UE20CS312
DATA ANALYTICS

PROJECT REPORT

ANALYSIS AND FORECASTING OF AIR QUALITY INDEX [AQI] IN INDIA

BY

PES1UG20CS134 DIVIJA L

PES1UG20CS150 GAURAV DNYANESH MAHAJAN

PREFACE

The daily air quality is reported using the Air Quality Index (AQI). It informs you of the cleanliness and pollution levels of the air as well as any potential health risks. Ground-level ozone, particle pollution (also known as particulate matter like PM10, PM2.5), carbon monoxide, sulphur dioxide, and nitrogen dioxide are the five main air pollutants that the EPA (Environmental Protection Agency) calculates the AQI for.

To safeguard the public's health, the EPA has established national air quality guidelines for each of these contaminants. The two pollutants that represent the biggest damage to human health in this country are ground-level ozone and airborne particles.

OBJECTIVE

A brief analysis of COVID-19's impact on India's AQI and Time Series Forecasting with SARIMA.

DATASET

Air Quality Data in India (2015 - 2020)

The dataset contains air quality data and AQI (Air Quality Index) at hourly and daily level of various stations across multiple cities in India.

The data has been made publicly available by the Central Pollution Control Board: https://cpcb.nic.in/ which is the official portal of Government of India. They also have a real-time monitoring app: https://app.cpcbccr.com/AQI_India/

This data is a cleaner version of the Historical Daily Ambient Air Quality Data released by the Ministry of Environment and Forests and Central Pollution Control Board of India under the National Data Sharing and Accessibility Policy (NDSAP).

The dataset contains the following features:

- **DATE**: Date on which the observations are recorded.
- **PM2.5** - the most harmful, a size of 2.5 microns, sources are industries, vehicles, power plants, crop and garbage burning, and diesel generators.
- **PM10** - Coarse (bigger) particles, can irritate your eyes, nose, and throat; sources are dust from roads, farms, dry riverbeds, construction sites, and mines.
- **NO, NO2, NOx** - High levels of nitrogen dioxide are also harmful to vegetation—damaging foliage, decreasing growth or reducing crop yields. Nitrogen dioxide can fade and discolor furnishings and fabrics, reduce visibility, and react with surfaces. Vehicular exhausts are the primary sources.
- **NH3**- NH3 plays a significant role in the formation of atmospheric particulate matter, visibility degradation and atmospheric deposition of nitrogen to sensitive ecosystems.
- **CO**- Carbon monoxide (CO) is a clear, odorless gas. Smoke and exhaust fumes often contain carbon monoxide. Carbon monoxide is a common air pollutant.
- **SO2**- Sulfur dioxide (SO_2) is a gaseous air pollutant composed of sulfur and oxygen. SO_2 forms when sulfur-containing fuel such as coal, oil, or diesel is burned. The main sources of sulfur dioxide emissions are from fossil fuel combustion and natural volcanic activity
- **O3** - Ground-level ozone forms when nitrogen oxides and volatile organic compounds react with each other in sunlight and hot temperatures. This pollution comes from vehicles, industry, and other sources and contributes to smog formation.
- **Benzene**- The benzene in indoor air comes from products that contain benzene such as glues, paints, furniture wax, and detergents. The air around hazardous waste sites or gas stations can contain higher levels of benzene than in other areas.
- **Toluene** - Motor vehicle emissions are the main source of toluene in the urban air environment, although evaporative losses from fuel storage facilities and service stations, as well as the use of toluene-based solvents and thinners are other contributors.
- **Xylene** - Motor vehicle emissions are the predominant source of xylene in the urban air environment. Evaporation from petroleum fuels storage facilities and service stations, and the use of products containing xylene-based solvents and thinners, are other ways xylene enters the air environment.
- **AQI**
- **AQI_Bucket** – Categorizes AQI as moderate, severe, high, low, etc.

METHODOLOGY

EDA AND VISUALISATIONS

IMPORTING AND INSTALLING PACKAGES:

```
# import the necessary libraries
import numpy as np
import pandas as pd
import os

# Visualisation libraries
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
!pip install pycountry
import pycountry
import plotly.express as px
from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
import plotly.offline as py
from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot
!pip install chart_studio
import chart_studio.plotly as py
import cufflinks
cufflinks.go_offline()
cufflinks.set_config_file(world_readable=True, theme='pearl')
#py.init_notebook_mode(connected=True)

#Geographical Plotting
import folium
from folium import Choropleth, Circle, Marker
from folium import plugins
from folium.plugins import HeatMap, MarkerCluster

#Racing Bar Chart
!pip install bar_chart_race
import bar_chart_race as bcr
from IPython.display import HTML
```

```
# Increase the default plot size and set the color scheme
plt.rcParams['figure.figsize'] = 8, 5
plt.style.use("fivethirtyeight")# for pretty graphs

# Disable warnings
import warnings
warnings.filterwarnings('ignore')
```

IMPORTING THE DATASET

```
[ ]  city_day = pd.read_csv('/content/city_day.csv')
     city_hour = pd.read_csv('/content/city_hour.csv',on_bad_lines='skip')
     station = pd.read_csv('/content/stations.csv')
     station_day = pd.read_csv('/content/station_day.csv')
     station_hour = pd.read_csv('/content/station_hour.csv')
     cities_db = pd.read_csv('/content/Indian Cities Database.csv')
```

BASIC DISPLAYING OF DATASETS:

```
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

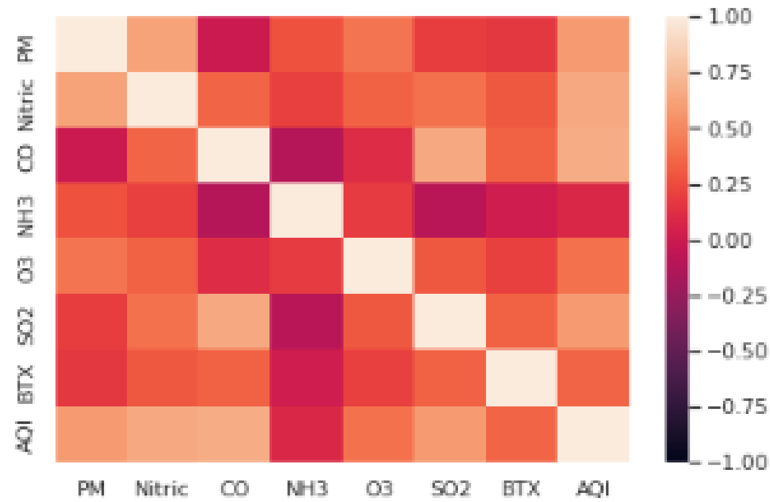
```
city_day.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29531 entries, 0 to 29530
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   City        29531 non-null  object 
 1   Date        29531 non-null  object 
 2   PM2.5       24933 non-null  float64
 3   PM10        18391 non-null  float64
 4   NO          25949 non-null  float64
 5   NO2         25946 non-null  float64
 6   NOx         25346 non-null  float64
 7   NH3         19203 non-null  float64
 8   CO          27472 non-null  float64
 9   SO2         25677 non-null  float64
 10  O3          25509 non-null  float64
 11  Benzene     23908 non-null  float64
 12  Toluene     21490 non-null  float64
 13  Xylene      11422 non-null  float64
 14  AQI         24850 non-null  float64
 15  AQI_Bucket  24850 non-null  object 
dtypes: float64(13), object(3)
memory usage: 3.6+ MB
```

CORRELATION MATRIX WITH SNS HEAT MAP:

```
[ ]  #plotting the correlation matrix with sns heatmap
     corr_matrix = cities_group_ym.iloc[:,3:].corr()
     print(corr_matrix)
     fig = plt.figure(figsize = (6, 4))
     sns.heatmap(corr_matrix, vmin=-1, vmax=1)
     plt.show()
```

```
              PM    Nitric        CO       NH3        O3       SO2       BTX  \
PM      1.000000  0.626912 -0.001708  0.273868  0.414142  0.194107  0.170518
Nitric  0.626912  1.000000  0.351381  0.205231  0.336590  0.398588  0.297628
CO     -0.001708  0.351381  1.000000 -0.107090  0.112073  0.658116  0.342239
NH3     0.273868  0.205231 -0.107090  1.000000  0.184410 -0.089157  0.016164
O3      0.414142  0.336590  0.112073  0.184410  1.000000  0.298710  0.207562
SO2     0.194107  0.398588  0.658116 -0.089157  0.298710  1.000000  0.339907
BTX     0.170518  0.297628  0.342239  0.016164  0.207562  0.339907  1.000000
AQI     0.589689  0.660347  0.672568  0.088668  0.404758  0.591202  0.348172

             AQI
PM      0.589689
Nitric  0.660347
CO      0.672568
NH3     0.088668
O3      0.404758
SO2     0.591202
BTX     0.348172
AQI     1.000000
```



INFERENCE DRAWN:

We see that BTX has the lowest correlation with AQI- which is perfectly in sync with the AQI calculation formula. The air quality index is composed of 8 pollutants ((PM10, PM2.5, NO2, SO2, CO, O3, NH3, and Pb), but does not directly account for BTX.

DETERMINING MISSING VALUES IN TERMS OF THEIR PROPORTIONS (IN TERMS OF PERCENTAGE)

```
[ ]  def missing_values_table(df):
             # Total missing values
             mis_val = df.isnull().sum()

             # Percentage of missing values
             mis_val_percent = 100 * df.isnull().sum() / len(df)

             # Make a table with the results
             mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)

             # Rename the columns
             mis_val_table_ren_columns = mis_val_table.rename(
             columns = {0 : 'Missing Values', 1 : '% of Total Values'})

             # Sort the table by percentage of missing descending
             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 some summary information
             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 the dataframe with missing information
             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.
```

	Missing Values	% of Total Values
Xylene	18109	61.300000
PM10	11140	37.700000
NH3	10328	35.000000
Toluene	8041	27.200000
Benzene	5623	19.000000
AQI	4681	15.900000
AQI_Bucket	4681	15.900000
PM2.5	4598	15.600000
NOx	4185	14.200000
O3	4022	13.600000
SO2	3854	13.100000
NO2	3585	12.100000
NO	3582	12.100000
CO	2059	7.000000

GOING THROUGH THE VARIOUS CITIES IN OUR DATASET:

```
[ ]  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(['Ahmedabad', 'Delhi', 'Mumbai', 'Bengaluru', 'Lucknow', 'Chennai',
       'Hyderabad', 'Patna', 'Gurugram', 'Visakhapatnam', 'Amritsar',
       'Jorapokhar', 'Jaipur', 'Thiruvananthapuram', 'Amaravati',
       'Brajrajnagar', 'Talcher', 'Kolkata', 'Guwahati', 'Coimbatore',
       'Shillong', 'Chandigarh', 'Bhopal', 'Ernakulam', 'Kochi', 'Aizawl'],
      dtype='object')
```

DATA TYPE CONVERSION:

Since the 'Date' attribute in the dataset is of ***string type***, we converted it to ***Datetime type.***

```
[ ]  # Convert string to datetime64
     city_day['Date'] = pd.to_datetime(city_day['Date'])
     #city_day.set_index('Date',inplace=True)
```

```
[ ]  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
```

TACKLING MISSING VALUES:

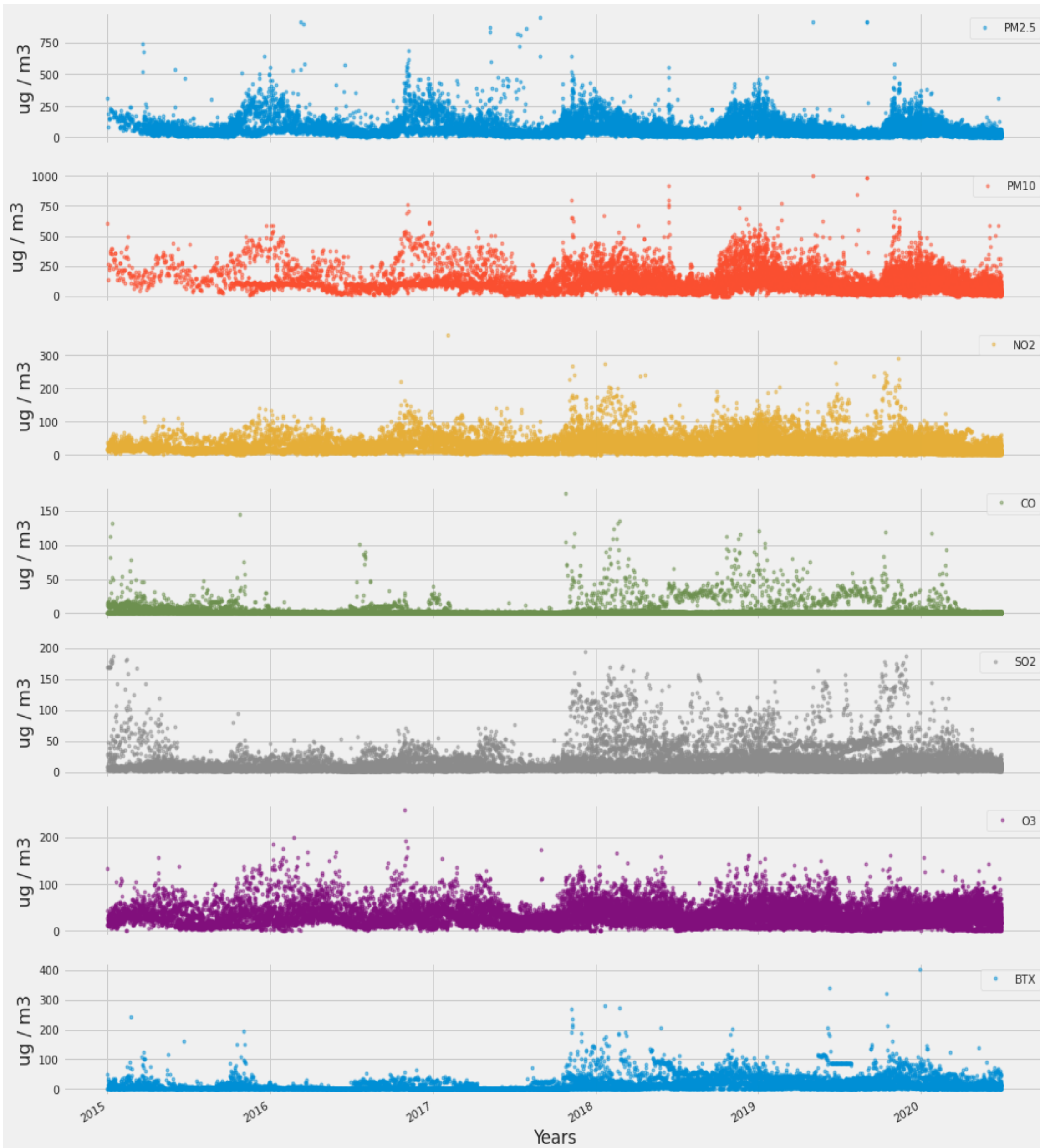
Since we observed the missing values were high in **Benzene, Toluene and Xylene**, we combined these three attributes to **BTX attribute**.

As well as, combined **PM2.5 AND PM10** into a **Particulate_Matter** attribute.

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

     city_day['Particulate_Matter'] = city_day['PM2.5']+city_day['PM10']

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

SCATTER PLOT OF VARIOUS POLLUTANTS QUANTITY VS YEARS

