# Rohit Chougule

# Air Quality Index in India

The Real-time Air Quality Index of different monitoring stations across India is obtained by the API provided by the Ministry of Environment & Forests, India and Central Pollution Control Board, India.
The API gives the real-time data for the hour which has the list of monitored pollutants like Carbon Monoxide(CO), Sulphur Dioxide(SO2), Nitrogen Dioxide(NO2), Particulate Matter(PM10 and PM2.5) and Ozone(O3).
The stations monitor Air pollution from different sectors, viz- Industrial Air Pollution, Residential Air Pollution, Vehicular Air Pollution, Environment, and Forest.

## Data Collection:

The source of the Data is Real time Air Quality Index from various locations. (https://data.gov.in/resources/real-time-air-quality-index-various-locations/api#/Resource/get_resource_3b01bcb8_0b14_4abf_b6f2_c1bfd384ba69)
The API gives the data for the current hour Air Quality Index of various stations. I started the collection of Data on 23/02/2020 by adding a cron-job scheduler on a linux server. The data collected is till 28th March 2020.
I created a short shell script which I could run as a cron-job on a linux machine, that will fetch the hourly data daily and append to a Master csv file which is further used for the Analysis.

**Bash Script (Script also attached in the Project Folder):**

In [1]:

```
%%script false

#!/bin/bash
var=($(date +%s))
#var is a temporary variable that gets the current time stamp in milliseconds to give u
myPATH="/home/"
#myPath is the folder for storing the temporary as well as the master csv file
echo $var
temp=$(wget -O $myPATH $var.csv "https://api.data.gov.in/resource/3b01bcb8-0b14-4abf-b6
#temp stores the file at myPATH/var.csv as a temporary file every hour the cron job is
cat $myPATH $var.csv >> ${myPATH}AirQualityData.csv
#Finally, the temporary file is appended to the master CSV raw data file
```

```
Couldn't find program: 'false'
```

**API URL with the key registered on the portal:**

https://api.data.gov.in/resource/3b01bcb8-0b14-4abf-b6f2-c1bfd384ba69?api-key=WRITE_YOUR_API_KEY&format=csv&offset=0&limit=10000 (https://api.data.gov.in/resource/3b01bcb8-0b14-4abf-b6f2-c1bfd384ba69?api-key=WRITE_YOUR_API_KEY&format=csv&offset=0&limit=10000)

***Another approach for fetching the same data will be running the below Python script on a cron-job:***

In [2]:

```
## All import statements for data pre-preocessing- collection, manipulation, transformo
import csv
import requests
import pandas as pd
```

In [3]:

```
#The below code can also be used to fetch raw data from API Data source by adding the p
#below function to the cron-job scheduler.

api_url = "https://api.data.gov.in/resource/"
resource = "3b01bcb8-0b14-4abf-b6f2-c1bfd384ba69"
#api_key is obtained after registering on the portal: https://api.data.gov.in
api_key = "YOUR_API_KEY"

csv_url = api_url+resource+"?api-key="+api_key+"&format=csv&offset=0&limit=10000"

#csv-url with appropriate link + resource id + api key and the desired format

# request_data() function will create a session with the csv_url and fetch the data fro

def request_data():
    with requests.Session() as mySession:
        download = mySession.get(csv_url)

        current_data = download.content.decode('utf-8')

        # Append the current session file to the existing AirQuality Data Master file e
        with open("AirQualityData_CurrentSample_Collection.csv",'a') as file_append:
            file_append.write(current_data)

request_data()
```

## Storing the collected data in appropriate format

In [4]:

```python
# The raw data file- AirQualityData.csv which appends hourly data every day, appends al
# The raw headers that are appended every time the file is written needs to removed fro

def read_raw_data(raw_file):
    count = 0
    data_headers = ""
    with open(raw_file) as fin:
        clean_fin = open("AirQualityData_Formatted.csv","w", newline="")
        for line in fin:
            if (line[:2]=="id" and count==0):
                count=1
                data_headers = line
                clean_fin.write(data_headers)
            elif (line[:2]!="id"):
                clean_fin.write(line)
    clean_fin.close()

# The above function ensures that the raw data file is read and only lines with the act
# header for the entire file

raw_file = 'Data/AirQualityData.csv'

read_raw_data(raw_file)

#The formatted data is written to a new file- AirQualityData_Formatted.csv
```

## Loading the Data Frame for preparation(cleaning/transforming/normalizing):

#### Reading the formatted file into a pandas dataframe

In [5]:

```
AQ_df=pd.read_csv('AirQualityData_Formatted.csv')

AQ_df.head()
```

Out[5]:

| | id | country | state | city | station | last_update | pollutant_id | pollutant_min |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | India | Andhra_Pradesh | Amaravati | Secretariat, Amaravati - APPCB | 23-02-2020 12:00:00 | PM2.5 | 20.0 |
| **1** | 2 | India | Andhra_Pradesh | Amaravati | Secretariat, Amaravati - APPCB | 23-02-2020 12:00:00 | PM10 | 34.0 |
| **2** | 3 | India | Andhra_Pradesh | Amaravati | Secretariat, Amaravati - APPCB | 23-02-2020 12:00:00 | NO2 | 9.0 |
| **3** | 4 | India | Andhra_Pradesh | Amaravati | Secretariat, Amaravati - APPCB | 23-02-2020 12:00:00 | NH3 | 2.0 |
| **4** | 5 | India | Andhra_Pradesh | Amaravati | Secretariat, Amaravati - APPCB | 23-02-2020 12:00:00 | SO2 | 2.0 |

In [6]:

```
AQ_df.tail()
```

Out[6]:

| | id | country | state | city | station | last_update | pollutant_id | pollutant_min |
|---|---|---|---|---|---|---|---|---|
| **1140279** | 1216 | India | West_Bengal | Kolkata | Victoria, Kolkata - WBPCB | 30-03-2020 01:00:00 | NO2 | 16.0 |
| **1140280** | 1217 | India | West_Bengal | Kolkata | Victoria, Kolkata - WBPCB | 30-03-2020 01:00:00 | NH3 | 6.0 |
| **1140281** | 1218 | India | West_Bengal | Kolkata | Victoria, Kolkata - WBPCB | 30-03-2020 01:00:00 | SO2 | 11.0 |
| **1140282** | 1219 | India | West_Bengal | Kolkata | Victoria, Kolkata - WBPCB | 30-03-2020 01:00:00 | CO | 38.0 |
| **1140283** | 1220 | India | West_Bengal | Kolkata | Victoria, Kolkata - WBPCB | 30-03-2020 01:00:00 | OZONE | 54.0 |

**Description of Data**

In [7]:

```
AQ_df.describe()
```

Out[7]:

| | id | pollutant_min | pollutant_max | pollutant_avg | pollutant_unit |
|---|---|---|---|---|---|
| **count** | 1.140284e+06 | 1.067754e+06 | 1.067754e+06 | 1.067754e+06 | 0.0 |
| **mean** | 6.643728e+02 | 2.586871e+01 | 9.187309e+01 | 5.142478e+01 | NaN |
| **std** | 3.840458e+02 | 2.931316e+01 | 9.071844e+01 | 5.228353e+01 | NaN |
| **min** | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | NaN |
| **25%** | 3.320000e+02 | 6.000000e+00 | 2.600000e+01 | 1.300000e+01 | NaN |
| **50%** | 6.640000e+02 | 1.400000e+01 | 6.700000e+01 | 3.400000e+01 | NaN |
| **75%** | 9.960000e+02 | 3.600000e+01 | 1.200000e+02 | 7.100000e+01 | NaN |
| **max** | 1.407000e+03 | 2.950000e+02 | 5.000000e+02 | 3.660000e+02 | NaN |

## Data Cleaning & Transformation

As the data returned by the API is hourly and from the glimpse of the data above, its observed that the time stamp in the data is ambiguous. The last_update column has the time-stamp for the collected data, however, it does not specify if its the day time or the evening time(AM/PM)

In [8]:

```python
# Sorting the dataframe by pollutant_id and station name to make the last_update columr
AQ_df=AQ_df.sort_values(by=["pollutant_id", "station"])

#Adding the column Timestamp which will be used in Analysis and the format/data-type oj

AQ_df["time_stamp"] = ""

myDictionary = {}

# There is a unique combination of last_update, pollutant_id, station for every hour.
# This unique combination can be used to identify the AM/PM (Morning/Evening) records.

# The data collection was started in morning time, hence it will be accurate for the fi
# the date-time will be identified and the time-stamp column will be updated accordingl
counter=0
for index, row in AQ_df.iterrows():
    val = str(str(row["last_update"])+"_"+str(row["pollutant_id"])+"_"+str(row["statior
    counter+=1
    if (val in myDictionary.keys()):
         # If second occurrence of the unique combination, add to the dictionary and ap
        ts = str(row["last_update"])
        AQ_df.at[index, "time_stamp"] = ts + (" PM")
        if counter <=10:
            print("Transforming date data for: ", val)
    else:
        # If first occurrence of the unique combination, add to the dictionary and appe
        myDictionary.update({
            val : val
        })
        ts = str(row["last_update"])
        AQ_df.at[index, "time_stamp"] = ts + (" AM")
        if counter<=10:
            print("Transforming date data for: ", val)
        elif counter==11:
            print("Output limited to 10 prints transforming rows....")
```

```
Transforming date data for:  23-02-2020 12:00:00_CO_Adarsh Nagar, Jaipur - R
SPCB
Transforming date data for:  23-02-2020 01:00:00_CO_Adarsh Nagar, Jaipur - R
SPCB
Transforming date data for:  23-02-2020 02:00:00_CO_Adarsh Nagar, Jaipur - R
SPCB
Transforming date data for:  23-02-2020 03:00:00_CO_Adarsh Nagar, Jaipur - R
SPCB
Transforming date data for:  23-02-2020 04:00:00_CO_Adarsh Nagar, Jaipur - R
SPCB
Transforming date data for:  23-02-2020 05:00:00_CO_Adarsh Nagar, Jaipur - R
SPCB
Transforming date data for:  23-02-2020 06:00:00_CO_Adarsh Nagar, Jaipur - R
SPCB
Transforming date data for:  23-02-2020 07:00:00_CO_Adarsh Nagar, Jaipur - R
SPCB
Transforming date data for:  23-02-2020 08:00:00_CO_Adarsh Nagar, Jaipur - R
SPCB
Transforming date data for:  23-02-2020 09:00:00_CO_Adarsh Nagar, Jaipur - R
SPCB
Output limited to 10 prints transforming rows....
```

