AIR QUALITY ANALYSIS IN TAMIL NADU USING COGNOS

Project Title: Air Quality Analysis In Tamil Nadu

Phase 5 : Project Documentation & Submission

Topic : Analysis by loading and preprocessing the air quality dataset using data manipulation libraries (e.g., pandas) and creating visualizations using data visualization libraries (e.g., Matplotlib, Seaborn) and visualization using cognos.

TEAM MEMBERS

DEEP R SHAH -91762115011

MOHANA SUNDARAM V -91762115306

GUNAALAN P G -2015018

PAVITHRA M -91762115308

Objectives:

The primary objectives of this project are to analyze air quality data for Tamil Nadu in 2014 and provide insights into air pollution trends and pollution levels.

Analysis Approach:

This analysis involves several steps, including data download, data preprocessing, visualization, and trend identification.

Step 1: Data Collection

- Visit the provided dataset link: Tamil Nadu Air Quality Dataset.
- Download the dataset in a format such as CSV or Excel.
- Ensure that the dataset is in a structured format, such as a CSV file, to facilitate analysis.

Step 2: Import Necessary Libraries

Before you start, make sure you have Python and the Pandas library installed. You can install Pandas using pip if it's not already installed.

Input: pip install pandas

Step 3: Load the Dataset

Assuming you have downloaded the dataset as a CSV file, you can load it into a Pandas DataFrame like this:

Input:
```
import pandas as pd
from google.colab import files
upload = files.upload()
df = pd.read_csv('/content/cpcb_dly_aq_tamil_nadu-2014.csv')
df.head()
```

Output :

	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 4: Data Preprocessing

Data preprocessing may involve various tasks, such as handling missing values, data cleaning, and feature engineering.

- Load the dataset into a Pandas DataFrame for further analysis.
- Address missing values, outliers, and inconsistencies in the data.
- Perform data cleaning, which may include removing duplicates, correcting data types, and ensuring data integrity.

Here are some common preprocessing steps:

Handle Missing Values: If there are missing values in the dataset, you can use Pandas to fill them or drop rows/columns with missing values.

Data Cleaning: Check for and clean any outliers or incorrect data.

Feature Engineering: Create new features or transform existing ones if needed.

Data Exploration: You can use various Pandas functions to explore and understand the data. For example, you can use df.describe(), df.info(), and df['column_name'].value_counts() to gain insights into the dataset.

Input: `print(df.isnull().sum())`

Output:

```
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                               11
NO2                               13
RSPM/PM10                          4
PM 2.5                          2879
dtype: int64
```

Input: `print(df.describe())`

Output:

```
         Stn Code          SO2          NO2    RSPM/PM10  PM 2.5
count  2879.000000  2868.000000  2866.000000  2875.000000     0.0
mean    475.750261    11.503138    22.136776    62.494261     NaN
std     277.675577     5.051702     7.128694    31.368745     NaN
min      38.000000     2.000000     5.000000    12.000000     NaN
25%     238.000000     8.000000    17.000000    41.000000     NaN
50%     366.000000    12.000000    22.000000    55.000000     NaN
75%     764.000000    15.000000    25.000000    78.000000     NaN
max     773.000000    49.000000    71.000000   269.000000     NaN
```

Input: `df.drop(["PM 2.5"],axis=1,inplace=True)`

Input:`df.info()`

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2879 entries, 0 to 2878
Data columns (total 10 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Stn Code                        2879 non-null   int64  
 1   Sampling Date                   2879 non-null   object 
 2   State                           2879 non-null   object 
 3   City/Town/Village/Area          2879 non-null   object 
 4   Location of Monitoring Station  2879 non-null   object 
 5   Agency                          2879 non-null   object 
 6   Type of Location                2879 non-null   object 
 7   SO2                             2868 non-null   float64
 8   NO2                             2866 non-null   float64
 9   RSPM/PM10                       2875 non-null   float64
dtypes: float64(3), int64(1), object(6)
memory usage: 225.0+ KB
```

Input:`df.columns`

Output:

```
Index(['Stn Code', 'Sampling Date', 'State', 'City/Town/Village/Area',
       'Location of Monitoring Station', 'Agency', 'Type of Location', 'SO2',
       'NO2', 'RSPM/PM10'],
      dtype='object')
```

Input:`df.shape`

Output:

```
(2879, 10)
```

Input: `df.dtypes`

Output:

```
Stn Code                              int64
Sampling Date                        object
State                                object
City/Town/Village/Area               object
Location of Monitoring Station       object
Agency                               object
Type of Location                     object
SO2                                 float64
NO2                                 float64
RSPM/PM10                           float64
dtype: object
```

Input: `df.index`

Output:

```
RangeIndex(start=0, stop=2879, step=1)
```

Input:`df["SO2"].fillna(0,inplace=True)`

Input: `df["NO2"].fillna(0,inplace=True)`

Input: `df["RSPM/PM10"].fillna(0,inplace=True)`

Input: `print(df.isnull().sum())`

Output:

```
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                               0
RSPM/PM10                         0
dtype: int64
```

Input: `df['SO2'].unique()`

Output:

```
array([11., 13., 12., 15., 14., 10., 16., 19.,  9., 20., 17., 18., 25.,
       21., 23., 26., 24., 32., 27., 30., 22.,  0.,  8., 31., 28., 29.,
        6., 49.,  3.,  7.,  5.,  2.,  4., 39.])
```

Input: `df.agg(['min', 'max'])`

Output:

	Stn Code	Sampling Date	State	City/Town/Village/Area	Location of Monitoring Station	Agency	Type of Location	SO2	NO2	RSPM/PM10
min	38	01-02-14	Tamil Nadu	Chennai	AVM Jewellery Building, Tuticorin	National Environmental Engineering Research In...	Industrial Area	0.0	0.0	0.0
max	773	31-12-14	Tamil Nadu	Trichy	Thiyagaraya Nagar, Chennai	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	49.0	71.0	269.0

Input: `df.mean()`

Output:

```
Stn Code      475.750261
SO2            11.459187
NO2            22.036818
RSPM/PM10      62.407433
dtype: float64
```

Input: `df.median()`

Output:

```
Stn Code      366.0
SO2            12.0
NO2            21.0
RSPM/PM10      55.0
dtype: float64
```

Input:`df.var()`

Output:

```
Stn Code      77103.726003
SO2              25.925974
NO2              52.792251
RSPM/PM10       988.051105
dtype: float64
```

Input: `df.std()`

Output:

```
Stn Code      277.675577
SO2             5.091756
NO2             7.265828
RSPM/PM10      31.433280
dtype: float64
```

Input: `df.kurtosis()`

Output:

```
Stn Code     -1.714667
SO2           2.200113
NO2           3.643779
RSPM/PM10     3.345704
dtype: float64
```

Step 5: Save the Preprocessed Data

If we want to save the preprocessed dataset for future use, we can save it to a new CSV file using to_csv():

Input: `df.to_csv('preprocessed_dataset.csv', index=False)`

Here is the [preprocessed dataset]()

Step 6: Create Visualizations

Use data visualization libraries such as Matplotlib and Seaborn to create visualizations that help convey your findings. Here are some visualization:

- Line Plot
- Scatter Plot
- Bar Plot
- Histogram
- Pie Chart

Input:

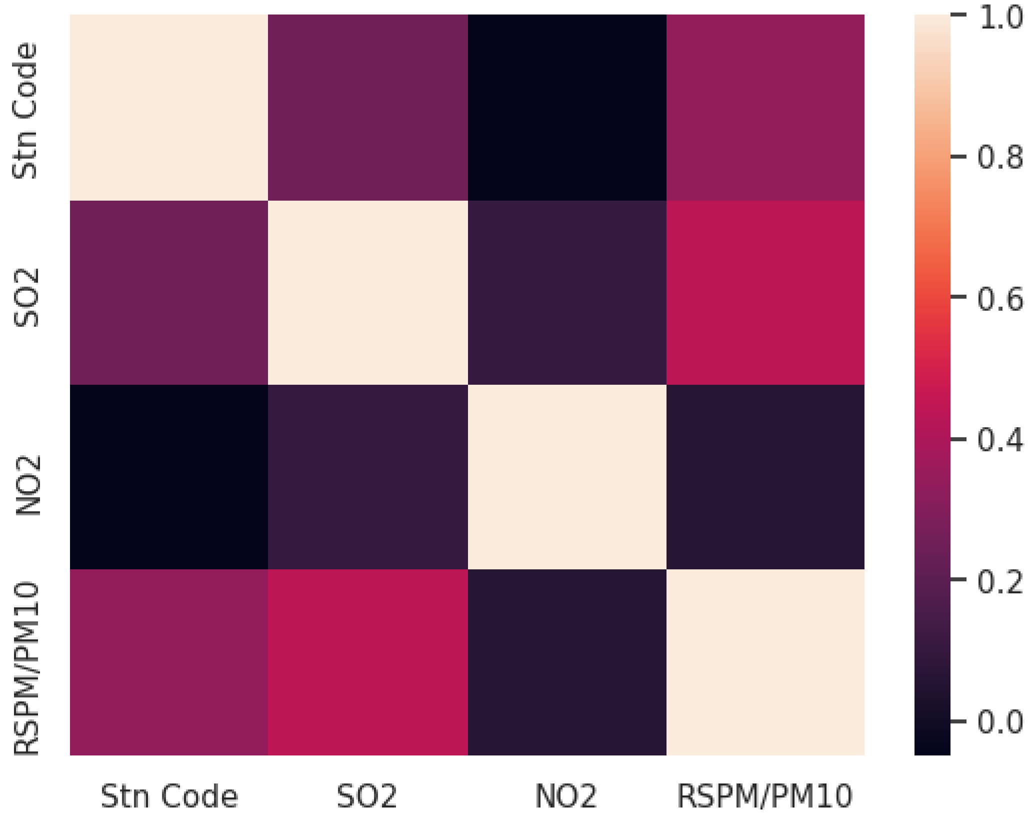
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()
```

Input:

```
import seaborn as sns
corr = df.corr()
print(corr)
sns.heatmap(corr,
         xticklabels=corr.columns,
         yticklabels=corr.columns)
```

Output:

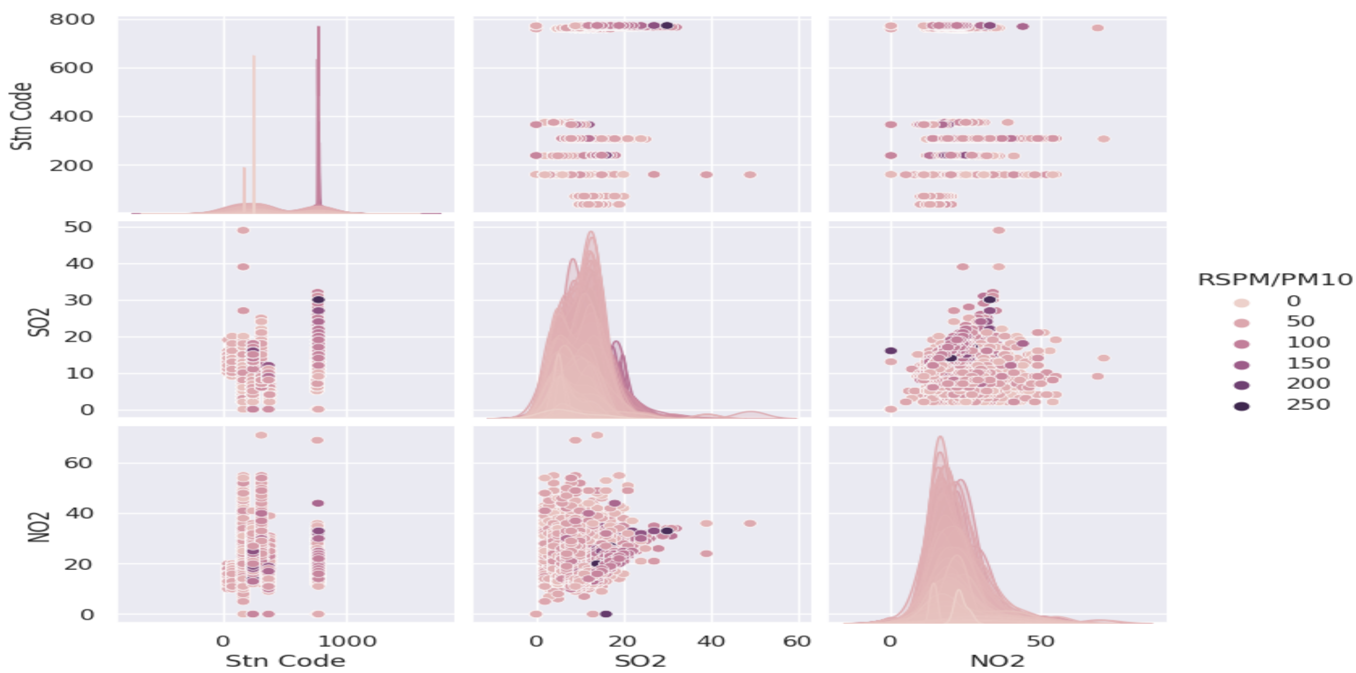
	Stn Code	SO2	NO2	RSPM/PM10
Stn Code	1.000000	0.257289	-0.048771	0.337457
SO2	0.257289	1.000000	0.100779	0.440141
NO2	-0.048771	0.100779	1.000000	0.060369
RSPM/PM10	0.337457	0.440141	0.060369	1.000000



Input:

```
import seaborn as sns
pd=sns.pairplot(df,hue = 'RSPM/PM10')
```

Output:

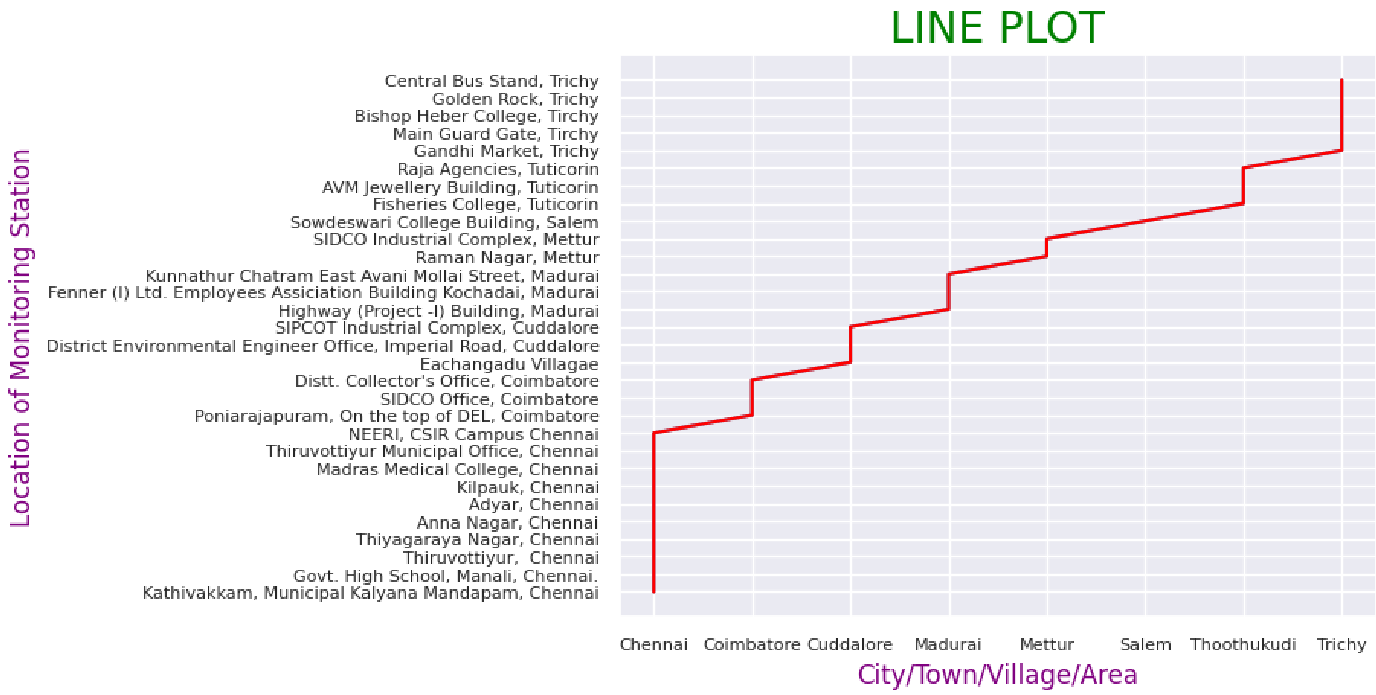


Input:

```
x=df['City/Town/Village/Area']
y=df['Location of Monitoring Station']
```

```
plt.plot(x,y)
plt.title("LINE PLOT",color="Green",fontsize=20)
plt.xlabel("City/Town/Village/Area",color="purple")
plt.ylabel("Location of Monitoring Station",color="purple")
plt.tick_params(axis='x', which='major', labelsize=8)
plt.tick_params(axis='y', which='major', labelsize=8)
plt.plot(x,y,color="red")
plt.show()
```

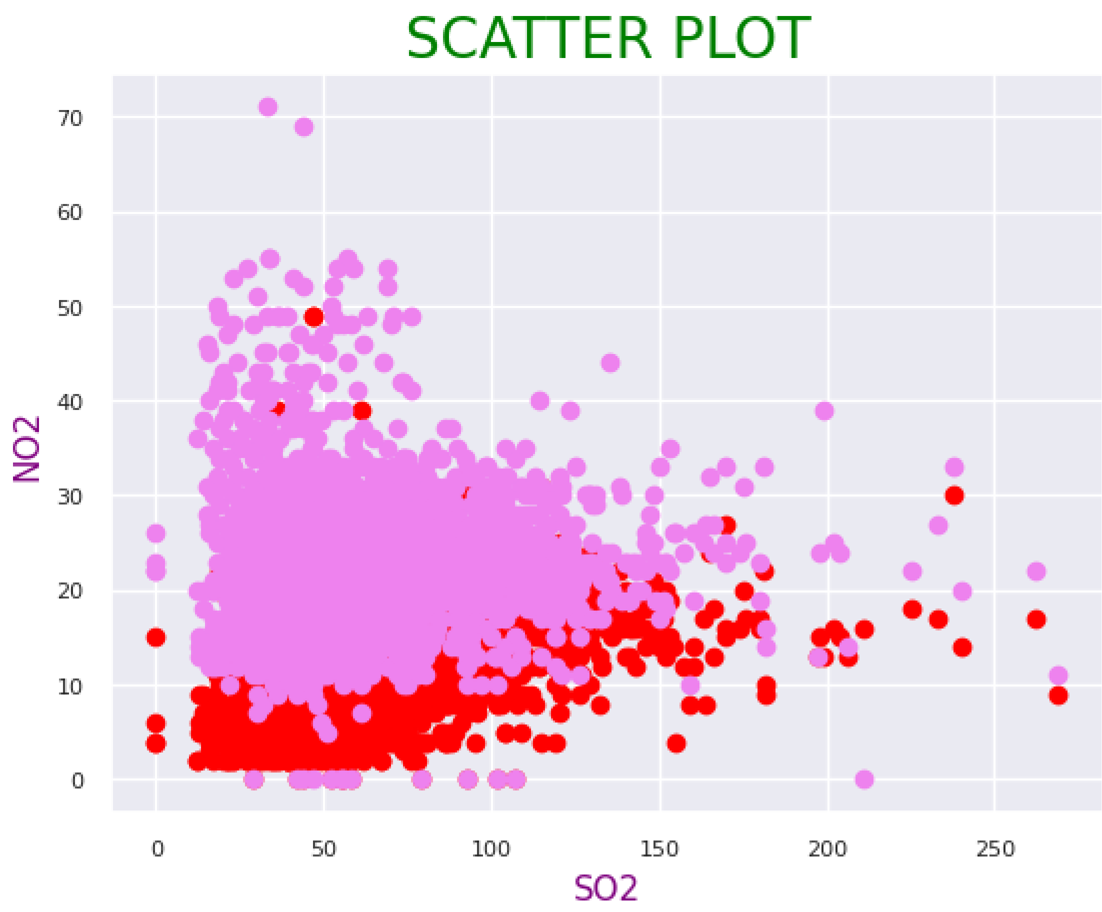
Output:



Input:

```
x=df['SO2']
y=df['NO2']
z=df['RSPM/PM10']
plt.scatter(z,x,c='red')
plt.scatter(z,y,c='violet')
plt.title("SCATTER PLOT",color="Green",fontsize=20)
plt.xlabel("SO2",color="purple")
plt.ylabel("NO2",color="purple")
plt.tick_params(axis='x', which='major', labelsize=8)
plt.tick_params(axis='y', which='major', labelsize=8)
plt.show()
```

