PYTHON PROJECT REPORT

(Project Semester: January-April 2025)

Title of the Project: Air Quality Analytics: A Dashboard-Driven Analysis

Submitted by: Shivendra Patel

Registration No.: **12315502**

Programme and Section**: B.Tech CSE** (K23ER)

Course Code: INT375

Under the Guidance of: Dr. Karan Bajaj (UID : 32130)

Discipline of CSE/IT Lovely School of Computer Science & Engineering Lovely Professional University,

Phagwara

DECLARATION

I, **Shivendra patel,** student of **Bachelors of Technology (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: 03-April-2025

Registration No.: 12315502

Name of the Student: Shivendra Patel

CERTIFICATE

This is to certify that **Shivendra Patel** bearing Registration No. **12315502** has completed **INT375** project titled **“Air Quality Analytics: A Dashboard-Driven Analysis”** under my guidance and supervision. To the best of my knowledge, the present work is the result of her original development, effort, and study.

Dr. Karan Bajaj Assistant Professor (School of Computer Science & Engineering)

Lovely Professional UniversityPhagwara, Punjab

Date: **04-April-2025**

ACKNOWLEDGMENT

I would like to express my sincere gratitude to **Dr. Karan Bajaj Sir**, my project guide, for their invaluable support, guidance, and encouragement throughout the development of this project. Their expert insights and constructive feedback have been instrumental in shaping the project's outcome.

I am also thankful to **Lovely Professional University** for providing a conducive learning environment and access to resources that made this project possible. Additionally, I extend my appreciation to my professors and peers for their continuous motivation and insightful discussions, which greatly enhanced my understanding of the subject.

Lastly, I would like to acknowledge the unwavering support of my family and friends, whose encouragement has been a source of inspiration throughout this journey.

TABLE OF CONTENTS

1. Introduction
2. Source of Dataset
3. Dataset Preprocessing
4. Analysis on Dataset (for each objective)
    - o i. General Description
    - o ii. Specific Requirements
    - o iii. Analysis Results
    - o iv. Visualization
5. Conclusion
6. Future Scope
7. References

1. INTRODUCTION

Air pollution is a pressing environmental concern with severe implications for public health, climate change, and overall quality of life. The rapid industrialization and urbanization in recent decades have significantly deteriorated air quality, especially in developing nations like India. As a result, continuous monitoring and analysis of air pollutants are essential to inform policymakers, researchers, and the public.

This project titled "Air Pollution Dashboard: Analysis & Insights" utilizes Microsoft Excel to perform a comprehensive analysis of air quality data collected from various monitoring stations across Indian cities. The objective is to explore pollution trends, identify hotspots, and present the findings in a user-friendly dashboard format.

Through this project, I have employed various data processing and visualization techniques such as data cleaning, statistical functions, Pivot Tables, Pivot Charts, conditional formatting, and dynamic slicers. These tools enable efficient data exploration and storytelling without the need for programming, showcasing the power and flexibility of Excel for data analytics.

The dashboard created not only provides insights into pollutant levels such as PM10, SO2, and NH3 but also highlights temporal and spatial patterns across regions. This project demonstrates how Excel can be a powerful ally in environmental data analysis, capable of driving awareness and supporting better decision-making.

2. SOURCE OF DATASET

The dataset used for this project is a compila on of publicly accessible air quality readings from the Indian Government's open data portals and Central Pollu on Control Board (CPCB). It contains real- me and historical records of pollutant concentra ons measured at different sta ons across states and ci es.

The dataset includes a ributes such as:

- Country, State, and City
- Sta on Name
- Date and Time of Reading (last_update)
- Geographic Coordinates (la tude, longitude)
- Pollutant ID (e.g., PM10, SO2, NH3)
- Minimum, Maximum, and Average Concentra on Levels

To ensure the dataset was suitable for analysis, it was downloaded in CSV format and preprocessed using Excel before any analy cal opera ons.

Link : h ps://www.data.gov.in/search? tle=air%20pollu on&type=resources&sortby=_score

3. DATASET PREPROCESSING

Raw environmental datasets often come with inconsistencies such as missing entries, redundant values, and formatting errors. To prepare the dataset for reliable analysis, the following preprocessing steps were undertaken:

1. **Replacement of Missing Values:**
   - The dataset contained several "na" entries in the numeric columns such as pollutant_min, pollutant_max, and pollutant_avg.
   - These values were replaced using column-wise mean imputation through Python .mean and filtering techniques.
2. **Column Standardization:**
   - Column headers initially appeared incorrect or repeated as "country" due to formatting issues.
   - Appropriate names such as state, city, station, pollutant_id, etc., were assigned.
3. **Data Type Verification:**
   - Numerical columns were reformatted to appropriate data types for accurate calculations.
   - Timestamps were converted into consistent date-time format to allow time-series plotting.
4. **Filtering and Sorting:**
   - The data was filtered by pollutant types and sorted based on time, region, and pollutant levels.
   - Duplicate rows were removed to ensure clean and unique observations.

After preprocessing, the dataset was ready for in-depth analysis using Excel functionalities.

4. ANALYSIS ON DATASET

4.1 General Description of the Dataset

4.1 General Description

The dataset comprises air quality readings taken from multiple cities and stations in India. The core metrics analyzed include the minimum, maximum, and average concentration levels of pollutants such as PM10, SO_2, and NH_3. Additional fields like date of recording and station location support temporal and spatial analysis.

