**Real-Time Data Processing System for**

**Weather Monitoring with Rollups and Aggregates**

**Objective:**

Develop a real-time data processing system to monitor weather conditions and provide summarized insights using rollups and aggregates. The system will utilize data from the OpenWeatherMap API [(https://openweathermap.org/](https://openweathermap.org/)).

**Data Source:**

The system will continuously retrieve weather data from the OpenWeatherMap API. You will need to sign up for a free API key to access the data. The API provides various weather parameters, and for this assignment, we will focus on:

* main: Main weather condition (e.g., Rain, Snow, Clear)
* temp: Current temperature in Centigrade
* feels\_like: Perceived temperature in Centigrade
* dt: Time of the data update (Unix timestamp)

**Processing and Analysis:**

* The system should continuously call the OpenWeatherMap API at a configurable interval (e.g., every 5 minutes) to retrieve real-time weather data for the metros in India. (Delhi, Mumbai, Chennai, Bangalore, Kolkata, Hyderabad)

● For each received weather update:

○ Convert temperature values from Kelvin to Celsius (tip : you can also use user preference).

**Rollups and Aggregates:**

1. **Daily Weather Summary:**
   1. Roll up the weather data for each day.

○ Calculate daily aggregates for:

■ Average temperature

■ Maximum temperature

■ Minimum temperature

■ Dominant weather condition (give reason on this)

○Store the daily summaries in a database or persistent storage for further analysis.

1. **Alerting Thresholds:**
   1. Define user-configurable thresholds for temperature or specific weather conditions (e.g., alert if temperature exceeds 35 degrees Celsius for two consecutive updates).

○ Continuously track the latest weather data and compare it with the thresholds.

○ If a threshold is breached, trigger an alert for the current weather conditions. Alerts could be displayed on the console or sent through an email notification system (implementation details left open-ended).

1. **Implement visualizations:**
   1. To display daily weather summaries, historical trends, and triggered alerts.

**Test Cases:**

1. **System Setup:**
   1. Verify system starts successfully and connects to the OpenWeatherMap API using a valid API key.
2. **Data Retrieval:**
   1. Simulate API calls at configurable intervals.

○ Ensure the system retrieves weather data for the specified location and parses the response correctly.

1. **Temperature Conversion:**
   1. Test conversion of temperature values from Kelvin to Celsius (or Fahrenheit) based on user preference.
2. **Daily Weather Summary:**
   1. Simulate a sequence of weather updates for several days.

○ Verify that daily summaries are calculated correctly, including average, maximum, minimum temperatures and dominant weather condition.

1. **Alerting Thresholds:**
   1. Define and configure user thresholds for temperature or weather conditions.

○ Simulate weather data exceeding or breaching the thresholds.

○ Verify that alerts are triggered only when a threshold is violated.

**Bonus:**

* Extend the system to support additional weather parameters from the OpenWeatherMap API (e.g., humidity, wind speed) and incorporate them into rollups/aggregates.
* Explore functionalities like weather forecasts retrieval and generating summaries based on predicted conditions.

**SOLUTION**

**(GITHUB LINK:** [**https://github.com/LikhithaKANURU/Real-Time-Weather-Monitoring-System**](https://github.com/LikhithaKANURU/Real-Time-Weather-Monitoring-System) **)**

**Overview**

This project aims to develop a real-time data processing system that monitors weather conditions for major metros in India and provides summarized insights using rollups and aggregates from data retrieved via the OpenWeatherMap API.

**Objective**

To create a system that:

1. Continuously retrieves real-time weather data.
2. Processes and analyzes the data to generate daily summaries.
3. Sends alerts based on user-defined thresholds.
4. Implements visualizations for better data representation.

**Data Source**

The system will pull weather data from the OpenWeatherMap API, focusing on:

* main: Main weather condition (e.g., Rain, Snow, Clear)
* temp: Current temperature in Celsius
* feels\_like: Perceived temperature in Celsius
* dt: Time of data update (Unix timestamp)

**Architecture**

1. **Data Retrieval Module**
   * Set up a cron job or a scheduled task to call the OpenWeatherMap API every 5 minutes.
   * Fetch data for: Delhi, Mumbai, Chennai, Bangalore, Kolkata, Hyderabad.
2. **Data Processing Module**
   * Parse the JSON response from the API.
   * Convert temperature from Kelvin to Celsius
3. **Rollups and Aggregates**
   * Maintain an in-memory data structure (like a dictionary) to store weather data for each day.
   * For each day, compute:
     + **Average Temperature**: Total temperature / Number of updates
     + **Maximum Temperature**: Highest temperature recorded during the day
     + **Minimum Temperature**: Lowest temperature recorded during the day
     + **Dominant Weather Condition**: The condition that occurs most frequently (calculate frequency from the main data).
4. **Alerting System**
   * Allow users to define thresholds for alerts (e.g., temperature > 35°C).
   * Track real-time data and check if it breaches any thresholds.
   * Trigger alerts via console output or email notification when conditions are met.
5. **Visualization Module**
   * Use libraries like Matplotlib or Plotly to visualize:
     + Daily summaries
     + Historical trends
     + Triggered alerts

