



University of New Haven

TAGLIATELA COLLEGE OF ENGINEERING

Electrical & Computer Engineering and Computer Science

# Electrical & Computer Engineering & Computer Science (ECECS)

## TECHNICAL REPORT



**SEMESTER 2**

# CONTENT

1

|                               |    |
|-------------------------------|----|
| Executive Summary             | 2  |
| Team Members                  | 3  |
| Title & Highlights of Project | 4  |
| Submitted on                  | 4  |
| Abstract                      | 5  |
| Methodology                   | 7  |
| Results                       | 9  |
| Discussion                    | 17 |
| Conclusion                    | 17 |
| Contributions/References      | 17 |

## Executive Summary

This project presents the development of a fully automated, real-time hospital patient analytics platform using Amazon Web Services (AWS). The primary objective is to simulate hospital patient events, process them efficiently through cloud-native pipelines, and deliver actionable insights via dynamic dashboards. The system leverages key AWS services including Lambda, Kinesis Firehose, S3, Glue, Athena, EventBridge, and QuickSight to achieve seamless data ingestion, transformation, reporting, and visualization without manual intervention.

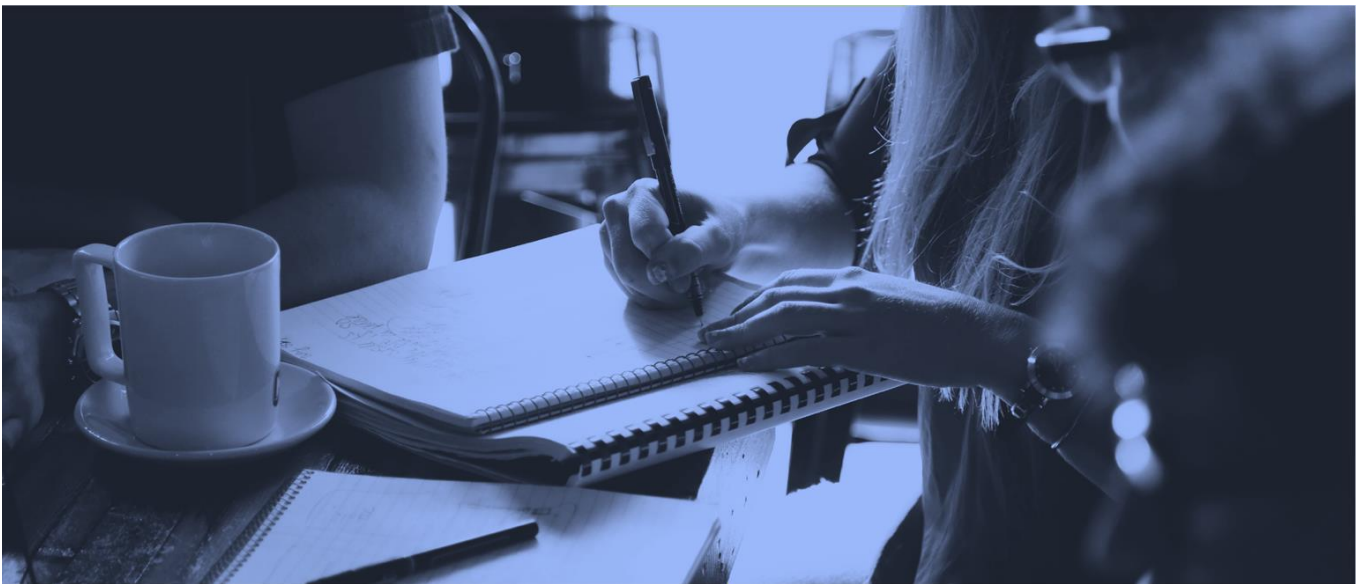
The data pipeline begins with a Lambda function that generates realistic hospital patient events such as admission details, vitals, diagnoses, and treatments. These events are streamed in near real-time to Amazon Kinesis Firehose, which buffers and stores them in an Amazon S3 data lake. AWS Glue crawlers automatically catalog the incoming data, making it queryable through Amazon Athena. A second Lambda function is scheduled via EventBridge to run daily Athena queries and generate multiple analytical reports — such as patient inflow trends, department load, financial payment breakdowns, emergency case handling, and resource utilization metrics. These reports are stored back in S3 for further analysis.

For visualization, Amazon QuickSight is employed to create interactive dashboards that provide a comprehensive view of hospital operations. The dashboards are divided into six major areas: Patient Inflow Overview, Hospital Load by Department, Financial Risk Monitoring, Doctor Critical Load and Emergency Outcomes, Resource Utilization, and Longest Stay Analysis. Each dashboard automatically pulls data from updated reports, offering hospital administrators real-time intelligence for operational decision-making.

By integrating real-time streaming, serverless processing, automated reporting, and business intelligence visualization, this project demonstrates a scalable, cost-effective, and highly efficient healthcare analytics system. It showcases how cloud technologies can revolutionize hospital data management by enabling timely insights, improving resource allocation, and supporting proactive patient care strategies.

Overall, this project successfully replicates an enterprise-grade healthcare analytics architecture on AWS Free Tier, showcasing practical skills in data engineering, cloud computing, and real-time business intelligence deployment.

## Team Members



Kishan Sai Saguturu

[ksagu1@unh.newhaven.edu](mailto:ksagu1@unh.newhaven.edu)

Sirisha Gajula

[sgaju8@unh.newhaven.edu](mailto:sgaju8@unh.newhaven.edu)

Vittu Ramadasu Darshan

[vdars1@unh.newhaven.edu](mailto:vdars1@unh.newhaven.edu)

# TITLE

## **AUTOMATED REAL TIME HOSPITAL PATIENT DATA ANALYTIC PLATFORM**



### **Highlights of Project:**

This project showcases the development of a fully automated, real-time hospital analytics platform using AWS services. Patient event data is simulated through Lambda, streamed using Kinesis Firehose, and stored in S3 for structured querying via Glue and Athena. A scheduled Lambda function generates daily analytical reports covering patient inflow, department load, financial trends, and emergency outcomes. These reports are visualized through interactive QuickSight dashboards, offering dynamic insights for hospital administrators. The system is fully automated through EventBridge scheduling and built entirely within AWS Free Tier limits, demonstrating efficient, scalable, and cost-effective healthcare data management.

