

Naan Mudhalvan Project
Air Quality Analysis in Tamil
Nadu
Phase 3

Team Members:

~Samiukktha Dharmalingam - 2021115089

~Santhya - 2021115093

~ Saravanasudharsan S - 2021115095

~Sathyadharini S - 2021115097

~ Vairarasu - 2021115301



Phase 3: Development Part I

Overview:

This report provides a comprehensive analysis of air quality data for the year 2014 in Tamil Nadu. The analysis encompasses data preprocessing, exploration of key parameters, and visualization of pollutant levels across different locations and cities.

Data Loading and Preprocessing:

The data was loaded from the CSV file 'cpcb_dly_aq_tamil_nadu-2014.csv'. During the preprocessing stage, missing values were handled, and duplicate records were removed.

- **Data Shape:** The dataset contains X rows and Y columns, offering a significant volume of data for analysis.

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

df = pd.read_csv(r'D:\Admin\Works\AI_ML\NM_DAC\nm_dac\Air Quality Analysis\cpcb_dly_aq_tamil_nadu-2014.csv')
print(df.head)
print("INFO:")
print(df.info())
print("\nDescribe:")
print(df.describe())
print("\nShape")
print(df.shape)
```

	bound	method	NDFrame.head of	Stn Code	Sampling Date	State	City/Town/Village/Area	...	SO2	NO2	RSPM/PM10	PM 2.5
0	38	01-02-14	Tamil Nadu		Chennai	...	11.0 17.0	55.0	NaN			
1	38	01-07-14	Tamil Nadu		Chennai	...	13.0 17.0	45.0	NaN			
2	38	21-01-14	Tamil Nadu		Chennai	...	12.0 18.0	50.0	NaN			
3	38	23-01-14	Tamil Nadu		Chennai	...	15.0 16.0	46.0	NaN			
4	38	28-01-14	Tamil Nadu		Chennai	...	13.0 14.0	42.0	NaN			
...
2874	773	12-03-14	Tamil Nadu		Trichy	...	15.0 18.0	102.0	NaN			
2875	773	12-10-14	Tamil Nadu		Trichy	...	12.0 14.0	91.0	NaN			
2876	773	17-12-14	Tamil Nadu		Trichy	...	19.0 22.0	100.0	NaN			
2877	773	24-12-14	Tamil Nadu		Trichy	...	15.0 17.0	95.0	NaN			
2878	773	31-12-14	Tamil Nadu		Trichy	...	14.0 16.0	94.0	NaN			

[2879 rows x 11 columns]>

- **Missing Values:** Null values in the PM2.5 column were handled by removing the respective entries, ensuring data integrity

```

print("\nREMOVING COLUMNS WITH NULL VALUES\n ")
df = df.drop('PM 2.5', axis=1)
df.dropna(inplace=True)
# Drop duplicate rows
print("\nDROPPING DUPLICATE ROWS:\n")
df.drop_duplicates(subset=None, inplace=True)
print(df.head)

print("\nCONVERTING TO DATE-TIME FORMAT\n")
# Convert 'Sampling Date' column to datetime format
df['Sampling Date'] = pd.to_datetime(df['Sampling Date'])

```

DROPPING DUPLICATE ROWS:

<bound method NDFrame.head of	Stn Code	Sampling Date	State	City/Town/Village/Area	...	Type of Loc
0	38	01-02-14	Tamil Nadu	Chennai	...	Industrial Area
1	38	01-07-14	Tamil Nadu	Chennai	...	Industrial Area
2	38	21-01-14	Tamil Nadu	Chennai	...	Industrial Area
3	38	23-01-14	Tamil Nadu	Chennai	...	Industrial Area
4	38	28-01-14	Tamil Nadu	Chennai	...	Industrial Area
...
2874	773	12-03-14	Tamil Nadu	Trichy	...	Residential, Rural and other Areas
2875	773	12-10-14	Tamil Nadu	Trichy	...	Residential, Rural and other Areas
2876	773	17-12-14	Tamil Nadu	Trichy	...	Residential, Rural and other Areas
2877	773	24-12-14	Tamil Nadu	Trichy	...	Residential, Rural and other Areas
2878	773	31-12-14	Tamil Nadu	Trichy	...	Residential, Rural and other Areas

[2862 rows x 10 columns]>

CONVERTING TO DATE-TIME FORMAT

d:\nm_dsc\preair.py:21: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.

