Phase 3

Air Quality Analysis

Phase 3: Development Part 1

In this part we will:

- begin building your project by loading and preprocessing the dataset.
- Begin the analysis by loading and preprocessing the air quality dataset.
- Load the dataset using Python and data manipulation libraries (e.g., pandas)

Step 1: Download the Dataset:

- Access the provided link for Air Quality Analysis dataset.

Dataset Link: https://tn.data.gov.in/resource/location-wise-daily-ambient-air-quality-tamil-nadu-year-2014

- Download the dataset to local working directory or preferred location.

Step 2: Loading the Dataset:

Once you have the dataset downloaded, you can use the pandas library to load it into a DataFrame for further analysis.

```
[1] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[2] import pandas as pd
    df= pd.read_csv('/content/drive/MyDrive/cpcb_dly_aq_tamil_nadu-2014.csv')
```

```
[3] df.head()
```

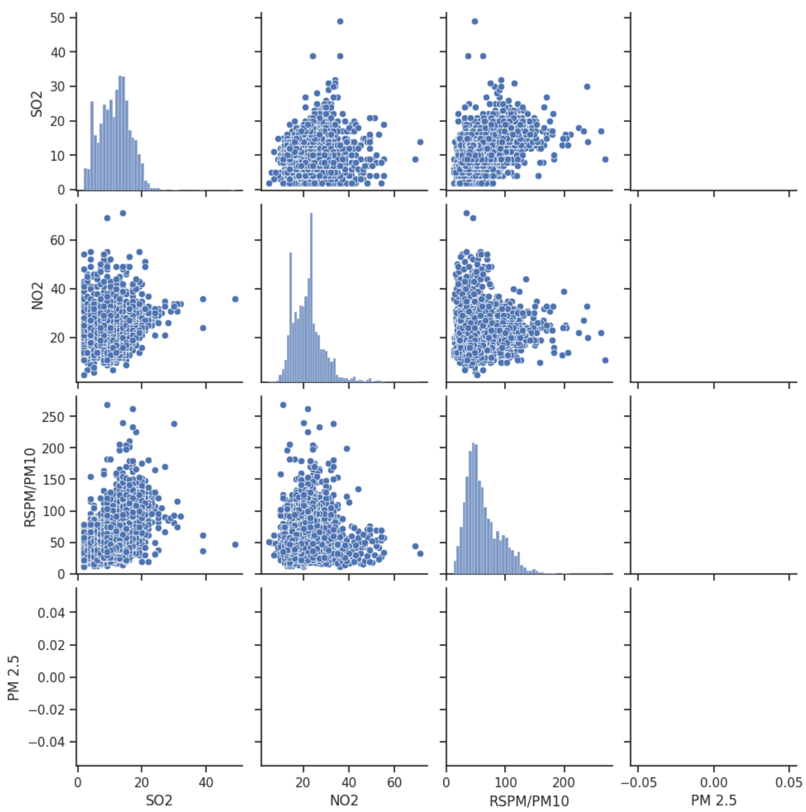
	Stn Code	Sampling Date	State	City/Town/Village/Area	Location of Monitoring Station	Agency	Type of Location	SO2	NO2	RSPM/PM10	PM 2.5
0	38	01-02-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	11.0	17.0	55.0	NaN
1	38	01-07-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	13.0	17.0	45.0	NaN
2	38	21-01-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	12.0	18.0	50.0	NaN
3	38	23-01-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	15.0	16.0	46.0	NaN
4	38	28-01-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	13.0	14.0	42.0	NaN

Step 3: Exploratory Data Analysis (EDA):

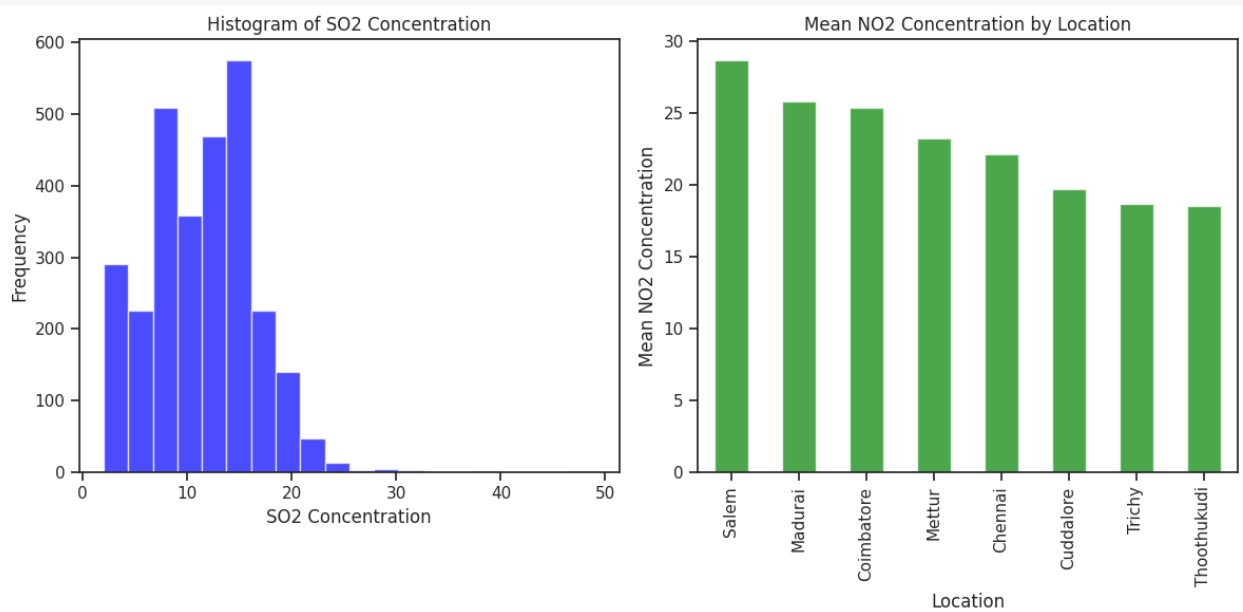
EDA is a crucial step in understanding any dataset. For our "Air Quality Analysis" project, you can perform the following EDA tasks:

- Compute summary statistics to understand the distribution of air quality parameters.
- Create histograms, box plots, and scatter plots to visualize the distribution and relationships between variables.
- Check for missing data and decide on an appropriate strategy to handle it.
- Identify trends and patterns in air quality over time, and across locations within the dataset.

