Data Analyst Assignment

Business Context

You are part of the tech-team developing energy management solutions for small and medium-sized buildings using IoT. Your task as a Data Analyst is to derive actionable insights from sensor data collected from multiple sites. The sensor data is critical for optimizing energy consumption, identifying faults, and ensuring system stability.

The task simulates real-world challenges where raw IoT data needs to be cleaned, analyzed, and visualized to provide insights to stakeholders. Your deliverables will be integrated into the backend Django based application framework(s) developed by the development team.

Dataset

The provided dataset IoT_Sensor_Data.csv contains two weeks of data from two **sites** with multiple **meters** and sensors measuring different parameters.

Columns in the dataset:

1. site_id: Unique identifier for the site (e.g., Site_A, Site_B).
2. meter_id: Unique identifier for meters under each site.
3. timestamp: Date and time of the reading (recorded every 5 minutes).
4. sensor_name: Type of sensor (e.g., temperature, power_consumption, humidity).
5. sensor_value: Value recorded by the sensor.
6. status: Status of the sensor (active, inactive, error).

Note: The full dataset has been generated and is attached in the ZipFile.

Tasks

1. **Database Design**
    a. Use SQLite/Postgres or any SaaS based database like Superbase for this assignment
    b. Create tables with the appropriate schema
2. **Create a Modular & Object-Oriented Python Solution:**
    a. Follow Object-Oriented Programming principles to design your solution. Ensure your solution is modular and reusable.
    b. Class 1: DataLoader:
        i. Loads data from provided csv
        ii. Data cleaning & pre-processing: Identify and handle missing or inconsistent data. Flag and filter out invalid sensor_value readings (e.g., negative values for temperature or power consumption).
        iii. Load the cleaned data into database
        iv. Ensure the function(s) can handle potential errors (e.g., duplicate rows, missing columns).
    c. Class 2: DatabaseHandler: Handles database connection & loads clean data into database.

d. SQL Task: Write SQL queries (from within your Python script) to:
    i. Find the **total power consumption** for each site over the two-week period.
    ii. Identify **meters contributing the highest power usage** at each site.
    iii. Identify any missing timestamps in the data for each meter.
e. Class 2: DataPreProcessor: Processes data using Pandas and NumPy.
    i. Fetch data from the database using SQL and process it using Pandas.
    ii. Perform the following operations:
        1. Normalize the sensor values (e.g., scale power values between 0 and 1).
        2. Create a time-series DataFrame with timestamps as the index.
        3. Fill missing timestamps with appropriate interpolated values.
f. Class 3: MetricsCalculator: Calculates metrics such as averages, maximums, and daily sums and store them in relevant table
g. Class 4: GraphGenerator: Generate relevant plots/visualizations from the metrics data and store them as PNGs
h. Class 5: DataReportGenerator: Calculated Metrics be exported into CSV

3. **Dockerize Your Solution (Bonus)**
    a. Write a `Dockerfile` to containerize your Python solution.
    b. Ensure the Docker container includes:
        i. Python runtime environment.
        ii. All required dependencies (e.g., Pandas, NumPy, SQLite3).
        iii. Instructions to run your Python script inside the container.
4. **Documentation**
    a. Provide clear documentation explaining:
        i. Steps taken in data cleaning and analysis.
        ii. SQL queries with explanations.
        iii. Challenges faced and how you overcame them.

Deliverables

1. GitHub repository containing a concise documentation file (README.md) including steps to execute the solution.
2. Python code (with comments and modular structure).
3. Generated plots/visualizations as PNGs stored in the relevant folder.
4. Generated Report stored as csv in the relevant folder.
5. Dockerized solution with a Dockerfile. (Bonus)
6. Demo video (if possible, or if shortlisted could be demonstrated during interview)

Evaluation Criteria

1. **Coding Standards**: Code readability, comments, modularity, and use of best practices.
2. **Analytical Rigor**: The ability to derive meaningful insights from the data.
3. **Problem Solving**: Handling edge cases, data anomalies, and performance optimization.
4. **Deployment Readiness**: A fully functional and Dockerized solution.
5. **Documentation**: Clear explanations of steps, challenges, and outcomes.
6. **Demo or Demo Video:** Showcasing the running of the solution

---