Mini Project 02

Search and Aggregate Students (Search and Aggregation Functions)

CSC-361 Database System Design

Dr. Qixin Deng

Due by: Mar. 15, 2026

Overview. In this mini project, you will extend the course demo database application by implementing two realistic and commonly used database features:

1. **Search student records** using `(WHERE + LIKE + REGEXP)` with user-provided information.
2. **Compute summary statistics** using **SQL Aggregation** (`COUNT`, `AVG`, `MIN`, `MAX`)

Goal: practice the full workflow

$$\textit{HTML Form} \rightarrow \textit{FastAPI Route} \rightarrow \textit{Database Function} \rightarrow \textit{SQL}.$$

You must finish this project Independently!

Part A: Multi-Mode Search (WHERE + LIKE + REGEXP)

In this part, you must implement a student search feature that supports **three different SQL search techniques**:

1. Exact match using `WHERE column = value`
2. Pattern match using `LIKE`
3. Regular expression match using `REGEXP`

The goal is to understand the difference between simple filtering, wildcard matching, and full regex matching in MySQL.

Web UI Requirements

- Add a search form above the student table.
- The form must use HTTP `GET`.
- The form must contain:
  - A text input named `keyword`
  - A dropdown selector named `mode`

The dropdown must allow the user to choose:

Mode	Meaning
Exact	Use `=`
Like	Use `LIKE`
Regex	Use `REGEXP`

If the keyword is empty, show all students.

Search Requirements

Your search must check **at least two columns**, a student should be included if the condition matches **any** supported column (combine conditions using `OR`).

Mode 1: Exact Match

Use SQL of the form:

```
WHERE name = ?  OR major = ?  OR city = ?
```

Mode 2: LIKE Match

Use SQL `LIKE` with wildcards:

```
WHERE name LIKE ? OR major LIKE ? OR city LIKE ?
```

You must automatically wrap the keyword with:

```
%keyword%
```

Mode 3: REGEXP Match (MySQL)

Use MySQL regular expression matching:

```
WHERE name REGEXP ? OR major REGEXP ? OR city REGEXP ?
```

Your regex mode must support features listed below:

- Anchors: `^...`, `...$`
- Alternation: `A|B`
- Character classes: `[A-Za-z]`
- Repetition: `+`, `*`, `{m,n}`

Error Handling

If the user selects Regex mode and enters an invalid pattern:

- Your server must not crash.
- Show a friendly error message.

Part B: Aggregation Dashboard

You must implement an aggregation feature that shows summary statistics about the students currently stored in the database.

Web UI Requirements

- Add a small **summary panel** above or beside the student table.
- The panel must display all aggregated values listed below computed from SQL:
  - Total number of students per major: `COUNT(*)`
  - Average GPA (ignore NULL GPAs): `AVG(gpa)`
  - Minimum GPA: `MIN(gpa)`
  - Maximum GPA: `MAX(gpa)`
  - Average credits: `AVG(credits)`

Aggregation Requirements (Required)

- Your aggregation must be computed **in SQL**, not in Python.
- NULL-handling must be correct:
  - `AVG(gpa)` should naturally ignore NULL GPAs (typical SQL behavior).
  - If there are **no** non-NULL GPAs, your UI should display something reasonable (e.g., "N/A").

Submission Requirements

Submit a single `.zip` file containing the following items:

1. **Source Code** (your full project folder).
2. **Report (PDF)** explaining all modifications you made.
   - Provide sufficient explanation for each change to demonstrate your design idea.
   - Submitting code means you fully understand what you submitted.
3. **Short Demo Video** (screen recording).
   - Record the entire screen.
   - Show the project open in your IDE.
   - Manually run the project.
   - Show **two examples per search mode for Part A**.
   - Show **all aggregation functions for Part B**.