**Data Lake Construction and Querying with PySpark**

Table of Contents

1 INTRODUCTION ………………………………………………………………………………………….. 3

1.1 Purpose of Document ………………………………………………………………………. 3

1.2 Project Overview ………………………………………………………………………………. 3

1.2.1 Objective ……………………………………………………………………………………. 3

2 Project Components …………………………………………………………………………………… 3

3 Technology Stack ……………………………………………………………………………………….. 4

4 GitHub Repository Overview ………………………………………………………………………. 4

4.1 Repository URL …………………………………………………………………………………... 5

5 Architectural Diagram …………………………………………………………………………….….. 6

6 AWS Services ……………………………………………………………………………………………… 6

6.1 Apache Airflow ………………………………………………………………………………. 6

* 1. AWS Lambda …………………………………………………………………………………. 7
  2. S3 Bucket ………………………………………………………………………………………. 8

6.3.1 S3 Folder Structure …………………………………………………………….…. 8

* 1. AWS Glue ………………………………………………………………………………………. 9
  2. AWS Glue Crawler ……………………………………………………………………….. 10
  3. CloudWatch Log ………………………………………………………………………….. 11

6.6.1 CloudWatch Log Link ………………………………………………………..…. 11

* 1. SNS Notification ………………………………………………………………………….. 12
  2. Amazon Athena …………………………………………………………………………… 13
  3. IAM ……………………………………………………………………………………………... 14

7 AWS Cost Metrics ……………………………………………………………………………………….14

8 Cost Optimization Tips ………………………………………………………………………………. 15

9 Conclusion …………………………………………………………………………………………………. 15

**1 Introductions**

**1.1 Purpose of Document:**

The purpose of this document is to provide an in-depth overview of the architecture and components involved in constructing and querying a data lake using AWS services such as Amazon S3, AWS Glue, Lambda, and Athena. It describes the workflow and interactions between these services, showcasing how they collaboratively process, transform, store, and analyze data efficiently within a PySpark-powered data lake pipeline

**1.2 Project Overview:**

In this project, a Data Lake will be designed and built to store raw, structured, and unstructured data while enabling querying capabilities using PySpark. The Data Lake will support scalable storage and efficient querying of large datasets, providing a foundation for future analysis and reporting.

* + 1. **Objectives**

To design and build a scalable Data Lake capable of storing and querying raw, structured, and unstructured data efficiently.

**2 Project Components:**

* **Data Ingestion:** Collect and store data from various sources (e.g., CSV, JSON, logs) in its raw format.
* **Data Processing:** Use PySpark to process and analyze the ingested data.
* **Data Storage:** Save data in a distributed file system such as HDFS, Amazon S3, or Delta Lake.
* **Querying and Reporting:** Enable querying capabilities using Spark SQL and tools like Amazon Athena for analytical insights.

**3 Technology Stack:**

* **Processing:** PySpark (v3.3) – Handles data processing and transformations within the pipeline.
* **Storage:** Amazon S3 – Provides scalable and durable storage for raw, processed, and structured data.
* **Querying:** Spark SQL – Facilitates querying and analysis of data stored in the Data Lake.
* **Orchestration:** Amazon Web Services (AWS) – Integrates and manages workflows using services like AWS Glue and Apache Airflow.
* **Output View:** Athena Table – Enables SQL-based querying and visualization of processed data.

**4 GitHub Repository Overview**

The repository contains all the essential code and configurations for the project "Data Lake Construction and Querying with PySpark." It is organized to ensure seamless execution and ease of use, including the following components:

* **Config Files:**  
  Setup and environment configurations essential for executing the data pipeline. These files include parameters like file types, table names, and queries.

**Example Configuration**:

{

"file\_type": "parquet",

"table\_name": "data\_lake\_query",

"ps\_query": "SELECT \* FROM data\_lake\_query WHERE `Date of birth` BETWEEN '2000-01-01' AND '2024-12-31'"

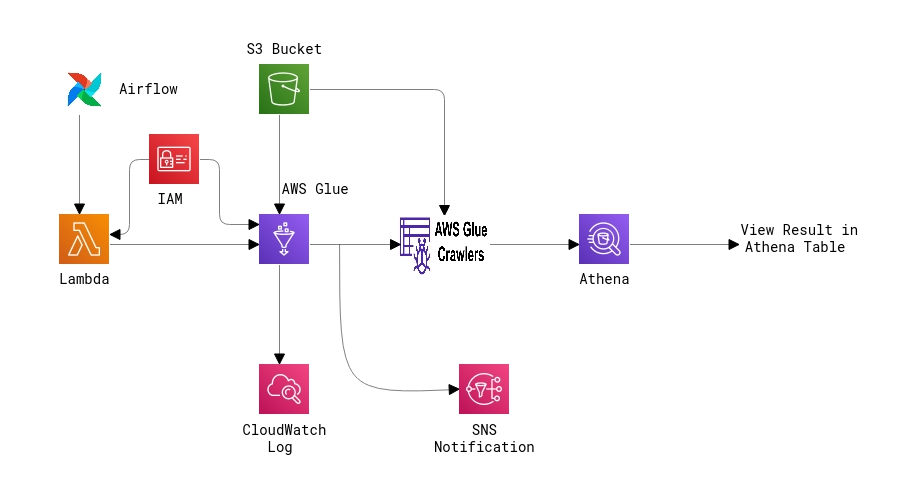
}

* + **File Type**: Defines the data format (e.g., parquet or csv).
  + **Table Name**: Specifies the table for querying (e.g., data\_lake\_query).
  + **Query**: Contains the SQL logic for filtering and selecting data.
* **Airflow DAGs**:  
  Code for orchestrating and scheduling workflows using Apache Airflow. The DAGs define the sequence of tasks for data ingestion, processing, and querying, ensuring smooth execution of the pipeline.
* **Lambda Functions**:  
  Custom serverless functions designed for event-driven processing. These functions are used for tasks such as triggering workflows, processing events, or sending notifications during the data pipeline execution.
* **Glue Scripts**:  
  ETL scripts for transforming, processing, and preparing data for analysis. These scripts enable efficient data transformation and querying in Amazon Athena.

All components are stored in the **master branch**, ensuring easy access and robust version control. This organization helps streamline the project workflow, supporting both development and deployment activities.

