****

**Data Science Tool Box Python Programming**

**PROJECT REPORT**

***(Pollution\_Data\_Analysis)***

**Submitted by: MUPPIDI MANASWI**

**Registration No. 12311427**

**Programme and Section. Data science, K23GN**

**Course Code. CSE375**

**Under the Guidance of: (Aashima)**

**Discipline of CSE/IT**

**Lovely School of Computer Science**

**Lovely Professional University, Phagwara**

**CERTIFICATE**

This is to certify that MUPPIDI MANASWI bearing Registration no. 12311427 has completed CSE-375 project titled, “Project Semester January-April 2025” 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.

**DECLARATION**

I MUPPIDI MANASWI student of Computer Science 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: Signature:**

**Registration No. 12311427 Muppidi Manaswi**

|  |  |
| --- | --- |
| **INDEX** | **Pg.no:** |
| **Introduction** | **3** |
| **Source of code** | **4** |
| **EDA Process** | **4** |
| **Analysis on Dataset** | **4** |
| **Objective 1: Analyse Pollution Trends Over Time** | **4** |
| **Objective 2: Evaluate Impact of Infrastructure (By Station)** | **5** |
| **Objective 3: Compare Performance Across States** | **6** |
| **Objective 4: Key Predictors - Correlation Analysis** | **8** |
| **Objective 5: Track Disparities by Pollutant Type** | **9** |
| **Conclusion** | **12** |
| **Future Scope** | **12** |
| **References** | **12** |
| **Source Code** | **12** |

**1. Introduction**

Exploratory Data Analysis (EDA) is an essential step in the data analysis pipeline. It involves summarizing the main characteristics of a dataset, often using visual methods. In this project, we focus on analysing pollution data gathered across various stations in India to understand pollution trends, patterns, and anomalies.

Why EDA is Important: EDA helps in:

* Understanding the structure of the dataset.
* Identifying missing or erroneous data.
* Spotting patterns, correlations, and trends.
* Detecting outliers and anomalies.
* Guiding feature selection for predictive modelling.

Objective of the Project: To perform detailed EDA on a pollution dataset and derive meaningful insights that can support environmental decision-making and public health policies.

**2. Source of Dataset**

The dataset used in this project is collected from an official governmental or environmental monitoring source, which records pollutant levels across various states and monitoring stations in India.

File Name: Dataset.csv Location: C:\Users\MANASWI\OneDrive\Desktop\Dataset.csv

**3. EDA Process**

**Step 1: Importing Libraries and Dataset**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from scipy.stats import zscore

Step 2: Reading and Inspecting the Data

file\_path = ("C:\\Users\\MANASWI\\OneDrive\\Desktop\\Dataset.csv")

df = pd.read\_csv(file\_path, encoding='utf-8')

print(df.info())

**Step 3: Handling Missing Data**

* Replaced blank spaces with NaN.
* Dropped rows where critical pollutant values were missing.
* Converted values to numeric format.

**4. Analysis on Dataset**

**Objective 1: Analyze Pollution Trends Over Time**

**i. Introduction: To monitor how pollution levels have evolved over time.**

**ii. General Description:**

* Columns used: last\_update, pollutant\_avg
* Line plot to represent trends.

