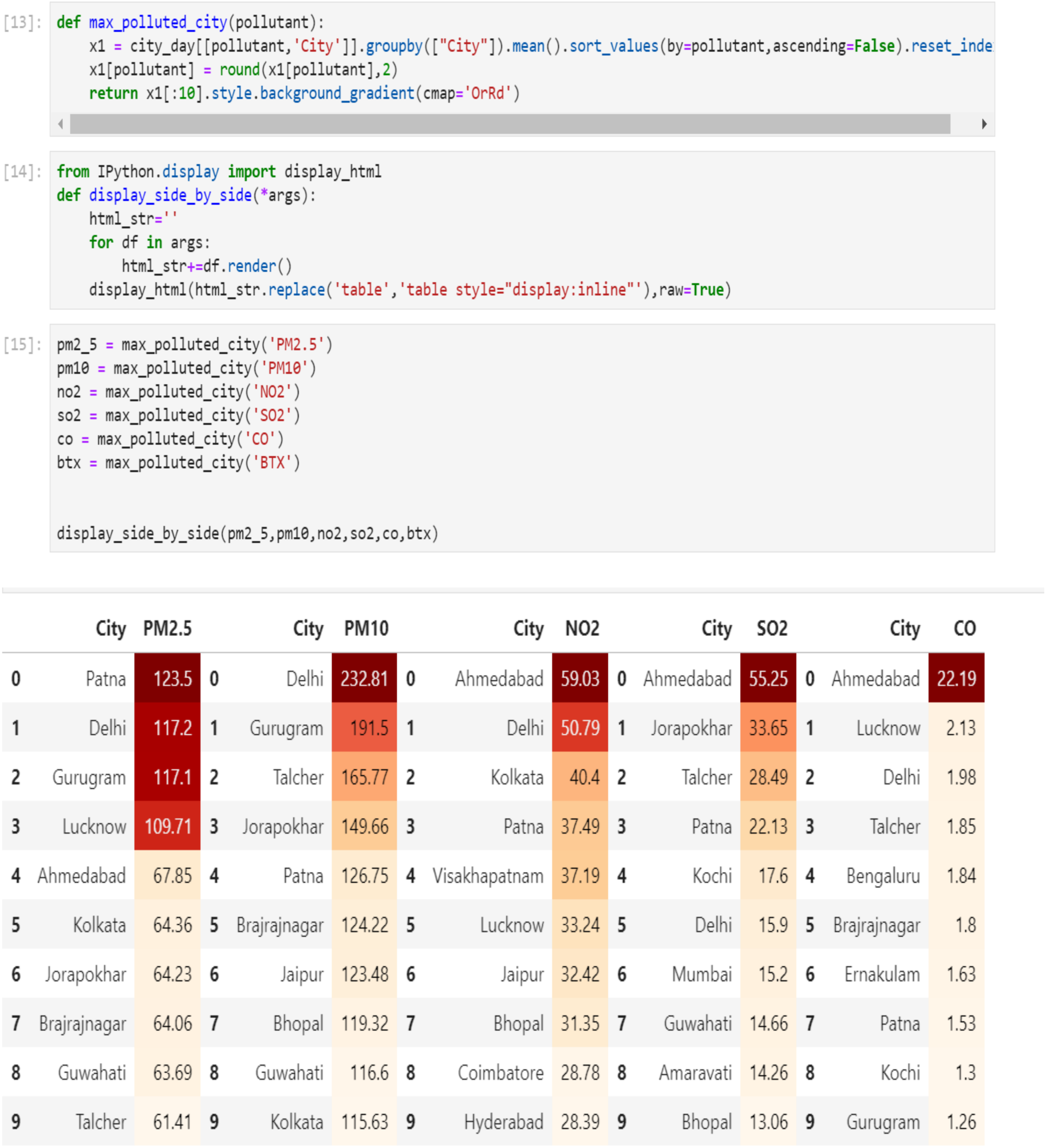
[12]:

```
city_day.set_index('Date',inplace=True)
axes = city_day[pollutants].plot(marker='.', alpha=0.5, linestyle='None', figsize=(16, 20), subplots=True)
for ax in axes:

    ax.set_xlabel('Years')
    ax.set_ylabel('ug / m3')
```