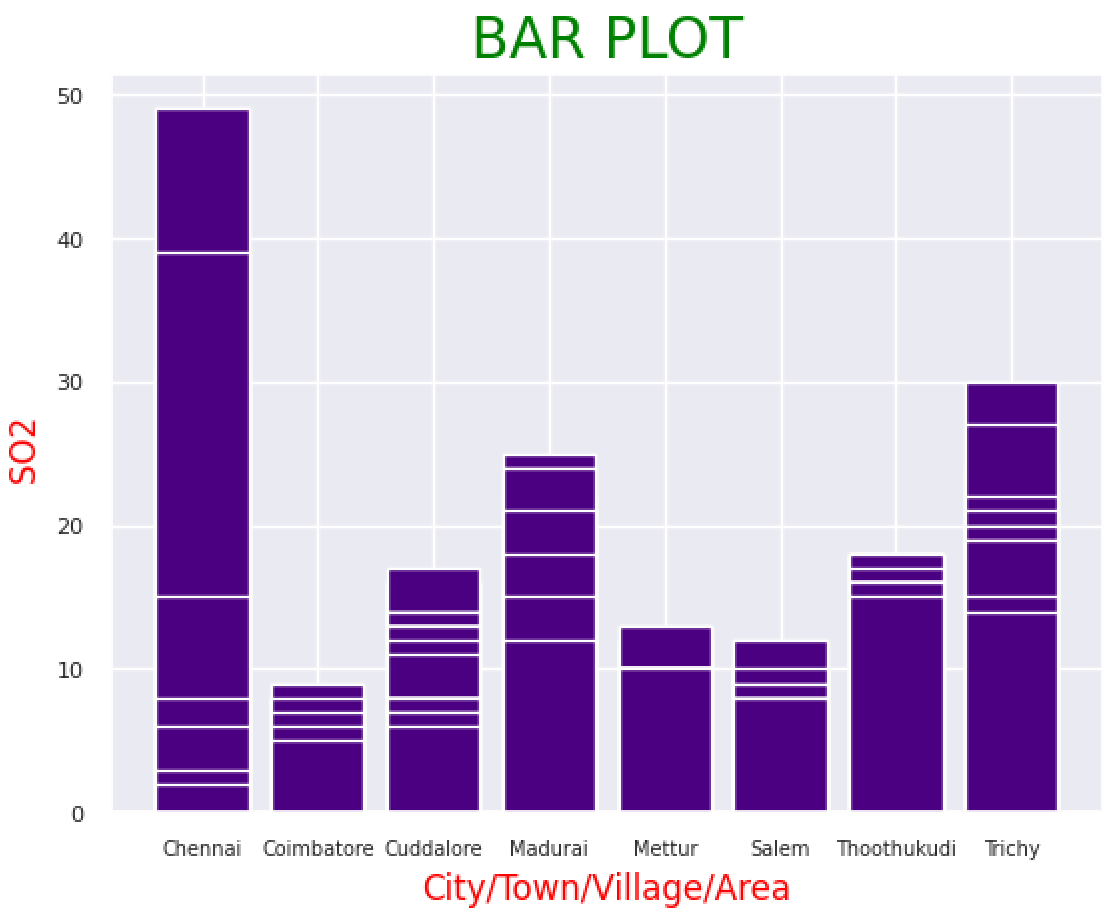
Output:



Input:

```
x=df['City/Town/Village/Area']
y=df['SO2']
plt.bar(x,y,color="indigo")
plt.title("BAR PLOT",color="Green",fontsize=20)
plt.xlabel("City/Town/Village/Area",color="red")
plt.ylabel("SO2 ",color="red")
plt.tick_params(axis='x', which='major', labelsize=7)
plt.tick_params(axis='y', which='major', labelsize=8)
plt.show()
```

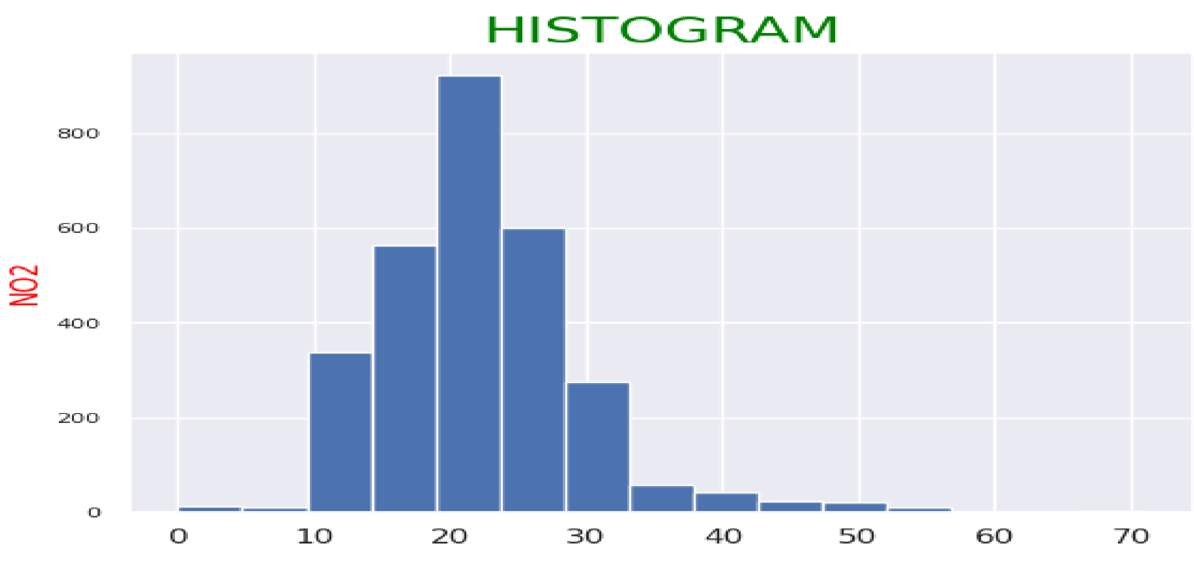
Output:



Input:

```
y=df['NO2']
plt.hist(y,bins=15)
plt.title("HISTOGRAM",color="Green",fontsize=20)
plt.ylabel("NO2",color="red")
plt.tick_params(axis='y', which='major', labelsize=8)
plt.show()
```

