LESSON 60

Running SQL Queries

80 minutes (2 periods) | Python Web Backend | Outcome 3.2 | Outcome 2.16 | Outcome 2.18 | Outcome 1.22 | Outcome 2.6 | Outcome 2.7 | Outcome 2.19

WHAT THIS LESSON DOES

In this lesson, you'll learn to run SQL queries in Visual Studio Code using the SQLite Explorer extension. Follow step-by-step instructions to set up, create and modify tables, manage records, and perform joins with the 'users.db' database.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
3.2	create a basic relational database to store and retrieve a variety of forms of data types
2.16	use data types that are common to procedural high-level languages
2.18	collect, store and sort both continuous and discrete data
1.22	read, write, test, and modify computer programs
2.6	construct algorithms using appropriate sequences, selections/conditionals, loops and operators to solve a range of problems, to fulfil a specific requirement
2.7	implement algorithms using a programming language to solve a range of problems
2.19	test solutions and decisions to determine their short-term and long-term outcomes

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the process of setting up and using SQLite Explorer in Visual Studio Code to manage databases.
- Master the creation, modification, and deletion of tables within a database.
- Develop proficiency in inserting, updating, deleting, and selecting records using SQL commands.
- Gain skills in performing simple joins to combine data from multiple tables.
- Apply SQL query techniques through hands-on challenges to reinforce database management concepts.

BEFORE THE LESSON

- Confirm the SQLite Explorer extension is installed in VS Code on every machine and that right-click gives an Open Database option. Test it against a sample .db before class.
- Have a ready-made users.db with populated users and hobbies tables saved where the class can copy it, so students without their previous project can still take part.
- Check the users table actually has a hobby_id column, otherwise the join steps will fail silently or error.

Running SQL Queries

HOW THE LESSON RUNS

- Introduction
- Open your FlaskSQLiteIntro Project
- Create a New Query
- Creating a Table
- Modifying a Table
- Deleting a Table
- Inserting Records
- Selecting Records
- Updating Records
- Deleting Records
- Simple Joins
- Challenge
- Wrap Up

TEACHING MOVES

- 2. Open your FlaskSQLiteIntro Project. Walk the room while they open the folder. The common trap is opening the .db file by double-click, not right-click Open Database. Show the SQLite Explorer panel appearing so everyone knows where results land.
- 4. Creating a Table. Stress that you must highlight the query before running when several statements share a tab. Point out IF NOT EXISTS makes re-running safe. Have them refresh the panel to see products appear so the loop feels real.
- 6. Deleting a Table. Make them practise DROP on my_temp_table only, never a real table. Say plainly that there is no undo. This is the moment to set the habit of caution before UPDATE and DELETE arrive.
- 8. Selecting Records. Run the same table three ways: plain, filtered with WHERE, sorted with ORDER BY. Ask what changed each time. Get them predicting the row count before they run each query.
- 9. Updating Records. Demonstrate the danger deliberately: point at the query with no WHERE and ask what it would do to every row. Then insist WHERE goes in first, before the SET value, as a working habit.
- 11. Simple Joins. Run the plain users select first, then LEFT JOIN, then INNER JOIN, side by side. Ask them to spot the NULL and explain why LEFT keeps unmatched users while INNER drops them. This is the key conceptual step.
- 12. Challenge. Let them attempt the orders table before revealing the solution. Circulate and check FOREIGN KEY syntax and that product_id values actually match real product ids, or the join returns nothing.

WHAT TO WATCH FOR

- Students run a whole tab of statements at once and only the first or last executes, so they think a command failed when it never ran. Show that you highlight the single statement you want, then Run Query. Make selecting-before-running a fixed routine from the first CREATE onward.
- Leaving out WHERE on UPDATE or DELETE, expecting it to change one row when it changes every row. Have them SELECT with the same condition first to see exactly which rows match, then reuse that WHERE in the UPDATE or DELETE. No WHERE means all rows, every time.
- Expecting an INNER JOIN to show every user, then panicking when users with no hobby vanish. Compare INNER and LEFT results directly. INNER keeps only matching pairs, LEFT keeps all users and fills missing hobbies with NULL. Choice depends on what you want to see.

DIFFERENTIATION

- Emerging: If SQLite Explorer will not open the database, pair them with a working machine or hand over the ready-made users.db copy so they run queries rather than fight setup. Give them the exact CREATE and INSERT statements to copy and run first, so they get one visible result before writing anything themselves.
- Developing: Ask them to predict each query's result and row count before running, then check. This builds reasoning about what SQL does rather than just typing it.
- Proficient: Push them to add a FOREIGN KEY constraint in the orders table and explain what it enforces, aligning with Higher Level relational database expectations. Have them write a three-table join or a query using ORDER BY with a WHERE filter, ready as evidence for a database-focused Applied Learning Task.