```python
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')
```



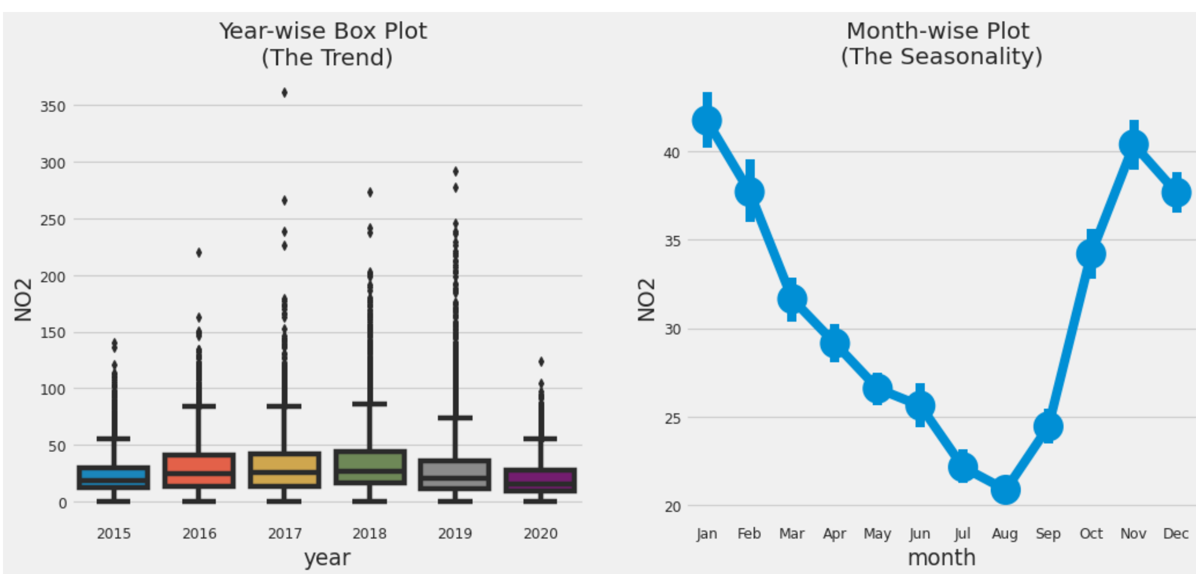
A YEARLY AND A MONTHLY BOX PLOT TREND FOR EVERY POLLUTANT

For better understanding, we have used Box plots for Yearly analysis that tells us about the trend component and also Box plots for Monthly analysis tells us about the seasonality of pollutants.

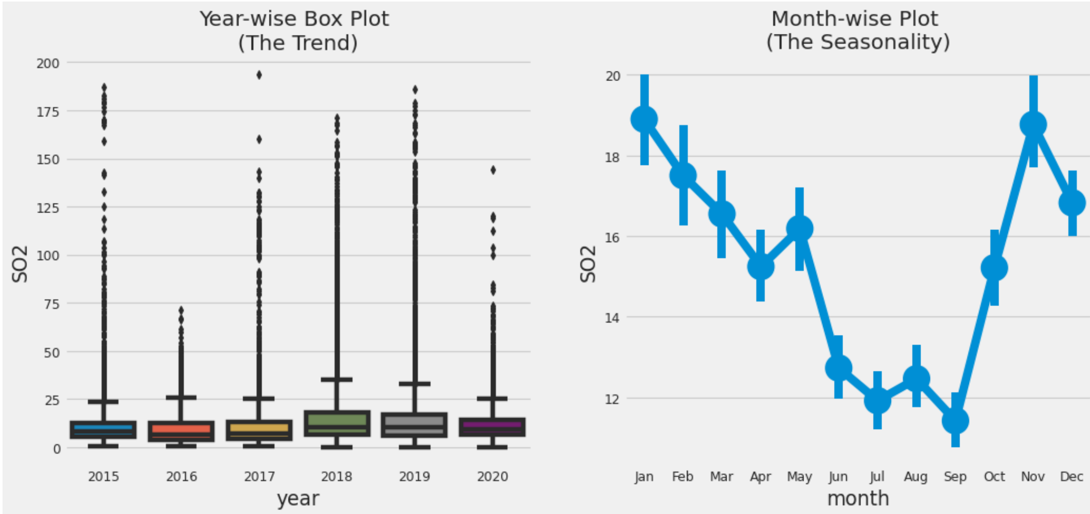
```
[ ]  def trend_plot(dataframe,value):

         # 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()
```

FOR NO2:

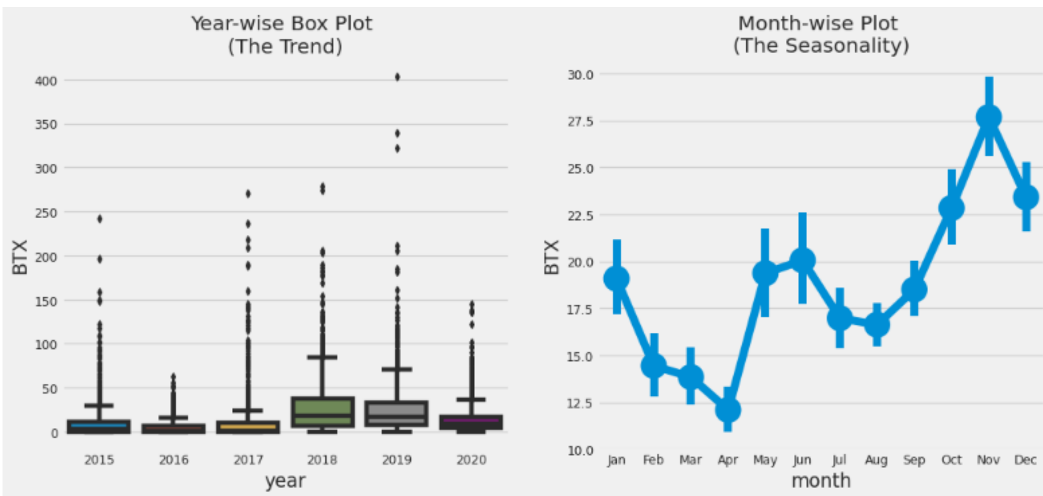
```
[ ]  city_day.reset_index(inplace=True)
     df = city_day.copy()
     value='NO2'
     trend_plot(df,value)
```



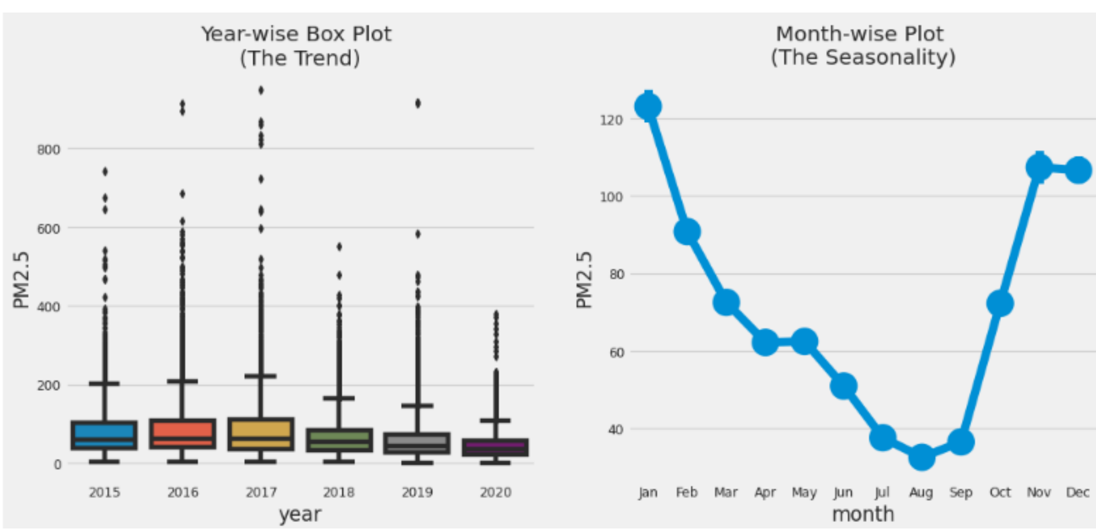
FOR SO2:



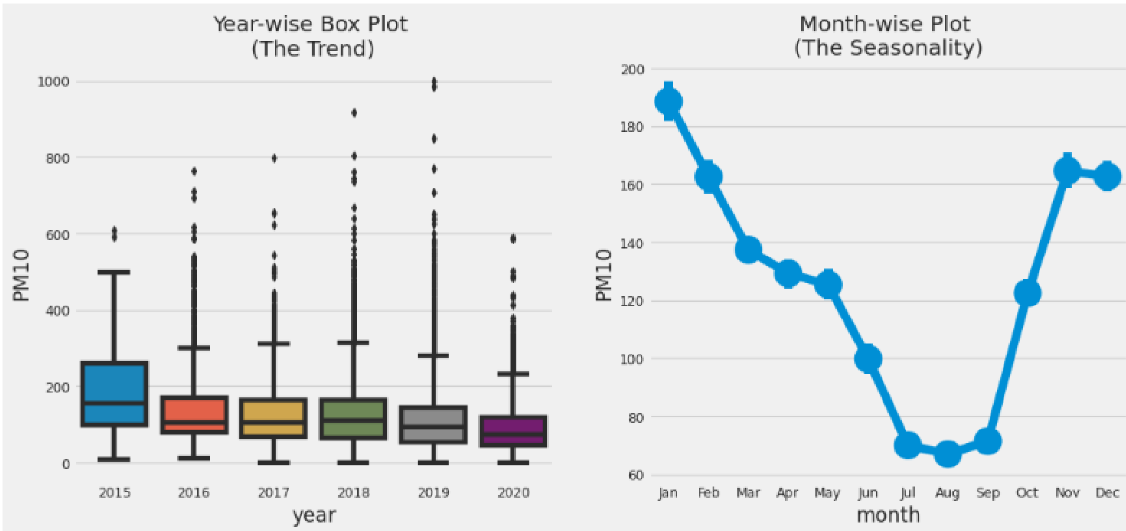
FOR BTX:



FOR PM2.5:



FOR PM10:



CALCULATING AND DISPLAYING THE TOP 10 POLLUTED CITIES BASED ON POLLUTANTS:

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

```
[ ] 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)
```

```
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
0	Patna	123.500000	0	Delhi	232.810000	0	Ahmedabad	59.030000
1	Delhi	117.200000	1	Gurugram	191.500000	1	Delhi	50.790000
2	Gurugram	117.100000	2	Talcher	165.770000	2	Kolkata	40.400000
3	Lucknow	109.710000	3	Jorapokhar	149.660000	3	Patna	37.490000
4	Ahmedabad	67.850000	4	Patna	126.750000	4	Visakhapatnam	37.190000
5	Kolkata	64.360000	5	Brajrajnagar	124.220000	5	Lucknow	33.240000
6	Jorapokhar	64.230000	6	Jaipur	123.480000	6	Jaipur	32.420000
7	Brajrajnagar	64.060000	7	Bhopal	119.320000	7	Bhopal	31.350000
8	Guwahati	63.690000	8	Guwahati	116.600000	8	Coimbatore	28.780000
9	Talcher	61.410000	9	Kolkata	115.630000	9	Hyderabad	28.390000

	City	SO2		City	CO		City	BTX
0	Ahmedabad	55.250000	0	Ahmedabad	22.190000	0	Kolkata	38.230000
1	Jorapokhar	33.650000	1	Lucknow	2.130000	1	Ahmedabad	37.110000
2	Talcher	28.490000	2	Delhi	1.980000	2	Delhi	26.860000
3	Patna	22.130000	3	Talcher	1.850000	3	Patna	17.430000
4	Kochi	17.600000	4	Bengaluru	1.840000	4	Visakhapatnam	15.030000
5	Delhi	15.900000	5	Brajrajnagar	1.800000	5	Gurugram	14.600000
6	Mumbai	15.200000	6	Ernakulam	1.630000	6	Amritsar	14.580000
7	Guwahati	14.660000	7	Patna	1.530000	7	Hyderabad	10.730000
8	Amaravati	14.260000	8	Kochi	1.300000	8	Chandigarh	9.090000
9	Bhopal	13.060000	9	Gurugram	1.260000	9	Amaravati	3.680000

PATNA HAS THE HIGHEST PM2.5 LEVEL:

Particulate pollution from domestic sources is one of the biggest causes of Patna's poor air quality. The continued usage of solid fuel in the form of wood and dung cakes on traditional cook stoves causes extremely high exposure to particulate pollution.

https://www.downtoearth.org.in/news/air/if-you-are-in-patna-you-are-exposed-to-very-high-levels-of-pm2-5-concentration-56759.