**Submitted on: 27<sup>th</sup> April, 2025**

## Abstract

The ability to analyze hospital data in real-time has become critical for improving operational efficiency, patient care, and resource management. This project focuses on building an automated, scalable hospital data analytics platform using Amazon Web Services (AWS) within the Free Tier framework. Simulated patient events, such as admissions, vitals, treatments, and outcomes, are generated through AWS Lambda and streamed to Amazon S3 using Kinesis Firehose. AWS Glue automatically catalogs the incoming data, making it available for SQL-based querying through Amazon Athena.

A second Lambda function processes these records daily, executing a series of predefined queries to generate detailed analytical reports. These reports provide insights into key operational metrics including patient inflow trends, department workloads, emergency outcomes, financial payment patterns, and bed utilization rates. The generated reports are stored in S3 and visualized through Amazon QuickSight dashboards, offering real-time, dynamic monitoring capabilities for hospital administrators.

Automation is orchestrated through Amazon EventBridge, allowing the entire system to run without manual intervention. The dashboards are updated daily, ensuring decision-makers have access to the latest information regarding patient activity and hospital resource usage. This project demonstrates the effectiveness of combining serverless computing, real-time data streaming, automated ETL, and business intelligence tools to build a robust, low-cost healthcare analytics solution. It highlights how cloud technologies can transform traditional hospital data systems by enabling more informed, data-driven decision-making.

## Introductory Section

In modern healthcare, data-driven decision-making is essential for optimizing patient care, managing hospital resources, and improving operational efficiency. Hospitals generate vast amounts of data daily, ranging from patient admissions and clinical records to financial transactions and resource allocations. However, without a real-time, automated system to process and analyze this data, valuable insights often remain untapped. Traditional data management approaches are slow, labor-intensive, and unable to meet the dynamic demands of hospital operations.

This project addresses these challenges by developing a real-time hospital data analytics platform using Amazon Web Services (AWS). By integrating serverless computing, real-time data streaming, automated ETL, and business intelligence visualization, the system delivers actionable insights with minimal manual intervention. Simulated patient data is generated and streamed into a cloud data lake, processed automatically, and visualized through interactive dashboards. The solution demonstrates how cloud technologies can enable hospitals to monitor patient inflow, department workloads, financial trends, emergency handling efficiency, and resource utilization on a daily basis.

The platform is designed to be scalable, cost-effective, and easily maintainable, making it suitable for real-world healthcare environments. By leveraging AWS services within the Free Tier, this project proves that even resource-constrained systems can achieve enterprise-grade data analytics capabilities, paving the way for smarter, more responsive hospital management.



## Methodology

### 1. Data Simulation & Streaming

The project begins with the simulation of realistic hospital patient events using an AWS Lambda function. This function randomly generates data fields such as patient names, insurance status, vital signs, reasons for admission, symptoms, treatments administered, and discharge notes. To maintain logical consistency, rules were implemented—for example, patients without insurance were restricted to cash or card payments, and days in hospital were correlated with criticality and diagnosis severity. Once generated, the event data is sent to Amazon Kinesis Firehose, a real-time data streaming service. Firehose acts as a buffer, batching incoming records and automatically delivering them to an Amazon S3 bucket, ensuring durable and cost-efficient storage of raw patient events.

### 2. Data Cataloging and Querying

To transform raw streamed data into queryable structured datasets, AWS Glue Crawlers were configured. The Glue Crawler automatically scans the new S3 data every 5 minutes, detects schema changes, and updates the AWS Glue Data Catalog. This process ensures that newly arriving patient data is consistently available for analysis through Amazon Athena. Athena allows serverless querying of data using standard SQL. No infrastructure provisioning is required, and queries can directly run on data stored in S3, making it an efficient and cost-effective solution for extracting insights from the hospital event data.

### 3. Automated Report Generation

To create detailed daily analytical reports, a second AWS Lambda function was deployed. This Lambda, triggered daily through Amazon EventBridge, runs a predefined set of SQL queries in Athena. Each query is designed to extract specific insights, such as the top reasons for patient visits, department load, financial payment breakdowns, emergency handling outcomes, bed type usage, and longest stay patients. The results of these queries are outputted as CSV files and saved back into the designated S3 bucket under a separate "hospital-reports/" folder. A total of 17 distinct reports are generated, ensuring comprehensive operational coverage.



## 4. Dashboard Development and Visualization

Amazon QuickSight is used to create interactive, visually appealing dashboards. Each report generated by Lambda is imported as a separate dataset into QuickSight. Dashboards are organized into six major themes:

- Patient Inflow Overview
- Hospital Load by Department
- Financial Risk Monitoring
- Doctor Critical Load and Emergency Outcomes
- Resource Utilization
- Longest Stay Analysis

Each dashboard uses KPI widgets, pie charts, bar charts, and stacked bar charts to present insights in a clean, actionable format. Filters, sorting, conditional formatting, and customized color schemes are applied to enhance interpretability.

## 5. Automation and Scheduling

Automation across the pipeline is achieved through Amazon EventBridge scheduling. EventBridge triggers the report-generating Lambda every 24 hours, ensuring that the dashboards stay updated with the latest patient data. AWS Glue Crawlers are similarly scheduled to run every 5 minutes to update the Athena table schemas. Although QuickSight datasets were initially created manually using manifest files, future enhancements would involve connecting datasets directly to Athena queries or S3 live connections with scheduled SPICE refreshes, ensuring complete end-to-end real-time automation.

