GOVERNMENT COLLEGE OF TECHNOLOGY COIMBATORE – 13

PROJECT NAME: Air Quality Assessment TN

TEAM MEMBERS:

1. Kokulavenkat K
2. Kayalvizhi T
3. Karunasri A
4. Keerthana R

PHASE 3: **PROJECT DEVELOPMENT LEVEL 1**

PROBLEM DESCRIPTION:

Gather air quality data from monitoring stations across Tamil Nadu. This data may include measurements of pollutants like PM2.5, PM10, nitrogen dioxide (NO2), sulfur dioxide (SO2), ozone (O3), carbon monoxide (CO), and other relevant parameters.

Ensure the data is structured, accurate, and includes timestamps for each measurement. Aggregate data if multiple stations provide data for the same location. To visualise the data including line charts, bar graphs, heatmaps, and scatter plots and compare different pollutants. Geographic visualizations using tools like GIS software or Python libraries like Folium to map air quality across different regions in Tamil Nadu.

Consider using color-coding to represent air quality levels and station locations on the map. Calculate AQI based on the collected data and the relevant formula for Tamil Nadu. To visualise the air quality find how different pollutants relate to each other and apply time series analysis techniques , autoregressive integrated moving average(ARIMA) or prophet for forecasting air quality.

LIBRARIES USED:

Air quality analysis in Tamil Nadu, like in other regions, can benefit from variouslibraries and tools. Here are some specific details about libraries that can be used for an air quality analysis project in Tamil Nadu:

1. **Pandas:** Pandas is a fundamental library for data analysis and manipulation. You can use it to load, clean, and process air quality data in various formats such as CSV, Excel, or databases. This library is essential for data preprocessing.

2. **Matplotlib and Seaborn**: Matplotlib and Seaborn are great choices for creating visualizations of air quality data. You can generate time series plots, scatter plots, bar

charts, and maps to understand the trends and patterns in air quality over time and across locations in Tamil Nadu.

3. **NumPy:** NumPy is essential for numerical operations. You can use it for various calculations, such as statistical analysis, averaging air quality values, and more.

4. **Scikit-Learn**: If you want to develop predictive models for air quality forecasting or classification, Scikit-Learn can be helpful. You can train machine learning models to predict air quality levels based on various factors.

APIS' USED:

1. **Statsmodels:** Statsmodels is useful for statistical analysis. You can perform hypothesis testing, regression analysis, and time series analysis to gain insights into the factors affecting air quality in Tamil Nadu.

2. **PyAirQuality:** PyAirQuality is a Python library designed for air quality analysis. It may provide specific tools and functions tailored to air quality data analysis. You can check if it has features relevant to Tamil Nadu.

3. **Tamil Nadu Pollution Control Board (TNPCB) API:** If TNPCB or other relevant agencies provide air quality data through APIs, you can use these to fetch real-time or historical data for your analysis.

LOADING AN DATASET:

Typically, load a dataset into a pandas DataFrame using one of the following methods:

1. **Using pd.read_csv() for a CSV file**:

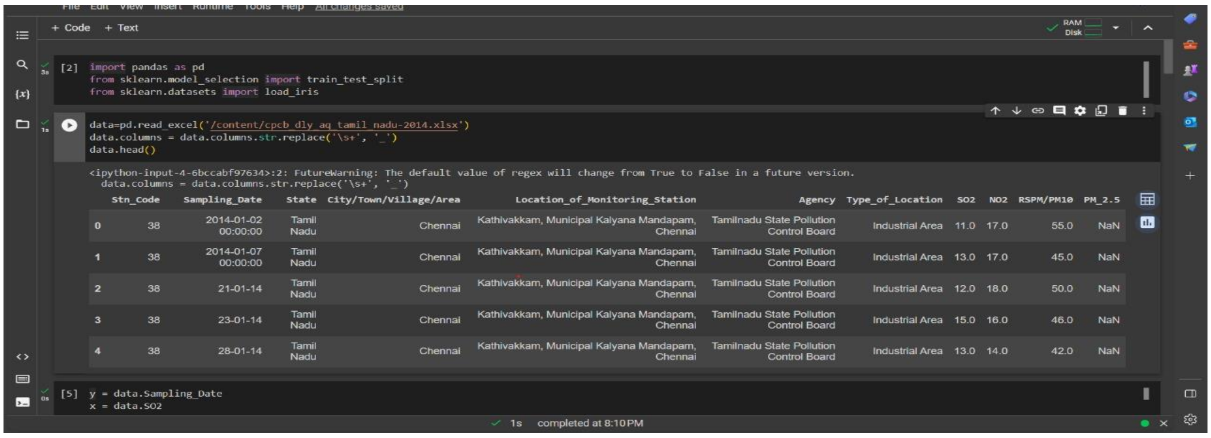
```
python
import pandas as pd
data = pd.read_csv('your_dataset.csv')
```