**Implementation Details**

* **Database**: Use a lightweight database like SQLite or a NoSQL option like MongoDB to store daily summaries for persistent storage.
* **Email Notifications**: Utilize SMTP libraries in Python (e.g., smtplib) to send alerts via email.
* **Configuration**: Use a JSON or YAML file to manage user preferences and thresholds.

**Test Cases**

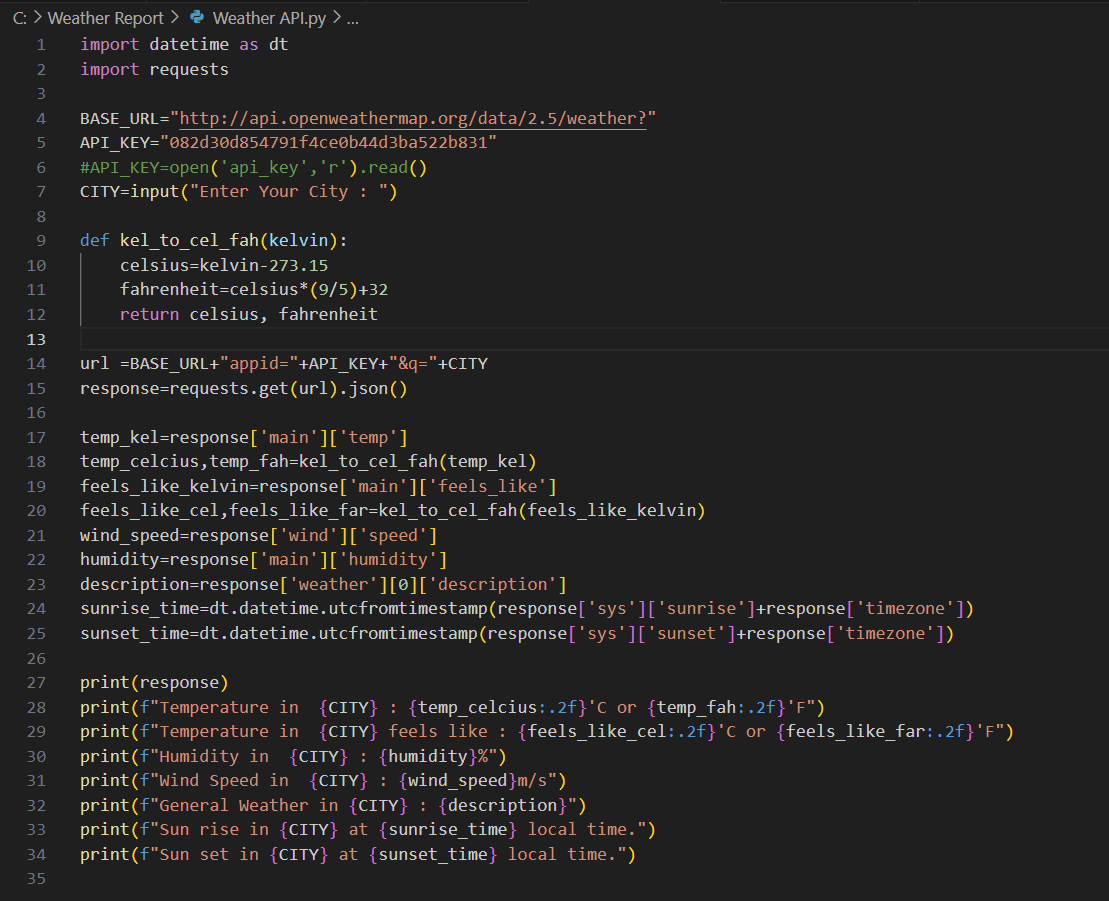
1. **System Setup**
   * Verify connection to the OpenWeatherMap API with a valid API key.
2. **Data Retrieval**
   * Simulate API calls and ensure correct weather data retrieval and parsing.
3. **Temperature Conversion**
   * Confirm that temperature values are accurately converted from Kelvin to Celsius.
4. **Daily Weather Summary**
   * Simulate several days of data and verify that aggregates (average, max, min temperatures, and dominant weather) are calculated correctly.
5. **Alerting Thresholds**
   * Configure thresholds and simulate weather data to verify that alerts are triggered correctly.