Finding the maximum polluted city according to pollutants

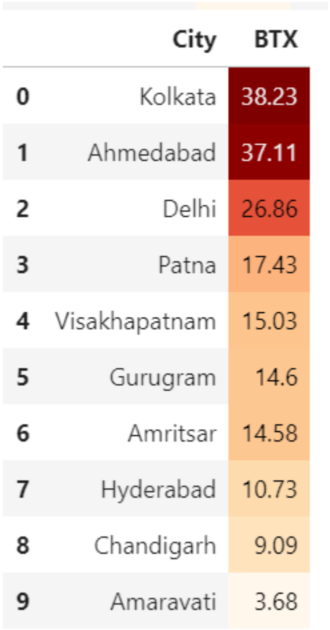


```
[13]: def max_polluted_city(pollutant):
          x1 = city_day[[pollutant,'City']].groupby(["City"]).mean().sort_values(by=pollutant,ascending=False).reset_inde
          x1[pollutant] = round(x1[pollutant],2)
          return x1[:10].style.background_gradient(cmap='OrRd')
```

```
[14]: from IPython.display import display_html
      def display_side_by_side(*args):
          html_str=''
          for df in args:
              html_str+=df.render()
          display_html(html_str.replace('table','table style="display:inline"'),raw=True)
```

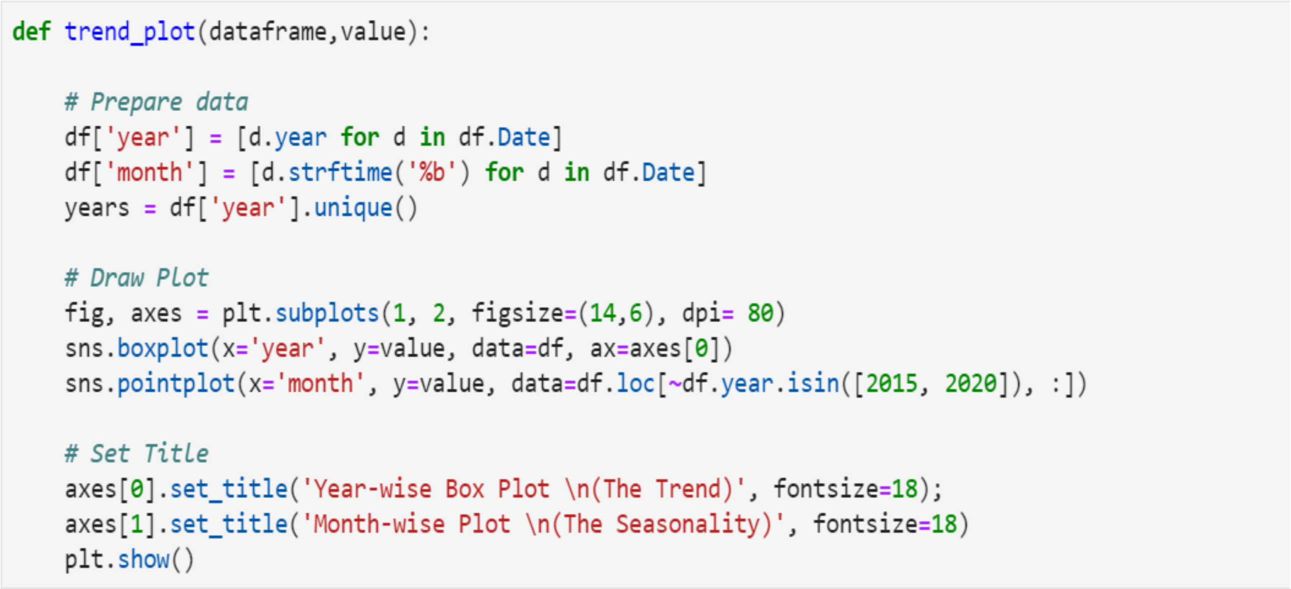
```
[15]: pm2_5 = max_polluted_city('PM2.5')
      pm10 = max_polluted_city('PM10')
      no2 = max_polluted_city('NO2')
      so2 = max_polluted_city('SO2')
      co = max_polluted_city('CO')
      btx = max_polluted_city('BTX')


      display_side_by_side(pm2_5,pm10,no2,so2,co,btx)
```

