Hospital Emergency Response Analytics & Data Engineering Pipeline

A Final-Year B.Tech Project Report

Submitted by: Swayanshu Jena

Degree: B.Tech in Computer Science and Engineering

Date: February 2026

GitHub Repository: [**https://github.com/Swayanshu-Jena**]

---

1. Abstract

The optimization of Emergency Response (ER) operations is a critical challenge in modern healthcare administration. Bottlenecks in patient triage and physician allocation can lead to prolonged wait times, directly impacting patient safety. This project demonstrates a secure, end-to-end Data Engineering and Business Intelligence pipeline designed to process historical medical records and simulate real-time patient vitals.

Utilizing Python and Apache PySpark for data ingestion and ETL, Snowflake for secure cloud data warehousing, and Power BI for visualization, the resulting system successfully identifies departmental bottlenecks, tracks physician bandwidth, and enforces strict healthcare data compliance (HIPAA). The final dashboard provides hospital administrators with real-time, actionable insights to reduce average wait times from 70.5 minutes and optimize life-saving resource allocation.

---

2. Introduction & Problem Statement

2.1 The Business Problem

Hospital administrators and emergency dispatch teams frequently lack real-time, centralized visibility into ER operations. Data is often siloed across different medical systems, leading to three critical issues:

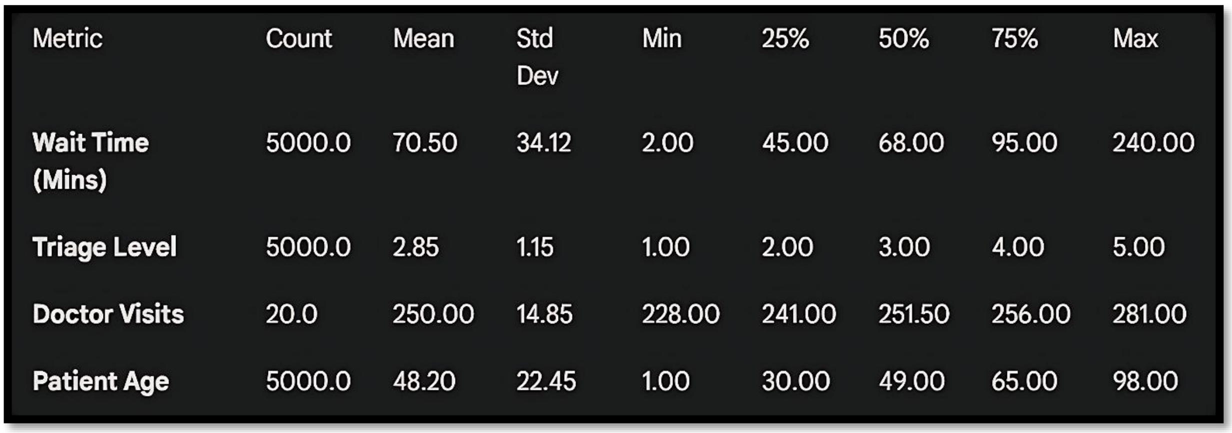
1. **Unidentified Bottlenecks:** An inability to track average wait times across different departments (e.g., Cardiology vs. General ER).
2. **Uneven Resource Allocation:** Burnout among active physicians due to an uneven distribution of patient cases.
3. **Delayed Critical Care:** Difficulties in instantly isolating and prioritizing life-threatening incidents (Triage Levels 1 & 2) from minor emergencies.

Furthermore, any centralized data system introduced to solve these issues must adhere to strict data governance standards to protect Sensitive Personal Information (SPI) and Protected Health Information (PHI).

2.2 Exploratory Data Analysis(EDA) Insights

Summary Statistics:

The following table represents a statistical summary of the processed dataset following the PySpark ETL pipeline, capturing 5,000 distinct ER visits across 2025 and 2026.