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="ticks")
sns.pairplot(df[['SO2', 'NO2', 'RSPM/PM10', 'PM 2.5']])
plt.show()
```



```
import matplotlib.pyplot as plt

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

# Create a histogram for 'SO2'
plt.subplot(1, 2, 1)
plt.hist(df['SO2'], bins=20, color='blue', alpha=0.7)
plt.xlabel('SO2 Concentration')
plt.ylabel('Frequency')
plt.title('Histogram of SO2 Concentration')

# Create a bar chart for 'NO2'
plt.subplot(1, 2, 2)
locations = df['City/Town/Village/Area']  # Assuming this column contains location names
mean_no2 = df['NO2'].groupby(locations).mean()  # Calculate the mean NO2 by location
mean_no2.sort_values(ascending=False).plot(kind='bar', color='green', alpha=0.7)
plt.xlabel('Location')
plt.ylabel('Mean NO2 Concentration')
plt.title('Mean NO2 Concentration by Location')

plt.tight_layout()
plt.show()
```



Step 4: Data Cleaning and Preprocessing

This step involves preparing the data for analysis. Tasks may include:

- Handling missing data by dropping, filling, or imputing values.
- Dealing with outliers if necessary.
- Formatting dates and times for time series analysis.
- Ensuring consistent data types.
- Handling any data quality issues identified during EDA.

```
import pandas as pd
import numpy as np


# Handling missing values by replacing 'NA' with NaN
df.replace('NA', np.nan, inplace=True)

# Check for missing values
missing_values = df.isnull().sum()
print("Missing Values:\n", missing_values)

# Address outliers (you may need to define outlier criteria)
outlier_criteria = (df['SO2'] > 50) | (df['NO2'] > 40)

# Identify and handle outliers (e.g., you can replace them with
NaN)
df[outlier_criteria] = np.nan

# Check for outliers
outliers = df.isnull().sum()
print("Outliers:\n", outliers)
```

```
Missing Values:
 Stn Code                           0
Sampling Date                      0
State                              0
City/Town/Village/Area             0
Location of Monitoring Station     0
Agency                             0
Type of Location                   0
SO2                                0
NO2                               13
RSPM/PM10                          4
PM 2.5                          2807
dtype: int64
Outliers:
 Stn Code                           0
Sampling Date                      0
State                              0
City/Town/Village/Area             0
Location of Monitoring Station     0
Agency                             0
Type of Location                   0
SO2                                0
NO2                               13
RSPM/PM10                          4
PM 2.5                          2807
dtype: int64
```

Step 5: Preprocessing

In this step, you can perform any additional preprocessing specific to your analysis objectives. For example, you may aggregate data to a daily or monthly level for trend analysis, or calculate averages across different monitoring stations.

```
import pandas as pd

# Convert 'Sampling Date' to datetime type
df['Sampling Date'] = pd.to_datetime(df['Sampling Date'])

# Create new columns for Year and Month
df['Year'] = df['Sampling Date'].dt.year
df['Month'] = df['Sampling Date'].dt.month

# Calculate monthly averages
monthly_averages = df.groupby(['Year',
'Month']).mean(numeric_only=True).reset_index()
```

```python
# Calculate yearly summaries
yearly_summaries                                                        =
df.groupby('Year').mean(numeric_only=True).reset_index()

# Calculate location-based aggregations
location_aggregations                                                   =
df.groupby('City/Town/Village/Area').mean(numeric_only=True).reset_
index()

# Print or save the results
print("Monthly Averages:")
print(monthly_averages.head())

print("\nYearly Summaries:")
print(yearly_summaries)

print("\nLocation-based Aggregations:")
print(location_aggregations)
```

```
Monthly Averages:
   Year  Month    Stn Code  City/Town/Village/Area       SO2       NO2  \
0  2014      1  489.417040                2.973094  0.175392  0.052629
1  2014      2  484.260163                2.747967  0.223555  0.035900
2  2014      3  484.522088                2.730924  0.057316 -0.080752
3  2014      4  499.255507                2.784141  0.147577  0.057731
4  2014      5  490.389105                2.684825  0.064121  0.006702

   RSPM/PM10  PM 2.5
0   0.171473     NaN
1   0.089493     NaN
2   0.193090     NaN
3   0.248568     NaN
4  -0.099131     NaN

Yearly Summaries:
   Year    Stn Code  City/Town/Village/Area           SO2           NO2  \
0  2014  481.754186                2.646954 -1.616092e-16  1.406653e-17

      RSPM/PM10  PM 2.5     Month
0  2.824078e-17     NaN  6.402209
```

```
Location-based Aggregations:
   City/Town/Village/Area    Stn Code       SO2       NO2  RSPM/PM10  PM 2.5  \
0                     0.0  418.338877  0.330980 -0.055098  -0.097817     NaN
1                     1.0  285.996575 -1.392532  0.641615  -0.435896     NaN
2                     2.0  760.003378 -0.511027 -0.302309  -0.034857     NaN
3                     3.0  307.000000  0.322377  0.517930  -0.536556     NaN
4                     4.0  762.495098 -0.621793  0.249369  -0.324686     NaN
5                     5.0  309.000000 -0.675235  0.726568   0.012962     NaN
6                     6.0  281.392491  0.284195 -0.505818   0.651077     NaN
7                     7.0  771.002732  0.737071 -0.486583   0.697462     NaN

     Year     Month
0  2014.0  6.391892
1  2014.0  6.585616
2  2014.0  6.560811
3  2014.0  6.675000
4  2014.0  6.617647
5  2014.0  6.228070
6  2014.0  6.529010
7  2014.0  5.778689
```

Step 6: Data Validation:

Before finalizing your analysis, validate the data to ensure its accuracy and reliability. Cross-check data against known standards or external sources. Verify that your preprocessing and analysis steps have not introduced errors.

