Project Title: Weather Data Collector and Database Inserter

Description:

This Python script fetches weather data for specified cities from the OpenWeatherMap API and inserts it into a PostgreSQL database table.

Functionalities:

- Loads configuration settings from a YAML file (info.yaml)
- Fetches weather data for a city and country using the OpenWeatherMap API
- Transforms the raw JSON response into a pandas DataFrame
- Creates a table named "weather_data" in a PostgreSQL database (if it doesn't exist)
- Inserts the weather data from the DataFrame into the database table

Dependencies:

- requests: Used to make HTTP requests to the OpenWeatherMap API
- time: Used for potential delays between API requests
- logging: Used for logging informational and error messages
- pandas: Used for data manipulation and DataFrame creation
- datetime: Used for working with date and time data
- psycopg2: Used for connecting to and interacting with the PostgreSQL database
- yaml: Used for parsing YAML configuration files

Configuration:

The script relies on a YAML configuration file named "info.yaml" which should contain the following keys:

- `api_key`: Your OpenWeatherMap API key
- `url`: The OpenWeatherMap API endpoint URL for weather data retrieval
- `sql_password`: The password for your PostgreSQL database user (replace with your actual password)
- (Optional) `city_country_pairs`: A list of dictionaries containing city and country code pairs for weather data retrieval. If not provided, defaults to London, UK.

Execution:

1. Save the script as a Python file (e.g., weather_collector.py)

2. Ensure you have the required libraries installed (`pip install requests pandas psycopg2 etc.`)
3. Create the configuration file "info.yaml" with your API key, database password, and optional city-country pairs.
4. Run the script from the command line: `python weather_collector.py`

Output:

The script will print the combined DataFrame containing all fetched weather data to the console. Additionally, it will log informational and error messages to a file named "weather_data.log".

Database:

The script connects to a PostgreSQL database and creates a table named "weather_data" (if it doesn't exist already) with the following columns:

- `timestamp`: Timestamp of the weather data (TIMESTAMP)
- `city`: Name of the city (VARCHAR(100))
- `country`: Country code of the city (VARCHAR(100))
- `temperature`: Temperature in Celsius (FLOAT)
- `humidity`: Humidity percentage (INTEGER)
- `wind_speed`: Wind speed in meters per second (FLOAT)
- `weather_description`: Description of the weather (VARCHAR(200))

The script then inserts the fetched weather data into this table.

Cron job

Set Up Cron Job:

- Open a terminal window and edit your crontab using the following command:
- In Bash run

```
crontab -e
```

- This will open a text editor where you can define your cron job schedule.

Define Cron Job Schedule:

- In the crontab file, add a line following the cron syntax to specify how often you want the script to run. Here are some common examples:
    - **Run every minute:**

```
* * * * * /path/to/python3 /path/to/weather_data.py >>
/path/to/weather_data.log 2>&1
```

    - **Run every hour:**

```
0 * * * * /path/to/python3 /path/to/weather_data.py >>
/path/to/weather_data.log 2>&1
```

    - **Run at a specific time (e.g., 8:00 AM):**

```
0 8 * * * /path/to/python3 /path/to/weather_data.py >>
/path/to/weather_data.log 2>&1
```

    - **Explanation of the cron syntax:**
        - The first five fields represent (minute, hour, day of month, month, day of week)
        - `*` in a field means "every"
        - `>>` redirects standard output and standard error to a log file (`weather_data.log` in this example)
        - `2>&1` combines standard output and standard error

Save and Close Crontab:

- Save the crontab file (usually by pressing `Ctrl+O` and then `Enter`).
- Exit the editor (usually by pressing `Ctrl+X`).

If just Test the Script (Optional):

- can manually run the script using:
- In Bash can run following command:

```
/path/to/python3 /path/to/weather_data.py
```