**iii. Functions and Formulas:**

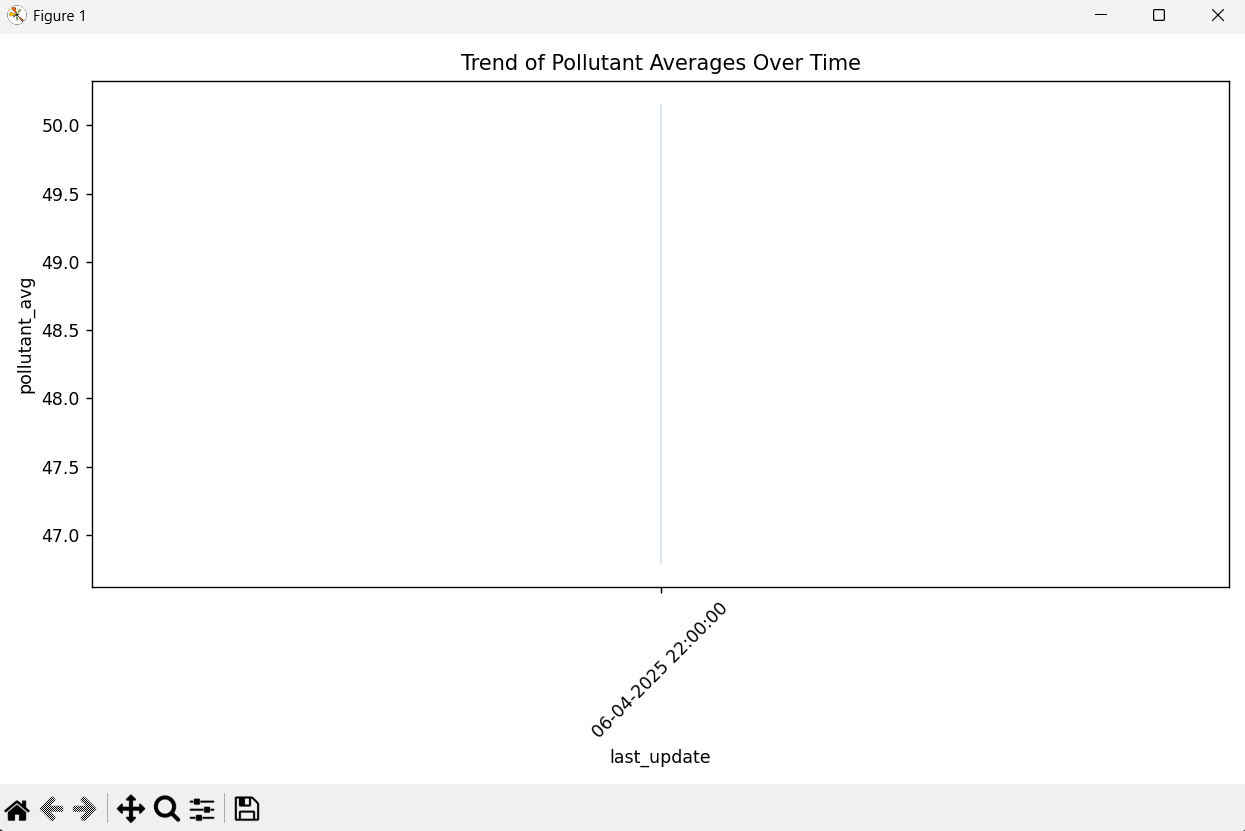
* sns.lineplot() from Seaborn
* plt. xticks(rotation=45) to improve readability

**iv. Analysis Results:**

* Observed fluctuations in average pollutant levels.
* Seasonal variations detected.

**v. Visualization:**

* Use a line plot to show pollutant\_avg over time. Graph 1: Trend of Pollutant Averages Over Time.

****

**Objective 2: Evaluate Impact of Infrastructure (By Station)**

**i. Introduction: To understand how pollution varies by monitoring station.**

**ii. General Description:**

* Station-wise pollutant comparison using boxplot.