**Bonus Features**

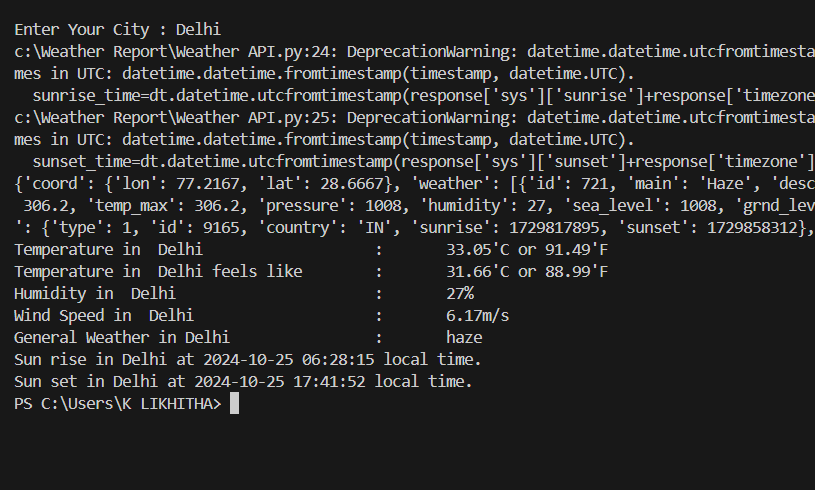
1. **Additional Weather Parameters**
   * Extend functionality to include humidity, wind speed, etc., in rollups/aggregates.
2. **Weather Forecasts**
   * Implement a feature to retrieve and summarize weather forecasts from the API.

**PROGRAMS AND SOME OUTPUTS:**

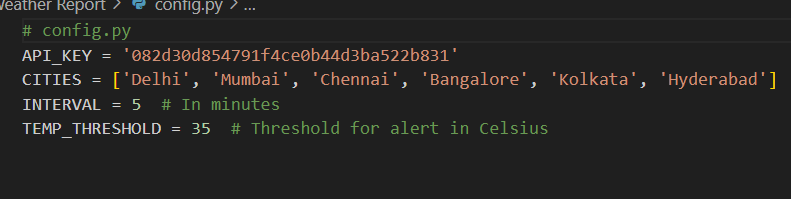
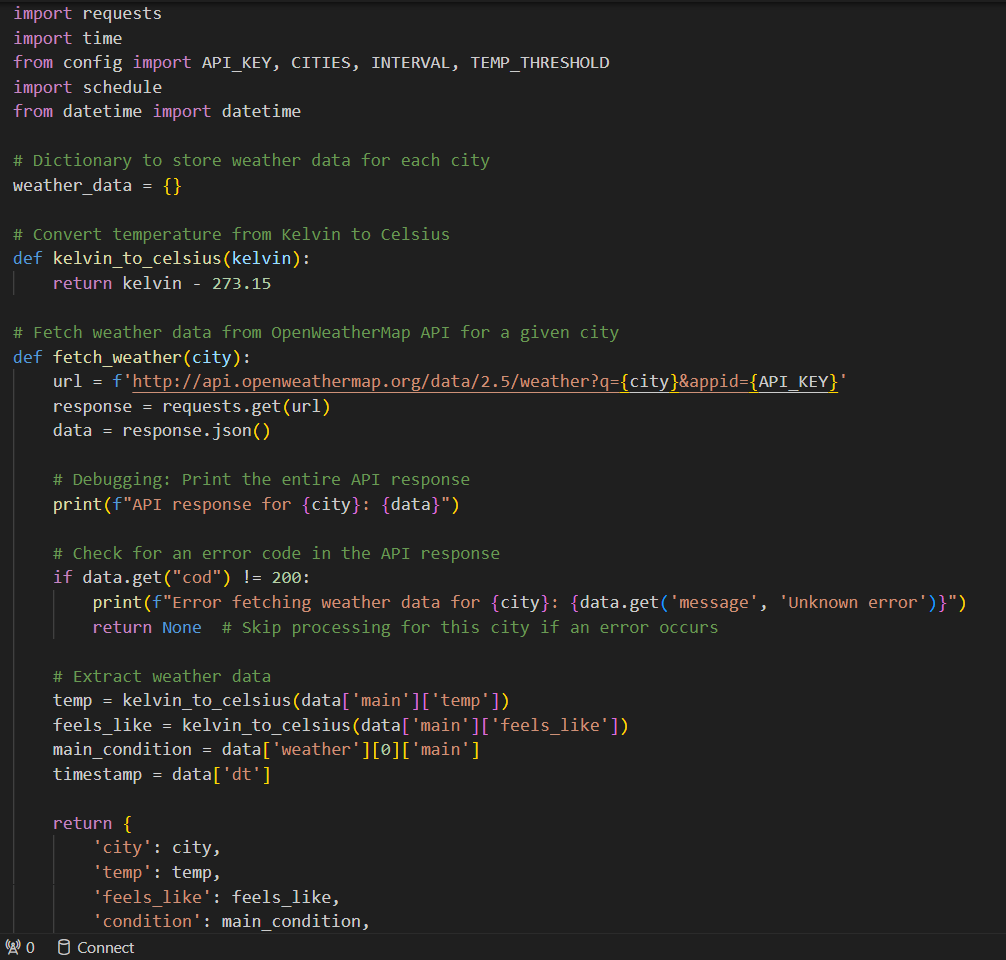
**1.System Setup:**

