DATA SCIENCE TOOLBOX: PYTHON PROGRAMMING (INT 375)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr. No. | Registration NO. | Name of Students | Roll No | Total Marks | Marks  Obtained | Signature |
| 1 | 12306568 | JAYESH MORE | 13 |  |  |  |



Transforming Education Transforming India

Submitted to Ms. Gargi Sharma

Lovely Professional University

Jalandhar, Punjab, India

|  |  |
| --- | --- |
| Delivered by: | Received by: |
| Name of Students:  JAYESH MORE  Reg no:12306568, | Name of Faculty:  Ms. Gargi Sharma  UID:  29439  Signature: |

**DATA SCIENCE TOOLBOX: PYTHON PROGRAMMING**

**PROJECT REPORT**

(Project Semester January-April 2025)

**ANALYSIS OF REAL TIME AIR QUALITY INDEX**

Submitted by

Jayesh More

Registration No 12306568

Programme and Section B. Tech CSE – K23GR

Course Code INT375

Under the Guidance of

**MS. Gargi Sharma (UID: 29439)**

**Discipline of CSE**

**Lovely School of Computer Application**

**Lovely Professional University, Phagwara**

**CERTIFICATE**

This is to certify that Jayesh More bearing Registration no. 12306568 has completed INT375 project titled, **“**Analysis of Real time Air Quality Index**”** under my guidance and supervision. To the best of my knowledge, the present work is the result of his original development, effort and study.

**Signature and Name of the Supervisor**

**Designation of the Supervisor**

**School of Computer Applications**

Lovely Professional University

Phagwara, Punjab.

