Lesson 1: Introduction to Databases & SQLite Setup

Duration: 20 minutes
Deliverable: `lesson1_setup.sql`

🎯 Learning Objectives

By the end of this lesson, you will be able to:

- Understand what databases are and why we use them
- Create your first SQLite database
- Design and create a table with appropriate data types
- Insert data into a table
- Use primary keys to uniquely identify records

📚 What is a Database?

A **database** is an organised collection of data that can be easily accessed, managed, and updated. Think of it like a digital filing cabinet where information is stored in a structured way.

Why Use Databases?

- **Organisation:** Keep related data together in a logical structure
- **Efficiency:** Quickly find and retrieve specific information
- **Consistency:** Ensure data follows specific rules and formats
- **Scalability:** Handle small or large amounts of data
- **Multi-user Access:** Multiple people can work with the same data

Types of Databases

1. **Relational Databases** (what we're learning!)
2. Data stored in tables with rows and columns
3. Tables can be related to each other
4. Examples: SQLite, MySQL, PostgreSQL

5. **Non-Relational Databases** (NoSQL)

6. Data stored in documents, key-value pairs, or graphs
7. More flexible structure
8. Examples: MongoDB, Redis

What is SQLite?

SQLite is a lightweight relational database that:

- Stores data in a single file (`.db`)
- Doesn't require a separate server
- Is perfect for learning and small-to-medium applications
- Is used in mobile apps, browsers, and embedded systems

🛠️ Part 1: Creating Your Database (3 minutes)

Step 1: Create Your SQL File

1. In the **Explorer** panel (left side of VSCode), navigate to the `lessons/` folder
2. Right-click on the `lessons/` folder
3. Select **New File**
4. Name it: `lesson1_setup.sql`
5. The file will open in the editor

Step 2: Add a Header Comment

In your `lesson1_setup.sql` file, type the following:

```
-- Lesson 1: Introduction to Databases & SQLite Setup
-- Student Name: [Your Name]
-- Date: [Today's Date]
--
-- This script creates the Star Wars characters database
```

What are comments?
Comments start with `--` and are ignored by SQL. They're for humans to read!

🏗️ Part 2: Creating Your First Table (10 minutes)

Understanding Tables

A **table** is like a spreadsheet with:

- **Columns:** Define what type of information is stored (like "Name" or "Age")
- **Rows:** Individual records (like one person's information)

The Characters Table

We'll create a table to store Star Wars characters with these columns:

Column Name	Data Type	Purpose
`id`	INTEGER	Unique identifier for each character
`name`	TEXT	Character's name
`species`	TEXT	What species they are
`homeworld`	TEXT	Their home planet

Step 3: Write the CREATE TABLE Statement

Add this code to your `lesson1_setup.sql` file:

```
-- Create the characters table
CREATE TABLE IF NOT EXISTS characters (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    species TEXT,
    homeworld TEXT
);
```

🔍 Understanding the Code

Let's break down what each part does:

- `CREATE TABLE` - Command to create a new table
- `IF NOT EXISTS` - Only create if table doesn't already exist (prevents errors)
- `characters` - Name of our table
- `id INTEGER PRIMARY KEY AUTOINCREMENT` - Creates an auto-numbering ID column
- `INTEGER` - Number data type
- `PRIMARY KEY` - Uniquely identifies each row
- `AUTOINCREMENT` - Automatically assigns the next number
- `name TEXT NOT NULL` - Text column that must have a value
- `TEXT` - Text data type
- `NOT NULL` - This field cannot be empty

- `species TEXT` - Text column (optional)
- `homeworld TEXT` - Text column (optional)

Step 4: Execute the CREATE TABLE Statement

1. **Select all the SQL code** you just wrote (click and drag, or Ctrl+A)
2. **Right-click** on the selected code
3. Choose **"Run Selected Query"** (or use the keyboard shortcut shown)
4. Check the **Output** panel at the bottom for confirmation

Expected Output: You should see a message confirming the table was created.

📝 Part 3: Inserting Data (7 minutes)

The INSERT Statement

Now let's add characters to our table! The `INSERT INTO` statement adds new rows.

Step 5: Add Character Data

Add this code to your `lesson1_setup.sql` file (below your CREATE TABLE statement):

```
-- Insert Star Wars characters
INSERT INTO characters (name, species, homeworld) VALUES
    ('Luke Skywalker', 'Human', 'Tatooine'),
    ('Leia Organa', 'Human', 'Alderaan'),
    ('Han Solo', 'Human', 'Corellia'),
    ('Chewbacca', 'Wookiee', 'Kashyyyk'),
    ('Obi-Wan Kenobi', 'Human', 'Stewjon'),
    ('Darth Vader', 'Human', 'Tatooine'),
    ('Yoda', 'Yoda''s species', 'Unknown'),
    ('R2-D2', 'Droid', 'Naboo');
```

🔍 Understanding the INSERT Statement

- `INSERT INTO characters` - Which table to insert into
- `(name, species, homeworld)` - Which columns we're filling (id is automatic!)
- `VALUES` - Keyword before the data
- `('Luke Skywalker', 'Human', 'Tatooine')` - One row of data
- Note: Text values are in **single quotes** `'like this'`
- Multiple rows separated by commas

Step 6: Execute the INSERT Statement

1. **Select only the INSERT code** you just added
2. **Right-click → "Run Selected Query"**
3. Check the Output panel for confirmation

Expected Output: "8 rows inserted" or similar message.