	City	PM2.5		City	PM10		City	NO2		City	SO2		City	CO
0	Patna	123.5	0	Delhi	232.81	0	Ahmedabad	59.03	0	Ahmedabad	55.25	0	Ahmedabad	22.19
1	Delhi	117.2	1	Gurugram	191.5	1	Delhi	50.79	1	Jorapokhar	33.65	1	Lucknow	2.13
2	Gurugram	117.1	2	Talcher	165.77	2	Kolkata	40.4	2	Talcher	28.49	2	Delhi	1.98
3	Lucknow	109.71	3	Jorapokhar	149.66	3	Patna	37.49	3	Patna	22.13	3	Talcher	1.85
4	Ahmedabad	67.85	4	Patna	126.75	4	Visakhapatnam	37.19	4	Kochi	17.6	4	Bengaluru	1.84
5	Kolkata	64.36	5	Brajrajnagar	124.22	5	Lucknow	33.24	5	Delhi	15.9	5	Brajrajnagar	1.8
6	Jorapokhar	64.23	6	Jaipur	123.48	6	Jaipur	32.42	6	Mumbai	15.2	6	Ernakulam	1.63
7	Brajrajnagar	64.06	7	Bhopal	119.32	7	Bhopal	31.35	7	Guwahati	14.66	7	Patna	1.53
8	Guwahati	63.69	8	Guwahati	116.6	8	Coimbatore	28.78	8	Amaravati	14.26	8	Kochi	1.3
9	Talcher	61.41	9	Kolkata	115.63	9	Hyderabad	28.39	9	Bhopal	13.06	9	Gurugram	1.26

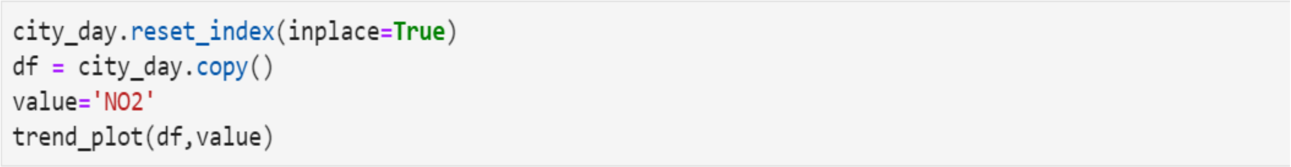


	City	BTX
0	Kolkata	38.23
1	Ahmedabad	37.11
2	Delhi	26.86
3	Patna	17.43
4	Visakhapatnam	15.03
5	Gurugram	14.6
6	Amritsar	14.58
7	Hyderabad	10.73
8	Chandigarh	9.09
9	Amaravati	3.68

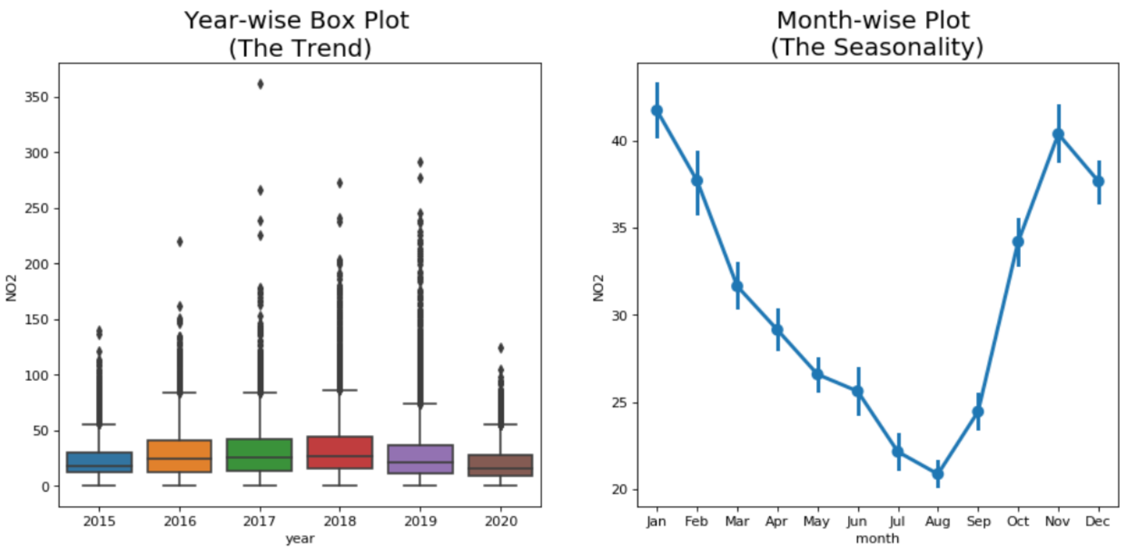
Trend plots:



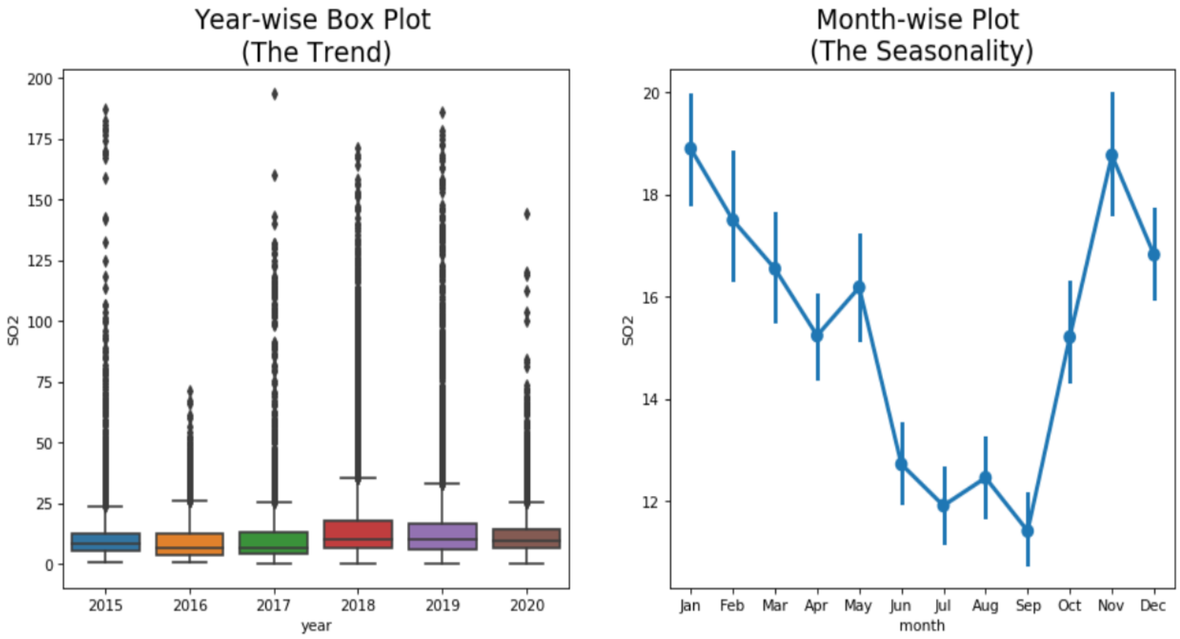
```python
def trend_plot(dataframe,value):

    # Prepare data
    df['year'] = [d.year for d in df.Date]
    df['month'] = [d.strftime('%b') for d in df.Date]
    years = df['year'].unique()

    # Draw Plot
    fig, axes = plt.subplots(1, 2, figsize=(14,6), dpi= 80)
    sns.boxplot(x='year', y=value, data=df, ax=axes[0])
    sns.pointplot(x='month', y=value, data=df.loc[~df.year.isin([2015, 2020]), :])

    # Set Title
    axes[0].set_title('Year-wise Box Plot \n(The Trend)', fontsize=18);
    axes[1].set_title('Month-wise Plot \n(The Seasonality)', fontsize=18)
    plt.show()
```



```python
city_day.reset_index(inplace=True)
df = city_day.copy()
value='NO2'
trend_plot(df,value)
```



```
[19]: df = city_day.copy()
      value='SO2'
      trend_plot(df,value)
```


Year-wise Box Plot
(The Trend)
SO2
year
Month-wise Plot
(The Seasonality)
SO2
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
month