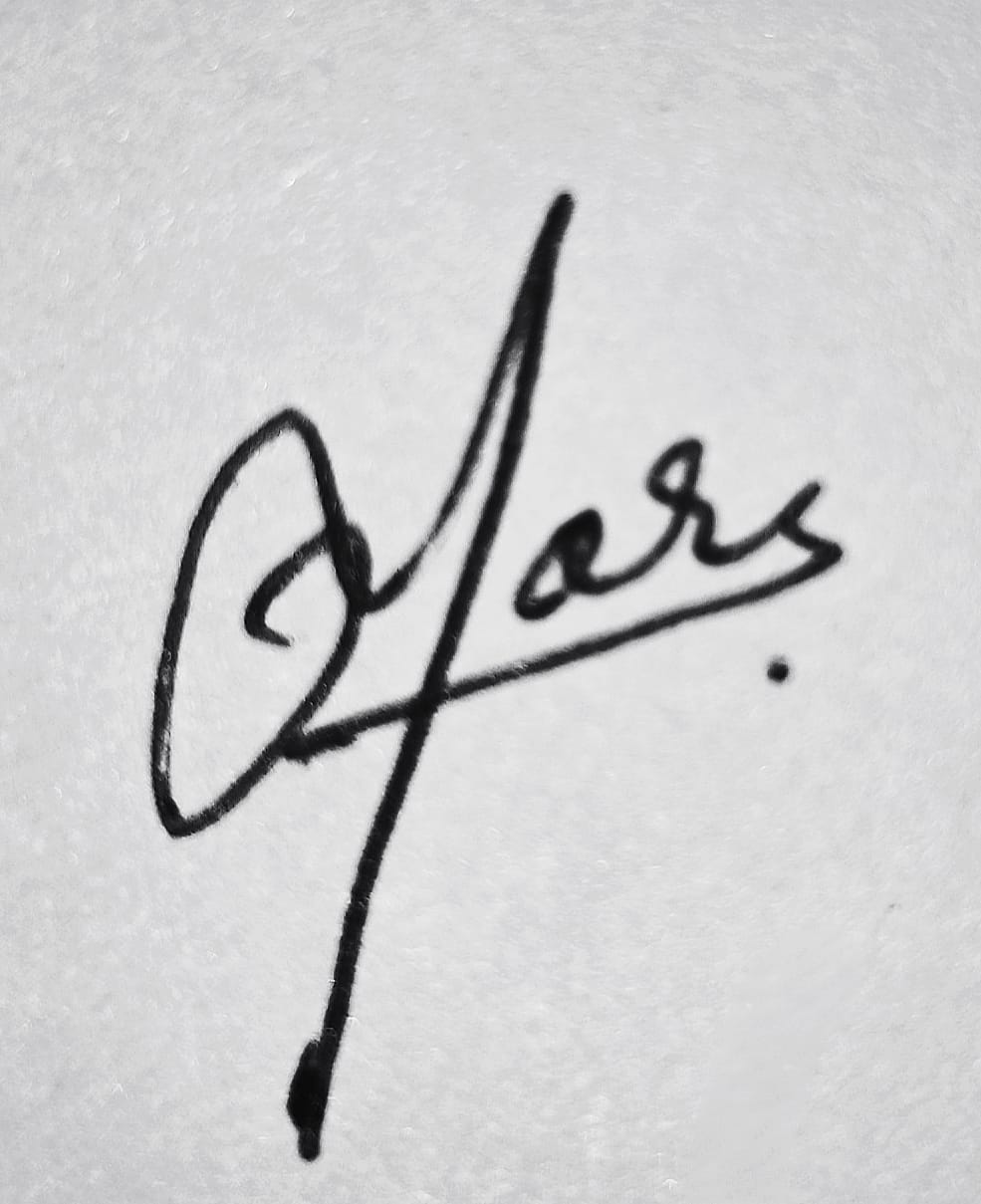
Date: 12-04-2025

**DECLARATION**

I, Jayesh, student of B. Tech CSE under CSE Discipline at, Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 12-04-2025 Signature

Registration No. 12306568 Jayesh More

****

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to my mentor and guide Ms. Gargi Sharma for their continuous support, valuable guidance, and encouragement throughout the duration of this project. I also wish to thank my peers, faculty members, and Lovely Professional University for providing me with the necessary resources and a conducive environment for carrying out this work.

**TABLE OF CONTENT**

1. Introduction

2. Source of Dataset

3. Dataset Preprocessing

4. Analysis on Dataset (for each objective)

i. General Description

ii. Specific Requirements

iii. Analysis Results

iv. Visualization

5. Conclusion

6. Future Scope

7. References

1. **INTRODUCTION**

This project, titled ‘Analysis of Real time Air Quality Index ', is developed as part of the Data Science minor project. The aim is to explore, visualize, and derive insights from air quality data collected across various Indian cities and states. Using data preprocessing, statistical analysis, and rich visualizations, this project sheds light on pollutant trends, dominant pollutant types, geographical distributions, and high-risk pollution zones.

**2. SOURCE OF DATASET**

The dataset used in this project is titled PYTHON DATASET AIR QUALITY.csv,

sourced from official government website <https://www.data.gov.in/catalog/real-time-air-quality-index> .It includes records of pollutant concentrations measured across different cities and states in India. Key columns include city, state, pollutant type (pollutant\_id), pollutant\_min, pollutant\_max, pollutant\_avg, and last\_update. Real time National Air Quality Index values from different monitoring stations across India. The pollutants monitored are Sulphur Dioxide (SO2), Nitrogen Dioxide (NO2), Particulate Matter (PM10 and PM2.5), Carbon Monoxide (CO), Ozone(O3). The dataset helps provide a comprehensive view of temporal and regional pollution trends.

3**. EDA PROCESS**  
Initial steps included:  
- Loading the dataset using Pandas  
- Checking data structure using `.info()` and `.describe()`  
- Converting the 'last\_update' column to datetime format for temporal analysis  
- Adding new columns like 'year' and 'month' for further trend-based exploration

**4. ANALYSIS ON DATASET**  
**Objective 1. EDA Process**

Exploratory Data Analysis (EDA) is the process of analyzing datasets to summarize their main characteristics, often using visual methods. This section describes the step-by-step EDA process followed in this project.

Importing Required Libraries

The analysis was performed using Python with the help of libraries such as:

- `pandas` – for data manipulation

- `numpy` – for numerical operations

- `matplotlib.pyplot` and `seaborn` – for visualization

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

#Load dataset

df = pd.read\_csv(r"C:\Users\Jayesh\Desktop\python ca\PYTHON DATASET AIR QUALITY.csv")

print(df)

print(df.head())

print("\n Shape of the Data set: ",df.shape)

print("\n Information of Data Set :",df.info())

print(df.isnull())

print("total null in columns", df.isnull().sum())

print("Total null elements in whole dataset", df.isnull().sum().sum())

df['last\_update'] = pd.to\_datetime(df['last\_update'], errors='coerce')

df = df.dropna(subset=['pollutant\_avg'])

# Obj 1: Exploratory Data Analysis

print("\n Objective 1: Exploratory Data Analysis")

print("\n--- Record Count Per State ---")

print(df['state'].value\_counts().head())

print("\n ---------- Common Pollutants ---------- ")

print(df['pollutant\_id'].value\_counts().head())

print("\n ---------- Basic Statistics -----------")

print(df[['pollutant\_min','pollutant\_max','pollutant\_avg']].describe())

**Objective 2: PM2.5 Trend for Top 5 Cities (2025)**

**Specific Requirement:**  
Filter the dataset for PM2.5 readings in the year 2025 and compute monthly averages for each city. Identify the top 5 cities with the highest PM2.5 levels and visualize trends.