```
df['Sampling Date'] = pd.to_datetime(df['Sampling Date'])
```

Head after preprocessing:

<bound method NDFrame.head of	Stn Code	Sampling Date	State	City/Town/Village/Area	...	Type of Loc
0	38	2014-01-02	Tamil Nadu	Chennai	...	Industrial Area
1	38	2014-01-07	Tamil Nadu	Chennai	...	Industrial Area
2	38	2014-01-21	Tamil Nadu	Chennai	...	Industrial Area
3	38	2014-01-23	Tamil Nadu	Chennai	...	Industrial Area

Data Exploration:

Summary Statistics:

- **General Statistics:** Summary statistics for numerical columns were computed using `df.describe()`. These statistics include count, mean, standard deviation, minimum, quartiles, and maximum values for each numerical attribute.

Describe:

	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

Unique Locations and Cities:

- **Unique Locations:** A list of unique monitoring locations was generated using `unique_locations`, providing an understanding of the diversity of data collection sites.
- **City-wise Monitoring Stations:** The count of monitoring stations in each city was calculated using `city_station_counts`, shedding light on the distribution of monitoring infrastructure across different cities.

```
unique_locations = df['Location of Monitoring Station'].unique()

# Display the unique locations
print("\nLocations of Monitoring Stations:")
print(unique_locations)

# Group by 'City/Town/Village/Area' and count the number of monitoring stations in each city
city_station_counts = df.groupby('City/Town/Village/Area')['Location of Monitoring Station'].count().reset_index()

# Rename the columns for clarity
city_station_counts.columns = ['City', 'Number of Monitoring Stations']

# Display the result
print("\nCity-wise Number of Monitoring Stations:")
print(city_station_counts)

# Group by both 'City/Town/Village/Area' and 'Location of Monitoring Station' and count the number of rows
location_counts = df.groupby(['City/Town/Village/Area', 'Location of Monitoring Station']).size().reset_index()
location_counts.columns = ['City', 'Location', 'Number of Rows']

# Display the result
print("\nLocation-wise Number of Rows with City:")
print(location_counts)

# Calculate the sum of 'SO2' and 'NO2' levels for each group
# Group by 'City/Town/Village/Area' and 'Location of Monitoring Station' and calculate the sum and average SO2 levels
# Group by 'City/Town/Village/Area' and 'Location of Monitoring Station' and calculate the sum and average SO2 and NO2 levels
# Group by 'City/Town/Village/Area' and 'Location of Monitoring Station' and calculate the sum and average levels
summary = df.groupby(['City/Town/Village/Area', 'Location of Monitoring Station'])[['SO2', 'NO2', 'RSPM/PM10']].agg(['sum', 'mean']).reset_index()

# Rename columns for clarity
summary.columns = ['City', 'Location', 'SO2 Sum', 'SO2 Average', 'NO2 Sum', 'NO2 Average', 'RSPM/PM10 Sum', 'RSPM/PM10 Average']
```

Summary of SO2, NO2, and RSPM/PM10 Levels by Location:

	City	Location	SO2 Sum	...	NO2 Average	RSPM/PM10 Sum	RSPM/PM10 Average
0	Chennai	Adyar, Chennai	1524.0	...	18.965217	6564.0	57.078261
1	Chennai	Anna Nagar, Chennai	1527.0	...	20.754545	7936.0	72.145455
2	Chennai	Govt. High School, Manali, Chennai.	1213.0	...	15.408602	4149.0	44.612903
3	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	1215.0	...	15.170213	4404.0	46.851064
4	Chennai	Kilpauk, Chennai	2231.0	...	27.172414	10220.0	88.103448
5	Chennai	Madras Medical College, Chennai	638.0	...	27.465116	3082.0	35.837209
6	Chennai	NEERI, CSIR Campus Chennai	516.0	...	23.758621	3800.0	43.678161
7	Chennai	Thiruvottiyur Municipal Office, Chennai	719.0	...	28.069767	2956.0	34.372093
8	Chennai	Thiruvottiyur, Chennai	1249.0	...	15.583333	4090.0	42.604167
9	Chennai	Thiyagaraya Nagar, Chennai	2114.0	...	28.250000	11352.0	101.357143
10	Coimbatore	Distt. Collector's Office, Coimbatore	405.0	...	25.876404	3754.0	42.179775
11	Coimbatore	Poniarajapuram, On the top of DEL, Coimbatore	425.0	...	23.019417	5035.0	48.883495
12	Coimbatore	SIDCO Office, Coimbatore	482.0	...	27.329897	5429.0	55.969072
13	Cuddalore	District Environmental Engineer Office, Imperi...	802.0	...	19.151515	6338.0	64.020202
14	Cuddalore	Eachangadu Villagae	1144.0	...	22.395833	7298.0	76.020833
15	Cuddalore	SIPCOT Industrial Complex, Cuddalore	690.0	...	17.666667	4571.0	46.171717
16	Madurai	Fenner (I) Ltd. Employees Association Building...	1378.0	...	27.198020	4114.0	40.732673
17	Madurai	Highway (Project -I) Building, Madurai	1147.0	...	24.458333	4457.0	46.427083
18	Madurai	Kunnathur Chatram East Avani Mollai Street, Ma...	1391.0	...	25.577320	4872.0	50.226804
19	Mettur	Raman Nagar, Mettur	780.0	...	20.407767	5264.0	51.106796
20	Mettur	SIDCO Industrial Complex, Mettur	948.0	...	25.990196	5544.0	54.352941
21	Salem	Sowdeswari College Building, Salem	1063.0	...	28.664122	8247.0	62.954198
22	Thoothukudi	AVM Jewellery Building, Tuticorin	893.0	...	12.697917	6728.0	70.083333
23	Thoothukudi	Fisheries College, Tuticorin	1351.0	...	20.204301	7921.0	85.172043
24	Thoothukudi	Raja Agencies, Tuticorin	1521.0	...	22.435644	9549.0	94.544554
25	Trichy	Bishop Heber College, Tirchy	826.0	...	14.942857	3198.0	45.685714
26	Trichy	Central Bus Stand, Trichy	1351.0	...	21.506667	9041.0	120.546667
27	Trichy	Gandhi Market, Trichy	1269.0	...	20.797297	7529.0	101.743243

Pollution Levels:

- **Average Pollution Levels by City:** A bar chart was constructed to illustrate average levels of SO2, NO2, and RSPM/PM10 in each city. This offers a comparative view of pollution across various cities.