**Data after adding a transfromed time_stamp column to existing data frame**

In [9]:

```
AQ_df.head()
```

Out[9]:

| | id | country | state | city | station | last_update | pollutant_id | pollutant_min | pollutan |
|---|---|---|---|---|---|---|---|---|---|
| **1409** | 971 | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 12:00:00 | CO | 20.0 | |
| **2704** | 969 | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 01:00:00 | CO | 20.0 | |
| **4001** | 971 | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 02:00:00 | CO | 22.0 | |
| **5298** | 971 | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 03:00:00 | CO | 25.0 | |
| **6595** | 971 | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 04:00:00 | CO | 25.0 | |

In [10]:

```
# Changing the data type of the new time-stamp column to datetime
AQ_df["time_stamp"] = pd.to_datetime(AQ_df["time_stamp"], dayfirst=True)
```

In [11]:

```
# Drop the columns id and, pollutant_unit as they donot have any significance
AQ_df=AQ_df.drop(['id', 'pollutant_unit'], axis=1)
AQ_df.head()
```

Out[11]:

| | country | state | city | station | last_update | pollutant_id | pollutant_min | pollutant_max |
|---|---|---|---|---|---|---|---|---|
| **1409** | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 12:00:00 | CO | 20.0 | 59. |
| **2704** | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 01:00:00 | CO | 20.0 | 59. |
| **4001** | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 02:00:00 | CO | 22.0 | 59. |
| **5298** | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 03:00:00 | CO | 25.0 | 59. |
| **6595** | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 04:00:00 | CO | 25.0 | 59. |

In [12]:

```python
# The source column last_update can be dropped from the dataframe now:

# AQ_df=AQ_df.drop(["last_update"], axis=1)

# Resetting the index of the data:
AQ_df.reset_index(drop=True, inplace=True)

#Description of data:
AQ_df.describe()
```

Out[12]:

| | pollutant_min | pollutant_max | pollutant_avg |
|---|---|---|---|
| **count** | 1.067754e+06 | 1.067754e+06 | 1.067754e+06 |
| **mean** | 2.586871e+01 | 9.187309e+01 | 5.142478e+01 |
| **std** | 2.931316e+01 | 9.071844e+01 | 5.228353e+01 |
| **min** | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 |
| **25%** | 6.000000e+00 | 2.600000e+01 | 1.300000e+01 |
| **50%** | 1.400000e+01 | 6.700000e+01 | 3.400000e+01 |
| **75%** | 3.600000e+01 | 1.200000e+02 | 7.100000e+01 |
| **max** | 2.950000e+02 | 5.000000e+02 | 3.660000e+02 |

In [13]:

```python
AQ_df.head()
```

Out[13]:

| | country | state | city | station | last_update | pollutant_id | pollutant_min | pollutant_max | p |
|---|---|---|---|---|---|---|---|---|---|
| **0** | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 12:00:00 | CO | 20.0 | 59.0 | |
| **1** | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 01:00:00 | CO | 20.0 | 59.0 | |
| **2** | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 02:00:00 | CO | 22.0 | 59.0 | |
| **3** | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 03:00:00 | CO | 25.0 | 59.0 | |
| **4** | India | Rajasthan | Jaipur | Adarsh Nagar, Jaipur - RSPCB | 23-02-2020 04:00:00 | CO | 25.0 | 59.0 | |

In [14]:

```
AQ_df.tail()
```

Out[14]:

| | country | state | city | station | last_update | pollutant_id | pollutant_min | po |
|---|---|---|---|---|---|---|---|---|
| **1140279** | India | Telangana | Hyderabad | Zoo Park, Hyderabad - TSPCB | 29-03-2020 09:00:00 | SO2 | 1.0 | |
| **1140280** | India | Telangana | Hyderabad | Zoo Park, Hyderabad - TSPCB | 29-03-2020 10:00:00 | SO2 | 1.0 | |
| **1140281** | India | Telangana | Hyderabad | Zoo Park, Hyderabad - TSPCB | 29-03-2020 11:00:00 | SO2 | 1.0 | |
| **1140282** | India | Telangana | Hyderabad | Zoo Park, Hyderabad - TSPCB | 30-03-2020 12:00:00 | SO2 | 1.0 | |
| **1140283** | India | Telangana | Hyderabad | Zoo Park, Hyderabad - TSPCB | 30-03-2020 01:00:00 | SO2 | 1.0 | |

In [15]:

```
#Number of Null/NA values in the dataset
AQ_df.isnull().sum()
```

Out[15]:

```
country              0
state                0
city                 0
station              0
last_update          0
pollutant_id         0
pollutant_min    72530
pollutant_max    72530
pollutant_avg    72530
time_stamp           0
dtype: int64
```

**Handling the NA/Missing values in the data frame**

The NA/missing values in the dataset are handled after carefully understanding the dataset.
The NA values are replaced by taking mean by grouping each combination to maintain maximum possible accuracy-

- station, pollutant, and timestamp
- station, pollutant, and date
- station, pollutant

  Even if there are data values with missing data after the above combinations, the below combinations are implemented:
- city, pollutant, timestamp(date+time)

- city, pollutant, time
- city, pollutant, date
- city, pollutant

If the data has missing values even after the above all combinations, it could be quite possible that the station never collects that data. For example, few stations are not configured to collect the pollution level of all the seven pollutants.

In [16]:

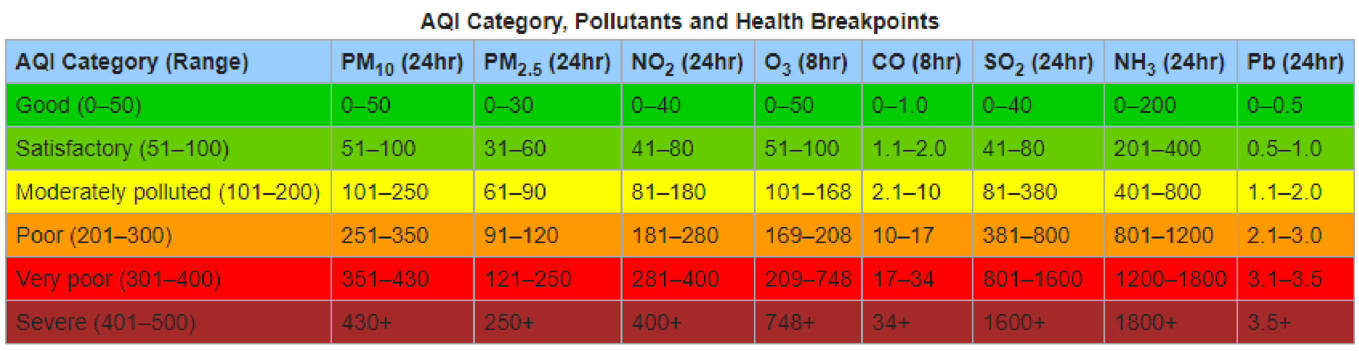
```
# Adding date and time columns separately to the dataframe, which can be crucial in fur

AQ_df["date"] = AQ_df["time_stamp"].dt.date
AQ_df["time"] = AQ_df["time_stamp"].dt.time

print("Number of missing values in the data set before processing: ")
print("*******************************************************************  ")
print(AQ_df.isnull().sum())
print("*******************************************************************")
# Aggregates and fills null values considering the same time, station and pollutant typ
AQ_df["pollutant_min"] = AQ_df.groupby(["station", "pollutant_id", "time"])["pollutant_
AQ_df["pollutant_max"] = AQ_df.groupby(["station", "pollutant_id", "time"])["pollutant_
AQ_df["pollutant_avg"] = AQ_df.groupby(["station", "pollutant_id", "time"])["pollutant_

# Aggregates and fills null values considering the same day, station and pollutant type
AQ_df["pollutant_min"] = AQ_df.groupby(["station", "pollutant_id", "date"])["pollutant_
AQ_df["pollutant_max"] = AQ_df.groupby(["station", "pollutant_id", "date"])["pollutant_
AQ_df["pollutant_avg"] = AQ_df.groupby(["station", "pollutant_id", "date"])["pollutant_

# Aggregates and fills null values considering the same station and pollutant type
AQ_df["pollutant_min"] = AQ_df.groupby(["station", "pollutant_id"])["pollutant_min"].tr
AQ_df["pollutant_max"] = AQ_df.groupby(["station", "pollutant_id"])["pollutant_max"].tr
AQ_df["pollutant_avg"] = AQ_df.groupby(["station", "pollutant_id"])["pollutant_avg"].tr

# Aggregates and fills null values considering the same time-stamp(day and time), city
AQ_df["pollutant_min"] = AQ_df.groupby(["city", "pollutant_id", "time_stamp"])["polluta
AQ_df["pollutant_max"] = AQ_df.groupby(["city", "pollutant_id", "time_stamp"])["polluta
AQ_df["pollutant_avg"] = AQ_df.groupby(["city", "pollutant_id", "time_stamp"])["polluta

# Aggregates and fills null values considering the same time, city and pollutant type
AQ_df["pollutant_min"] = AQ_df.groupby(["city", "pollutant_id", "time"])["pollutant_mi
AQ_df["pollutant_max"] = AQ_df.groupby(["city", "pollutant_id", "time"])["pollutant_ma
AQ_df["pollutant_avg"] = AQ_df.groupby(["city", "pollutant_id", "time"])["pollutant_av

# Aggregates and fills null values considering the same day, city and pollutant type
AQ_df["pollutant_min"] = AQ_df.groupby(["city", "pollutant_id", "date"])["pollutant_mi
AQ_df["pollutant_max"] = AQ_df.groupby(["city", "pollutant_id", "date"])["pollutant_ma
AQ_df["pollutant_avg"] = AQ_df.groupby(["city", "pollutant_id", "date"])["pollutant_av

# Aggregates and fills null values considering the same city and pollutant type
AQ_df["pollutant_min"] = AQ_df.groupby(["city", "pollutant_id"])["pollutant_min"].trans
AQ_df["pollutant_max"] = AQ_df.groupby(["city", "pollutant_id"])["pollutant_max"].trans
AQ_df["pollutant_avg"] = AQ_df.groupby(["city", "pollutant_id"])["pollutant_avg"].trans

print("Number of missing values in the data set after processing: ")
print("*******************************************************************")
print(AQ_df.isnull().sum())
print("*******************************************************************")

# Drops the null value from the data frame as there is no data collected and no further
AQ_df=AQ_df.dropna()

print("Final number of missing values in the data set after processing and removing rec
print("*******************************************************************")
print(AQ_df.isnull().sum())
print("*******************************************************************")
```

```
Number of missing values in the data set before processing:
*****************************************************************
country              0
state                0
city                 0
station              0
last_update          0
pollutant_id         0
pollutant_min    72530
pollutant_max    72530
pollutant_avg    72530
time_stamp           0
date                 0
time                 0
dtype: int64
*****************************************************************
Number of missing values in the data set after processing:
*****************************************************************
country             0
state               0
city                0
station             0
last_update         0
pollutant_id        0
pollutant_min    2942
pollutant_max    2942
pollutant_avg    2942
time_stamp          0
date                0
time                0
dtype: int64
*****************************************************************
Final number of missing values in the data set after processing and removi
ng records with missing data:
*****************************************************************
country          0
state            0
city             0
station          0
last_update      0
pollutant_id     0
pollutant_min    0
pollutant_max    0
pollutant_avg    0
time_stamp       0
date             0
time             0
dtype: int64
*****************************************************************
```