**4.1 Repository URL:** <https://github.com/rdinesh808/Data_Lake_Construction_and_Querying_With_PySpark>

1. **Architectural Diagram:**

****

1. **AWS Services:**

AWS (Amazon Web Services) is a comprehensive and widely adopted cloud platform that offers a collection of services to help businesses build, manage, and scale applications without the need for on-premises infrastructure. These services are hosted in Amazon's global network of data centers, providing scalability, reliability, and security.

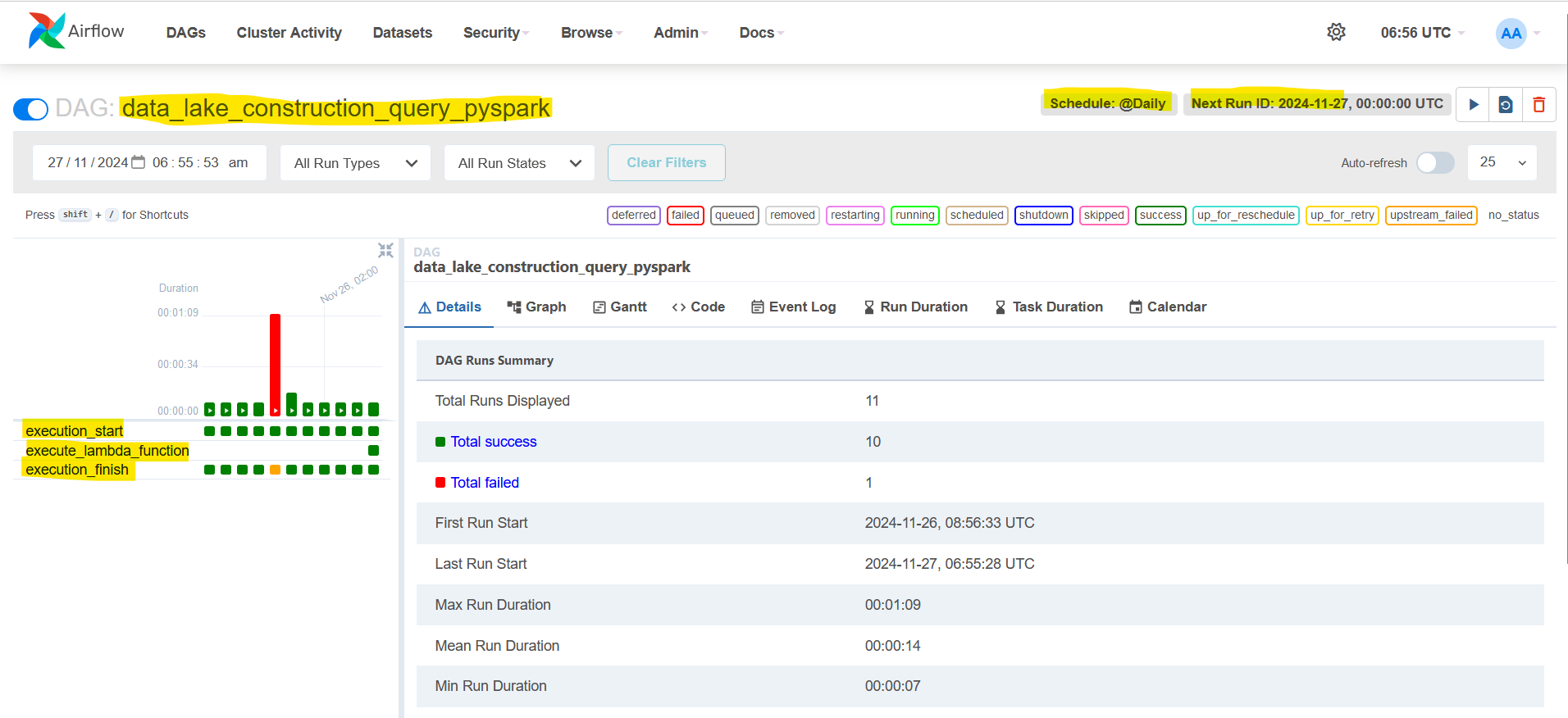
**6.1 Apache Airflow**

Apache Airflow is an open-source workflow orchestration platform used to create, schedule, and monitor complex workflows programmatically. It is widely utilized for automating and managing ETL (Extract, Transform, Load) processes, data pipelines, and other task-based workflows.

In this project, Airflow is used to orchestrate and automate the steps involved in constructing and querying the data lake. The workflows are defined as Directed Acyclic Graphs (DAGs), which outline the sequence of tasks and their dependencies.

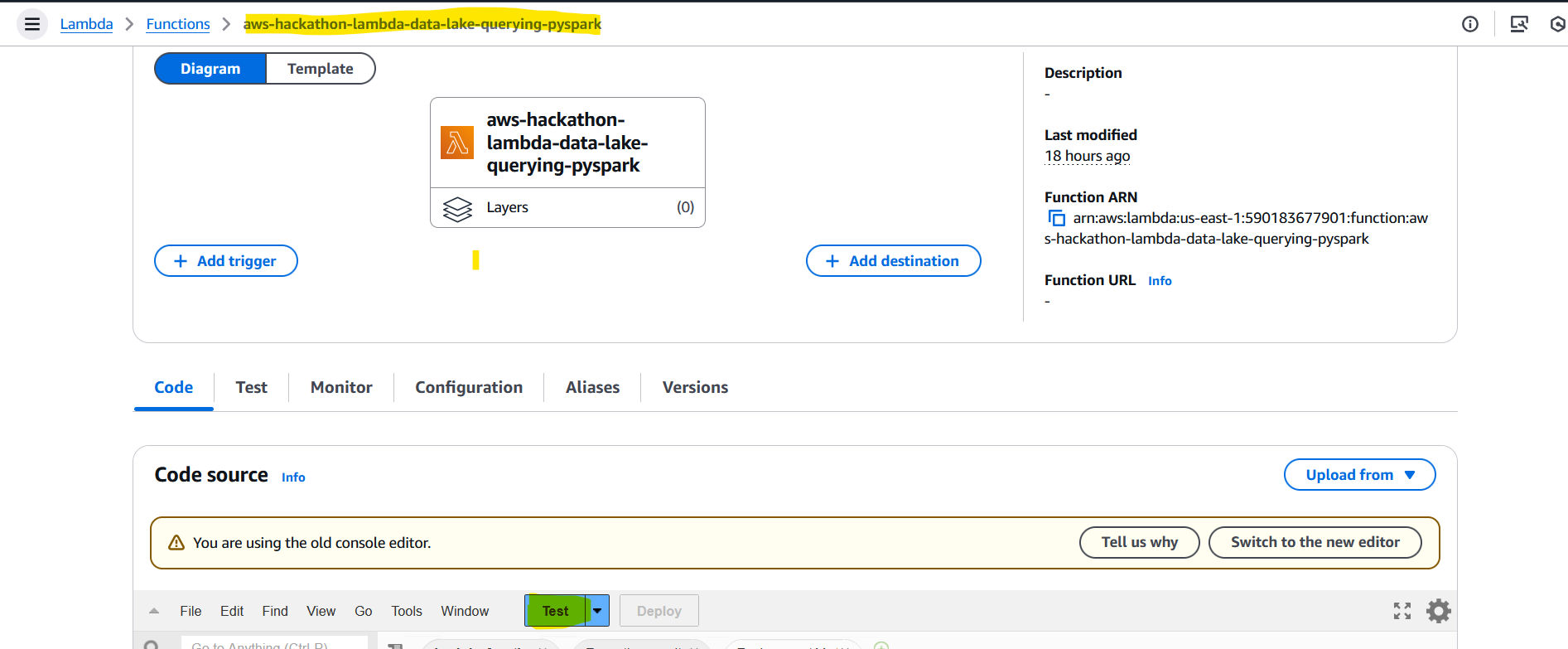
* **DAG Name**: data\_lake\_construction\_query\_pyspark
* **Scheduled Time**: Scheduled to run daily at 12:00 AM (midnight).