The data covers multiple Indian states and includes sufficient granularity to compare pollution patterns across cities, times of day, and pollutant types.

4.2 Specific Objectives

The primary objectives of this project include:

1. Identifying cities and stations with the highest levels of pollutants.
2. Analyzing pollution trends over time for various pollutants.
3. Comparing average pollutant levels across states.
4. Creating an interactive Excel dashboard with filters for pollutant type, city, and station.
5. Highlighting pollution hotspots and clean zones.

4.3 Analysis Results

- **High Pollution Areas:** Cities such as Delhi, Mumbai, and Lucknow showed consistently high levels of PM10.
- **Cleaner Regions:** Cities in the Northeastern states such as Assam showed lower pollution levels.
- **Temporal Patterns:** Peak pollution was often observed in the early morning and late evening hours.
- **Pollutant Comparison:** PM10 was found to be the most prevalent and most dangerous pollutant in terms of volume and impact.

Tools and Libraries Used

- **Python 3.10+**
- **Jupyter Notebook**
- **Pandas** – for data manipulation
- **NumPy** – for numerical computations
- **Matplotlib** and **Seaborn** – for static visualizations
- **Plotly** – for interactive graphs
- **Datetime** – for timestamp formatting
- **Missingno** – for visualizing missing data

4.2 Specific Requirements and Objectives

The primary goal of this project was to analyze and visualize air quality data using **Python** and its data science libraries. This project leverages tools like **Pandas, Matplotlib, Seaborn, and Plotly** to process pollution data, identify high-risk regions, study pollutant trends over time, and build interactive charts and graphs for insight extraction.

The key objectives of this project were:

1. To Determine Which City or Station Has the Highest Pollution Levels

This objective focused on identifying cities and stations with the highest average pollutant concentrations, particularly **PM10**, **SO_2**, and **NH_3**.

- **Why it's important:**
  Identifying the most polluted areas helps environmental authorities prioritize mitigation strategies and allocate resources for pollution control.
- **How it was done:**
  Using **Pandas**, the dataset was grouped by city and station, and mean values of pollutant concentrations were calculated. The results were sorted to reveal the worst-affected locations.
- python
- CopyEdit
- top_polluted_cities = df.groupby('city')['pollutant_avg'].mean().sort_values(ascending=False)

2. To Compare Pollution Levels Across Different Pollutants

This objective explored which pollutants were most prevalent in the dataset and posed the greatest environmental threat.

- **Approach:**
  The data was grouped by pollutant_id, and the mean values were calculated to compare PM10, SO2, and NH3. Pie charts and bar graphs (using **Matplotlib** and **Seaborn**) were used to visualize their overall contribution.
- python
- CopyEdit
- pollutant_distribution = df.groupby('pollutant_id')['pollutant_avg'].mean()

3. To Analyze Temporal Trends in Air Quality

This section aimed to understand how pollution varied across time—by days, weeks, or seasons.

- **Method:**
  The last_update column was converted to a datetime object. Data was resampled to daily or weekly frequency to calculate and visualize average pollution trends using **line plots**.
- python
- CopyEdit
- df['last_update'] = pd.to_datetime(df['last_update'])
- time_trends = df.set_index('last_update').resample('W')['pollutant_avg'].mean()

4. To Compare State-Wise Air Pollution Performance

This objective focused on comparing average pollutant levels across different Indian states.

- **Steps Taken:**
  The dataset was grouped by state, and average values of each pollutant were calculated. Horizontal bar charts highlighted the most and least polluted states.
- python
- CopyEdit

- state_avg = df.groupby('state')['pollutant_avg'].mean().sort_values(ascending=False)

5. To Build Interactive and Static Visualizations

A final goal was to translate the findings into both static and interactive visual formats for clear, data-driven storytelling.

- **Tools Used:**
  - **Matplotlib & Seaborn** for standard visuals
  - **Plotly** for interactive bar, line, and pie charts
  - **Missingno** for missing value diagnostics

4.3 Analysis Results

The Python ecosystem enabled deeper, more dynamic analysis using a variety of tools and visual techniques:

Data Grouping and Aggregation

Using **Pandas groupby**, pollution data was grouped and summarized by:

- city, state, and pollutant_id
- last_update (for time-series analysis)

Metrics computed included:

- Mean, max, and min pollutant levels
- Weekly and daily averages
- Pollution trends over time

Missing Value Handling

- 'na' entries were converted to NaN using pandas.to_numeric(errors='coerce')
- Missing values in pollutant_min, pollutant_max, and pollutant_avg were filled with column-wise means:

python

CopyEdit

df['pollutant_min'].fillna(df['pollutant_min'].mean(), inplace=True)

- **Missingno** was used to visualize missing patterns.

Sorting and Filtering

Python enabled flexible sorting/filtering to:

- Isolate top 10 polluted cities/stations
- Focus on specific pollutants (e.g., only PM10)
- Analyze pollution on specific dates or regions

Custom Calculations

- Average pollutant levels were computed from pollutant_min and pollutant_max when pollutant_avg was missing.
- Time-based resampling helped detect seasonal or temporal shifts.

4.4 Visualizations and Insights

Python visualizations revealed several critical findings:

City Comparison

Bar charts (Matplotlib/Seaborn) were used to highlight the top 10 most polluted cities.

