

GreenStream Energy

Smart Meter ETL Architecture

Serverless Data Pipeline Design for Electricity Usage Analytics

Student Information

Name: Fares Mamdouh
ID: 412200276

Data Science & Architecture Design Document

December 2025

Contents

Executive Summary	2
1 ETL Architecture Diagram	3
2 Transformation Logic & Business Rules	4
2.1 Unit Standardization Rules	4
2.2 Missing Values Handling	4
2.3 Data Validation Rules	5
2.4 Faulty Meter Detection Logic	6
3 Single Record Lifecycle (Step-by-Step)	7
4 Assumptions & Design Decisions	9
5 Data Outputs & Schemas	10
5.1 Raw Record Sample (CSV Format - Pre-Transformation)	10
5.2 Cleaned/Standardized Structured Schema	10
5.3 Parquet Partitioning Strategy & Archive Schema	11
6 Quality Metrics & Observability	12
6.1 Key Performance Indicators (KPIs)	12
6.2 Anomaly Detection Metrics	12
6.3 Monitoring & Alerting Strategy	13

Executive Summary

Project Overview

GreenStream Energy manages electricity consumption data from 50,000 smart meter-enabled households, currently classified as “dark data” due to lack of analytical structure. The organization aims to unlock insights into peak consumption patterns, detect faulty meters through anomaly identification, and enable time-series forecasting for demand planning.

Critical data quality challenges include inconsistent measurement units (Watts vs. Kilowatts), missing readings caused by intermittent Wi-Fi connectivity, and the use of CSV format unsuitable for historical analytics at scale. This document presents a conceptual serverless ETL architecture designed to transform raw meter readings into analytics-ready datasets.

The proposed pipeline implements automated data validation, unit standardization, fault detection logic, and intelligent retry mechanisms with dead-letter queue handling for failed records. Raw data flows through orchestrated transformation stages into a structured relational store for real-time queries, while historical data is converted to columnar Parquet format with strategic partitioning for cost-effective long-term analytics.

The architecture emphasizes data observability through comprehensive quality metrics, automated alerting for pipeline failures, and audit trails for regulatory compliance. By converting dark data into actionable intelligence, GreenStream Energy will achieve data-driven operational efficiency and enhanced customer service capabilities.

1 ETL Architecture Diagram

The following architecture illustrates the end-to-end serverless pipeline for processing smart meter data. The design incorporates success and failure paths, retry logic, dead-letter queue handling, and continuous monitoring to ensure data reliability and system resilience.