## 6. Project Testing and Validation

Each component of the system was individually tested during development. Lambda functions were tested using simulated events, Kinesis Firehose delivery streams were verified through

S3 object monitoring, and Glue Crawlers were validated to detect new schema changes properly. Athena queries were tested with sample data to ensure correctness, and QuickSight dashboards were updated to verify that visualizations matched the expected outputs. Stress testing was also performed by generating a large volume of patient records (~500 at a time) to ensure the scalability and reliability of Firehose, S3 storage, Glue Crawlers, and Athena queries under increased load.

## Implementation & Results Section

### 1. Setting Up Data Generation

- Developed an AWS Lambda function to simulate realistic patient event data, including patient vitals, diagnosis, treatments, and billing information.
- Ensured logical consistency across fields, such as linking critical conditions to longer hospital stays.

```

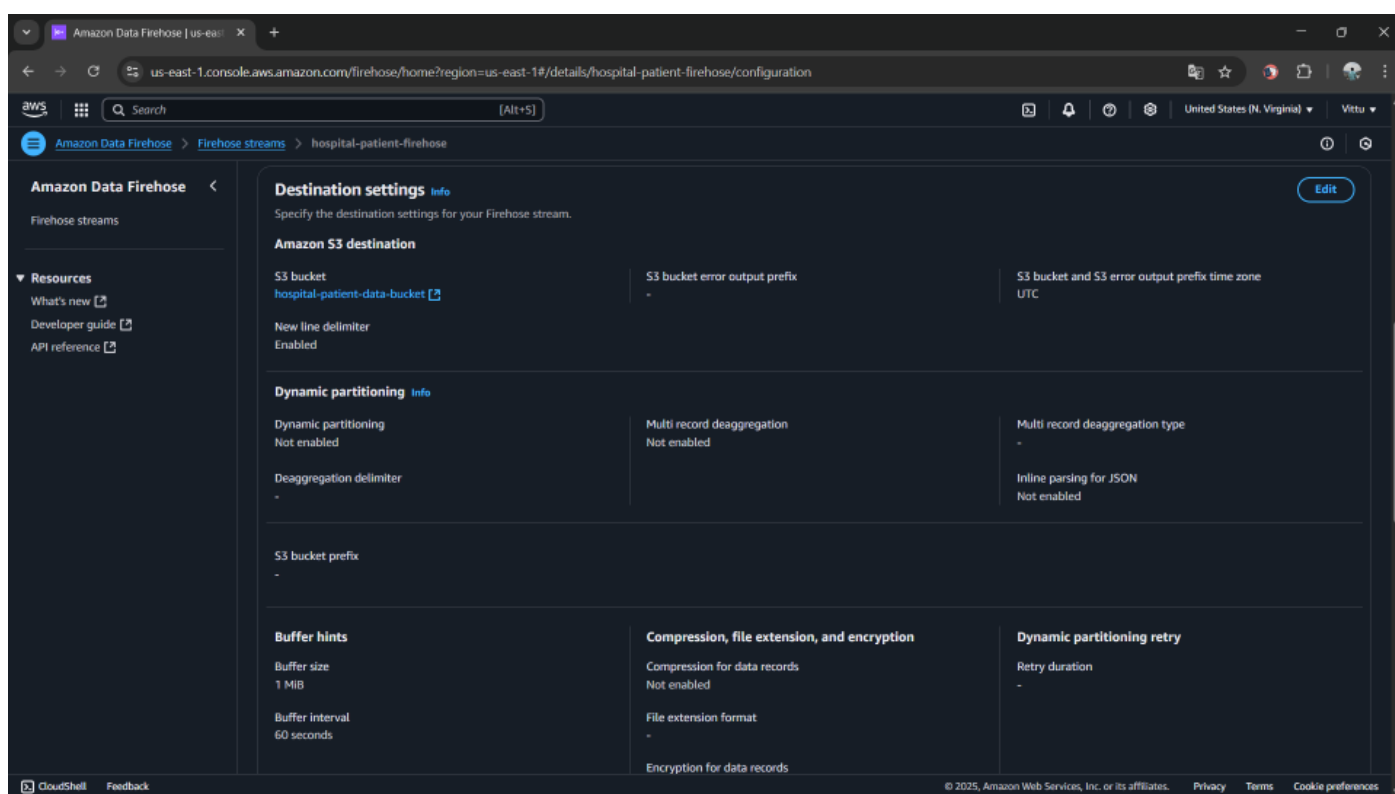
1 import json
2 import boto3
3 import random
4 import datetime
5
6 # Firehose Setup
7 FIREHOSE_NAME = 'hospital-patient-firehose'
8 firehose_client = boto3.client('firehose')
9
10 # Reference data
11 departments = ['ER', 'Radiology', 'Surgery', 'ICU', 'Recovery', 'Discharge']
12 genders = ['Male', 'Female']
13 first_names = ['John', 'Jane', 'Amit', 'Sara', 'Wei', 'Ali', 'Maria', 'Liam', 'Noah', 'Elena', 'Olivia', 'Aarav', 'Vivaan', 'Zara']
14 last_names = ['Smith', 'Johnson', 'Patel', 'Ali', 'Chen', 'Garcia', 'Khan', 'Lee', 'Kumar', 'Williams', 'Brown', 'Singh', 'Sharma', 'Thomas']
15 insurance_companies = ['Aetna', 'Cigna', 'Blue Cross', 'UnitedHealth', 'Humana', 'No Insurance']
16 hospital_branches = ['Downtown Hospital', 'Uptown Hospital', 'Eastside Clinic']
17 payment_methods = ['Insurance', 'Cash', 'Credit Card']
18
19 # Reason mappings with stay_range and discharge_notes
20 reason_mapping = {
21     "Chest Pain": {
22         "symptoms": ["Chest tightness", "Shortness of breath"],
23         "tests": ["ECG", "Blood Test", "Chest X-ray"],
24         "medications": ["Aspirin", "Nitroglycerin"],
25         "critical": True,
26         "stay_range": (5, 8),
27         "discharge_notes": ["Patient stabilized", "Prescribed heart medication", "Scheduled follow-up ECG"]
28     },
29     "High Fever": {
30         "symptoms": ["Chills", "Sweating", "Body pain"],
31         "tests": ["Blood Test", "COVID Test"],
32     }
33 }