Insight Drawn:
Delhi, Mumbai, and Lucknow consistently reported high PM10 levels, significantly exceeding safe limits.

Pollutant Analysis

Pie charts showed the overall contribution of each pollutant (PM10, SO2, NH3).

Insight Drawn:
PM10 was the dominant pollutant across most cities. NH3 was less frequently recorded and typically lower in concentration.

State-Wise Pollution

Horizontal bar charts compared average pollution by state.

Insight Drawn:
States in North and Central India—like Uttar Pradesh and Delhi—exhibited the highest pollution levels. States like Assam had cleaner air comparatively.

Time-Based Trends

Line graphs plotted daily/weekly pollutant averages using resampled data.

Insight Drawn:
Pollution levels consistently spiked during winter (likely due to vehicular emissions and stubble burning). The monsoon season showed a noticeable dip in pollutant concentration.

5. CONCLUSION

This project provided an in-depth, programmatic analysis of air pollution levels across multiple Indian cities using **Python as the core analytical tool**. Leveraging Python's robust data science libraries, we transformed a raw and incomplete dataset into a well-structured foundation for meaningful environmental insights.

The process began with **data cleaning and preprocessing**, where missing values in pollutant measurements (PM10, SO_2, and NH_3) were addressed through statistical imputation techniques such as mean replacement. Data types were corrected, timestamps were parsed for time-series analysis, and unnecessary records were filtered out — all using efficient and scalable Python code with libraries like **Pandas** and **NumPy**.

Once cleaned, the dataset was explored using a combination of **grouping**, **aggregation**, and **custom metrics** to evaluate pollution across geographic and temporal dimensions. Cities and states were ranked by average pollution levels, revealing critical hotspots. **Time-series analysis** uncovered seasonal trends, with pollution peaking during winter months, and **pollutant-wise comparisons** highlighted PM10 as the most persistent and hazardous pollutant overall.

One of the major findings from the project was the identification of cities like **Delhi, Mumbai, and Lucknow** as consistently high in PM10 levels, emphasizing their vulnerability to air pollution-related health issues. In contrast, **Northeastern states** such as **Assam** displayed significantly cleaner air quality. These insights were derived through Python's analytical precision, enabling data-driven conclusions that support environmental planning and public health responses.

The analysis was supported by compelling **visualizations** created with **Matplotlib**, **Seaborn**, and **Plotly**. These included interactive line charts, bar graphs, pie charts, and heatmaps — all designed to enhance interpretability and make the data engaging for both technical and non-technical audiences. Python's ability to build **dynamic, filterable, and customizable visuals** added significant value to the project, allowing for deeper exploration and real-time analysis.

Beyond visualization, Python enabled **reproducibility and scalability** — key for expanding this analysis to larger datasets or integrating real-time monitoring systems in the future. Compared to traditional tools, Python offers greater flexibility for automation, integration with APIs, and advanced analytics.

In summary, the project not only achieved its analytical goals but also demonstrated the power of Python as a modern platform for environmental data science. It highlighted how open-source tools can be effectively utilized to address real-world challenges like air pollution, turning raw environmental data into knowledge, insight, and impact. The work lays a strong foundation for future projects involving **predictive modeling**, **geospatial mapping**, and **health-risk correlation**, ultimately contributing to a more sustainable and data-informed approach to environmental management.

6. FUTURE SCOPE

This project effectively demonstrates how Python can be leveraged as a powerful tool to extract, process, analyze, and visualize air quality data using a structured and scalable approach. While this analysis focused primarily on descriptive analytics and static reporting, the flexibility of Python and its ecosystem opens several opportunities to expand this work into real-time monitoring, predictive modeling, and interactive platforms.

The following enhancements outline the broader future potential of this Python-based project:

1. Integration of Real-Time Pollution Monitoring via APIs and IoT

Future developments could include the integration of **real-time air quality data** via open APIs such as:

- **Central Pollution Control Board (CPCB) API**
- **OpenAQ API**
- **BreezoMeter API**
- **IQAir / AirVisual API**

Python libraries like requests, json, and pandas can be used to fetch, parse, and analyze data in real time.

Additionally, Python can be integrated with **IoT devices** and sensors (via MQTT, HTTP, or serial interfaces) to track live air pollution levels and automatically update dashboards or trigger alerts when thresholds are exceeded.

2. Advanced Visualization Using Interactive Dashboards

While this project used static plots with **Matplotlib** and **Seaborn**, future enhancements could involve **fully interactive dashboards** built using:

- **Plotly Dash**
- **Streamlit**
- **Bokeh**

These platforms allow real-time data exploration with filters, sliders, maps, and live updates — offering a user-friendly interface for environmental researchers, students, or municipal bodies.

3. Predictive Modeling with Machine Learning

Using Python's machine learning libraries like **scikit-learn**, **XGBoost**, or **TensorFlow**, the project can transition from descriptive to **predictive analytics**:

- **Time Series Forecasting (ARIMA, LSTM, Prophet)** to forecast future pollutant levels.
- **Classification Models** to predict "high-risk" pollution days.
- **Anomaly Detection** using unsupervised algorithms to detect outlier readings or environmental irregularities.

These capabilities can support early warning systems and proactive policy decisions for cities experiencing seasonal pollution spikes.