**iii. Functions and Formulas:**

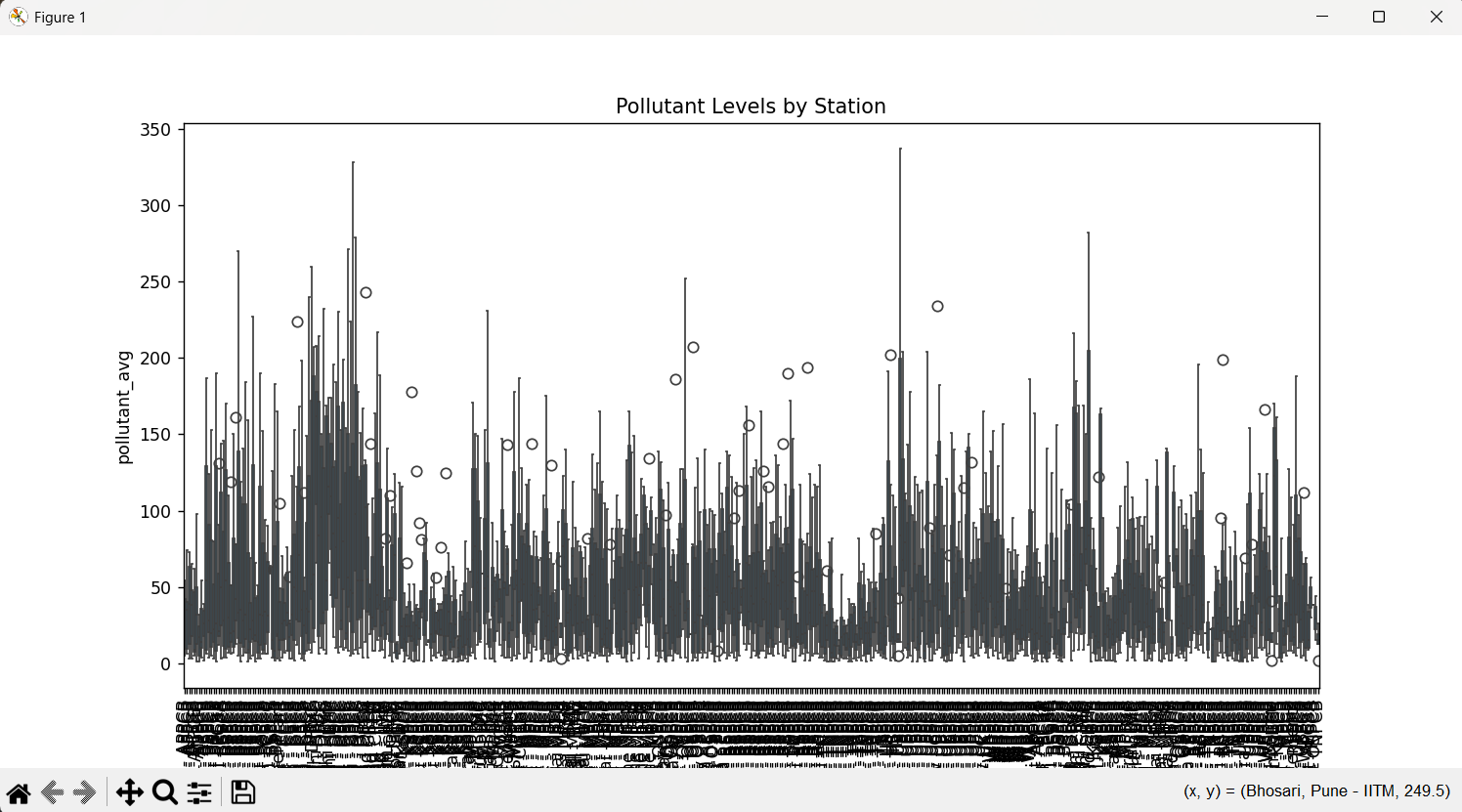
* sns.boxplot()

**iv. Analysis Results:**

* Some stations report significantly higher average pollutants.

**v. Visualization:**

* Use a boxplot to compare pollutant\_avg for each station. Graph 2: Pollutant Levels by Station.

****

**Objective 3: Compare Performance Across States**

**i. Introduction: Compare pollution levels across Indian states.**

**ii. General Description:**

* Grouped by state.
* Bar plot used to show averages.