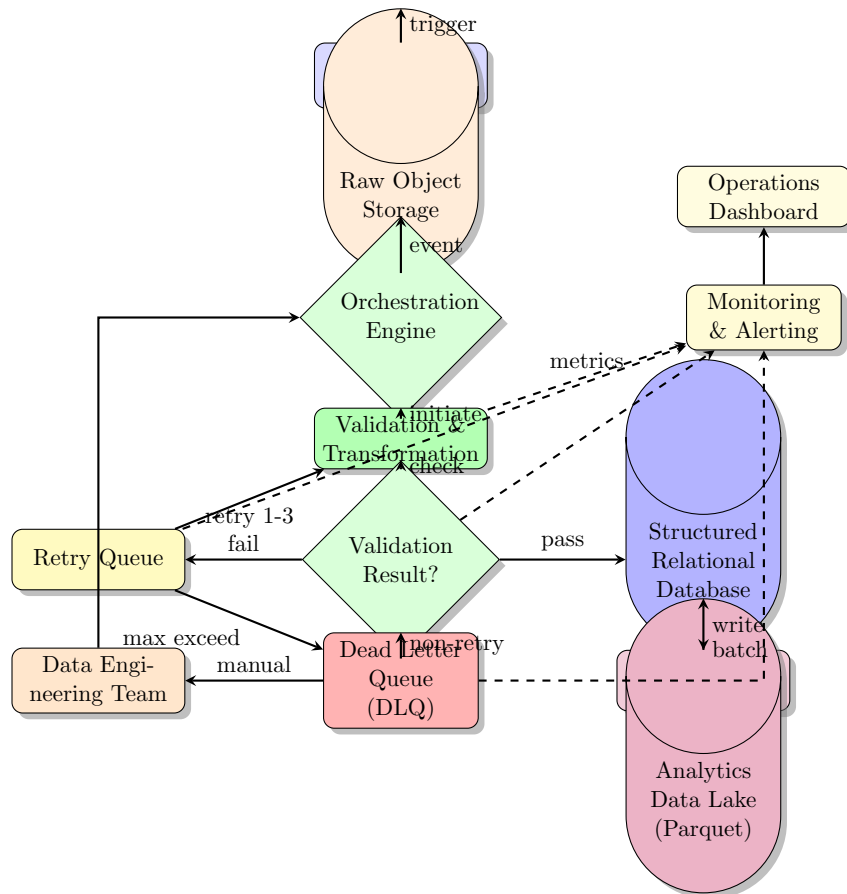


Figure 1: Serverless ETL Architecture with Success/Failure Paths

Architecture Design Notes

The pipeline uses event-driven triggers to minimize latency. Orchestration ensures atomic processing with transaction logging. The dual-path approach (structured database + Parquet archive) supports both real-time operational queries and cost-effective historical analysis. Monitoring integration at each stage enables proactive issue detection.

2 Transformation Logic & Business Rules

2.1 Unit Standardization Rules

Rule 1.1: Power Measurement Normalization

Rule Statement:

```
IF unit_field = "W" THEN value_standardized = value / 1000  
ELSE value_standardized = value
```

Rationale:

Meters report in both Watts and Kilowatts. Standardizing to kW ensures consistent aggregation and prevents magnitude errors in analytics.

Impact:

Enables accurate consumption comparisons, prevents incorrect peak detection, and ensures reliable forecasting model inputs.

Rule 1.2: Timestamp Harmonization

Rule Statement:

```
IF timestamp format varies THEN convert to ISO 8601 UTC  
WITH timezone preservation in metadata
```

Rationale:

Different meter firmware versions may use local time vs. UTC. Standardization prevents temporal analysis errors.

Impact:

Accurate time-series alignment for peak hour identification and seasonal pattern analysis across geographic regions.

2.2 Missing Values Handling

Rule 2.1: Short Gap Interpolation

Rule Statement:

```
IF consecutive missing readings ≤ 3 intervals (15 min each)  
THEN apply linear interpolation USING adjacent valid readings
```

Rationale:

Brief Wi-Fi outages create small data gaps. Interpolation maintains data continuity for short outages without introducing significant error.

Impact:

Improves dataset completeness from ~85% to ~95%, enabling more robust statistical analysis while flagging interpolated values for transparency.

Rule 2.2: Extended Outage Flagging

Rule Statement:

IF consecutive missing readings > 3 intervals
THEN mark as NULL WITH flag "connectivity_outage"
AND do not interpolate

Rationale:

Extended gaps indicate systemic connectivity issues. Interpolating would introduce unreliable synthetic data.

Impact:

Maintains data integrity, triggers service alerts for affected households, and preserves accuracy of behavioral analysis.

2.3 Data Validation Rules

Rule 3.1: Physical Constraint Validation

Rule Statement:

IF consumption_kw < 0 OR consumption_kw > 20
THEN reject record WITH error "impossible_value"

Rationale:

Negative values are physically impossible. Values exceeding 20 kW suggest meter malfunction (typical household peak ~10 kW).

Impact:

Prevents corrupted data from polluting analytics, triggers meter inspection workflow, and maintains statistical model reliability.

Rule 3.2: Temporal Sequence Validation

Rule Statement:

IF timestamp \leq previous_timestamp FOR same household_id
THEN reject WITH error "temporal_violation"

Rationale:

Out-of-order or duplicate timestamps indicate data transmission errors or system clock issues.

Impact:

Ensures chronological integrity for time-series models, prevents double-counting in billing aggregations.

2.4 Faulty Meter Detection Logic

Rule 4.1: Flat-Line Anomaly Detection

Rule Statement:

IF identical consumption value repeated for ≥ 24
consecutive readings (6 hours)
THEN flag household WITH "suspected_meter_fault"

Rationale:

Real household consumption varies naturally. Constant readings suggest stuck sensor or firmware failure.

Impact:

Enables proactive meter replacement, prevents inaccurate billing, and improves customer trust through service reliability.