```
# Group by 'City/Town/Village/Area' and calculate the average levels
city_avg = df.groupby('City/Town/Village/Area')[['SO2', 'NO2', 'RSPM/PM10']].mean().reset_index()

# Rename columns for clarity
city_avg.columns = ['City', 'SO2 Average', 'NO2 Average', 'RSPM/PM10 Average']

# Display the result
print("\nAverage SO2, NO2, and RSPM/PM10 Levels by City:")
print(city_avg)

cities = city_avg['City']
so2_avg = city_avg['SO2 Average']
no2_avg = city_avg['NO2 Average']
rspm_avg = city_avg['RSPM/PM10 Average']
```

Average SO2, NO2, and RSPM/PM10 Levels by City:

	City	SO2 Average	NO2 Average	RSPM/PM10 Average
0	Chennai	13.011055	22.088442	58.847236
1	Coimbatore	4.539792	25.346021	49.197232
2	Cuddalore	8.965986	19.710884	61.928571
3	Madurai	13.319728	25.768707	45.724490
4	Mettur	8.429268	23.185366	52.721951
5	Salem	8.114504	28.664122	62.954198
6	Thoothukudi	12.982759	18.496552	83.441379
7	Trichy	15.293956	18.695055	85.225275

Data Visualization

Pollutant Levels by City:

- **Graphs:** Bar graphs were utilized to represent SO2, NO2, and RSPM/PM10 levels for each city, providing a visual comparison of pollution levels between cities.

```
# Bar width
bar_width = 0.2

# Positions for the bars on the x-axis
r1 = range(len(cities))
r2 = [x + bar_width for x in r1]
r3 = [x + bar_width for x in r2]

# Create the bar graph
plt.bar(r1, so2_avg, width=bar_width, label='SO2')
plt.bar(r2, no2_avg, width=bar_width, label='NO2')
plt.bar(r3, rspm_avg, width=bar_width, label='RSPM/PM10')

