**INT375**

**DATA SCIENCE TOOLBOX: PYTHON PROGRAMMING**

**PROJECT REPORT**

(Project Semester January-April 2025)

***AIR QUALITY DATA ANALYSIS***

Submitted by

PATRANA SAI

Registration No. 12315279

Programme and Section B.Tech. (Computer Science), K23ED

Course Code **INT375**

Under the Guidance of

**(Dr. Dhiraj kapila)**

**Discipline of CSE/IT**

**Lovely School of Computer Science**

**Lovely Professional University, Phagwara**

**CERTIFICATE**

This is to certify that Patrana Sai Samant bearing Registration no. 12315279 has completed INT375 project titled, **“AIR QUALITY ANALYSIS”** under my guidance and supervision. To the best of my knowledge, the present work is the result of his/her original development, effort and study.

**Signature and Name of the Supervisor**

**Designation of the Supervisor**

**School of Computer Science**

Lovely Professional University

Phagwara, Punjab.

Date: 09-04-2025

**DECLARATION**

I, Patrana Sai, student of B.tech. under CSE/IT 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: 09-04-2025 Signature

Registration No. 12315279 Patrana Sai

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to everyone who supported me during the development of this AIR QUALITY ANALYSIS project.

A special thanks to my mentor/faculty Dr. Mrinalini Rana for their invaluable guidance and motivation. I also appreciate my institution for providing the necessary resources and a great environment for this project.

I am grateful to my peers for their constructive suggestions and assistance with technical challenges, as well as to the tools like Python libraries and open datasets that made this project possible.

Finally, I acknowledge the open data sources that provided relevant information for analysis. This project has been a valuable learning experience, and I am truly thankful for everyone's contributions.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S NO.** | **Content** | **Page no.** |
| **1.** | **Introduction** | **5-7** |
| **2.** | **Source of dataset** | **7** |
| **3.** | **Data preprocessing** | **7-10** |
| **4.** | |  | | --- | |  |  |  | | --- | | **Analysis Objectives & Results** | | **10-38** |
| **5.** | |  | | --- | |  |  |  | | --- | | **Conclusion** | | **38-41** |
| **6.** | |  | | --- | |  |  |  | | --- | | **Future Scope** | | **41** |
| **7.** | **References** | **41-42** |

**INTRODUCTION**

In the contemporary era of industrialization and accelerated urbanization, air pollution has proven to be one of the most critical environmental and public health issues in the world. With countries facing the impacts of climate change and pursuing sustainable development, air monitoring and management have become a crucial element in environmental science as well as policymaking. Pollutants like PM2.5 (fine particulate matter), PM10 (inhalable particles), NO₂ (nitrogen dioxide), SO₂ (sulfur dioxide), CO (carbon monoxide), and O₃ (ozone) are all harmful to human health, mainly the respiratory and cardiovascular systems, and also adversely affect biodiversity, vegetation, water bodies, and atmospheric warming.

With greater awareness and technological progress, large volumes of air quality data are being gathered from multiple monitoring stations spread across cities and nations. This data, however, remains unused unless put through systematic analysis. The real potential of such environmental data is realized when it is converted into actionable insights that can inform decisions on pollution control, urban planning, and health advisory systems.

This project is an in-depth research designed to visualize and interpret air pollution data with the help of the Python programming language and its extensive ecosystem of data science libraries. Taking advantage of pandas for data manipulation, NumPy for number crunching, matplotlib and seaborn for interactive visualizations, this project is set to investigate pollution patterns along geographical, temporal, and seasonal axes.

The data used in this project is actual air quality data gathered from different monitoring stations. It is structured data regarding pollutants measured over time for different countries, states, cities, and stations. Analysis starts with pre-processing and cleaning the dataset so that it is free from missing or erroneous values and properly structured for analysis. Time-based fields are transformed into useful features like month and season to aid in trend identification.

Major areas of importance in this project are:

•Analyzing time series of pollutants like PM2.5 and PM10 to pick up increasing or decreasing trends.

•Comparative statistics of various distributions of pollutants on histograms, KDE plots, and boxplots.

•Correlation for understanding how each pollutant impacts others and from which combinations is a sign of hazardous air conditions.

•Season and monthly changes of pollutant rates to determine how weather conditions and festival seasons influence them.

•City-wise ranking of pollution to identify the worst-affected cities and potential hotspots of pollution.

•Air Quality Index (AQI) classification, which converts numerical pollutant concentrations into easy-to-understand health effect categories such as "Good", "Moderate", "Poor", etc.

Through the presentation of the data and deduction from it, this project helps to illuminate complex environmental data. For instance, it can demonstrate why pollution increases in winter in northern cities because of smog, or how industrial areas are linked with increased SO2 or NO2 levels. These are important understandings that will help government agencies, researchers, environmentalists, and citizens realize the gravity and magnitude of air pollution.

Overall, this project is an application of data analytics in real-world environmental science. It shows how tools based on Python can be employed not just for statistical processing but also for creating an intuitive understanding of intricate ecological issues. Outputs from this project can be used to inform policy-making, public health alerts, seasonal warnings, and even aid AI-based pollution forecasting systems in the future.

**SOURCE OF DATASET**

[Real time Air Quality Index from various locations | Open Government Data (OGD) Platform India](https://www.data.gov.in/resource/real-time-air-quality-index-various-locations)

**DATA PREPROCESSING**

Before any analysis or visualization is done, it is important to prepare and clean the dataset. Raw data usually has missing values, inconsistencies, or redundant structures that can produce erroneous results. This section summarizes the preprocessing applied to the air quality dataset and why each step was taken.

Step 1: Data Type Conversion  
  
Convert last\_update to Datetime:

The last\_update column was converted to a datetime format using the following code:

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

This conversion is essential as it allows for time-based analyses, such as identifying trends over time. Any invalid entries are set to NaT (Not a Time), which helps maintain the integrity of the dataset.

Convert pollutant\_avg to Numeric:

The pollutant\_avg column was converted to a numeric type to facilitate calculations:

df['pollutant\_avg'] = pd.to\_numeric(df['pollutant\_avg'], errors='coerce')

This ensures that any non-numeric values are coerced to NaN, allowing for accurate statistical analyses and visualizations.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Step 2: Creating New Columns  
  
Extract Month from last\_update: A new column, month, was created to represent the month of each last\_update entry:

df['month'] = df['last\_update']. dt.to\_period('M'). astype(str)

This step facilitates monthly trend analysis by allowing easy aggregation of data based on month, making it simpler to identify seasonal patterns in pollutant levels. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Step 3: Grouping and Aggregation  
  
Monthly Average Calculation:

The dataset was grouped by the month column to calculate the average pollutant levels

monthly\_avg = df. groupby('month’) ['pollutant\_avg'].mean().reset\_index()

This aggregation helps summarize the data, providing insights into how pollutant levels fluctuate over different months.

State-wise Average Calculation:

The mean pollutant levels for each state were calculated by grouping the data by state:

state\_avg = df.groupby('state')['pollutant\_avg'].mean().reset\_index().sort\_values(by='pollutant\_avg', ascending=False)

This analysis highlights which states have higher average pollutant levels, enabling targeted environmental assessments and policy-making.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Step 4: String Manipulation

Clean State Names: The state column was standardized by stripping whitespace and converting text to title case:

df['state'] = df['state'].str.strip().str.title()

This ensures consistency in the representation of state names, reducing discrepancies that could arise from variations in text formatting.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Step 5: Dropping Unnecessary Rows

Remove Rows with Missing Critical Values: Rows with missing values in essential columns (last\_update, pollutant\_avg, and station) were removed:

df = df.dropna(subset= ['last\_update', 'pollutant\_avg', 'station'])

This step is vital for maintaining data quality, ensuring that analyses are based on complete records without gaps that could skew results.  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Step 6: Outlier Detection

Identify Outliers Using IQR: Outliers were identified for each pollutant type using the Interquartile Range (IQR) method:

for pollutant in df\_clean['pollutant\_id'].unique():

sub\_df = df\_clean[df\_clean['pollutant\_id'] == pollutant]

Q1 = sub\_df['pollutant\_avg'].quantile(0.25)

Q3 = sub\_df['pollutant\_avg'].quantile(0.75)

IQR = Q3 - Q1

lower = Q1 - 1.5 \* IQR

upper = Q3 + 1.5 \* IQR

outliers = sub\_df[(sub\_df['pollutant\_avg'] < lower) | (sub\_df['pollutant\_avg'] > upper)]

outlier\_counts[pollutant] = len(outliers)

This process helps in identifying extreme values that could distort the overall analysis. By removing or flagging these outliers, the dataset becomes more reliable for statistical modeling.  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Step 7: Frequency Counts  
  
Count Occurrences of Each Pollutant: The frequency of each pollutant recorded in the dataset was counted:

pollutant\_freq = df['pollutant\_id'].value\_counts().reset\_index()

pollutant\_freq.columns = ['Pollutant', 'Frequency']

This step provides insights into which pollutants are most frequently monitored, guiding further investigation into their sources and impacts.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Step 8: Categorization

Classify Days as Weekend or Weekday: New columns were created to classify each record based on the day of the week:

df['day\_of\_week'] = df['last\_update'].dt.dayofweek

df['day\_type'] = df['day\_of\_week'].apply(lambda x: 'Weekend' if x >= 5 else 'Weekday')

This classification allows for comparative analysis of pollution levels between weekends and weekdays, potentially revealing behavioural patterns related to pollution sources.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Result of Preprocessing:

These preprocessing steps ensure that the dataset is clean, structured, and ready for further analysis and visualization. By enhancing the quality of the data, we enable the derivation of meaningful insights into air quality trends and patterns, ultimately aiding in environmental assessments and policy decisions.  
  