****

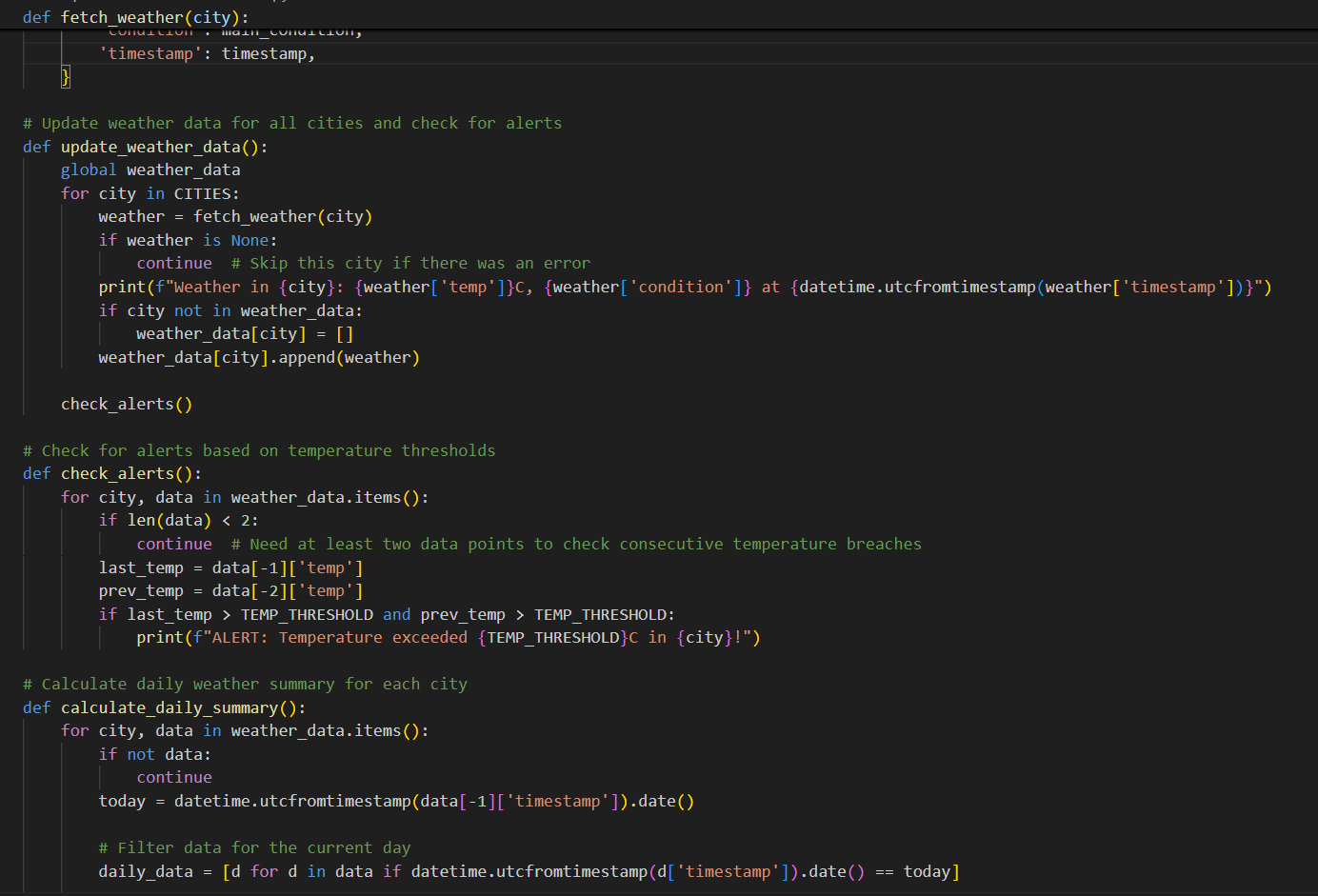
**OUTPUT:**

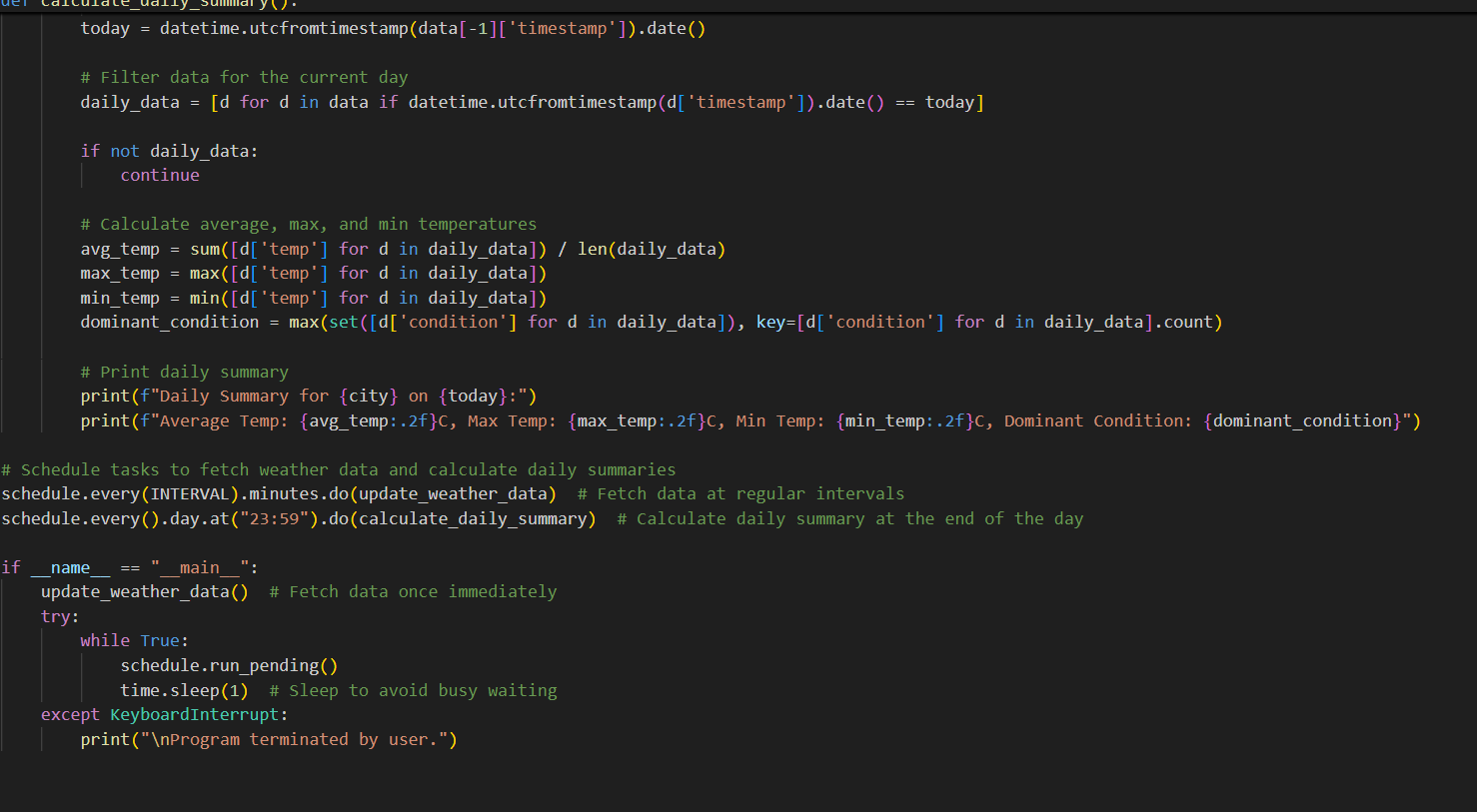
