Naan Mudhalvan Project

Air Quality Analysis in Tamil Nadu

Phase 5

Team Members:

~Samiukktha Dharmalingam - 2021115089

~Santhya - 2021115093

~ Saravanasudharsan S - 2021115095

~Sathyadharini S - 2021115097

~ Vairaarasu - 2021115301

Phase 5: Project Documentation & Submission

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.

Indian AQI:

The Indian AQI range differs from that of US-EPA. To calculate AQI, a minimum of three parameters should be taken out of which one must be either PM10 or PM2.5.

According to the Indian Government (CPCB), Indian AQI range is from 0-500, from 0 being good and 500 being severe. There are eight major pollutants to be taken into account for AQI calculation, viz. particulate matter (PM 10 and PM 2.5), carbon monoxide (CO), ozone (O3), nitrogen dioxide (NO2), sulfur dioxide (SO2), ammonia (NH3), and lead (Pb). To calculate AQI, data for a minimum of three pollutants must be present, of which one should be either PM10 or PM2.5, AQI ranging from 0-500 has different concentrations for each pollutant and has health effects accordingly.

Indian AQI range & probable impacts:

0-50: This range defines air quality as good as it shows minimal or no impact on health.

51-100: This is a satisfactory air quality range and it can show effects such as breathing difficulty in sensitive groups.

101-200: The range shows moderate air quality with impacts such as breathing discomfort for children and elderly people, and people already suffering from lung disorders and heart disease.

201-300: AQI falling in this range communicates that the air quality is poor and shows health effects on people when exposed for the long term. People already suffering from heart diseases can experience discomfort from short exposure.

301-400: This range shows very poor air quality and causes respiratory illness for a longer duration of exposure.

401-500: This is the severe range of AQI causing health impacts to normal and diseased people. It also causes severe health impacts on sensitive groups.

Calculation of AQI:

To calculate sub-indices, 16 hours of data is needed.

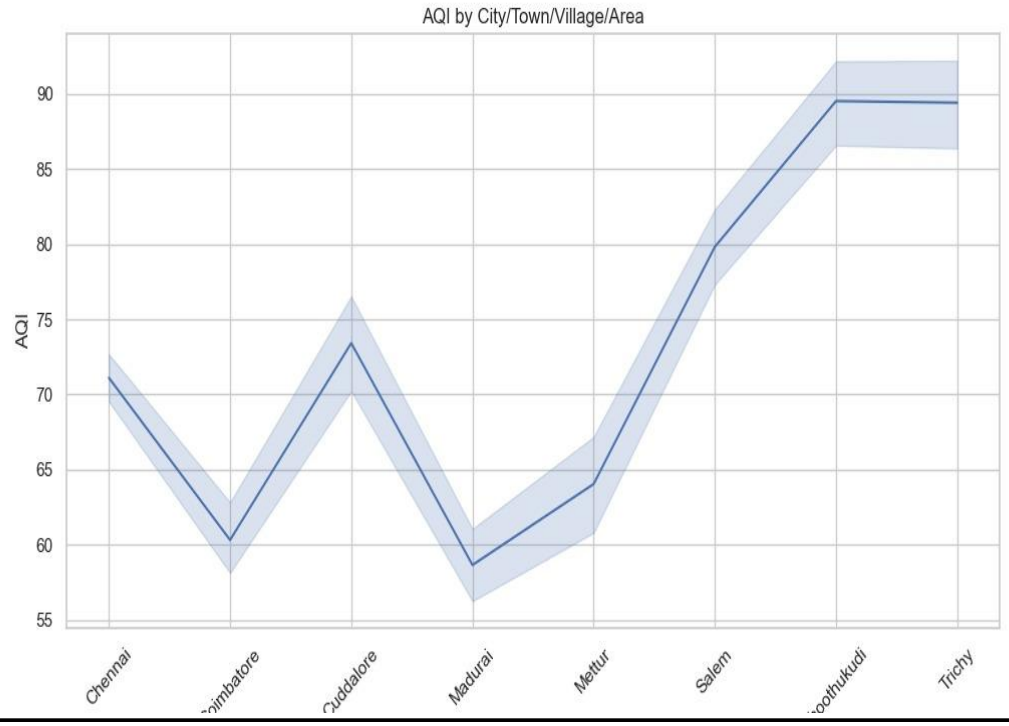
For example: If you wish to calculate AQI on the basis of PM2.5, CO, and ozone, calculate the sub-index for each parameter separately.