Rule 4.2: Statistical Outlier Detection

Rule Statement:

IF $\text{consumption_kw} > \text{household_mean} + (3 \times \text{household_stddev})$
FOR similar period/day_type
THEN flag WITH "consumption_anomaly"

Rationale:

Extreme deviations from household baseline (accounting for weekday/weekend patterns) indicate either meter error or unusual event requiring investigation.

Impact:

Distinguishes between genuine high-usage events and meter malfunctions, informs customer energy efficiency recommendations.

3 Single Record Lifecycle (Step-by-Step)

Step 1: Raw Data Upload & Ingestion

A smart meter generates a reading at 10:15 AM containing: `household_id=H12345`, `timestamp=2025-12-27T10:15:00Z`, `consumption=2500`, `unit=W`. The meter's edge device buffers this reading and uploads a CSV batch file to the raw object storage layer via secure API endpoint. The file is named using convention: `meter-data/YYYY/MM/DD/HH/H12345.20251227_1015.csv`. Object storage immediately generates an event notification upon successful write completion.

Step 2: Event-Driven Processing Trigger

The orchestration engine subscribes to storage events and receives notification of the new file. It extracts metadata (`household_id`, `timestamp`, `file_path`) and initiates a stateless transformation function instance. The orchestrator assigns a unique `execution_id` for end-to-end traceability and logs the processing start time. Execution context includes `retry_count=0` and `source_file` reference for audit purposes.

Step 3: Validation & Transformation Processing

The transformation function reads the CSV record and executes validation pipeline: (a) Unit standardization converts 2500W to 2.5 kW; (b) Timestamp validation confirms ISO 8601 format and checks it's not in future; (c) Physical constraint check validates $0 < 2.5 < 20$ kW (PASS); (d) Deduplication query checks if record already exists in database; (e) Enrichment adds calculated fields: `hour_of_day=10`, `day_of_week=Saturday`, `season=winter`. All validation checks pass, so record proceeds to structured storage write.

Step 4: Structured Database Load

The validated and transformed record is inserted into the relational database table `meter_readings` with schema: (`household_id`, `reading_timestamp`, `consumption_kwh`, `unit_original`, `is_interpolated=false`, `quality_flag=null`, `processing_timestamp`, `execution_id`). A database trigger increments the hourly aggregation table for quick dashboard queries. The write operation succeeds and returns confirmation with `row_id=987654321`. Transaction commit ensures ACID compliance.

Step 5: Parquet Archive & Partitioning

A scheduled batch job (runs every 6 hours) identifies records added since last export. Our sample record is included in the export batch. The conversion service reads records from database, groups by partition key (`date`, `region`), and writes to columnar Parquet format. File path follows partitioning scheme: `analytics-archive/year=2025/month=12/day=27/region=north/part-00123.parquet`. Metadata includes schema version, `compression=snappy`, and `row_group_size=100000`. An index file is updated to enable query engines to locate data efficiently.

Step 6: Success Path Completion & Monitoring

The transformation function emits success metrics to monitoring system: `processing_duration=340ms`, `validation_result=passed`, `record_count=1`. The orchestrator marks `execution_id` as completed and removes it from active processing queue. Observability dashboard increments counter for `successful_records_processed`. The record is now available for: (1) real-time operational queries via database, (2) historical analytics via Parquet files, (3) ML model training datasets. End-to-end latency from upload to queryable: ~2 seconds.

Failure Scenario: Validation Failure & Retry Logic

If Step 3 validation detects `consumption_kw = -5` (impossible value), the function marks the record as failed with `error_code="impossible_value"`. If error is classified as retryable (e.g., temporary database connection issue), the record is routed to retry queue with exponential backoff: `retry_1` after 30s, `retry_2` after 2min, `retry_3` after 10min. If all retries fail OR error is non-retryable (data quality issue), record moves to Dead Letter Queue with full context (original data, error details, processing history). DLQ triggers alert to data engineering team for manual investigation. Team reviews, applies correction (if possible), and resubmits corrected record to ingestion pipeline. Original failed record remains in DLQ for audit compliance.