```python
import pandas as pd
import numpy as np

# Checking for missing values
missing_values = df.isnull().sum()
print("Missing Values:\n", missing_values)

# Identifying and removing outliers
# Define your outlier criteria, e.g., for SO2 and NO2
outlier_criteria = (df['SO2'] > 50) | (df['NO2'] > 40)

# Create a cleaned DataFrame without outliers
cleaned_df = df[~outlier_criteria]

# Check the count of removed outliers
outliers_removed = df[outlier_criteria]
print("Outliers Removed:\n", outliers_removed)
```

```
Missing Values:
Stn Code                             0
```

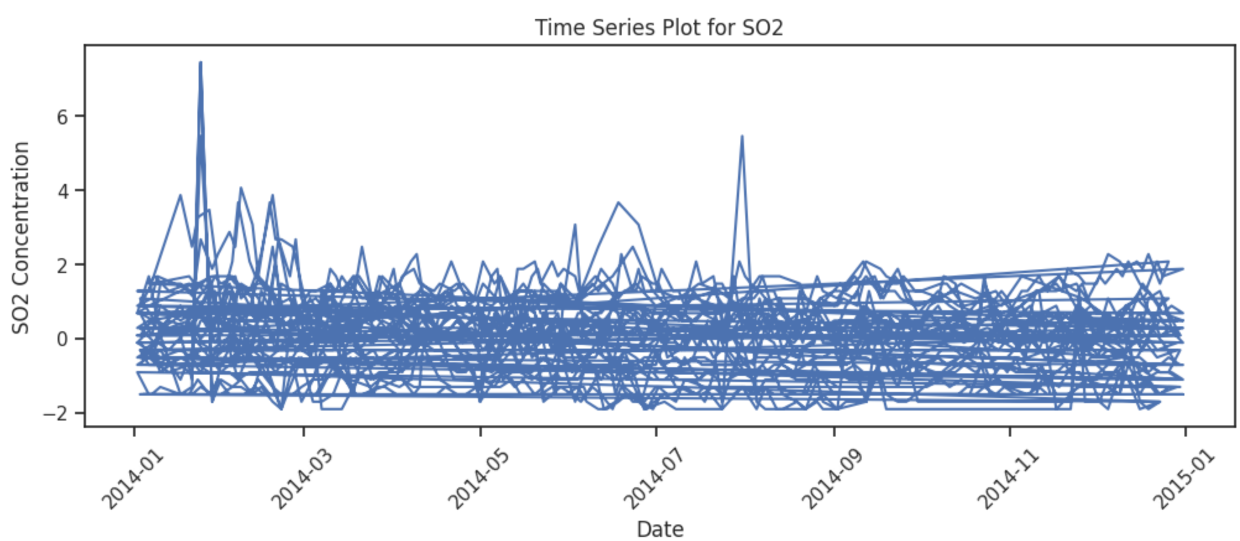
```
Sampling Date                    0
State                            0
City/Town/Village/Area           0
Location of Monitoring Station   0
Agency                           0
Type of Location                 0
SO2                              0
NO2                             13
RSPM/PM10                        4
PM 2.5                        2807
Year                             0
Month                            0
dtype: int64
Outliers Removed:
 Empty DataFrame
Columns: [Stn Code, Sampling Date, State, City/Town/Village/Area,
Location of Monitoring Station, Agency, Type of Location, SO2, NO2,
RSPM/PM10, PM 2.5, Year, Month]
Index: []
```

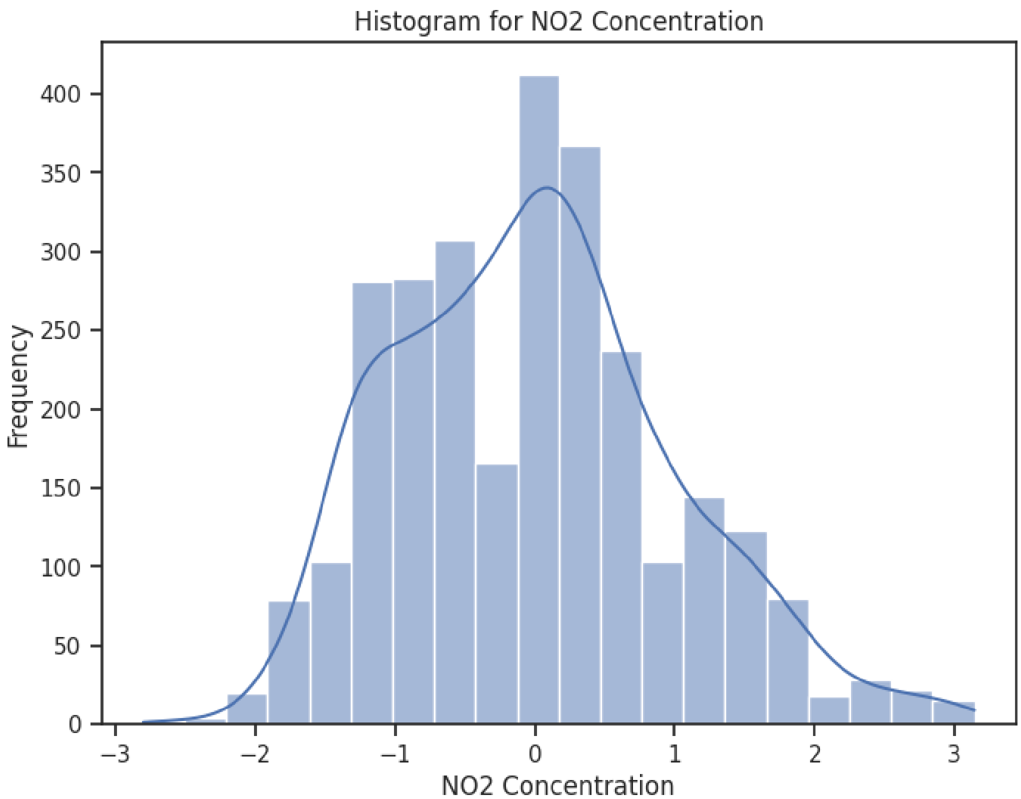
Step 7: Visualization