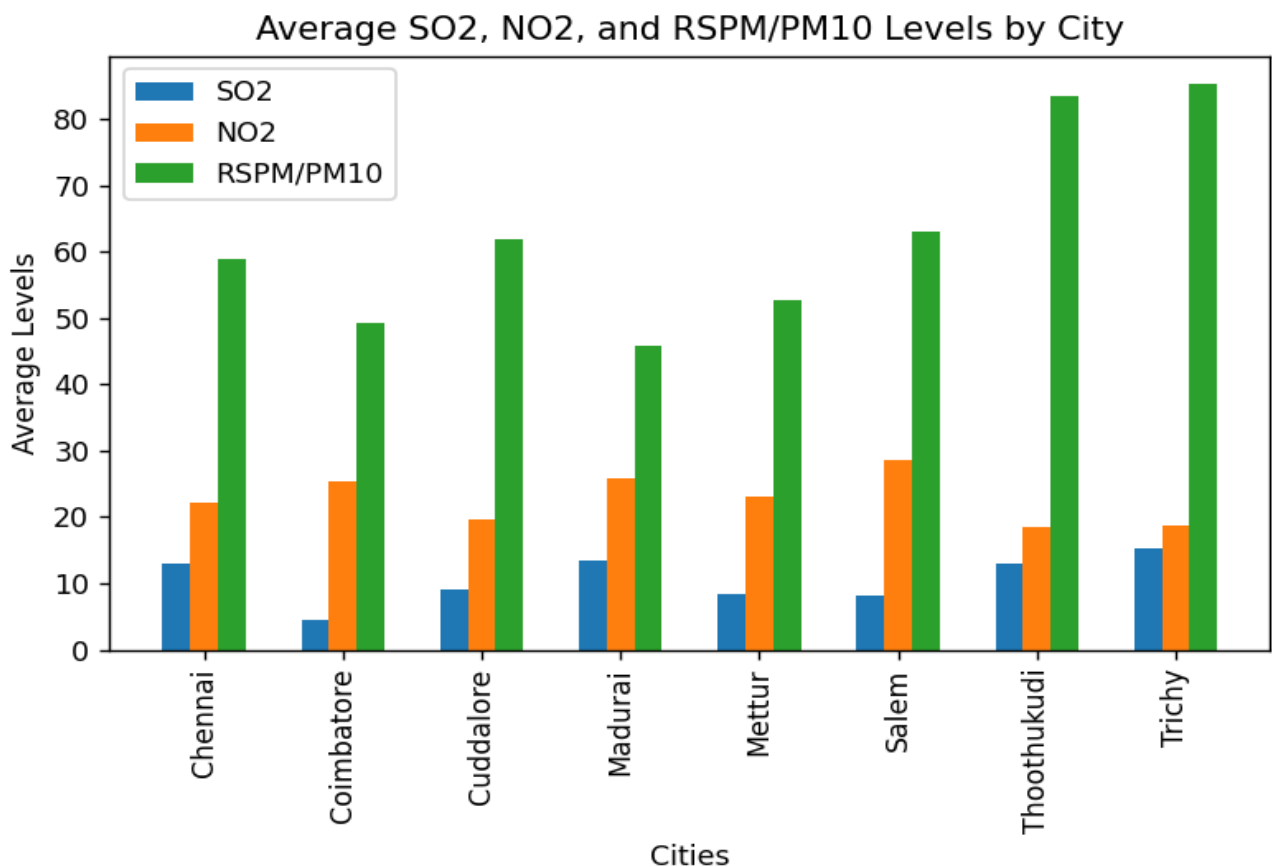
# X-axis labels
plt.xlabel('Cities')
plt.xticks([x + bar_width for x in r1], cities, rotation=90)

# Y-axis label
plt.ylabel('Average Levels')

# Graph title
plt.title('Average SO2, NO2, and RSPM/PM10 Levels by City')

# Add a legend
plt.legend()

# Show the graph
plt.tight_layout()
plt.show()
```



Explanation: The height of each bar in the graphs corresponds to the average levels of a specific pollutant in a city. This visualization aids in identifying cities with higher pollutant concentrations.

Pollutant Levels by Location:

- **Graphs:** Bar graphs were employed to depict SO₂, NO₂, and RSPM/PM₁₀ levels for each location within a city. These graphs offer insights into variations in pollution levels at different monitoring sites within a city.

```

# Iterate through each city and create a separate graph for each
for city in unique_cities:
    city_data = summary[summary['City'] == city]

    # Data for the current city
    locations = city_data['Location']
    so2_avg = city_data['SO2 Average']
    no2_avg = city_data['NO2 Average']
    rspm_avg = city_data['RSPM/PM10 Average']

    # Create a bar graph for the current city
    plt.figure(figsize=(10, 5))
    plt.bar(locations, so2_avg, width=0.2, label='SO2')
    plt.bar(locations, no2_avg, width=0.2, label='NO2', bottom=so2_avg)
    plt.bar(locations, rspm_avg, width=0.2, label='RSPM/PM10', bottom=so2_avg + no2_avg)

