Lab 1 — Environment Setup & First SQL (SQLite)

Module I · Week 1 · Duration: 2 hours Deliverable: None — this lab is for environment setup and practice.

Objectives

By the end of this lab you will be able to:

1. Set up and verify your database toolchain (SQLite CLI, DB Browser, Python `sqlite3`, and confirm PostgreSQL is reachable for later labs).
2. Create tables using `CREATE TABLE` and define constraints.
3. Test your tables using simple `INSERT`, `SELECT`, and `DELETE` commands.

Time budget (≈2 h): Part 0 setup ~45 min · Part 1 table creation & testing ~75 min.

Part 0 — Environment setup & verification (≈30 min)

Follow `infrastructure.md` for full install steps. Then verify each tool.

0.1 SQLite CLI

```
sqlite3 --version
```

Create a scratch database and confirm the shell works:

```
sqlite3 lab1.db
sqlite> .databases
sqlite> .quit
```

0.2 DB Browser for SQLite (GUI)

- Launch DB Browser, choose **New Database**, save as `lab1_gui.db`.

0.3 Python `sqlite3`

```
python3 -c "import sqlite3; print('sqlite3 module OK', sqlite3.sqlite_version)"
```

0.4 Confirm PostgreSQL is reachable (for later labs — no data work yet)

```
psql --version        # command exists?
# If installed/running:
psql -U csl303 -c "SELECT version();"   # or note the connection error to fix before Lab 4
```

Checkpoint: SQLite CLI runs, DB Browser opens, Python imports `sqlite3`. PostgreSQL is a *nice-to-have* today; it is **required by Lab 4**.

Part 1 — Create tables and test them (≈75 min)

Work in the SQLite CLI (`sqlite3 lab1.db`) **or** DB Browser's *Execute SQL* tab. Put every statement into your solution script as you go for your own reference.

1.1 Enable foreign keys

SQLite does **not** enforce foreign keys unless you turn them on **per connection**:

```
PRAGMA foreign_keys = ON;
```

1.2 Create two tables

Create `Students` and `Faculty`:

Students

Column	Type	Constraint
`student_id`	INTEGER	PRIMARY KEY
`first_name`	TEXT	NOT NULL
`last_name`	TEXT	NOT NULL
`discipline`	TEXT	

Faculty

Column	Type	Constraint
`faculty_id`	INTEGER	PRIMARY KEY
`first_name`	TEXT	NOT NULL
`last_name`	TEXT	NOT NULL
`department`	TEXT	

```
CREATE TABLE Students (
    student_id INTEGER PRIMARY KEY,
    first_name TEXT NOT NULL,
    last_name  TEXT NOT NULL,
    discipline TEXT
);
-- Task: write the Faculty table yourself.
```

1.3 Testing with simple DML

Now that you have created the tables, you need a way to test if they work as expected and if your constraints are properly enforced. To do this, you will use simple Data Manipulation Language (DML) commands: `INSERT`, `SELECT`, and `DELETE`.

INSERT: Adds a new row of data into a table. *Syntax*: `INSERT INTO table_name (column1, column2) VALUES (value1, value2);` *Example*: `INSERT INTO Students (student_id, first_name, last_name, discipline) VALUES (1, 'Aarav', 'Sharma', 'CSE');`

SELECT: Retrieves data from a table so you can see what is currently stored. *Syntax*: `SELECT * FROM table_name;` (The `*` means "return all columns"). *Example*: `SELECT * FROM Students;`

DELETE: Removes rows from a table based on a condition. *Syntax*: `DELETE FROM table_name WHERE condition;` *Example*: `DELETE FROM Students WHERE student_id = 1;`

Tasks: 1. Insert at least 3 rows into the `Students` table and 3 rows into the `Faculty` table. 2. Use `SELECT * FROM Students;` and `SELECT * FROM Faculty;` to verify your data was inserted correctly. 3. Test your constraints! Try to insert a row with a missing `first_name` (which is `NOT NULL`) or a duplicate `student_id` (which is the `PRIMARY KEY`). Observe the error SQLite gives you to ensure your constraints are working. 4. Insert a student whose `discipline` is 'Physics'.

Then, use a `DELETE` statement to remove them from the table (`WHERE discipline = 'Physics'`). Finally, use `SELECT *` again to verify they are gone.

 Checkpoint: You have successfully created tables, inserted data, tested constraints, and deleted a row.

Submission

There is no formal submission required for Lab 1. Use this session to ensure your local environment is fully working and that you are comfortable interacting with the database using DDL and simple testing commands. Save your scripts for your own reference!