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/).

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:**
    - 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.

2. **Alerting Thresholds:**
    - 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).

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

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.

1.System Setup:

C: > Weather Report > Weather API.py > ...

```python
import datetime as dt
import requests

BASE_URL="http://api.openweathermap.org/data/2.5/weather?"
API_KEY="082d30d854791f4ce0b44d3ba522b831"
#API_KEY=open('api_key','r').read()
CITY=input("Enter Your City : ")

def kel_to_cel_fah(kelvin):
    celsius=kelvin-273.15
    fahrenheit=celsius*(9/5)+32
    return celsius, fahrenheit

url =BASE_URL+"appid="+API_KEY+"&q="+CITY
response=requests.get(url).json()

temp_kel=response['main']['temp']
temp_celcius,temp_fah=kel_to_cel_fah(temp_kel)
feels_like_kelvin=response['main']['feels_like']
feels_like_cel,feels_like_far=kel_to_cel_fah(feels_like_kelvin)
wind_speed=response['wind']['speed']
humidity=response['main']['humidity']
description=response['weather'][0]['description']
sunrise_time=dt.datetime.utcfromtimestamp(response['sys']['sunrise']+response['timezone'])
sunset_time=dt.datetime.utcfromtimestamp(response['sys']['sunset']+response['timezone'])

print(response)
print(f"Temperature in  {CITY} : {temp_celcius:.2f}'C or {temp_fah:.2f}'F")
print(f"Temperature in  {CITY} feels like : {feels_like_cel:.2f}'C or {feels_like_far:.2f}'F")
print(f"Humidity in  {CITY} : {humidity}%")
print(f"Wind Speed in  {CITY} : {wind_speed}m/s")
print(f"General Weather in {CITY} : {description}")
print(f"Sun rise in {CITY} at {sunrise_time} local time.")
print(f"Sun set in {CITY} at {sunset_time} local time.")

```

OUTPUT:

```
Enter Your City : Delhi
c:\Weather Report\Weather API.py:24: DeprecationWarning: datetime.datetime.utcfromtimesta
mes in UTC: datetime.datetime.fromtimestamp(timestamp, datetime.UTC).
  sunrise_time=dt.datetime.utcfromtimestamp(response['sys']['sunrise']+response['timezone
c:\Weather Report\Weather API.py:25: DeprecationWarning: datetime.datetime.utcfromtimesta
mes in UTC: datetime.datetime.fromtimestamp(timestamp, datetime.UTC).
  sunset_time=dt.datetime.utcfromtimestamp(response['sys']['sunset']+response['timezone']
{'coord': {'lon': 77.2167, 'lat': 28.6667}, 'weather': [{'id': 721, 'main': 'Haze', 'desc
 306.2, 'temp_max': 306.2, 'pressure': 1008, 'humidity': 27, 'sea_level': 1008, 'grnd_lev
': {'type': 1, 'id': 9165, 'country': 'IN', 'sunrise': 1729817895, 'sunset': 1729858312},
Temperature in  Delhi                 :       33.05'C or 91.49'F
Temperature in  Delhi feels like      :       31.66'C or 88.99'F
Humidity in  Delhi                    :       27%
Wind Speed in  Delhi                  :       6.17m/s
General Weather in Delhi              :       haze
Sun rise in Delhi at 2024-10-25 06:28:15 local time.
Sun set in Delhi at 2024-10-25 17:41:52 local time.
PS C:\Users\K LIKHITHA>
```

2.CONFIGURATION

```
Weather Report > config.py > ...
# config.py
API_KEY = '082d30d854791f4ce0b44d3ba522b831'
CITIES = ['Delhi', 'Mumbai', 'Chennai', 'Bangalore', 'Kolkata', 'Hyderabad']
INTERVAL = 5  # In minutes
TEMP_THRESHOLD = 35  # Threshold for alert in Celsius
```