## Data Analysis :

**As the data is now cleaned, we can further analyse to gain some insights on the Air Quality Index of India**

Before we proceed with the analysis of the Air Quality Index, the below image gives overview of parameters for evaluating the Air Quality Index depending on the Pollutant Levels:



AQI Category, Pollutants and Health Breakpoints

| AQI Category (Range) | $PM_{10}$ (24hr) | $PM_{2.5}$ (24hr) | $NO_2$ (24hr) | $O_3$ (8hr) | CO (8hr) | $SO_2$ (24hr) | $NH_3$ (24hr) | Pb (24hr) |
|---|---|---|---|---|---|---|---|---|
| Good (0–50) | 0–50 | 0–30 | 0–40 | 0–50 | 0–1.0 | 0–40 | 0–200 | 0–0.5 |
| Satisfactory (51–100) | 51–100 | 31–60 | 41–80 | 51–100 | 1.1–2.0 | 41–80 | 201–400 | 0.5–1.0 |
| Moderately polluted (101–200) | 101–250 | 61–90 | 81–180 | 101–168 | 2.1–10 | 81–380 | 401–800 | 1.1–2.0 |
| Poor (201–300) | 251–350 | 91–120 | 181–280 | 169–208 | 10–17 | 381–800 | 801–1200 | 2.1–3.0 |
| Very poor (301–400) | 351–430 | 121–250 | 281–400 | 209–748 | 17–34 | 801–1600 | 1200–1800 | 3.1–3.5 |
| Severe (401–500) | 430+ | 250+ | 400+ | 748+ | 34+ | 1600+ | 1800+ | 3.5+ |

Image Source: https://cpcb.nic.in/ (https://cpcb.nic.in/)

## Summary Statistics of Pollution levels in cities and States:

In [17]:

```
byState=AQ_df.groupby(["state", "date"], as_index=False)["pollutant_avg"].mean()

avgbyState=byState.groupby(["state"])["pollutant_avg"].mean()

avgbyState=avgbyState.to_frame()
print("-----------------------------------------------------------------")
print("\033[0m"+"State with maximum average pollution daily (considering all Pollutants
print("\033[1m"+"\033[91m"+avgbyState["pollutant_avg"].idxmax()," : ", avgbyState["pol

print("\033[0m"+"-----------------------------------------------------------------")
print("State with minimum average pollution daily (considering all Pollutants): ")
print("\033[1m"+"\033[92m"+avgbyState["pollutant_avg"].idxmin()," : ", avgbyState["pol

byCity=AQ_df.groupby(["city","date"], as_index=False)["pollutant_avg"].mean()

avgbyCity=byCity.groupby(["city"])["pollutant_avg"].mean()

avgbyCity=avgbyCity.to_frame()
print("\033[0m"+"-----------------------------------------------------------------")
print("City with maximum average pollution daily (considering all Pollutants): ")
print("\033[1m"+"\033[91m"+avgbyCity["pollutant_avg"].idxmax()," : ", avgbyCity["pollut

print("\033[0m"+"-----------------------------------------------------------------")
print("City with minimum average pollution daily (considering all Pollutants): ")

print("\033[1m"+"\033[92m"+avgbyCity["pollutant_avg"].idxmin(), " : ", avgbyCity["pollu
print("\033[0m"+"-----------------------------------------------------------------")
```

```
-----------------------------------------------------------------
State with maximum average pollution daily (considering all Pollutants):
Jharkhand  :  65.64897248803005
-----------------------------------------------------------------
State with minimum average pollution daily (considering all Pollutants):
Meghalaya  :  23.705677732676097
-----------------------------------------------------------------
City with maximum average pollution daily (considering all Pollutants):
Greater_Noida  :  85.86069806061994
-----------------------------------------------------------------
City with minimum average pollution daily (considering all Pollutants):
Shillong  :  23.705677732676097
-----------------------------------------------------------------
```

**Insights**:
The above cell gives a general idea about States and cities with highest and lowest average pollution considering all the pollutants. We cannot compare it on a certain scale or unit of pollutant as it considers all the pollutants.
It just gives a summary of cities & states with highest & lowest average pollution in consideration with other cities & states in India.

In [18]:

```
# All import statements used in creating visualisations
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from scipy import stats
import matplotlib
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px
```

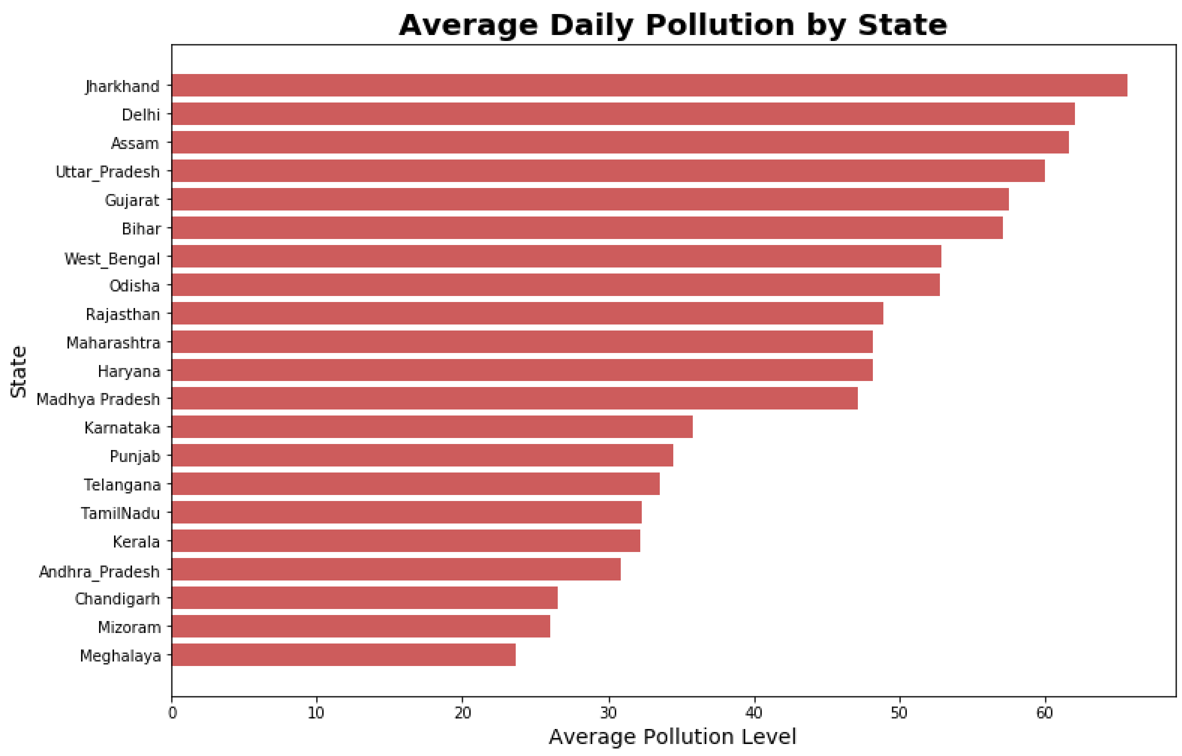
## What is the average state wise pollution across all states?