****

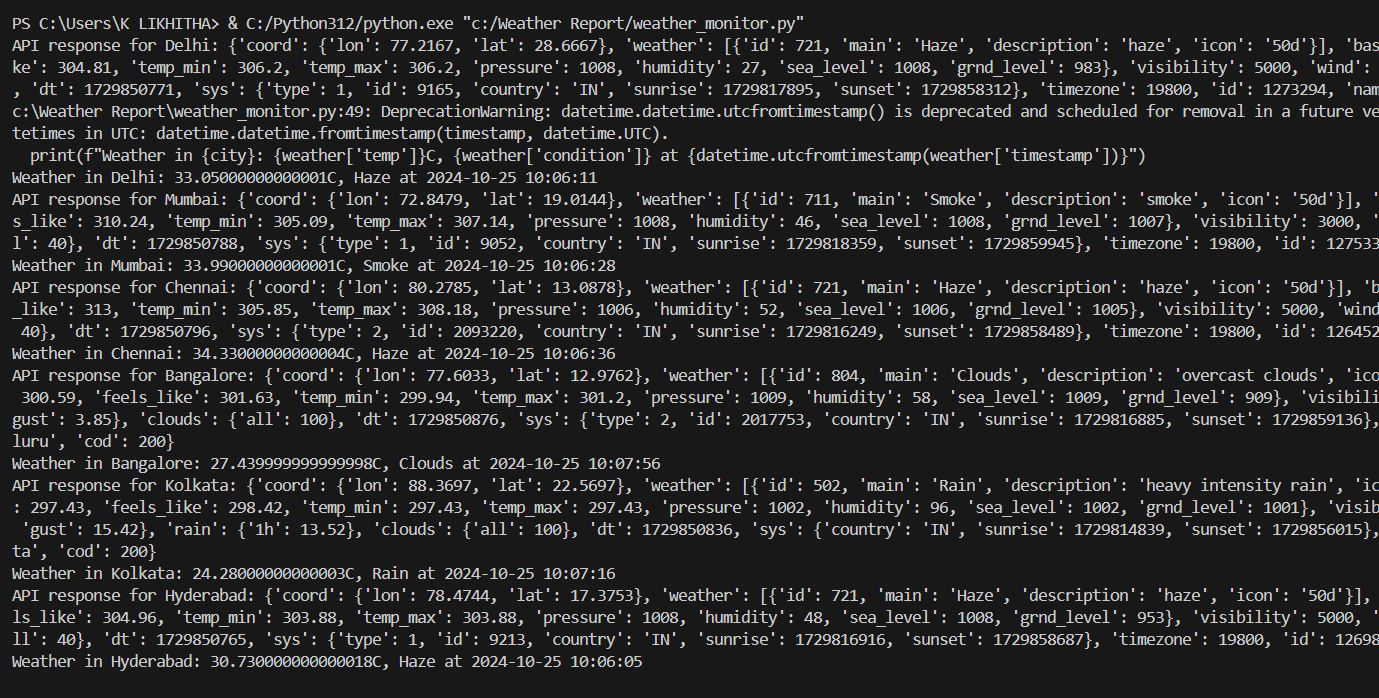
**2.CONFIGURATION**

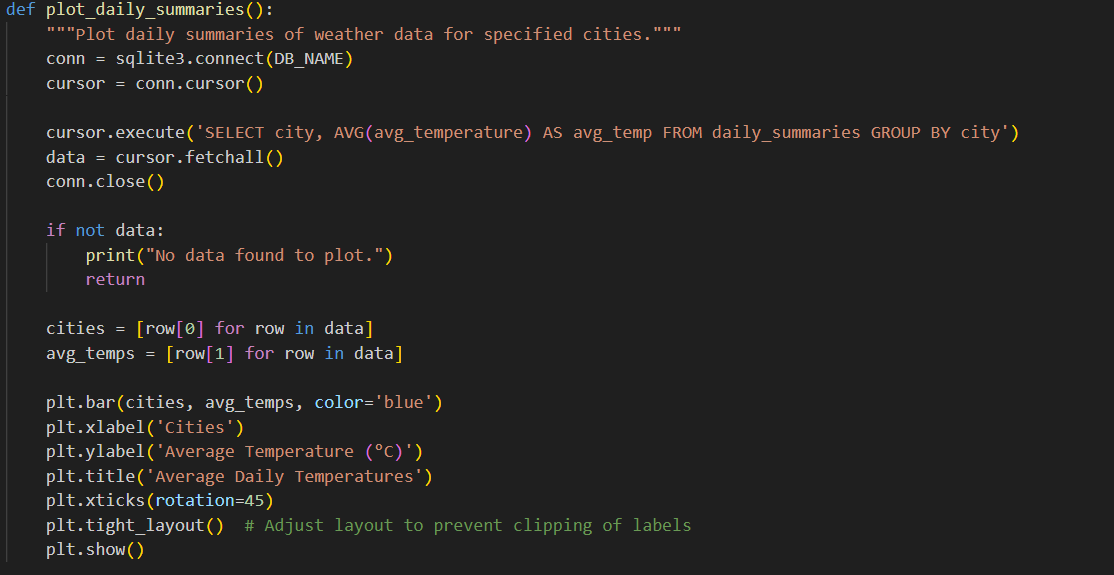


