Assignment 7 – Flask + SQLite Bookstore

Overview

In this assignment, you will convert your previous Bookstore project from using Python lists to using a SQLite relational database. You will implement a search feature that uses SQL queries to return matching books and a book detail page that loads details from your database.

This assignment builds on your understanding of:

- Connecting Python to a database
- Executing SQL queries from Flask
- Rendering dynamic pages using templates
- Organizing a back-end and front-end within the same project
- Using SQLite inside EdStem

You will reuse your existing templates, but the data will now come entirely from a database instead of Python lists.

Deliverables

Submit your EdStem workspace containing:

- `app.py`
- `bookstore.db`
- `templates/` folder (HTML templates)
- `static/` folder (images and CSS)

Your Flask site must run without errors in the EdStem environment.

1. Create the SQLite Database

Open the EdStem terminal and create a SQLite database file:

```
sqlite3 bookstore.db
```

Inside the SQLite shell, create the required tables:

```
CREATE TABLE categories (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL
);

CREATE TABLE books (
    id INTEGER PRIMARY KEY,
    categoryId INTEGER,
    title TEXT NOT NULL,
    author TEXT,
    isbn TEXT,
    price REAL,
    image TEXT,
    readNow INTEGER,
    FOREIGN KEY (categoryId) REFERENCES categories(id)
);
```

Add your data by inserting at least four categories and sixteen books.

You may write the SQL manually or write a small Python script that inserts your previous book lists into the database.

2. Update Your Flask App to Use SQLite

Use this helper function to connect to the database:

```
def get_db_connection():
    conn = sqlite3.connect('bookstore.db')
    conn.row_factory = sqlite3.Row
    return conn
```

Replace your hard-coded lists

Remove the Python lists storing categories and books from your previous assignment. All data should now come from SQLite.

---

3. Load Categories From the Database

Create a reusable function:

```
def get_categories():
    conn = get_db_connection()
    categories = conn.execute("SELECT * FROM categories").fetchall()
    conn.close()
    return categories
```

---

4. Home Page Route

Use your template exactly as before, but pass in database-driven categories:

```
@app.route('/')
def home():
    return render_template("index.html", categories=get_categories())
```

---

5. Category Page Route

Use a SQL query instead of filtering Python lists:

```
@app.route('/category')
def category():
    category_id = request.args.get("categoryId", type=int)
    conn = get_db_connection()
    books = conn.execute(
        "SELECT * FROM books WHERE categoryId = ?",
        (category_id,)
    ).fetchall()
    conn.close()

    return render_template(
        "category.html",
        selectedCategory=category_id,
        categories=get_categories(),
        books=books
    )
```

Your existing `category.html` template should work without modification.

---

6. Search Feature

Create a dedicated search route that performs a SQL query:

```
@app.route('/search', methods=['POST'])
def search():
    term = request.form.get("search", "").strip()

    conn = get_db_connection()
    books = conn.execute(
        "SELECT * FROM books WHERE lower(title) LIKE lower(?)",
        (f"%{term}%",)
    ).fetchall()
    conn.close()

    return render_template(
        "search.html",
        categories=get_categories(),
        books=books,
        term=term
    )
```

Search Results Template

Create `search.html`. Display “no results found” if needed:

```
{% if books|length == 0 %}
<p>No results found for "{{ term }}".</p>
{% else %}
    <!-- loop through books just like category page -->
{% endif %}
```

---

7. Book Detail Page

implementing individual book pages:

```
@app.route('/book/<int:book_id>')
def book_detail(book_id):
    conn = get_db_connection()
    book = conn.execute("""
        SELECT books.*, categories.name AS categoryName
        FROM books
        JOIN categories ON categories.id = books.categoryId
        WHERE books.id = ?
    """, (book_id,)).fetchone()
    conn.close()

    return render_template("book_detail.html", book=book)
```

Templates may reuse your existing layout and styling.

---

8. Final Check

Your submission is complete when:

- All data is coming from SQLite, not Python lists
- Category pages load dynamically
- Search is fully functional and uses an SQL query
- Book details are available
- Templates render correctly
- The project runs cleanly in EdStem