Case Study Data & Process Analytics

Background

At Greenlyte, we operate a direct air capture system that removes CO_2 from the atmosphere while producing hydrogen. To optimize this complex chemical process, we collect large amounts of time-series data from different sensors in the plant.

This case study is designed to simulate a realistic one-day challenge. It focuses on moving data from initial collection all the way to basic post-processing, using tools that are available via free trials or free tiers.

Objective

Set up a minimal, end-to-end data pipeline to handle plant time-series data. The goal is to simulate real-world signals, package and store the data effectively, and run a basic processing job on it.

Step 1: Ingest Data to S3

Tasks:

- Generate time-series data in Node-Red (in JSON or CSV packets).
- Simulate eight different sensors:
    - pH
    - Conductivity

    - Pressure
    - Temperature
    - Fill level
    - Flow
    - Humidity
    - CO_2 concentration

- Decide on a clear packet size and structure. Time period and interval?

- Clearly define a file naming convention.

- Push these data files to an AWS S3 bucket. You can use the AWS free tier, or if not available, simulate this with a local folder structure that mirrors S3.

- Document your S3 bucket structure
- Include any lifecycle or retention policies you would suggest

Deliverables:

- A small code snippet or Node-RED flow export that shows how you generate the data packets and handle the upload to S3 (or simulated local structure).

- A short diagram (can be a sketch or text diagram) showing the S3 folder structure and data flow.

Scope Estimate: ~3 hours

Step 2: Post-Processing & First Pipeline

Tasks:

- Choose a processing tool that is available in a free tier or as a trial. Examples include AWS Lambda (free tier), Google Cloud Functions (free tier), or Databricks Community Edition.

- Build a simple pipeline that does the following:

    - Reads data packets from S3.

    - Performs basic preprocessing tasks:

        - Calculate mean and standard deviation for each sensor in each packet.

        - Flag any out-of-range values

        - Check for missing timestamps or any irregular gaps in data.

    - Optionally, write a cleaned or summarized version of the data back to a different folder in S3

Documentation:

- Explain why you chose the specific processing tool and how it would scale.

- Describe how you would trigger the job (e.g., automatically when a new file lands in S3, or on a schedule).

- Include any additional steps you would consider adding if you had more time (e.g., data quality monitoring, error alerts).

Deliverables:

- A small code snippet or function logic that shows your preprocessing steps.

- A short written explanation of the tool choice and job design.

Scope Estimate: ~3 hours

Discussion Points

- How did you decide on the packet size and structure?

- What were the main factors in choosing your processing tool?

- How would you ensure data quality and handle unexpected sensor behavior?

- What improvements would you make to the pipeline if you had additional time (e.g., dashboards, anomaly detection, predictive models)?

Expected Outputs

Deliverable	Description
Node-RED setup	Description of sensors, packet strategy, and upload logic.
S3 ingestion plan	File naming, folder structure, and retention strategy.
Processing pipeline plan	Code snippet and explanation of preprocessing approach.

Final Note

This case study isn’t about building a perfect or production-ready system. The focus is on clear thinking, practical design, and showing how you break down a complex data flow into actionable steps — all using free or trial tools only.