Take Home Assignment

Objective:

Design a data pipeline to pull the weather information, store and analyze the result. This assignment will test your programming skills and the ability to interact with external systems using API endpoints. Please make sure your code is robust, maintainable and well documented.

Dataset

Using the National Weather public endpoint, we can read the weather data from various locations(stations). Below are relevant API for this exercise.

1. Available stations can be found using `/stations` endpoint.
2. Weather observations at a station can be fetched using `/observations` endpoint.

Other technical details can be found in the website's **Specifications** tab.

Tasks:

1. Pick your favorite programming language and database for this exercise. 2. Write a data pipeline to get weather information for a single station of your choice (Sample station ids: `0112W`, `0128W`).

3. Load the data into the database table(s). Database must store at least below data points:
    - Station Id
    - Station name
    - Station Timezone
    - Latitude/Longitude
    - Observation timestamp
    - Temperature (round to two decimal places)
    - Wind Speed (round to two decimal places)
    - Humidity (round to two decimal places)
4. Use the stored data and write SQL statements for below metrics
    a. Average observed temperature for last week(Mon-Sun).

b. Maximum wind speed change between two consecutive observations in the last 7 days.

Pipeline specifications

1. Process last 7-days data into the database table(s) when running the pipeline for the first time. Consequent runs must insert only the new records.
2. Re-running the pipeline multiple times, must not result in duplicate records in the table.
3. Feel free to assume other details as required. Document your assumptions.

Deliverables:

A GitHub repository or a zip file containing:

- Source code and database script files.
- Any configuration files or instructions needed to run your code.

Evaluation Criteria:

- **Functionality**: Does the solution correctly ingest, transform, and analyze the data as required?
- **Error handling**: Does the code handle errors gracefully?
- **Efficiency**: Are the ingestion and transformation processes optimized for performance?
- **Code Quality**: Is the code clean, well-organized, and documented? Are best practices followed?
- **Documentation**: Is the documentation clear and comprehensive?