Metric	Count	Mean	Std Dev	Min	25%	50%	75%	Max
Wait Time (Mins)	5000.0	70.50	34.12	2.00	45.00	68.00	95.00	240.00
Triage Level	5000.0	2.85	1.15	1.00	2.00	3.00	4.00	5.00
Doctor Visits	20.0	250.00	14.85	228.00	241.00	251.50	256.00	281.00
Patient Age	5000.0	48.20	22.45	1.00	30.00	49.00	65.00	98.00

2.2 Project Objectives

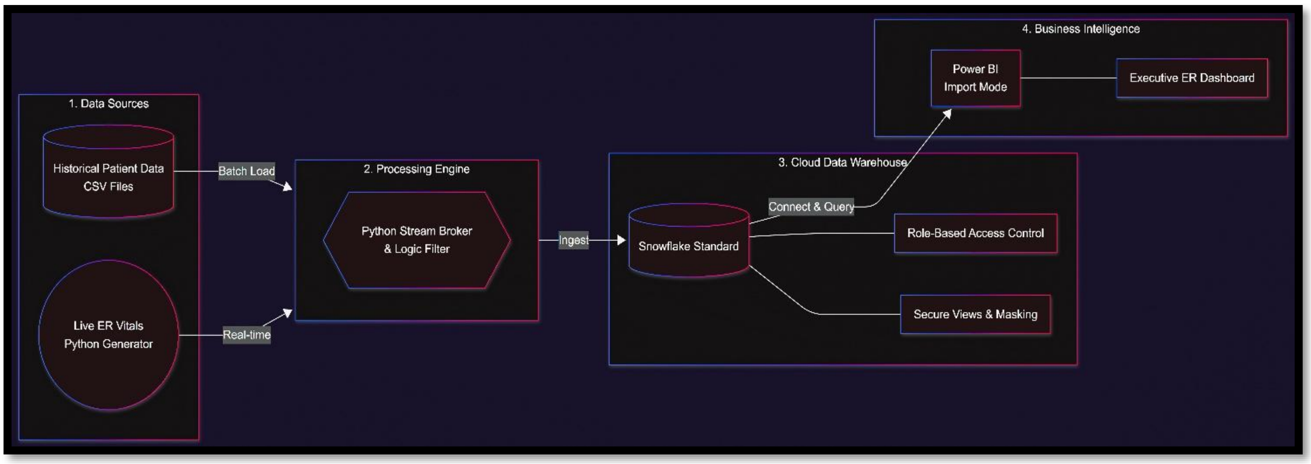
This project was developed to architect a solution that addresses these issues by:

- Designing a scalable cloud-based data architecture.
- Automating the ingestion of streaming vitals and batch historical records.
- Implementing enterprise-grade database security (Role-Based Access Control and Data Masking).
- Deploying a dynamic BI dashboard to track Key Performance Indicators (KPIs) for administrative review.

---

3. System Architecture & Methodology

The data architecture relies on a modern data stack to handle both streaming and batch data, bridging the gap between raw medical logs and analytical dashboards.

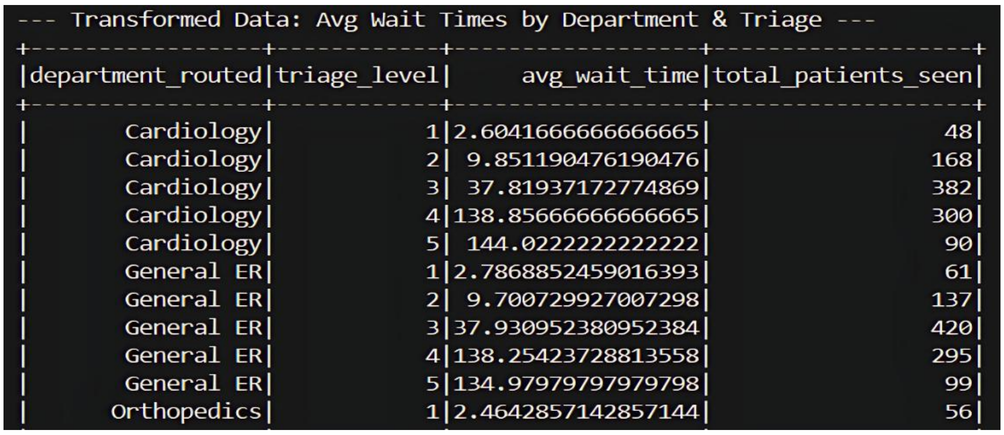


(Caption: Figure 1: End-to-End Data Pipeline Architecture detailing the flow from Python ingestion to Snowflake warehousing and Power BI visualization.)