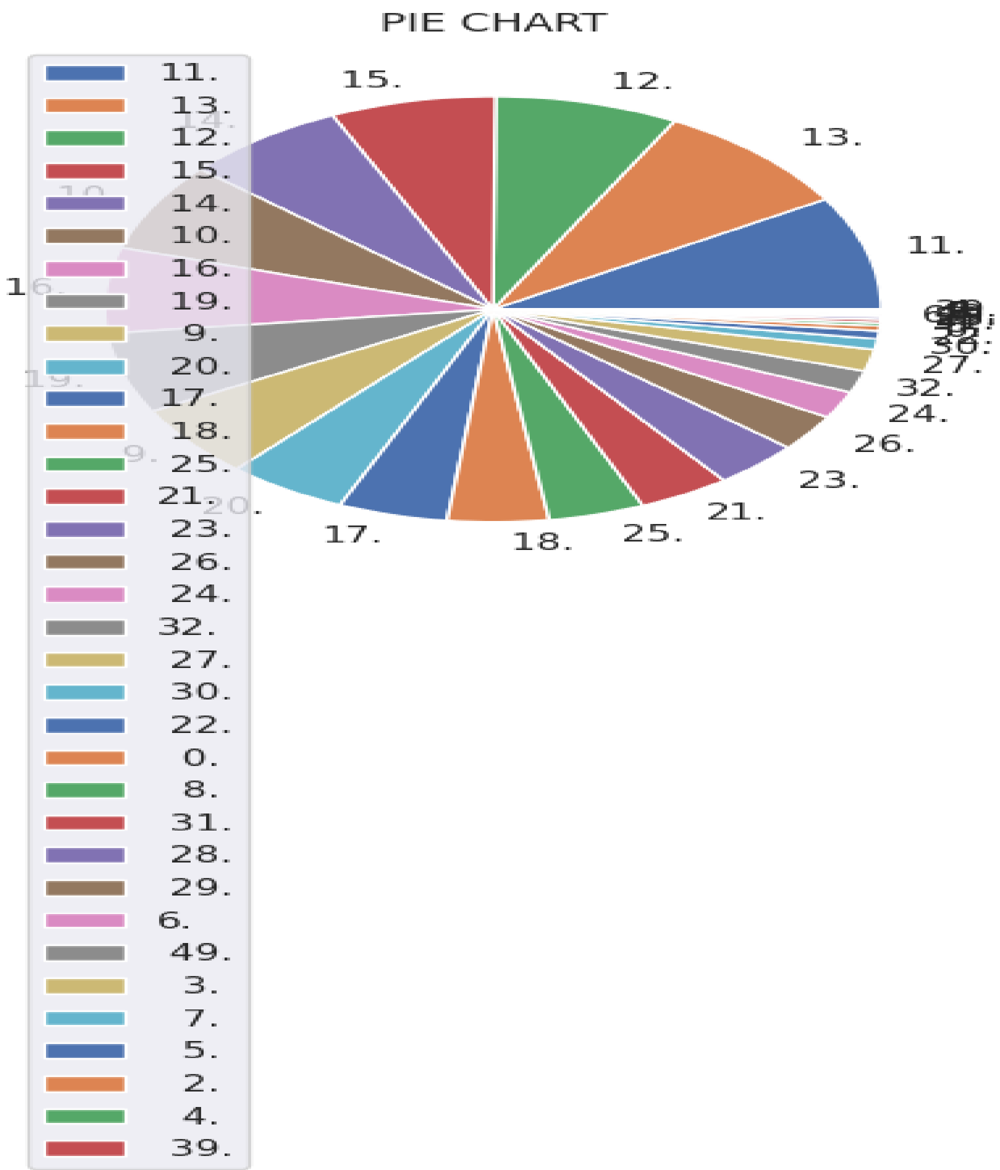
Output:



Input:

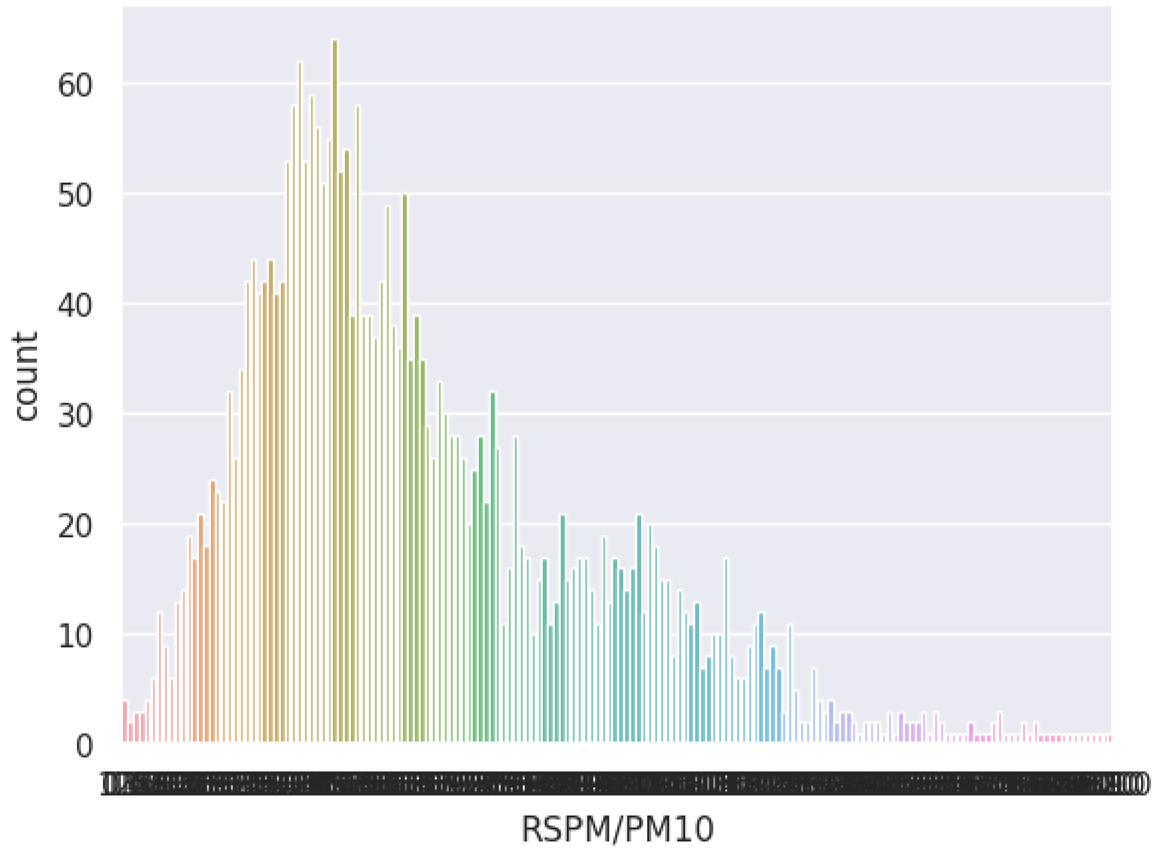
```
x=df['SO2'].value_counts()
y=['11.',' 13.',' 12.',' 15.',' 14.',' 10.',' 16.',' 19.','
9.',' 20.', '17.',' 18.',' 25.', '21.',' 23.',' 26.',' 24.',
'32.',' 27.',' 30.',' 22.','  0.','  8.',' 31.',' 28.','
29.','6.',' 49.','  3.','  7.','  5.','  2.','  4.',' 39.']
plt.pie(x.values,labels=y)
plt.title('PIE CHART')
plt.tick_params(axis='x', which='major', labelsize=8)
plt.tick_params(axis='y', which='major', labelsize=8)
plt.legend()
```

Output:



Input: `sns.countplot(x='RSPM/PM10',data=df)`

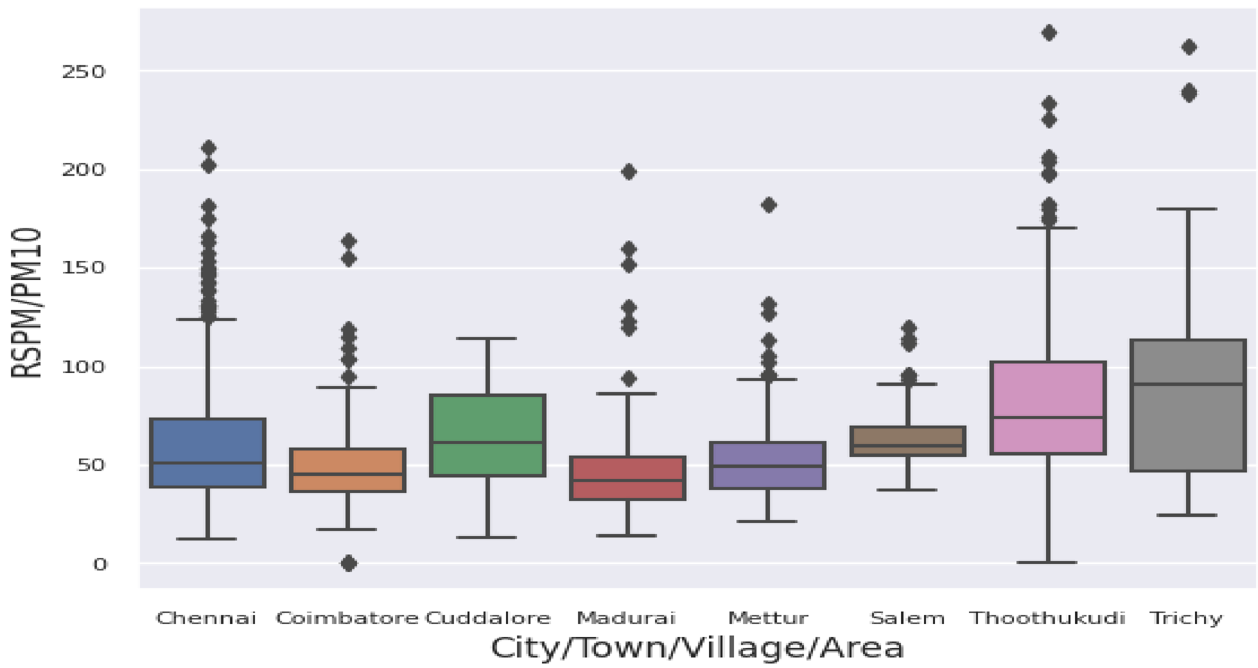
Output:



Input:

```
sns.boxplot(x='City/Town/Village/Area',y='RSPM/PM10',data=df)
plt.tick_params(axis='x', which='major', labelsize=8)
plt.tick_params(axis='y', which='major', labelsize=8)
```

Output:



Step 7: Calculate Average Pollution Levels

Calculate the average levels of SO2, NO2, and RSPM/PM10 across different monitoring stations, cities, or areas. You can use the Pandas groupby function to group the data by the relevant columns (e.g., station name, city), and then calculate the means.

Input:

```
# Group by monitoring station and calculate average levels
avg_so2 = df.groupby('Location of Monitoring
Station')['SO2'].mean()
avg_no2 = df.groupby('Location of Monitoring
Station')['NO2'].mean()
avg_rspm_pm10 = df.groupby('Location of Monitoring
Station')['RSPM/PM10'].mean()
```

Step 8: Identify Pollution Trends:

Analyze the calculated averages to identify pollution trends and areas with high pollution levels. You can sort and filter the data to find the highest and lowest pollution levels.

Input:

```
# Find the station with the highest average SO2 level
highest_so2_station = avg_so2.idxmax()
highest_so2_value = avg_so2.max()

# Find the station with the highest average NO2 level
highest_no2_station = avg_no2.idxmax()
highest_no2_value = avg_no2.max()

# Find the station with the highest average RSPM/PM10 level
highest_rspm_pm10_station = avg_rspm_pm10.idxmax()
highest_rspm_pm10_value = avg_rspm_pm10.max()
```

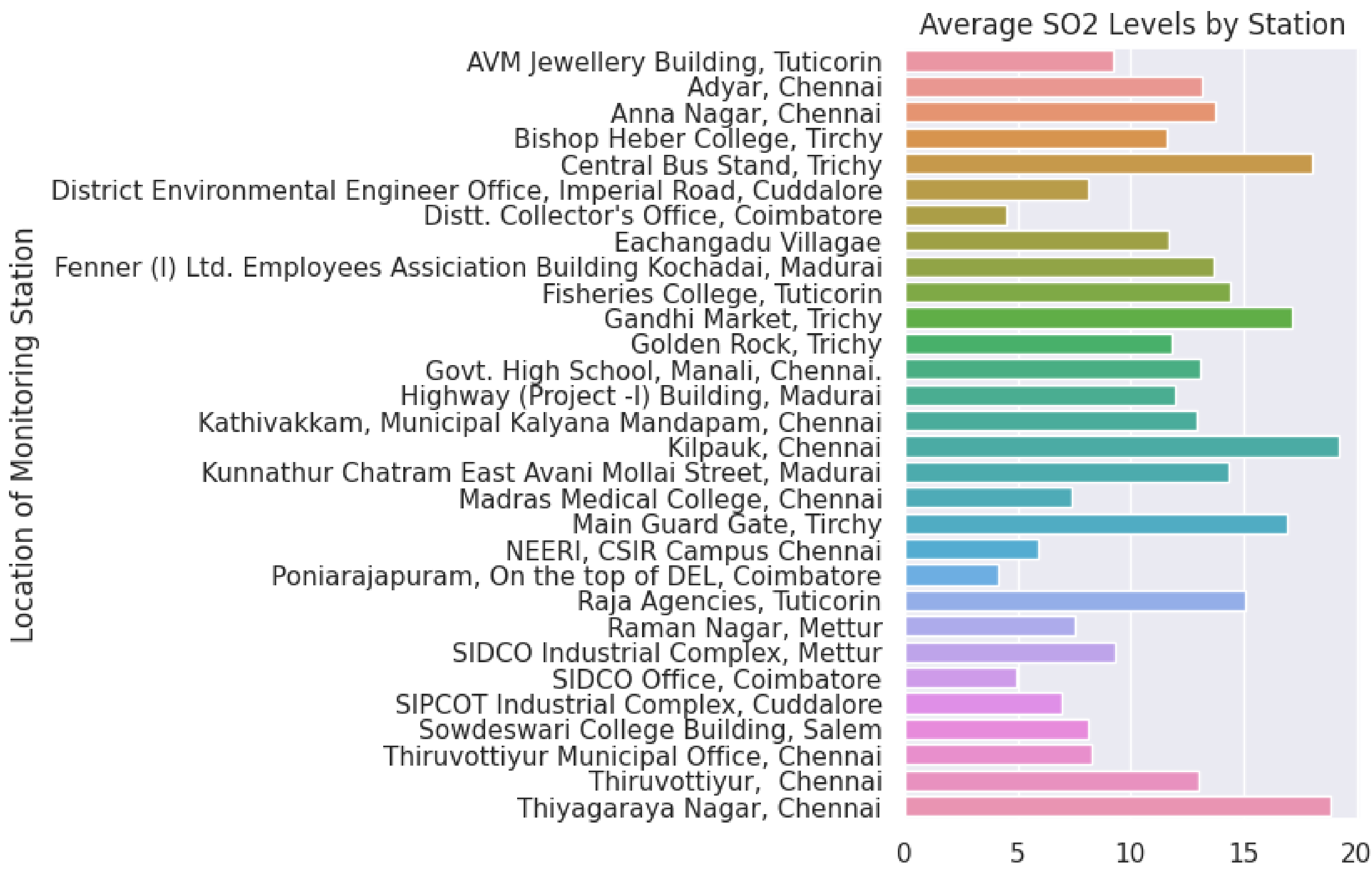
Step 9: Create Visualizations

Use data visualization libraries such as Matplotlib and Seaborn to create visualizations that help convey your findings.To create bar plots to visualize the average pollution levels:

Input:

```
# Create bar plots for average pollution levels
plt.figure(figsize=(12, 6))
plt.subplot(131)
sns.barplot(x=avg_so2.values, y=avg_so2.index)
plt.title('Average SO2 Levels by Station')
```

Output:



Input:

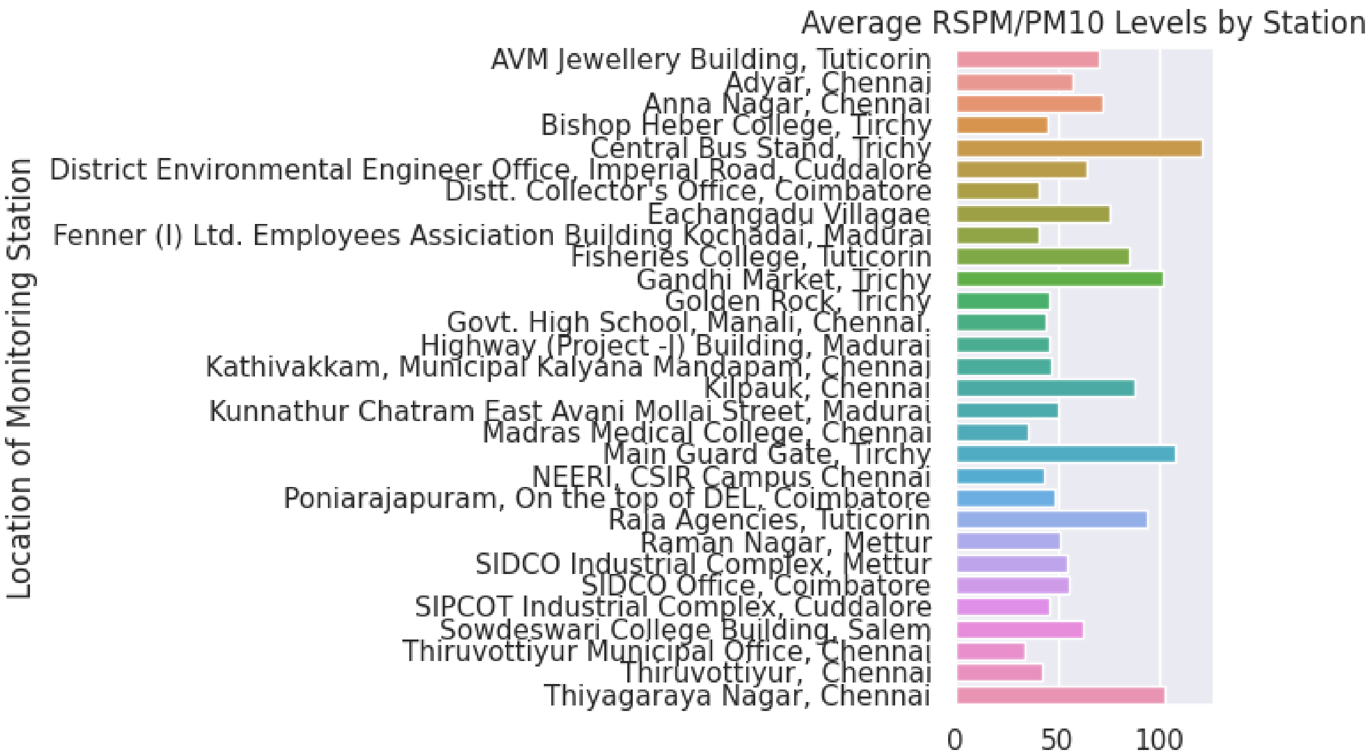
```
plt.subplot(132)
sns.barplot(x=avg_no2.values, y=avg_no2.index)
plt.title('Average NO2 Levels by Station')
```

Output:

Input:

```
plt.subplot(133)
sns.barplot(x=avg_rspm_pm10.values, y=avg_rspm_pm10.index)
plt.title('Average RSPM/PM10 Levels by Station')
plt.tight_layout()
plt.show()
```

Output:



Step 10: Visualization based on City/Town/Village/Area and calculate average levels

Input:

```
# Group by City/Town/Village/Area  and calculate average
levels
avg_so2 = df.groupby('City/Town/Village/Area')['SO2'].mean()
avg_no2 = df.groupby('City/Town/Village/Area')['NO2'].mean()
avg_rspm_pm10 =
df.groupby('City/Town/Village/Area')['RSPM/PM10'].mean()
```

Step 11: Identify Pollution Trends

Analyze the calculated averages to identify pollution trends and areas with high pollution levels. Here's how you can find the stations with the highest average levels:

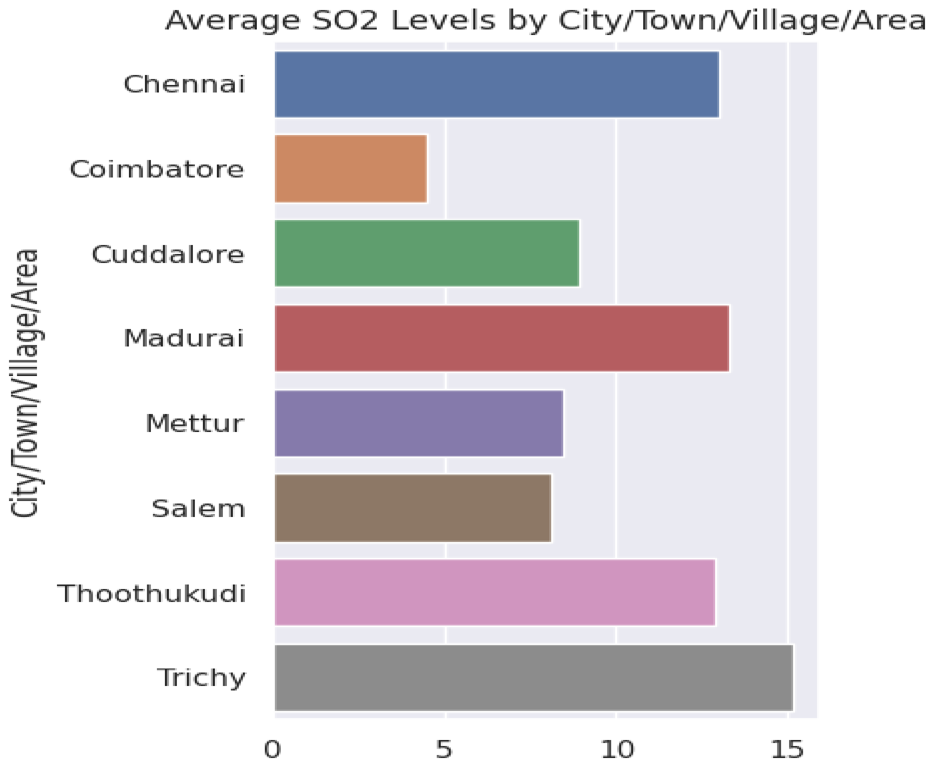
Input:

```
# Find the City/Town/Village/Area  with the highest average
SO2 level
highest_so2_station = avg_so2.idxmax()
highest_so2_value = avg_so2.max()

# Find the City/Town/Village/Area  with the highest average
NO2 level
highest_no2_station = avg_no2.idxmax()
highest_no2_value = avg_no2.max()

# Find the City/Town/Village/Area  with the highest average
RSPM/PM10 level
highest_rspm_pm10_station = avg_rspm_pm10.idxmax()
highest_rspm_pm10_value = avg_rspm_pm10.max()
```