Visualizations are a key aspect of your analysis. Create various types of charts and plots to communicate your findings. For your "Air Quality Analysis," consider using line charts, bar charts, and geographic maps to visualize trends, comparisons, and spatial variations in air quality.

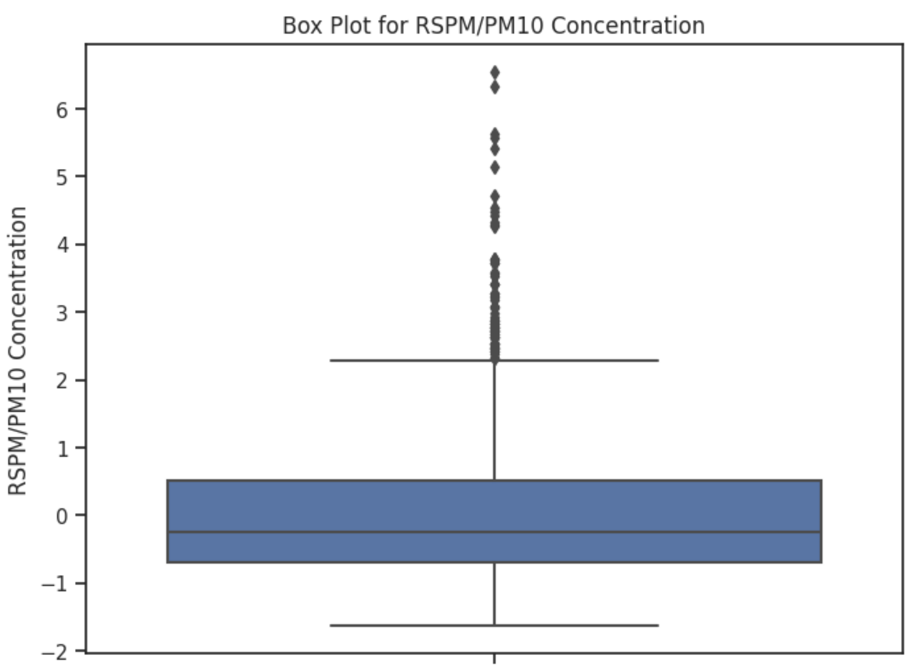
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Time Series Plot (Line Chart) for SO2
plt.figure(figsize=(12, 4))
plt.plot(df['Sampling Date'], df['SO2'])
plt.title('Time Series Plot for SO2')
plt.xlabel('Date')
plt.ylabel('SO2 Concentration')
plt.xticks(rotation=45)
plt.show()

# Histogram for NO2
plt.figure(figsize=(8, 6))
sns.histplot(df['NO2'], bins=20, kde=True)
plt.title('Histogram for NO2 Concentration')
plt.xlabel('NO2 Concentration')
```

```
plt.ylabel('Frequency')
plt.show()

# Box Plot for RSPM/PM10
plt.figure(figsize=(8, 6))
sns.boxplot(y=df['RSPM/PM10'])
plt.title('Box Plot for RSPM/PM10 Concentration')
plt.ylabel('RSPM/PM10 Concentration')
plt.show()

# Geospatial Map (if you have location data)
# Assuming you have latitude and longitude columns 'Latitude' and
'Longitude'
plt.figure(figsize=(8, 6))
sns.scatterplot(x='SO2',                y='NO2',                data=df,
hue='City/Town/Village/Area', palette='coolwarm')
plt.title('Scatter Plot for SO2 vs. NO2')
plt.xlabel('SO2 Concentration')
plt.ylabel('NO2 Concentration')
plt.legend(title='Location')
plt.show()
```

