SQLite Setup & Tools

Command Line & DB Browser for SQLite

Session Objectives

✅ Understand what **SQLite** is and when to use it
✅ Install and use the **SQLite command-line shell**
✅ Install and use **DB Browser for SQLite**
✅ Run your first SQL queries

What is SQLite?

📌 A **self-contained**, **serverless** SQL database engine
📌 The entire database is stored in a **single file** (e.g. `library.db`)
📌 No installation of a server required – perfect for learning and prototyping
📌 Used in mobile apps, browsers, embedded systems, and education

SQLite vs PostgreSQL

📌 PostgreSQL → client/server, requires a running service
📌 SQLite → file-based, no server, no configuration
📌 Both speak SQL – minor syntax differences (as seen in previous lessons)
📌 For this course: **SQLite is our practice environment**

Two Ways to Use SQLite

✅ **Command-line shell** (`sqlite3`) – lightweight, always available

✅ **DB Browser for SQLite** – graphical interface, easier to explore data

Part 1

SQLite Command Line

Installing SQLite CLI

macOS – pre-installed, open Terminal and type:

```
sqlite3 --version
```

Linux (Debian/Ubuntu)

```
sudo apt install sqlite3
```

Windows – download from sqlite.org/download.html
→ Download **sqlite-tools-win-x64.zip**, extract, add folder to PATH

Opening a Database

Create or open a database file:

```
sqlite3 library.db
```

📌 If the file does not exist, SQLite creates it automatically
📌 You will see the `sqlite>` prompt – you are now inside the shell

Essential Shell Commands

These are SQLite-specific commands (start with .):

```
.help               → list all available commands
.tables             → list all tables in the database
.schema             → show CREATE TABLE statements
.quit               → exit the shell
.headers on         → show column names in query results
.mode column        → align output in columns
```

📌 Shell commands do **not** end with ;

Enable Foreign Keys

⚠️ Foreign key enforcement is **off by default** in SQLite
Run this at the start of every session:

```
PRAGMA foreign_keys = ON;
```

📌 This must be repeated every time you open the database
📌 Tip: put it at the top of your `.sql` script files

Running SQL Statements

Type SQL directly at the prompt:

```
CREATE TABLE Users (
    id INTEGER PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL
);
```

📌 Every SQL statement must end with **;**

📌 You can press Enter mid-statement – SQLite waits for ;

Running a SQL File

Save your SQL to a file (e.g. `schema.sql`) and run it:

```
sqlite3 library.db < schema.sql
```

Or from inside the shell:

```
.read schema.sql
```

📌 Useful for running the full schema creation script at once

Inspecting the Database

```
-- List all tables
.tables

-- Show structure of a specific table
.schema Users

-- Quick query
SELECT * FROM Users;
```

Exiting the Shell

`.quit`

Or press `Ctrl + D`

📌 SQLite saves data to the file automatically – no `COMMIT` required for DDL

Part 2

DB Browser for SQLite

What is DB Browser for SQLite?

📌 A free, open-source **graphical interface** for SQLite
📌 Lets you create databases, design tables, browse data, and run queries – all visually
📌 Available on **Windows, macOS, Linux**
📌 Download: sqlitebrowser.org

Installation

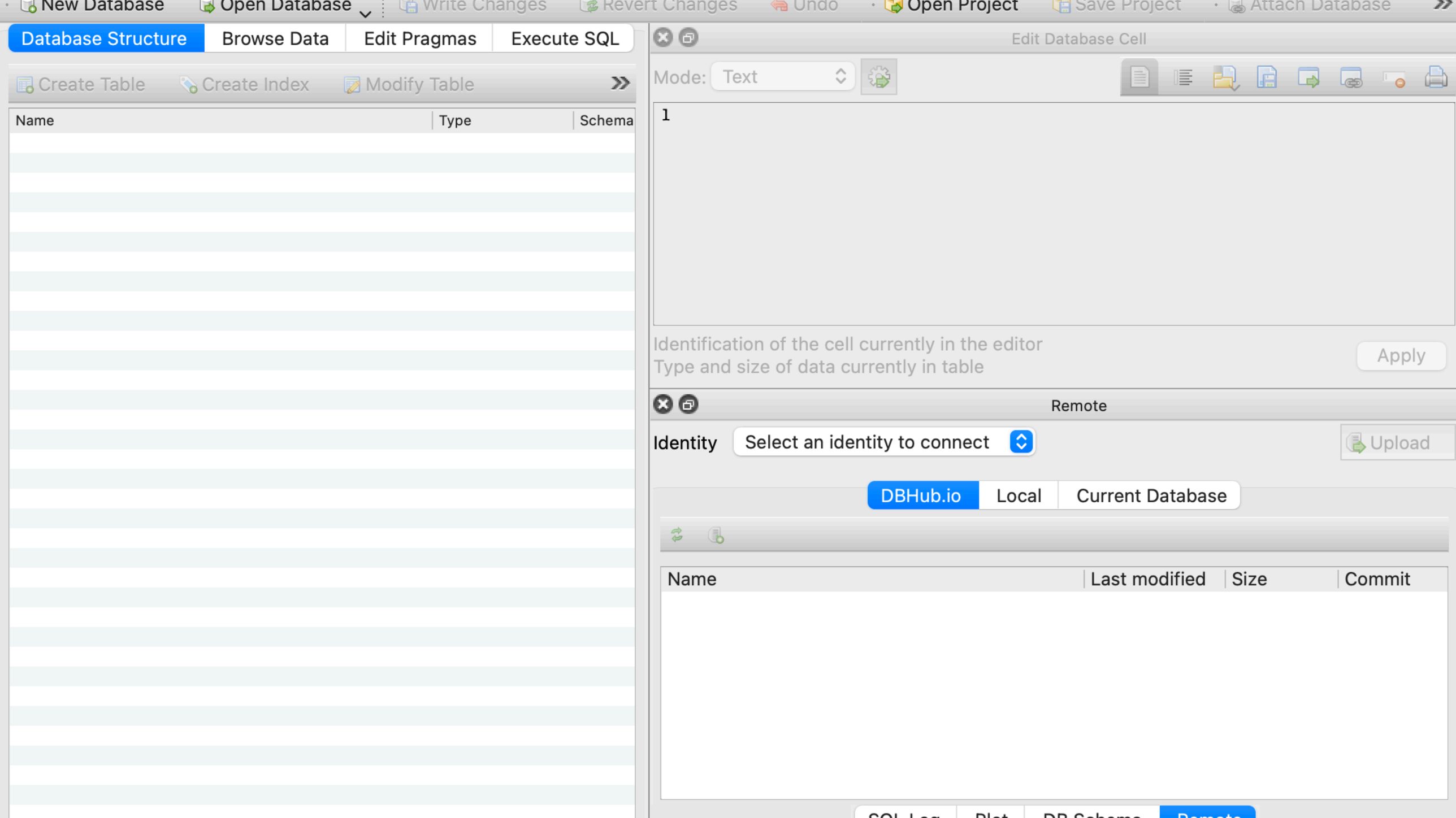
macOS (via Homebrew):

```
brew install --cask db-browser-for-sqlite
```