```

### Python Code to Generate Data

## 2. Real-Time Data Streaming via Kinesis Firehose

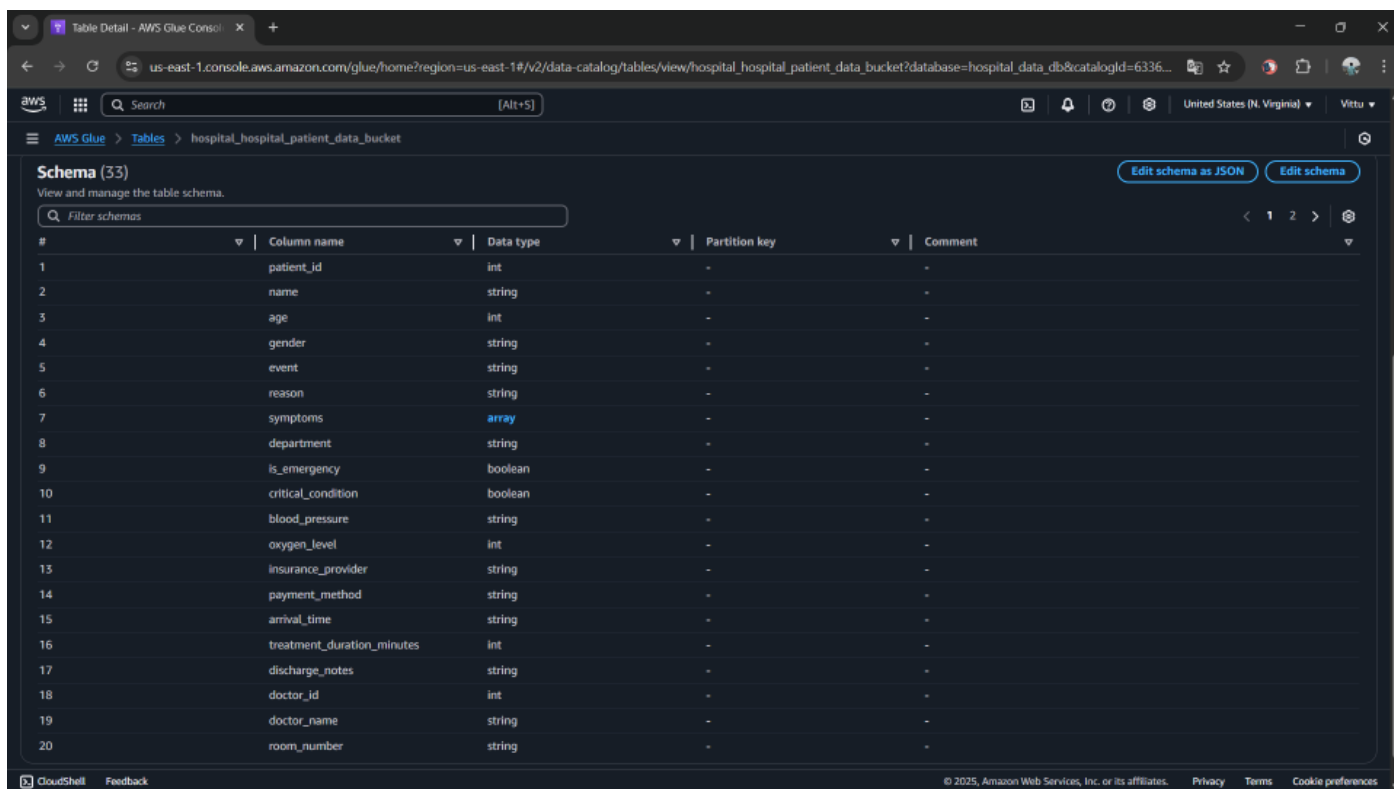
- Configured a Kinesis Firehose delivery stream to receive simulated patient events from Lambda and store them into an S3 bucket.
- Chose the Direct PUT method with default buffering settings to manage incoming data.



## Kinesis Firehose Configuration

### 3. Data Storage and Cataloging

- Created an Amazon S3 bucket for raw patient events and configured AWS Glue Crawler to scan the S3 bucket every 5 minutes.
- Successfully catalogued the incoming data for querying via Athena.



The screenshot shows the AWS Glue Console interface. The breadcrumb navigation indicates the path: **AWS Glue** > **Tables** > **hospital\_hospital\_patient\_data\_bucket**. The main content area displays the **Schema (33)** for the selected table. A search bar labeled "Filter schemas" is present. The schema is presented as a table with columns: #, Column name, Data type, Partition key, and Comment. The table lists 20 columns, with the 'symptoms' column highlighted in blue. At the top right of the schema section, there are buttons for "Edit schema as JSON" and "Edit schema".

| #  | Column name                | Data type | Partition key | Comment |
|----|----------------------------|-----------|---------------|---------|
| 1  | patient_id                 | int       | -             | -       |
| 2  | name                       | string    | -             | -       |
| 3  | age                        | int       | -             | -       |
| 4  | gender                     | string    | -             | -       |
| 5  | event                      | string    | -             | -       |
| 6  | reason                     | string    | -             | -       |
| 7  | symptoms                   | array     | -             | -       |
| 8  | department                 | string    | -             | -       |
| 9  | is_emergency               | boolean   | -             | -       |
| 10 | critical_condition         | boolean   | -             | -       |
| 11 | blood_pressure             | string    | -             | -       |
| 12 | oxygen_level               | int       | -             | -       |
| 13 | insurance_provider         | string    | -             | -       |
| 14 | payment_method             | string    | -             | -       |
| 15 | arrival_time               | string    | -             | -       |
| 16 | treatment_duration_minutes | int       | -             | -       |
| 17 | discharge_notes            | string    | -             | -       |
| 18 | doctor_id                  | int       | -             | -       |
| 19 | doctor_name                | string    | -             | -       |
| 20 | room_number                | string    | -             | -       |

**Glue Table Schema**

#### 4. SQL Querying and Daily Report Generation

- Deployed a second AWS Lambda function to execute Athena queries every 24 hours, triggered by Amazon EventBridge.
- Generated 17 analytical CSV reports summarizing patient inflow, department load, financial payments, critical cases, and bed usage.
- Reports were stored automatically in a new S3 folder.