2. **Using pd.read_excel() for an Excel file**:

```
python
import pandas as pd
data = pd.read_excel('your_dataset.xlsx')
```

3. **Using other methods for different data formats (e.g., JSON, SQL, etc.):**

Depending on the data format, you would use an appropriate function to load the data into a pandas DataFrame.

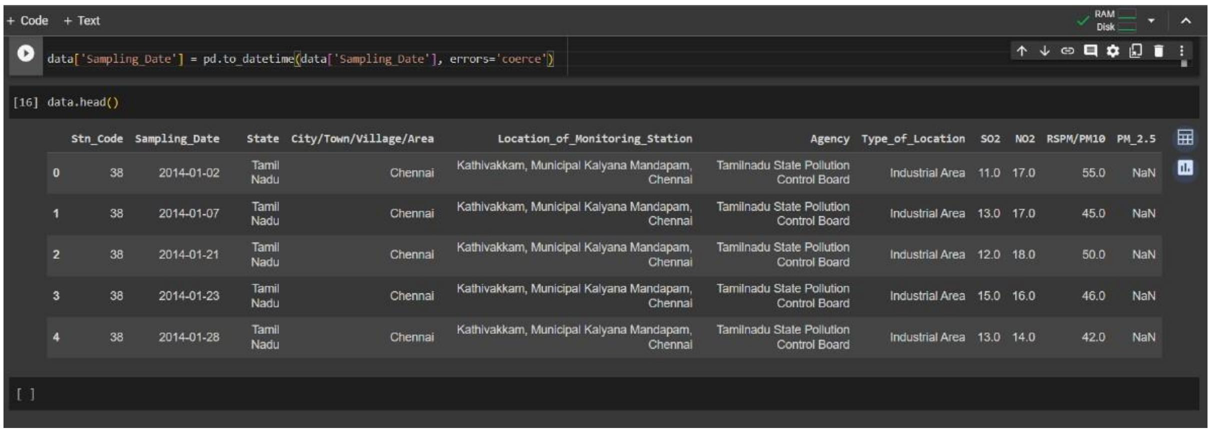


```
[2] import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.datasets import load_iris

data=pd.read_excel('/content/cpcb_dly_aq_tamil_nadu-2014.xlsx')
data.columns = data.columns.str.replace('\s+', '_')
data.head()

<ipython-input-4-6bccabf97634>:2: FutureWarning: The default value of regex will change from True to False in a future version.
  data.columns = data.columns.str.replace('\s+', '_')
```

	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	2014-01-02 00:00:00	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	11.0	17.0	55.0	NaN
1	38	2014-01-07 00:00:00	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

```
[5] y = data.Sampling_Date
    x = data.SO2
```

DATE CONVERSION:



```
data['Sampling_Date'] = pd.to_datetime(data['Sampling_Date'], errors='coerce')

[16] data.head()
```