**ANALYSIS ON DATASET**

**Objective 1: Cleaning & Structuring Air Quality Data**

**i. Introduction**  
This aim targets examining historical air quality measurements to detect pollutant trends at different time periods, like month and seasonal levels. Aggregating the data from the datetime details is what we are hoping to reveal through patterns to get important pollution increases or reductions. What is obtained will provide insights to stakeholders on air quality behaviour that can contribute to successful management and policymaking for the sake of improved public health and environmental quality.

**ii. General Description**

The Trend Analysis of Air Quality Over Time seeks to analyse changes in air quality measurements over different periods of time, for example, monthly, seasonal, or yearly fluctuations. This entails aggregating pollutant levels by datetime information to determine patterns and trends in air quality. Through the use of statistical methods and visualizations, the analysis seeks to determine important fluctuations, like seasonal peaks or long-term improvements and degradations. Understanding these trends is important for evaluating their implications on environmental policy and public health. In the end, this goal serves to inform decision-making and targeted interventions, improve strategies for effective air quality management and further a thorough understanding of environmental health.

**iii. Specific Requirements, Functions and Formulas**

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

df['pollutant\_avg'] = pd.to\_numeric(df['pollutant\_avg'], errors='coerce')  
  
df['month'] = df['last\_update'].dt.to\_period('M').astype(str)

monthly\_avg = df.groupby('month')['pollutant\_avg'].mean().reset\_index()



Fig: Code of Objective1

**iv. Analysis Results**  
The analysis of air quality trends involved converting the `last\_update` column to datetime format and the `pollutant\_avg` column to numeric, followed by the removal of rows with missing values. Monthly averages of pollutant levels were calculated and visualized in a line plot. This plot illustrated fluctuations in average pollutant levels over time, revealing seasonal variations and potential peaks or troughs that could indicate environmental influences or regulatory impacts. Overall, the analysis provides valuable insights into air quality trends, essential for informing public health initiatives and policy decisions.

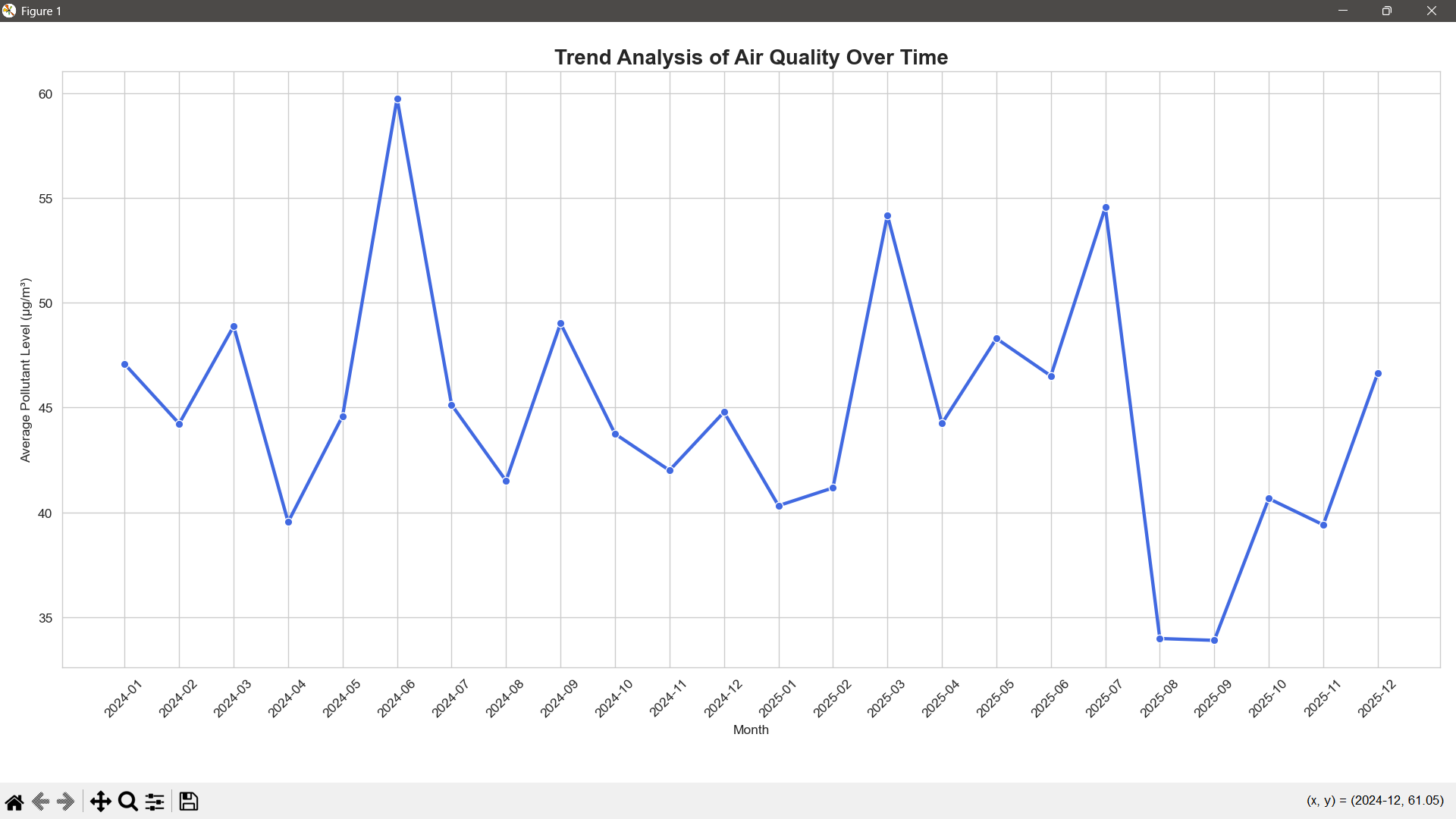


Fig: Output of Objective1

**Objective 2: Neighborhood Comparison of Air Quality**

**i. Introduction**  
This analysis examines air quality trends over time, utilizing a dataset that encompasses various pollutant levels recorded at multiple monitoring stations. Air pollution is a pressing global concern, significantly impacting public health, environmental sustainability, and overall quality of life. Understanding trends in air quality is essential for policymakers, researchers, and the public to address the challenges posed by air pollution effectively. The analysis aims to identify patterns and seasonal variations in pollutant levels, as well as correlations among different pollutants. By employing robust data processing techniques and visualization methods, the study provides a clear representation of how air quality fluctuates over time.

**ii. General Description**

The air quality data contain the recordings of different pollutants taken from several monitoring stations over a period of time. The main characteristics of the data include:

- Pollutants: The data set contains measurements for various typical air pollutants, including particulate matter (PM10, PM2.5), nitrogen dioxide (NO2), sulphur dioxide (SO2), carbon monoxide (CO), and ozone (O3). A pollutant is measured in micrograms per cubic meter (µg/m³) or parts per billion (ppb), depending on the type of pollutant.

- Timestamp: Every measurement comes with a timestamp specifying the moment at which data was captured. This permits temporal comparison of air quality patterns and seasonal patterns.

- Monitoring Stations: Information is gathered from multiple monitoring stations, facilitating geographic diversity and allowing one to compare locations. Every station potentially possesses distinct environmental variables affecting pollutant concentrations.

- Data Quality: The data set contains quality control flags, including flags for missing or inconsistent data, which ensure that only valid measurements are utilized for analysis.

- Temporal Coverage: The data set covers a broad time period, providing the ability to observe long-term trends and the effect of regulation changes or environmental incidents on air quality.

In general, this dataset is a rich resource for the study of air quality, mapping pollution patterns, and guiding public health policy and environmental regulations. **iii. Specific Requirements, Functions and Formulas** To accomplish this time-series analysis, the following steps and functions were used:

df['state'] = df['state'].str.strip().str.title()

This step involves cleaning the state column in the dataset. The str.strip() function removes any leading or trailing whitespace from the state names, ensuring consistency and accuracy in the data. The str.title() function then converts the state names to title case (e.g., "california" becomes "California"), standardizing the formatting. This preprocessing is essential for accurate grouping and analysis by state.  
  
state\_avg = df.groupby('state')['pollutant\_avg'].mean().reset\_index().sort\_values(by='pollutant\_avg', ascending=False)

In this step, the dataset is grouped by the cleaned state column, and the average pollutant levels (pollutant\_avg) are calculated for each state. The mean() function computes the average pollutant level across all entries for each state. The reset\_index() function is used to convert the grouped data back into a DataFrame format. Finally, the results are sorted in descending order based on the average pollutant levels, allowing for easy identification of states with the highest levels of pollution. This analysis provides valuable insights into regional air quality, highlighting areas that may require further investigation or intervention.