The screenshot displays the AWS Athena Query Editor interface. The SQL query being executed is:

```
1 SELECT bed_type,
2        COUNT(*) AS total_patients,
3        ROUND(AVG(days_in_hospital), 2) AS avg_stay
4 FROM hospital_hospital_patient_data_bucket
5 GROUP BY bed_type;
```

The query has been completed successfully. The results are displayed in a table with 5 rows and 4 columns: #, bed\_type, total\_patients, and avg\_stay.

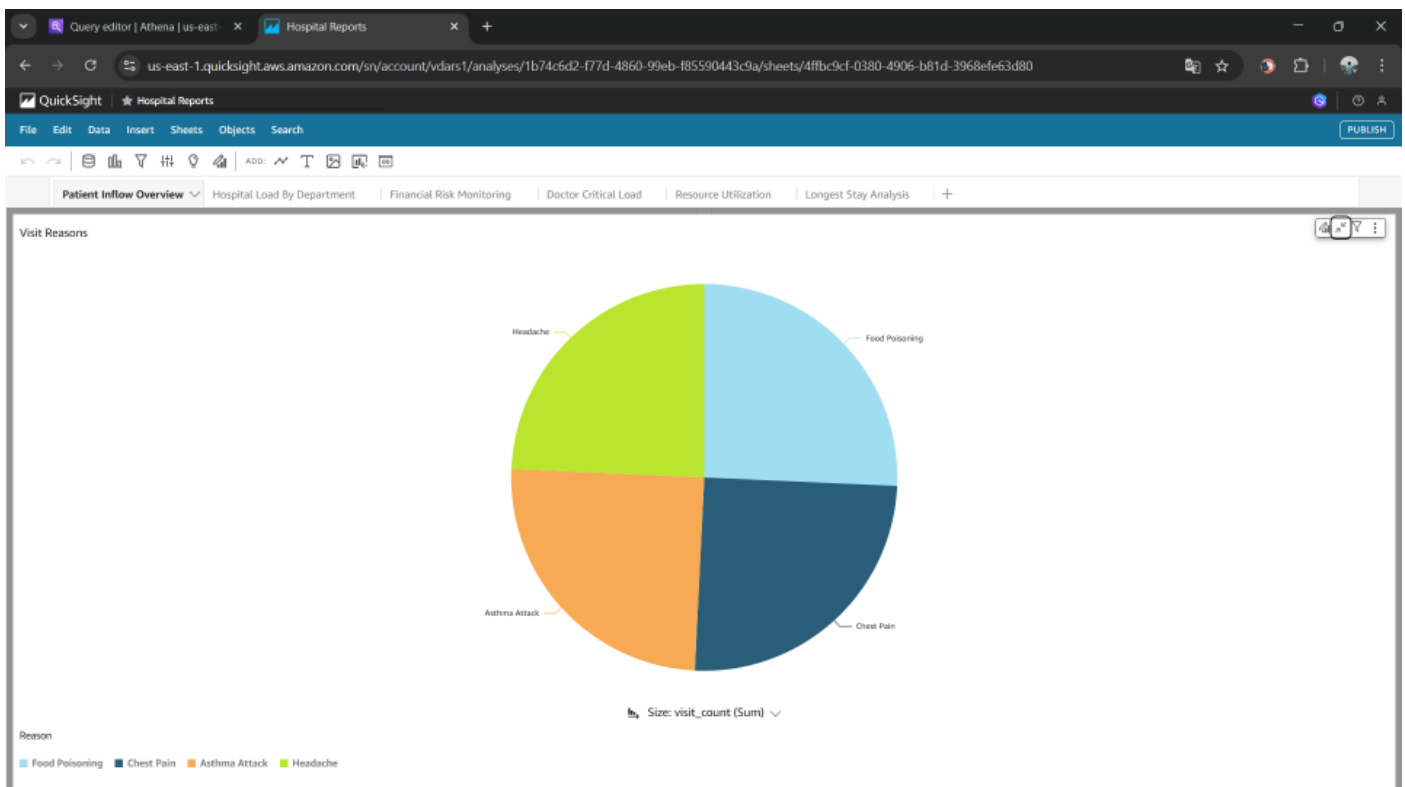
| # | bed_type     | total_patients | avg_stay |
|---|--------------|----------------|----------|
| 1 |              | 5489           |          |
| 2 | ICU          | 1412           | 3.56     |
| 3 | Private      | 1377           | 3.51     |
| 4 | General      | 1331           | 3.63     |
| 5 | Semi-Private | 1380           | 3.6      |

The interface also shows query statistics: Time in queue: 102 ms, Run time: 896 ms, Data scanned: 4.50 MB. There are buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'. The 'Query results' tab is active, and the 'Query stats' tab is also visible.