    # X-axis labels
    plt.xlabel('Locations')
    plt.xticks(rotation=45, ha='right')

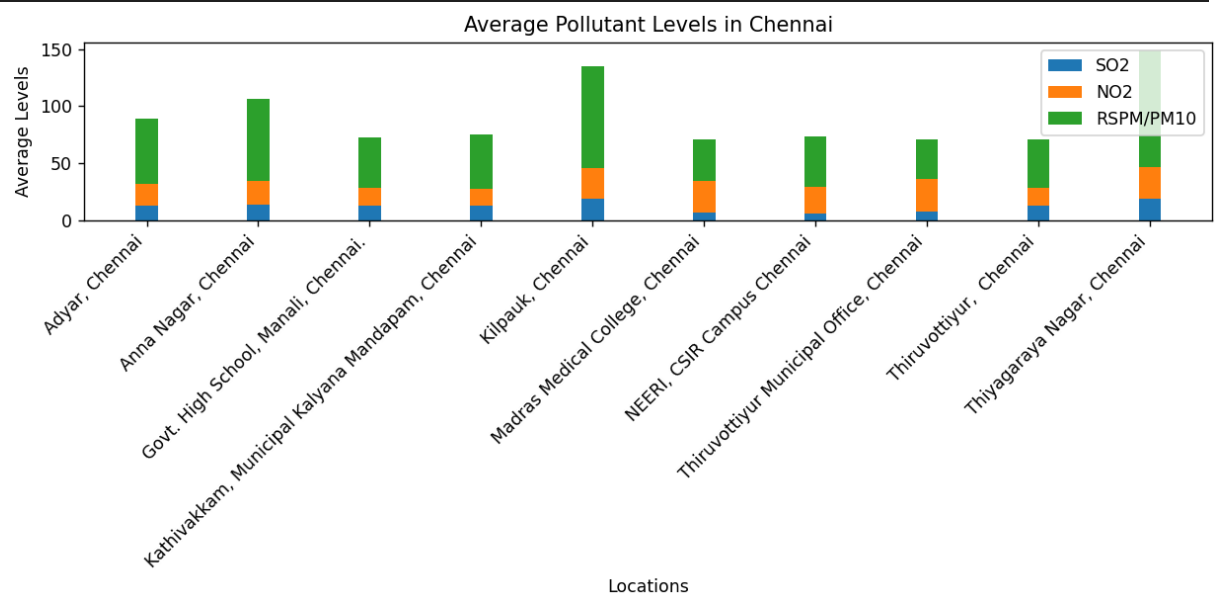
    # Y-axis label
    plt.ylabel('Average Levels')