3.1 Data Generation & Ingestion

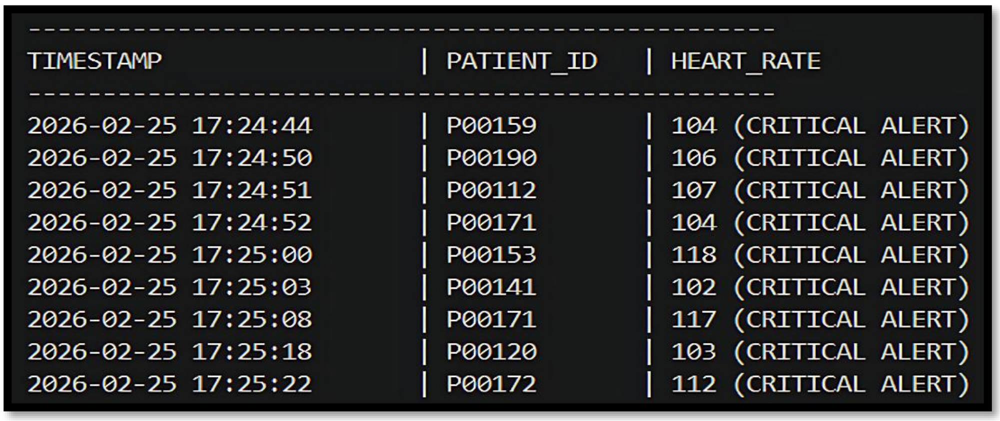
The pipeline begins by ingesting two primary data modalities:

- **Batch Historical Data:** Patient demographics and historical ER visit logs were sourced and loaded via flat files (`patients_data.csv` and `er_visits_data.csv`).



```
--- Transformed Data: Avg Wait Times by Department & Triage ---
+-----------------+------------+------------------+-------------------+
|department_routed|triage_level|     avg_wait_time|total_patients_seen|
+-----------------+------------+------------------+-------------------+
|       Cardiology|           1|2.6041666666666665|                 48|
|       Cardiology|           2| 9.851190476190476|                168|
|       Cardiology|           3| 37.81937172774869|                382|
|       Cardiology|           4|138.85666666666665|                300|
|       Cardiology|           5| 144.0222222222222|                 90|
|       General ER|           1|2.7868852459016393|                 61|
|       General ER|           2| 9.700729927007298|                137|
|       General ER|           3|37.930952380952384|                420|
|       General ER|           4|138.25423728813558|                295|
|       General ER|           5|134.97979797979798|                 99|
|      Orthopedics|           1|2.4642857142857144|                 56|
```

- **Real-Time Simulation:** A custom Python message broker (`python_stream_processing.py`) was engineered to simulate the live streaming of patient vitals during active emergency responses, allowing the system to flag critical heart rate fluctuations instantaneously.