Airflow ensures reliable and repeatable execution of workflows, with features like task retries, logging, and monitoring via the Airflow web interface.

****

**6.2 AWS Lambda**

AWS Lambda is a serverless compute service used to run code without provisioning or managing servers. In this project, the Lambda function **aws-hackathon-lambda-data-lake-querying-pyspark** is utilized to trigger specific tasks in the data lake pipeline, such as invoking workflows, event-driven processing, or sending notifications.

****

**6.3 Amazon S3 Bucket**

Amazon S3 (Simple Storage Service) is used for storing raw, processed, and structured data within the data lake. It provides scalable, durable, and low-cost storage, serving as the central repository for all project data. In this case, the **aws-hackathon-s3-data-lake-querying-pyspark** bucket stores the data ingested from various sources and serves as the foundation for further analysis and querying in the data lake pipeline.

**6.3.1 S3 Folder Structure:**

| - config

| - data\_lake\_config.json

| - input\_data  
 | - csv

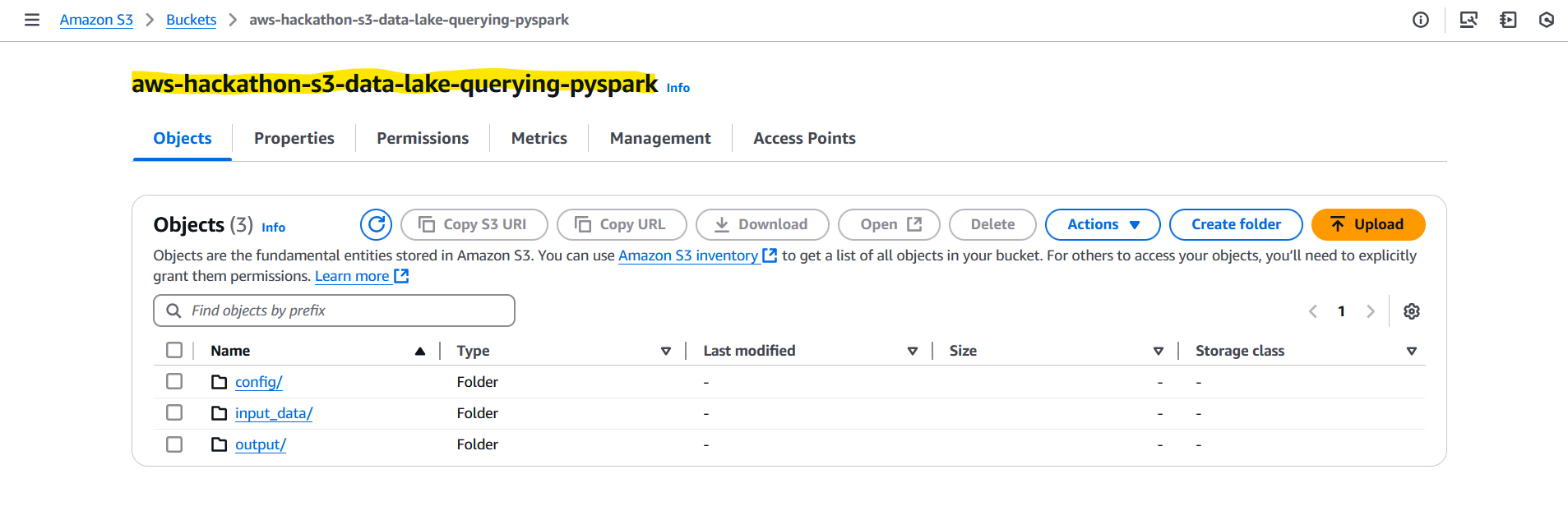
| - people.csv

| - parquet

| - people.parquet

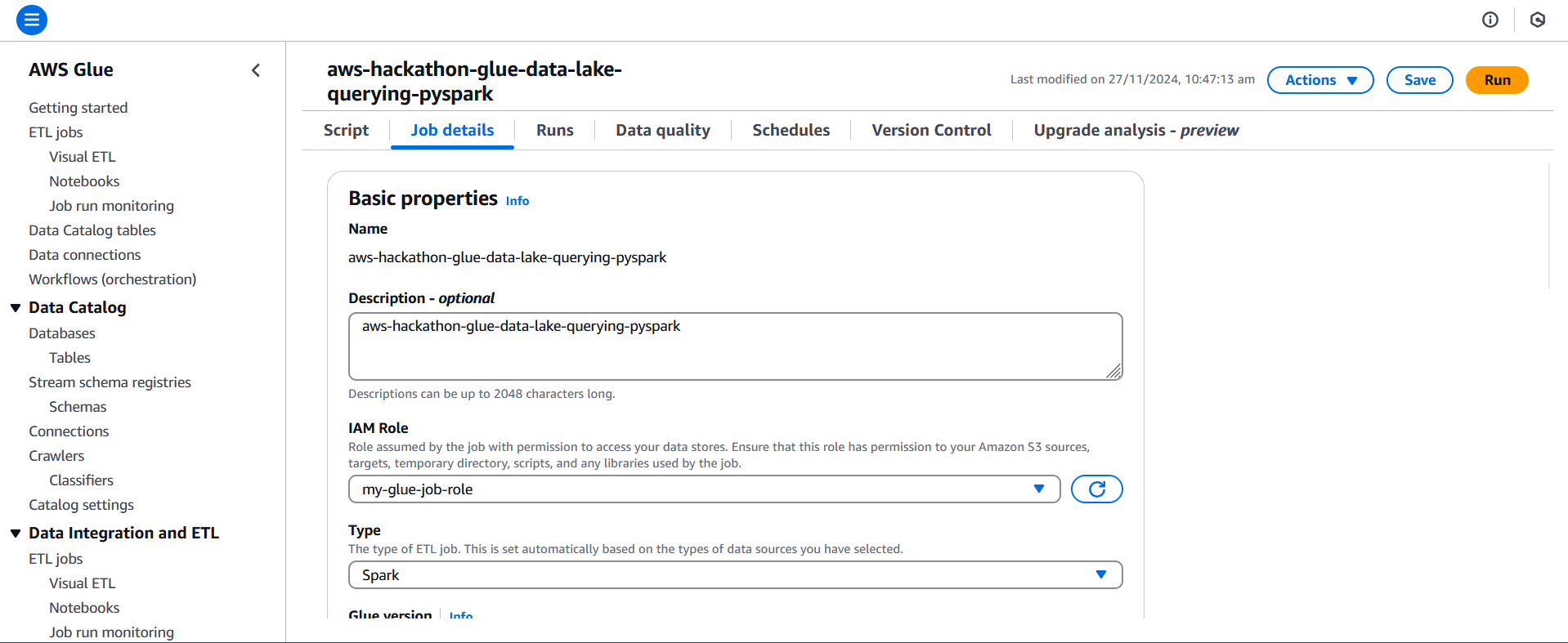
| - output

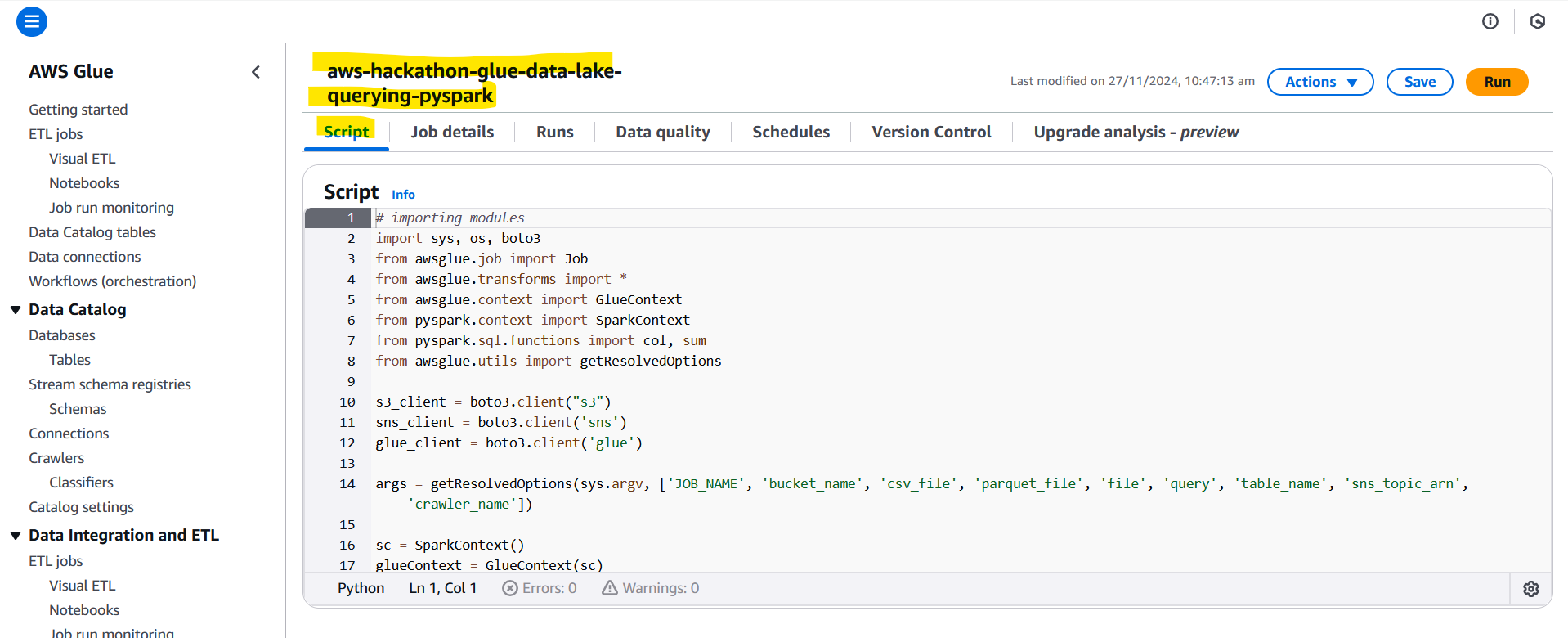
| - The output file store in based on run time