In [19]:

```
avgbyState=avgbyState.sort_values('pollutant_avg', ascending=True)
plt.figure(figsize=(12,8))
plt.title("Average Daily Pollution by State", fontsize=20, fontweight="bold")
plt.barh(avgbyState.index, avgbyState["pollutant_avg"], align="center", color="indianre
plt.ylabel("State", fontsize=14)
plt.xlabel("Average Pollution Level", fontsize=14)
```

Out[19]:

```
Text(0.5, 0, 'Average Pollution Level')
```



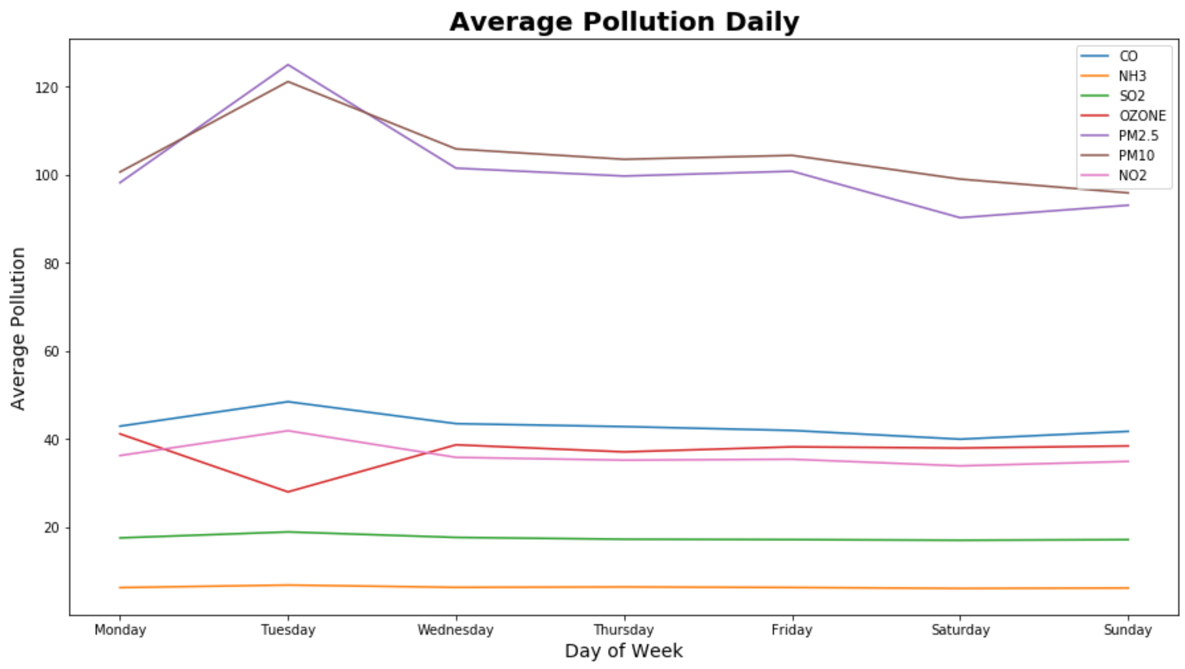
**Insights:**

The above bar chart compares state-wise pollution level in each state in India that collects the Air Quality Index at various stations.
The states with highest pollution level are clearly those with the highest population as well as with large number of industries. Whereas the states with lowest pollution level and good air quality index are those with a large number of vegetation and forests and sparesly populated regions in India

## What is the level of pollution for each pollutant on different days of week?

In [20]:

```
AQ_df["dayofweek"] = AQ_df["time_stamp"].dt.dayofweek

avgbyDay=AQ_df.groupby(["dayofweek", "pollutant_id"], as_index=False)["pollutant_avg"].

avgbyDay=avgbyDay.pivot(index="dayofweek", columns="pollutant_id", values="pollutant_av

plt.figure(figsize=(15,8))
plt.plot(avgbyDay.index, avgbyDay["CO"], label="CO")
plt.plot(avgbyDay.index, avgbyDay["NH3"], label="NH3")
plt.plot(avgbyDay.index, avgbyDay["SO2"], label="SO2")
plt.plot(avgbyDay.index, avgbyDay["OZONE"], label="OZONE")
plt.plot(avgbyDay.index, avgbyDay["PM2.5"], label="PM2.5")
plt.plot(avgbyDay.index, avgbyDay["PM10"], label="PM10")
plt.plot(avgbyDay.index, avgbyDay["NO2"], label="NO2")
plt.xlabel('Day of Week', fontsize=14)
plt.ylabel('Average Pollution', fontsize=14)
plt.title('Average Pollution Daily', fontsize=20, fontweight="bold")
plt.legend()
x_labels= ["Monday", "Tuesday", "Wednesday","Thursday","Friday","Saturday","Sunday"]
plt.xticks(np.arange(0,7),labels=x_labels)

plt.show()
```



**Insights:**
The above line chart shows a trend on how the average Air Quality Index of all the cities various days in week, each line denotes a different pollutant. In general it can be observed that the pollution level on Weekdays is

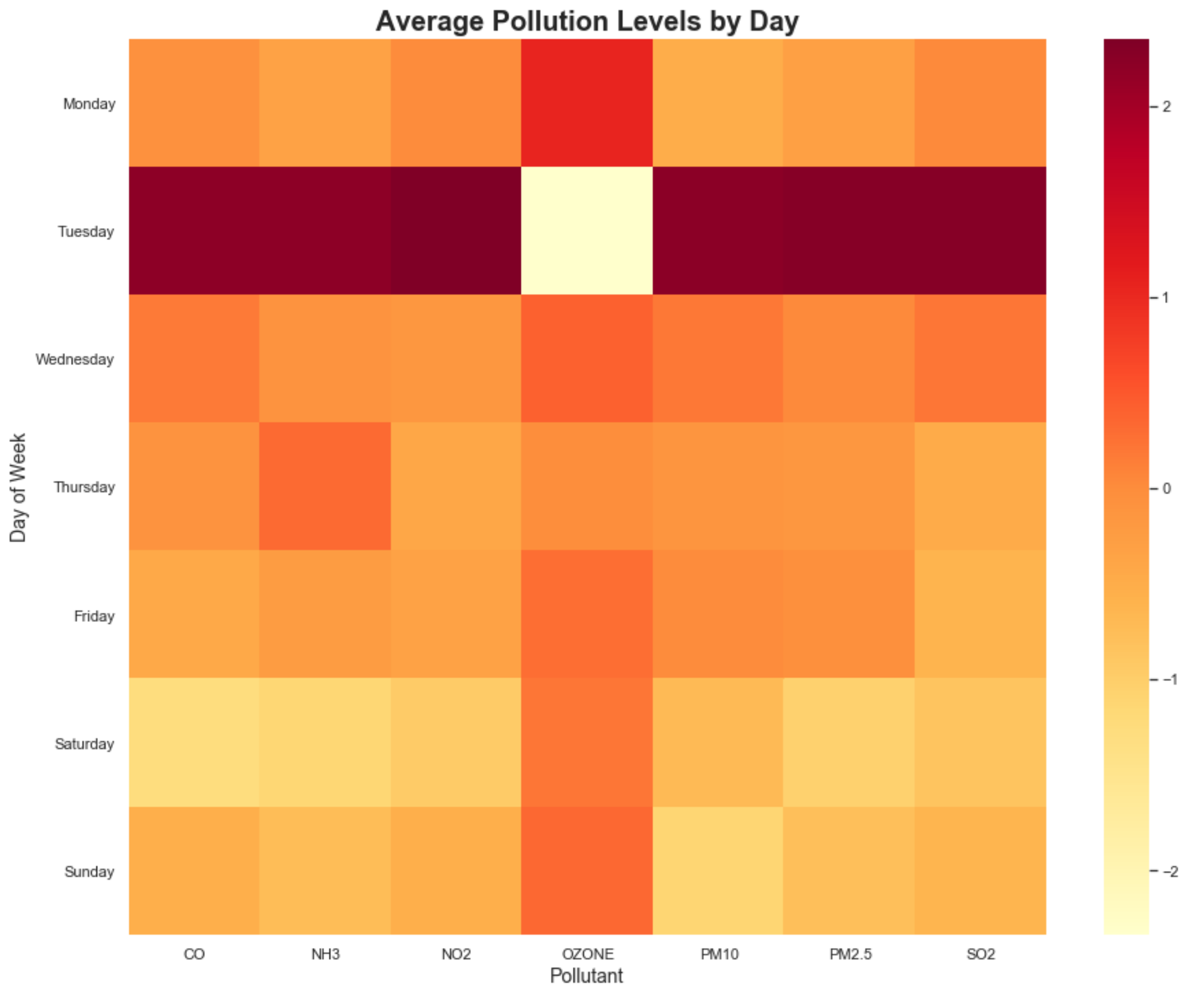
high as compared to weekends.
From this Line graph, we can identify that NH3 is the pollutant, which contributes least to the over all pollution level Index, where as PM2.5 and PM10 are the major contributors in pollution level.

In [21]:

```
avgbyDay_hm=pd.DataFrame(columns=avgbyDay.columns, index=avgbyDay.index)

for col in avgbyDay_hm.columns:
    test=stats.zscore(avgbyDay[[col]])
    avgbyDay_hm[col] = pd.DataFrame(test)

# z-score normalisation is being used to normalise the avg_pollutant column

sns.set()
f, ax = plt.subplots(figsize=(15, 12))
daysofWeek= ["Monday", "Tuesday", "Wednesday","Thursday","Friday","Saturday","Sunday"]
sns.heatmap(avgbyDay_hm, annot=False, linewidths=0, ax=ax, cmap="YlOrRd",yticklabels=da
plt.yticks(rotation=0)
plt.xlabel('Pollutant', fontsize=14)
plt.ylabel('Day of Week', fontsize=14)
plt.title('Average Pollution Levels by Day', fontsize=20, fontweight="bold")
```

Out[21]:

```
Text(0.5, 1, 'Average Pollution Levels by Day')
```



**Insights:**
The previous Line chart compared each and every pollutants national average for every day of the week. This Heatmap has the data which is normalized using zscore, so as to compare all of the pollutants on a similar

scale in comparison to their levels of all other days.
The heatmap clearly indicates that the pollution levels are higher on Weekdays and less on weekends for all pollutants.