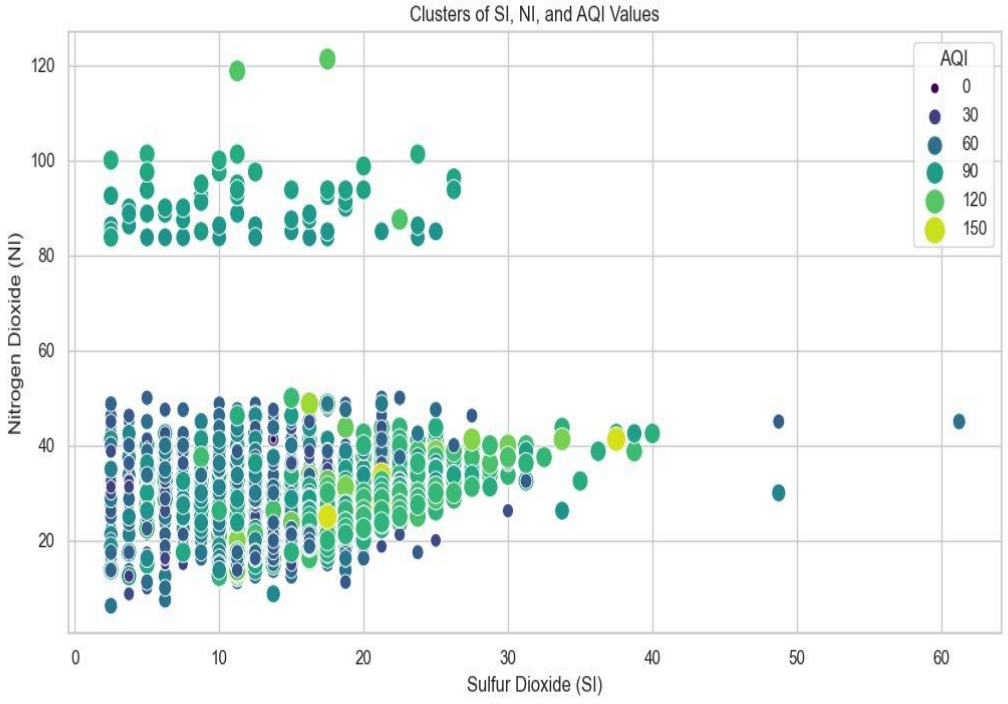
If the current concentration of PM2.5 is 110 ug/m3, then referring to AQI range as per Indian standards BPHi = 120, BPLo = 91, IHi = 300 and ILo = 201.

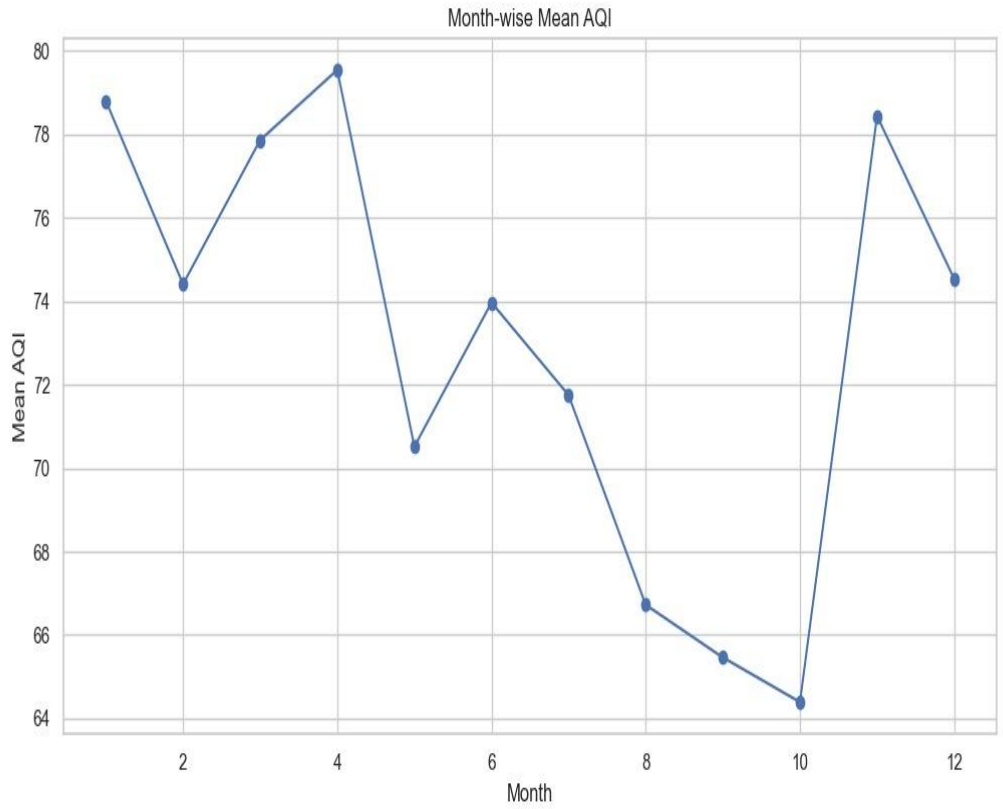
Putting the values in equation and solving:

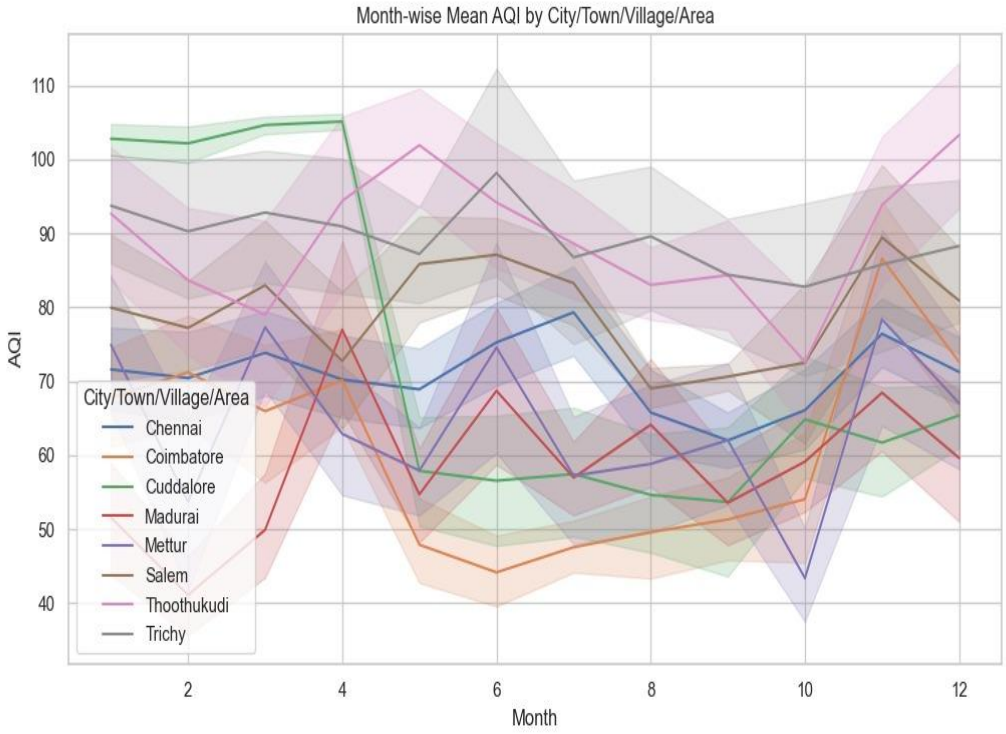
Sub Index= [(300-201)/ (120-91)] (110-91) + 201 = 265.86