**iii. Functions and Formulas:**

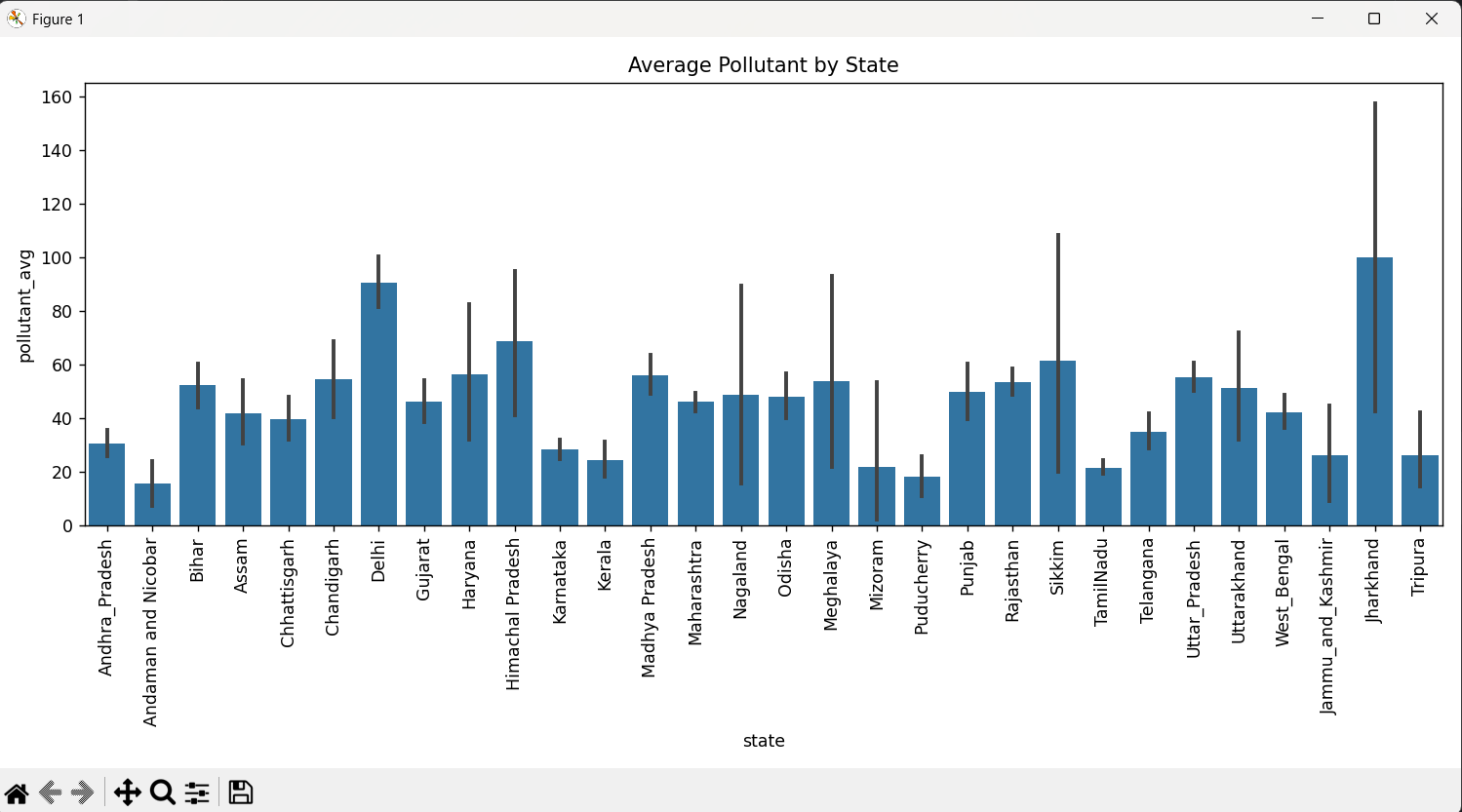
* sns.barplot() with estimator=np.mean

**iv. Analysis Results:**

* States like Delhi, Uttar Pradesh show higher averages.

**v. Visualization:**

* Use a barplot to visualize average pollution across states. Graph 3: Average Pollutant by State.

****

**Objective 4: Key Predictors - Correlation Analysis**

i. Introduction: Identify relationships between pollutant metrics.

**ii. General Description:**

* Used correlation matrix and heatmap.