In [22]:

```
avgbyTime=AQ_df.groupby(["time", "pollutant_id"], as_index=False)["pollutant_avg"].mear

avgbyTime=avgbyTime.pivot(index="time", columns="pollutant_id", values="pollutant_avg")

#Evaluating the pollution level by hourly basis with respect to all pollutants.
```

In [23]:

```
avgbyTime_hm=pd.DataFrame(stats.zscore(avgbyTime), columns=avgbyDay.columns)

# Getting z-score normalised results of the avg pollution level for each hour.
```

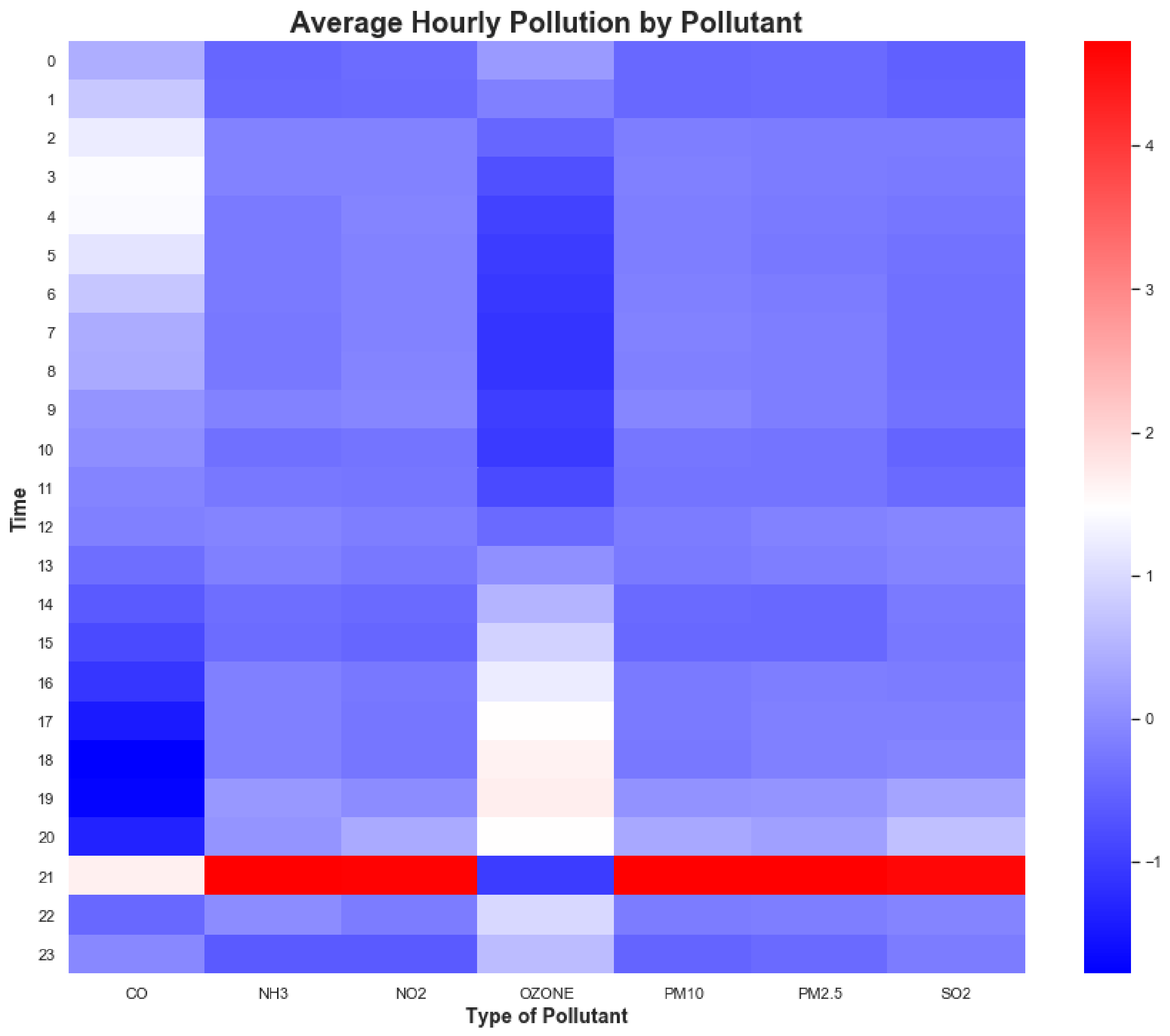
## What is the level of average hourly pollution for each pollutant?

In [24]:

```
sns.set()
f, ax = plt.subplots(figsize=(15, 12))
sns.heatmap(avgbyTime_hm, annot=False, linewidths=0, ax=ax, cmap="bwr")
plt.yticks(rotation=0)
plt.ylabel('Time', fontsize=14, fontweight="bold")
plt.xlabel('Type of Pollutant', fontsize=14, fontweight="bold")
plt.title('Average Hourly Pollution by Pollutant', fontsize=20, fontweight="bold")
```

Out[24]:

```
Text(0.5, 1, 'Average Hourly Pollution by Pollutant')
```



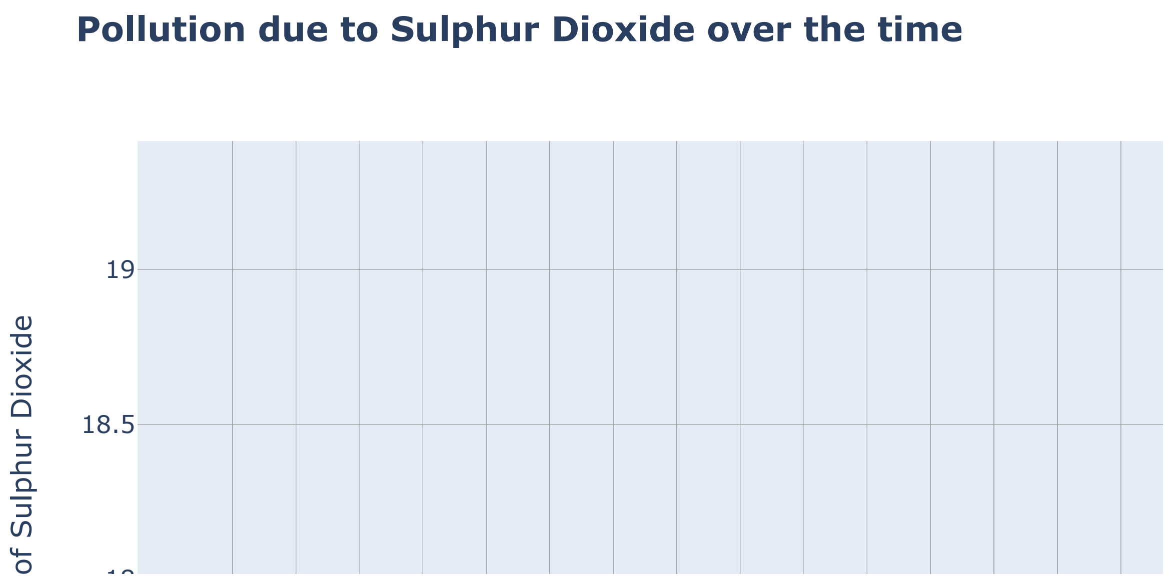
**Insights:**
The above heatmap gives general idea on how the pollution level various across the day in 24 hours for each of the pollutant.
It can be observed that most of the pollution level is high during the evening and moderate during the morning times.
The color denotes the level of the pollution, Dark blue denotes lowest and red the highest

**Sulphur Dioxide is the major constituent of vehicular air pollution.**

## How does the level of Sulphur Dioxide (SO2) vary through a day?

In [25]:

```
fig = go.Figure(data=go.Scatter(x=avgbyTime.index,
                                y=avgbyTime["SO2"],
                                mode='markers'))
fig.update_layout(title='<b>Pollution due to Sulphur Dioxide over the time</b>', yaxis_
                  xaxis_title="Time of the Day", yaxis_title="Level of Sulphur Dioxide"
fig.update_traces(mode='markers', marker_line_width=2, marker_size=10)
fig.show()
```



**Insights:**
From the above scatter plot we can find a general trend that the level of SO2 pollutant starts increasing around 8 AM in the morning and continues to increase until 9-10PM in the evening.
SO2 is one of the major constituent in the vehicular pollution, hence it is obvious to find such a trend with the SO2 pollutant during those hours in the day.

In [26]:

```
avgbyCity=AQ_df.groupby(["city","pollutant_id"], as_index=False)["pollutant_avg"].mean(

avgbyCity=avgbyCity.pivot(index="city", columns="pollutant_id", values="pollutant_avg"
```

## Which are the cities with Highest average pollution in each type of the pollutant?

From few of the below Summary tables we can identify the top 5 cities with highest average pollution in each type of pollutant.