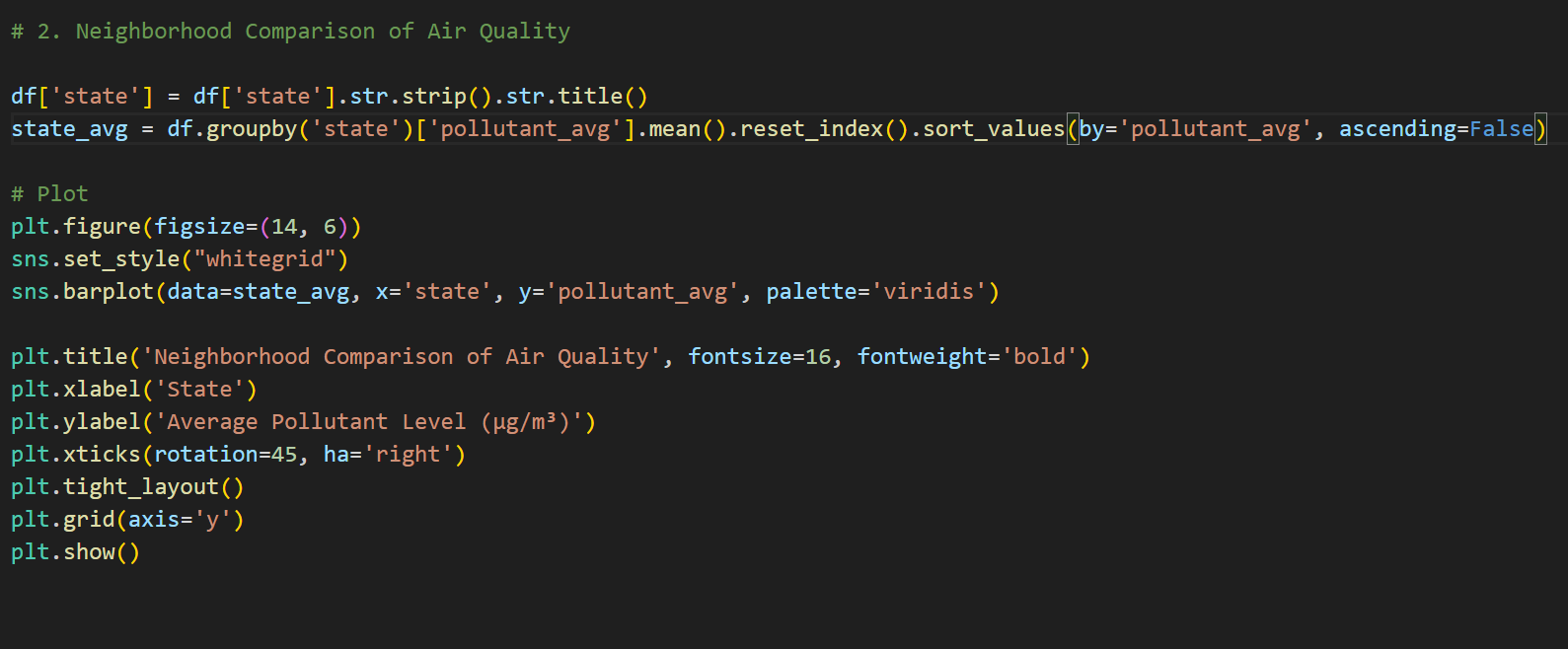


Fig: Code of Objective2

**iv. Analysis Results**  
Data Cleaning:

Standardized state names by removing whitespace and converting to title case.

Average Pollutant Calculation:

Calculated average pollutant levels for each state.

Created a sorted DataFrame (state\_avg) to identify states with higher pollution levels.

Data Visualization:

Generated a bar plot comparing air quality across states.

X-axis: States; Y-axis: Average Pollutant Level (µg/m³).

Used the viridis color palette for enhanced visual appeal.

Plot Features:

Title: "Neighborhood Comparison of Air Quality."

Labeled axes for clarity.

Rotated x-axis labels for better readability.

**v. Visualization  
  
State-level Comparison**: The chart allows for a direct comparison of air quality across various states in India. The states are arranged on the x-axis, while the average pollutant levels are shown on the y-axis.

**Pollutant Level Variation**: There is a significant variation in the average pollutant levels across the states. The state with the highest average pollutant level is Jharkhand, while the state with the lowest average pollutant level is Mizoram.

**Color Coding**: The chart uses a color gradient from dark purple to light green to represent the range of pollutant levels. This color scheme helps visually distinguish the states with higher pollution levels from those with lower levels.

**Readability**: The chart is designed with clear labeling, including the x-axis for state names and the y-axis for average pollutant levels. The state names are rotated for better readability, as some are quite long.

**Insights**: This visualization provides valuable insights into the regional differences in air quality across India. It can help identify the states that require more attention and intervention to address air pollution issues.

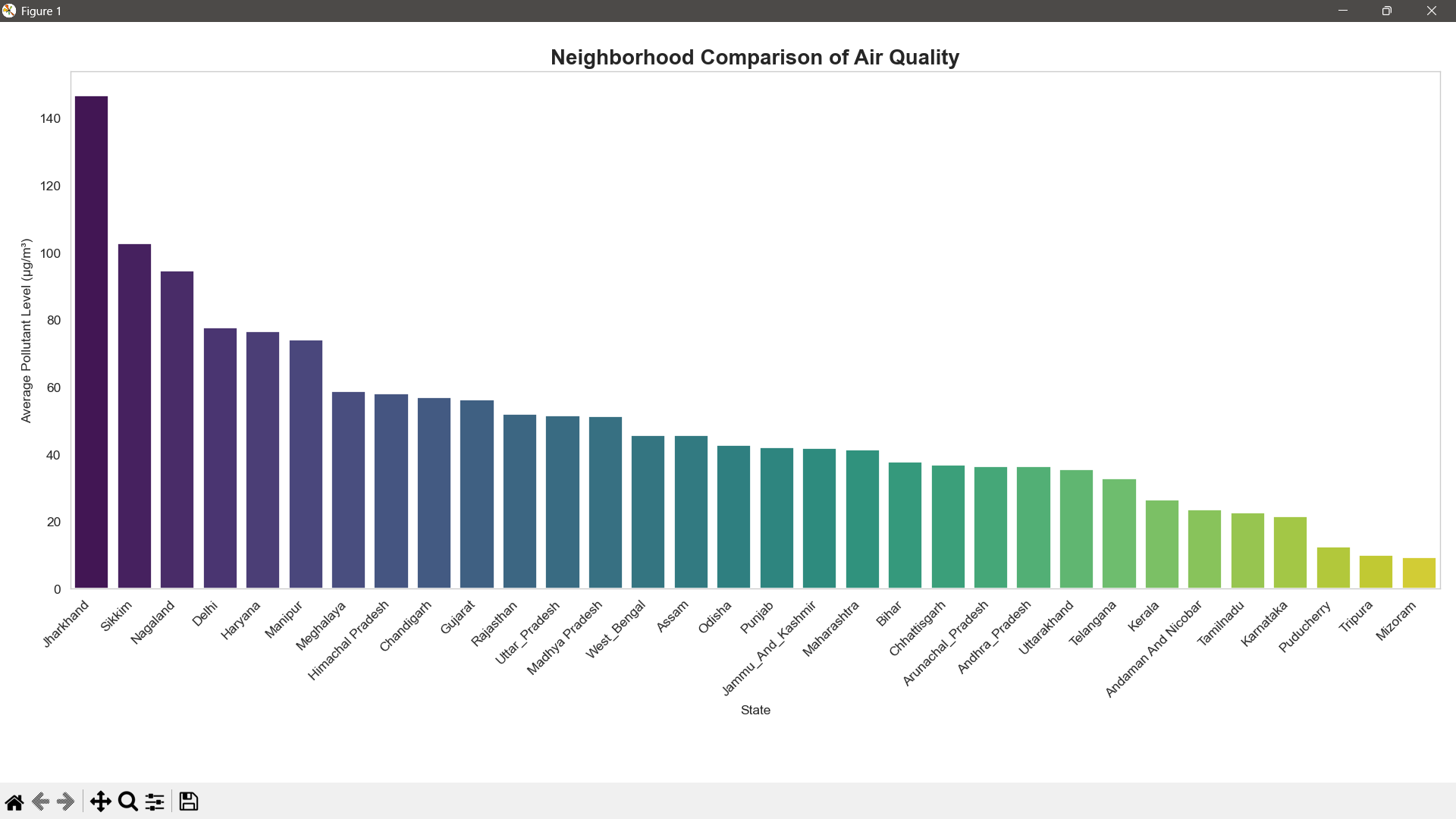


Fig: Output of Objective2

**Objective 3: Correlation Analysis Between Pollutants**

**i. Introduction**The bar chart provides a clear visual representation of the differences in air quality across various states in India. By examining the data, we can identify the states with the highest and lowest average pollutant levels, which is a crucial step in understanding the regional variations and targeting areas that require more attention.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ii. General Description**

**Prioritizing Interventions**: Identifying the states with the highest pollutant levels allows policymakers and environmental agencies to prioritize their efforts and resources to address the most pressing air quality issues.

**Benchmarking Performance**: Comparing the pollutant levels across states can help establish benchmarks and set targets for improving air quality in underperforming regions.

**Identifying Best Practices**: The states with the lowest pollutant levels can serve as models, allowing for the study and adoption of their best practices in air quality management.