DELHI HAS THE HIGHEST PM10 LEVEL:

Motor vehicle emissions are one of the causes of poor air quality in Delhi. Other causes include wood-burning fires, cow dung cake combustion, fires on agricultural land, exhaust from diesel generators, dust from construction sites, burning garbage and illegal industrial activities in Delhi. Although pollution is at its worst from November to February. It is a noxious mix of emissions from its 9 million vehicles, construction dust and burning of waste. On the worst days, the air quality index, a benchmark ranging from zero (good) to 500 (hazardous), exceeds 400. Although Delhi is kerosene free and 90% of the households use LPG for cooking, the remaining 10% uses wood, crop residue, cow dung, and coal for cooking. Fire in Bhalswa landfill is a major reason for airborne particles in Delhi.

https://en.wikipedia.org/wiki/Air_pollution_in_Delhi

AHMEDABAD HAS THE HIGHEST NO2, SO2 AND CO:

The study found that Ahmedabad have: 30% road dust; 25% power plants; 20% vehicle exhaust; 15% industry; 5% domestic cooking and heating; 2% diesel generator sets; 2% waste burning and 1% construction activities.

https://www.nrdc.org/sites/default/files/ahmedabad_aqi_-_final.pdf

KOLKATA HAS THE HIGHEST BTX :

Exhaust from different types of motor vehicles plying on the streets of Calcutta and smoke plumes from coal-based stoves were also sampled. Even unleaded petrol, which has been introduced in metropolitan cities, has high levels of benzene.

https://www.downtoearth.org.in/news/cancer-in-kolkata-air-4361

FILTERING THE TOP METRO CITIES:

```
cities = ['Ahmedabad','Delhi','Bengaluru','Mumbai','Hyderabad','Chennai']

filtered_city_day = city_day[city_day['Date'] >= '2019-01-01']
AQI = filtered_city_day[filtered_city_day.City.isin(cities)][['Date','City','AQI','AQI_Bucket']]
AQI.head()
```

	Date	City	AQI	AQI_Bucket
1461	2019-01-01	Ahmedabad	1474.0	Severe
1462	2019-01-02	Ahmedabad	1246.0	Severe
1463	2019-01-03	Ahmedabad	1719.0	Severe
1464	2019-01-04	Ahmedabad	1264.0	Severe
1465	2019-01-05	Ahmedabad	1127.0	Severe

```
AQI_pivot = AQI.pivot(index='Date', columns='City', values='AQI')
AQI_pivot.fillna(method='bfill',inplace=True)
AQI_beforeLockdown = AQI_pivot['2020-01-01':'2020-03-25']
AQI_afterLockdown = AQI_pivot['2020-03-26':'2020-05-01']
print(AQI_beforeLockdown.mean())
print(AQI_afterLockdown.mean())
```

```
City
Ahmedabad    383.776471
Bengaluru     96.023529
Chennai       80.317647
Delhi        246.305882
Hyderabad     94.435294
Mumbai       148.776471
dtype: float64
City
Ahmedabad    127.810811
Bengaluru     68.486486
Chennai       62.351351
Delhi        107.270270
Hyderabad     65.567568
Mumbai        73.891892
dtype: float64
```

NEXT WE TAKE THE city_day.csv FOR :

- Extracting year and month for each record
- Clubbing all particulate matter
- Clubbing nitrogen oxides
- Clubbing Benzene, toluene and Xylene together
- Grouping the cities based on year, month and pollutants by their mean.

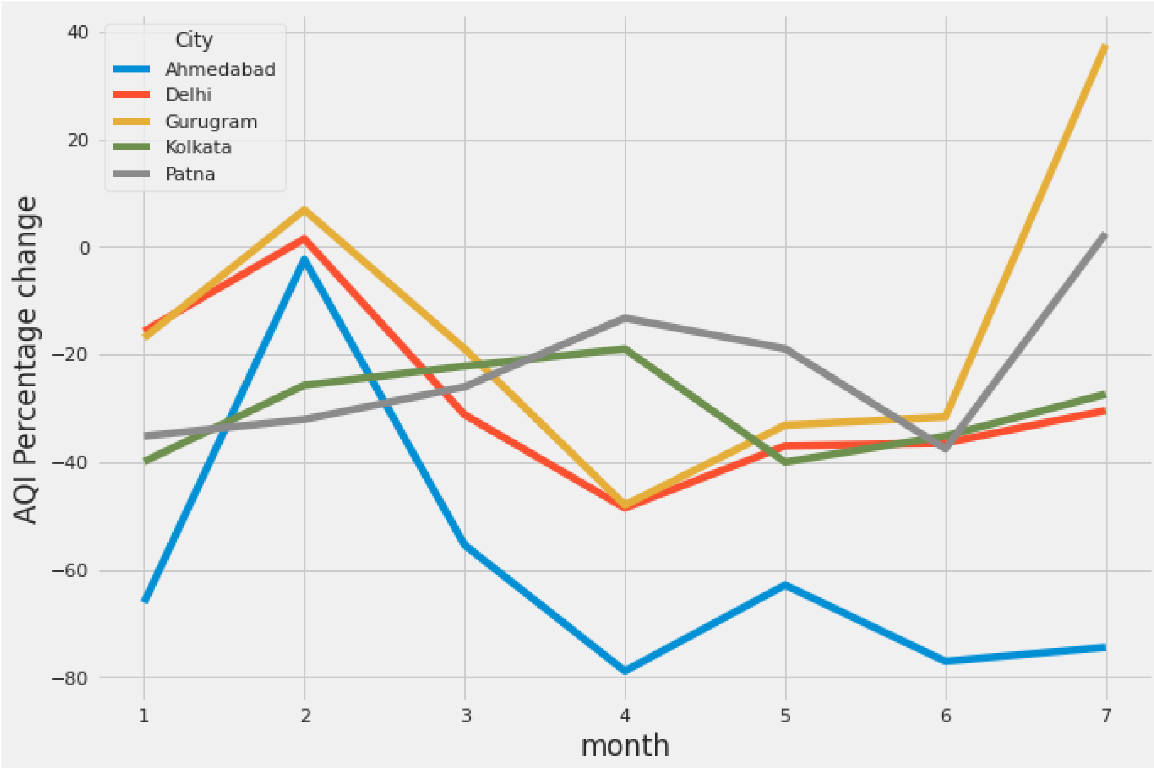
```python
cities = pd.read_csv('/content/city_day.csv')
cities['Date']=pd.to_datetime(cities['Date'])
cities.fillna(0,inplace=True)

#extracting year and month for each record
cities['year'] = pd.DatetimeIndex(cities['Date']).year
cities['month'] = pd.DatetimeIndex(cities['Date']).month

#clubbing all particulate matter
cities['PM']=cities['PM2.5'] + cities['PM10']

#clubbing nitrogen oxides
cities['Nitric']=cities['NO'] + cities['NO2']+ cities['NOx']

#clubbing Benzene, toluene and Xylene together
cities['BTX']=cities['Benzene'] + cities['Toluene']+ cities['Xylene']

cities_group_ym=cities.groupby(['City','year','month'])[['PM','Nitric','CO','NH3','O3','SO2','BTX','AQI']].mean()

cities_group_ym=cities_group_ym.reset_index(['City','year','month'])
cities_group_ym.head()
```

	City	year	month	PM	Nitric	CO	NH3	O3	SO2	BTX	AQI
0	Ahmedabad	2015	1	10.668710	88.680000	22.352258	0.0	46.350645	43.602903	6.971613	33.903226
1	Ahmedabad	2015	2	103.662143	92.985714	19.482143	0.0	43.437857	56.423214	35.357143	464.857143
2	Ahmedabad	2015	3	106.905806	80.510000	13.585484	0.0	44.276774	56.975161	41.357419	378.064516
3	Ahmedabad	2015	4	101.682000	54.992667	7.306333	0.0	31.376000	51.233333	14.496333	257.200000
4	Ahmedabad	2015	5	74.919355	50.607419	8.529677	0.0	31.624194	35.977419	19.677419	254.967742

PLOTTING THE AQI OF HIGHLY POLLUTED CITIES:

- Creating a list of some of the most polluted cities
- Forming two df's- containing data from 2019 and 2020 respectively
- Computing the percentage change in AQI
- Plotting line plot for AQI change for a few highly polluted cities

```python
#creating a list of some of the most polluted cities
most_polluted=['Delhi','Patna','Ahmedabad','Gurugram','Kolkata']

#forming two df's- containing data from 2019 and 2020 respectively
cities_2019= cities_group_ym[(cities_group_ym['City'].isin(most_polluted)) & (cities_group_ym['year']==2019)]
cities_2020= cities_group_ym[(cities_group_ym['City'].isin(most_polluted)) & (cities_group_ym['year']==2020)]

cities_19_vs_20 = pd.merge(cities_2019, cities_2020, how="inner", on=["City", "month"])

#computing the percentage change in AQI
cities_19_vs_20['AQI Percentage change']=100*(cities_19_vs_20['AQI_y']-cities_19_vs_20['AQI_x'])/cities_19_vs_20['AQI_

#plotting AQI change for a few highly polluted cities
fig = plt.figure(figsize=(10,7))
sns.lineplot(
    data=cities_19_vs_20,
    x="month", y="AQI Percentage change",hue='City',linewidth=4.5,
    markers=True, dashes=False
)
```



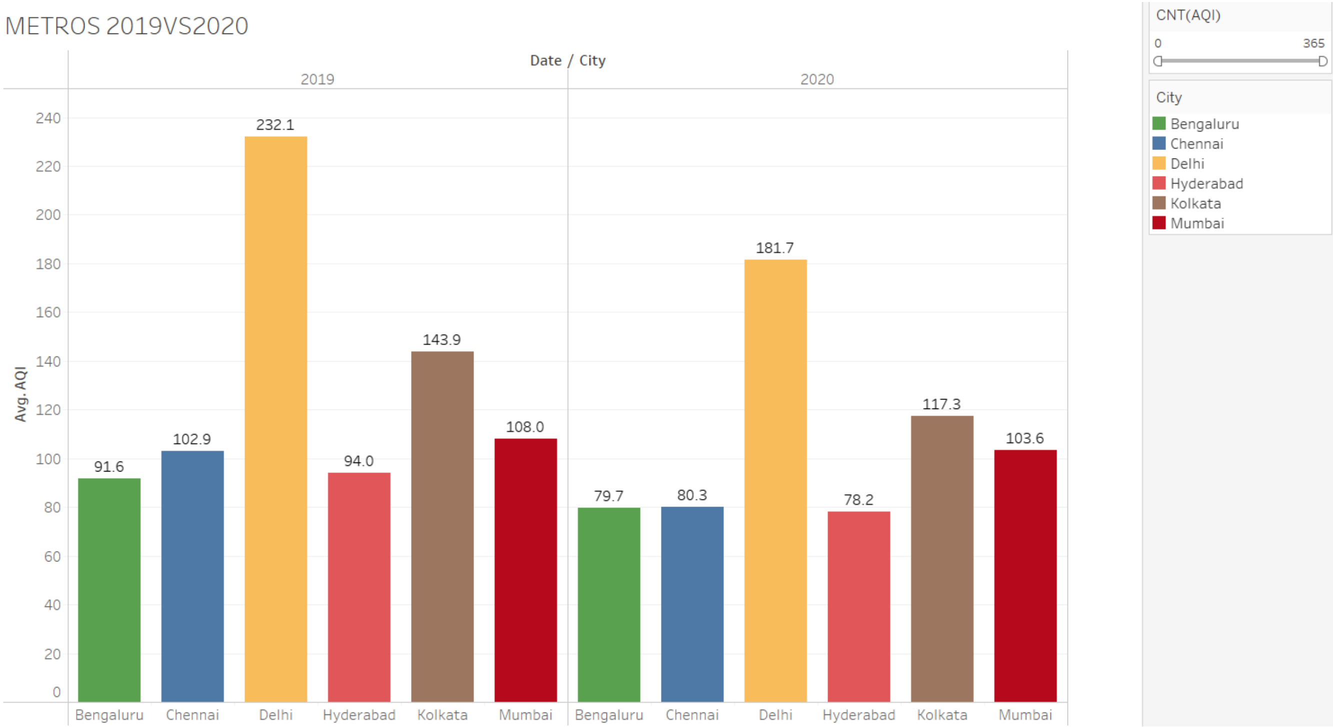
INFERENCE DRAWN:

The general trend shows that the AQI indeed decreased for the lockdown months, signifying a major improvement in Air quality with reduced pollution levels.