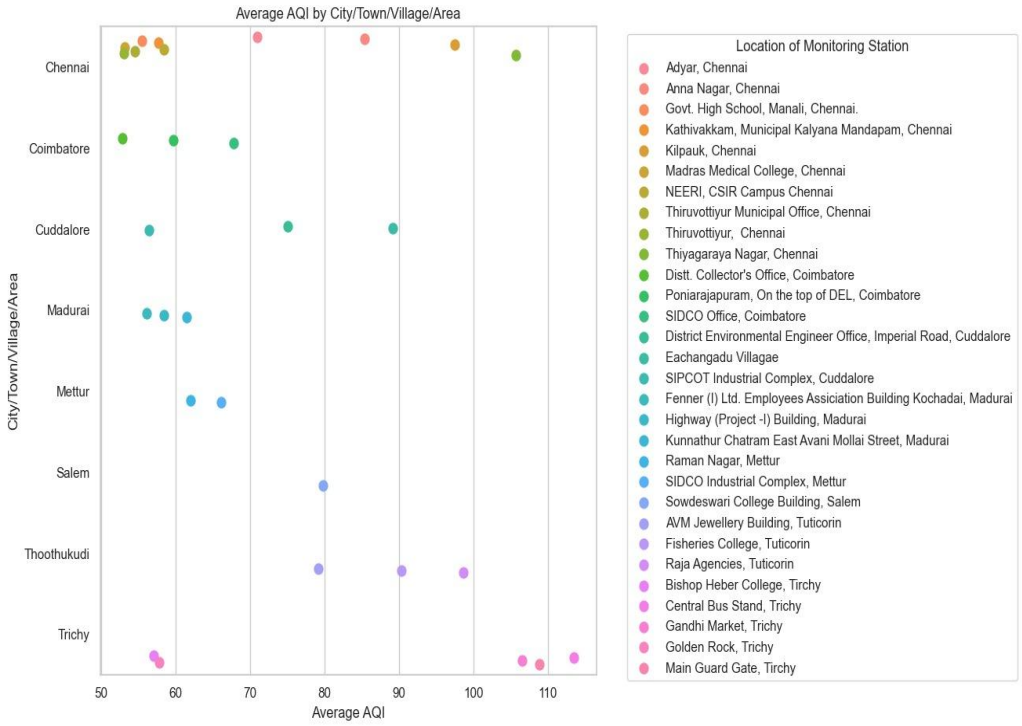
RESULTS:

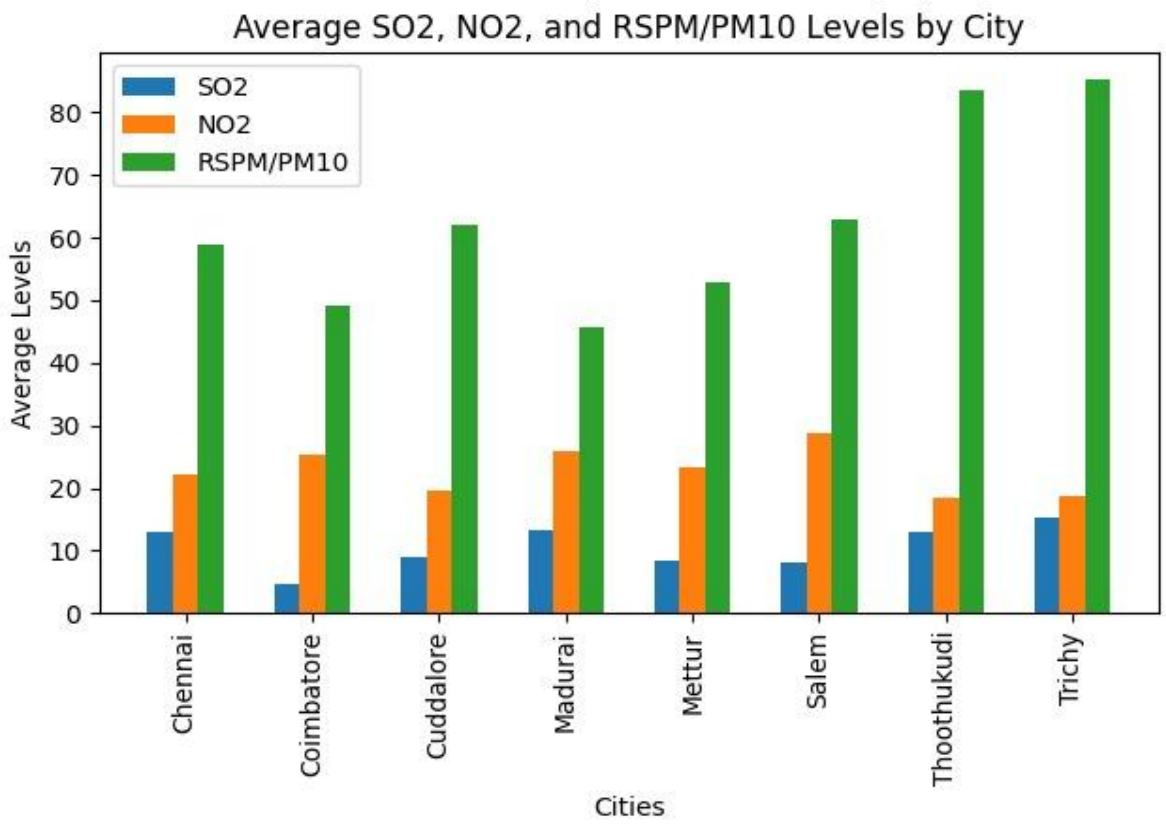

AQI by City/Town/Village/Area
AQI
90
85
80
75
70
65
60
55
Chennai
Coimbatore
Cuddalore
Madurai
Mettur
Salem
Thoothukudi
Trichy


Clusters of SI, NI, and AQI Values
Nitrogen Dioxide (NI)
Sulfur Dioxide (SI)
AQI
0
30
60
90
120
150
120
100
80
60
40
20
0
10
20
30
40
50
60


Month-wise Mean AQI
Mean AQI
Month
80
78
76
74
72
70
68
66
64
2
4
6
8
10
12


Month-wise Mean AQI by City/Town/Village/Area
AQI
Month
City/Town/Village/Area
Chennai
Coimbatore
Cuddalore
Madurai
Mettur
Salem
Thoothukudi
Trichy


Average AQI by City/Town/Village/Area
City/Town/Village/Area
Average AQI
Location of Monitoring Station
Adyar, Chennai
Anna Nagar, Chennai
Govt. High School, Manali, Chennai.
Kathivakkam, Municipal Kalyana Mandapam, Chennai
Kilpauk, Chennai
Madras Medical College, Chennai
NEERI, CSIR Campus Chennai
Thiruvottiyur Municipal Office, Chennai
Thiruvottiyur, Chennai
Thiyagaraya Nagar, Chennai
Distt. Collector's Office, Coimbatore
Poniarajapuram, On the top of DEL, Coimbatore
SIDCO Office, Coimbatore
District Environmental Engineer Office, Imperial Road, Cuddalore
Eachangadu Villagae
SIPCOT Industrial Complex, Cuddalore
Fenner (I) Ltd. Employees Assiciation Building Kochadai, Madurai
Highway (Project -I) Building, Madurai
Kunnathur Chatram East Avani Mollai Street, Madurai
Raman Nagar, Mettur
SIDCO Industrial Complex, Mettur
Sowdeswari College Building, Salem
AVM Jewellery Building, Tuticorin
Fisheries College, Tuticorin
Raja Agencies, Tuticorin
Bishop Heber College, Tirchy
Central Bus Stand, Trichy
Gandhi Market, Trichy
Golden Rock, Trichy
Main Guard Gate, Tirchy



CODE:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose



df = pd.read_csv(r'D:\nm_dsc\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)

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'])

# Display the first few rows after preprocessing
print("\nHead after preprocessing:")

print(df.head)

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']

# Display the result
print("\nSummary of SO2, NO2, and RSPM/PM10 Levels by Location:")
print(summary)
print()

# 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']

# 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()

# Group by both 'City/Town/Village/Area' and 'Location of Monitoring Station'
and count the number of rows
import matplotlib.pyplot as plt

# Assuming you have the 'summary' DataFrame with 'City', 'Location', 'SO2 Sum',
'SO2 Average', 'NO2 Sum', 'NO2 Average', 'RSPM/PM10 Sum', 'RSPM/PM10 Average'
columns