**Monitoring Progress**: Tracking the changes in pollutant levels over time can help evaluate the effectiveness of air quality improvement initiatives and guide future policy decisions.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**iii. Specific Requirements, Functions and Formulas**

df['pollutant\_avg'] = pd.to\_numeric(df['pollutant\_avg'], errors='coerce')

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

numeric\_data = df.select\_dtypes(include=['number'])

correlation\_matrix = numeric\_data.corr()  
  
These operations assisted us in graphing the data suitably without affecting visual clarity and interpretability.

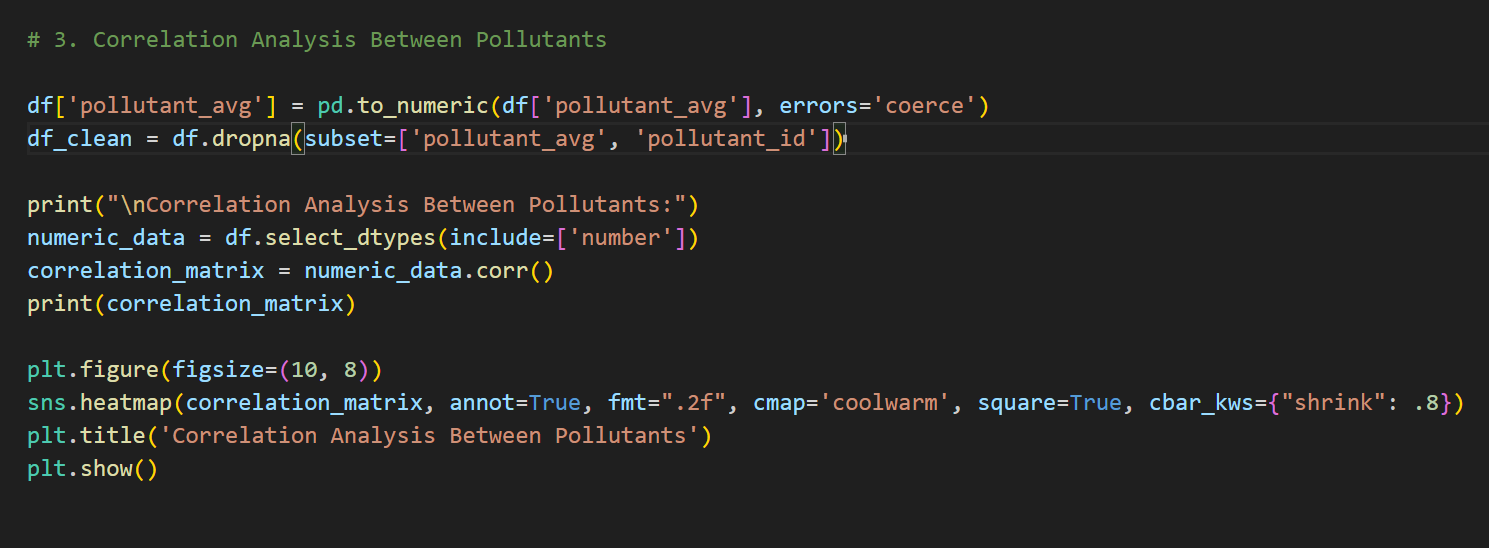


Fig: Code of Objective3

**iv. Analysis Results**

* pollutant\_min, pollutant\_max, and pollutant\_avg have strong positive correlations with each other.
* latitude and longitude show very weak correlation with pollutant values.
* High correlation among pollutant metrics indicates internal data consistency.
* Geographical coordinates have minimal impact on pollutant levels in this dataset.
* Diagonal values are 1.00, representing perfect self-correlation.
* **v. Visualization**  
  The heatmap displays **correlation coefficients** between numeric variables in the dataset.
* Dark red (near 1) indicates a **strong positive correlation**, while dark blue (near 0 or negative) indicates **weak or negative correlation**.
* pollutant\_avg shows high correlation with both pollutant\_max (0.88) and pollutant\_min (0.82), meaning it is heavily influenced by them.
* pollutant\_min and pollutant\_max also have a strong correlation (0.64), showing they tend to vary together.
* latitude and longitude show **very low correlation** with pollutants, implying that **geographic location doesn't significantly affect pollution levels** in this data.
* The color bar on the right helps interpret the correlation strength visually.



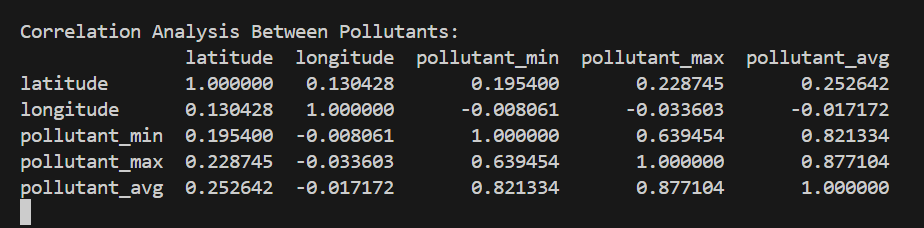


Fig: Output of Objective3

**Objective 4: Pollution Hotspot & Correlation**

**i. Introduction**  
Outlier detection in air quality data involves identifying unusually high or low pollutant readings that deviate significantly from typical values. These outliers may result from sensor errors, extreme weather conditions, or pollution spikes and can distort analysis if left unaddressed. Detecting them helps improve data accuracy and highlights critical pollution events. Common techniques include the Interquartile Range (IQR), Z-score analysis, and visual tools like boxplots. Focusing on columns such as `pollutant\_min`, `pollutant\_max`, and `pollutant\_avg` allows for effective identification of these anomalies.

**ii. General Description**  
Outlier detection in air quality data is important for identifying abnormal pollution levels that deviate sharply from typical patterns. These outliers can occur due to sudden environmental events (like industrial accidents or fires), sensor malfunctions, or human error during data entry. Recognizing and handling outliers helps improve the accuracy of statistical analysis and visualizations. It also aids in highlighting potentially hazardous conditions that might require immediate attention or public health alerts. By isolating or correcting outliers, we ensure the dataset reflects a more accurate and meaningful representation of air quality trends.

**iii. Specific Requirements, Functions and Formulas**

**Specific Requirements:**

 Dataset must include clean numeric values for pollutant\_avg and pollutant\_id.

 Each pollutant should have enough data to compute statistical metrics like quartiles.

 Requires matplotlib and seaborn for visualization.

**Functions Used** plt.figure() & sns.boxplot(): Create a box plot for visualizing data distribution and outliers.

 plt.title(), plt.xlabel(), plt.ylabel(), etc.: Customize the appearance of the plot.

 .quantile(): Calculate Q1 and Q3 for each pollutant.

 Dictionary to count outliers per pollutant.

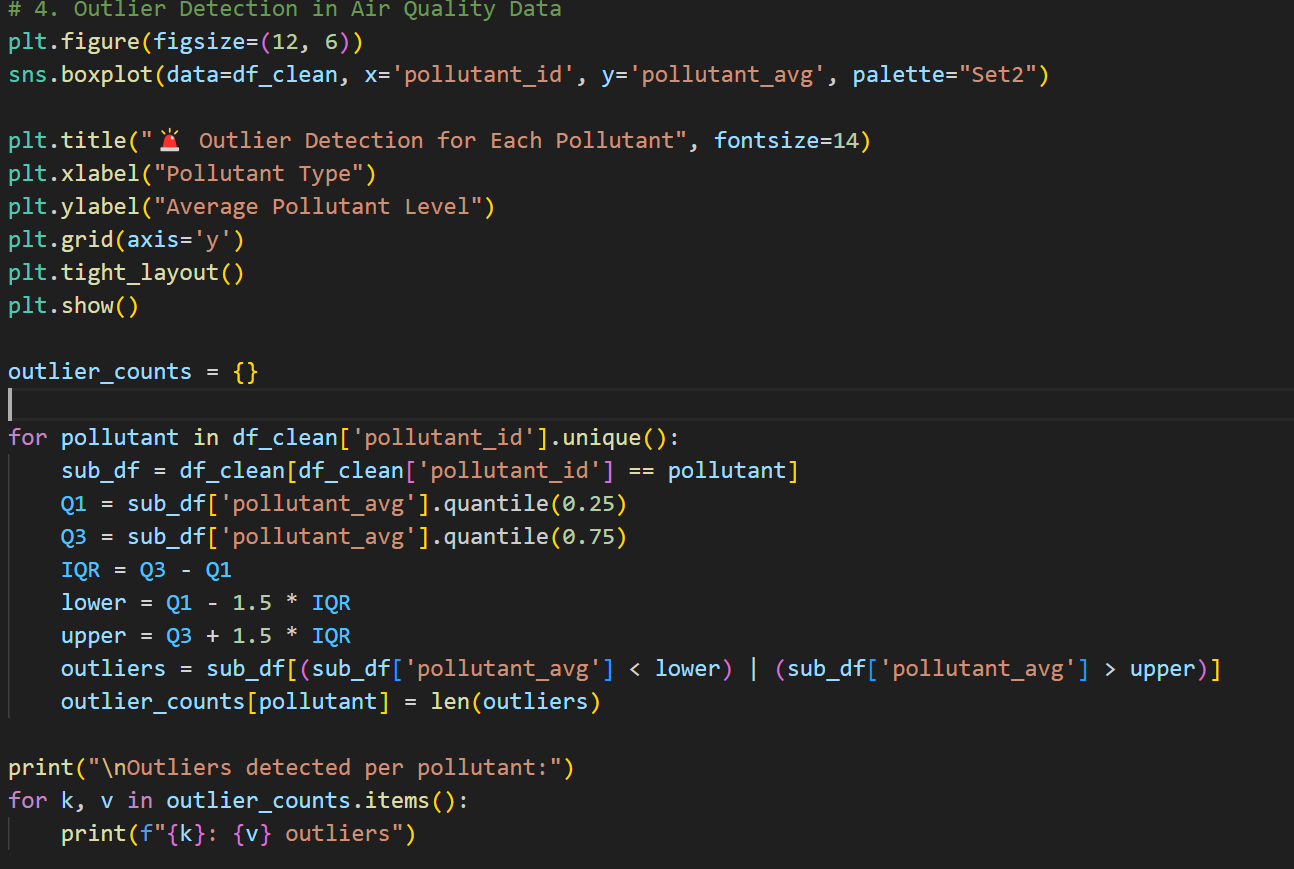


Fig: Code of Objective4

**iv. Analysis Results**  
Outlier Detection Analysis

A boxplot is generated for each pollutant\_id to visualize the distribution of average pollutant levels (pollutant\_avg), highlighting potential outliers as individual points beyond whiskers.

