CloudWatch Logs Transformation and Querying Workflow Report

# Objective

The goal of this task was to build an automated workflow to extract, clean, and transform raw CloudWatch logs into a structured format that enables easy querying and analysis in Amazon Athena.

This involved:

1. Using AWS Glue (PySpark) for ETL (Extract, Transform, Load).
2. Storing cleaned logs in S3.
3. Querying structured logs with Amazon Athena.

# Raw Input

The raw dataset was a CSV file exported from CloudWatch Logs into a S3 Bucket in this format:

timestamp,message

1757511342804,"INFO 2025-09-10T13:35:42,804 10153 org.apache.spark.metrics.source.StageSkewness [Thread-10] 29 [Observability] Skewness metric using Skewness Factor = 5"

The challenge with this raw data:

* The message column contained multiple attributes mixed together (timestamp, log level, class name, thread, etc.).
* The structure was not suitable for querying (e.g., filtering by ERROR, aggregating by date).

# Feature Engineering & Transformations

To make the logs query able, we applied **regex-based parsing** inside an AWS Glue PySpark job.

## Steps Taken

1. **Regex Extraction**  
   Extracted key components from the message field:
   * **timestamp\_raw** → extracted event timestamp string.
   * **log\_level** → captured INFO, ERROR, DEBUG, etc.
   * **thread** → extracted thread name (e.g., Thread-10).
   * **class** → Java/Python class that logged the event.
   * **line** → line number in source file.
   * **message\_clean** → actual log message text.
2. **Type Conversion**
   * Converted timestamp\_raw → **timestamp** column (timestamp type).
   * Extracted **date** (date type) from timestamp for partitioning and easy filtering.
3. **Filtering Noise**
   * Removed rows where message\_clean was empty or null (avoided blank entries).

# Final Schema

After transformations, the structured dataset contained the following columns:

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| **timestamp** | timestamp | Event timestamp with full precision. |
| **log\_level** | string | Log level (INFO, ERROR, DEBUG, WARN, TRACE). |
| **thread** | string | Thread name that produced the log. |
| **class** | string | Class/package name of the log source. |
| **line** | int | Line number in the source file. |
| **message\_clean** | string | Cleaned human-readable log message. |
| **date** | date | Event date (extracted from timestamp). |

# Workflow Overview

## Extract

* + Source: CloudWatch logs exported to S3 (log-events-viewer-result.csv).
  + Format: CSV.

## Transform (Glue Job / PySpark)

* + Parsed message using regex.
  + Added structured fields (log\_level, class, etc.).
  + Converted timestamps into proper types.
  + Removed empty rows.

## Load

* + Saved cleaned data as CSV in S3 under s3://cloudwatch-cleaned-logs-aip-71/cleaned-logs/.
  + Ensured single file output (part-0000\*.csv).

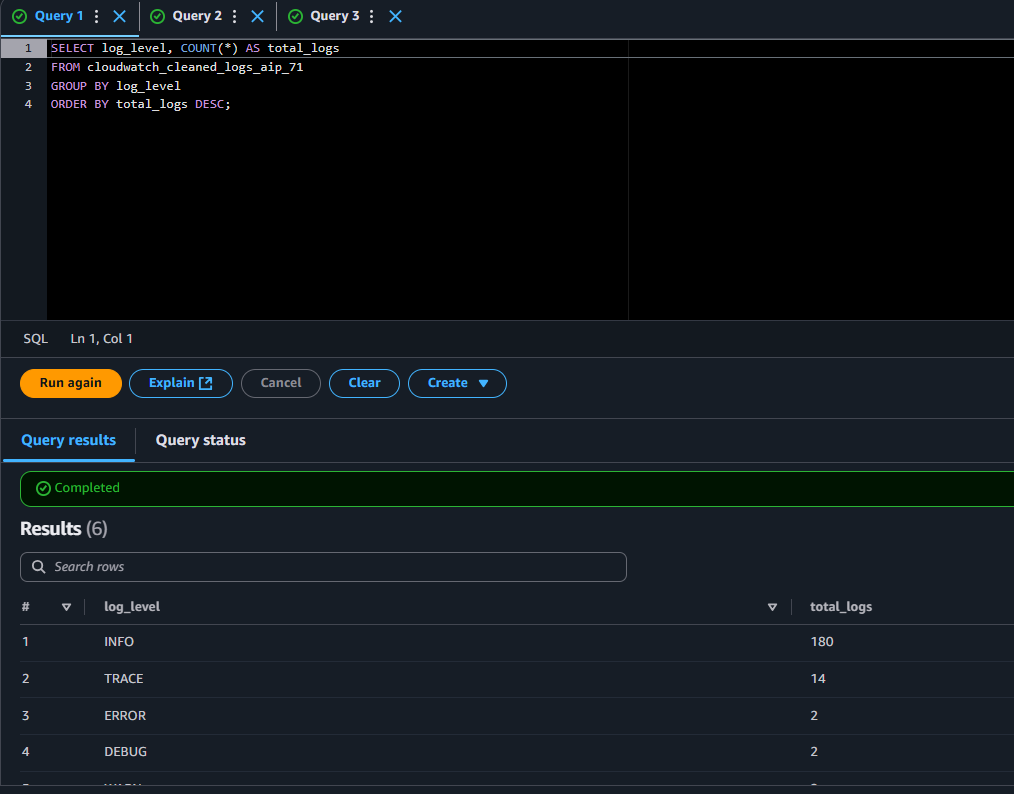
## Query (Athena)

* + Created Athena external table over the cleaned logs.
  + Queried logs by date, log\_level, and keywords.

## Why These Features Were Created

* **timestamp** → Enables time-series queries (e.g., logs per hour/day).
* **date** → Optimized partitioning in Athena, allows fast date filtering.
* **log\_level** → Separates ERROR/INFO logs for reliability monitoring.
* **thread** → Useful for debugging concurrency or thread-specific issues.
* **class** → Helps trace which component of the system generated logs.
* **line** → Assists in pinpointing exact code lines for debugging.
* **message\_clean** → Human-readable message for deeper insights and keyword search.

## Example Athena Queries



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# Conclusion

By engineering new features from the unstructured message column, we converted raw CloudWatch logs into a structured dataset suitable for analytical querying.  
This workflow:

* Improves log observability.
* Enables faster debugging and monitoring via Athena.
* Provides flexibility for future extensions (e.g., partitioning by log\_level or date for performance).