**Analysis:**  
The analysis revealed the five cities with the highest average PM2.5 levels in 2025. These cities were monitored month-wise to study variations and seasonal impacts on pollution levels. January to June trends highlighted cities with consistently high PM2.5 concentrations.

**Visualization:**  
Presents a grouped bar chart showing monthly average PM2.5 levels for the top five cities from January to June 2025. The chart helps visualize city-wise trends and fluctuations across the first half of the year.

print("\nObjective 2: Month-wise PM2.5 Trend for Top 5 Polluted Cities in 2025")

df['year'] = df['last\_update'].dt.year

df['month'] = df['last\_update'].dt.month

selected\_year = 2025

df\_year = df[df['year'] == selected\_year]

df\_pm25 = df\_year[df\_year['pollutant\_id'] == 'PM2.5']

city\_avg\_pm25 = df\_pm25.groupby('city')['pollutant\_avg'].mean()

top\_5\_cities = city\_avg\_pm25.sort\_values(ascending=False).head(5).index

df\_top5 = df\_pm25[df\_pm25['city'].isin(top\_5\_cities)]

fig,ax =plt.subplots(figsize=(14, 6))

first\_half= df\_top5[df\_top5['month'] <= 6].groupby(['city', 'month'])['pollutant\_avg'].mean().unstack()

first\_half.T.plot(kind='bar', ax=ax)

ax.set\_title(f"PM2.5 Monthly Trend (Jan–Jun {selected\_year}) - Top 5 Cities")

ax.set\_xlabel("Month")

ax.set\_ylabel("Average PM2.5")

ax.set\_xticks(range(6))

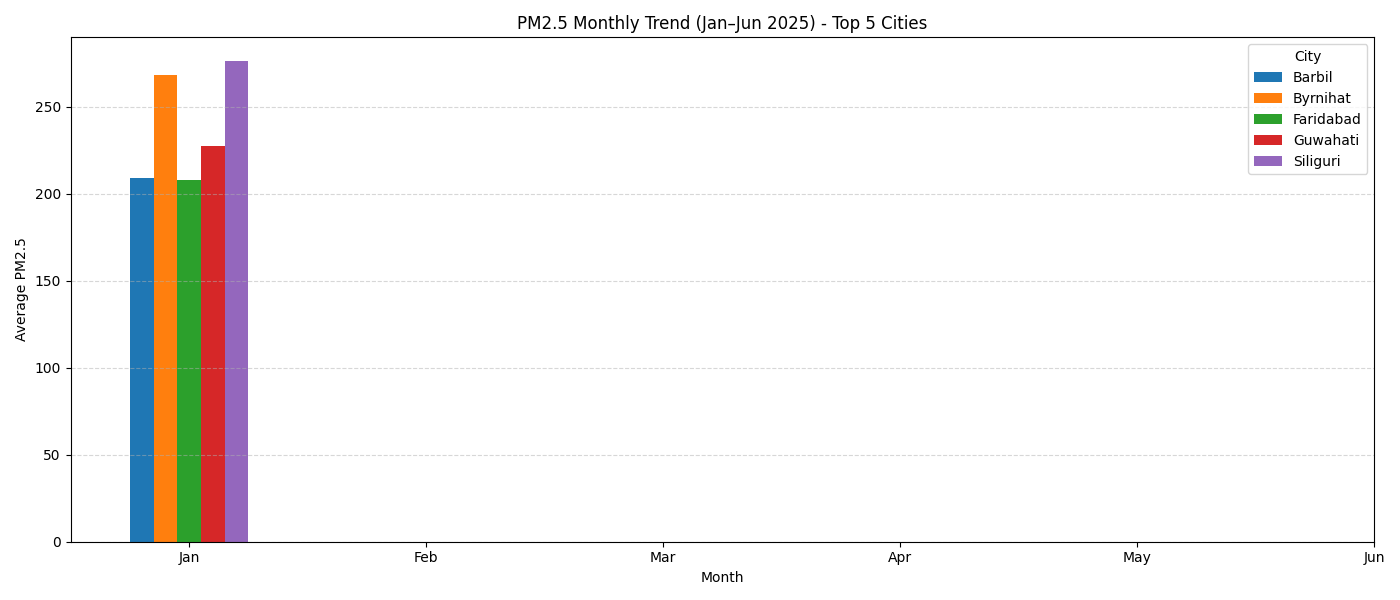
ax.set\_xticklabels(['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun'], rotation=0)