4. Geospatial and Satellite-Based Pollution Mapping

Python offers powerful geospatial tools such as:

- **GeoPandas**
- **Folium**
- **Kepler.gl**
- **Rasterio / Earth Engine Python API**

These tools can be used to map pollution data on interactive city maps or satellite overlays. Visualizations can highlight:

- Urban vs. rural pollution distribution
- Pollution sources near roads, industrial zones, or low-vegetation areas
- Hotspot tracking and spatial clustering of pollutants

5. Public Health Impact Correlation and Risk Assessment

By combining air quality data with **health records** or **hospital data** (e.g., via public health APIs or CSV records), Python can be used to:

- Correlate pollution levels with respiratory disease incidence
- Predict risk zones for asthma, bronchitis, or cardiac issues
- Conduct spatial health risk assessments

Libraries like statsmodels and scipy would enable regression and statistical testing, while tools like Altair or Plotly could visualize disease exposure correlations.

6. Expanding Dataset Scope, Depth, and Diversity

To strengthen the analytical power of the project, future versions could incorporate:

- **Historical data** spanning multiple years (for trend analysis)
- **Additional pollutants** (e.g., CO, NO_2, PM2.5, O_3)
- **Meteorological variables** (temperature, humidity, wind speed) via weather APIs
- **Demographic overlays** (population, age, income) for impact segmentation

This would enable **multivariate environmental modeling** and deeper causal insights.

Conclusion of Scope

By integrating real-time data streams, predictive analytics, advanced mapping tools, and public health statistics, this project can evolve into a robust **air quality intelligence platform**. Powered by Python, it could serve:

- Urban planners in identifying and mitigating pollution hotspots
- Public health agencies in monitoring environmental risk
- Researchers and policymakers seeking data-driven insight
- Citizens interested in understanding and protecting their local air quality

Ultimately, Python’s flexibility, scalability, and ecosystem make it an ideal foundation for the next generation of environmental monitoring and decision-making tools.

Linkedin post link:-

[redacted]

7.REFERENCES

[1] National Archives, "Air Pollution", *Data.gov*, [Online]. Available: [2] M. Friendly, “The History of the Modern Graph,” *Statistical Science*, vol. 14, no. 1, pp. 1–14, 1999.[3] T. Alwafi, "A Study on Air pollution ," *IEEE International Conference on Data Science and Analytics* ,:- pp. 234-238, 2020.[4] Microsoft Corporation, “Create a PivotTable to analyze worksheet data,” *Microsoft Support*, [Online]. Available: https://support.microsoft.com/en-us/excel[5] K. Kaushik, "Web Analytics 2.0: The Art of Online Accountability and Science of Customer Centricity", *Sybex*, 2009.

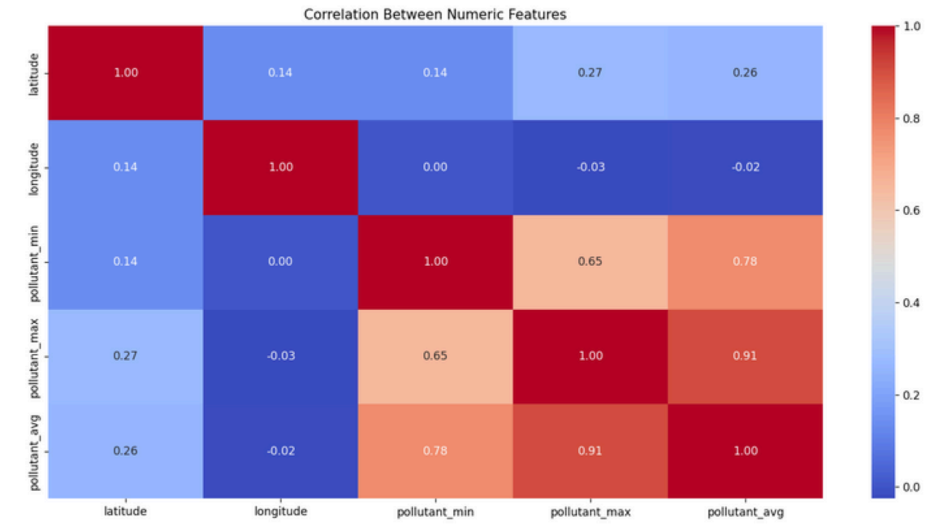
[6] A. Gandomi and M. Haider, “Beyond the Hype: Big Data Concepts, Methods, and Analytics,” *International Journal of Information Management*, vol. 35, no. 2, pp. 137–144, Apr. 2015.

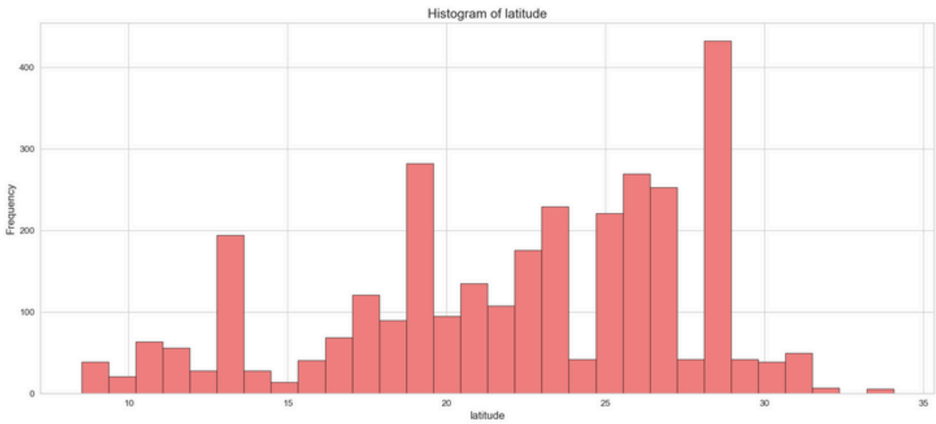
[7] N. Stieglitz, S. Mirbabaie, B. Ross, and M. Neuberger, “Social Media Analytics – Challenges in Topic Discovery, Data Collection, and Data Preparation,” *International Journal of Information Management*, vol. 39, pp. 156–168, Dec. 2018.