# Get a list of unique cities
unique_cities = summary['City'].unique()

# 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()




# Group by 'Location of Monitoring Station' and count the number of rows for
each location


# Merge the two DataFrames on 'Location of Monitoring Station' to combine the
results


"""
#calculating individual index for so2 and no2
#so2
def calculate_si(SO2):
    si=0
    if (SO2<=40):
     si= SO2*(50/40)
    if (SO2>40 and SO2<=80):
     si= 50+(SO2-40)*(50/40)
    if (SO2>80 and SO2<=380):
     si= 100+(SO2-80)*(100/300)
    if (SO2>380 and SO2<=800):
     si= 200+(SO2-380)*(100/800)
    if (SO2>800 and SO2<=1600):
     si= 300+(SO2-800)*(100/800)
    if (SO2>1600):
     si= 400+(SO2-1600)*(100/800)
    return si
df['si']=df['SO2'].apply(calculate_si)
df[['SO2','si']]
print(df.head(20))
# Check the column names in your DataFrame
print(df.columns)


def calculate_ni(NO2):
    ni=0
    if(NO2<=40):
     ni= NO2*50/40
    elif(NO2>40 and NO2<=80):
     ni= 50+(NO2-14)*(50/40)
```

```
    elif(NO2>80 and NO2<=180):
     ni= 100+(NO2-80)*(100/100)
    elif(NO2>180 and NO2<=280):
     ni= 200+(NO2-180)*(100/100)
    elif(NO2>280 and NO2<=400):
     ni= 300+(NO2-280)*(100/120)
    else:
     ni= 400+(NO2-400)*(100/120)
    return ni
df['ni']=df['NO2'].apply(calculate_ni)
df[['NO2','ni']]
print(df.head(20))
"""1. Indian equation for AQI:
The Indian AQI range differs from that of US-EPA. To calculate AQI, a minimum
of three parameters should be taken out of which one must be either PM10 or
PM2.5.
To calculate sub-indices, 16 hours of data is needed.

For example: If you wish to calculate AQI on the basis of PM2.5, CO, and ozone,
calculate the sub-index for each parameter separately.
If the current concentration of PM2.5 is 110 ug/m3, then referring to AQI
range as per Indian standards BPHi = 120, BPLo = 91, IHi = 300 and ILo = 201.
Putting the values in equation and solving:
Sub Index= [(300-201)/ (120-91)] (110-91) + 201 = 265.86

Similarly, for other parameters, the sub-index can be calculated and the worst
sub-index shows the AQI."""
df.rename(columns={'RSPM/PM10': 'rspm'}, inplace=True)

def calculate_(rspm):
    rpi=0
    if(rspm<=30):
     rpi=rspm*50/30
    elif(rspm>30 and rspm<=60):
     rpi=50+(rspm-30)*50/30
    elif(rspm>60 and rspm<=90):
     rpi=100+(rspm-60)*100/30
    elif(rspm>90 and rspm<=120):
     rpi=200+(rpi-90)*100/30
    elif(rspm>120 and rspm<=250):
     rpi=300+(rspm-120)*(100/130)
    else:
     rpi=400+(rspm-250)*(100/130)
    return rpi
df['rpi']=df['rspm'].apply(calculate_si)
df[['rspm','rpi']]

print(df.tail(20))
```

```
print(df.columns)

#function to calculate the air quality index (AQI) of every data value
#its is calculated as per indian govt standards
def calculate_aqi(si,ni,rpi):
    aqi=0
    if(si>ni and si>rpi):
     aqi=si
    if(ni>si and ni>rpi):
     aqi=ni
    if(rpi>si and rpi>ni):
     aqi=rpi
    return aqi
df['AQI']=df.apply(lambda x:calculate_aqi(x['si'],x['ni'],x['rpi']),axis=1)
df[['si','ni','rpi','AQI']]
print(df.head(20))


grouped_data = df.groupby(['City/Town/Village/Area', 'Location of Monitoring
Station'])


# Calculate the average AQI for each unique combination
average_aqi = grouped_data['AQI'].mean().reset_index()

# Print the first few rows of the resulting DataFrame to check the data
print(average_aqi)


sns.set(style="whitegrid")

# Create a point plot
plt.figure(figsize=(12, 8))
sns.pointplot(x="AQI", y="City/Town/Village/Area", data=average_aqi,
hue="Location of Monitoring Station", join=False, dodge=True)
plt.title("Average AQI by City/Town/Village/Area")
plt.xlabel("Average AQI")
plt.ylabel("City/Town/Village/Area")
plt.legend(title="Location of Monitoring Station", bbox_to_anchor=(1.05, 1),
loc='upper left')
plt.tight_layout()

plt.show()

print(df.columns)
df.to_csv(r'D:\nm_dsc\cpcb_dly_aq_tamil_nadu-2014-modified.csv', index=False)
print("saved")
```