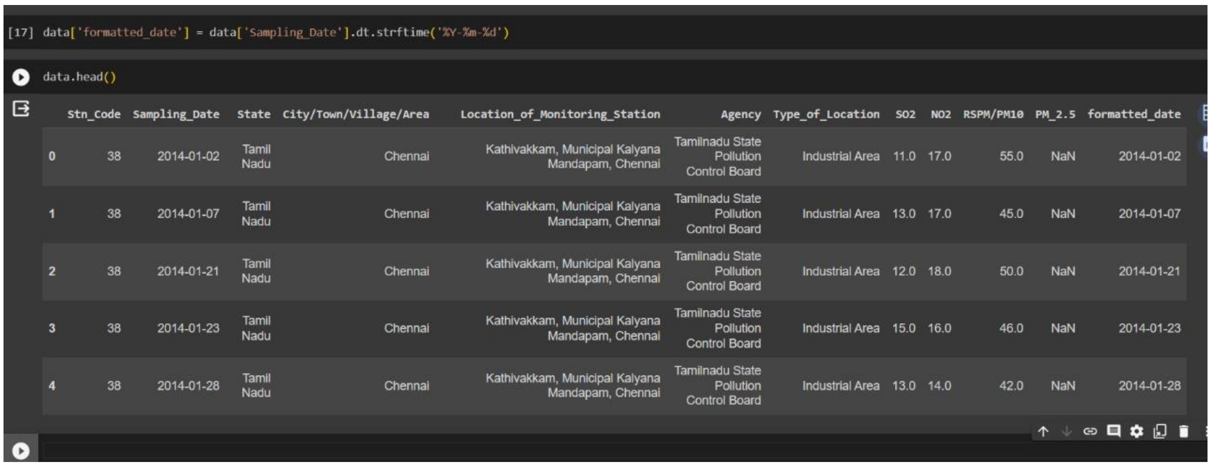
	Stn_Code	Sampling_Date	State	City/Town/Village/Area	Location_of_Monitoring_Station	Agency	Type_of_Location	SO2	NO2	RSPM/PM10	PM_2.5
0	38	2014-01-02	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	11.0	17.0	55.0	NaN
1	38	2014-01-07	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	13.0	17.0	45.0	NaN
2	38	2014-01-21	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	12.0	18.0	50.0	NaN
3	38	2014-01-23	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	15.0	16.0	46.0	NaN
4	38	2014-01-28	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	13.0	14.0	42.0	NaN

The first line of code converts the 'Sampling_Date' column to a datetime format. This is important for any date-related operations that might be required in the analysis.

data['Sampling_Date'] =pd.to_datetime(data['Sampling_Date'], errors='coerce'):

This line is converting the 'Sampling_Date' column in the DataFrame 'data' to a datetime format. The pd.to_datetime() function is used for this purpose. The errors='coerce' parameter is set to handle any conversion errors by converting them to NaT (Not a Timestamp) values, which is a way to represent missing or invalid dates.

Date Formatting:



```
[17] data['formatted_date'] = data['Sampling_Date'].dt.strftime('%Y-%m-%d')
```

```
data.head()
```