The code uses the Interquartile Range (IQR) method to detect outliers:

For each pollutant, it calculates Q1 (25th percentile) and Q3 (75th percentile).

Any value below Q1 - 1.5 × IQR or above Q3 + 1.5 × IQR is flagged as an outlier.

A count of outliers per pollutant is printed, helping identify which pollutants have more extreme or irregular readings.

The result is useful for understanding anomalous pollution events or data irregularities, which can inform deeper investigation or policy decisions.

**v. Visualization**

 The boxplot displays **average pollutant levels** (pollutant\_avg) for each pollutant type (NO2, PM2.5, NH3, OZONE, PM10, SO2, CO).

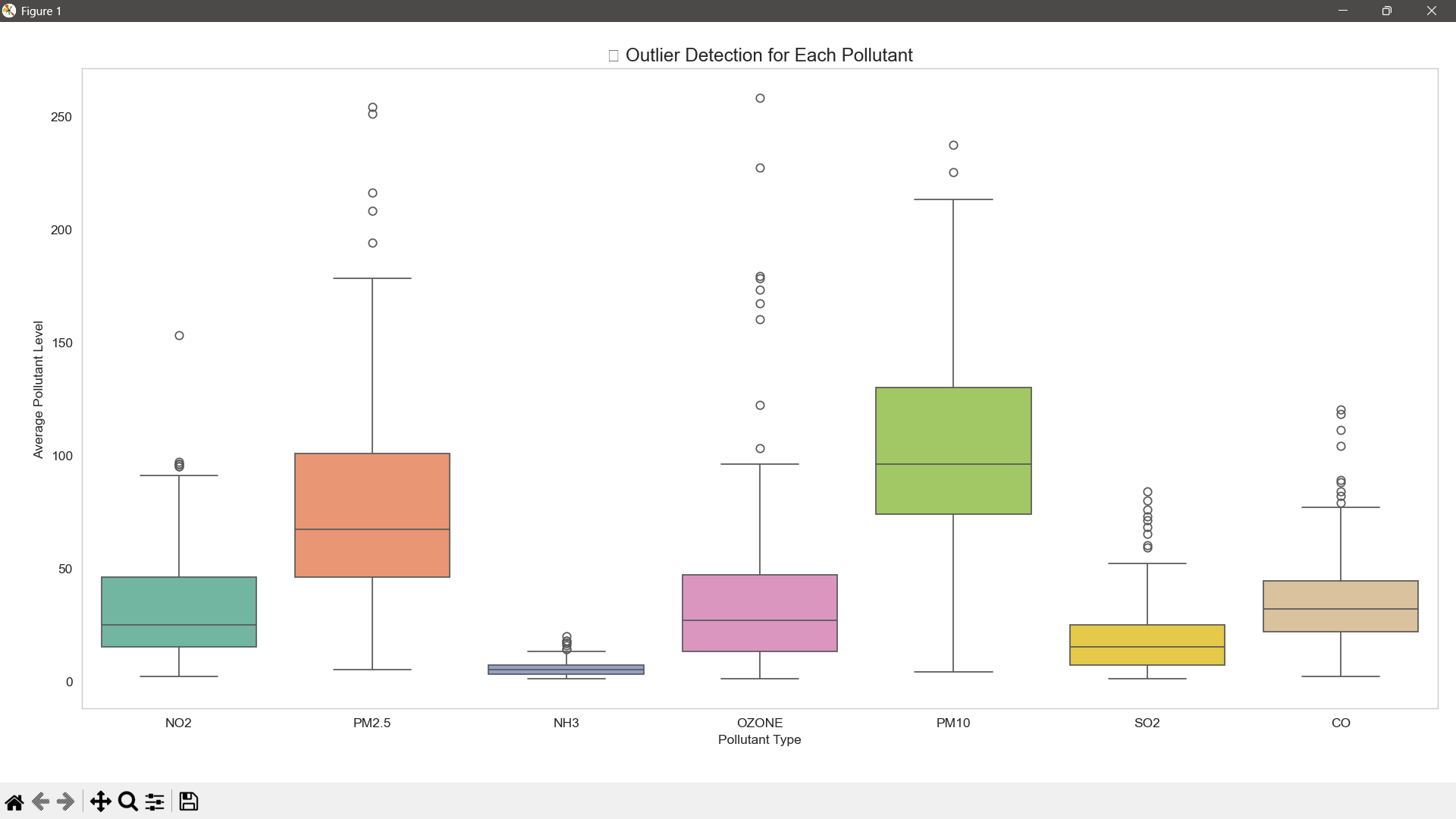
 Each box shows the **interquartile range (IQR)**, with the middle line representing the **median**.

 **Whiskers** extend to 1.5 × IQR; data points beyond this range are **outliers**, shown as individual dots.

 Pollutants like **OZONE**, **PM2.5**, and **PM10** show a **high number of outliers**, indicating irregular or extreme pollution levels.

 **NH3** and **SO2** have tighter IQRs, showing less variability in average levels.

 The visualization helps identify pollutants with **unstable behavior** or **exceptional readings** needing further analysis.



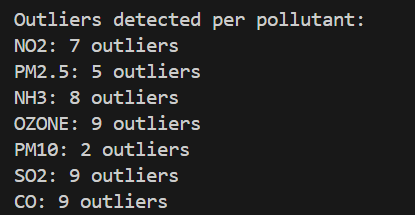


Fig: Output of Objective4

**Objective 5: Seasonal Variation Analysis**

**i. Introduction**  
Data Visualization for Public Awareness plays a crucial role in making complex air quality data understandable and accessible to the general public. Here's a concise explanation:

Data visualization translates numerical air quality data into clear, visual formats like charts, maps, and infographics. These visuals help non-technical audiences quickly grasp trends, health risks, and pollution hotspots. Effective visualization can raise awareness, influence public behaviour (like reducing outdoor activities on high pollution days), and support informed decision-making by both citizens and policymakers. In short, it transforms raw data into actionable insight that can drive community engagement and environmental responsibility.

**ii. General Description**  
Data visualization for public awareness is the process of transforming complex air quality data into clear, impactful visuals—such as heatmaps, line graphs, and box plots. These visuals help non-technical audiences quickly understand the severity and patterns of pollution in different areas. By highlighting trends, outliers, and comparisons across pollutants or locations, such visual tools raise awareness about environmental issues, promote informed decision-making, and encourage the public to support or engage in pollution control efforts. Effective visualizations serve as a bridge between data analysts, policymakers, and everyday citizens.

**iii. Specific Requirements, Functions and Formulas**

**Data Aggregation & Cleaning**

* value\_counts() – Counts frequency of each pollutant.
* pd.to\_datetime() – Converts date strings to datetime objects.
* dropna() – Removes rows with missing values.
* pd.to\_numeric() – Ensures numeric values, coercing errors to NaN.
* groupby() + mean() – Computes average pollution level by day type.
* sort\_values() + head() – Extracts top 5 pollution spikes.
* apply(lambda...) – Checks if pollutant levels exceed defined limits.

 **Matplotlib & Seaborn for Visualization**

* plt.figure(figsize=...) – Sets figure size.
* sns.barplot(...) – Creates bar plots for categories.
* sns.histplot(...) – Plots histogram with optional KDE curve.
* plt.title(), plt.xlabel(), plt.ylabel() – Adds labels and titles.
* plt.grid() – Adds grid lines for better readability.
* barplot.text(...) – Annotates bars with exact values.