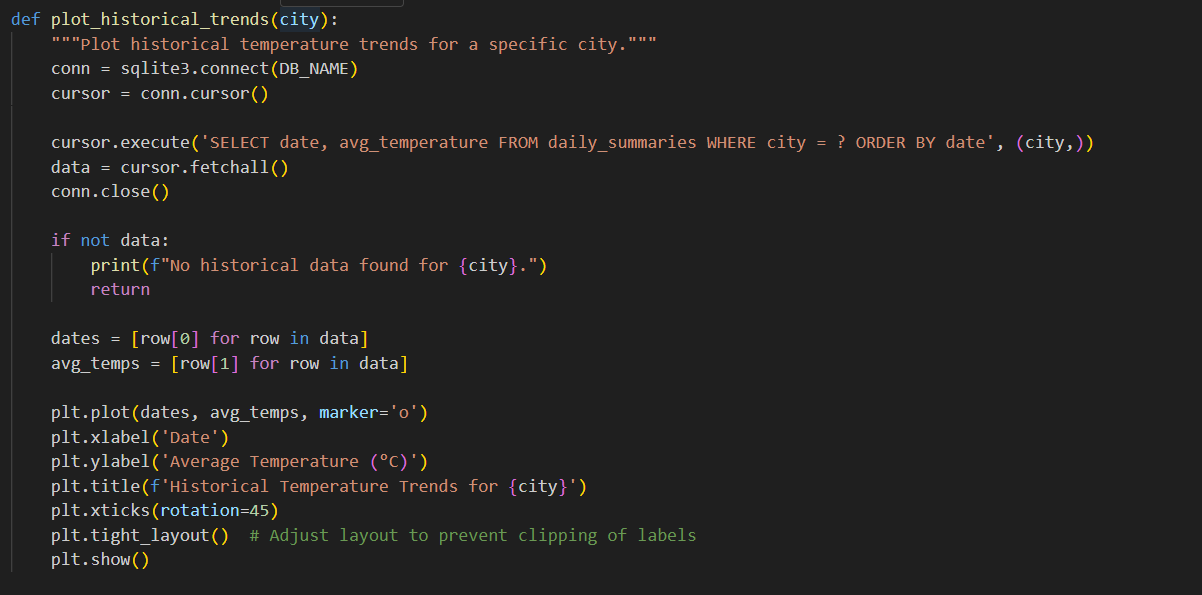
**3.WEATHER MONITORING:**



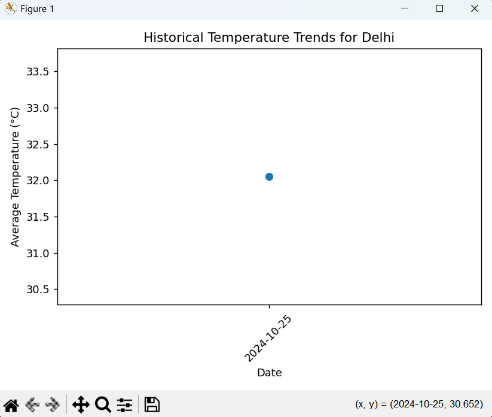