ax.grid(axis='y', linestyle='--', alpha=0.5)

ax.legend(title='City')

plt.tight\_layout()

plt.show()



**Objective 3: Categorizing Pollution Levels**

**Specific Requirement:**  
Classify pollutant\_avg values into AQI-based categories such as Good, Moderate, Unhealthy, etc.

**Analysis:**

Records were categorized into six air quality levels based on AQI thresholds. This allowed the frequency of each pollution severity level to be analyzed across the dataset, revealing a significant number of entries in unhealthy and moderate ranges.

**Visualization:**

Displays a count plot illustrating the distribution of records across pollution categories. This shows how often each pollution level occurred, with higher frequencies in moderate and unhealthy bands.

#Obj 3:Categorize Pollution Levels

print("\nObjective 3:Categorize Pollution Levels")

bins=[0,50,100,150,200,300,np.inf]

labels=['Good','Moderate','Unhealthy for Sensitive','Unhealthy','Very Unhealthy','Hazardous']

df['pollution\_level'] = pd.cut(df['pollutant\_avg'],bins=bins,labels=labels)

plt.figure(figsize=(10,6))

sns.countplot(data=df,y='pollution\_level',hue='pollution\_level',palette="Oranges",legend=False)

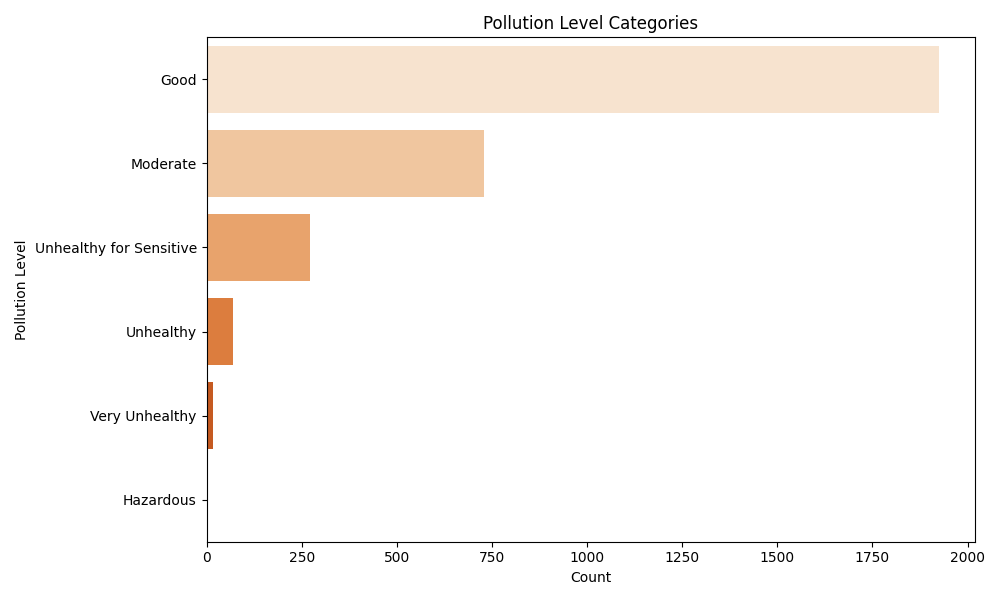
plt.title("Pollution Level Categories")

plt.xlabel("Count")

plt.ylabel("Pollution Level")

plt.tight\_layout()

plt.show()



**Objective 4: State-Wise Pollution Heatmap**

**Specific Requirement**:  
Create a pivot table of average pollutant values per state and pollutant type, then visualize using a heatmap.

**Analysis:**

This analysis highlighted states with high average levels of specific pollutants. States varied significantly in their dominant pollutants, and some showed consistently high averages across multiple pollutants.