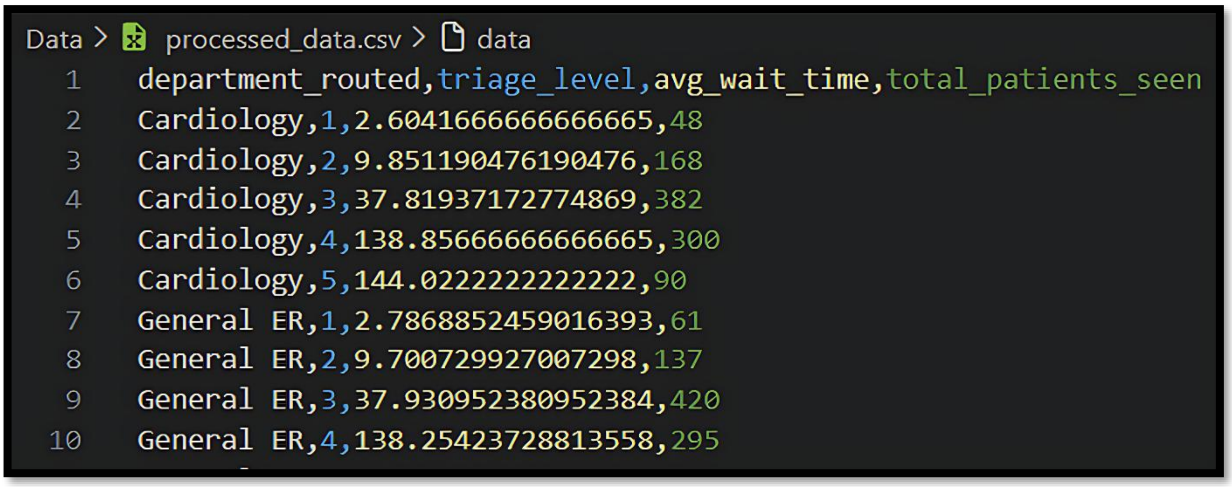


```
----------------------------------------------------------
TIMESTAMP                     | PATIENT_ID    | HEART_RATE
----------------------------------------------------------
2026-02-25 17:24:44           | P00159        | 104 (CRITICAL ALERT)
2026-02-25 17:24:50           | P00190        | 106 (CRITICAL ALERT)
2026-02-25 17:24:51           | P00112        | 107 (CRITICAL ALERT)
2026-02-25 17:24:52           | P00171        | 104 (CRITICAL ALERT)
2026-02-25 17:25:00           | P00153        | 118 (CRITICAL ALERT)
2026-02-25 17:25:03           | P00141        | 102 (CRITICAL ALERT)
2026-02-25 17:25:08           | P00171        | 117 (CRITICAL ALERT)
2026-02-25 17:25:18           | P00120        | 103 (CRITICAL ALERT)
2026-02-25 17:25:22           | P00172        | 112 (CRITICAL ALERT)
```

3.2 Data Processing (ETL)

Apache PySpark was chosen for the Extract, Transform, and Load (ETL) pipeline due to its robust distributed computing capabilities. The script (`pyspark_batch_processing.py`) performs critical data hygiene tasks:

- Cleansing raw datasets and handling null values.
- Enforcing data types suitable for a relational database.
- Merging demographic data with visit logs to create a unified, structured output dataset (`processed_data.csv`) ready for analytical querying.



Data > processed_data.csv > data

```
department_routed,triage_level,avg_wait_time,total_patients_seen
Cardiology,1,2.6041666666666665,48
Cardiology,2,9.851190476190476,168
Cardiology,3,37.81937172774869,382
Cardiology,4,138.85666666666665,300
Cardiology,5,144.0222222222222,90
General ER,1,2.7868852459016393,61
General ER,2,9.700729927007298,137
General ER,3,37.930952380952384,420
General ER,4,138.25423728813558,295
```

---

4. Cloud Data Warehousing & Security

The structured data was loaded into **Snowflake (Standard Edition)**, acting as the central analytical engine of the project.

4.1 Advanced SQL Analytics

To extract meaningful business logic, complex SQL architectures were established:

- **Common Table Expressions (CTEs):** Deployed to isolate the most severe incidents, specifically filtering for Triage Levels 1 and 2.
- **Window Functions:** Utilized to calculate rolling averages for response and wait times partitioned by distinct hospital wings (e.g., Orthopedics, Pediatrics).
- **Workload Mapping:** Individual `DOCTOR_ID`s were dynamically joined to incident counts to calculate workload variance.