In [27]:

```
top5_CO=pd.DataFrame(avgbyCity.sort_values("CO", ascending=False)["CO"].head(5))
top5_CO=top5_CO.round(2)

fig = go.Figure(data=[go.Table(header=dict(values=['<b>City</b>', '<b>Average Pollutior
                 cells=dict(values=[top5_CO.index,top5_CO.values]))
                     ])
fig.update_layout(
    title={
        'text': "<b>Top 5 Cities with Highest Average CO Pollutant</b>"}, width=500, he

fig.show()
```

**Top 5 Cities with Highest Average CO Pollutant**

| City | Average Pollution |
|---|---|
| Nandesari | 115.25 |
| Bathinda | 98.9 |
| Solapur | 90.69 |
| Jorapokhar | 86.91 |
| Pune | 85.12 |

In [28]:

```python
top5_NO2=pd.DataFrame(avgbyCity.sort_values("NO2", ascending=False)["NO2"].head(5))
top5_NO2=top5_NO2.round(2)

top5_NO2
fig = go.Figure(data=[go.Table(header=dict(values=['<b>City</b>', '<b>Average Pollutio
                 cells=dict(values=[top5_NO2.index,top5_NO2.values]))
                     ])
fig.update_layout(
    title={
        'text': "<b>Top 5 Cities with Highest Average NO2 Pollutant</b>"}, width=500, 

fig.show()
```

**Top 5 Cities with Highest Average NO2 Pollutant**

| **City** | **Average Pollution** |
|---|---|
| Panipat | 98.08 |
| Indore | 93.89 |
| Greater_Noida | 78.18 |
| Bhiwadi | 76.06 |
| Nandesari | 67.81 |

In [29]:

```
top5_SO2=pd.DataFrame(avgbyCity.sort_values("SO2", ascending=False)["SO2"].head(5))
top5_SO2=top5_SO2.round(2)

fig = go.Figure(data=[go.Table(header=dict(values=['<b>City</b>', '<b>Average Pollutio
                 cells=dict(values=[top5_SO2.index,top5_SO2.values]))
                     ])
fig.update_layout(
    title={
        'text': "<b>Top 5 Cities with Highest Average SO2 Pollutant</b>"}, width=500, h

fig.show()
```

**Top 5 Cities with Highest Average SO2 Pollutant**

| **City** | **Average Pollution** |
|---|---|
| Ratlam | 61.09 |
| Bhiwani | 60.48 |
| Ahmedabad | 57.79 |
| Karnal | 55.39 |
| Singrauli | 53.18 |

In [30]:

```
top5_PM2_5=pd.DataFrame(avgbyCity.sort_values("PM2.5", ascending=False)["PM2.5"].head(5
top5_PM2_5=top5_PM2_5.round(2)

fig = go.Figure(data=[go.Table(header=dict(values=['<b>City</b>', '<b>Average Pollution
                 cells=dict(values=[top5_PM2_5.index,top5_PM2_5.values]))
                     ])
fig.update_layout(
    title={
        'text': "<b>Top 5 Cities with Highest Average PM2.5 Pollutant</b>"}, width=500,

fig.show()
```

**Top 5 Cities with Highest Average PM2.5 Pollutant**

| **City** | **Average Pollution** |
|---|---|
| Greater_Noida | 198.81 |
| Bhiwadi | 196.8 |
| Muzaffarpur | 189.9 |
| Ghaziabad | 184.16 |
| Guwahati | 179.52 |

In [31]:

```python
top5_PM10=pd.DataFrame(avgbyCity.sort_values("PM10", ascending=False)["PM10"].head(5))
top5_PM10=top5_PM10.round(2)

fig = go.Figure(data=[go.Table(header=dict(values=['<b>City</b>', '<b>Average Pollutior
                 cells=dict(values=[top5_PM10.index,top5_PM10.values]))
                     ])
fig.update_layout(
    title={
        'text': "<b>Top 5 Cities with Highest Average PM10 Pollutant</b>"}, width=500,

fig.show()
```

**Top 5 Cities with Highest Average PM10 Pollutant**

| **City** | **Average Pollution** |
|---|---|
| Greater_Noida | 182.91 |
| Kalyan | 177.63 |
| Panipat | 154.65 |
| Bhiwadi | 151.34 |
| Greater Noida | 151.33 |

In [32]:

```python
top5_OZONE=pd.DataFrame(avgbyCity.sort_values("OZONE", ascending=False)["NO2"].head(5))
top5_OZONE=top5_OZONE.round(2)

fig = go.Figure(data=[go.Table(header=dict(values=['<b>City</b>', '<b>Average Pollution
                 cells=dict(values=[top5_OZONE.index,top5_OZONE.values]))
                     ])
fig.update_layout(
    title={
        'text': "<b>Top 5 Cities with Highest Average OZONE Pollutant</b>"}, width=600
    
fig.show()
```

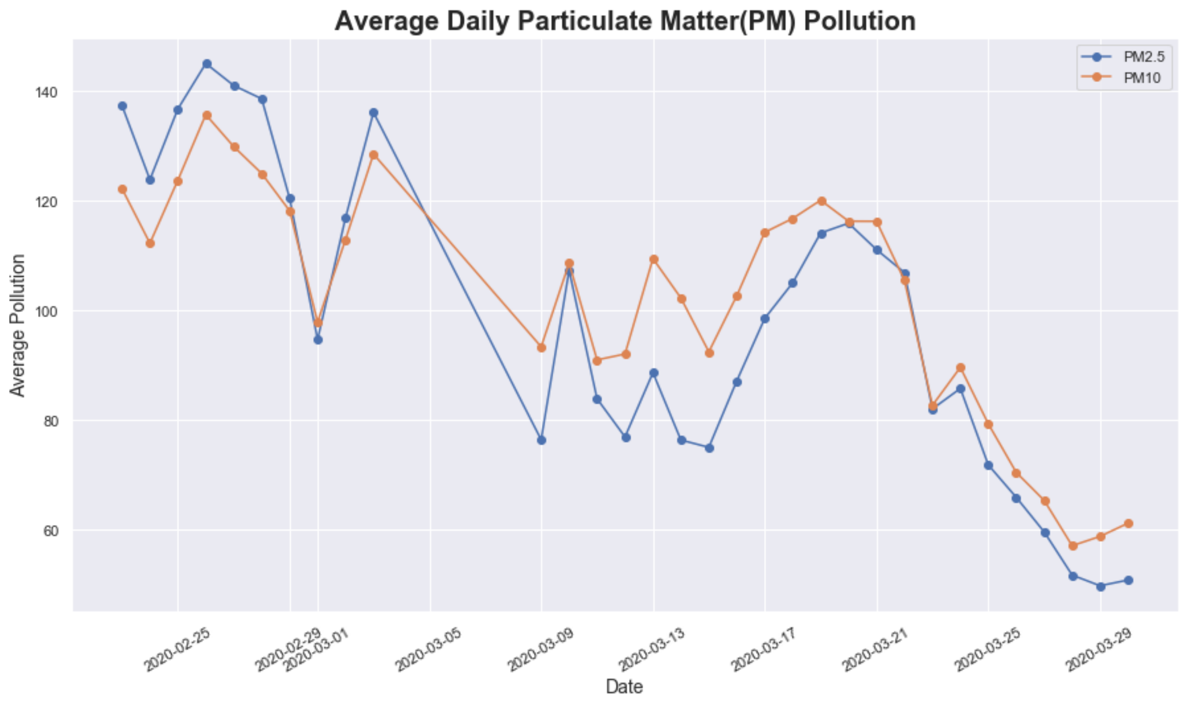
**Top 5 Cities with Highest Average OZONE Pollutant**

| **City** | **Average Pollution** |
|---|---|
| Bhiwani | 27.53 |
| Gaya | 8.43 |
| Dewas | 30.04 |
| Ujjain | 27.37 |
| Kalyan | 32.51 |

**Particulate Matter (PM) are the major constituent of the Air Pollution in the Air Quality Index in India, how exactly does it vary over different days in the last month.**

In [33]:

```
avgbyDay=AQ_df.groupby(["date", "pollutant_id"], as_index=False)["pollutant_avg"].mean(

#avgbyDay gives average daily pollution of each day for every pollutant.
#Pivoting the dataset to plot a timeseries line chart.

avgbyDay=avgbyDay.pivot(index="date", columns="pollutant_id", values="pollutant_avg")
plt.figure(figsize=(15,8))
fig=plt.plot(avgbyDay.index, avgbyDay["PM2.5"], label="PM2.5", marker="o")
fig=plt.plot(avgbyDay.index, avgbyDay["PM10"], label="PM10", marker="o")
plt.xlabel('Date', fontsize=14)
plt.ylabel('Average Pollution', fontsize=14)
plt.title('Average Daily Particulate Matter(PM) Pollution', fontsize=20, fontweight="bo
plt.xticks(rotation=30)
plt.legend()
plt.show()
```



**Insights:**
From the above chart it is evident that the pollution level due to PM2.5 and PM10 almost goes hand in hand.
We see a general trend of less pollution due to these Particulate Matters is reduced since last couple of weeks or in general since start of March.
The trend could be due to a rise in cases in the COVID-19 pandemic and the government taking strict measures to contain the spread.