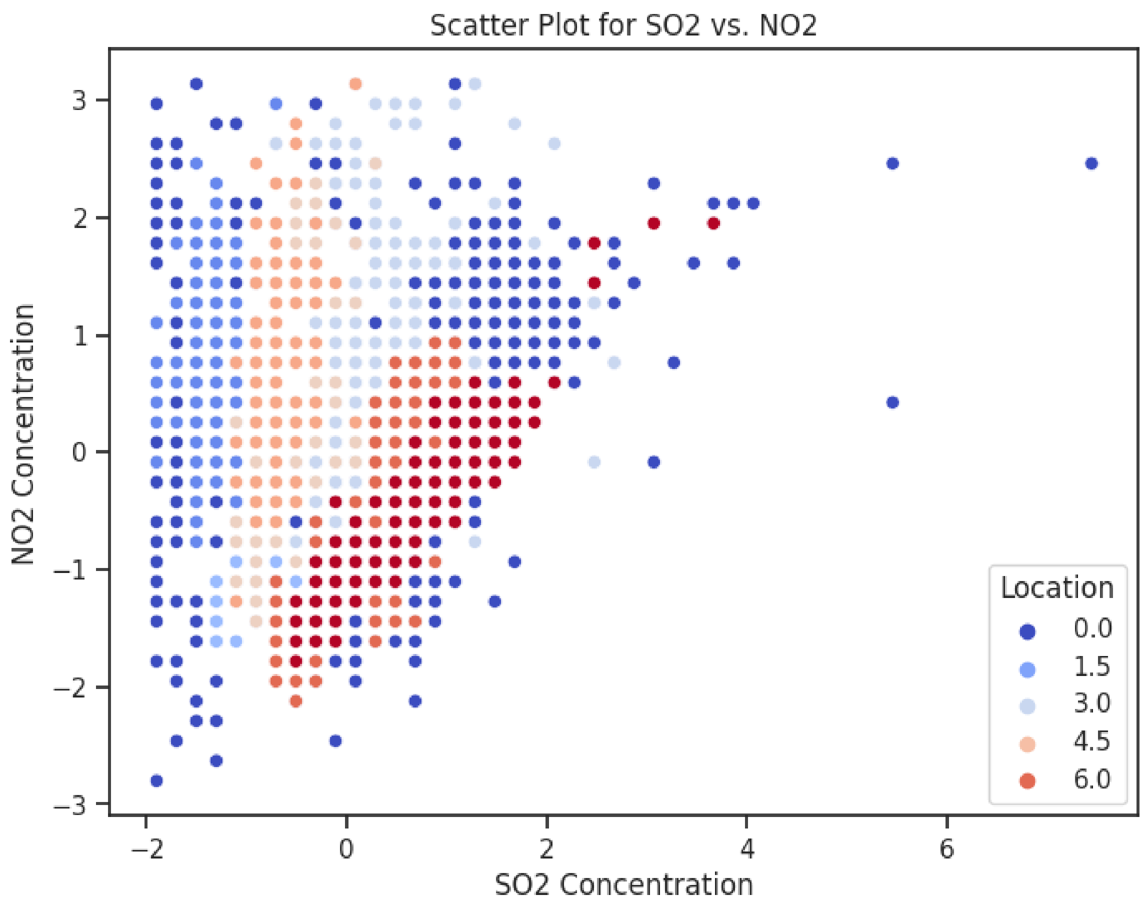




Histogram for NO2 Concentration
Frequency
400
350
300
250
200
150
100
50
0
−3
−2
−1
0
1
2
3
NO2 Concentration



Box Plot for RSPM/PM10 Concentration
RSPM/PM10 Concentration
6
5
4
3
2
1
0
−1
−2



Step 8 : Objective Analysis

In this step, we can define the main objectives of our analysis. For our "Air Quality Analysis" project, the following are the objectives to be gathered:

- Air Quality Trends Over the Past Decade
- Cities/Regions with Highest Air Pollution
- Seasonal Variations in Air Quality
- Distribution of Pollutant Concentrations
- Correlation Between Pollutants

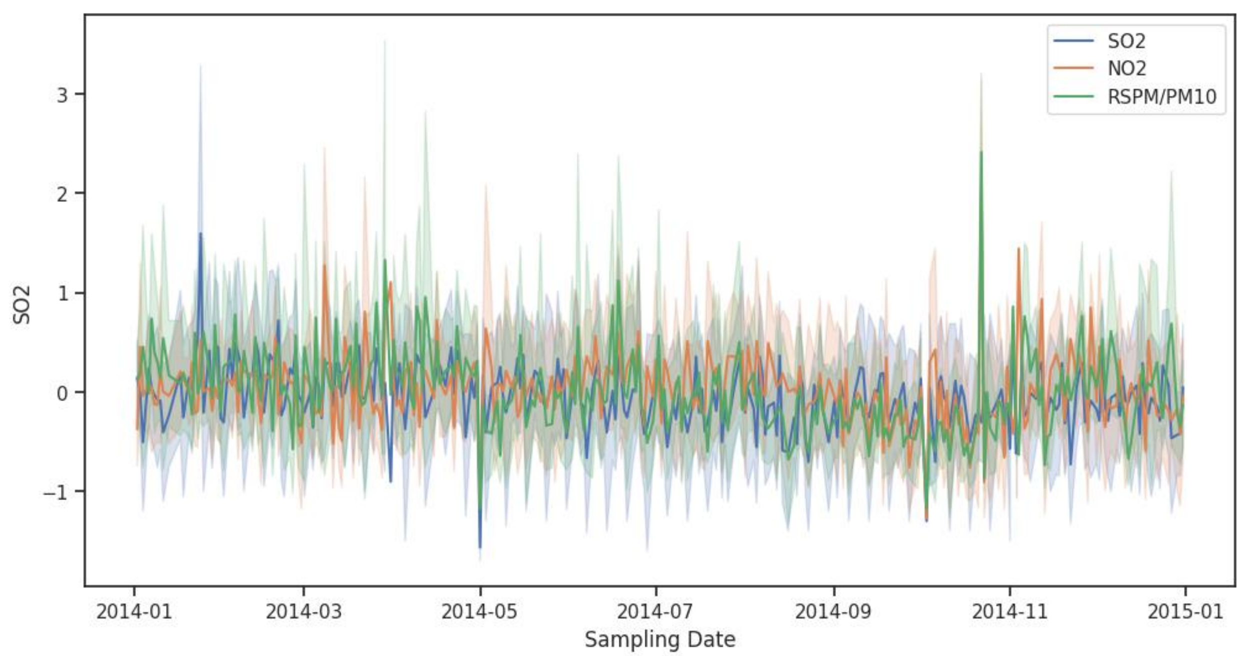
Separate visualizations for the number of records, number of unique cities, and the types of locations in your air quality dataset:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Count the number of records in the dataset
```