**Formulas & Logic**

1. Day Classification  
     
   df['day\_of\_week'] = df['last\_update'].dt.dayofweek

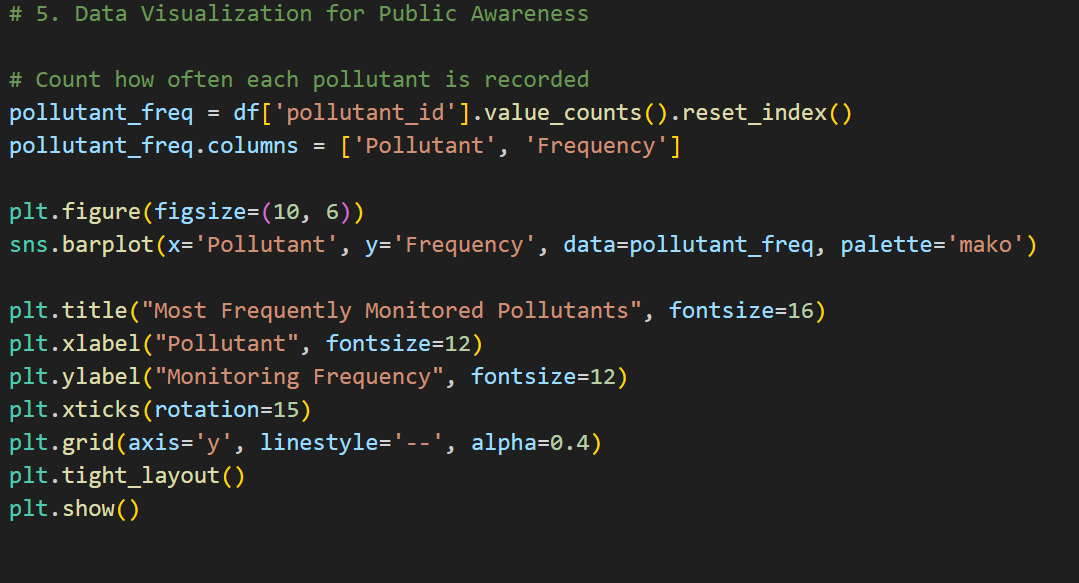
df['day\_type'] = df['day\_of\_week'].apply(lambda x: 'Weekend' if x >= 5 else 'Weekday')

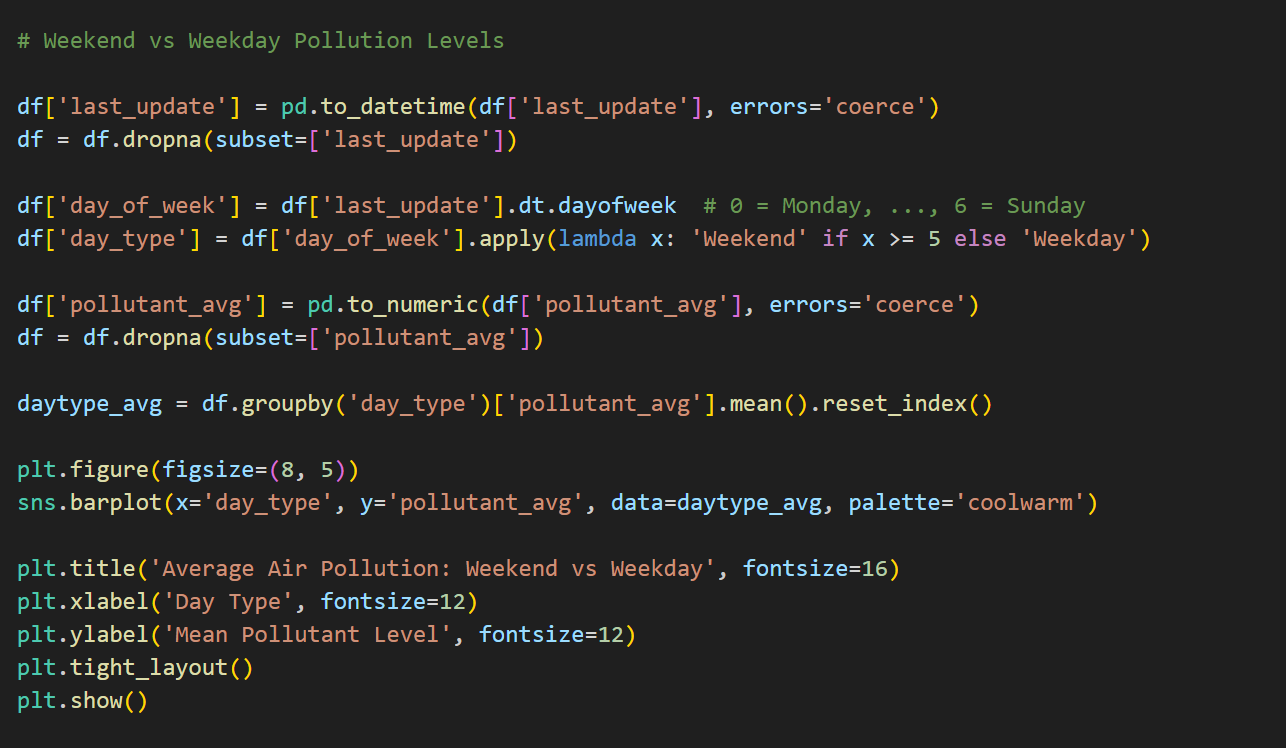
1. Safe Limits Check

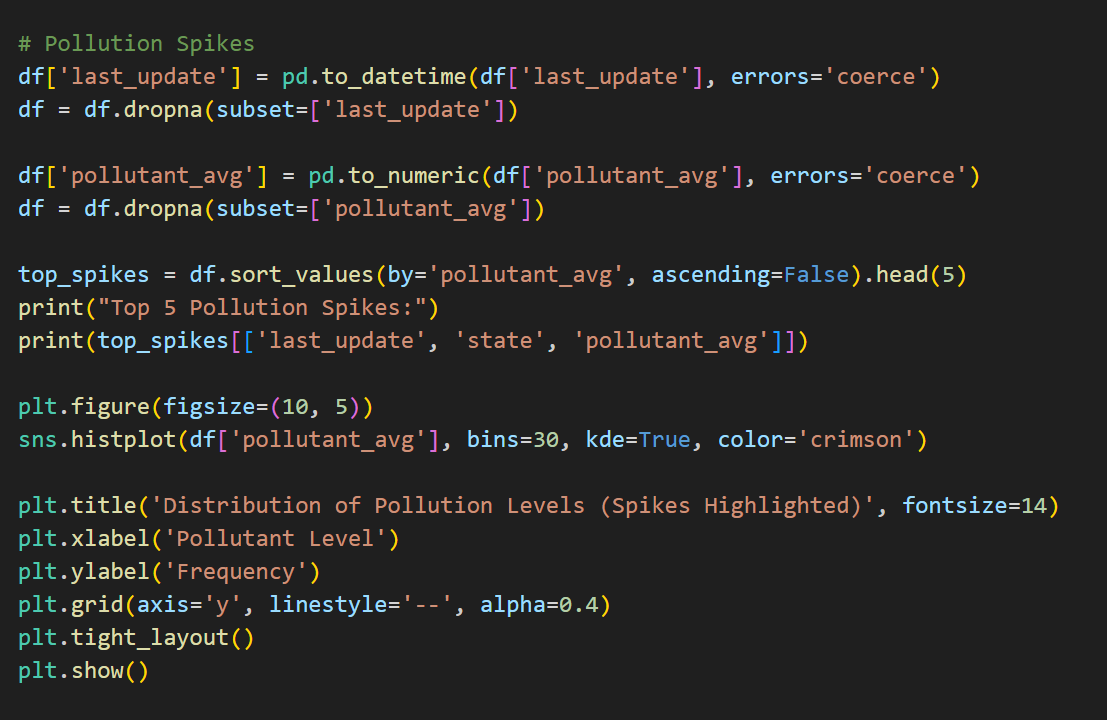
row['pollutant\_max'] > safe\_limits[row['pollutant\_id']]

1. Pollution Spike Detection

Top 5 highest pollutant averages sorted by value.







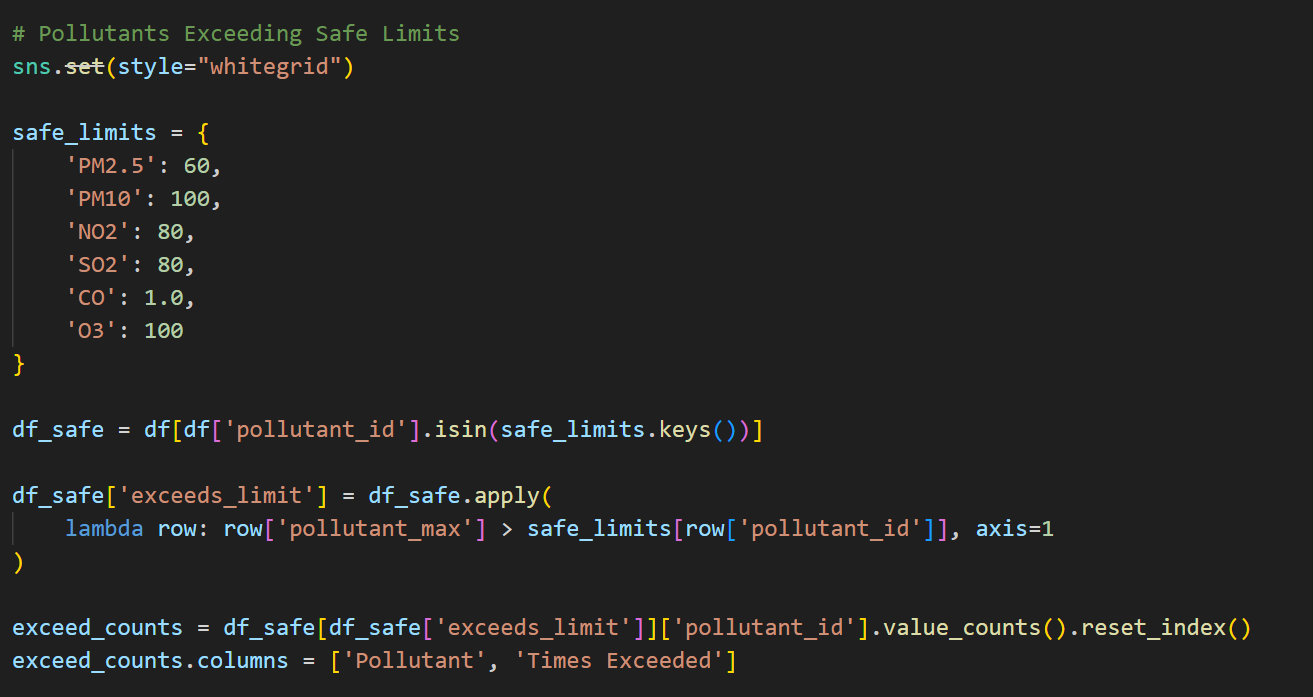




Fig: Code of Objective5

**iv. Analysis Results**

**Most Frequently Monitored Pollutants**

* Identifies which pollutants are most often recorded.
* Highlights focus areas like **PM2.5** or **PM10**.