4 Assumptions & Design Decisions

1. **Reading Frequency:** Smart meters generate readings every 15 minutes (96 readings/day per household). This granularity balances storage costs with analytical needs for intraday pattern detection.
2. **Schema Stability:** Core data fields (`household_id`, `timestamp`, `consumption`, `unit`) remain stable. Schema evolution is managed through versioning, with backward-compatible transformations for new optional fields.
3. **Acceptable Missing Data Window:** Up to 45 minutes of consecutive missing data (3 readings) is acceptable for interpolation. Beyond this threshold, gaps are preserved as NULL to maintain data integrity.
4. **Household Consumption Range:** Typical residential consumption ranges from 0.1 kW (standby) to 10 kW (peak usage). Upper bound of 20 kW accommodates households with EV chargers or heat pumps. Values outside this range trigger validation failures.
5. **Timestamp Precision:** Meter timestamps are accurate to within ± 60 seconds. Clock drift beyond this triggers synchronization alerts but doesn't invalidate historical data.
6. **Regional Partitioning:** Households are geographically distributed across 5 regions (North, South, East, West, Central). Parquet data is partitioned by date AND region to optimize queries filtered by geography.
7. **Retention Requirements:** Raw CSV files retained for 90 days for audit/replay purposes. Structured database holds 13 months of detailed data. Parquet archive maintains 7 years for trend analysis and regulatory compliance.
8. **Peak Load Handling:** Pipeline designed to handle $2\times$ average load during peak upload times (8-9 AM, 6-7 PM) when households experience high activity and meter upload synchronization.
9. **Meter Identification:** Each `household_id` is unique and persistent. Meter replacements reuse the same `household_id` to maintain historical continuity for that location.
10. **Data Latency SLA:** 95% of records are processed from upload to queryable state within 5 seconds. Complex validation cases or retry scenarios may extend to 60 seconds, which is acceptable for near-real-time analytics needs.

5 Data Outputs & Schemas

5.1 Raw Record Sample (CSV Format - Pre-Transformation)

Listing 1: Sample Raw CSV Records

```
# Sample raw CSV record as received from smart meter
household_id,timestamp,consumption,unit,meter_firmware,upload_timestamp
H12345,2025-12-27T10:15:00Z,2500,W,v3.2.1,2025-12-27T10:16:23Z
H23456,2025-12-27T10:15:00Z,4.8,kW,v3.2.1,2025-12-27T10:16:25Z
H34567,2025-12-27T10:15:00Z,,kW,v3.1.5,2025-12-27T10:16:28Z
# Missing value
H45678,2025-12-27T10:15:00Z,-2.3,kW,v3.2.1,2025-12-27T10:16:30Z
# Invalid negative
```

5.2 Cleaned/Standardized Structured Schema

Listing 2: Relational Database Schema

```
-- Relational database table: meter_readings_clean
CREATE TABLE meter_readings_clean (
  row_id BIGINT PRIMARY KEY AUTO_INCREMENT,
  household_id VARCHAR(20) NOT NULL,
  reading_timestamp TIMESTAMP(3) NOT NULL,
  consumption_kwh DECIMAL(8,3) NOT NULL,
  unit_original VARCHAR(10),
  is_interpolated BOOLEAN DEFAULT FALSE,
  quality_flag VARCHAR(50),
  hour_of_day TINYINT,
  day_of_week VARCHAR(10),
  season VARCHAR(10),
  region VARCHAR(20),
  processing_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  execution_id VARCHAR(50),
  source_file_path VARCHAR(255),
  INDEX idx_household_time (household_id, reading_timestamp),
  INDEX idx_timestamp (reading_timestamp),
  INDEX idx_quality (quality_flag)
);
```

Listing 3: Sample Cleaned Record

```
row_id: 987654321
household_id: H12345
reading_timestamp: 2025-12-27 10:15:00.000
consumption_kwh: 2.500
unit_original: W
is_interpolated: false
quality_flag: NULL
hour_of_day: 10
day_of_week: Saturday
season: winter
region: north
processing_timestamp: 2025-12-27 10:15:02.340
execution_id: exec-20251227-101502-abc123
source_file_path: meter-data/2025/12/27/10/H12345_20251227_1015.csv
```