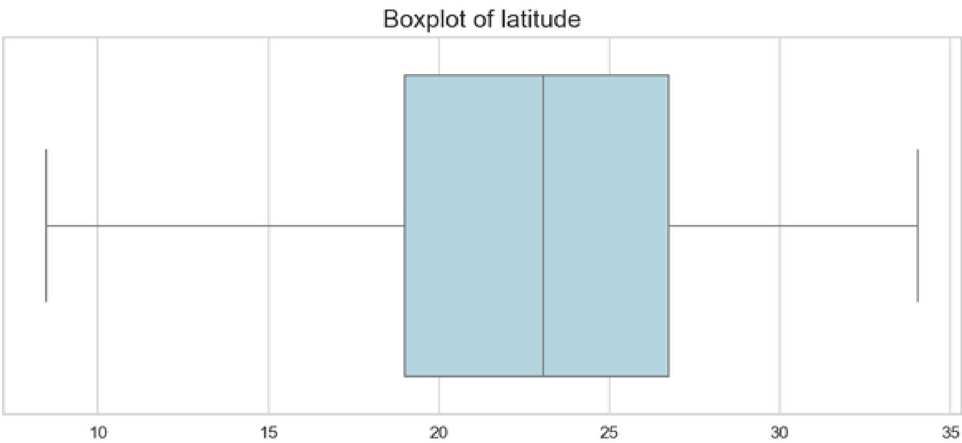
```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:\Users\Shiva\Downloads\python project\Pyproject.py ========
  country  state      city  ... pollutant_min pollutant_max  pollutant_avg
0   India  Assam  Guwahati  ...           5.0           5.0            5.0
1   India  Assam  Guwahati  ...          19.0          21.0           19.0
2   India  Assam  Guwahati  ...          15.0          49.0           18.0
3   India  Assam  Guwahati  ...           7.0          14.0           12.0
4   India  Assam  Guwahati  ...          47.0         124.0           89.0

[5 rows x 11 columns]
     country        state     city  ... pollutant_min pollutant_max  pollutant_avg
3188   India  West_Bengal  Kolkata  ...          15.0         115.0           36.0
3189   India  West_Bengal  Kolkata  ...           9.0          13.0           11.0
3190   India  West_Bengal  Kolkata  ...          21.0         165.0           43.0
3191   India  West_Bengal  Kolkata  ...          46.0         134.0           82.0
3192   India  West_Bengal  Kolkata  ...          26.0          36.0           28.0

[5 rows x 11 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3193 entries, 0 to 3192
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   country        3193 non-null   object
 1   state          3193 non-null   object
 2   city           3193 non-null   object
 3   station        3193 non-null   object
 4   last_update    3193 non-null   object
 5   latitude       3193 non-null   float64
 6   longitude      3193 non-null   float64
 7   pollutant_id   3193 non-null   object
 8   pollutant_min  3052 non-null   float64
 9   pollutant_max  3052 non-null   float64
 10  pollutant_avg  3052 non-null   float64
dtypes: float64(5), object(6)
memory usage: 274.5+ KB
None
```

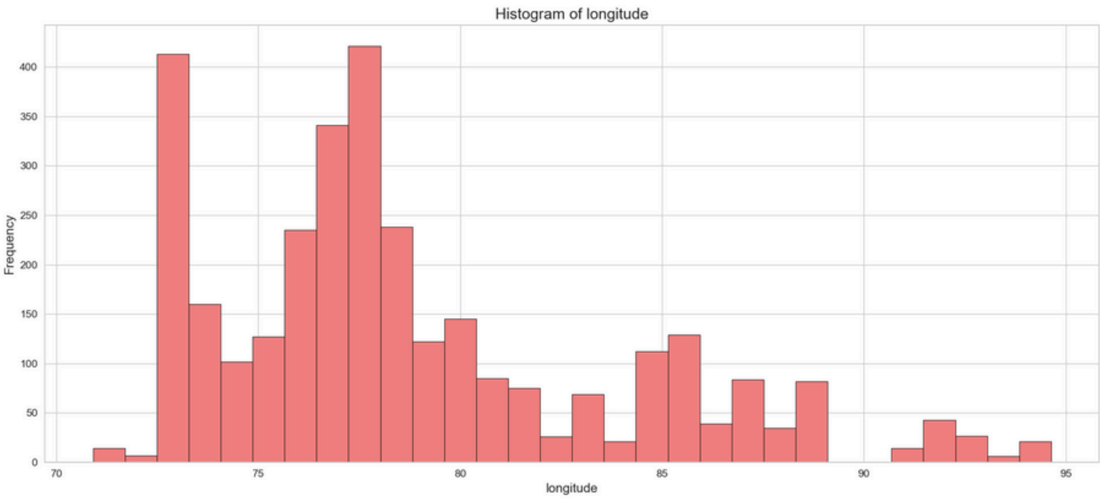
```
count  3193.000000  3193.000000  3052.000000  3052.000000  3052.000000
mean     22.240313    78.829966    23.428571    87.670380    48.316514
std       5.547880     4.996175    25.773383    92.512811    49.740277
min       8.514909    70.909168     1.000000     1.000000     1.000000
25%      19.000083    75.565602     5.000000    20.000000    13.000000
50%      23.076793    77.508730    14.000000    60.000000    32.000000
75%      26.766433    80.948222    33.000000   116.000000    66.000000
max      34.066206    94.636574   240.000000   500.000000   399.000000
country            0
state              0
city               0
station            0
last_update        0
latitude           0
longitude          0
pollutant_id       0
pollutant_min    141
pollutant_max    141
pollutant_avg    141
dtype: int64
country          0
state            0
city             0
station          0
last_update      0
latitude         0
longitude        0
pollutant_id     0
pollutant_min    0
pollutant_max    0
pollutant_avg    0
dtype: int64
Columns: Index(['country', 'state', 'city', 'station', 'last_update', 'latitude',
       'longitude', 'pollutant_id', 'pollutant_min', 'pollutant_max',
       'pollutant_avg'],
      dtype='object')
country: 1 unique values
state: 32 unique values
city: 254 unique values
```