**6.4 AWS Glue**

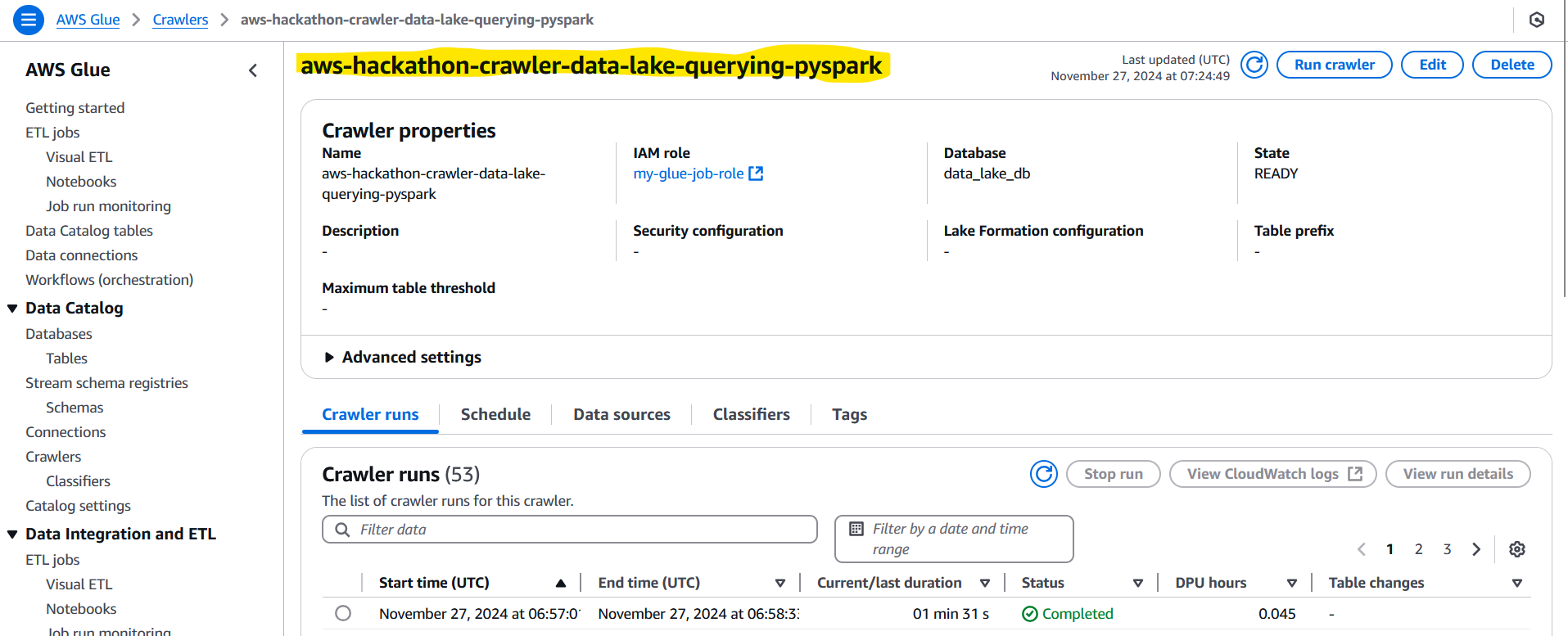
AWS Glue is a fully managed ETL (Extract, Transform, Load) service that automates the process of data discovery, transformation, and loading into data lakes or warehouses. In this project, **aws-hackathon-glue-data-lake-querying-pyspark** is the Glue job that processes data, performs necessary transformations, and prepares it for querying in Amazon Athena. Glue simplifies the management of ETL workflows, enabling faster data processing and more efficient querying of large datasets.





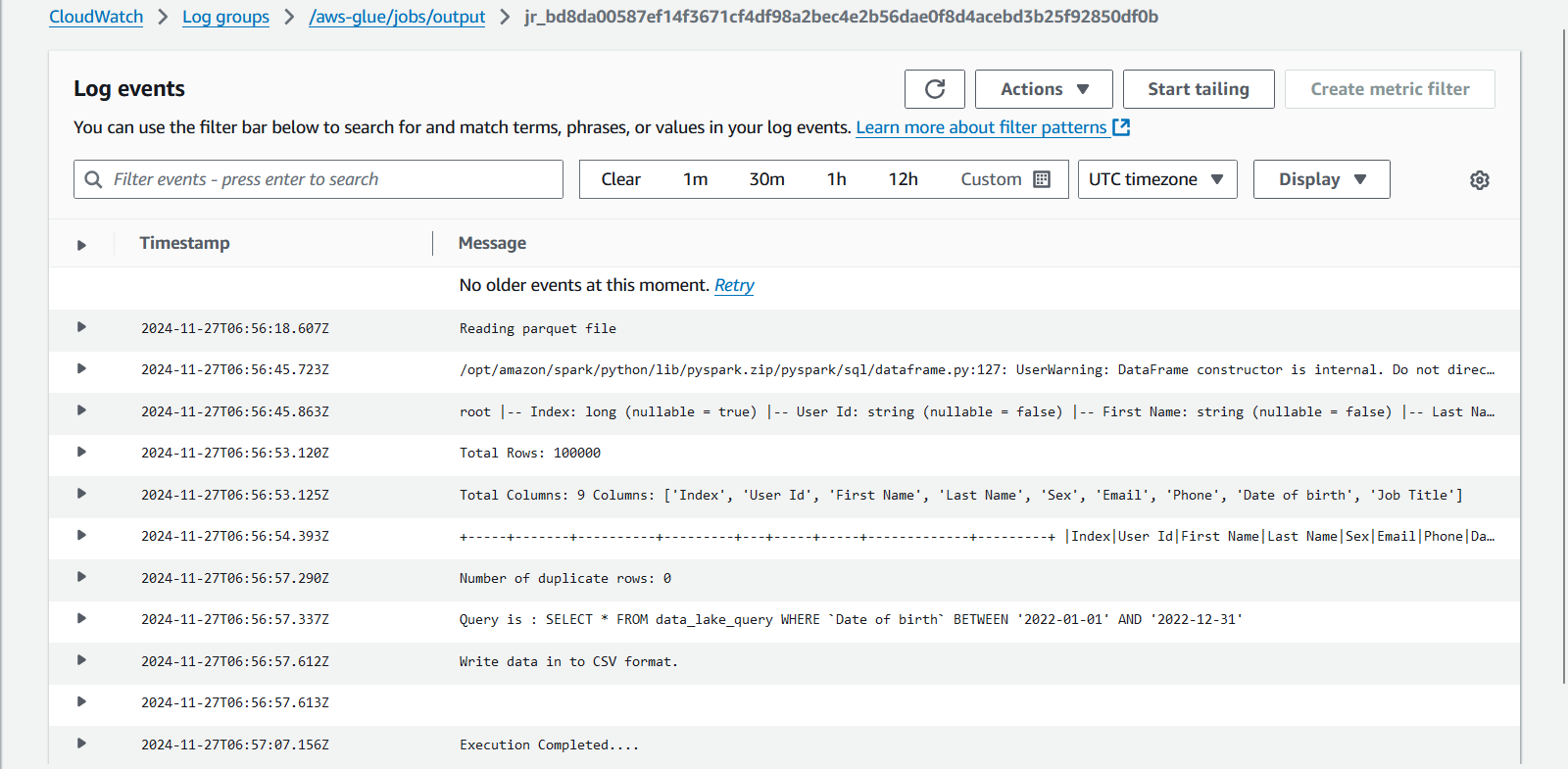
**6.5 AWS Glue Crawler**

AWS Glue Crawlers automatically discover and catalog metadata from data stored in Amazon S3, allowing you to efficiently query and analyse the data with tools like Amazon Athena. In this project, the **aws-hackathon-crawler-data-lake-querying-pyspark** Glue Crawler is used to scan the data stored in Amazon S3, infer the schema, and populate the AWS Glue Data Catalog. This makes the data ready for querying in Athena by creating tables and defining their structure.