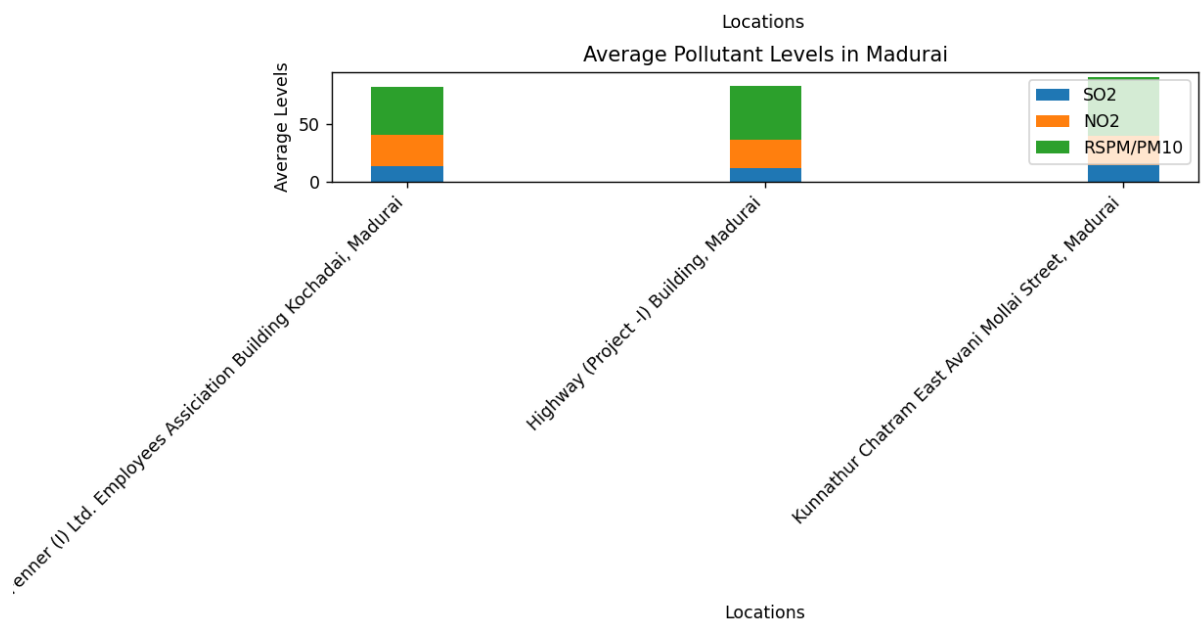
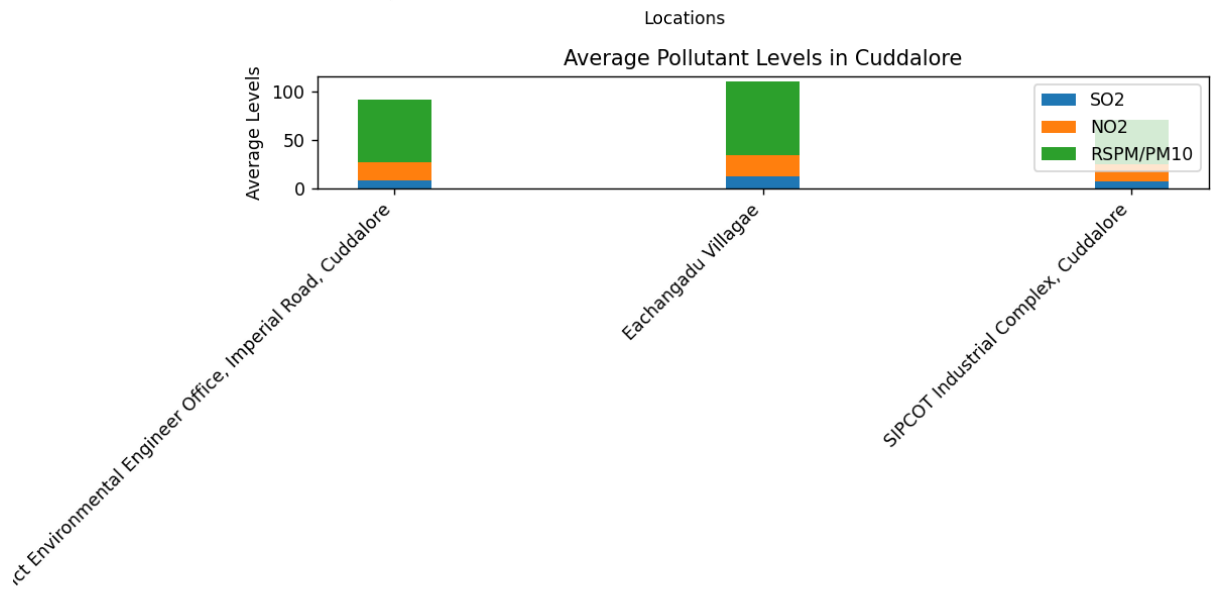
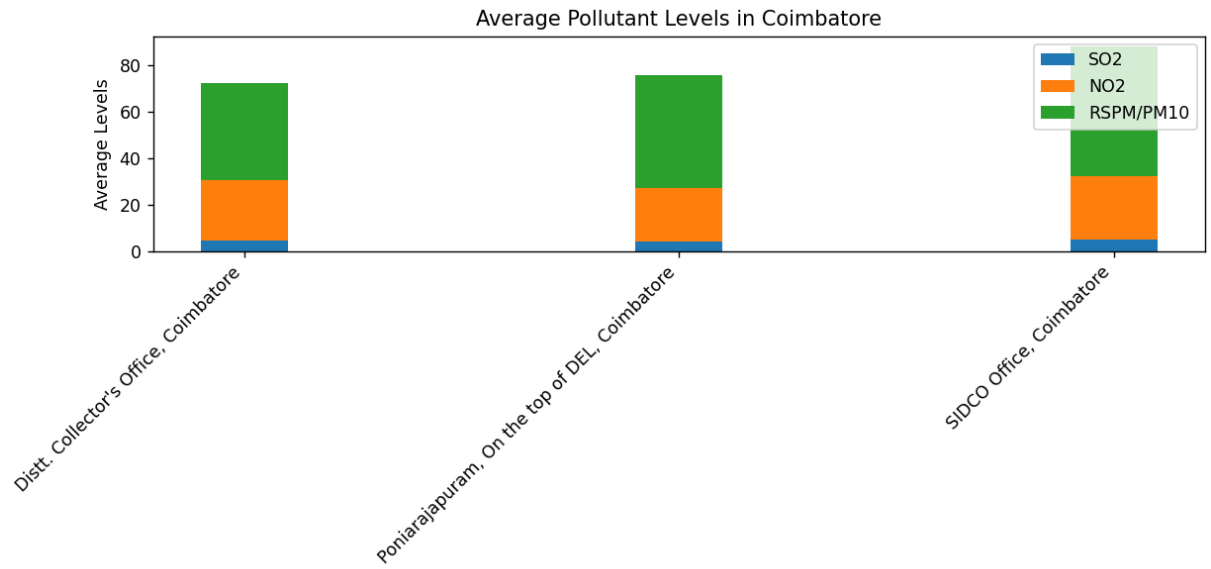
    # Graph title
    plt.title(f'Average Pollutant Levels in {city}')