Windows – download the `.msi` installer from sqlitebrowser.org/dl

Linux (Ubuntu/Debian):

```
sudo apt install sqlitebrowser
```


New Database
Open Database
Write Changes
Revert Changes
Undo
Open Project
Save Project
Attach Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Create Table
Create Index
Modify Table
Name
Type
Schema
Edit Database Cell
Mode:
Text
1
Identification of the cell currently in the editor
Type and size of data currently in table
Apply
Remote
Identity
Select an identity to connect
Upload
DBHub.io
Local
Current Database
Name
Last modified
Size
Commit

Interface Overview

The main window has **4 tabs**:

✅ **Database Structure** → view tables, indexes, views
✅ **Browse Data** → view and edit rows directly
✅ **Edit Pragmas** → configure SQLite settings (including foreign keys)
✅ **Execute SQL** → write and run SQL queries

Creating a New Database

1. Click **File → New Database**
2. Choose a location and filename (e.g. `library.db`)
3. DB Browser immediately asks you to create the first table
   → You can skip this and use the SQL tab instead

📌 The database file is created on disk immediately

Enabling Foreign Keys (GUI)

1. Go to the **Edit Pragmas** tab
2. Find **Foreign Keys** in the list
3. Set it to **ON**
4. Click **Write Changes**

📌 This is equivalent to running `PRAGMA foreign_keys = ON;`

Creating Tables - SQL Tab

1. Click the **Execute SQL** tab
2. Type your `CREATE TABLE` statement
3. Click the ▶ **Run** button (or press `F5`)

```
CREATE TABLE Users (
    id INTEGER PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    user_type VARCHAR(20)
);
```

1. Click **Write Changes** to save to disk

Creating Tables – Graphical Wizard

1. Click **Database Structure** tab
2. Click **Create Table**
3. Enter the table name
4. Add columns: name, type, constraints (PK, NOT NULL, UNIQUE…)
5. Click **OK** → the table is created

Inserting Data

Via Browse Data tab:

1. Select a table from the dropdown
2. Click **New Record**
3. Fill in the fields
4. Click **Write Changes**

Via Execute SQL tab:

```
INSERT INTO Users (name, email, user_type)
VALUES ('Mario Rossi', 'mario@uni.edu', 'student');
```

Browsing and Editing Data

1. Go to **Browse Data** tab
2. Select the table from the dropdown
3. All rows are displayed in a grid
4. Click any cell to edit it directly
5. Click **Write Changes** to save

⚠️ Changes are **not saved automatically** – always click **Write Changes**

Running Queries

1. Click the **Execute SQL** tab
2. Type your query in the top panel
3. Press **F5** or click ▶
4. Results appear in the bottom panel

```
SELECT name, email
FROM Users
WHERE user_type = 'student'
ORDER BY name;
```

📌 You can have multiple queries open in separate tabs

Saving and Loading SQL Scripts

Save a query to file:

File → Save SQL file → `my_query.sql`

Load a SQL script:

File → Import → SQL file

→ The script runs automatically

📌 Useful for sharing exercises or loading the full course schema

Exporting Data

Export table to CSV:

1. **Browse Data** tab → select table
2. File → **Export** → **Table(s) as CSV**

Export query result:

1. Run a query in **Execute SQL**
2. Right-click on the results → **Save as CSV**

Quick Reference – CLI vs GUI

Action	CLI (`sqlite3`)	DB Browser
Open database	`sqlite3 file.db`	File → Open
Create table	SQL in prompt	SQL tab or wizard
Insert data	SQL `INSERT`	Browse Data tab
Run a query	SQL in prompt	Execute SQL tab
Save changes	Automatic	**Write Changes** button
Enable FK	`PRAGMA foreign_keys = ON`	Edit Pragmas tab

Session Summary

✅ SQLite stores the entire database in a **single file**
✅ CLI: open with `sqlite3 file.db`, use `.commands` for shell, `;` for SQL
✅ Always run `PRAGMA foreign_keys = ON;` at the start of each session
✅ DB Browser: use **Execute SQL** for queries, **Write Changes** to save
✅ Both tools work with the same `.db` file – you can switch freely