SDE-2

Overview

As an **SDE-2**, your task is to build a production-grade data pipeline that ingests, transforms, validates, and stores agricultural sensor data coming from multiple sources. The goal is to deliver clean, enriched, and query-optimized data for downstream analytics.

Our agricultural monitoring system collects data from sensors placed across farmlands. These sensors track parameters such as **soil moisture**, **temperature**, **humidity**, **light intensity**, and **battery levels**. The raw data is stored in Parquet files, split by date and sensor type. Your job is to build a pipeline to process, validate, and enrich this data.

Tasks

1. **Data Ingestion**
   Build a modular ingestion component that:
   - Reads daily Parquet files from a `data/raw/` directory (local or mounted cloud storage).
     Example file: `data/raw/2023-06-01.parquet`
     `(`[`https://drive.google.com/file/d/1JzvmQU1ETr4MBgOzTYbpjEm43QVOwXIW/view?usp=sharing`](https://drive.google.com/file/d/1JzvmQU1ETr4MBgOzTYbpjEm43QVOwXIW/view?usp=sharing)`)`
     Schema:
     ```
     sensor_id (string)
     timestamp (ISO datetime)
     reading_type (string)  -- e.g., temperature, humidity
     value (float)
     battery_level (float)
     ```
   - Supports **incremental loading** (e.g., using file naming or timestamp-based checkpointing).
   - Uses **DuckDB** to:
     - Inspect the file schema quickly.
     - Run validation queries (e.g., missing columns, data ranges).
     - Log ingestion summary via DuckDB SQL aggregation.
   - Handles and logs:
     - Corrupt or unreadable files
     - Schema mismatches
     - Missing or invalid values
   - **Logs ingestion statistics**:
     - Number of files read

        - Records processed
        - Records skipped/failed

2. **Data Transformation**

   Apply the following transformation logic:

    - **Data cleaning**:
        - Drop duplicates
        - Fill or drop missing values
        - Detect and correct outliers (e.g., z-score > 3)
    - **Derived fields**:
        - Daily average value per sensor and reading_type
        - 7-day rolling average per sensor
        - `anomalous_reading = true` if value outside expected range (define a simple range per reading_type)
        - Normalize using calibration logic (e.g., `value = raw_value * multiplier + offset` — hardcode/sample calibration params)
    - **Timestamp processing**:
        - Convert to ISO 8601 format
        - Adjust to UTC+5:30

3. **Data Quality Validation (Using DuckDB)**

   Use **DuckDB queries and/or validation framework** to:

    - Validate types (e.g., value is float, timestamp is ISO format)
    - Check expected value ranges per `reading_type`

      Detect **gaps in hourly data** (use DuckDB `generate_series` to simulate expected times)
    - Profile:
        - % missing values per reading_type
        - % anomalous readings
        - Time coverage gaps per sensor

   Save the validation output as `data_quality_report.csv`

4. **Data Loading & Storage**
    - Store cleaned and transformed data in **Parquet** format under `data/processed/`
    - Optimize for analytical queries:
        - Columnar format
        - Partition by `date` and optionally by `sensor_id`
        - Apply compression (e.g., `snappy`, `gzip`)

Deliverables

1. **Codebase** in **any language of your choice** that implements the pipeline described above:
    - Modular structure for ingestion, transformation, validation, and loading
    - Code should be clean, documented, and testable
2. **Unit tests** for core logic (e.g., transformation, anomaly detection, validation)
3. **Unit test coverage**
4. **Docker setup**:
    - Dockerfile to build and run your code locally
5. **Sample Data**:
    - Include at least 3 Parquet files in `data/raw/` (e.g., 3 different dates) to demonstrate your pipeline
6. **Documentation**:
    - README with:
        - Setup & run instructions
        - Overview of architecture and components
        - Explanation of calibration & anomaly logic
    - Example of generated **data quality report**

7. **Submission**:
    - Push your code and sample data to a **public GitHub repository**
    - Include a short demo video
        - Ingestion
        - Transformation
        - Validation
        - Sample query on DuckDB

For any doubts or queries, reach out to us at : aravindhan@satsure.co