```
df = pd.read_csv(r'D:\nm_dsc\cpcb_dly_aq_tamil_nadu-2014-modified.csv')
print(df.head)

#monthwise mean aqi which speaks on the pollution level

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

# Extract year and month
df['Year'] = df['Sampling Date'].dt.year
df['Month'] = df['Sampling Date'].dt.month

# Group the data by 'Stn Code,' Year, and Month
grouped_data = df.groupby(['City/Town/Village/Area', 'Year', 'Month'])

# Calculate the mean 'AQI' for each group
month_wise_mean_aqi = grouped_data['AQI'].mean().reset_index()

# Print the first few rows of the resulting DataFrame
print(month_wise_mean_aqi.head(25))




sns.set(style="whitegrid")

# Create a figure and axis
fig, ax = plt.subplots(figsize=(12, 6))

# Plot the data for each 'City/Town/Village/Area'
sns.lineplot(x="Month", y="AQI", hue="City/Town/Village/Area", data=df, ax=ax)

# Set labels and title
ax.set_xlabel("Month")
ax.set_ylabel("AQI")
ax.set_title("Month-wise Mean AQI by City/Town/Village/Area")

# Add a legend
ax.legend(title="City/Town/Village/Area")

# Show the plot
plt.show()




#overall month trends
# Convert the 'Sampling Date' column to datetime
df['Sampling Date'] = pd.to_datetime(df['Sampling Date'])
```

```
# Extract the year and month from the 'Sampling Date'
df['Year'] = df['Sampling Date'].dt.year
df['Month'] = df['Sampling Date'].dt.month

# Calculate the mean AQI for all data (no grouping by 'City/Town/Village/Area')
month_wise_mean_aqi = df.groupby(['Year', 'Month'])['AQI'].mean().reset_index()

# Print the first few rows of the resulting DataFrame to check the data
print(month_wise_mean_aqi.head(100))




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

# Extract the year and month from the 'Sampling Date'
df['Year'] = df['Sampling Date'].dt.year
df['Month'] = df['Sampling Date'].dt.month

# Calculate the mean AQI for all data (no grouping by 'City/Town/Village/Area')
month_wise_mean_aqi = df.groupby(['Year', 'Month'])['AQI'].mean().reset_index()

# Create a line plot for month-wise mean AQI
plt.figure(figsize=(12, 6))
plt.plot(month_wise_mean_aqi['Month'], month_wise_mean_aqi['AQI'], marker='o')
plt.title('Month-wise Mean AQI')
plt.xlabel('Month')
plt.ylabel('Mean AQI')
plt.grid(True)

# Show the plot
plt.show()


plt.figure(figsize=(12, 6))
sns.scatterplot(x='si', y='ni', hue='AQI', data=df, palette='viridis',
size='AQI', sizes=(20, 200))

# Set labels and title
plt.xlabel('Sulfur Dioxide (SI)')
plt.ylabel('Nitrogen Dioxide (NI)')
plt.title('Clusters of SI, NI, and AQI Values')

# Show the legend
plt.legend(title='AQI')

# Show the plot
plt.show()
```

```
#In this code, we use sns.scatterplot from the Seaborn library to create a scatter plot that visualizes the clusters of SI, NI, and AQI values. The hue parameter is set to 'AQI' to color the data points based on the AQI values. The size parameter is also set to 'AQI' to represent AQI values with different point sizes. You can customize the colors, size range, and other plot aesthetics as needed.


# Your previous code to read and preprocess the data







sns.set(style="whitegrid")

# Create a figure and axis
fig, ax = plt.subplots(figsize=(12, 6))

# Plot the data for each 'City/Town/Village/Area'
sns.lineplot(x="City/Town/Village/Area", y="AQI", data=df, ax=ax)

# Set labels and title
ax.set_xlabel("City/Town/Village/Area")
ax.set_ylabel("AQI")
ax.set_title("AQI by City/Town/Village/Area")

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.show()
```