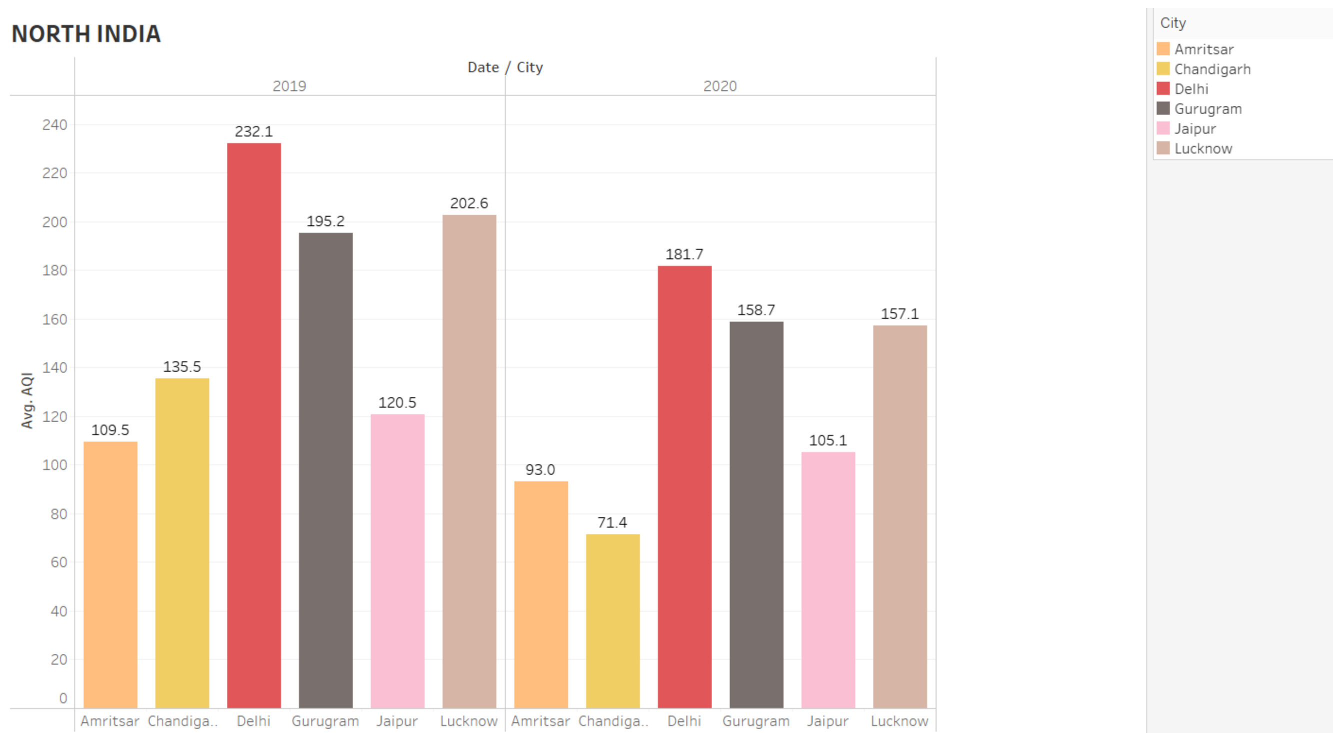
However, we will now investigate the cities which fared the best in these 4 months and also the ones which showed anomalies with a spike in AQI.

AVERAGE AQI LEVELS ACROSS METROPOLITAN CITIES (using Tableau)

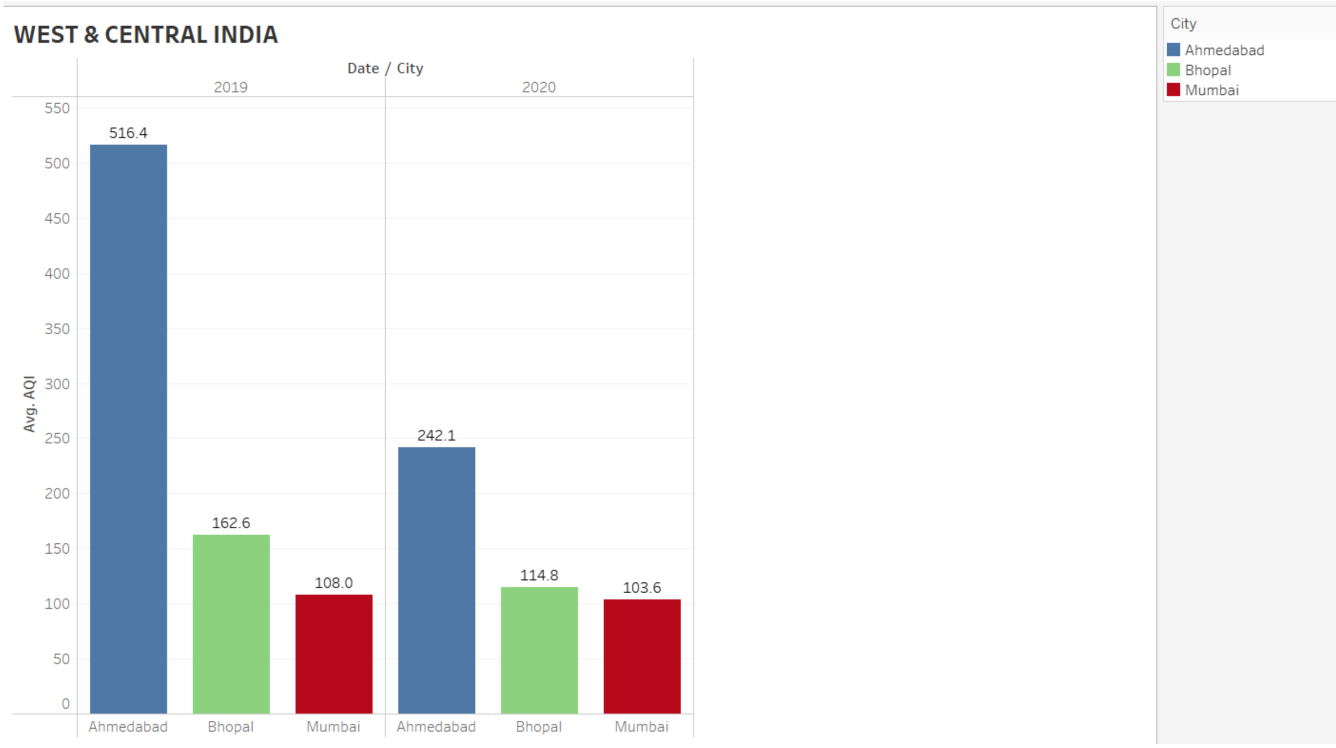
2019 VS 2020



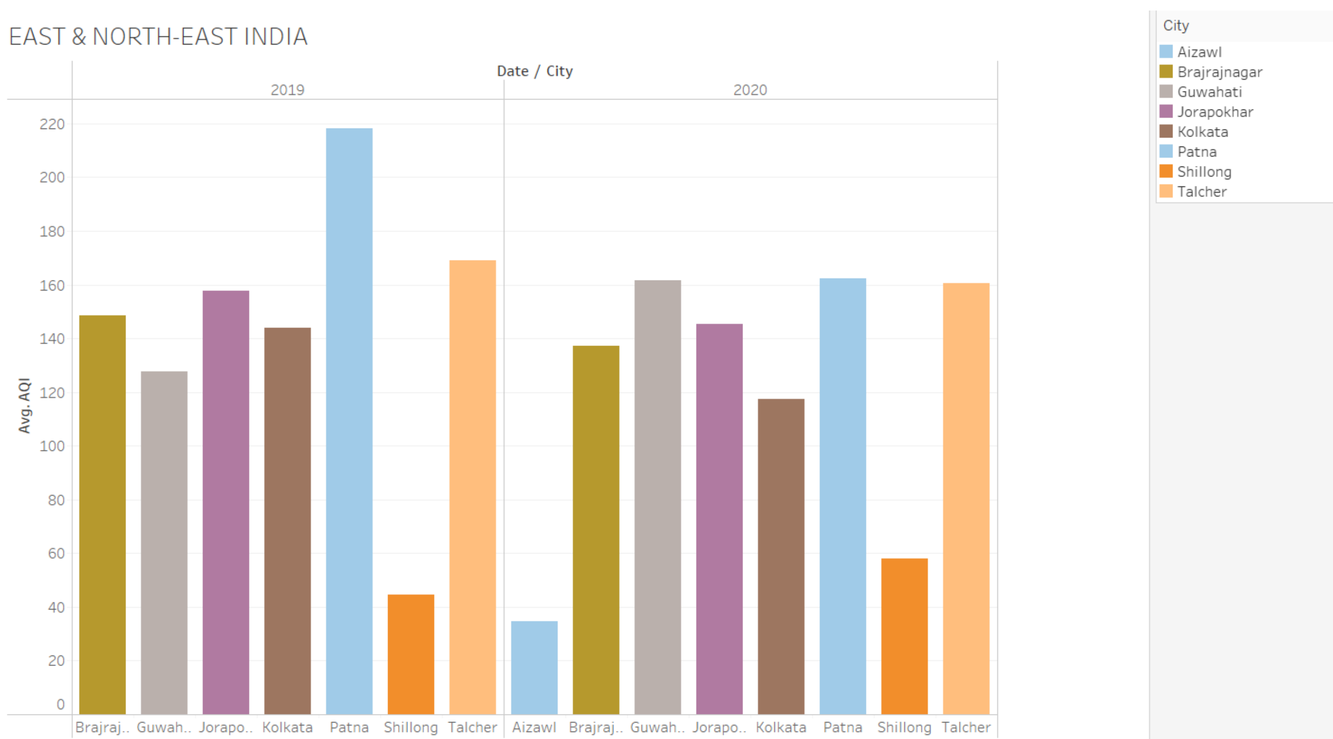
AQI READING ACROSS NORTH INDIA (2019 VS 2020):



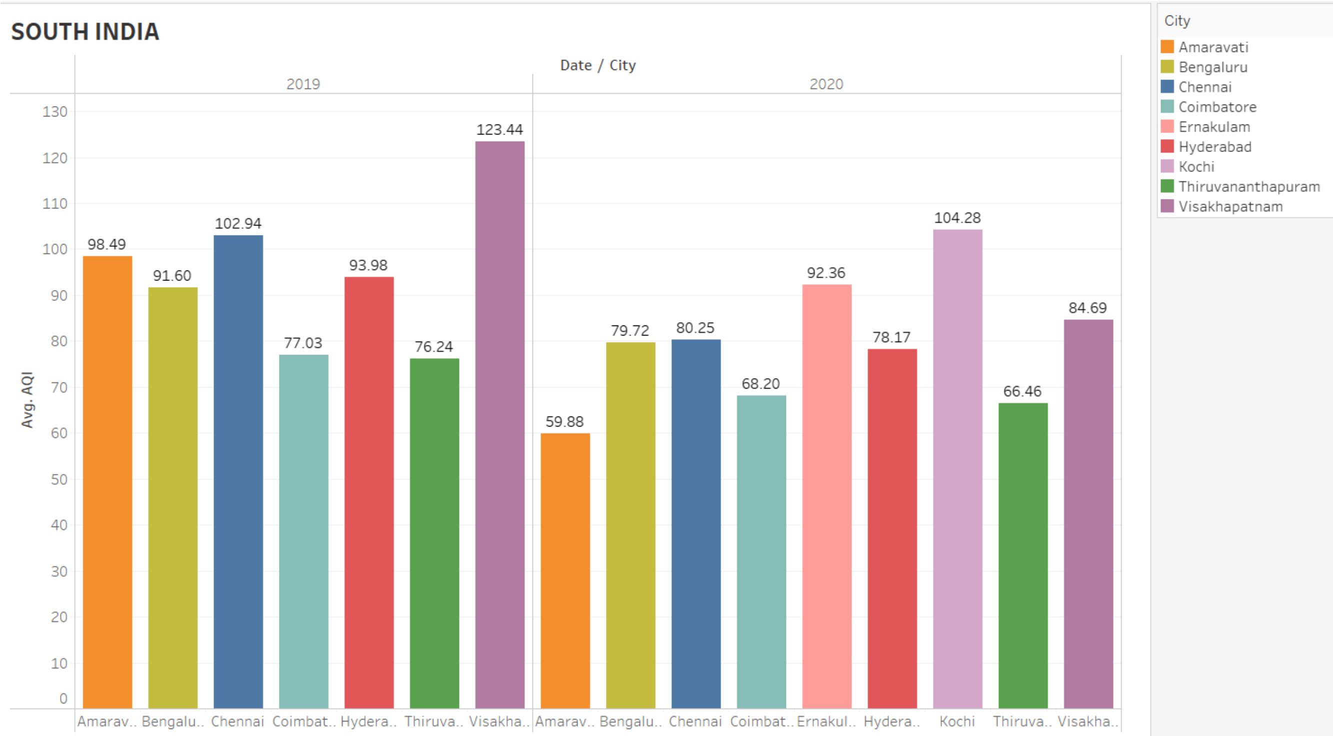
AQI READINGS OF WEST AND CENTRAL INDIA (2019 VS 2020):



AQI READING FROM EAST AND THE NORTH-EAST (2019 VS 2020):