3.WEATHER MONITORING:

```
def fetch_weather(city):
        'timestamp': timestamp,
    }

# Update weather data for all cities and check for alerts
def update_weather_data():
    global weather_data
    for city in CITIES:
        weather = fetch_weather(city)
        if weather is None:
            continue  # Skip this city if there was an error
        print(f"Weather in {city}: {weather['temp']}C, {weather['condition']} at {datetime.utcfromtimestamp(weather['timestamp'])}")
        if city not in weather_data:
            weather_data[city] = []
        weather_data[city].append(weather)

    check_alerts()

# Check for alerts based on temperature thresholds
def check_alerts():
    for city, data in weather_data.items():
        if len(data) < 2:
            continue  # Need at least two data points to check consecutive temperature breaches
        last_temp = data[-1]['temp']
        prev_temp = data[-2]['temp']
        if last_temp > TEMP_THRESHOLD and prev_temp > TEMP_THRESHOLD:
            print(f"ALERT: Temperature exceeded {TEMP_THRESHOLD}C in {city}!")

# Calculate daily weather summary for each city
def calculate_daily_summary():
    for city, data in weather_data.items():
        if not data:
            continue
        today = datetime.utcfromtimestamp(data[-1]['timestamp']).date()

        # Filter data for the current day
        daily_data = [d for d in data if datetime.utcfromtimestamp(d['timestamp']).date() == today]
```

```
def calculate_daily_summary():
        today = datetime.utcfromtimestamp(data[-1]['timestamp']).date()

        # Filter data for the current day
        daily_data = [d for d in data if datetime.utcfromtimestamp(d['timestamp']).date() == today]

        if not daily_data:
            continue

        # Calculate average, max, and min temperatures
        avg_temp = sum([d['temp'] for d in daily_data]) / len(daily_data)
        max_temp = max([d['temp'] for d in daily_data])
        min_temp = min([d['temp'] for d in daily_data])
        dominant_condition = max(set([d['condition'] for d in daily_data]), key=[d['condition'] for d in daily_data].count)

        # Print daily summary
        print(f"Daily Summary for {city} on {today}:")
        print(f"Average Temp: {avg_temp:.2f}C, Max Temp: {max_temp:.2f}C, Min Temp: {min_temp:.2f}C, Dominant Condition: {dominant_condition}")

# Schedule tasks to fetch weather data and calculate daily summaries
schedule.every(INTERVAL).minutes.do(update_weather_data)  # Fetch data at regular intervals
schedule.every().day.at("23:59").do(calculate_daily_summary)  # Calculate daily summary at the end of the day

if __name__ == "__main__":
    update_weather_data()  # Fetch data once immediately
    try:
        while True:
            schedule.run_pending()
            time.sleep(1)  # Sleep to avoid busy waiting
    except KeyboardInterrupt:
        print("\nProgram terminated by user.")
```

OUTPUT:

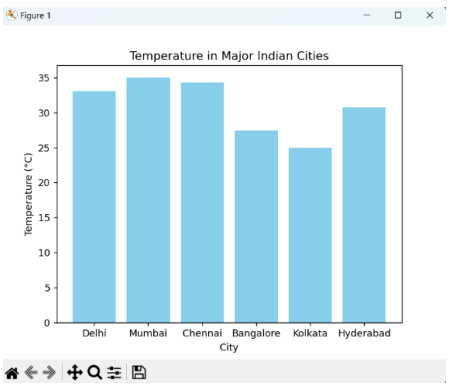
```
PS C:\Users\K LIKHITHA> & C:/Python312/python.exe "c:/Weather Report/weather_monitor.py"
API response for Delhi: {'coord': {'lon': 77.2167, 'lat': 28.6667}, 'weather': [{'id': 721, 'main': 'Haze', 'description': 'haze', 'icon': '50d'}], 'bas
ke': 304.81, 'temp_min': 306.2, 'temp_max': 306.2, 'pressure': 1008, 'humidity': 27, 'sea_level': 1008, 'grnd_level': 983}, 'visibility': 5000, 'wind':
, 'dt': 1729850771, 'sys': {'type': 1, 'id': 9165, 'country': 'IN', 'sunrise': 1729817895, 'sunset': 1729858312}, 'timezone': 19800, 'id': 1273294, 'nam
c:\Weather Report\weather_monitor.py:49: DeprecationWarning: datetime.datetime.utcfromtimestamp() is deprecated and scheduled for removal in a future ve
tetimes in UTC: datetime.datetime.fromtimestamp(timestamp, datetime.UTC).
  print(f"Weather in {city}: {weather['temp']}C, {weather['condition']} at {datetime.utcfromtimestamp(weather['timestamp'])}")
Weather in Delhi: 33.05000000000001C, Haze at 2024-10-25 10:06:11
API response for Mumbai: {'coord': {'lon': 72.8479, 'lat': 19.0144}, 'weather': [{'id': 711, 'main': 'Smoke', 'description': 'smoke', 'icon': '50d'}], '
s_like': 310.24, 'temp_min': 305.09, 'temp_max': 307.14, 'pressure': 1008, 'humidity': 46, 'sea_level': 1008, 'grnd_level': 1007}, 'visibility': 3000, '
l': 40}, 'dt': 1729850788, 'sys': {'type': 1, 'id': 9052, 'country': 'IN', 'sunrise': 1729818359, 'sunset': 1729859945}, 'timezone': 19800, 'id': 12753
Weather in Mumbai: 33.99000000000001C, Smoke at 2024-10-25 10:06:28
API response for Chennai: {'coord': {'lon': 80.2785, 'lat': 13.0878}, 'weather': [{'id': 721, 'main': 'Haze', 'description': 'haze', 'icon': '50d'}], 'b
_like': 313, 'temp_min': 305.85, 'temp_max': 308.18, 'pressure': 1006, 'humidity': 52, 'sea_level': 1006, 'grnd_level': 1005}, 'visibility': 5000, 'wind
 40}, 'dt': 1729850796, 'sys': {'type': 2, 'id': 2093220, 'country': 'IN', 'sunrise': 1729816249, 'sunset': 1729858489}, 'timezone': 19800, 'id': 126452
Weather in Chennai: 34.33000000000004C, Haze at 2024-10-25 10:06:36
API response for Bangalore: {'coord': {'lon': 77.6033, 'lat': 12.9762}, 'weather': [{'id': 804, 'main': 'Clouds', 'description': 'overcast clouds', 'ico
 300.59, 'feels_like': 301.63, 'temp_min': 299.94, 'temp_max': 301.2, 'pressure': 1009, 'humidity': 58, 'sea_level': 1009, 'grnd_level': 909}, 'visibili
gust': 3.85}, 'clouds': {'all': 100}, 'dt': 1729850876, 'sys': {'type': 2, 'id': 2017753, 'country': 'IN', 'sunrise': 1729816885, 'sunset': 1729859136},
luru', 'cod': 200}
Weather in Bangalore: 27.4399999999999998C, Clouds at 2024-10-25 10:07:56
API response for Kolkata: {'coord': {'lon': 88.3697, 'lat': 22.5697}, 'weather': [{'id': 502, 'main': 'Rain', 'description': 'heavy intensity rain', 'ic
: 297.43, 'feels_like': 298.42, 'temp_min': 297.43, 'temp_max': 297.43, 'pressure': 1002, 'humidity': 96, 'sea_level': 1002, 'grnd_level': 1001}, 'visib
 'gust': 15.42}, 'rain': {'1h': 13.52}, 'clouds': {'all': 100}, 'dt': 1729850836, 'sys': {'country': 'IN', 'sunrise': 1729814839, 'sunset': 1729856015},
ta', 'cod': 200}
Weather in Kolkata: 24.28000000000003C, Rain at 2024-10-25 10:07:16
API response for Hyderabad: {'coord': {'lon': 78.4744, 'lat': 17.3753}, 'weather': [{'id': 721, 'main': 'Haze', 'description': 'haze', 'icon': '50d'}],
ls_like': 304.96, 'temp_min': 303.88, 'temp_max': 303.88, 'pressure': 1008, 'humidity': 48, 'sea_level': 1008, 'grnd_level': 953}, 'visibility': 5000, '
ll': 40}, 'dt': 1729850765, 'sys': {'type': 1, 'id': 9213, 'country': 'IN', 'sunrise': 1729816916, 'sunset': 1729858687}, 'timezone': 19800, 'id': 12698
Weather in Hyderabad: 30.730000000000018C, Haze at 2024-10-25 10:06:05
```

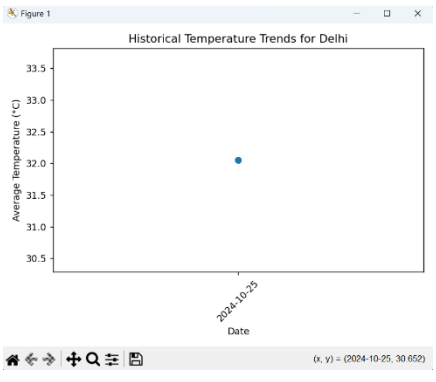
4.VISUALIZATION:

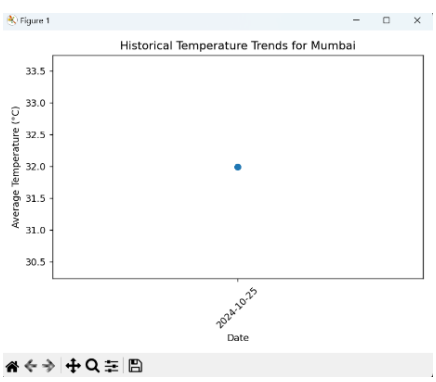
```python
def plot_daily_summaries():
    """Plot daily summaries of weather data for specified cities."""
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    cursor.execute('SELECT city, AVG(avg_temperature) AS avg_temp FROM daily_summaries GROUP BY city')
    data = cursor.fetchall()
    conn.close()

    if not data:
        print("No data found to plot.")
        return

    cities = [row[0] for row in data]
    avg_temps = [row[1] for row in data]

    plt.bar(cities, avg_temps, color='blue')
    plt.xlabel('Cities')
    plt.ylabel('Average Temperature (°C)')
    plt.title('Average Daily Temperatures')
    plt.xticks(rotation=45)
    plt.tight_layout()  # Adjust layout to prevent clipping of labels
    plt.show()
```

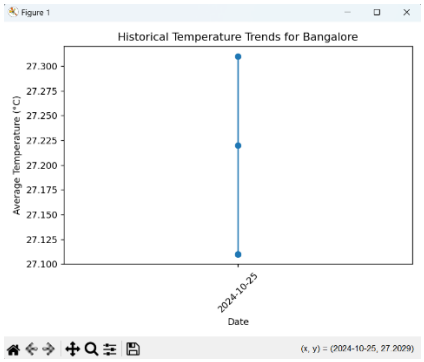
```python
def plot_historical_trends(city):
    """Plot historical temperature trends for a specific city."""
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    cursor.execute('SELECT date, avg_temperature FROM daily_summaries WHERE city = ? ORDER BY date', (city,))
    data = cursor.fetchall()
    conn.close()

    if not data:
        print(f"No historical data found for {city}.")
        return

    dates = [row[0] for row in data]
    avg_temps = [row[1] for row in data]

    plt.plot(dates, avg_temps, marker='o')
    plt.xlabel('Date')
    plt.ylabel('Average Temperature (°C)')
    plt.title(f'Historical Temperature Trends for {city}')
    plt.xticks(rotation=45)
    plt.tight_layout()  # Adjust layout to prevent clipping of labels
    plt.show()
```

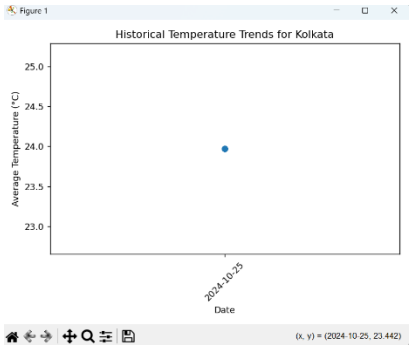
OUTPUT:

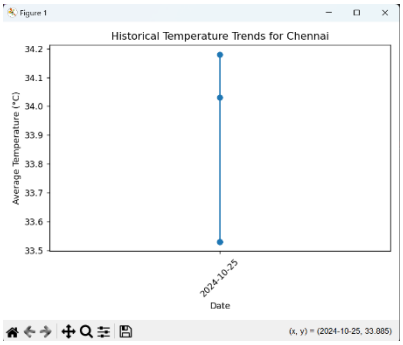












5.ALERTING:

```python
# Function to send email alerts
def send_email_alert(city, temp):
    msg = MIMEMultipart()
    msg['From'] = EMAIL_ADDRESS
    msg['To'] = TO_EMAIL
    msg['Subject'] = f"Weather Alert: High Temperature in {city}"

    body = f"The temperature in {city} has reached {temp:.2f}°C, exceeding the threshold of {ALERT_THRESHOLD_TEMP}°C."
    msg.attach(MIMEText(body, 'plain'))

    try:
        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
            server.starttls()
            server.login(EMAIL_ADDRESS, EMAIL_PASSWORD)
            server.send_message(msg)
            print(f"Email alert sent for {city} with temperature {temp:.2f}°C")
    except Exception as e:
        print(f"Failed to send email alert: {e}")
```

OUTPUT:

```
Fetching weather data...
Weather in Delhi: 32.05000000000001°C, Condition: Haze
Alert! Temperature in Delhi exceeds threshold with 32.05000000000001°C
Email alert sent for Delhi with temperature 32.05°C
Weather in Mumbai: 31.99000000000001°C, Condition: Smoke
Alert! Temperature in Mumbai exceeds threshold with 31.99000000000001°C
Email alert sent for Mumbai with temperature 31.99°C
Weather in Chennai: 33.460000000000036°C, Condition: Haze
Alert! Temperature in Chennai exceeds threshold with 33.460000000000036°C
Email alert sent for Chennai with temperature 33.46°C
```