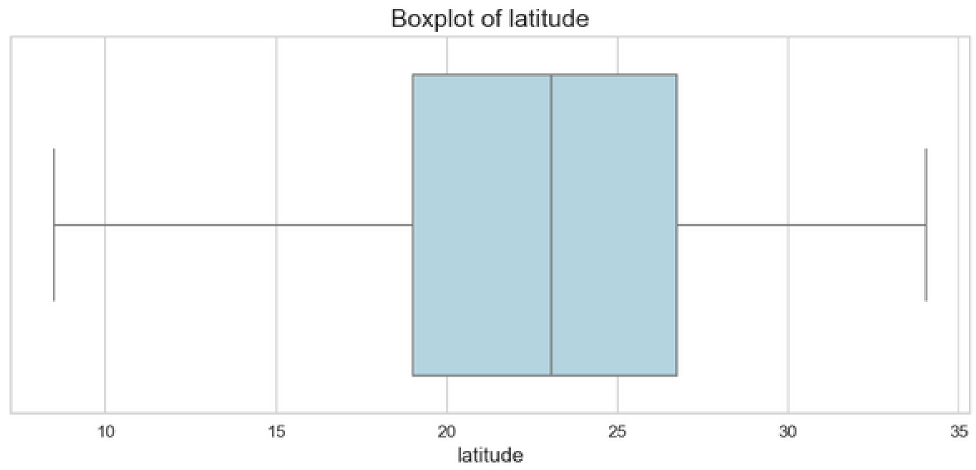
```
state              0
city               0
station            0
last_update        0
latitude           0
longitude          0
pollutant_id       0
pollutant_min    141
pollutant_max    141
pollutant_avg    141
dtype: int64
country          0
state            0
city             0
station          0
last_update      0
latitude         0
longitude        0
pollutant_id     0
pollutant_min    0
pollutant_max    0
pollutant_avg    0
dtype: int64
Columns: Index(['country', 'state', 'city', 'station', 'last_update', 'latitude',
       'longitude', 'pollutant_id', 'pollutant_min', 'pollutant_max',
       'pollutant_avg'],
      dtype='object')
country: 1 unique values
state: 32 unique values
city: 254 unique values
station: 481 unique values
last_update: 1 unique values
latitude: 481 unique values
longitude: 481 unique values
pollutant_id: 7 unique values
pollutant_min: 132 unique values
pollutant_max: 379 unique values
pollutant_avg: 232 unique values
```