In [34]:

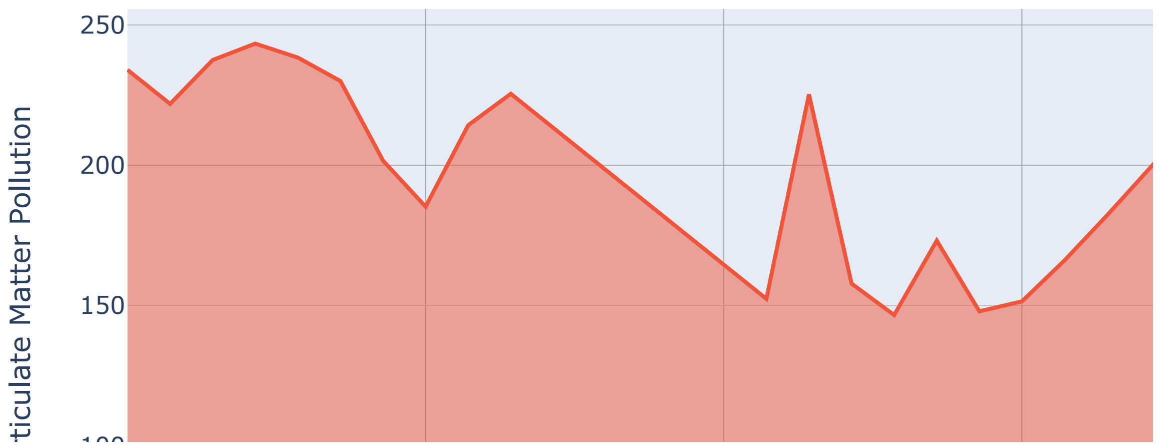
```
avgbyDay_min=AQ_df.groupby(["date", "pollutant_id"], as_index=False)["pollutant_min"].r
avgbyDay_min=avgbyDay_min.pivot(index="date", columns="pollutant_id", values="pollutant
# Saving the minimum and maxiumum values of each pollutant for each- minimum and maximu
avgbyDay_max=AQ_df.groupby(["date", "pollutant_id"], as_index=False)["pollutant_max"].r
avgbyDay_max=avgbyDay_max.pivot(index="date", columns="pollutant_id", values="pollutant
```

### What are the low and high level of PM2.5 and PM10 in general during various days?

In [35]:

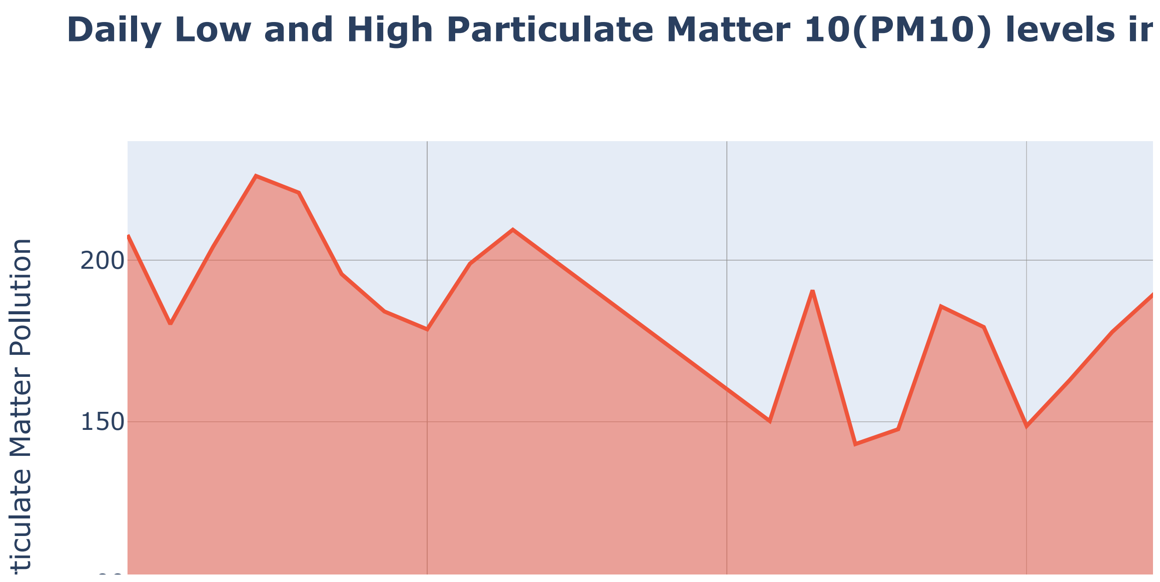
```
fig = go.Figure()
fig.add_trace(go.Scatter(x=avgbyDay_min.index, y=avgbyDay_min["PM2.5"], fill=None, name
fig.add_trace(go.Scatter(x=avgbyDay_min.index, y=avgbyDay_max["PM2.5"], fill='tonexty',
fig.update_layout(showlegend=True,
                  title={"text":"<b>Daily Low and High Particulate Matter 2.5(PM2.5) le
                  xaxis_title="Date", yaxis_title="Level of Particulate Matter Pollutio
                xaxis_tickformat = '%d %B<br>%Y')
fig.show()
```

**Daily Low and High Particulate Matter 2.5(PM2.5) levels**



In [36]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=avgbyDay_max.index, y=avgbyDay_min["PM10"], fill=None, name=
fig.add_trace(go.Scatter(x=avgbyDay_max.index, y=avgbyDay_max["PM10"], fill='tonexty',
fig.update_layout(showlegend=True,
                  title={"text":"<b>Daily Low and High Particulate Matter 10(PM10) leve
                  xaxis_title="Date", yaxis_title="Level of Particulate Matter Polluti
                xaxis_tickformat = '%d %B<br>%Y')
fig.show()
```

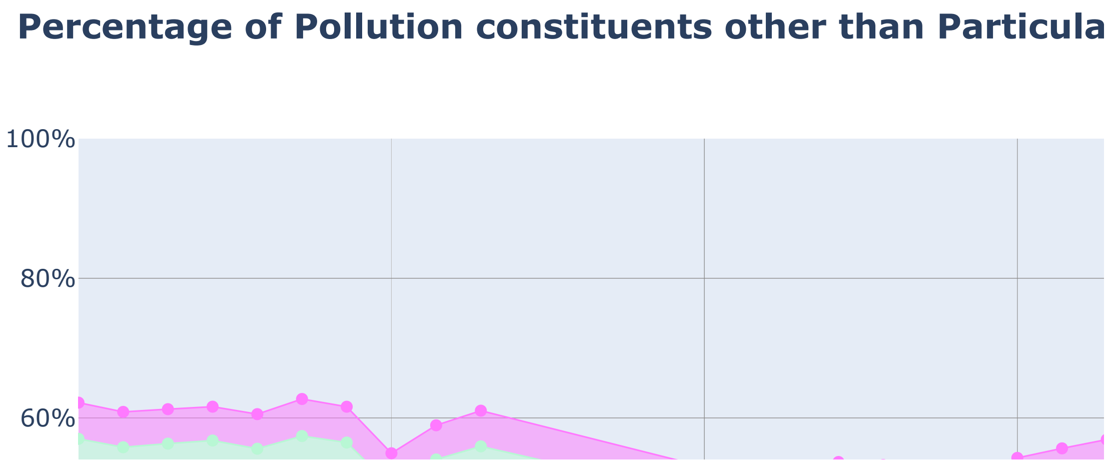


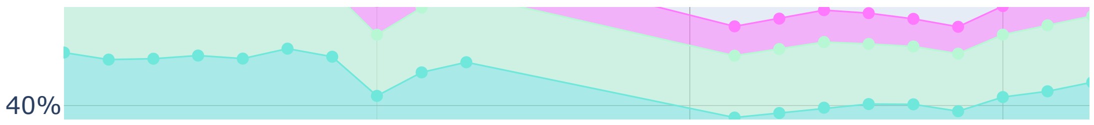
**Insights:**
From the above area charts we can get a general idea minimum and maximum levels of PM2.5 and PM10 throughout the day.
The lower blue line specifies the minimum average level of PM2.5 and PM10 in the chart, where as the Orange line specifies the highest level.
The area between those lines gives an estimate of average Particulate matter present in the air present on a specific day.

## What are the other major constituents/pollutants in the air pollution apart from PM2.5 and PM10 ?

In [37]:

```python
x=avgbyDay_min.index

fig = go.Figure()
fig.add_trace(go.Scatter(
    x=x, y=avgbyDay_min["CO"],
    hoverinfo='x+y',
    mode='lines+markers',
    line=dict(width=0.8, color='rgb(131, 90, 241)'),
    stackgroup='one', # define stack group,
    name="CO"
))
fig.add_trace(go.Scatter(
    x=x, y=avgbyDay_min["NO2"],
    hoverinfo='x+y',
    mode='lines+markers',
    line=dict(width=0.8, color='rgb(111, 231, 219)'),
    stackgroup='one',
    name="NO2"
))
fig.add_trace(go.Scatter(
    x=x, y=avgbyDay_min["SO2"],
    hoverinfo='x+y',
    mode='lines+markers',
    line=dict(width=0.8, color='rgb(184, 247, 212)'),
    stackgroup='one',
    name="SO2"
))
fig.add_trace(go.Scatter(
    x=x, y=avgbyDay_min["NH3"],
    hoverinfo='x+y',
    mode='lines+markers',
    line=dict(width=0.8, color='rgb(381, 121, 312)'),
    stackgroup='one',
    name="NH3"
))