**Weekend vs Weekday Pollution Levels**

* Compares average pollution on weekdays vs weekends.
* Reveals impact of **human activity patterns** on air quality.

**Pollution Spikes**

* Shows top 5 highest pollution events with time and location.
* Histogram shows overall pollution level distribution.
* Helps detect **outliers and extreme events**.

**Pollutants Exceeding Safe Limits**

* Counts how often each pollutant exceeds defined safe thresholds.
* Identifies **health-risk pollutants** (e.g., frequent PM2.5 breaches).

**v. Visualization**

* **SO2 and CO** are the most frequently monitored pollutants (both ~200 times).
* **PM2.5 and PM10** follow closely behind.
* **NH3** is the least monitored among the listed pollutants.

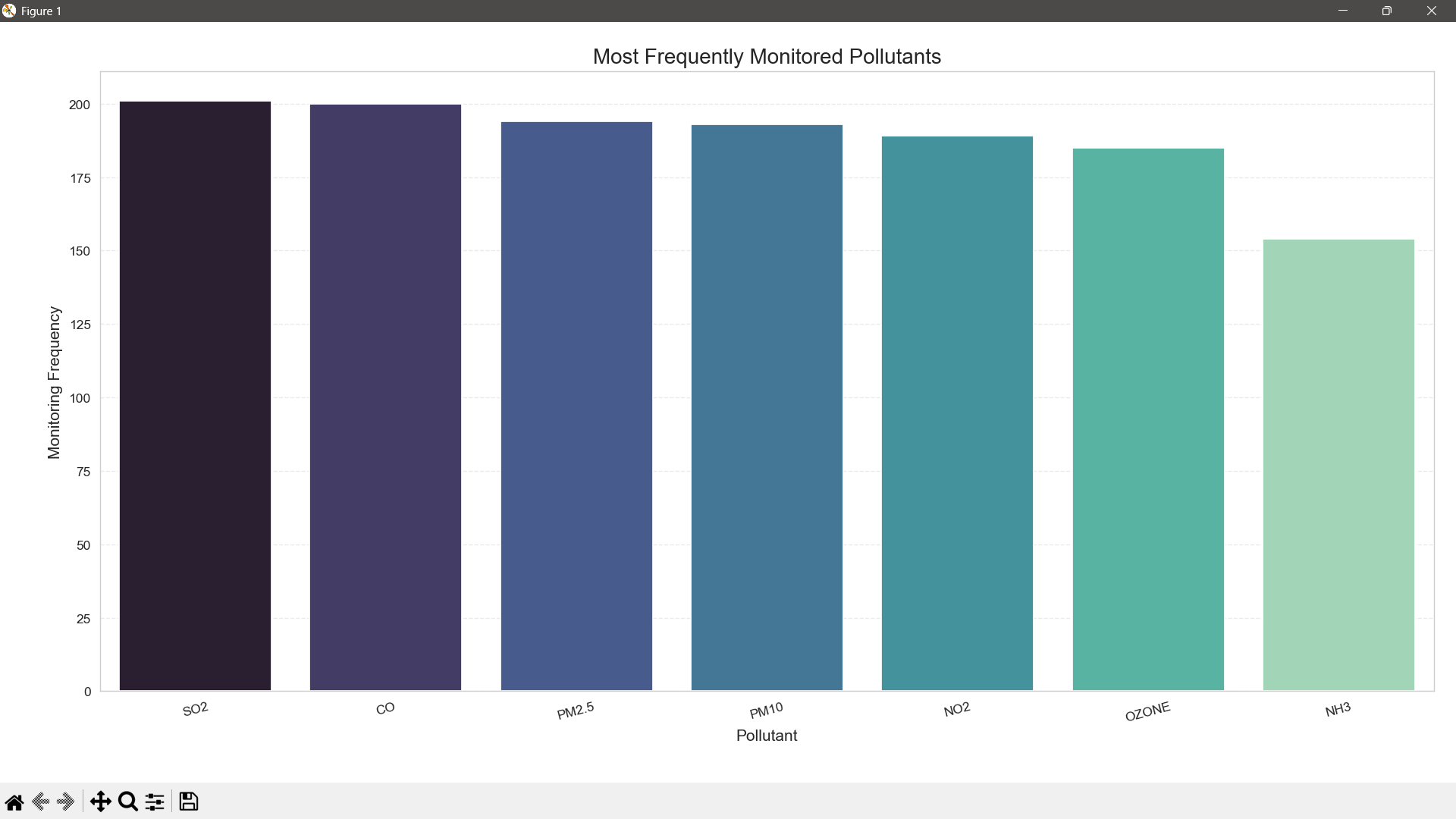


Fig: Output of Objective5

* **Weekday pollution is higher** than weekend pollution.
* Suggests increased pollution from **weekday activities** such as traffic and industrial operations.
* Indicates a **drop in emissions** during weekends, possibly due to reduced human activity.

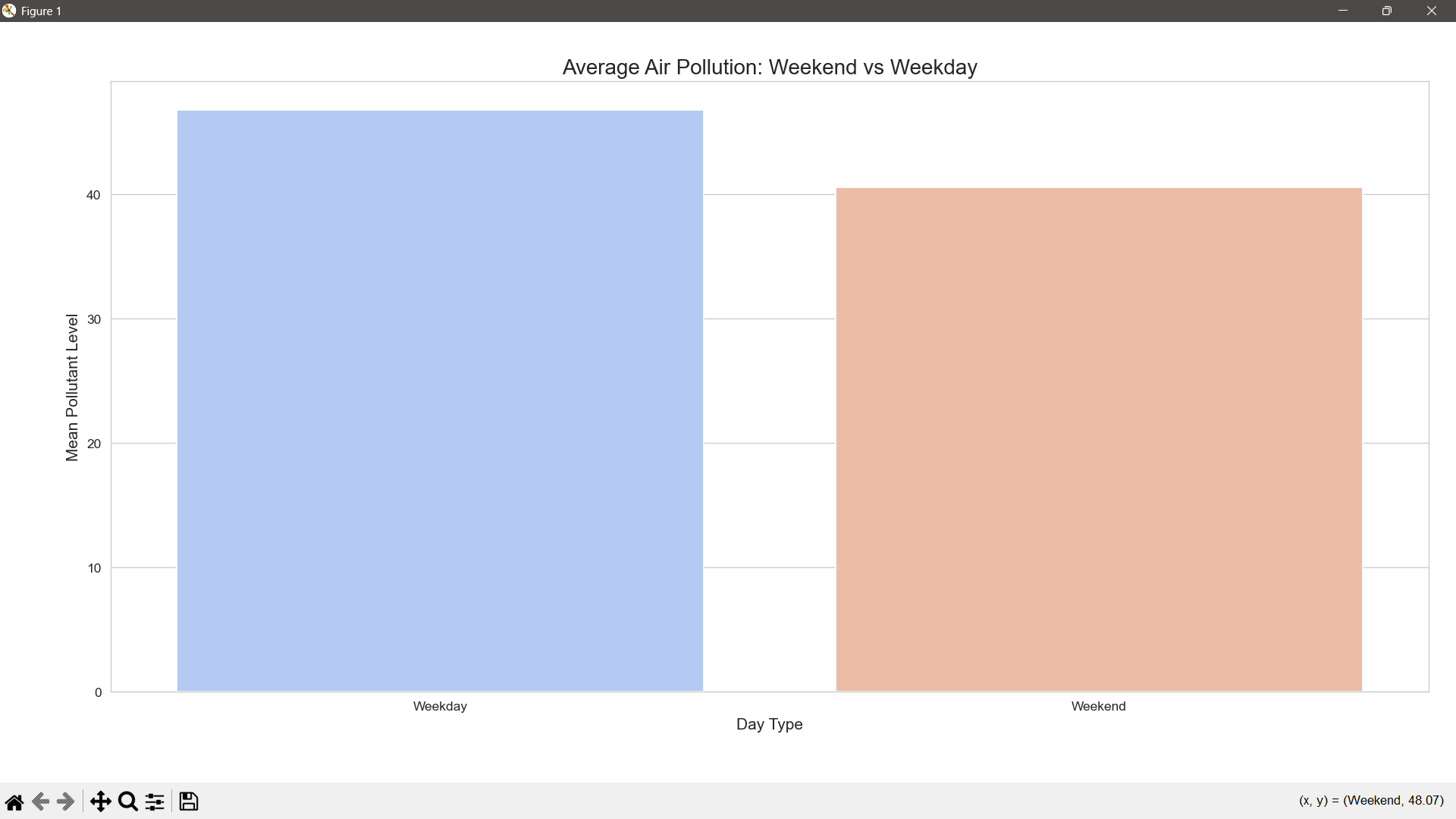


Fig: Output of Objective5

* **Skewed Distribution**: Most pollution levels are concentrated below **50**, indicating low to moderate pollution is common.
* **Pollution Spikes**: A **long right tail** shows **few extreme high values** (spikes) — these are rare but significant.
* **Potential Outliers**: Values above 150–200 could be investigated for causes (e.g., industrial events, traffic, weather).