Input:

```
# Create bar plots for average pollution levels
plt.figure(figsize=(12, 6))
plt.subplot(131)
sns.barplot(x=avg_so2.values, y=avg_so2.index)
plt.title('Average SO2 Levels by City/Town/Village/Area')
```

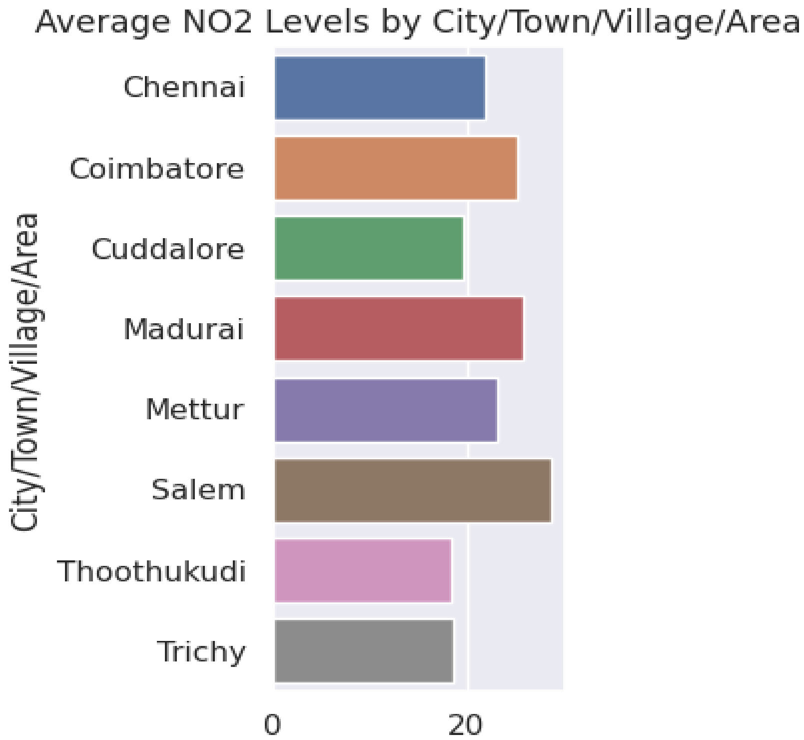
Output:



Input:

```
plt.subplot(132)
sns.barplot(x=avg_no2.values, y=avg_no2.index)
plt.title('Average NO2 Levels by City/Town/Village/Area')
```

Output:

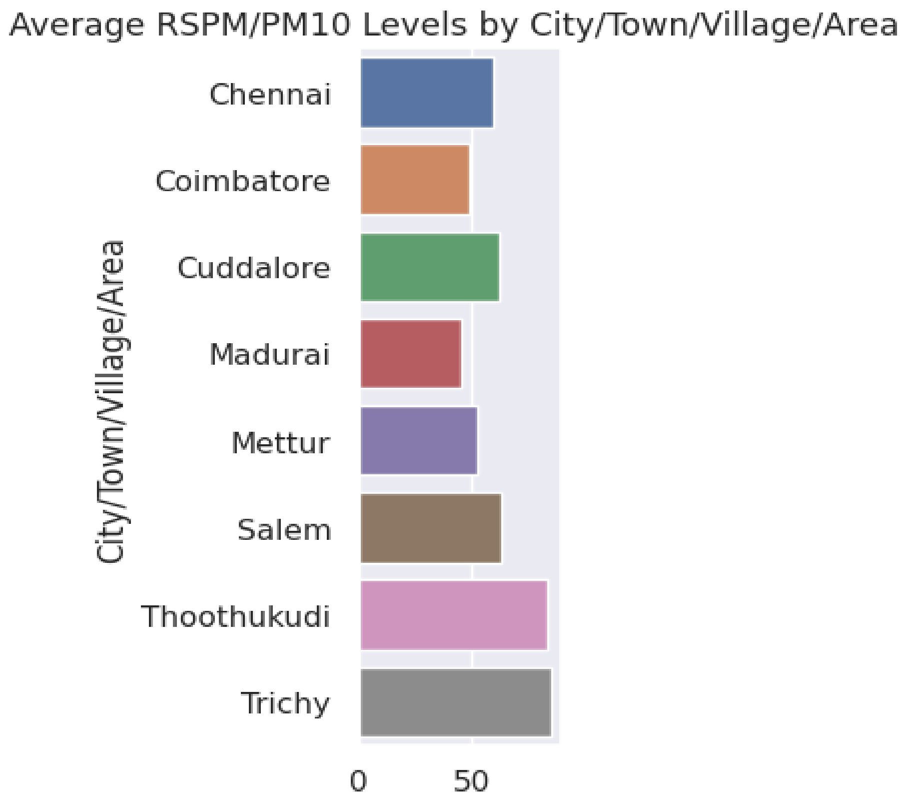


Input:

```
plt.subplot(133)
```

```
sns.barplot(x=avg_rspm_pm10.values, y=avg_rspm_pm10.index)
plt.title('Average RSPM/PM10 Levels by City/Town/Village/Area
')
plt.tight_layout()
plt.show()
```

Output:



Step 11: Save the Visualizations:

If you want to save the visualizations as image files, you can use plt.savefig('visualization.png') after creating the plot.

Replicating the Analysis:

To replicate the analysis, follow these steps:

- Download the dataset from the provided link.
- Install Python and the Pandas library if not already installed: pip install pandas.
- Load the dataset into a Pandas DataFrame using the provided code.
- Preprocess the data as needed.

- Follow the code to create visualizations and calculate average pollution levels.
- Analyze the findings to gain insights into air pollution trends and pollution levels.

Key Findings Of Project:

The code provides a visual overview of the data and initial insights into air quality in Tamil Nadu in 2014.

The highest average pollution levels and corresponding monitoring stations or areas are identified for SO2, NO2, and RSPM/PM10.

CONCLUSION:

By combining user-centric data collection, cutting-edge technology, and policy advocacy, this innovative approach aims to create a sustainable and holistic solution for addressing air quality in Tamil Nadu. It leverages the power of AI, IoT, and blockchain to engage the community, businesses, and policymakers in a collective effort to combat air pollution and improve the quality of life in the region.