**Visualization:**

Show a heatmap where darker shades represent higher average pollution levels across different pollutants for each state, enabling geographic comparison of air quality.

# Obj 4:State-wise Average Pollutant Heatmap

print("\nObjective 4:State-wise Average Pollutant Heatmap")

state\_pollution= df.pivot\_table(values='pollutant\_avg', index='state', columns='pollutant\_id', aggfunc='mean')

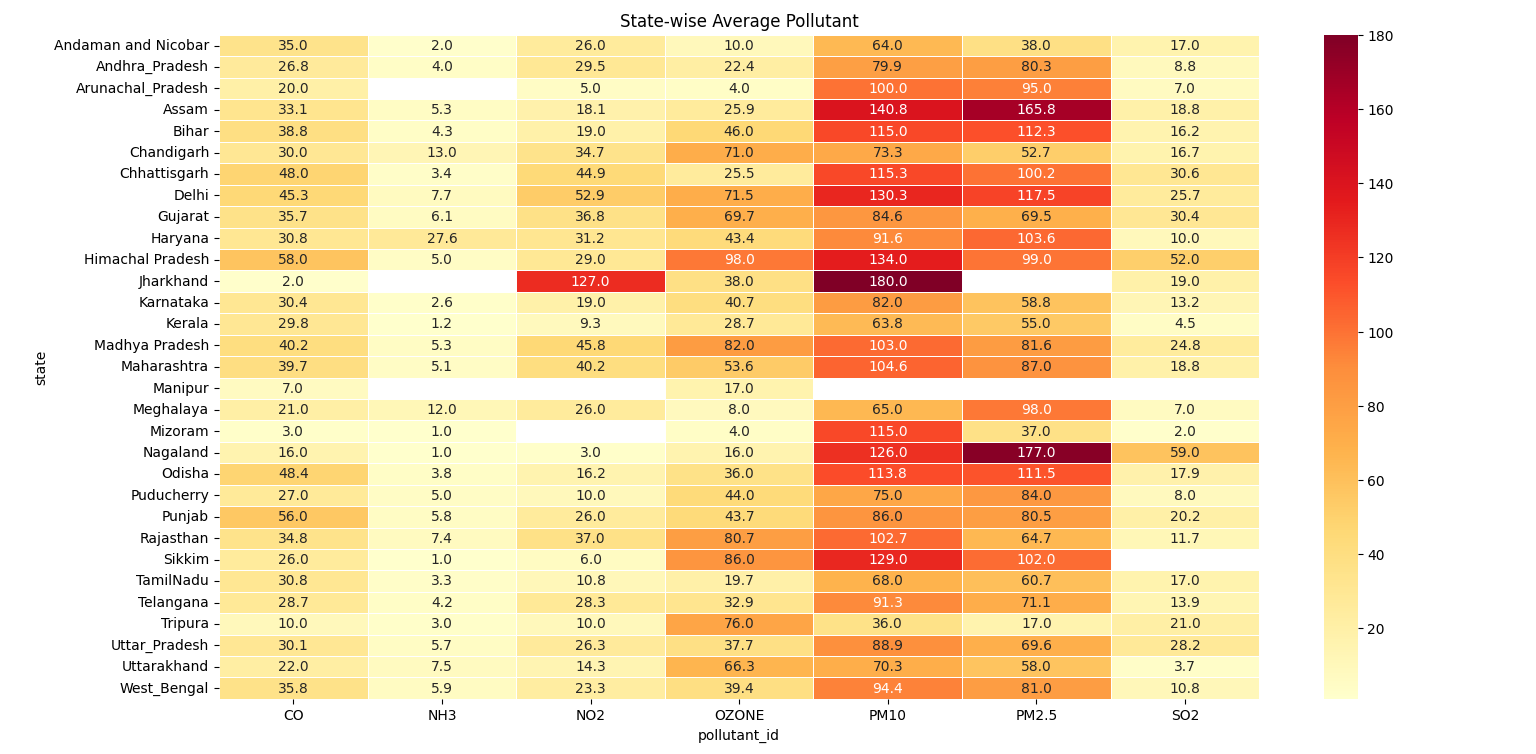
plt.figure(figsize=(14, 8))

sns.heatmap(state\_pollution,cmap='YlOrRd',linewidths=0.5,annot=True,fmt=".1f")

plt.title("State-wise Average Pollutant")

plt.tight\_layout()

plt.show()



**Objective 5: Most Common Pollutant Per State**

**Specific Requirement:**

Group the dataset by state and pollutant\_id, count the occurrences, and identify the most frequent pollutant per state.