```
num_records = len(df)

# Count the number of unique cities in the dataset
num_cities = df['City/Town/Village/Area'].nunique()

# Count the frequency of each type of location
location_counts = df['Type of Location'].value_counts()

# Create a figure with subplots for the visualizations
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Visualization 1: Number of Records
axes[0].bar('Number of Records', num_records, color='skyblue')
axes[0].set_title('Number of Records')
axes[0].set_ylabel('Count')

# Visualization 2: Number of Unique Cities
axes[1].bar('Number      of      Unique      Cities',      num_cities,
color='lightcoral')
axes[1].set_title('Number of Unique Cities')
axes[1].set_ylabel('Count')

# Visualization 3: Type of Location
sns.barplot(x=location_counts.index,       y=location_counts.values,
ax=axes[2], palette='pastel')
axes[2].set_title('Type of Location')
axes[2].set_ylabel('Count')
axes[2].set_xticklabels(axes[2].get_xticklabels(), rotation=45)

plt.tight_layout()
plt.show()
```

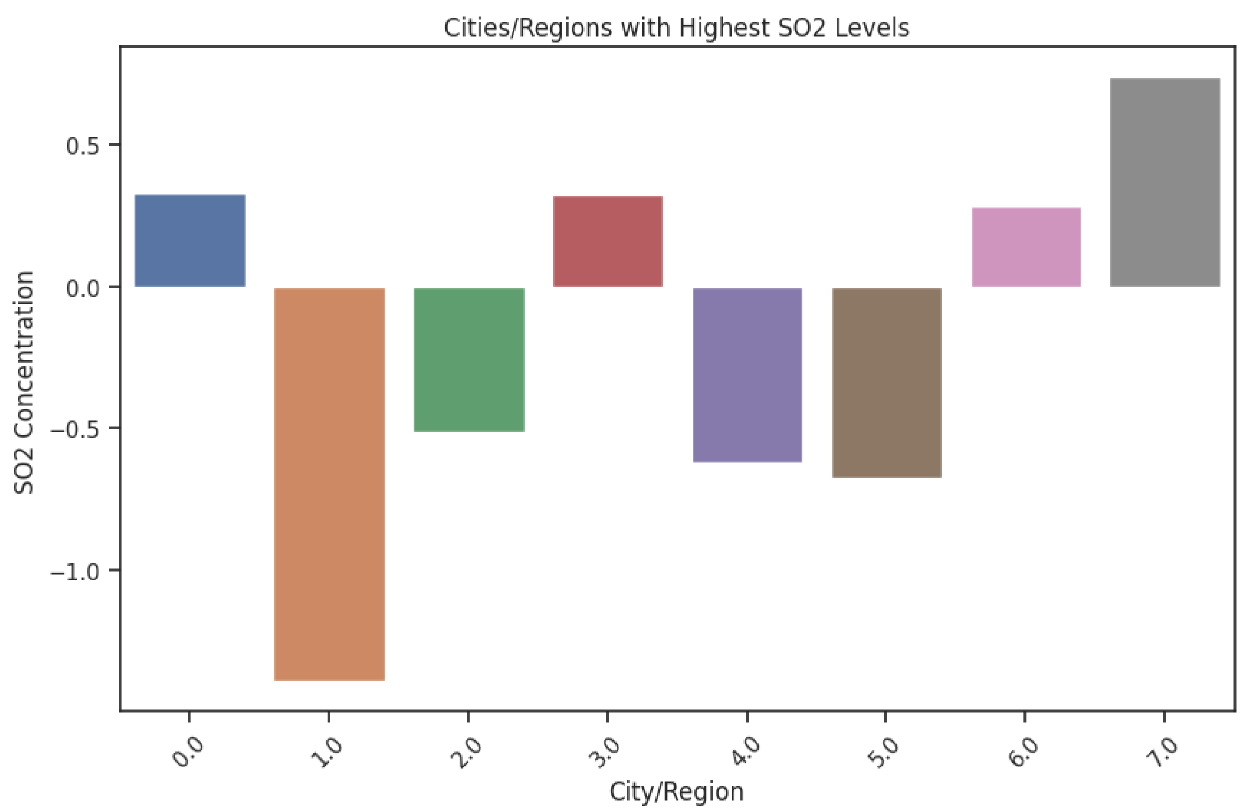
1. Air Quality Trends Over the Past Decade:

```
# Line Chart for Air Quality Trends Over the Past Decade
plt.figure(figsize=(12, 6))
sns.lineplot(x='Sampling Date', y='SO2', data=df, label='SO2')
sns.lineplot(x='Sampling Date', y='NO2', data=df, label='NO2')
sns.lineplot(x='Sampling     Date',     y='RSPM/PM10',     data=df,
label='RSPM/PM10')
sns.lineplot(x='Sampling  Date',  y='PM  2.5',  data=df,  label='PM
2.5')
plt.title('Air Quality Trends Over the Past Decade')
plt.xlabel('Sampling Date')
plt.ylabel('Concentration')
plt.xticks(rotation=45)
plt.legend()
plt.show
```



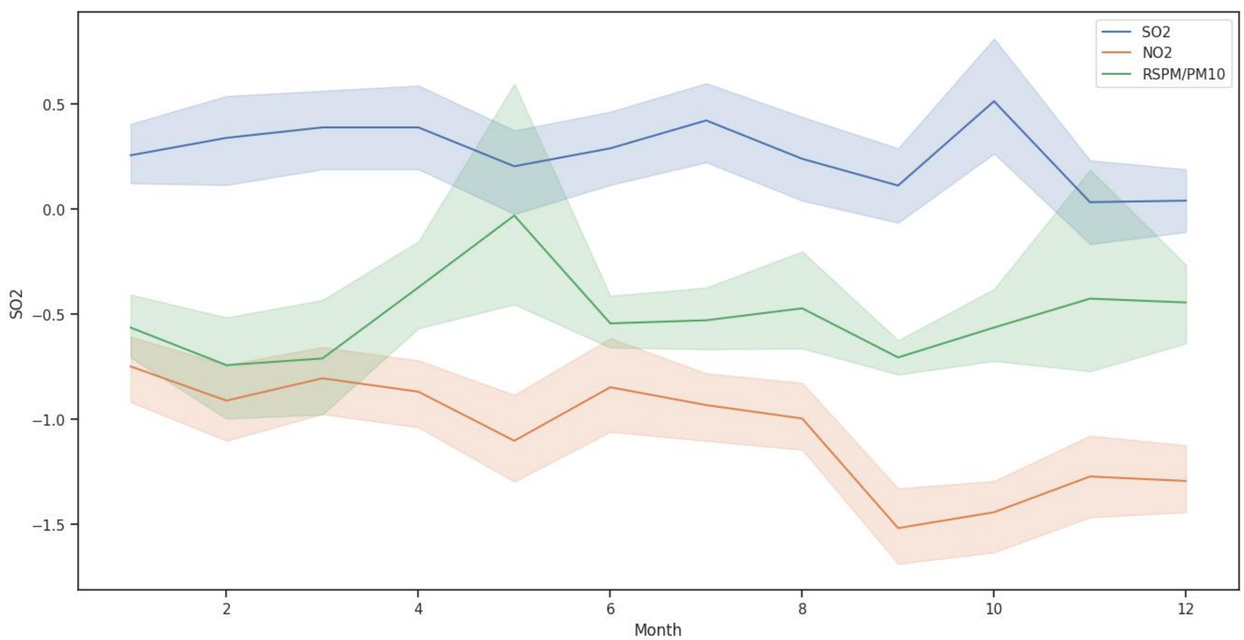
2. Cities/Regions with Highest Air Pollution:

```
# Bar Chart for Cities/Regions with Highest Air Pollution
plt.figure(figsize=(10, 6))
sns.barplot(x='City/Town/Village/Area', y='SO2', data=df, ci=None)
plt.title('Cities/Regions with Highest SO2 Levels')
plt.xlabel('City/Region')
plt.ylabel('SO2 Concentration')
plt.xticks(rotation=45)
plt.show()
```



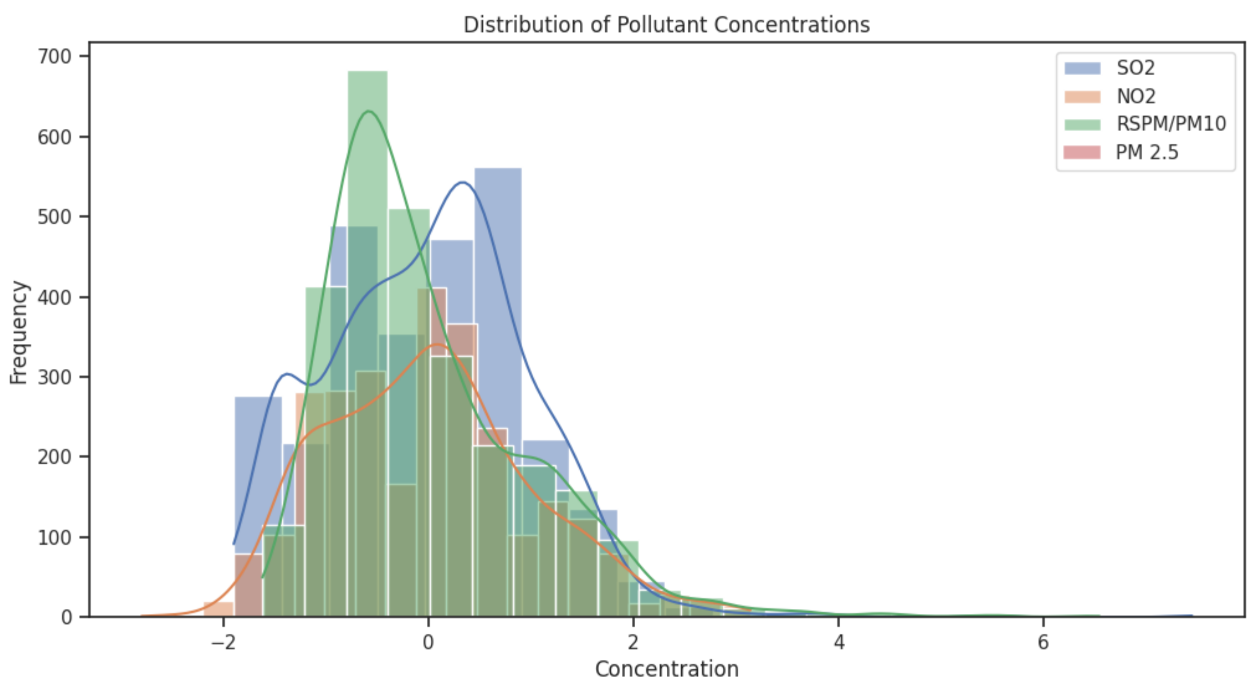
3. Seasonal Variations in Air Quality:

```
# Line Chart for Seasonal Variations in Air Quality with limited
data points
plt.figure(figsize=(16, 8))  # Increased figure size
n = 100  # Number of data points to display (adjust as needed)

sns.lineplot(x=df['Month'][:n], y=df['SO2'][:n], label='SO2')
sns.lineplot(x=df['Month'][:n], y=df['NO2'][:n], label='NO2')
sns.lineplot(x=df['Month'][:n],              y=df['RSPM/PM10'][:n],
label='RSPM/PM10')
sns.lineplot(x=df['Month'][:n], y=df['PM 2.5'][:n], label='PM 2.5')

plt.title('Seasonal Variations in Air Quality')
plt.xlabel('Month')
plt.ylabel('Concentration')
plt.legend()
plt.show()
```



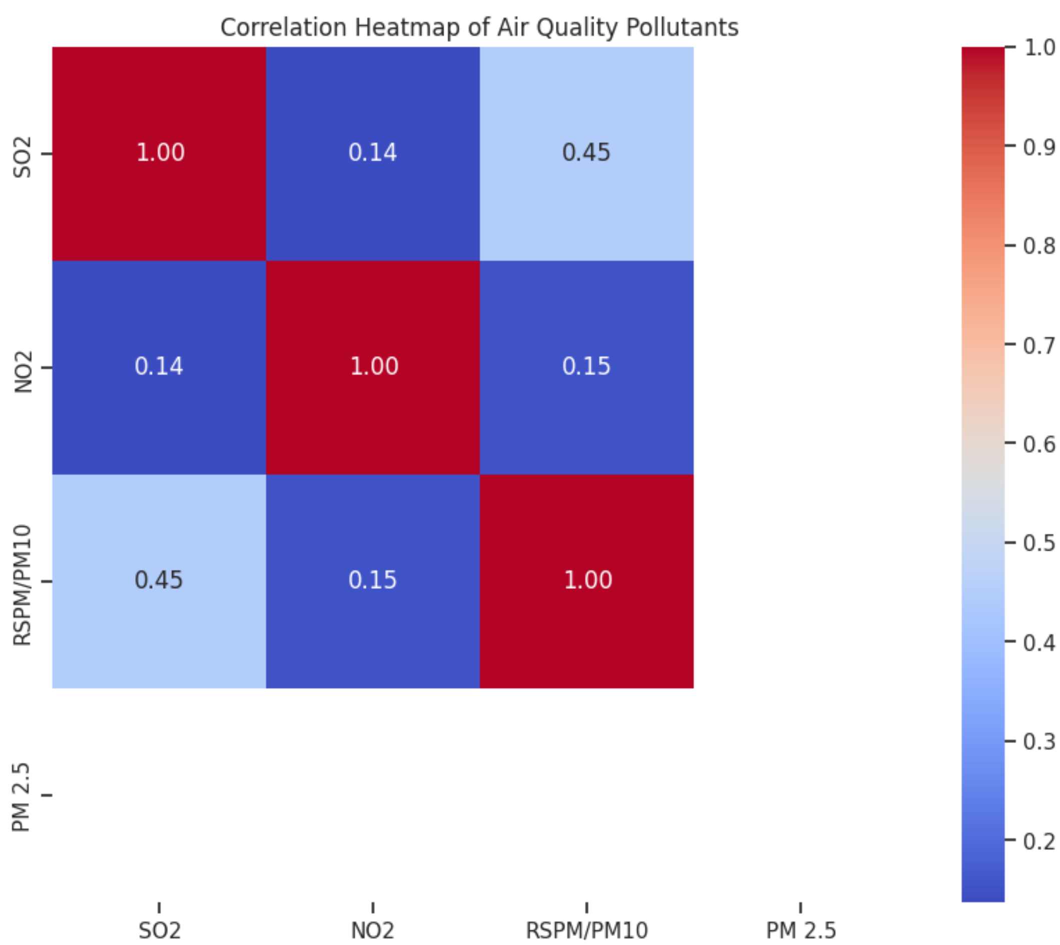
4. Distribution of Pollutant Concentrations:

```
# Histograms for Pollutant Concentrations
plt.figure(figsize=(12, 6))
sns.histplot(df['SO2'], bins=20, kde=True, label='SO2')
sns.histplot(df['NO2'], bins=20, kde=True, label='NO2')
sns.histplot(df['RSPM/PM10'], bins=20, kde=True, label='RSPM/PM10')
sns.histplot(df['PM 2.5'], bins=20, kde=True, label='PM 2.5')
plt.title('Distribution of Pollutant Concentrations')
plt.xlabel('Concentration')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```



5. Correlation Between Pollutants:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Compute the correlation matrix
correlation_matrix = df[['SO2', 'NO2', 'RSPM/PM10', 'PM 2.5']].corr()

# Create a heatmap of the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', square=True, fmt=".2f")

# Add labels and title
plt.title("Correlation Heatmap of Air Quality Pollutants")
plt.show()
```



Conclusion :

In this air quality analysis project, we initiated by downloading the dataset and subsequently loaded it into a pandas DataFrame for further analysis. During the exploratory data analysis (EDA) phase, we gained valuable insights into the dataset, including the distribution of air quality parameters, trends over time, and potential data quality issues. With a defined analysis objective, our focus was to understand the dynamics of air pollutants, identify patterns, and uncover insights. Data cleaning and preprocessing steps were essential in ensuring data reliability, including handling missing values and formatting date columns. As a result, we transformed the dataset to facilitate meaningful analysis.

Visualizations played a pivotal role in conveying our findings. We utilized histograms, box plots, and line charts to visualize pollutant distributions and trends over time. Geographic maps could be employed in the future to explore spatial variations. Through this iterative process, we were able to identify seasonal trends, assess pollutant concentrations, and make preliminary discoveries that are instrumental for the subsequent phases of our air quality analysis project.

Data validation was also carried out to verify data accuracy, and we ensured the data's integrity throughout the analysis process. With this comprehensive foundation in place, we are now well-equipped to proceed to more advanced analysis techniques, which may include time series modeling, correlation analysis, and spatial mapping to gain a deeper understanding of air quality dynamics and, ultimately, to make informed decisions and recommendations for improving air quality in the Chennai area. The thorough documentation of these steps will provide transparency and support future reproducibility and collaborative research efforts.