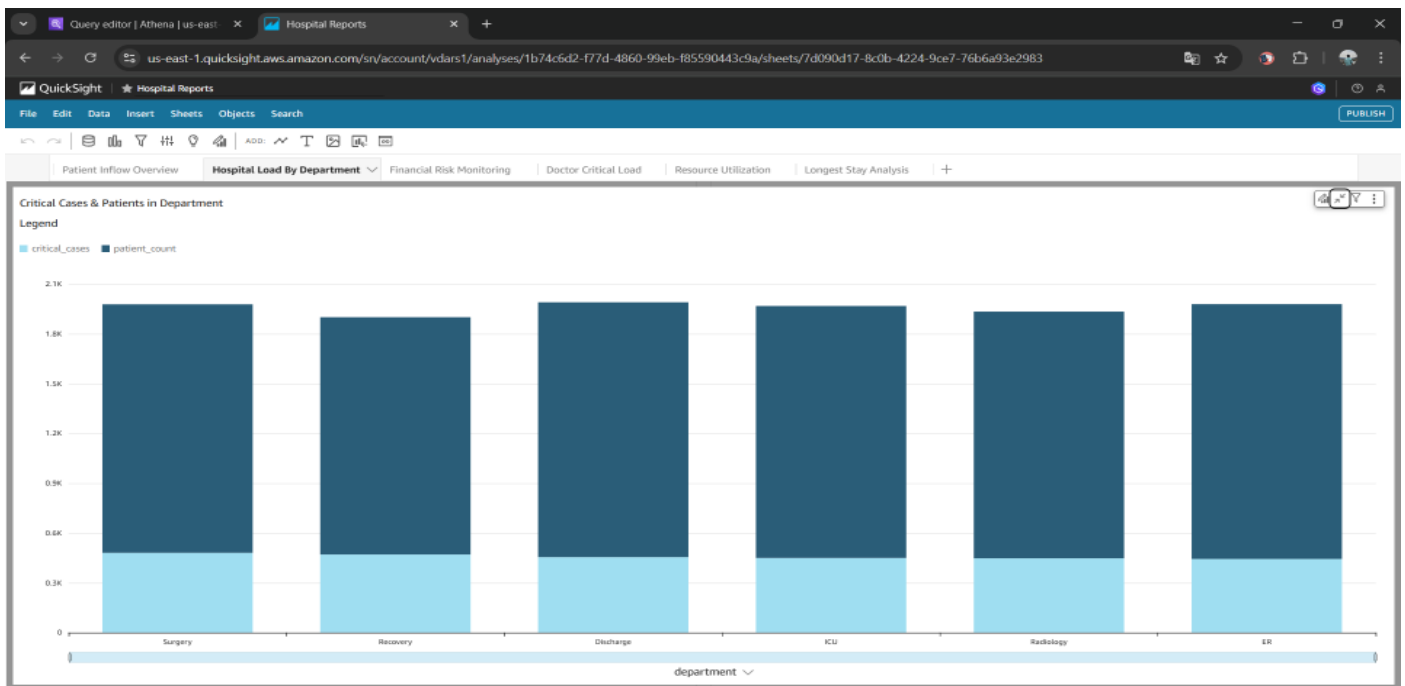
Sample Query in Athena

## 5. Dashboard Creation in Quicksight

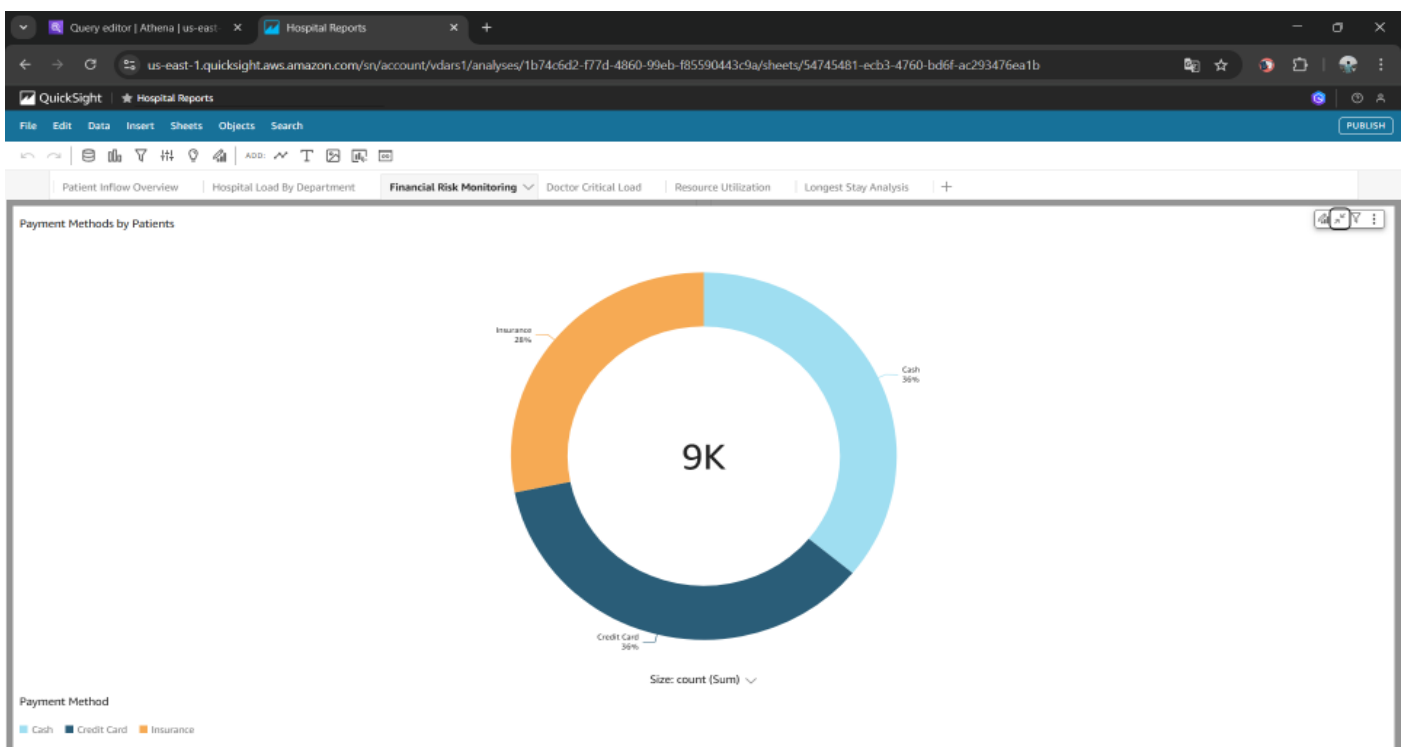
- Imported the generated reports as datasets in QuickSight and created six structured dashboards:
  - Patient Inflow Overview
  - Hospital Load by Department
  - Financial Risk Monitoring
  - Doctor Critical Load & Emergency Outcomes
  - Resource Utilization
  - Longest Stay Analysis
- Visualizations included KPIs, pie charts, bar charts, and stacked bars, ensuring dynamic and actionable insights.