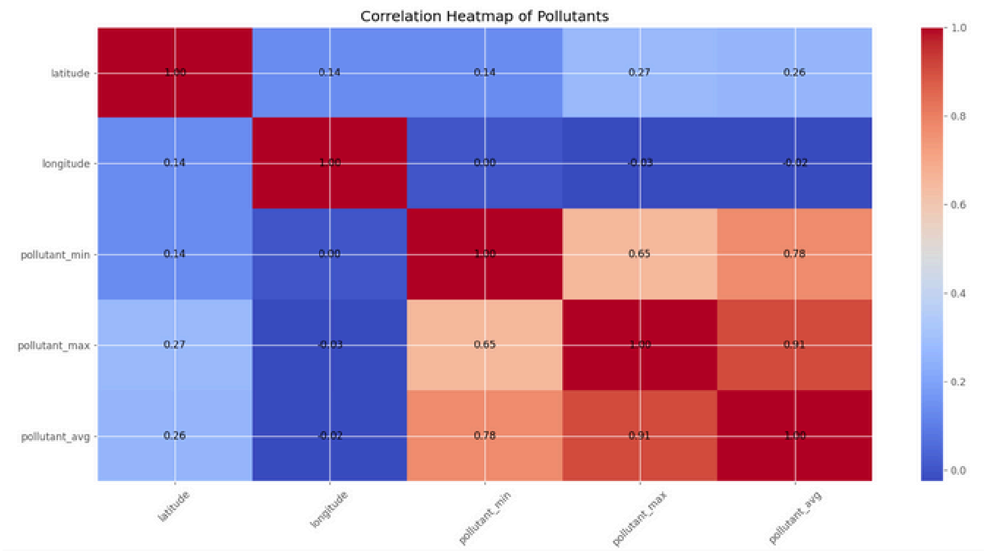

Correlation Between Numeric Features
latitude
longitude
pollutant_min
pollutant_max
pollutant_avg
1.00 0.14 0.14 0.27 0.26
0.14 1.00 0.00 -0.03 -0.02
0.14 0.00 1.00 0.65 0.78
0.27 -0.03 0.65 1.00 0.91
0.26 -0.02 0.78 0.91 1.00
1.0
0.8
0.6
0.4
0.2
0.0

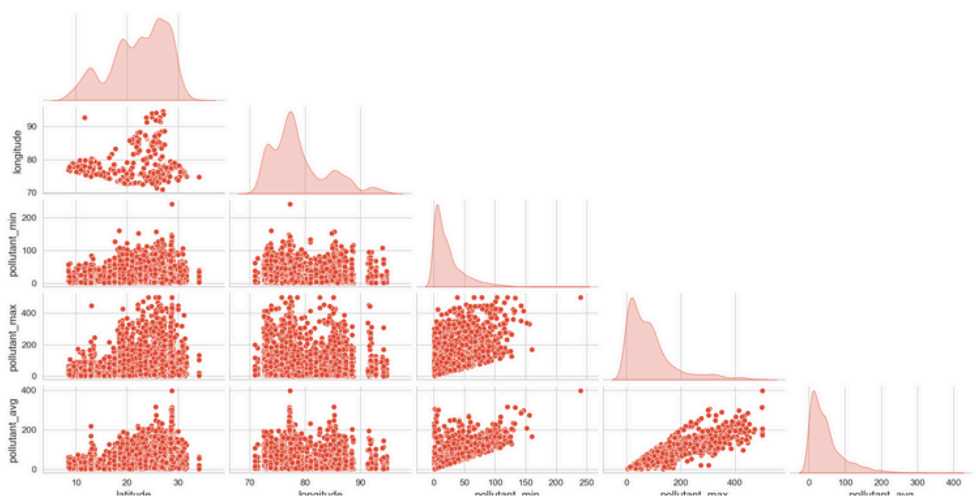

Histogram of latitude
Frequency
400
300
200
100
0
10
15
20
25
30
35
latitude

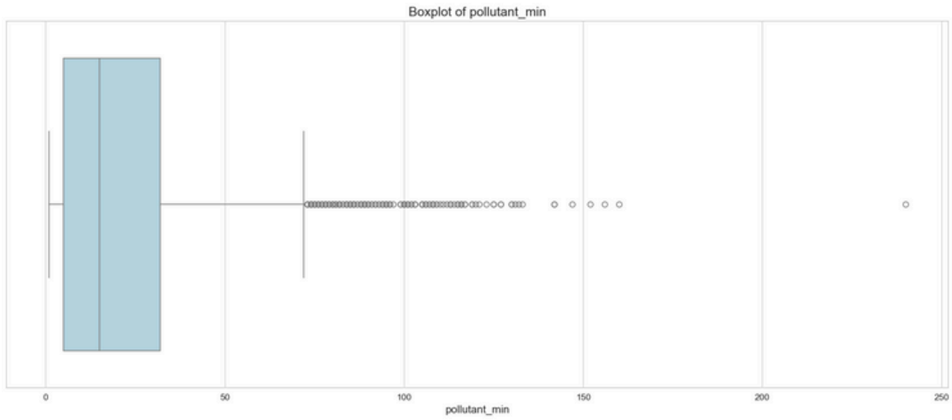

Boxplot of latitude
10
15
20
25
30
35

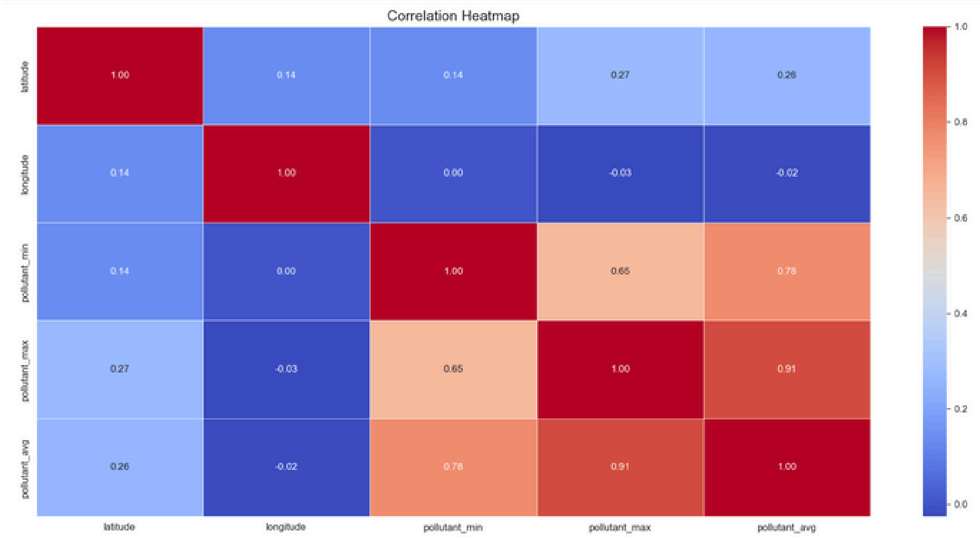

Histogram of longitude
Frequency
400
350
300
250
200
150
100
50
0
70
75
80
85
90
95
longitude