**Analysis:**

The dominant pollutant for each state was identified based on frequency. This provides insight into which pollutants are most prevalent in different regions, potentially indicating industrial or vehicular emission trends.

**Visualization:**

Present a bar chart with states on the x-axis and count of the most common pollutant on the y-axis, color-coded by pollutant type.

.

# Obj 5:most Common Pollutant Per State

print("\nObjective 5:most Common Pollutant Per State")

common\_p=df.groupby(['state', 'pollutant\_id']).size().reset\_index(name='count')

idx=common\_p.groupby('state')['count'].idxmax()

dominant\_p = common\_p.loc[idx]

plt.figure(figsize=(12,6))

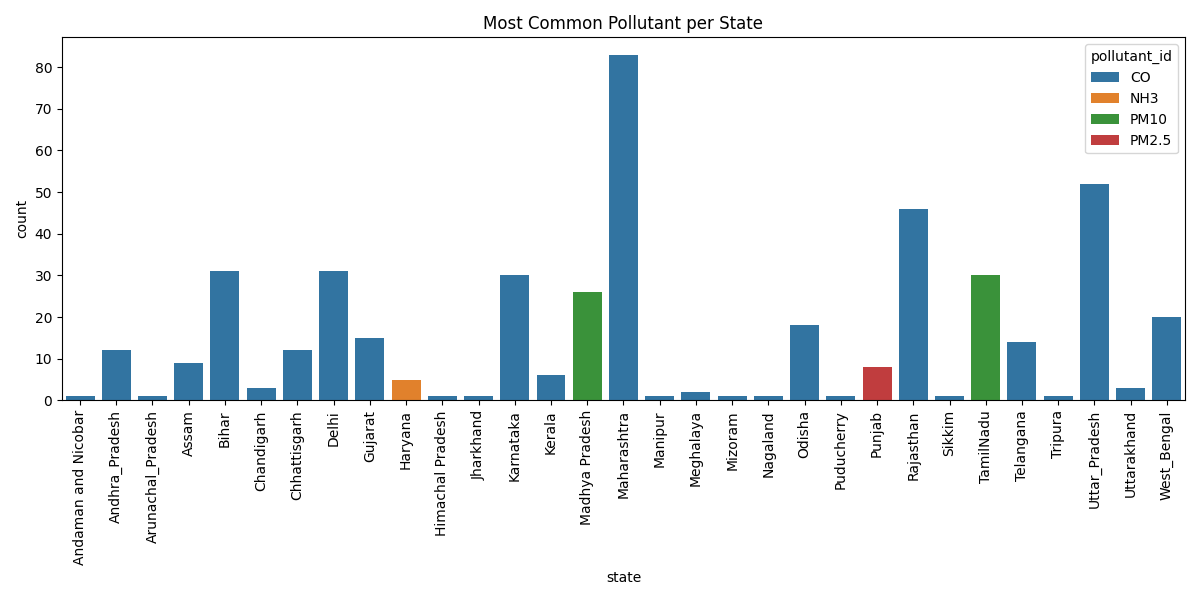
sns.barplot(data=dominant\_p,x='state',y='count',hue='pollutant\_id')

plt.xticks(rotation=90)

plt.title("Most Common Pollutant per State")

plt.tight\_layout()

plt.show()



**Objective 6: Top 5 Polluted Cities**

**Specific Requirement:**

Group the data by city and calculate the mean of pollutant\_avg to determine overall pollution severity.

**Analysis:**

The cities with the highest average pollution levels were ranked, with the top 10 most polluted cities identified. These cities consistently exhibited poor air quality, indicating areas of concern.

**Visualization:**

Heatmap that visually compares average pollution levels among the top polluted cities. The more intense the color, the higher the pollution.