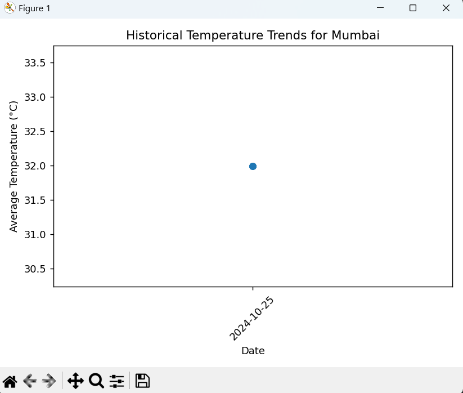
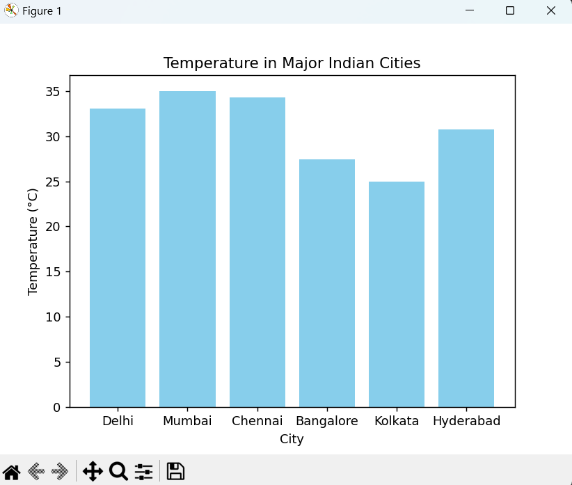