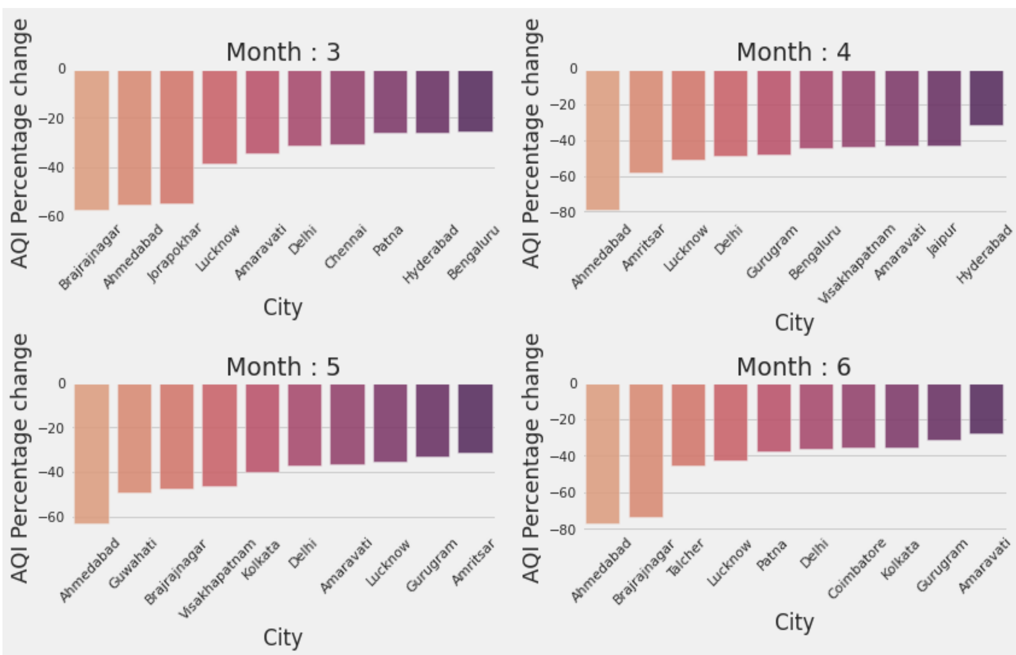
AQI READINGS FROM SOUTH (2019 VS 2020):



CITIES WHICH HAD UNDERWENT THE MOST DRASTIC IMPROVEMENT IN AIR QUALITY:

- Forming two separate dataframes for years 2019 and 2020
- Joining the two dataframes to get a comparative view of AQI value in 2019 and 2020
- Lockdown months- which we will be analyzing
- Plotting the top 10 cities for the months March-June 2020 which had the most improvement in AQI
- Plotting bar plots

```python
#forming two seperate dataframes for years 2019 and 2020
cities_19_all= cities_group_ym[cities_group_ym['year']==2019]
cities_20_all= cities_group_ym[cities_group_ym['year']==2020]

#joining the two df's to get a comparitive view of AQI value in 2019 and 2020
cities_19_vs_20_all = pd.merge(cities_19_all, cities_20_all, how="inner", on=["City", "month"])
cities_19_vs_20_all['AQI Percentage change']=100*(cities_19_vs_20_all['AQI_y']-cities_19_vs_20_all['AQI_x'])/cities_19_vs_20_all['AQI_x']

#lockdown months- which we will be analysing
months=[3,4,5,6]
fig, axes = plt.subplots(ncols=2, nrows=2,figsize=(12, 7))

#plotting the top 10 cities for the months March-June 2020 which had the most improvement in AQI
for i, ax in zip(months, axes.flat):
    cities_AQI_comp=cities_19_vs_20_all[(cities_19_vs_20_all['AQI_y']!= 0.000000) & (cities_19_vs_20_all['month']==i)]
    cities_AQI_comp_10=cities_AQI_comp[['City','month','AQI_x','AQI_y','AQI Percentage change']].sort_values(by='AQI Percentage change',
    ascending=True).iloc[:10,:]

    h=sns.barplot(data=cities_AQI_comp_10, x="City", y='AQI Percentage change', palette="flare", alpha=.9, ax=ax)
    h.set(title='Month : {}'.format(i))
    h.set_xticklabels(h.get_xticklabels(), rotation=45)

plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



INFERENCE DRAWN:

We can see that there has been a significant improvement in the air quality for these cities over the four months.

CITIES WHICH SHOWED AN INCREASED AQI AS COMPARED TO 2019 IN THE LOCKDOWN-MONTHS:

- Analyzing the cities which showed a positive AQI change percentage, denoting an increased AQI in 2020.

```
[ ] #finding cities which have a higher AQI in months March April, May, June of 2020 as compared to the same months last year
    cities_19_vs_20_all[(cities_19_vs_20_all['AQI Percentage change']>=0)&(cities_19_vs_20_all['month'].
            isin([3,4,5,6]) )][['City','month','AQI_x','AQI_y','AQI Percentage change']].sort_values(by='AQI Percentage change', ascending=False)
```

	City	month	AQI_x	AQI_y	AQI Percentage change
83	Jorapokhar	6	0.000000	136.533333	inf
125	Thiruvananthapuram	6	28.266667	45.400000	60.613208
60	Guwahati	4	105.933333	127.833333	20.673379
82	Jorapokhar	5	113.709677	135.580645	19.234043
31	Brajrajnagar	4	101.633333	119.533333	17.612332
116	Talcher	4	118.466667	127.733333	7.822172
81	Jorapokhar	4	113.833333	121.400000	6.647145

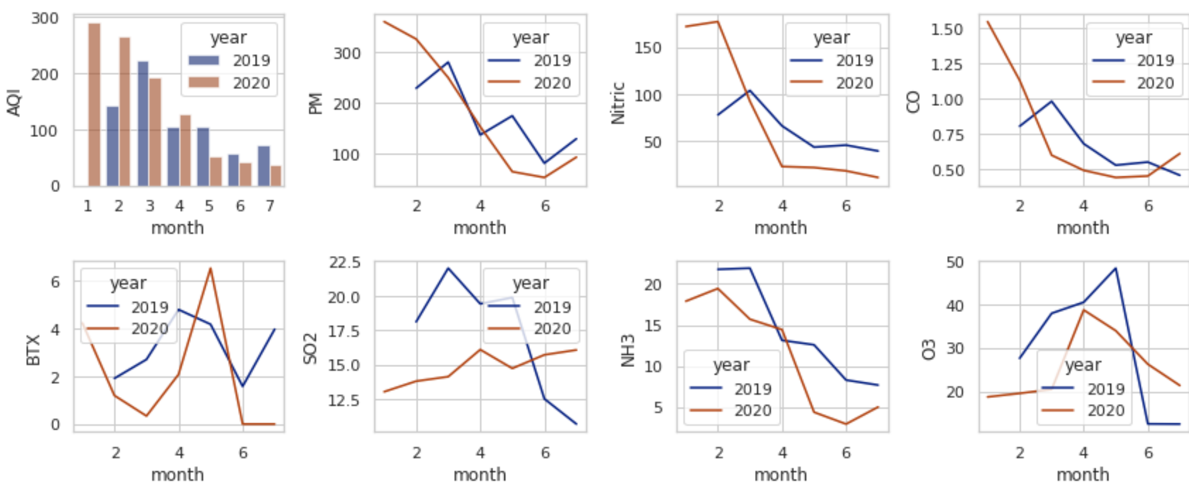
- Analyzing in detail the towns/cities which recorded a higher AQI in April and May of 2020 as compared to 2019.
- cities displayed above which showed a higher AQI in the lockdown months of 2020 as compared to 2019.
- understanding the rise of pollutants which contributed to the increased AQI in 2020 by comparing the levels of each pollutant in 2019 and 2020.

```
#cities displayed above which showed a higher AQI in the lockdown months of 2020 as compared to 2019
anomalies=['Guwahati','Jorapokhar','Brajrajnagar','Talcher']

#understanding the rise of pollutants which contributed to the increased AQI in 2020 by comparing the levels of each pollutant in 2019 and 20:
for i in anomalies:
    city_19_20= cities_group_ym[(cities_group_ym['City']==i) &
                               (cities_group_ym['year'].isin([2019,2020]))&
                              (cities_group_ym['month']<8)]

    sns.set_theme(style="whitegrid")
    fig = plt.figure()
    fig, axes = plt.subplots(2,4,figsize=(12, 5))

    sns.barplot(
        data=city_19_20,
        x="month", y="AQI", hue="year",
         palette="dark", alpha=.6,ax=axes[0,0]
    )
    sns.lineplot(
        data=city_19_20,
        x="month", y="PM", hue="year",palette='dark',
        markers=True, dashes=False,ax=axes[0,1]
    )
    sns.lineplot(
        data=city_19_20,
        x="month", y="Nitric", hue="year",palette='dark',
        markers=True, dashes=False,ax=axes[0,2]
    )
    sns.lineplot(
        data=city_19_20,
        x="month", y="CO", hue="year",palette='dark',
        markers=True, dashes=False,ax=axes[0,3]
    )
```

```python
sns.lineplot(
    data=city_19_20,
    x="month", y="BTX", hue="year",palette='dark',
    markers=True, dashes=False,ax=axes[1,0]
)
sns.lineplot(
    data=city_19_20,
    x="month", y="SO2", hue="year",palette='dark',
    markers=True, dashes=False,ax=axes[1,1]
)
sns.lineplot(
    data=city_19_20,
    x="month", y="NH3", hue="year",palette='dark',
    markers=True,ax=axes[1,2]
)
sns.lineplot(
    data=city_19_20,
    x="month", y="O3", hue="year",palette='dark',
    markers=True, ax=axes[1,3]
)