**6.6 CloudWatch Log**

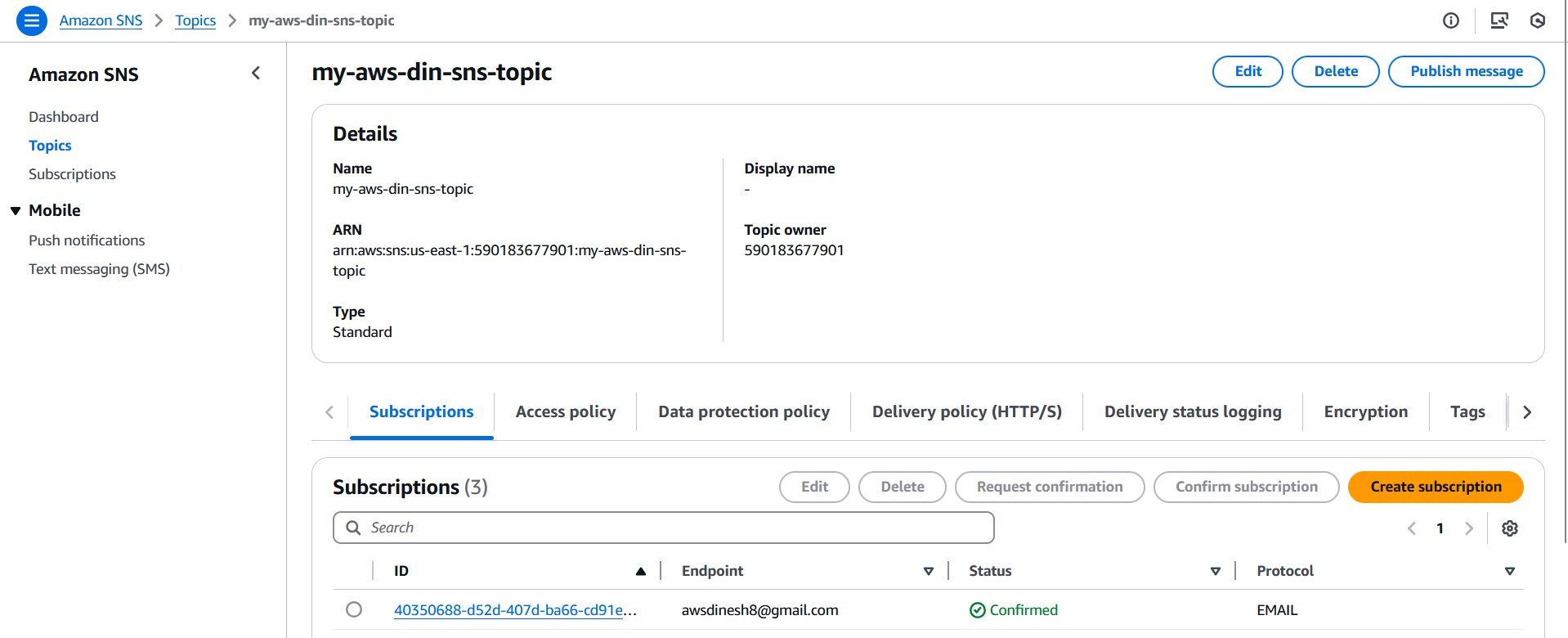
Amazon CloudWatch Logs is a monitoring and logging service that enables you to collect, store, and access logs from various AWS services, including AWS Glue, Lambda, and more. In this project, **CloudWatch Logs** is used to capture and store logs generated by the services within the data lake pipeline (e.g., AWS Glue jobs, Lambda functions). This helps in debugging, monitoring performance, and maintaining operational health by tracking the execution details and errors across the entire workflow.

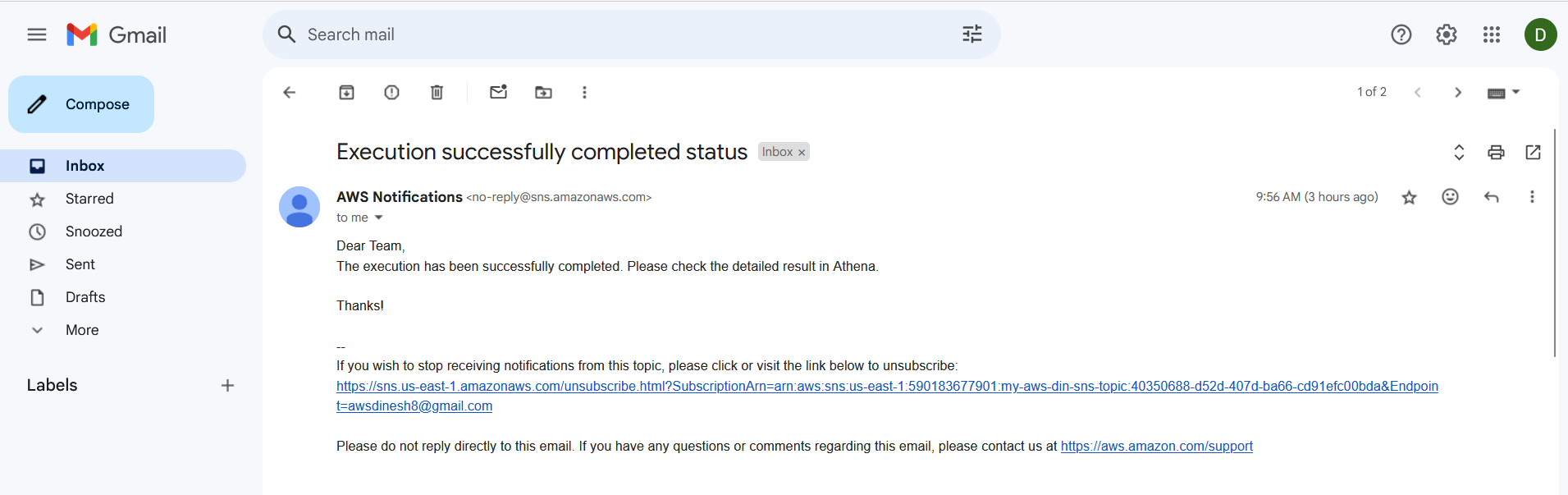


CloudWatch Log Link: [https://us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#logsV2:log-groups/log-group/$252Faws-glue$252Fjobs$252Foutput/log-events/jr\_bd8da00587ef14f3671cf4df98a2bec4e2b56dae0f8d4acebd3b25f92850df0b](https://us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1%23logsV2:log-groups/log-group/$252Faws-glue$252Fjobs$252Foutput/log-events/jr_bd8da00587ef14f3671cf4df98a2bec4e2b56dae0f8d4acebd3b25f92850df0b)

**6.7 SNS Notification**

Amazon Simple Notification Service (SNS) is a fully managed messaging service used to send notifications from applications or AWS services. In this project, **SNS Notification** is utilized to send alerts for important events such as job completions, failures, or any other significant state changes in the data lake pipeline. SNS ensures timely and automated communication by sending notifications via email, SMS, or other protocols.