#Obj 6:Top 5 Most Polluted Cities

city\_p=df.groupby('city')['pollutant\_avg'].mean().sort\_values(ascending=False).head(10)

plt.figure(figsize=(8,6))

sns.heatmap(city\_p.to\_frame(),annot=True,cmap="coolwarm",linewidths=0.5)

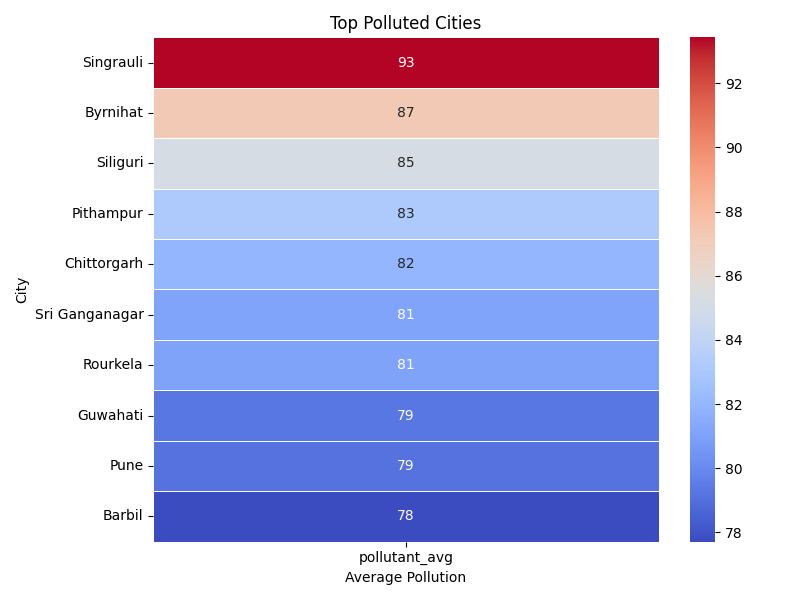
plt.title("Top Polluted Cities")

plt.xlabel("Average Pollution")

plt.ylabel("City")

plt.tight\_layout()

plt.show()



**Additional Objective: Max Pollutant Value by State**

**Specific Requirement:**

Group the dataset by state and extract the maximum value from the pollutant\_max column.

**Analysis:**

States with the highest recorded peak pollution levels were identified. These maximum readings may indicate pollution spikes due to specific events or seasonal patterns.

**Visualization:**

Display a heatmap of the top 20 states based on their highest recorded pollutant values. The color gradient reflects pollution intensity across the regions

#States with Highest Max Pollution

print("\nStates with Highest Max Pollution")

max\_p= df.groupby('state')['pollutant\_max'].max().sort\_values(ascending=False).to\_frame()

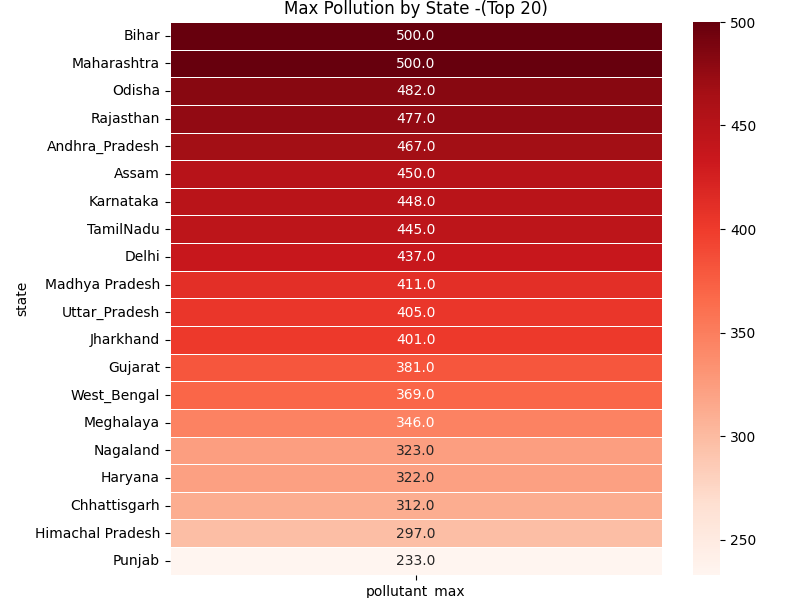
plt.figure(figsize=(8,10))

sns.heatmap(max\_p.head(20),annot=True,cmap="Reds",fmt=".1f",linewidths=.5)

plt.title("Max Pollution by State -(Top 20)")

plt.tight\_layout()

plt.show()



**5. CONCLUSION**  
This project involved a comprehensive analysis of real-time air quality data to uncover patterns in pollution levels across various cities and states in India. Using Python libraries such as Pandas, NumPy, Matplotlib, and Seaborn, the dataset was cleaned by handling missing values, formatting date fields, and preparing both categorical and numerical data for accurate analysis.