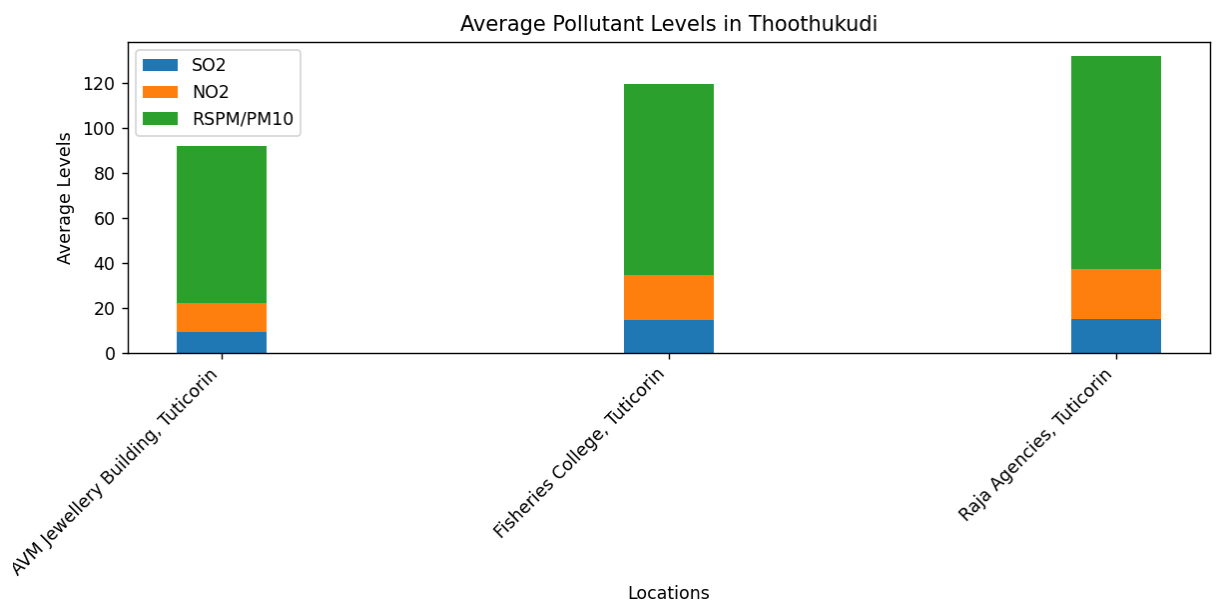
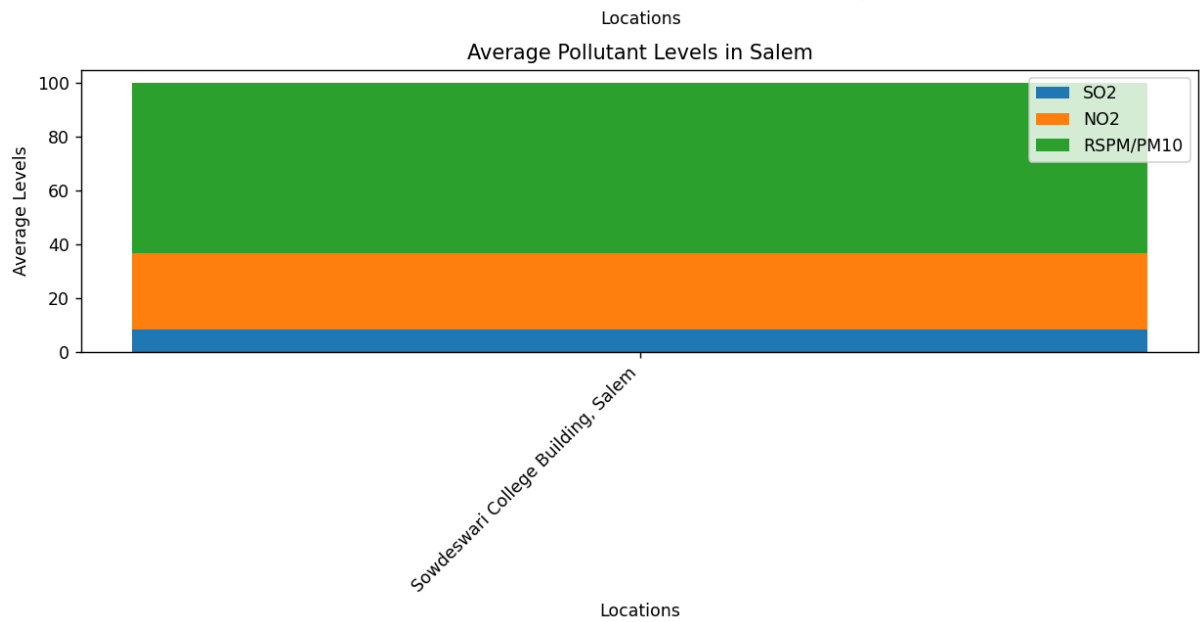
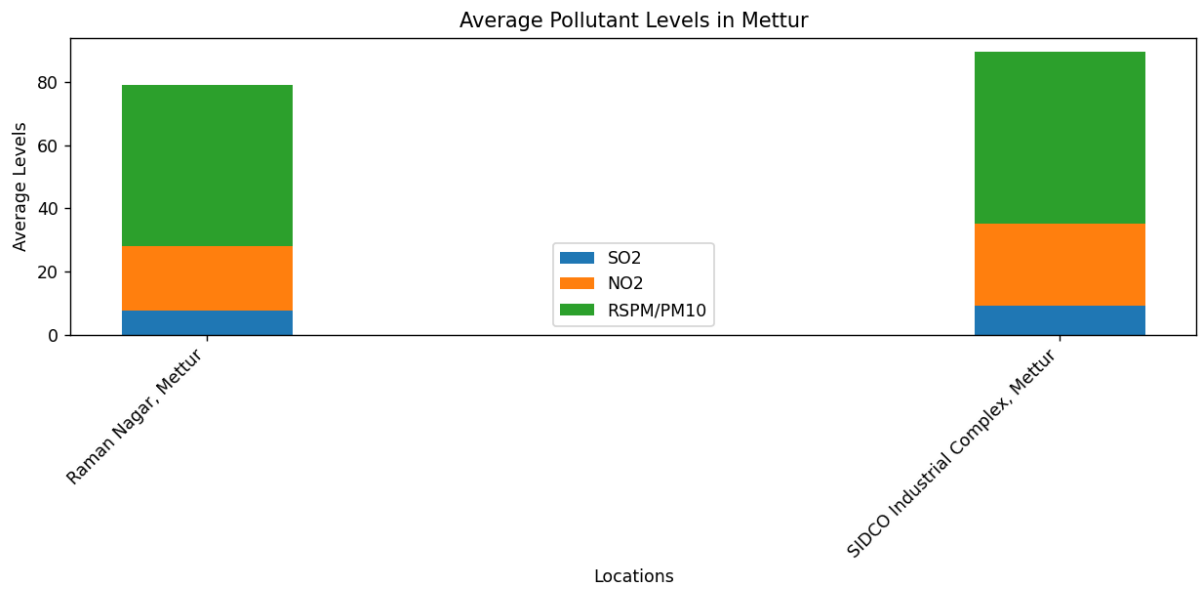
    # Add a legend
    plt.legend()

