Senior Data Engineer – Technical Take-Home Assessment

Overview

This exercise is designed to assess your data engineering design, implementation, and reasoning skills. We are not looking for perfection or production-scale completeness, but for clarity of thought, correctness, and engineering judgement.

Estimated time: 4–6 hours
Tools: Use open-source tools only
Submission: Git repository (GitHub / GitLab / zip)

Scenario

You are working on a platform that ingests high-volume IoT telemetry data from multiple devices in near-real time.

The data must:

- Be ingested reliably
- Support schema evolution
- Serve operational queries and analytics workloads
- Maintain data quality and traceability

Part 1: Data Ingestion & Streaming Design (Conceptual)

Task

Design a streaming ingestion architecture for the following requirements:

- Data is published from devices every few seconds
- Data arrives via Kafka
- Data must be stored in:
    - A time-series operational store (e.g. TimescaleDB)
    - A data lake / analytical store (e.g. Iceberg)
- Downstream consumers include BI and ML teams

Deliverables

- Architecture diagram (PNG, PDF, or Markdown)
- Short written explanation (1–2 pages max)

You should cover:

- Topic design & partitioning strategy
- Schema management and evolution

- Error handling and replay strategy
- Performance and scalability considerations
- How BI / ML teams consume the data

Part 2: Practical Implementation (Core Task)

Input Data

You are given Kafka-like JSON events (you may simulate Kafka input with files or a producer script).

Example event:

```
{
  "device_id": "sensor-123",
  "timestamp": "2025-01-20T10:15:30Z",
  "temperature": 21.4,
  "humidity": 68,
  "firmware_version": "v1.2.0"
}
```

Later, the schema evolves:

```
{
  "device_id": "sensor-123",
  "timestamp": "2025-01-20T10:16:00Z",
  "temperature": 21.6,
  "humidity": 67,
  "pressure": 1013,
  "firmware_version": "v1.3.0"
}
```

Task

Implement a data ingestion and transformation pipeline that:

1. Ingests the events (simulated Kafka input is acceptable)
2. Handles schema evolution gracefully
3. Writes data to:
   - A time-series table (simulated TimescaleDB / Postgres is acceptable)
   - A lake-style table (Parquet files or Iceberg if you prefer)
4. Ensures data quality checks:
   - Required fields present

    - Valid timestamps
    - Reasonable value ranges (basic validation)

Technical Expectations

You may use:

- Python (preferred)
- SQL
- Kafka / Kafka-Connect (simulated is fine)
- Docker / Docker Compose (optional but encouraged)
- Kubernetes (optional but encouraged)

Focus on:

- Clear structure
- Idempotent processing
- Readability and maintainability

Part 3: Data Quality & Monitoring

Task

Implement basic data quality monitoring.

At minimum:

- Log ingestion failures
- Flag invalid records
- Produce a simple data quality summary (table, log output, or report)

Part 4: Migration & Transformation

Task

Write a **migration or transformation script** that:

- Reads historical data from the lake layer
- Transforms it into an operational-friendly format
- Loads it into the time-series store

Explain:

- Why this transformation is needed

- How you ensure correctness and performance

Part 5: Documentation & Reasoning

Required

Include a README.md covering:

- How to run the solution
- Key design decisions
- Trade-offs you made
- What you would improve with more time
- Bonus point for unit testing