Final project iteration 02

Emergency Department Database and Analysis System

Team 22
Suk Jin Mun (NUID: 002082427),
Xiaobai Li(NUID: 002036414),
Shaobo (Ben) Chen (NUID:001836752)

October 25, 2025
DS 5110: Introduction to data management and processing

1 Project kickoff

1.1 Specific goals and expected outcomes

This project creates an Emergency Center Analysis System combining a web app, backend server, and structured database. It enables users to view patient records, referrals, and disease trends through interactive charts. The backend processes data and computes metrics like referral and waiting times, while the database stores all core emergency data.

By designing and analyzing a hospital emergency center database, this project seeks to generate data-driven insights that could inform better resource allocation, patient flow management, and treatment prioritization. The project goes beyond a single metric, aiming to understand the overall performance of the ER from multiple perspectives — patient outcomes, waiting time distribution, referral patterns, and disease characteristics.

1.2 Key deliverables and deadlines for each phase

Iteration 02 (completes October 25, 2025): proposal, tracker, GitHub repository.
Iteration 03 (completes October 30, 2025): Database schema, simulated data, at least 8 analytical SQL queries such as patient counts by urgency level, average wait times, staff workload distribution, visit trends, repeat patients, case ratios, resource utilization, and arrival/completion time-series.
Iteration 04 (completes November 12, 2025): data cleaning, statistical models, visualizations, optional Flask web app.
Iteration 05 (completes November 29 2025): final report, presentation, optional web demo.

1.3 Team capabilities and alignment with objectives

The team has foundational database, SQL, Python, and visualization skills from coursework. Roles: Suk Jin (backend API developer) handles API endpoints, business logic, calculations, and serves as integration layer; Shaobo (database architect) manages schema, data generation, ETL, and SQL queries; Xiaobai (frontend developer) creates user interactive visualizations and web interface calling backend API. The 3-tier architecture ensures frontend accesses database only through backend API. Growth areas include statistical modeling, Poisson regression, and Flask API development.

1.4 Dataset availability and requirements

The project generates simulated ED data to avoid HIPAA concerns, using distributions from https://data.gov/, CDC's National Hospital Ambulatory Medical Care Survey, and Emergency Severity Index. The methodology applies small variations to published statistics, generating records with realistic distributions for demographics, arrival patterns, vital signs, urgency levels, and wait times. The data generation process creates approximately 5,000 to 10,000 patient visit records spanning one year of simulated operations. Demographic attributes include age, gender, and insurance status following national ED patient distributions. Arrival times follow realistic patterns with peaks during evening hours and weekends. Vital signs correlate appropriately with assigned urgency levels. Wait times reflect typical ED operations accounting for patient volume and urgency. Data cleaning demonstrates handling missing alues, validating realistic ranges (heart rate 40-200 bpm, for instance), removing duplicates, standardizing formats and enforcing referential integrity. This synthetic approach demonstrates ETL skills while avoiding privacy compliance issues.

2 Team discussions

2.1 Core skills each team member contributes

Suk Jin: backend API development, Python, Flask API endpoints, request/JSON handling, business logic, Pandas/NumPy, database access layer, Git. Xiaobai: frontend development, visualization, HTML/JavaScript, Flask interfaces calling backend API, presentations. Shaobo: database architecture, ERD, normalization, SQL, data generation, ETL pipelines.

2.2 How expertise supports specific tasks and components

Suk Jin: trains 1-2 simple models (optional) such as logistic regression and Poisson regression, designs Flask API endpoints receiving frontend requests/returning JSON, implements business logic validation, accesses database on behalf of frontend, calculates statistics (conversion rates, averages), auto-calculates wait times. Xiaobai: creates user interactive visualizations to present hospital activity trends and resource usage insights using plotly, builds Flask frontend with HTML/CSS calling backend API for operations and displaying results. Frontend cannot directly access database. Shaobo: implements schema, generates ED data with realistic distributions, builds ETL pipelines, develops SQL queries executed by backend API.

2.3 Missing skills and potential challenges

Primary challenges: maintaining communication for timely completion, coordinating dependencies (Shaobo's schema/data before modeling), time management across projects. Learning requirements not yet covered: business logic validation, Poisson regression, database indexing, complex SQL (window functions, CTEs), model evaluation (cross-validation, confusion matrices), optional Flask API development with 3-tier architecture.

2.4 Tools and technologies team experience

Team experience: Python, Pandas, NumPy, SQLite, Matplotlib, Plotly, Seaborn, Git, Overleaf, Flask, HTML/CSS/JavaScript.
Core learning: scikit-learn, statsmodels. Optional: RESTful API (endpoints, request/JSON handling).

2.5 Programming languages and platforms selection

Python 3.9, SQLite, Jupyter, optional 3-tier web application architecture (Frontend Backend API Database). SQLite provides file-based database management requiring no separate server process (MySql platforms like Aiven can be used for web deployment if needed). Pandas enables efficient data manipulation. Plotly creates user interactive visualizations. Scikit-learn implements classification and regression. Statsmodels provides Poisson regression. Git tracks changes. Overleaf facilitates collaborative LaTeX preparation.

3 Skills and tools assessment

3.1 External resources for areas lacking expertise

Course materials, office hours, Python documentation, Stack Overflow, Flask documentation (if web track pursued). The team leverages course lectures on database normalization, SQL optimization, and statistical modeling. Office hours provide clarification on complex concepts. Python library documentation offers API references. Stack Overflow provides community solutions. Pair programming facilitates knowledge transfer.

3.2 Tools, frameworks, and libraries fitting project scope

SQLite (database), Pandas/NumPy (data manipulation/generation), Plotly/Matplotlib/Seaborn (visualization), scikit-learn/statsmodels (modeling), Git/GitHub (version control), Flask (web application framework), HTML/CSS/JavaScript (frontend).

3.3 Ensuring team proficiency with selected tools

Build on coursework experience, official documentation, pair programming sessions ensure code quality and knowledge sharing among members.

3.4 Task assignments and role clarity

Suk Jin (backend API) - trains models, designs API endpoints receiving requests/returning JSON, implements business logic validation, accesses database for frontend, calculates statistics (conversion rates, average wait times), auto-calculations (wait time from timestamps), GitHub management. Xiaobai (frontend) - creates user interactive visualizations to present hospital activity trends and resource usage insights, builds Flask frontend calling backend API, presentations. Shaobo (database) - schema/ERD design, generates ED data using https://data.gov/, ETL pipelines, SQL queries. Dependencies: Shaobo completes schema/data before Suk Jin's modeling; Suk Jin's API ready for Xiaobai's frontend. Weekly meetings coordinate progress. Clear communication channels ensure timely issue resolution.

4 Initial setup

4.1 Development environment requirements

Anaconda Python 3.9+, Jupyter Notebook, VS Code, SQLite, Overleaf, 8GB RAM. Optional: Flask via pip if web track pursued.

4.2 Version control configuration

Git with GitHub repository at
https://github.com/SukjinMun/DS5110-Final-Project

Python .gitignore, standard Git workflow (pull, add, commit, push), comprehensive README. Directory structure: data(generated datasets), scripts(ETL and calculation modules), notebooks(analysis), reports(LaTeX/PDFs), web(optional API/frontend). Team members have write access. Gitignore excludes data files, bytecode, checkpoints, and configurations.

5 Submission verification

This submission meets all Iteration 02 requirements. The proposal covers all required sections, the progress tracker documents tasks in iterations 02-05, and the GitHub repository contains complete documentation. All deliverables have been reviewed by the team and are ready for stakeholder review.