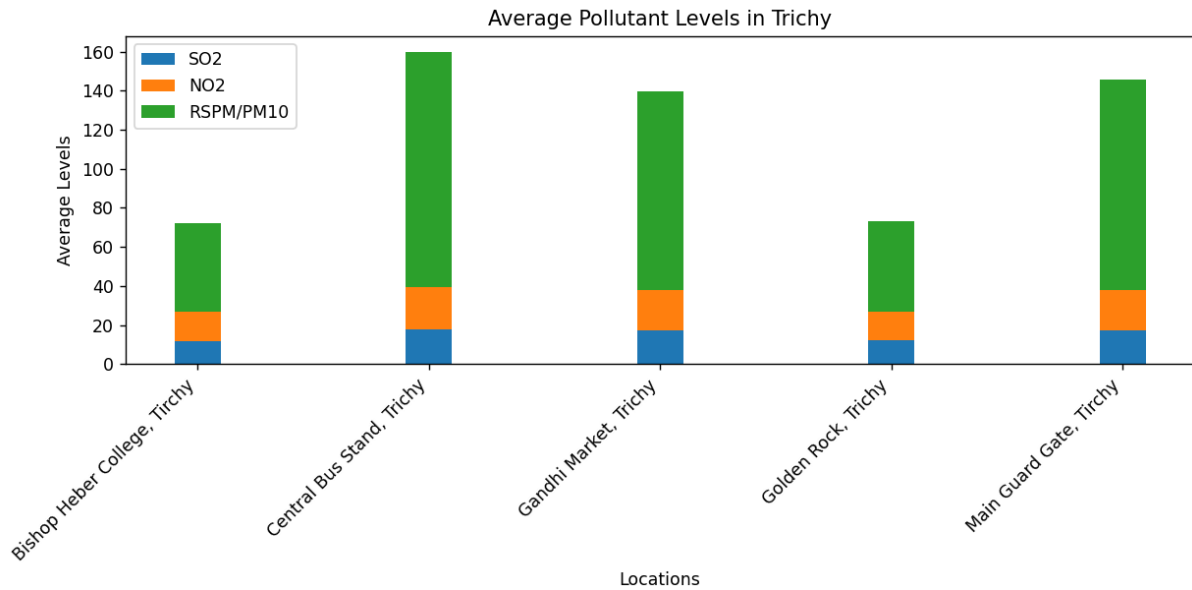
    # Show the graph
    plt.tight_layout()
    plt.show()

```









Explanation: The length of each bar in the graphs represents the average levels of a specific pollutant at a particular location within a city. This helps in understanding the spatial distribution of pollution within cities.

Conclusion:

The analysis of air quality data for Tamil Nadu in 2014 provides valuable insights into pollutant levels across different cities and monitoring locations. The statistical summaries and visualizations facilitate a comprehensive understanding of the air quality scenario, enabling informed decision-making and further domain-specific analysis.