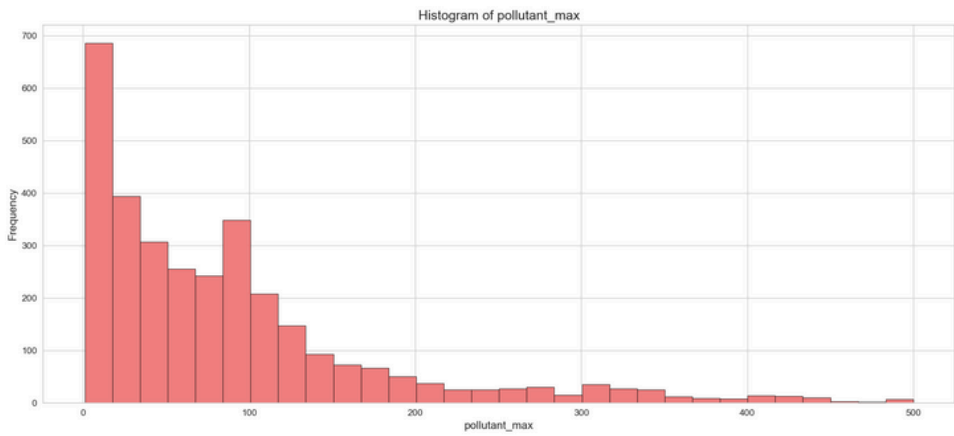

Boxplot of latitude
10
15
20
25
30
35
latitude

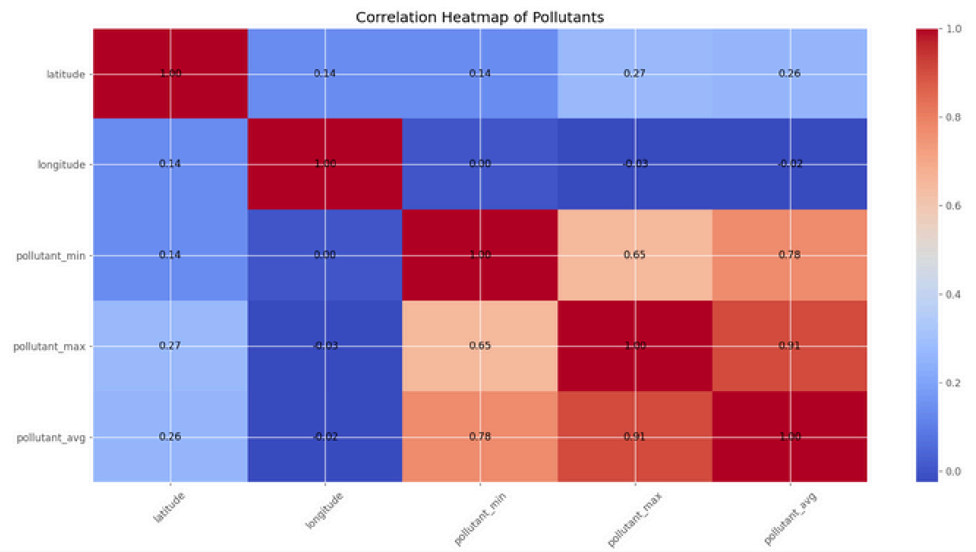

Correlation Heatmap of Pollutants
latitude
longitude
pollutant_min
pollutant_max
pollutant_avg
1.00
0.14
0.14
0.27
0.26
0.14
1.00
0.00
0.03
0.02
0.14
0.00
1.00
0.65
0.78
0.27
0.03
0.65
1.00
0.91
0.26
0.02
0.78
0.91
1.00
1.0
0.8
0.6
0.4
0.2
0.0


longitude
pollutant_min
pollutant_max
pollutant_avg


Boxplot of pollutant_min
0
50
100
150
200
250
pollutant_min


Correlation Heatmap
latitude
longitude
pollutant_min
pollutant_max
pollutant_avg
1.00
0.14
0.14
0.27
0.26
0.14
1.00
0.00
-0.03
-0.02
0.14
0.00
1.00
0.65
0.78
0.27
-0.03
0.65
1.00
0.91
0.26
-0.02
0.78
0.91
1.00
latitude
longitude
pollutant_min
pollutant_max
pollutant_avg
1.0
0.8
0.6
0.4
0.2
0.0


Histogram of pollutant_max
700
600
500
400
300
200
100
0
Frequency
0
100
200
300
400
500
pollutant_max


Correlation Heatmap of Pollutants
latitude
longitude
pollutant_min
pollutant_max
pollutant_avg
1.00
0.14
0.14
0.27
0.26
0.14
1.00
0.00
-0.03
-0.02
0.14
0.00
1.00
0.65
0.78
0.27
-0.03
0.65
1.00
0.91
0.26
-0.02
0.78
0.91
1.00
latitude
longitude
pollutant_min
pollutant_max
pollutant_avg
1.0
0.8
0.6
0.4
0.2
0.0