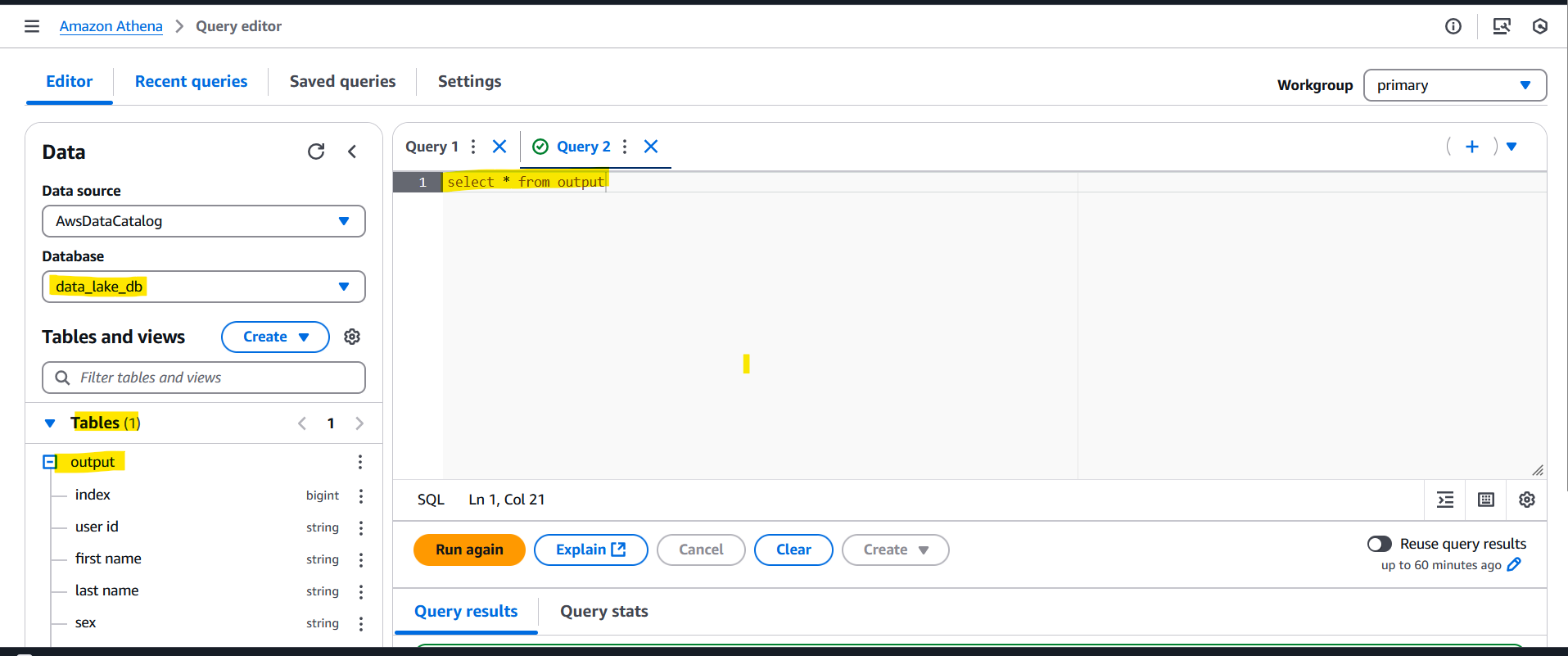


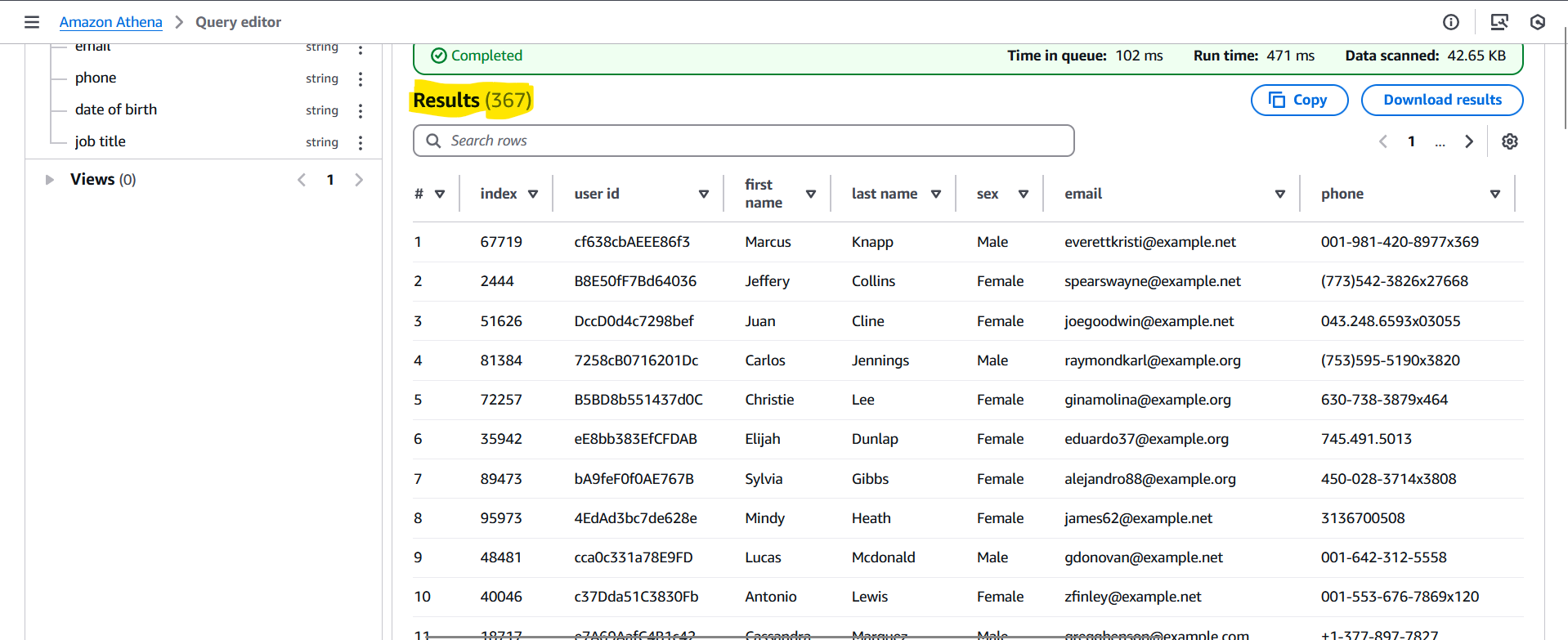


email notification document.