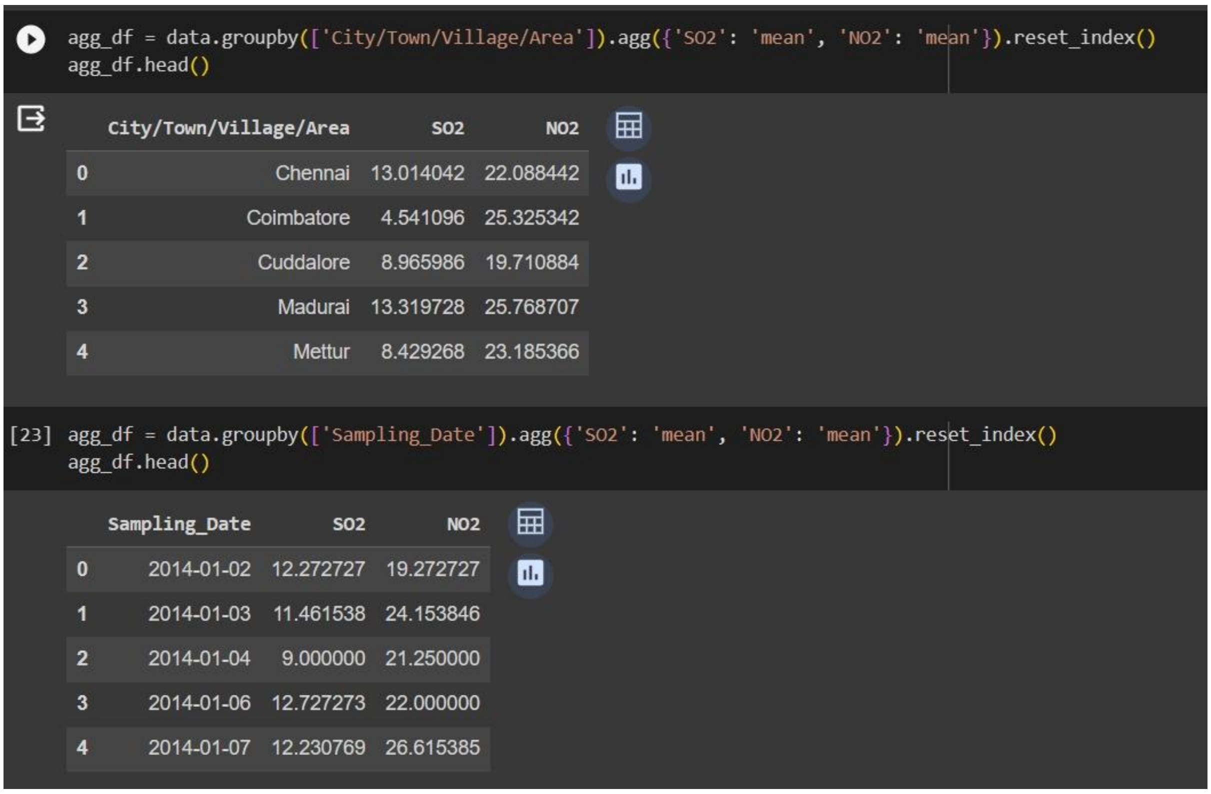
```
   Stn_Code  Sampling_Date  State  City/Town/Village/Area  Location_of_Monitoring_Station                Agency                   Type_of_Location  SO2   NO2   RSPM/PM10  PM_2.5  formatted_date
0  38        2014-01-02     Tamil  Chennai                 Kathivakkam, Municipal Kalyana Mandapam, Chennai  Tamilnadu State Pollution Control Board  Industrial Area  11.0  17.0  55.0  NaN  2014-01-02
                            Nadu
1  38        2014-01-07     Tamil  Chennai                 Kathivakkam, Municipal Kalyana Mandapam, Chennai  Tamilnadu State Pollution Control Board  Industrial Area  13.0  17.0  45.0  NaN  2014-01-07
                            Nadu
2  38        2014-01-21     Tamil  Chennai                 Kathivakkam, Municipal Kalyana Mandapam, Chennai  Tamilnadu State Pollution Control Board  Industrial Area  12.0  18.0  50.0  NaN  2014-01-21
                            Nadu
3  38        2014-01-23     Tamil  Chennai                 Kathivakkam, Municipal Kalyana Mandapam, Chennai  Tamilnadu State Pollution Control Board  Industrial Area  15.0  16.0  46.0  NaN  2014-01-23
                            Nadu
4  38        2014-01-28     Tamil  Chennai                 Kathivakkam, Municipal Kalyana Mandapam, Chennai  Tamilnadu State Pollution Control Board  Industrial Area  13.0  14.0  42.0  NaN  2014-01-28
                            Nadu
```

The second line of code creates a new column called 'formatted_date' by formatting the 'Sampling_Date' column in aThe provided code appears to be related to loading and preprocessing a dataset, likely using the Python library pandas.

data['formatted_date'] = data['Sampling_Date'].dt.strftime('%Y-%m-d'):

This line is creating a new column 'formatted_date' in the DataFrame 'data' by applying a specific date format to the 'Sampling_Date' column. The .dt.strftime('%Y-%m-d') function extracts the year, month, and day components from the datetime values and formats them as 'YYYY-MM-DD'.

Data Aggregation:



```
agg_df = data.groupby(['City/Town/Village/Area']).agg({'SO2': 'mean', 'NO2': 'mean'}).reset_index()
agg_df.head()
```

```
   City/Town/Village/Area  SO2        NO2
0                 Chennai  13.014042  22.088442
1              Coimbatore   4.541096  25.325342
2               Cuddalore   8.965986  19.710884
3                 Madurai  13.319728  25.768707
4                  Mettur   8.429268  23.185366
```

```
[23] agg_df = data.groupby(['Sampling_Date']).agg({'SO2': 'mean', 'NO2': 'mean'}).reset_index()
     agg_df.head()
```

```
   Sampling_Date  SO2        NO2
0  2014-01-02     12.272727  19.272727
1  2014-01-03     11.461538  24.153846
2  2014-01-04      9.000000  21.250000
3  2014-01-06     12.727273  22.000000
4  2014-01-07     12.230769  26.615385
```

The last line of code aggregates the data. It groups the DataFrame by two columns, 'City/Town/Village/Area' and 'Sampling_Date', and then calculates the mean values of 'SO2' and 'NO2' for each group. This aggregation can be helpful when we want to summarize data for analysis, especially when dealing with time-series or geographical data.