fig.update_layout(title={"text":"<b>Percentage of Pollution constituents other than Par
    yaxis=dict(
        type='linear',
        range=[1, 100],
        ticksuffix='%'),xaxis_tickformat = '%d %B<br>%Y')
fig.show()
```





**Insights:**
From the above area chart it is evident that the major constituent of air pollution after particulate matters is CO and SO2. Whereas NH3 is the least contributor to the overall average pollution

In [38]:

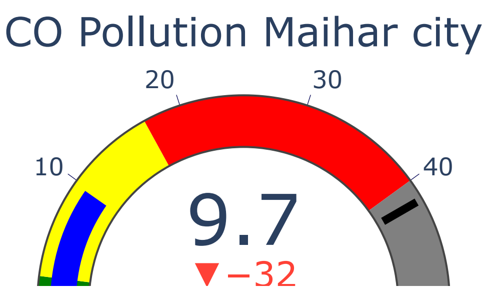
```
top5_CO=pd.DataFrame(avgbyCity.sort_values("CO", ascending=True)["CO"].head(5))
top5_CO=top5_CO.round(2)
top5_NO2=pd.DataFrame(avgbyCity.sort_values("NO2", ascending=True)["NO2"].head(5))
top5_NO2=top5_NO2.round(2)
top5_SO2=pd.DataFrame(avgbyCity.sort_values("SO2", ascending=True)["SO2"].head(5))
top5_SO2=top5_SO2.round(2)
top5_PM2_5=pd.DataFrame(avgbyCity.sort_values("PM2.5", ascending=True)["PM2.5"].head(5
top5_PM2_5=top5_PM2_5.round(2)
top5_PM10=pd.DataFrame(avgbyCity.sort_values("PM10", ascending=True)["PM10"].head(5))
top5_PM10=top5_PM10.round(2)
top5_OZONE=pd.DataFrame(avgbyCity.sort_values("OZONE", ascending=True)["NO2"].head(5))
top5_OZONE=top5_OZONE.round(2)
```

## What are the best cities with respect to minimum level of air pollution and an excellent air quality index?

On the below gauge indicators, the green range specifies best air quality index, yellow denotes moderately polluted, red denotes poor, where as grey denotes severe health issues can occur due to that levels.
The blue stream displays the average level of pollution in that city. Also, to get the exact number, the number in large font denotes the exact levels.
The black tick in the gauge denotes the national average for the particular pollutant.
The number below the cities pollution level with a delta reference denotes, how less or more it is than the national average for that pollutant.

In [39]:

```
fig1 = go.Figure(go.Indicator(
    mode = "delta+number+gauge",
    value = top5_CO.values.min(),
    title = {'text': str("CO Pollution "+top5_CO["CO"].idxmin()+" city")},
    delta = {'reference': avgbyCity["CO"].mean(), 'increasing': {'color': "RebeccaPurpl
    domain = {'x': [0, 0.25], 'y': [1, 1], 'row':1},
    align= "left",
    gauge = {'axis': {'range': [None, 50], 'tickwidth': 1, 'tickcolor': "darkblue"},
             'bar': {'color': "blue"},
             'steps' : [
                 {'range': [0, 2], 'color': "green"},
                 {'range': [2, 17], 'color': "yellow"},
                 {'range': [17, 40], 'color': "red"},
                 {'range': [40, 50], 'color': "gray"}],
             'threshold' : {'line': {'color': "black", 'width': 4}, 'thickness': 0.75,
))

fig1.update_layout(autosize=True)

fig1.show()
```

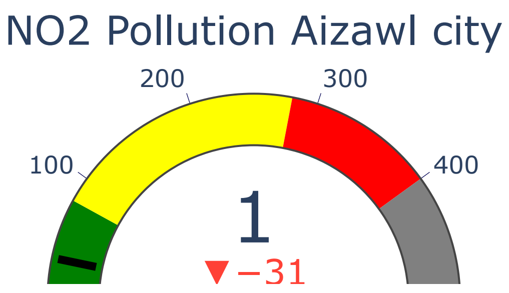


**Insights:**

The above gauge indicator indicates the best city with lowest pollution with respect to CO levels.

In [40]:

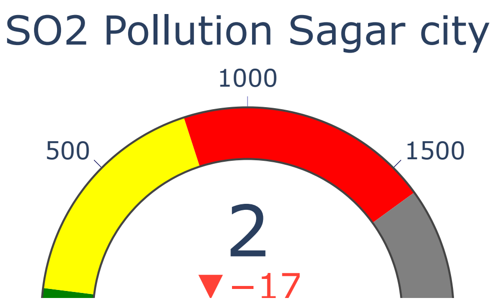
```
fig1 = go.Figure(go.Indicator(
    mode = "delta+number+gauge",
    value = top5_NO2.values.min(),
    title = {'text': str("NO2 Pollution "+top5_NO2["NO2"].idxmin()+" city")},
    delta = {'reference': avgbyCity["NO2"].mean(), 'increasing': {'color': "RebeccaPurp
    domain = {'x': [0, 0.25], 'y': [1, 1], 'row':1},
    align= "left",
    gauge = {'axis': {'range': [None, 500], 'tickwidth': 1, 'tickcolor': "darkblue"},
             'bar': {'color': "blue"},
             'steps' : [
                 {'range': [0, 80], 'color': "green"},
                 {'range': [80, 280], 'color': "yellow"},
                 {'range': [280, 400], 'color': "red"},
                 {'range': [400, 500], 'color': "gray"}],
             'threshold' : {'line': {'color': "black", 'width': 4}, 'thickness': 0.75,
))

fig1.update_layout(
    autosize=True)

fig1.show()
```



**Insights:**
The above gauge indicator indicates the best city with lowest pollution with respect to NO2 levels.

In [41]:

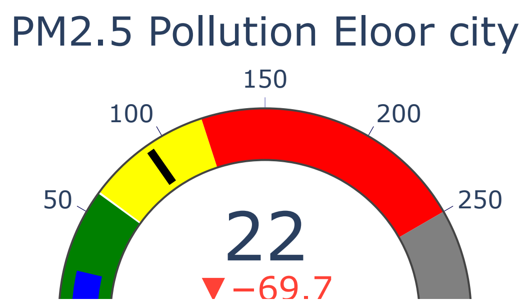
```
fig1 = go.Figure(go.Indicator(
    mode = "delta+number+gauge",
    value = top5_SO2.values.min(),
    title = {'text': str("SO2 Pollution "+top5_SO2["SO2"].idxmin()+" city")},
    delta = {'reference': avgbyCity["SO2"].mean(), 'increasing': {'color': "RebeccaPurp
    domain = {'x': [0, 0.25], 'y': [1, 1], 'row':1},
    align= "left",
    gauge = {'axis': {'range': [None, 2000], 'tickwidth': 1, 'tickcolor': "darkblue"},
             'bar': {'color': "blue"},
             'steps' : [
                 {'range': [0, 80], 'color': "green"},
                 {'range': [80, 800], 'color': "yellow"},
                 {'range': [800, 1600], 'color': "red"},
                 {'range': [1600, 2000], 'color': "gray"}],
             'threshold' : {'line': {'color': "black", 'width': 4}, 'thickness': 0.75,
))

fig1.update_layout(
    autosize=True)

fig1.show()
```



**Insights:**
The above gauge indicator indicates the best city with lowest pollution with respect to SO2 levels.

In [42]:

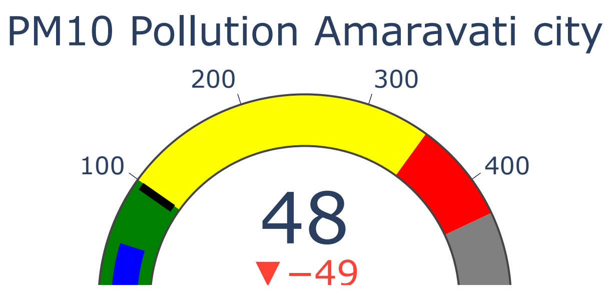
```
fig1 = go.Figure(go.Indicator(
    mode = "delta+number+gauge",
    value = top5_PM2_5.values.min(),
    title = {'text': str("PM2.5 Pollution "+top5_PM2_5["PM2.5"].idxmin()+" city")},
    delta = {'reference': avgbyCity["PM2.5"].mean(), 'increasing': {'color': "RebeccaPu
    domain = {'x': [0, 0.25], 'y': [1, 1], 'row':1},
    align= "left",
    gauge = {'axis': {'range': [None, 300], 'tickwidth': 1, 'tickcolor': "darkblue"},
             'bar': {'color': "blue"},
             'steps' : [
                 {'range': [0, 60], 'color': "green"},
                 {'range': [61, 120], 'color': "yellow"},
                 {'range': [120, 250], 'color': "red"},
                 {'range': [250, 300], 'color': "gray"}],
             'threshold' : {'line': {'color': "black", 'width': 4}, 'thickness': 0.75,
))

fig1.update_layout(
    autosize=True)

fig1.show()
```



**Insights:**
The above gauge indicator indicates the best city with lowest pollution with respect to PM2.5 levels.

In [43]:

```
fig1 = go.Figure(go.Indicator(
    mode = "delta+number+gauge",
    value = top5_PM10.values.min(),
    title = {'text': str("PM10 Pollution "+top5_PM10["PM10"].idxmin()+" city")},
    delta = {'reference': avgbyCity["PM10"].mean(), 'increasing': {'color': "RebeccaPur
    domain = {'x': [0, 0.25], 'y': [1, 1], 'row':1},
    align= "left",
    gauge = {'axis': {'range': [None, 500], 'tickwidth': 1, 'tickcolor': "darkblue"},
             'bar': {'color': "blue"},
             'steps' : [
                 {'range': [0, 100], 'color': "green"},
                 {'range': [100, 350], 'color': "yellow"},
                 {'range': [350, 430], 'color': "red"},
                 {'range': [430, 500], 'color': "gray"}],
             'threshold' : {'line': {'color': "black", 'width': 4}, 'thickness': 0.75,
))

fig1.update_layout(
    autosize=True)

fig1.show()
```



**Insights:**
The above gauge indicator indicates the best city with lowest pollution with respect to PM10 levels.

## Conclusion:

From the above exploratory data analysis on the Air Quality Index in India of various cities, we are able to get a general overview on how the pollution level or the Air Quality Index varies across various days in a week. Additionally, how it changes during different times of days for various pollutants.
In addition to the above, we can identify and rank most polluted cities, states. Get details on cities with lowest pollution levels. We were also able to identify the major constituents of Air Pollution that deteroriate the Air Quality Index.
We also saw a trend in decreasing pollution levels recently due to a lockdown in India to prevent community spread of COVID-19 as compared to previous weeks.