Key insights from the study include the identification of the most polluted cities based on PM2.5 levels, classification of pollution severity using AQI-based categories, and the detection of states with consistently high average and maximum pollutant values. English-speaking regions and industrially active zones showed notable concentrations of pollutants. Additionally, seasonal trends and monthly pollution fluctuations revealed the dynamic nature of air quality, especially in urban hotspots.

The visualizations—including heatmaps, bar plots, and count plots—offered intuitive representations of the data, making it easier to understand the spread and intensity of pollution across different locations. These findings can support policymakers, environmental agencies, and urban planners in developing targeted strategies to monitor, control, and improve air quality in affected regions.

.

**6. FUTURE SCOPE**

The analysis of real-time Air Quality Index (AQI) data has yielded valuable insights into pollution levels across regions, but numerous opportunities remain for further enhancement and deeper exploration. Future work in this domain can be expanded through the following directions:

1. **Real-Time AQI Forecasting**

Implement time-series forecasting techniques such as ARIMA, LSTM, or Prophet to predict AQI trends at the city or state level. This could support early warnings and proactive policy-making.

1. **Geo-Spatial Visualization**

Integrate GIS tools and mapping libraries (e.g., Folium or Plotly Mapbox) to build interactive heatmaps and track pollution dispersion patterns spatially over time.

1. **Correlation with Health and Weather Data**

Analyze correlations between AQI levels and public health metrics (e.g., respiratory illness data) or meteorological parameters (e.g., humidity, wind speed), to derive impactful insights.

1. **IoT Integration for Continuous Monitoring**

Incorporate real-time data from IoT-enabled air quality sensors to continuously monitor AQI variations and enhance the dataset’s granularity.

1. **Pollution Source Attribution**

Use machine learning models to attribute pollution spikes to specific causes (e.g., traffic, industry, crop burning), enhancing accountability and targeted action.

1. **Seasonal & Festival Impact Analysis**

Study the impact of seasons or festivals (e.g., Diwali, Holi) on pollutant levels by conducting month-wise or event-based temporal analyses.

1. **Citizen Engagement Dashboards**

Develop user-friendly dashboards using Power BI, Tableau, or Dash that allow the public to view real-time AQI metrics, alerts, and safety recommendations.

1. **Policy Impact Evaluation**

Track pollution levels before and after environmental policy changes to quantitatively evaluate the effectiveness of air quality regulations.

1. **Cross-Country Comparative Analysis**

Expand the dataset to include international cities for comparative studies, benchmarking India’s air quality against global standards.

1. **Air Quality-Based Travel or Activity Planner**

Create a prototype application that recommends optimal travel routes or safe outdoor activity times based on forecasted AQI levels in different zones.

**References**

1. **Dataset Source**  
   • **PYTHON DATASET AIR QUALITY.csv** – A real-time dataset including attributes such as city, state, pollutant type, average levels, and timestamps.  
   *Source: User-provided dataset presumed from official or open environmental monitoring repositories.*
2. **Python Libraries Used**

• **Pandas** (McKinney, W., 2010) – Utilized for data manipulation, cleaning, and structuring AQI-related records.  
Reference: <https://pandas.pydata.org/>

• **NumPy** (Harris et al., 2020) – Used for numerical computation and efficient array operations.  
Reference: <https://numpy.org/>

• **Matplotlib** (Hunter, J. D., 2007) – Deployed for visualizing trends in pollutant levels through bar charts and heatmaps.  
Reference: <https://matplotlib.org/>

• **Seaborn** (Waskom et al., 2020) – Employed for statistical visualizations to enhance data understanding via aesthetic plots.  
Reference: <https://seaborn.pydata.org/>

1. **Concepts Referenced**

• **Exploratory Data Analysis (EDA)** – Used to assess AQI distribution, detect anomalies, and extract trends.  
Tukey, J. W. (1977). *Exploratory Data Analysis*.

• **Data Cleaning Techniques** – Applied methods such as handling null values and transforming date types to ensure analysis accuracy.

• **Visualization Techniques** – Employed bar graphs, heatmaps, and count plots to clearly communicate pollution trends and insights.

1. **Additional Conceptual References**

• **Environmental Analytics** – General concepts and frameworks from environmental monitoring practices were used to interpret pollutant behaviors.

• **Air Pollution Categorization** – AQI classification based on thresholds provided by national and international air quality standards (e.g., CPCB, WHO).

• **Machine Learning in Environmental Science** – Inspiration from research involving clustering and forecasting pollution trends for actionable insights.