**Visit Reasons**



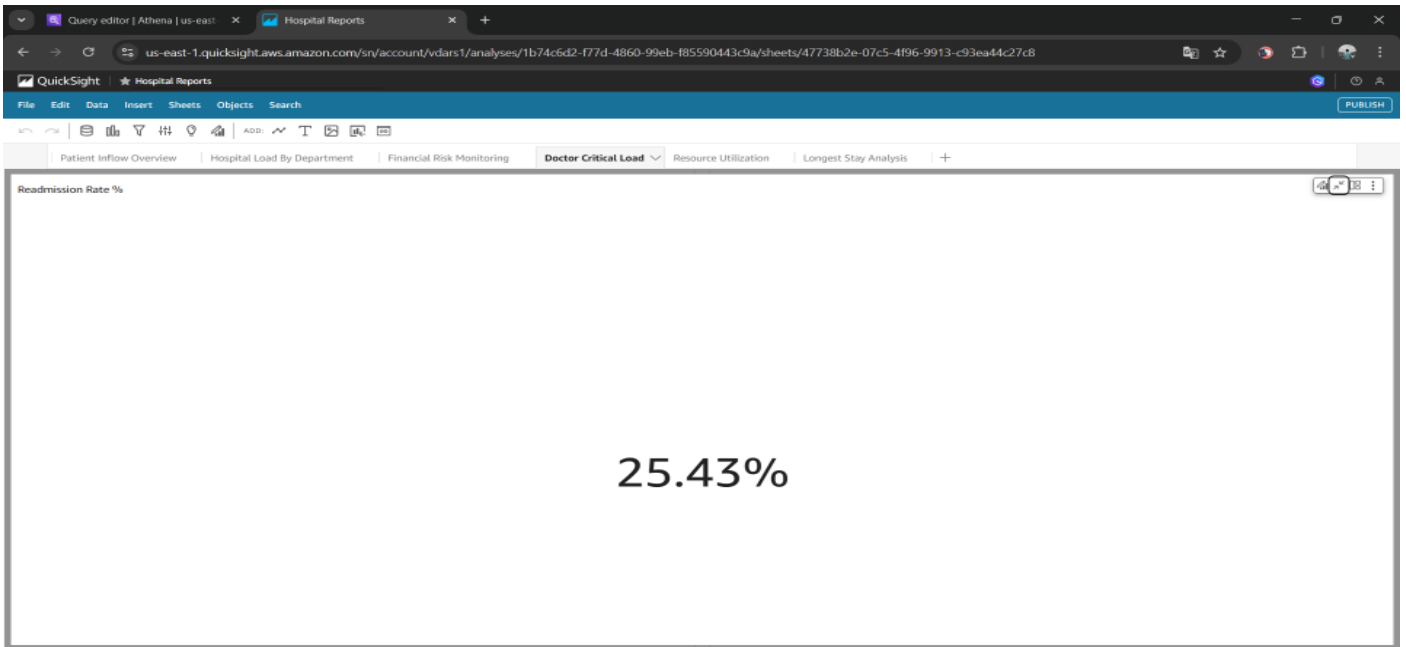
## Critical Cases & Patients



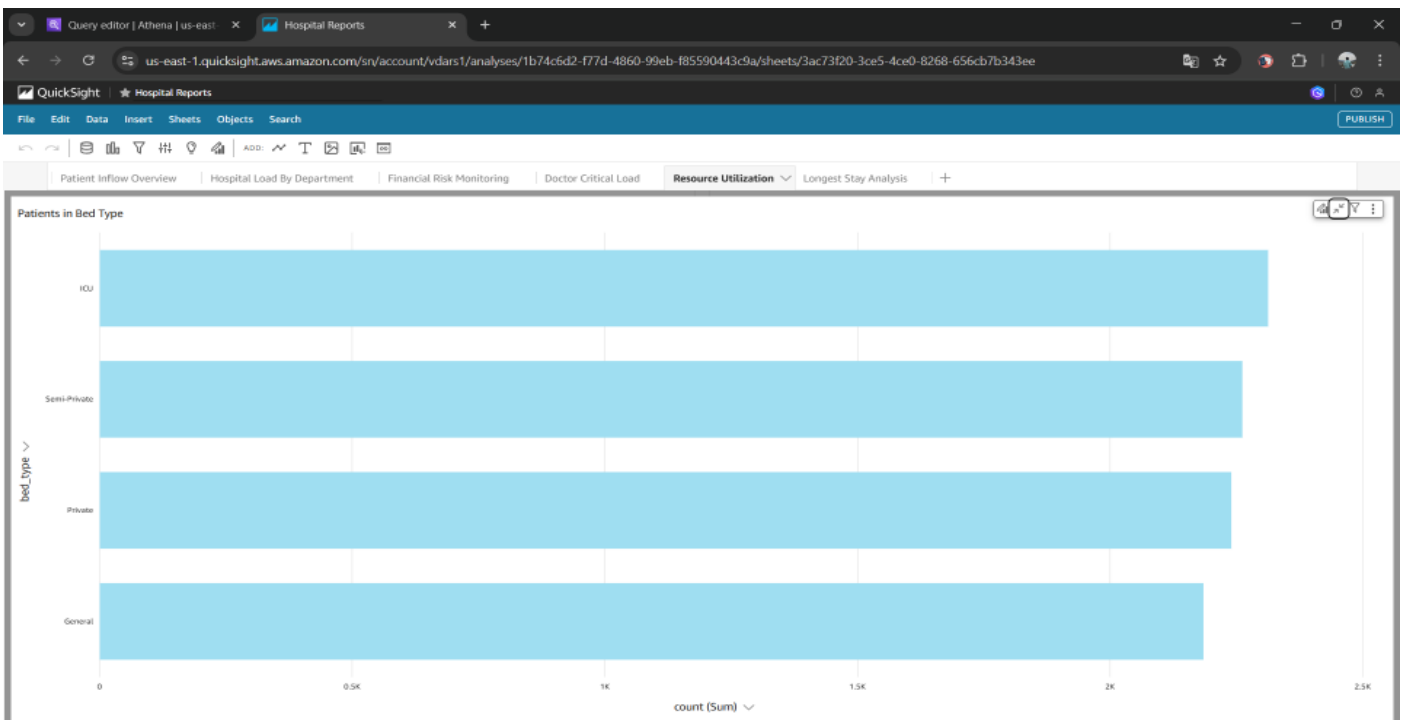
## Payment Methods



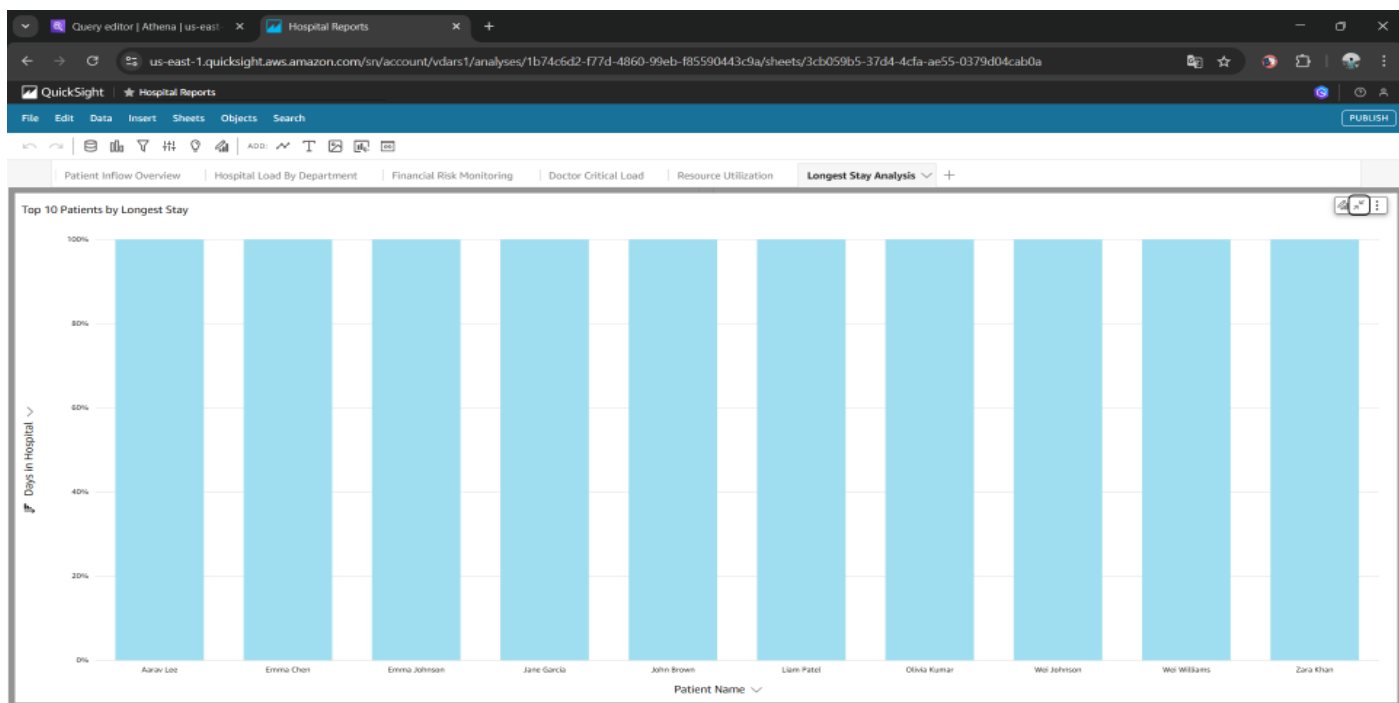
15



**Readmission Rate %**



**Patients By Bed Type**



**Top 10 Patients by Longest Stay**

## Discussion

This project successfully demonstrated the use of AWS serverless technologies to automate real-time hospital analytics. Despite minor challenges like managing Lambda execution limits, the system achieved seamless data streaming, automated reporting, and dynamic dashboarding, fulfilling real-world healthcare analytics needs within Free Tier constraints.

## Conclusion

This project demonstrated that real-time hospital data analytics can be effectively achieved using AWS serverless architecture. By automating data flow, reporting, and visualization, the platform provides hospitals with timely, actionable insights while maintaining cost-efficiency and scalability within a cloud-native environment.

## Contributions/References

### AWS Lambda Documentation

<https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>

### Amazon Kinesis Firehose Documentation

<https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html>

### AWS Glue Documentation

<https://docs.aws.amazon.com/glue/latest/dg/what-is-glue.html>

### Amazon Athena SQL Reference

<https://docs.aws.amazon.com/athena/latest/ug/queries.html>

### Amazon QuickSight Documentation

<https://docs.aws.amazon.com/quicksight/latest/user/welcome.html>

### AWS Developer Forums (for community discussions and troubleshooting)

<https://repost.aws/>