**iii. Functions and Formulas:**

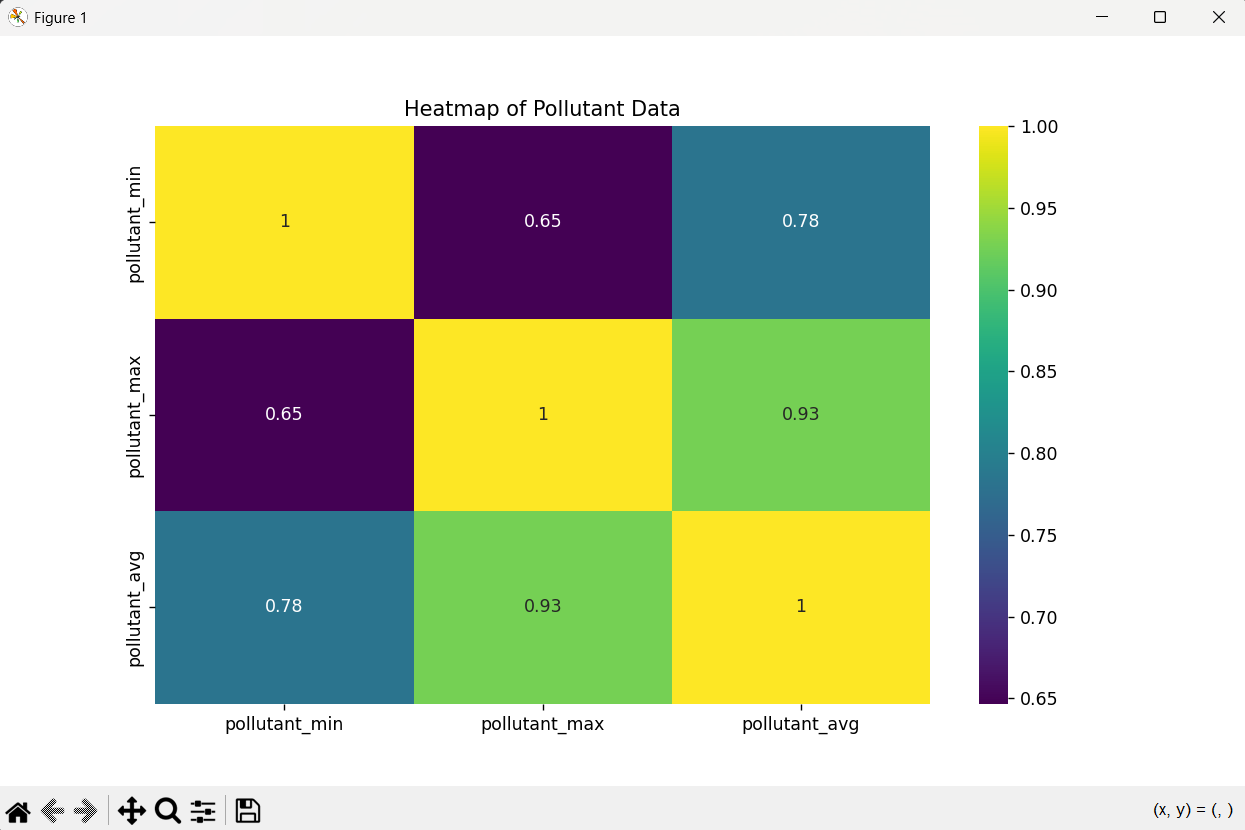
* df[['pollutant\_min', 'pollutant\_max', 'pollutant\_avg']].corr()
* sns.heatmap()

**iv. Analysis Results:**

* Strong correlation between max and avg values.

**v. Visualization:**

* Use a heatmap to show correlation values. Graph 4: Correlation Heatmap.

****

**Objective 5: Track Disparities by Pollutant Type**

i. Introduction: Study how different pollutants behave.

**ii. General Description:**

* Grouped by pollutant ID.
* Boxplot used to compare distributions.