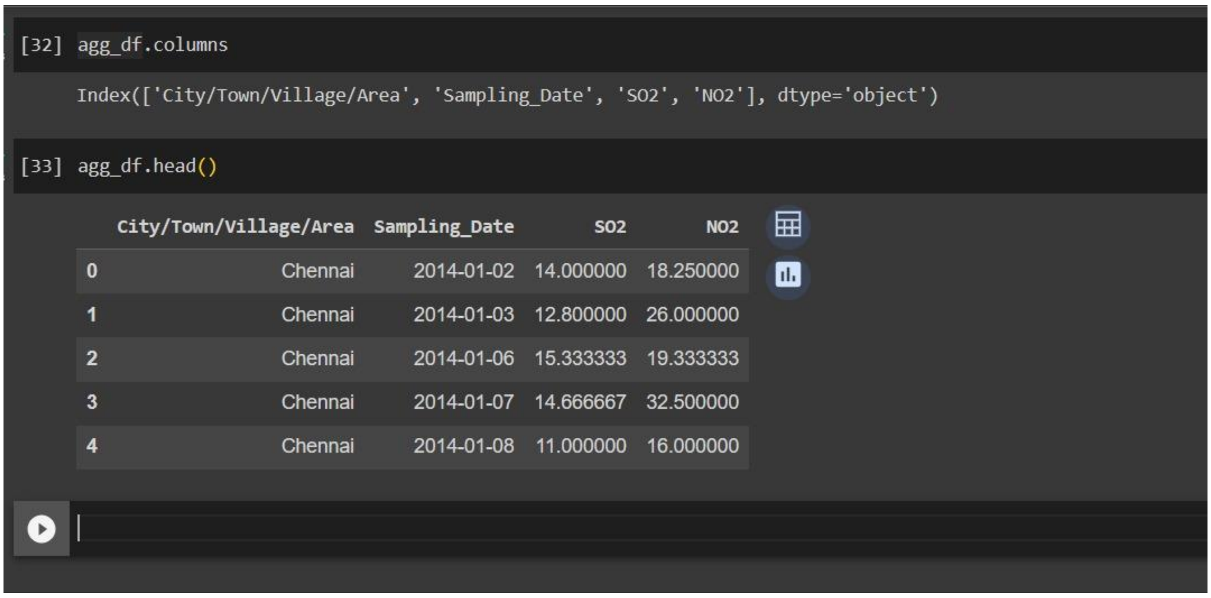
agg_df = data.groupby(['City/Town/Village/Area', 'Sampling_Date']).agg({'SO2': 'mean', 'NO2': 'mean'}).reset_index():

In this line, the DataFrame 'data' is being grouped by two columns: 'City/Town/Village/Area' and 'Sampling_Date'.

The .agg() method is used to compute the mean of the 'SO2' and 'NO2' columns for each group.

Finally, .reset_index() is applied to reset the index of the resulting DataFrame 'agg_df', creating a new index column.

FILTERING UNWANTED DATA:



```
[32] agg_df.columns

     Index(['City/Town/Village/Area', 'Sampling_Date', 'SO2', 'NO2'], dtype='object')

[33] agg_df.head()
```

	City/Town/Village/Area	Sampling_Date	SO2	NO2
0	Chennai	2014-01-02	14.000000	18.250000
1	Chennai	2014-01-03	12.800000	26.000000
2	Chennai	2014-01-06	15.333333	19.333333
3	Chennai	2014-01-07	14.666667	32.500000
4	Chennai	2014-01-08	11.000000	16.000000

To fetch specific columns from a DataFrame, use DataFrame indexing. In this case, to select the 'NO2', 'SO2', 'City', and 'Sampling_Date' columns. Here's an explanation of how this is done:

data[['NO2', 'SO2', 'City', 'Sampling_Date']]:

- data is the original DataFrame containing your dataset.

- ['NO2', 'SO2', 'City', 'Sampling_Date'] is a list of column names that you want to extract from the DataFrame.

- The double square brackets [['...']] are used to index the DataFrame, indicating that you want to select specific columns.

CONCLUSION:

In summary, the preprocessing steps in this code involve converting date columns, formatting them, and aggregating data for further analysis. These steps ensure that the data is in a suitable format for various types of analysis and visualization.