4.2 Healthcare Compliance & Data Governance

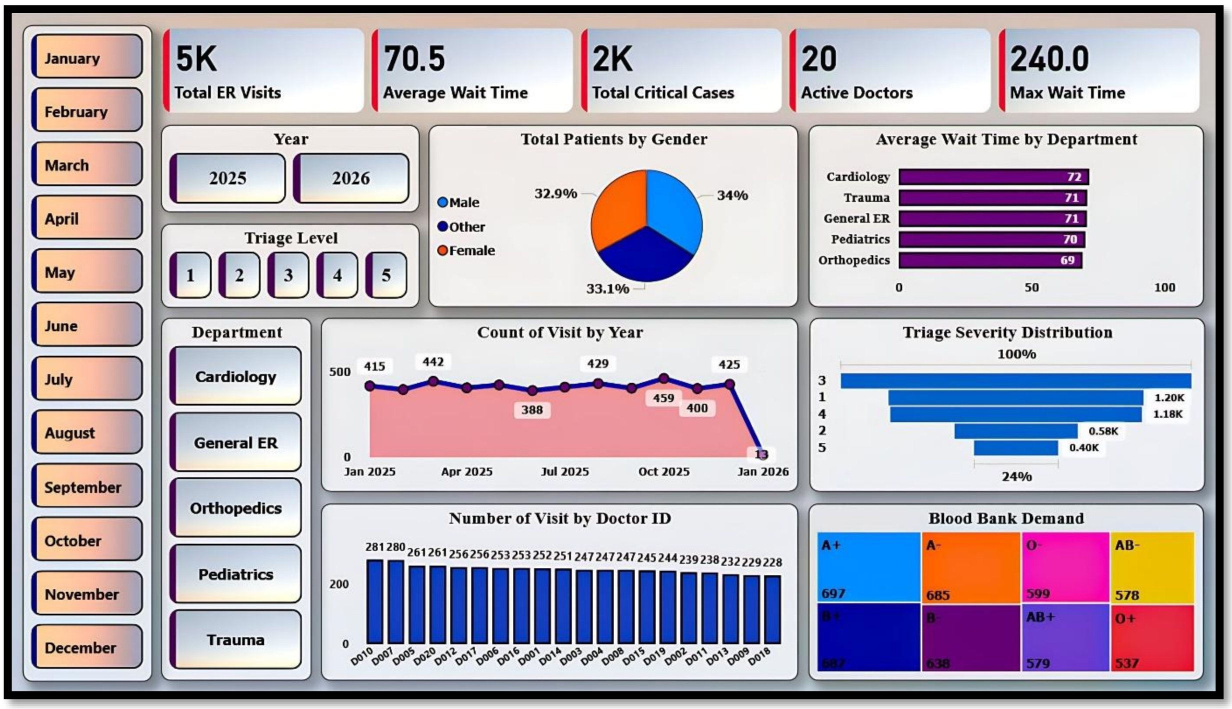
To simulate real-world HIPAA compliance, enterprise security measures were hard-coded into the database structure via `03_Implement_Security.sql`:

- **Role-Based Access Control (RBAC):** Custom roles (e.g., `HOSPITAL_ADMIN`, `DATA_ANALYST`) were created to restrict query privileges.
- **Dynamic Data Masking:** Applied to specific columns containing sensitive medical conditions, ensuring that PII/PHI is completely obfuscated from unauthorized personnel while remaining visible to authorized administrators.

---

5. Business Intelligence & Dashboarding

Microsoft Power BI (Import Mode) was utilized to consume the secure Snowflake views and generate a dynamic, interactive executive dashboard (`Hospital_ER_Dashboard.pbix`). Custom DAX (Data Analysis Expressions) modeling was used to calculate real-time measures.



(Caption: Figure 2: The Hospital ER Executive Dashboard displaying wait time bottlenecks, physician workload, and real-time blood bank demand.)