**6.8 Amazon Athena**

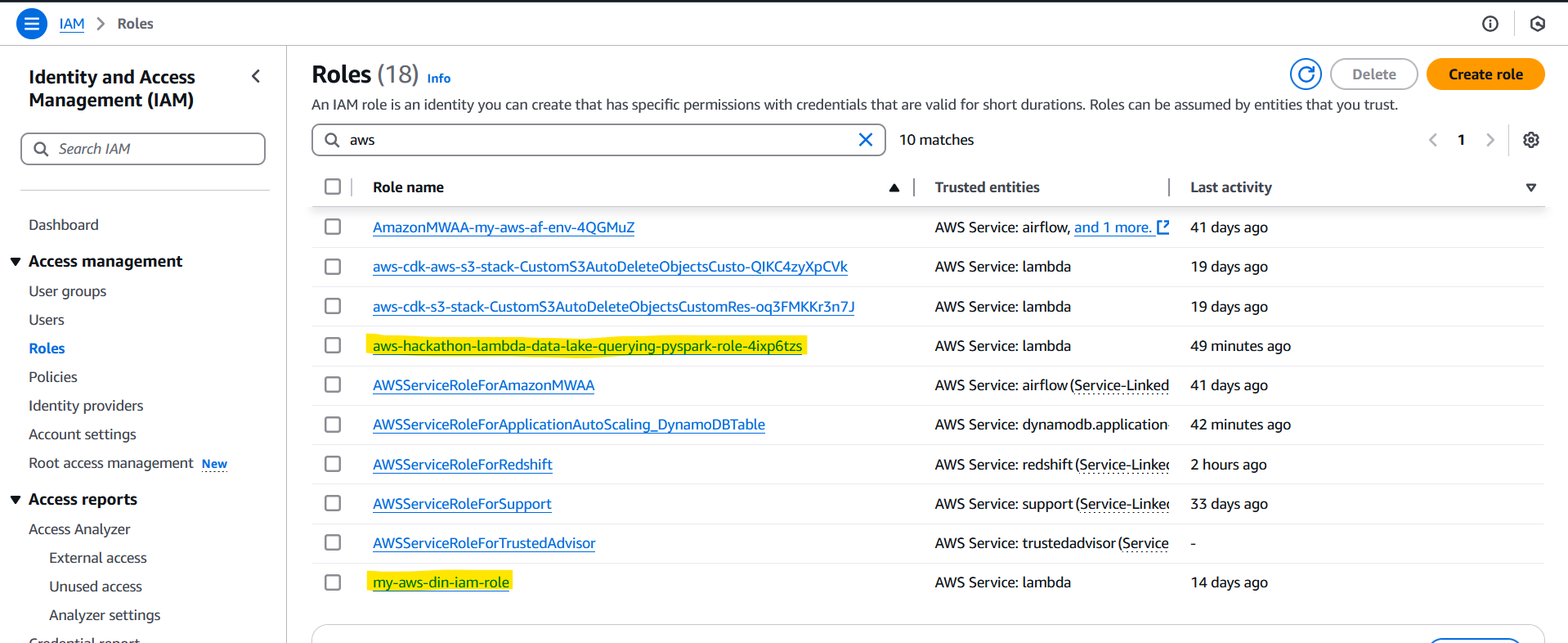
Amazon Athena is a serverless interactive query service that allows you to analyse data directly in Amazon S3 using standard SQL. In this project, Amazon Athena is used to query the processed data stored in the data lake (Amazon S3) and retrieve insights using SQL queries. With its pay-per-query pricing model, Athena enables cost-effective and scalable querying of large datasets without the need to manage infrastructure.





**6.9 IAM (Identity and Access Management)**

AWS Identity and Access Management (IAM) allows you to securely control access to AWS services and resources. In this project, IAM is used to define and manage the roles and permissions that allow AWS services (such as Lambda, Glue, S3, Athena, etc.) to interact with each other. IAM ensures that each service has the required permissions to access and modify data securely, while also enforcing security policies to prevent unauthorized access.



**7 AWS Cost Metrics**

The estimated monthly cost for AWS services used in the data lake architecture is as follows:

| Service | Estimated Monthly Cost |
| --- | --- |
| AWS Glue | $5.00 |
| Glue Crawlers | $1.00 |
| Amazon Athena | $0.50 |
| AWS Lambda | $0.90 |
| Amazon S3 | $5.00 |
| Amazon CloudWatch | $2.00 |
| Amazon SNS | $0.10 |
| Total Estimated Monthly Cost | ~$14.50 |

**8 Cost Optimization Tips**

To reduce AWS costs, consider the following tips for each service:

* **AWS Glue:** Reduce unnecessary transformations and use partitioned datasets to save on processing costs.
* **Glue Crawlers:** Limit crawler runs and scan only relevant directories to reduce costs.
* **Amazon Athena:** Query only partitioned data and optimize queries to minimize data scanned.
* **AWS Lambda:** Consolidate functions, set appropriate memory limits, and manage execution time to reduce costs.
* **Amazon S3:** Use appropriate storage classes (e.g., Glacier for archived data), implement lifecycle policies, and clean up unused files.
* **Amazon CloudWatch:** Set retention policies for logs and use custom metrics sparingly.
* **Amazon SNS:** Limit SNS usage to essential notifications and batch publish messages when possible.

**9 Conclusion:**

This project showcases the effective use of AWS services to design and implement a scalable, efficient data lake for processing and querying large datasets. By leveraging tools like Amazon S3, AWS Glue, Lambda, and Athena, the pipeline automates data ingestion, transformation, and analysis, enabling cost-effective and seamless workflows. This architecture provides a robust foundation for real-time analytics and future enhancements, such as integrating AI/ML capabilities for predictive insights and advanced reporting.