**iii. Functions and Formulas:**

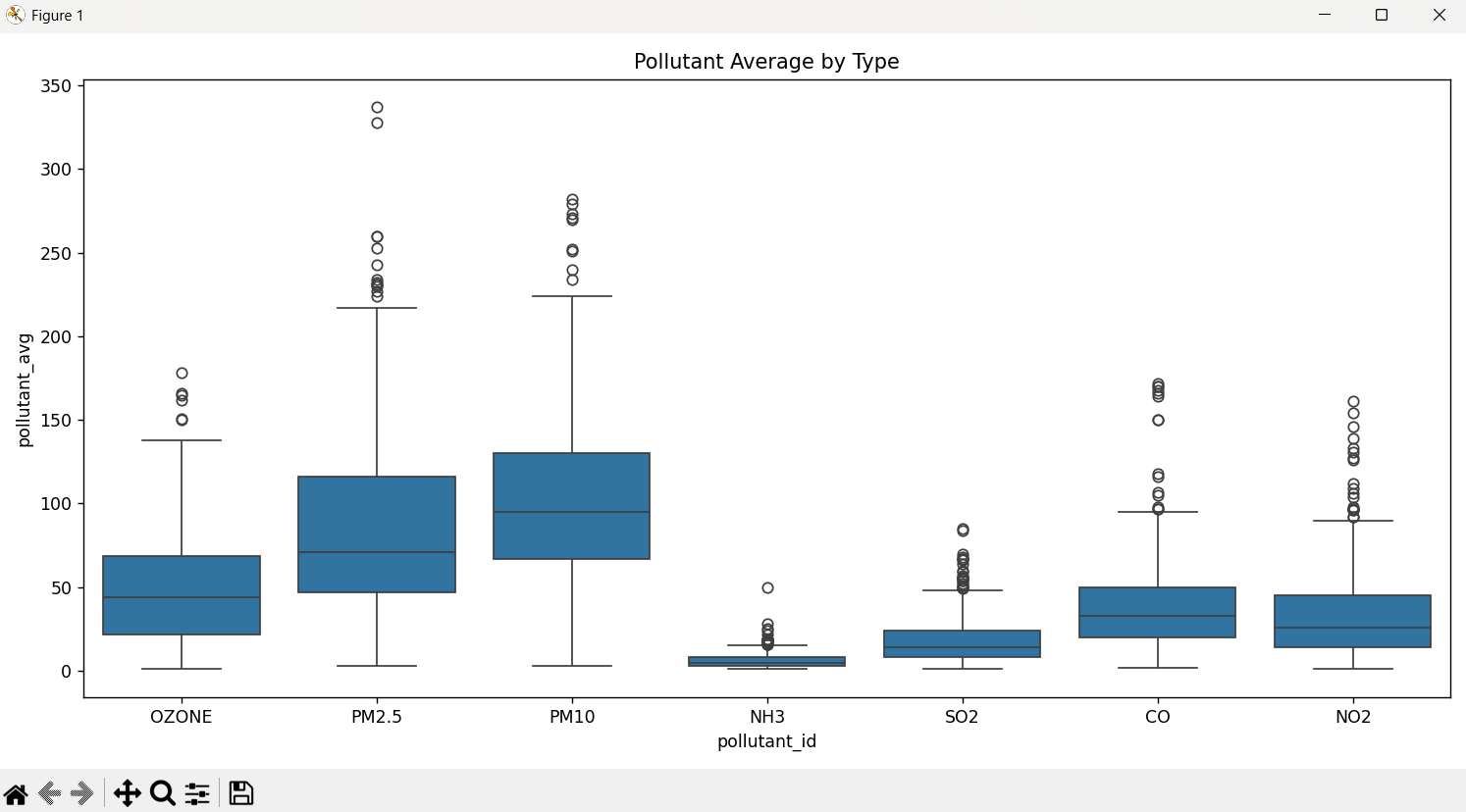
* sns.boxplot()

**iv. Analysis Results:**

* Certain pollutants show higher variance.

**v. Visualization:**

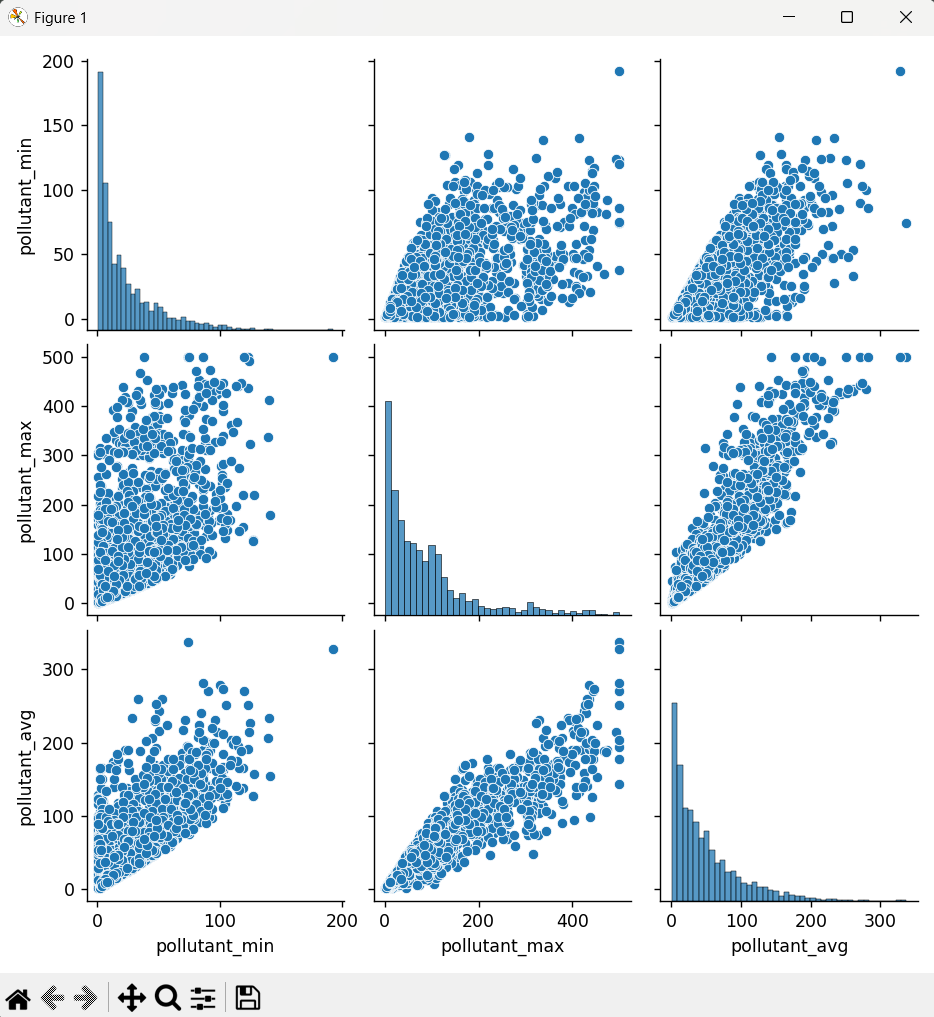
* Use a boxplot to display pollutant\_avg for each pollutant\_id. Graph 5: Pollutant Average by Type.