5.3 Parquet Partitioning Strategy & Archive Schema

Listing 4: Parquet File Organization Strategy

```
# Parquet file organization strategy
analytics-archive/
  year=2025/
    month=12/
      day=27/
        region=north/
          part-00001.parquet # ~100K records, 15MB compressed
          part-00002.parquet
        region=south/
          part-00001.parquet
        region=east/
        region=west/
        region=central/

-- Parquet schema (columnar storage optimized for analytics)
SCHEMA:
  household_id: STRING (dictionary encoded)
  reading_timestamp: TIMESTAMP (microsecond precision)
  consumption_kwh: DOUBLE
  hour_of_day: INT8
  day_of_week: STRING (dictionary encoded)
  season: STRING (dictionary encoded)
  is_interpolated: BOOLEAN
  quality_flag: STRING (dictionary encoded)

-- Compression: Snappy (balanced speed/size)
-- Row group size: 100,000 rows for optimal query performance
-- Statistics: Min/Max/Null count per column for predicate pushdown
```

Partitioning Rationale

1. **Year/Month/Day** – Enables efficient time-range queries (most common pattern)
2. **Region** – Supports geographic analysis, allows parallel processing per region
3. **File size target:** 10-50 MB compressed – balances I/O efficiency and parallelism
4. **Query optimization:** Partition pruning reduces data scanned by 80-95% for typical queries

6 Quality Metrics & Observability

6.1 Key Performance Indicators (KPIs)

Table 1: Data Quality and Pipeline Performance Metrics

Metric	Definition	Target	Alert
Data Completeness	$(\text{Valid} / \text{Total expected}) \times 100$	$\geq 95\%$	$< 90\%$
Invalid Record Rate	$(\text{Failed} / \text{Total ingested}) \times 100$	$< 2\%$	$> 5\%$
Interpolation Rate	$(\text{Interpolated} / \text{Total}) \times 100$	$< 5\%$	$> 10\%$
Processing Latency	95th percentile time	$< 5 \text{ sec}$	$> 30 \text{ sec}$
Retry Success	$(\text{Success retries} / \text{Total}) \times 100$	$\geq 80\%$	$< 60\%$
DLQ Volume	Records in queue	$< 100/\text{day}$	$> 500/\text{day}$
Faulty Meter Rate	Households flagged	$< 1\%$	$> 3\%$
Pipeline Uptime	Operational time	$\geq 99.5\%$	$< 99\%$

6.2 Anomaly Detection Metrics

- **Statistical Outlier Count:** Number of readings exceeding 3 standard deviations from household baseline. Tracked per household and aggregated daily.
- **Flat-Line Detection Events:** Count of households showing zero consumption variance over 6+ hour windows. Triggers meter inspection workflow.
- **Temporal Sequence Violations:** Rate of out-of-order or duplicate timestamps detected. High rates indicate clock synchronization issues.
- **Unit Distribution Ratio:** Proportion of raw records in Watts vs. Kilowatts. Sudden changes indicate firmware update rollouts or configuration issues.

6.3 Monitoring & Alerting Strategy

Observability Framework

The pipeline implements multi-layered observability to ensure data reliability and operational visibility. Real-time metrics are emitted from each processing stage and aggregated into a centralized monitoring dashboard accessible to data engineering and operations teams.

Alert Tiers: Critical alerts (pipeline failure, DLQ overflow, completeness < 90%) trigger immediate notifications with 15-minute SLA. Warning alerts (latency degradation, retry rate increase) generate notifications for next-business-day review. Informational events (successful batch completions, meter replacements) are logged for audit trails.

Automated Responses: When retry queue depth exceeds 1000 records, auto-scaling provisions additional processing capacity. If DLQ accumulates 500+ records within 1 hour, automated diagnostics run to identify common error patterns and generate troubleshooting reports. Sustained high interpolation rates (> 15% for 3+ hours) trigger investigation of network infrastructure issues.

Data Quality Dashboards: Operations team has access to real-time visualizations showing: hourly ingestion volume trends, validation pass/fail ratios by error type, regional data completeness heatmaps, household-level anomaly flags requiring review, and end-to-end processing latency distributions. Historical trend analysis enables proactive capacity planning and identifies seasonal patterns in data quality issues.

Document Information

GreenStream Energy Smart Meter ETL Architecture
Data Science Design Document

Author: Fares Mamdouh (ID: 412200276)

Version: 1.0 — **Date:** December 2025