fig.tight_layout()
print(i,':')
plt.show()
```

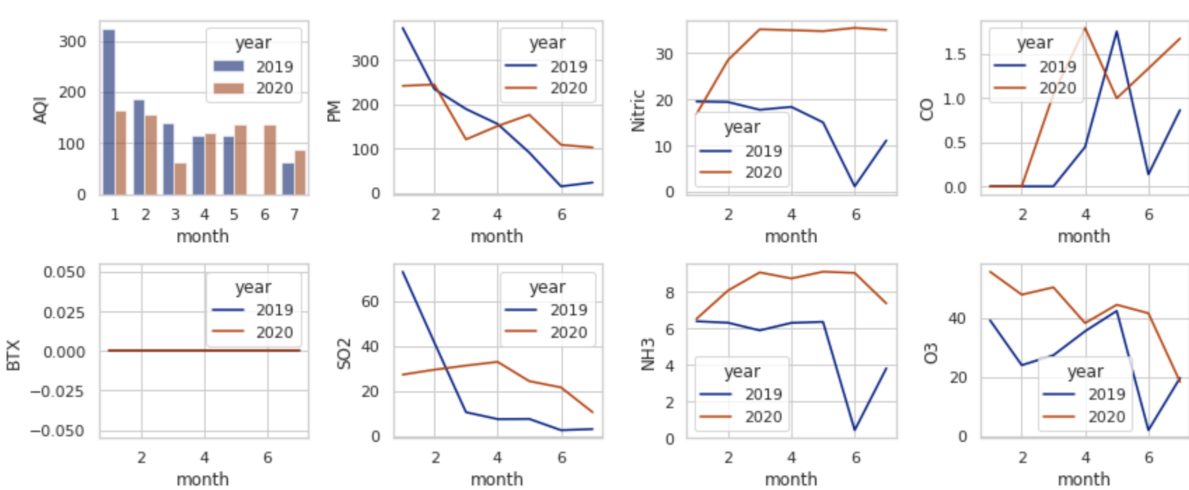
Guwahati:

<Figure size 576x360 with 0 Axes>



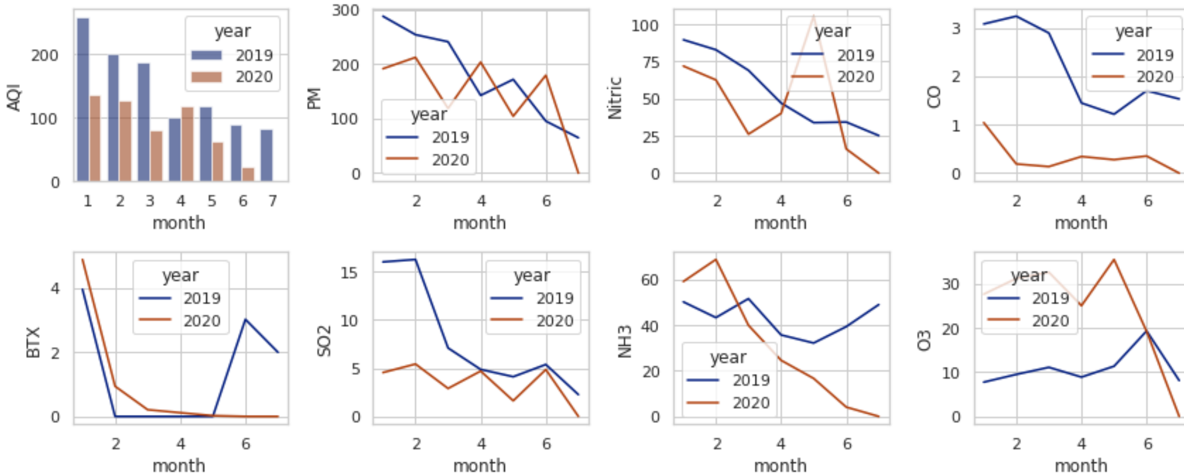
Jorapokhar:

<Figure size 576x360 with 0 Axes>



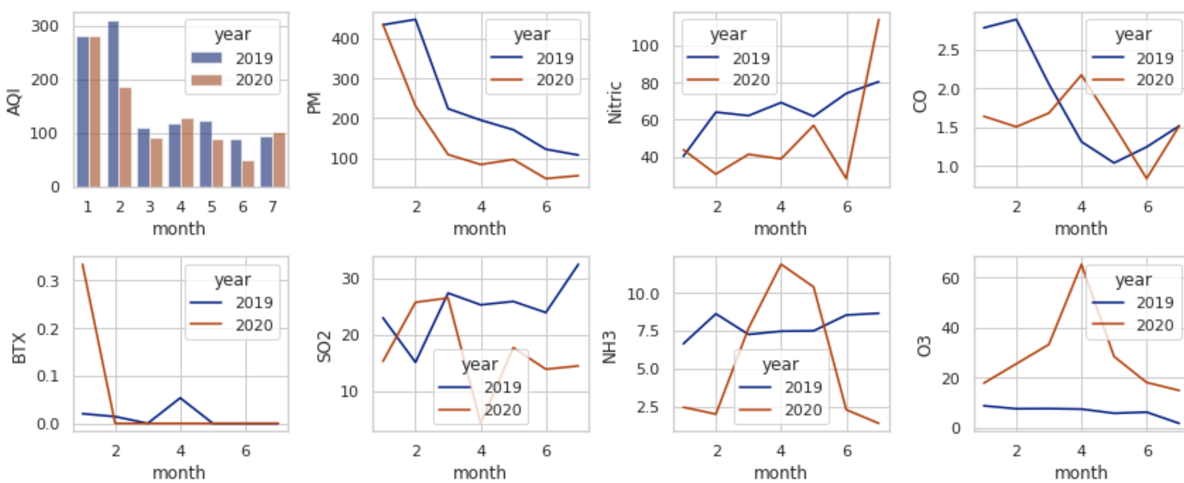
Brajrajnagar:

<Figure size 576x360 with 0 Axes>



Talcher:

<Figure size 576x360 with 0 Axes>



INFERENCE DRAWN:

We see that the four cities mentioned above did not witness an improvement in AQI during the COVID-19 induced lockdown as expected. The reason might be manifold: sparse AQI readings in 2020, flouting of lockdown norms, or any other natural phenomenon overriding the positive impact of decreased human and industrial activity.

****Guwahati:***

We see that AQI for April '20 is more than 20% higher as compared to April '19. Particulate matter and NH3 were the increased contributing factors.

Jorapokhar:

We see that AQI for May'20 is substantially higher as compared to May'19. Concentration of almost all pollutants have increased.

Brajrajnagar:

Higher AQI in April '20 as compared to April '19. Can be attributed to increased O3 and PM levels.

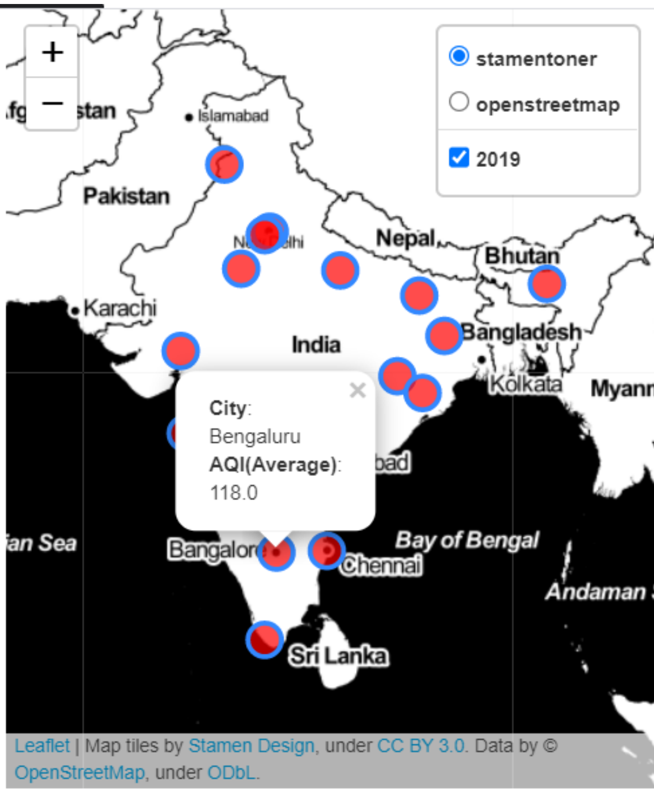
Talcher:

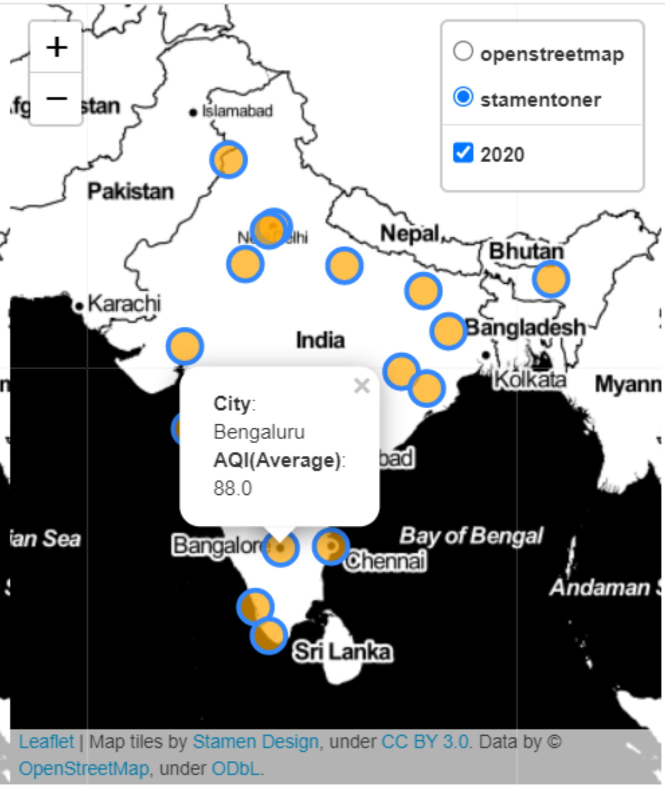
Higher AQI in April '20 as compared to April '19. Can be attributed to increased CO,O3 and NH3 levels.

REPRESENTATION OF NAMMA BENGALURU 2019 VS 2020

```
m = plugins.DualMap(location=(22.9734, 78.6569), tiles=None, zoom_start=5)

folium.TileLayer('Stamen Toner').add_to(m)
folium.TileLayer('openstreetmap').add_to(m)


fg_1 = folium.FeatureGroup(name='2019').add_to(m.m1)
fg_2 = folium.FeatureGroup(name='2020').add_to(m.m2)



for lat, lon, value, name in zip(df_2019_AQI['Lat'], df_2019_AQI['Long'], df_2019_AQI['AQ
I'], df_2019_AQI['City']):
    folium.CircleMarker([lat, lon],
                        radius=10,
                        icon=folium.Icon(color='red'),
                        popup = ('<strong>City</strong>: ' + str(name).capitalize() + '<br>'
                                '<strong>AQI(Average)</strong>: ' + str(value) + '<br>'),
                        fill_color='red',
                        fill_opacity=0.7 ).add_to(fg_1)
```

```
for lat, lon, value, name in zip(df_2020_AQI['Lat'], df_2020_AQI['Long'], df_2020_AQI['AQ
I'], df_2020_AQI['City']):
    folium.CircleMarker([lat, lon],
                        radius=10,
                        icon=folium.Icon(color='orange'),
                        popup = ('<strong>City</strong>: ' + str(name).capitalize() + '<br>'
                                '<strong>AQI(Average)</strong>: ' + str(value) + '<br>'),
                        fill_color='orange',
                        fill_opacity=0.7 ).add_to(fg_2)


folium.LayerControl(collapsed=False).add_to(m)

m
```





Time Series Analysis and Forecasting

Importing day-wise data of the cities.

```
import warnings
warnings.filterwarnings('ignore')

#importing day-wise data of cities
df= pd.read_csv('city_day.csv')

df['Date'] = pd.to_datetime(df['Date'])

#visualizing the top rows of the dataset
df.tail(5)
```

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
29526	Visakhapatnam	2020-06-27	15.02	50.94	7.68	25.06	19.54	12.47	0.47	8.55	23.30	2.24	12.07	0.73	41.0	Good
29527	Visakhapatnam	2020-06-28	24.38	74.09	3.42	26.06	16.53	11.99	0.52	12.72	30.14	0.74	2.21	0.38	70.0	Satisfactory
29528	Visakhapatnam	2020-06-29	22.91	65.73	3.45	29.53	18.33	10.71	0.48	8.42	30.96	0.01	0.01	0.00	68.0	Satisfactory
29529	Visakhapatnam	2020-06-30	16.64	49.97	4.05	29.26	18.80	10.03	0.52	9.84	28.30	0.00	0.00	0.00	54.0	Satisfactory
29530	Visakhapatnam	2020-07-01	15.00	66.00	0.40	26.85	14.05	5.20	0.59	2.10	17.05	NaN	NaN	NaN	50.0	Good

We pivot the values from the 'City' column, so that we can have a comparative view of the value of every city's AQI through every day.

Then we resample them to find the mean of every month, so now our dataset contains month-wise data.

```
[4] cities_all = df.pivot_table(values='AQI', index=['Date'], columns='City')
    cities_all=cities_all.add_suffix('_AQI')
    cities=cities_all.resample(rule='MS').mean()
    cities.head()
```

City Date	Ahmedabad_AQI	Aizawl_AQI	Amaravati_AQI	Amritsar_AQI	Bengaluru_AQI	Bhopal_AQI	Brajrajnagar_AQI	Chandigarh_AQI	Chennai_AQI	Coimbatore_AQI	...	Jorapokhar_AQI	Kochi_AQI	Kolkata_AQI	Lucknow_AQI	Mumbai_AQI
2015-01-01	350.333333	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
2015-02-01	520.640000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
2015-03-01	418.571429	NaN	NaN	NaN	130.545455	NaN	NaN	NaN	363.800000	NaN	...	NaN	NaN	NaN	264.272727	NaN
2015-04-01	308.640000	NaN	NaN	NaN	113.733333	NaN	NaN	NaN	175.862069	NaN	...	NaN	NaN	NaN	118.586207	NaN
2015-05-01	263.466667	NaN	NaN	NaN	102.774194	NaN	NaN	NaN	176.129032	NaN	...	NaN	NaN	NaN	137.000000	NaN

5 rows × 26 columns

FORM A NEW COLUMN CONTAINING INDIA'S AQI FOR EVERY MONTH BY TAKING THE AVERAGE OF ALL CITIES FOR THAT MONTH

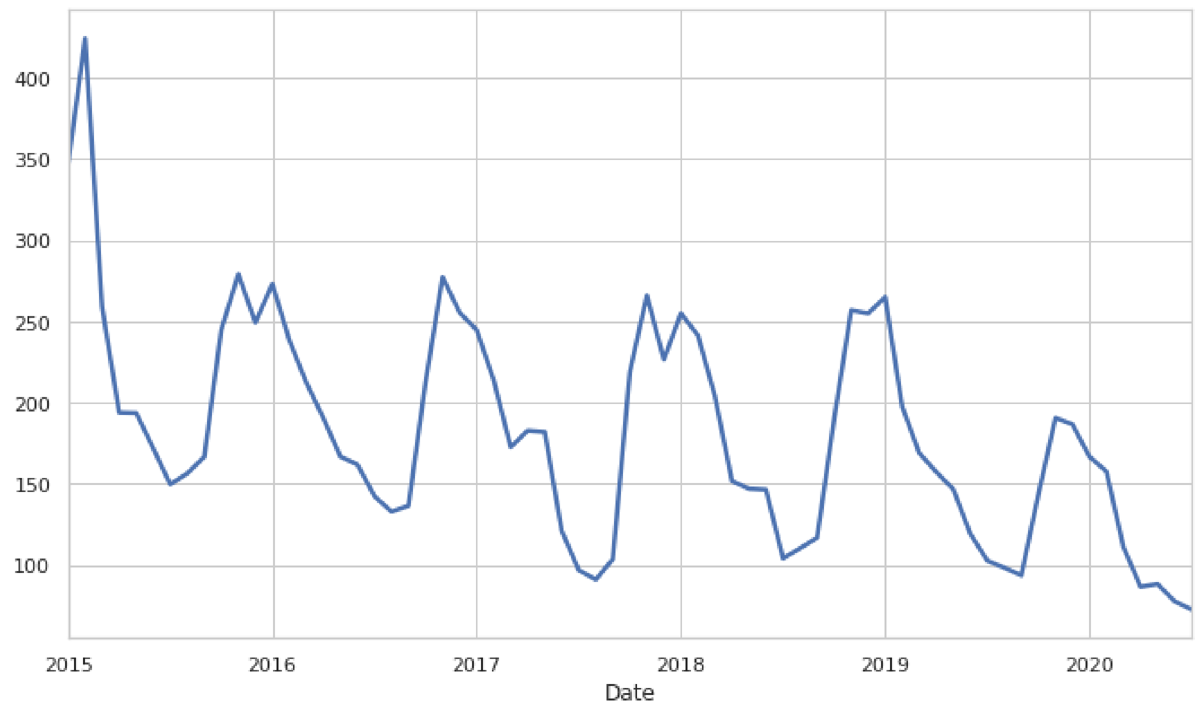
```
[5] #form a new column containing India's AQI for every month by taking the average of all cities for that month
    cities['India_AQI']=cities.mean(axis=1)
    cities.head()
```

City Date	Ahmedabad_AQI	Aizawl_AQI	Amaravati_AQI	Amritsar_AQI	Bengaluru_AQI	Bhopal_AQI	Brajrajnagar_AQI	Chandigarh_AQI	Chennai_AQI	Coimbatore_AQI	...	Kochi_AQI	Kolkata_AQI	Lucknow_AQI	Mumbai_AQI	Patna_AQI	Shill
2015-01-01	350.333333	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	
2015-02-01	520.640000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	
2015-03-01	418.571429	NaN	NaN	NaN	130.545455	NaN	NaN	NaN	363.800000	NaN	...	NaN	NaN	264.272727	NaN	NaN	
2015-04-01	308.640000	NaN	NaN	NaN	113.733333	NaN	NaN	NaN	175.862069	NaN	...	NaN	NaN	118.586207	NaN	NaN	
2015-05-01	263.466667	NaN	NaN	NaN	102.774194	NaN	NaN	NaN	176.129032	NaN	...	NaN	NaN	137.000000	NaN	NaN	

5 rows × 27 columns

PLOTTING INDIA'S AQI