****

**Pairwise Relationship**

**Purpose: Understand relationships among pollutant metrics.**

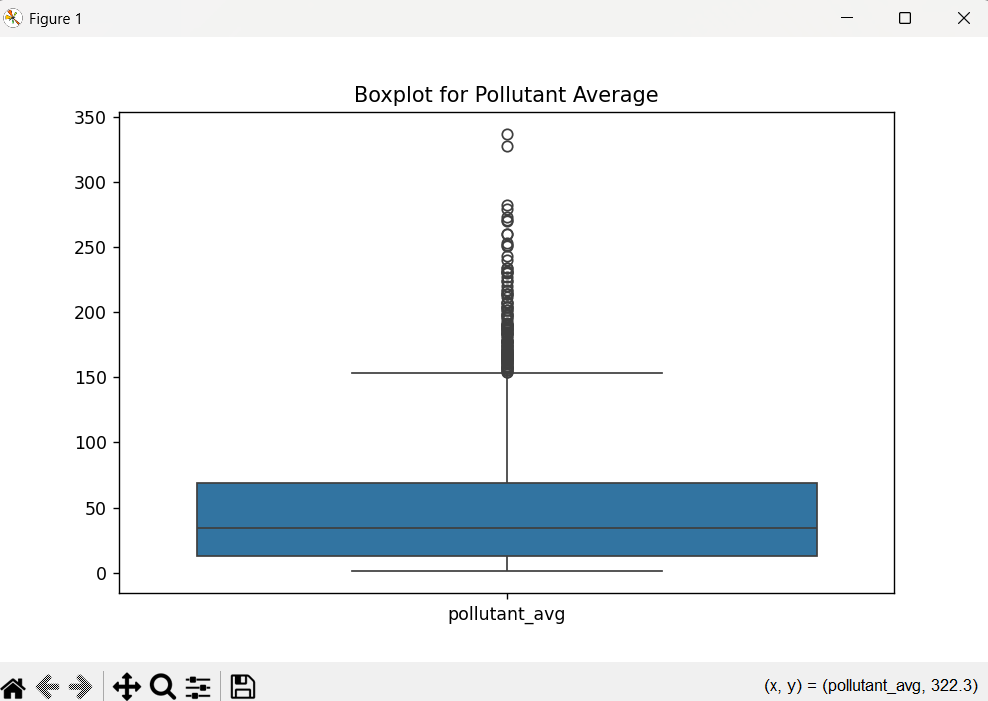
* Use a pairplot to visualize all variable relationships. Graph 6: Pairplot of Pollutant Metrics

****

**Outlier Detection**

**i. Boxplot Analysis**

* Use a boxplot to detect visual outliers. Graph 7: Boxplot for Pollutant Average.

****

**ii. Z-Score Method**

z\_scores = zscore(df[['pollutant\_avg']])

outliers = df[(np.abs(z\_scores) > 3).any(axis=1)]

print(f"Total Records: {df.shape[0]}, Outliers Detected: {outliers.shape[0]}")

* Results: Number of detected outliers printed.