Step 7: Verify Your Data

Let's check that the data was inserted correctly. Add this query:

```
-- View all characters
SELECT * FROM characters;
```

Execute this query to see all your characters displayed in a table format!

🎓 Practice Exercise

Now it's your turn! Add **3 more characters** of your choice.

Step 8: Add Your Own Characters

Add this code and fill in your chosen characters:

```
-- Additional characters (Practice Exercise)
INSERT INTO characters (name, species, homeworld) VALUES
    ('Character Name 1', 'Species 1', 'Homeworld 1'),
    ('Character Name 2', 'Species 2', 'Homeworld 2'),
    ('Character Name 3', 'Species 3', 'Homeworld 3');
```

Suggestions:

- Padmé Amidala (Human, Naboo)
- Mace Windu (Human, Haruun Kal)
- Ahsoka Tano (Togruta, Shili)
- Boba Fett (Human, Kamino)
- Jabba the Hutt (Hutt, Nal Hutta)

Execute your INSERT statement, then run `SELECT * FROM characters;` again to see all 11 characters!

🐛 Common Errors & Troubleshooting

Error: "table characters already exists"

Problem: You've run the CREATE TABLE statement multiple times.

Solution: Either:

- Use `CREATE TABLE IF NOT EXISTS` (already in our code!)
- Delete the table first: `DROP TABLE IF EXISTS characters;`

Error: "syntax error near..."

Problem: Missing comma, quote, or parenthesis.

Solution:

- Check each line carefully
- Make sure text values are in single quotes: `'like this'`
- Ensure commas between each row of data
- Match opening and closing parentheses

Error: "NOT NULL constraint failed"

Problem: Trying to insert a row without a value for `name`.

Solution: The `name` column is required. Make sure every INSERT includes a name value.

VSCode Plugin Not Responding

Problem: Can't execute queries or see the database.

Solution:

1. Check the file is saved (Ctrl+S)
2. Look for the SQLite3 Editor icon in the sidebar
3. Close and reopen the `.sql` file
4. Check the Output panel for error messages

Can't See My Database File

Problem: `database/starwars.db` doesn't appear in the file explorer.

Solution:

- The database file is created when you run your first query
- Refresh the file explorer (right-click → Refresh)
- Check you're in the correct Codespace

✅ Checkpoint: What You've Learnt

Before moving on, make sure you can:

- ✅ Explain what a database is and why it's useful
- ✅ Create a table with appropriate columns and data types
- ✅ Understand what a PRIMARY KEY is
- ✅ Insert single and multiple rows of data
- ✅ Use SELECT to view your data
- ✅ Write comments in SQL using `--`

🎯 Challenge Problem (Optional)

Task: Create a second table called `droids` with these columns:

- `id` (INTEGER, PRIMARY KEY, AUTOINCREMENT)
- `name` (TEXT, NOT NULL)
- `model` (TEXT)
- `function` (TEXT)

Insert at least 3 droids:

- R2-D2 (Astromech, Repair & Navigation)
- C-3PO (Protocol Droid, Translation)
- BB-8 (Astromech, Reconnaissance)

Write your solution at the end of your `lesson1_setup.sql` file.

💾 Save Your Work with Git

Now that you've completed the lesson, save your work to your repository.

Step 1: Open the Terminal

1. Click **Terminal** in the top menu
2. Select **New Terminal**
3. A terminal panel will open at the bottom

Step 2: Check What's Changed

```
git status
```

You should see `lesson1_setup.sql` listed as a new or modified file.

Step 3: Stage Your Changes

```
git add lessons/lesson1_setup.sql
```

This tells Git you want to save this file.

Step 4: Commit Your Changes

```
git commit -m "Completed Lesson 1: Created characters table and inserted data"
```

This creates a save point with a description.

Step 5: Push to GitHub

```
git push
```

This uploads your changes to your GitHub repository.

Expected Output: You should see confirmation that your changes were pushed successfully.

📖 Key SQL Commands Learnt

Command	Purpose	Example
`CREATE TABLE`	Create a new table	`CREATE TABLE characters (...)`
`INSERT INTO`	Add new rows to a table	`INSERT INTO characters VALUES (...)`
`SELECT`	Retrieve data from a table	`SELECT * FROM characters;`
`PRIMARY KEY`	Uniquely identify each row	`id INTEGER PRIMARY KEY`
`NOT NULL`	Column must have a value	`name TEXT NOT NULL`

🎉 Well Done!

You've created your first database and table! In the next lesson, you'll learn how to query this data using SELECT statements with filtering conditions.

Ready to continue? Move on to `lesson2_instructions.md`

Need Help?

- Review the troubleshooting section above
- Check your syntax carefully
- Ask your instructor
- Compare with the solution file (after attempting yourself!)