```
cities.reset_index()

sns.set_theme(style='whitegrid')

#plot India's AQI
cities['India_AQI'].plot(kind='line',grid=True,figsize=(10, 6), linewidth=2.5)
```



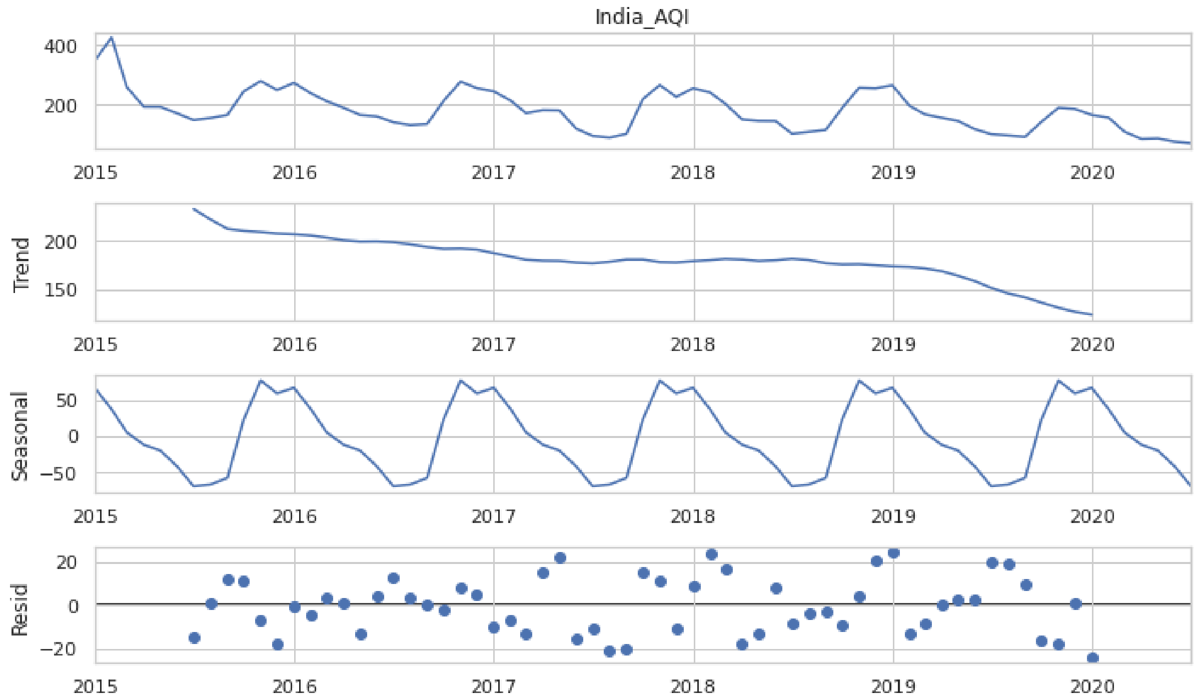
INFERENCE DRAWN:

From the plot above, we can visually see that there is a slight downward trend and a seasonality present. However, we will decompose the plot into trend, seasonality and residuals to get a clearer picture.

SEASONALITY DECOMPOSITION

- We have used an additive model.

```
[7]  from statsmodels.tsa.seasonal import seasonal_decompose

     plt.rcParams['figure.figsize'] = (10, 6);
     cities['India_AQI']=cities.mean(axis=1)
     fig = seasonal_decompose(cities['India_AQI'], model='additive').plot()
```



INFERENCE DRAWN:

We can see a clear seasonality and trend present here. The AQI decreases towards mid-year before rising again.

TESTING THE STATIONARITY:

- ### Augmented Dicky Fuller Test:

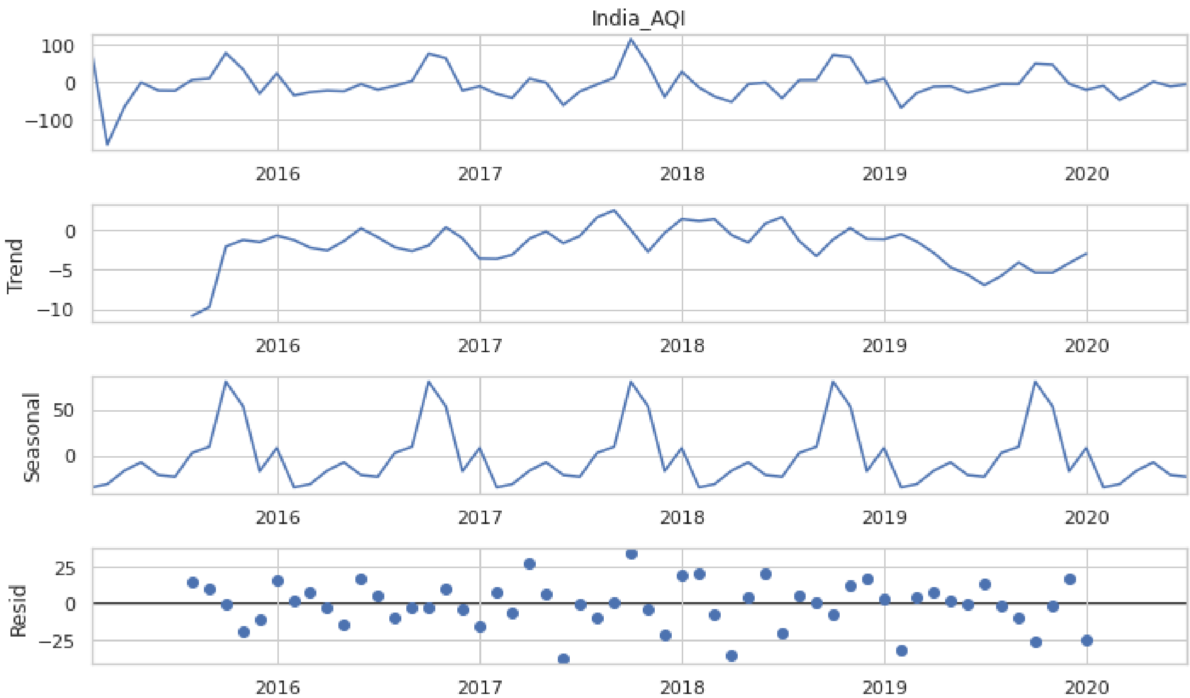
  We'll perform the ADF for determining stationarity of the time series.

  We observed that the p-value is 0.94, which means that this time series is not stationary. We perform a first order differencing to remove the trend and then perform the ADF test again.

```
[8]  from statsmodels.tsa.stattools import adfuller

     dftest = adfuller(cities['India_AQI'], autolag='AIC')
     dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
     for key,value in dftest[4].items():
         dfoutput['Critical Value (%s)'%key] = value
     dfoutput
```

```
Test Statistic                  -0.114224
p-value                          0.948003
#Lags Used                      10.000000
Number of Observations Used     56.000000
Critical Value (1%)             -3.552928
Critical Value (5%)             -2.914731
Critical Value (10%)            -2.595137
dtype: float64
```

```
[9]  diff = cities['India_AQI'].diff(periods=1)
     diff.dropna(inplace=True)
     fig = seasonal_decompose(diff, model='additive').plot()
```



INFERENCE DRAWN:

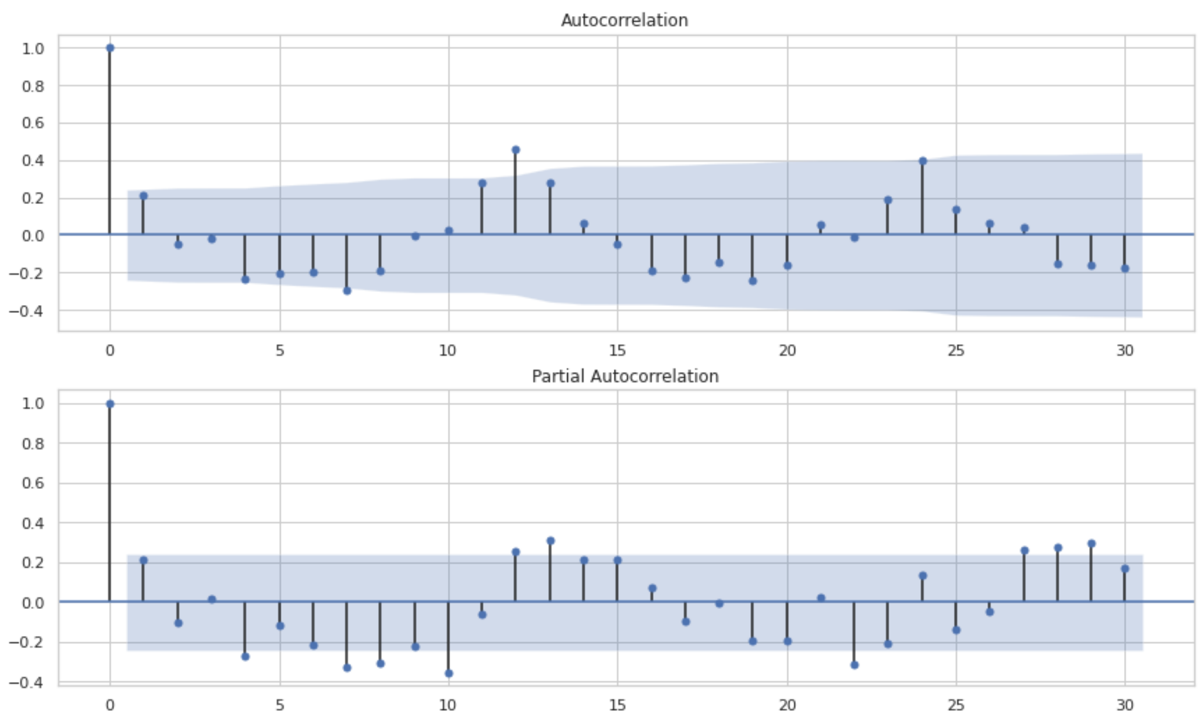
Multiplicative seasonality is not appropriate for zero and negative values. Therefore, we go ahead using the additive model.

```
[10] dftest = adfuller(diff)
     dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
     for key,value in dftest[4].items():
         dfoutput['Critical Value (%s)'%key] = value
     dfoutput
```

```
Test Statistic                 -8.385232e+00
p-value                         2.448599e-13
#Lags Used                      9.000000e+00
Number of Observations Used     5.600000e+01
Critical Value (1%)            -3.552928e+00
Critical Value (5%)            -2.914731e+00
Critical Value (10%)           -2.595137e+00
dtype: float64
```

From the p-value and the Test Statistic, we can conclude that with one differencing, the time series becomes stationary. Therefore, d=1.

PLOTTING THE AUTOCORRELATION AND PARTIAL AUTOCORRELATION GRAPHS:



We decided to go for auto-arima function to determine the parameters and alo to determine the best fitting SARIMAX

USE AUTO-ARIMA TO DETERMINE THE PARAMETERS OF THE SARIMA MODEL:

- Installing pmdarima

```
!pip install pmdarima;
from pmdarima import auto_arima;
```

```
auto_arima(y=cities['India_AQI'],start_p=1,start_P=1,start_q=1,start_Q=1,seasonal=True,m=12, stepwise=\
         True).summary()
```

SARIMAX Results

Dep. Variable:	y	No. Observations:	67
Model:	SARIMAX(0, 1, 2)x(1, 0, [1], 12)	Log Likelihood	-316.908
Date:	Sun, 13 Nov 2022	AIC	643.816
Time:	04:57:41	BIC	654.765
Sample:	01-01-2015	HQIC	648.143
	- 07-01-2020		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	0.0189	0.059	0.320	0.749	-0.097	0.135
ma.L2	-0.8363	0.069	-12.077	0.000	-0.972	-0.701
ar.S.L12	0.9444	0.062	15.221	0.000	0.823	1.066
ma.S.L12	-0.5623	0.229	-2.458	0.014	-1.011	-0.114
sigma2	694.3699	142.982	4.856	0.000	414.130	974.610

Ljung-Box (L1) (Q):	0.95	Jarque-Bera (JB):	2.99
Prob(Q):	0.33	Prob(JB):	0.22
Heteroskedasticity (H):	0.38	Skew:	-0.52
Prob(H) (two-sided):	0.03	Kurtosis:	2.99

- dividing into train and test
- Building the model
- printing summary of model results

```python
import statsmodels.api as sm
#dividing into train and test:
train_data=cities['India_AQI'][:'2018-12']
test_data=cities['India_AQI'][:'2019-12']

#Building the model:
model= sm.tsa.SARIMAX(train_data,order=(0,1,2),seasonal_order=(1,0,1,12), trend='n')
results=model.fit()