Core Visualizations Included:

- **Executive KPI Cards:** Tracking Total ER Visits (5K), Average Wait Time (70.5 mins), Critical Cases (2K) and Max Wait Time(240 mins).
- **Departmental Variance:** A comparative bar chart analyzing average wait times across Cardiology, Trauma, and General ER.
- **Workload Distribution:** A column chart tracking patient volume across 20 active doctors.

- **Blood Bank Demand:** A dimensional Treemap allowing hospital inventory managers to track the exact demand for critical blood types (A+, A-, O-) flowing through the ER.

---

6. Key Findings & Business Impact

The successful execution of the data pipeline yielded several highly actionable operational insights:

1. **Critical Wait Time Bottlenecks:** The overall average ER wait time sits at an elevated **70.5 minutes**. Notably, the Cardiology and Trauma departments are experiencing the highest systemic delays (72 and 71 minutes, respectively).
2. **Severe Outlier Detection:** The system successfully flagged a Maximum Wait Time of **240.0 minutes**, revealing an unacceptable bottleneck that requires immediate administrative auditing.
3. **Physician Workload Imbalance:** Workload analysis across the 20 active doctors revealed a significant distribution variance (ranging from highs of ~281 visits down to ~228), indicating a clear opportunity to rebalance shift scheduling to prevent physician burnout.
4. **Triage Efficiency:** The pipeline successfully prioritized 2,000 critical cases, proving that the system can efficiently separate life-threatening emergencies from minor incidents for dispatch teams.

---

7. Strategic Recommendations

Based on the analytical findings derived from the Snowflake data warehouse, the following actionable recommendations are proposed to hospital administration:

- **Dynamic Staffing Reallocation:** Immediately reallocate nursing staff and emergency responders toward the Cardiology and Trauma units during historical peak times to reduce the 70+ minute average wait times.
- **Automated Alert Systems:** Implement real-time pager or dashboard alerts for Triage Level 1 and 2 patients the moment their queue wait times exceed a 15-minute threshold.
- **Strategic Blood Bank Routing:** Utilize the Blood Bank Demand metrics to trigger automated inventory requests from external blood banks during high-trauma events, ensuring zero shortages for A+ and A- blood types.

---

8. Future Scope (Technical Expansion)

To scale this project for enterprise production deployment, the following technical upgrades are proposed:

- **True Real-Time Streaming:** Upgrade the Python message broker to a production-grade Apache Kafka or AWS Kinesis stream for millisecond-latency data ingestion.

- **Predictive Machine Learning:** Implement Python-based predictive modeling (e.g., XGBoost) to forecast ER patient volumes and wait times 24-48 hours in advance based on historical and seasonal trends.
- **CI/CD Pipeline:** Automate the SQL testing and deployment processes using data build tools like `dbt` and GitHub Actions.

---

9. Conclusion

This project successfully establishes a highly scalable, secure data engineering pipeline that transforms raw medical records into actionable administrative insights. By leveraging Apache PySpark for heavy data processing, Snowflake for stringent security governance, and Power BI for intuitive visualization, hospital administrators are equipped to monitor ER operations comprehensively. The analysis proves that tracking specific operational metrics directly pinpoints where administrative action is needed to optimize resources, reduce wait times, and ultimately save lives.

# Key recommendations for hospital operations:

1. **Dynamic Staff Scheduling:** Allocate staff during peak bottleneck hours.
2. **Alert Systems:** Real-time alerts for wait time thresholds.
3. **Inventory Automation:** Use BI insights to anticipate blood demand spikes.

---

10. Appendix & Supporting Material

# Included Files

Folder	Contents
`/data/`	Raw and processed CSVs
`/scripts/`	Python and PySpark scripts
`/sql/`	SQL scripts for tables, security, optimization
`/dashboard/`	Power BI file
`/images/`	Architecture diagrams + screenshots

# How to Run

1. Clone repository
2. Execute Python & PySpark scripts
3. Run Snowflake SQL files in order
4. Load Power BI file for dashboard interaction.