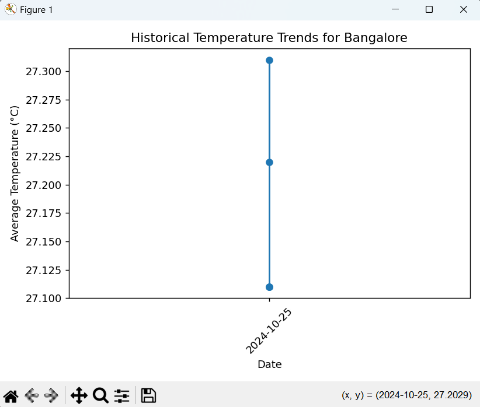
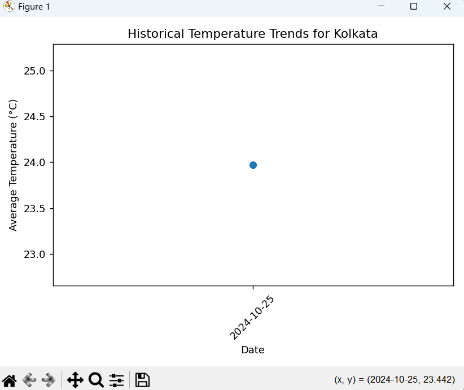
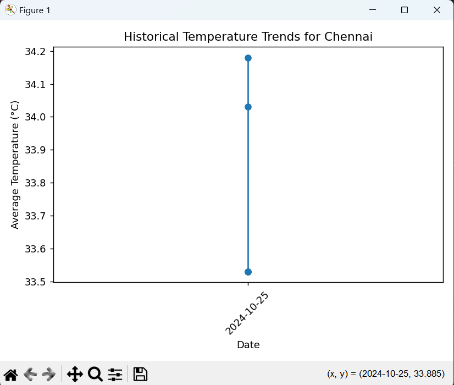
**OUTPUT:**

**4.VISUALIZATION:**

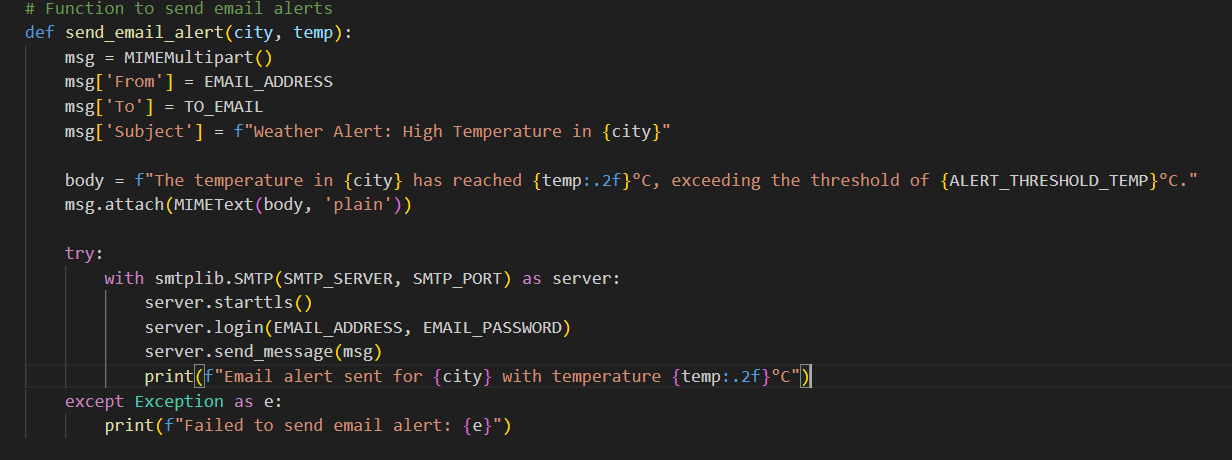
****

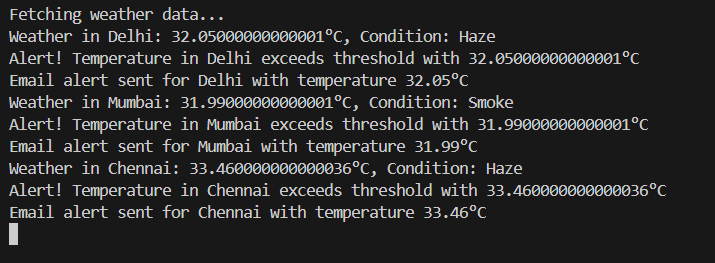
**OUTPUT:**

****

****

**5.ALERTING:**

****

**OUTPUT:**