**5. Conclusion**

This project successfully implemented EDA on a real-world pollution dataset. We observed temporal trends, geographical disparities, and outlier values, offering a foundational understanding of pollution patterns in India.

**6. Future Scope**

* Predictive Modeling: Use the insights for forecasting pollution levels.
* Real-time Analysis: Integrate with APIs for live data.
* Policy Planning: Help authorities target high-pollution areas.
* Health Impact Studies: Correlate pollution data with health reports.

**7. References**

* Seaborn Documentation: https://seaborn.pydata.org/
* Matplotlib Documentation: https://matplotlib.org/
* Pandas Documentation: https://pandas.pydata.org/
* Scipy Documentation: https://docs.scipy.org/
* Pollution Dataset Source: [Add actual source or URL here]

**Source Code:**

import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
from scipy.stats import zscore  
  
# Step 1: Import the dataset (with encoding fix)  
file\_path = ("C:\\Users\\MANASWI\\OneDrive\\Desktop\\Dataset.csv")  
df = pd.read\_csv(file\_path, encoding='utf-8')  
  
# Step 2: Get info  
print(df.info())  
  
# Step 3: Handle missing data  
df.replace(" ", np.nan, inplace=True)  
df.dropna(subset=['pollutant\_min', 'pollutant\_max', 'pollutant\_avg'], inplace=True)  
  
# Convert to numeric just in case  
df['pollutant\_min'] = pd.to\_numeric(df['pollutant\_min'], errors='coerce')  
df['pollutant\_max'] = pd.to\_numeric(df['pollutant\_max'], errors='coerce')  
df['pollutant\_avg'] = pd.to\_numeric(df['pollutant\_avg'], errors='coerce')  
df.dropna(inplace=True)  
  
# Step 4: Objectives  
  
## Objective 1: Analyze Student Performance Trends (pollutant\_avg over time)  
plt.figure(figsize=(10,6))  
sns.lineplot(data=df, x='last\_update', y='pollutant\_avg')  
plt.xticks(rotation=45)  
plt.title("Trend of Pollutant Averages Over Time")  
plt.tight\_layout()  
plt.show()  
  
## Objective 2: Evaluate Impact of Infrastructure (by Station)  
plt.figure(figsize=(12,6))  
sns.boxplot(data=df, x='station', y='pollutant\_avg')  
plt.xticks(rotation=90)  
plt.title("Pollutant Levels by Station")  
plt.tight\_layout()  
plt.show()  
  
## Objective 3: Compare Performance Across States  
plt.figure(figsize=(12,6))  
sns.barplot(data=df, x='state', y='pollutant\_avg', estimator=np.mean)  
plt.xticks(rotation=90)  
plt.title("Average Pollutant by State")  
plt.tight\_layout()  
plt.show()  
  
## Objective 4: Key Predictors (correlation heatmap)  
plt.figure(figsize=(6,4))  
sns.heatmap(df[['pollutant\_min', 'pollutant\_max', 'pollutant\_avg']].corr(), annot=True, cmap='coolwarm')  
plt.title("Correlation Between Pollutant Metrics")  
plt.show()  
  
## Objective 5: Track Type Disparities (by pollutant type)  
plt.figure(figsize=(12,6))  
sns.boxplot(data=df, x='pollutant\_id', y='pollutant\_avg')  
plt.title("Pollutant Average by Type")  
plt.tight\_layout()  
plt.show()  
  
# Step 5: Relationship between variables - Pairplot  
sns.pairplot(df[['pollutant\_min', 'pollutant\_max', 'pollutant\_avg']])  
plt.show()  
  
# Step 6: Heatmap  
plt.figure(figsize=(10,6))  
sns.heatmap(df[['pollutant\_min', 'pollutant\_max', 'pollutant\_avg']].corr(), annot=True, cmap='viridis')  
plt.title("Heatmap of Pollutant Data")  
plt.show()  
  
# Step 7: Outliers  
  
## Boxplot  
plt.figure(figsize=(8,5))  
sns.boxplot(data=df[['pollutant\_avg']])  
plt.title("Boxplot for Pollutant Average")  
plt.show()  
  
## Z-score  
z\_scores = zscore(df[['pollutant\_avg']])  
outliers = df[(np.abs(z\_scores) > 3).any(axis=1)]  
print(f"Total Records: {df.shape[0]}, Outliers Detected: {outliers.shape[0]}")