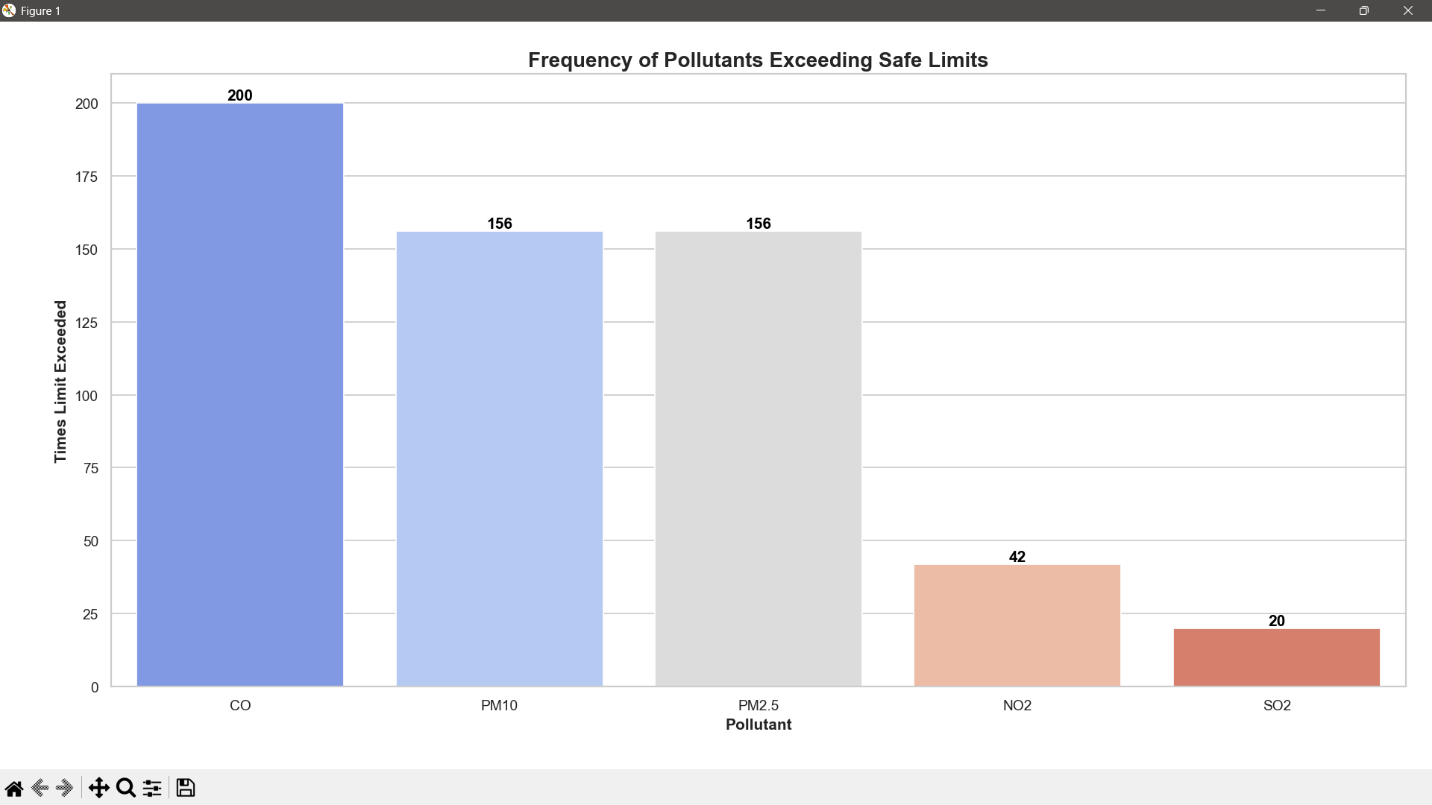


Fig: Output of Objective5

**CONCLUSION**

This air quality data analysis project aimed to uncover patterns and raise public awareness about environmental pollution. Using Python and data visualization tools, we explored various aspects such as the frequency of pollutant monitoring, weekend vs. weekday pollution trends, extreme pollution spikes, and pollutants exceeding safe limits.

We found that certain pollutants like PM2.5 and PM10 are monitored more frequently, reflecting their high impact on public health. The analysis also revealed that pollution levels tend to vary slightly between weekdays and weekends, suggesting the influence of human activity such as traffic and industry. A detailed spike analysis helped identify instances where pollution levels were abnormally high, while the safe limit comparison exposed pollutants that often cross recommended thresholds.

Visual tools like bar charts, histograms, and boxplots were used to make the findings more accessible and engaging. These charts effectively communicated risks and trends to a non-technical audience, aiding in public understanding and awareness.

In conclusion, the project not only highlights the current state of air quality but also emphasizes the need for proactive measures. It serves as a useful guide for policymakers, environmentalists, and the general public to make informed decisions and promote a healthier, cleaner environment.

Overall, the findings of this project have significant significance for environmental analysts, policymakers, and the general public. Having a greater understanding of how contaminants evolve over time and what conditions influence their levels allows for more informed decisions to be made about policies for reducing pollution and public health protection. This study showcases the ability of data science in resolving real environmental issues of the actual world by exhibiting how facts backed by data can influence meaningful actions to improve the air and construct a cleaner world for all.

**FUTURE SCOPE**

* **Real-Time Data Integration:** Enhance the system to process live air quality data using APIs from monitoring agencies**.**
* **Predictive Modelling:** Implement machine learning algorithms to forecast pollution levels and potential spikes**.**
* **Geospatial Analysis:** Use GIS tools to map pollutant concentration across different regions for localized insights**.**
* **Health Impact Correlation:** Link air quality data with public health records to assess environmental effects on human health.
* **Mobile App/Website Development:** Create an interactive platform to share real-time pollution insights with the public.
* **Seasonal & Weather Pattern Study**: Analyze the influence of seasons and weather variables on pollution trends.
* **Policy Recommendation System:** Use data insights to suggest targeted policies for pollution control in highly affected areas.
* **Multi-City Comparison:** Scale the project to compare air quality across multiple cities or countries.
* **IoT Device Integration:** Connect to smart sensors for continuous air quality monitoring and automated alerts**.**
* **Public Engagement Dashboard:** Develop a user-friendly dashboard for citizens to monitor, report, and stay aware of local air quality.

**REFERENCES**

**[1] J. Grus, *Data Science from Scratch*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2019. ISBN: 9781492041139.**

**[2] K. Behrman, *Foundational Python for Data Science*, 1st ed. Boston, MA, USA: Pearson, 2022. ISBN: 9780136624356.**

**[3] R. Thareja, *Data Science and Machine Learning Using Python*, 1st ed. New Delhi, India: McGraw-Hill, 2022. ISBN: 9355322143, 9789355322142.**

**[4] M. A. Hameed, *Python for Data Science*, 1st ed. New Delhi, India: Wiley, 2021. ISBN: 9789354243479.**

**[5] DataCamp, “Learn data science online,” [Online]. Available:** [**https://www.datacamp.com/**](https://www.datacamp.com/)

**[6] W3Schools, “Python Tutorial,” [Online]. Available:** [**https://www.w3schools.com/**](https://www.w3schools.com/)

**[7] GeeksforGeeks, “Python Introduction – Matplotlib,” [Online]. Available:** [**https://www.geeksforgeeks.org/python-introduction-matplotlib/?ref=gcse\_ind&\_gl=1\*1m9aw1h\*\_up\*MQ..\*\_gs\*MQ..&gclid=Cj0KCQjwnui\_BhDlARIsAEo9GuuN56fYbs1OT3P8Yo06P3O8dvCIAVGrqgXQ71c5TUHn8X5rxFrtVwIaAhcdEALw\_wcB&gbraid=0AAAAAC9yBkC\_PJFGR9mdaD4wvLFUXV1JT**](https://www.geeksforgeeks.org/python-introduction-matplotlib/?ref=gcse_ind&_gl=1*1m9aw1h*_up*MQ..*_gs*MQ..&gclid=Cj0KCQjwnui_BhDlARIsAEo9GuuN56fYbs1OT3P8Yo06P3O8dvCIAVGrqgXQ71c5TUHn8X5rxFrtVwIaAhcdEALw_wcB&gbraid=0AAAAAC9yBkC_PJFGR9mdaD4wvLFUXV1JT)