#printing summary of model results
results.summary()
```

SARIMAX Results

Dep. Variable:	India_AQI	**No. Observations:**	48
Model:	SARIMAX(0, 1, 2)x(1, 0, [1], 12)	**Log Likelihood**	-229.813
Date:	Sun, 13 Nov 2022	**AIC**	469.625
Time:	05:00:41	**BIC**	478.876
Sample:	01-01-2015	**HQIC**	473.106
	- 12-01-2018		
Covariance Type:	opg		

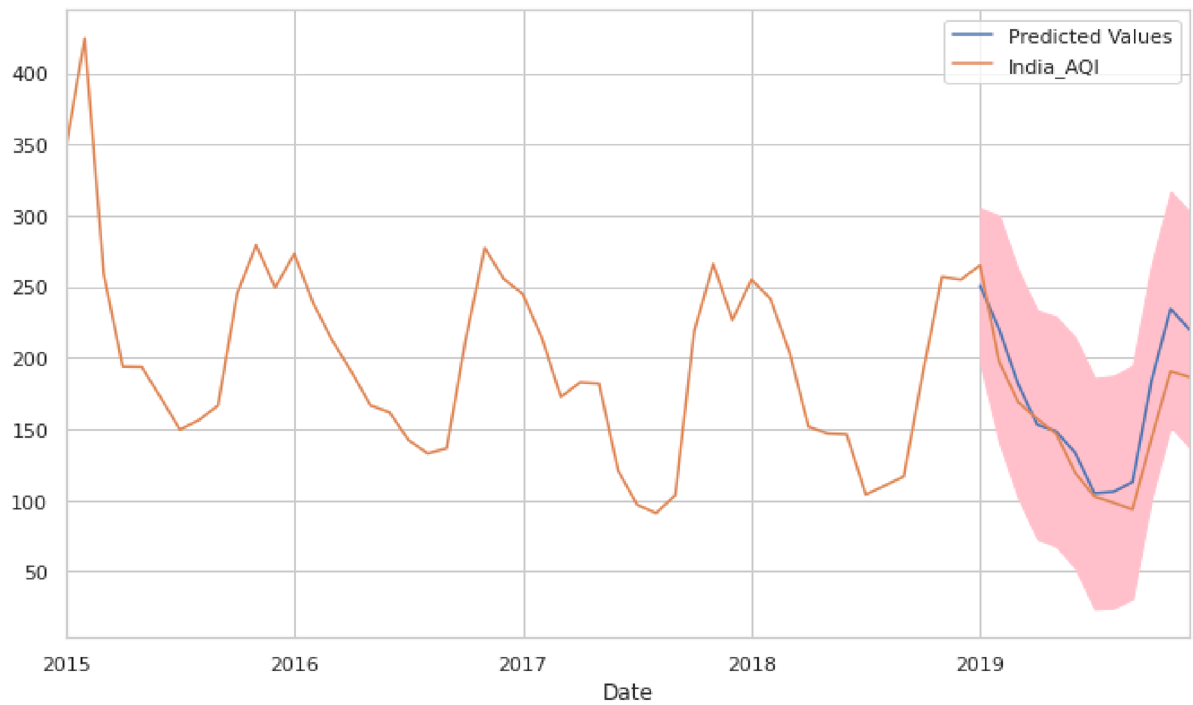
	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	0.0646	0.612	0.106	0.916	-1.135	1.264
ma.L2	-0.9325	0.600	-1.553	0.120	-2.109	0.244
ar.S.L12	0.9183	0.097	9.440	0.000	0.728	1.109
ma.S.L12	-0.4473	0.301	-1.485	0.138	-1.038	0.143
sigma2	767.0398	453.370	1.692	0.091	-121.549	1655.629

Ljung-Box (L1) (Q):	0.23	**Jarque-Bera (JB):**	3.61
Prob(Q):	0.63	**Prob(JB):**	0.16
Heteroskedasticity (H):	0.24	**Skew:**	-0.66
Prob(H) (two-sided):	0.01	**Kurtosis:**	3.28

PLOTTING THE PREDICTED VS ACTUAL AQI

- predict the next 12 months values to compare with the test dataset
- find the confidence intervals
- plot the forecasted mean data for the next 12 months and the confidence interval
- plotting the actual value from test data

```python
[16] fig, ax= plt.subplots(figsize=(10,6))

     #predict the next 12 months values to compare with the test dataset
     forecasts = results.get_forecast(steps=12, dynamic=True)

     #find the confidence intervals
     confidence_intervals=forecasts.conf_int()
     lower_limits = confidence_intervals.loc[:,'lower India_AQI']
     upper_limits = confidence_intervals.loc[:,'upper India_AQI']

     #plot the forecasted mean data for the next 12 months and the confidence interval
     forecasts.predicted_mean.plot(legend=True, ax=ax, label ='Predicted Values')
     plt.fill_between(confidence_intervals.index, lower_limits, upper_limits, color='pink')

     #plotting the actual value from test data
     test_data.plot(legend=True, ax=ax)
```



CALCULATED THE RMSE AND MAPE VALUES:

```
[17] from sklearn.metrics import mean_squared_error

     test= cities['India_AQI']['2019-01':'2019-12']
     RMSE=np.sqrt(mean_squared_error(forecasts.predicted_mean,test))
     print('RMSE = ',RMSE)

     y_true=test
     y_pred= forecasts.predicted_mean
     mape= np.mean(np.abs((y_true - y_pred) / y_true)) * 100
     print('MAPE = ', mape)

     RMSE =  22.754818000512174
     MAPE =  11.639901236022794
```

INFERENCE DRAWN:

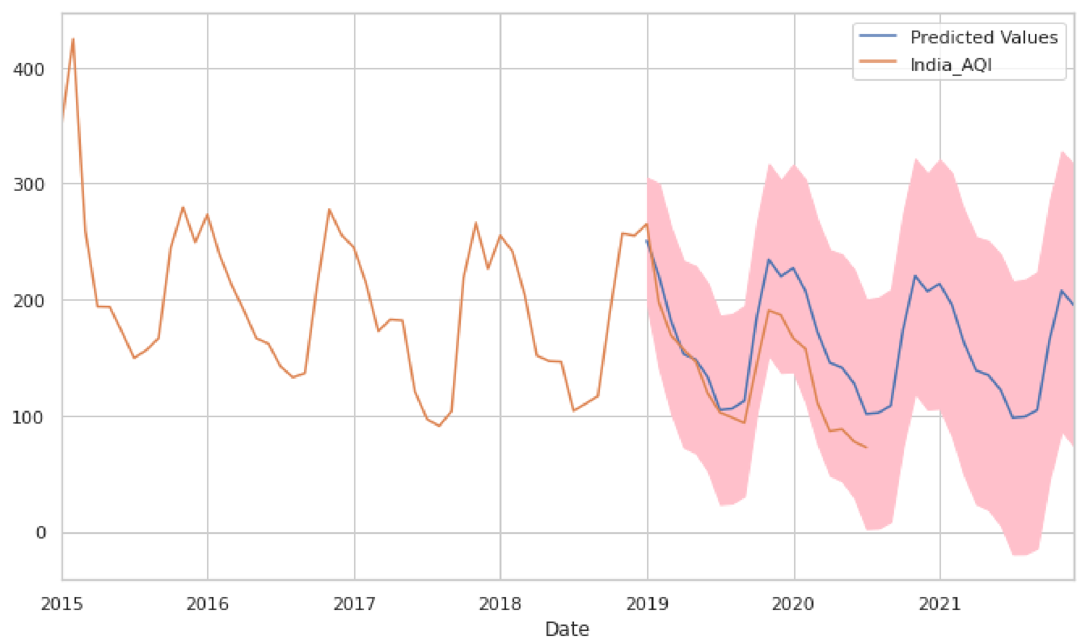
We see that the model has an RMSE of 22.75 on the test data set. Now, we can use this model to predict values into the future.

We'll be forecasting AQI values for 2021. However, 2020 yielded unexpected AQI values owing to the lockdown imposed due to COVID-19, as we saw earlier. So our prediction might have a wider margin of error to be considered.

FURTHER FORECAST FOR YEARS 2020 AND 2021

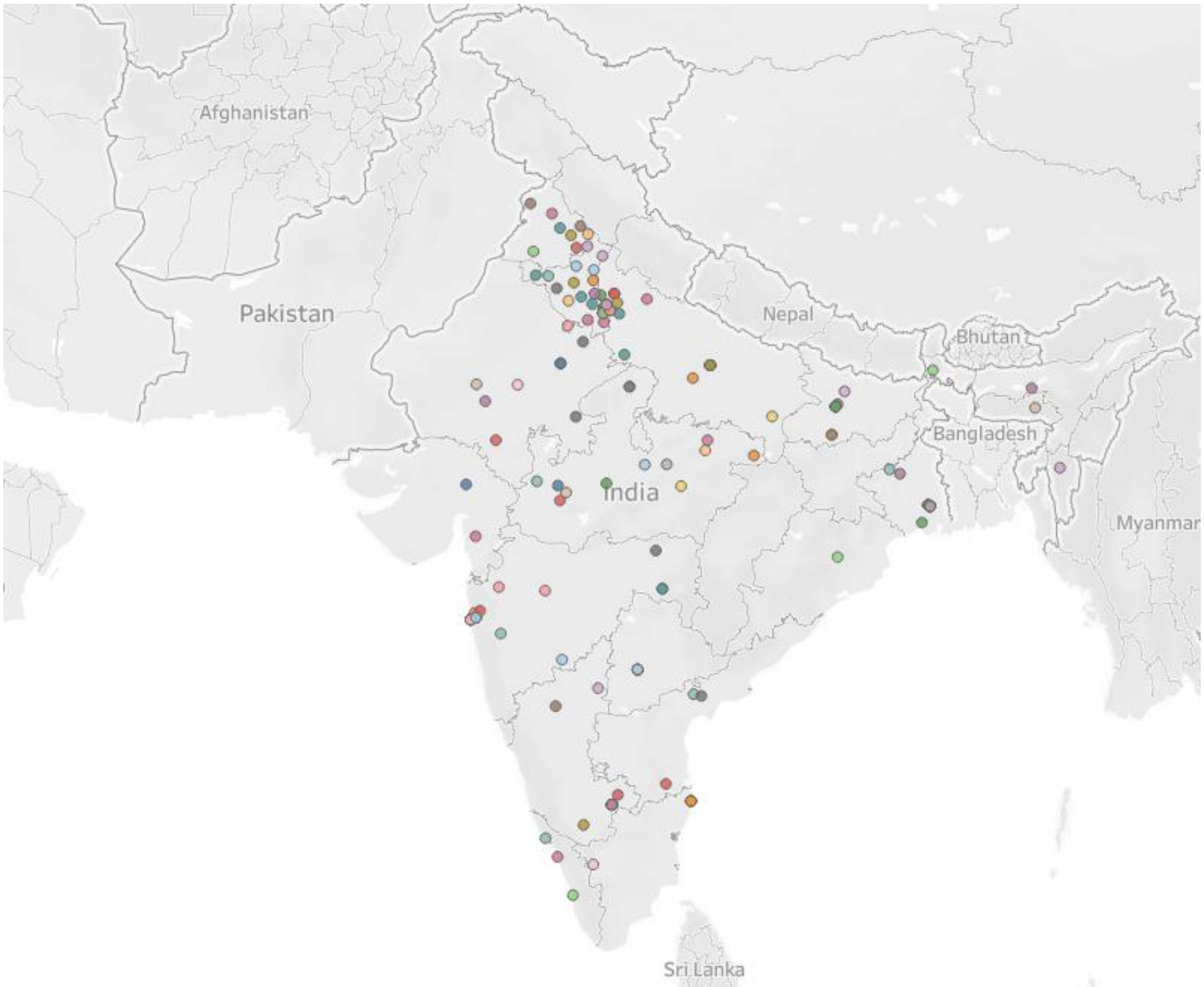
- Calculated the confidence intervals
- Plotted the forecasted data
- Plotted the confidence interval as the shaded area
- Plot India's AQI Data

```
fig, ax= plt.subplots(figsize=(10,6))

#predict the next 12 months values to compare with the test dataset
forecasts = results.get_forecast(steps=12, dynamic=True)

#find the confidence intervals
confidence_intervals=forecasts.conf_int()
lower_limits = confidence_intervals.loc[:,'lower India_AQI']
upper_limits = confidence_intervals.loc[:,'upper India_AQI']

#plot the forecasted mean data for the next 12 months and the confidence interval
forecasts.predicted_mean.plot(legend=True, ax=ax, label ='Predicted Values')
plt.fill_between(confidence_intervals.index, lower_limits, upper_limits, color='pink')

#plotting the actual value from test data
test_data.plot(legend=True, ax=ax)
```



INFERENCE:

We decide to go for a SARIMAX model because of the high seasonality that is evident in the visualizations that we have plotted. Also, we tried the random forest classifier and XGBoost which did not yield a good result. For real time AQI situations, it is better to use short term forecasting like SARIMAX because of the constant changes in weather every day. Well, no two days are the same!!



PRESENCE OF POLLUTION CONTROL BOARD (PCB) CENTERS ACROSS INDIA RESPONSIBLE FOR DATA COLLECTION

INFERENCE:

The centres are sparsely scattered. We need more of such centres .

CONCLUSION:

- **One of the key takeaways for us was the huge amount of inconsistent data in certain features thus indicating the lack of accuracy and precision with which the data is recorded.**

- **Additionally, it raises concerns about the infrastructure of meteorological facilities.**

- **The seasonality in the data highly suggests the higher concentration of pollutants in air during winter months rather than summer or monsoon months.**
  (This explains why Delhi hits the news headlines every year)

- **Indian festivals could be a legitimate reason for the seasonality.**

- **Other reason could be burning of more fuels and biomass to keep warm which causes more pollution than usual**

- **During winters the planetary boundary layer is thinner as the cooler air near the earth's surface is dense. The cooler air is trapped under the warm air above that forms a kind of atmospheric 'lid'. This phenomenon is called winter inversion. Since the vertical mixing of air happens only within this layer, the pollutants released lack enough space to disperse in the atmosphere.**

- **So a key of local,geographical and climatic factors play a role on the AQI of place.**

SPECIAL THANKS TO SURESH JAMADAGNI SIR FOR ALL THE SUPPORT AND EXTENDED DEADLINES!!