

# SQLite Tutorial: Managing a Library System

Dr. Frank Xu

February 11, 2025

## Introduction

In this tutorial, we will create a simple SQLite database to manage a library system. The database will store information about books, including their title, author, publication year, and genre.

## Step 1: Create a New SQLite Database

Open your terminal or command prompt and create a new SQLite database called `library.db`:

```
1 sqlite3 library.db
```

Once you run the above command, you should see something like this in your terminal:

```
1 user@linux:~$ sqlite3 library.db
2 SQLite version 3.37.2 2022-01-06 13:25:41
3 Enter ".help" for usage hints.
4 sqlite>
```

This indicates that SQLite has successfully created the `library.db` file and opened the SQLite shell. You can now start executing SQL commands directly in this shell.

## Step 2: Create a Table

Now, let's create a table called `books` to store information about the books in our library. The table will have the following columns:

- `id`: A unique identifier for each book (Primary Key).
- `title`: The title of the book.
- `author`: The author of the book.
- `year`: The year the book was published.

- **genre**: The genre of the book.

Run the following SQL command to create the table:

```
1 CREATE TABLE books (
2     id INTEGER PRIMARY KEY AUTOINCREMENT,
3     title TEXT NOT NULL,
4     author TEXT NOT NULL,
5     year INTEGER,
6     genre TEXT
7 );
```

After running the command, you won't see any output, but the table will be created. To verify the table schema, use the '.schema' command:

```
1 sqlite> .schema books
2 CREATE TABLE books (
3     id INTEGER PRIMARY KEY AUTOINCREMENT,
4     title TEXT NOT NULL,
5     author TEXT NOT NULL,
6     year INTEGER,
7     genre TEXT
8 );
```

The '.schema' command displays the exact SQL statement used to create the table, allowing you to confirm its structure.

## Step 3: Insert Data into the Table

Now that we have a table, let's insert some sample data into it.

```
1 INSERT INTO books (title, author, year, genre)
2 VALUES ('To Kill a Mockingbird', 'Harper Lee', 1960, 'Fiction');
3
4 INSERT INTO books (title, author, year, genre)
5 VALUES ('1984', 'George Orwell', 1949, 'Dystopian');
6
7 INSERT INTO books (title, author, year, genre)
8 VALUES ('The Great Gatsby', 'F. Scott Fitzgerald', 1925, 'Classic')
9 ;
10 INSERT INTO books (title, author, year, genre)
11 VALUES ('Pride and Prejudice', 'Jane Austen', 1813, 'Romance');
```

After inserting the data, you won't see any output, but you can verify the data by running a 'SELECT' query:

```
1 sqlite> SELECT * FROM books;
2 1|To Kill a Mockingbird|Harper Lee|1960|Fiction
3 2|1984|George Orwell|1949|Dystopian
4 3|The Great Gatsby|F. Scott Fitzgerald|1925|Classic
5 4|Pride and Prejudice|Jane Austen|1813|Romance
```

## Step 4: Query the Data

Now that we have some data in the table, let's query it.

## 1. Select All Books

To retrieve all the books in the library, run:

```
1 SELECT * FROM books;
```

The result will look like this:

```
1 sqlite> SELECT * FROM books;
2 1|To Kill a Mockingbird|Harper Lee|1960|Fiction
3 2|1984|George Orwell|1949|Dystopian
4 3|The Great Gatsby|F. Scott Fitzgerald|1925|Classic
5 4|Pride and Prejudice|Jane Austen|1813|Romance
```

## 2. Select Books Published After 1900

To find books published after the year 1900, run:

```
1 SELECT * FROM books WHERE year > 1900;
```

The result will look like this:

```
1 sqlite> SELECT * FROM books WHERE year > 1900;
2 1|To Kill a Mockingbird|Harper Lee|1960|Fiction
3 2|1984|George Orwell|1949|Dystopian
4 3|The Great Gatsby|F. Scott Fitzgerald|1925|Classic
```

## 3. Select Books by Genre

To find all books in the "Fiction" genre, run:

```
1 SELECT * FROM books WHERE genre = 'Fiction';
```

The result will look like this:

```
1 sqlite> SELECT * FROM books WHERE genre = 'Fiction';
2 1|To Kill a Mockingbird|Harper Lee|1960|Fiction
```

## Step 5: Update Data

Let's say we want to update the genre of "1984" from "Dystopian" to "Science Fiction". We can do this with the UPDATE statement:

```
1 UPDATE books SET genre = 'Science Fiction' WHERE title = '1984';
```

After running the command, you won't see any output, but you can verify the change by querying the data again:

```
1 sqlite> SELECT * FROM books WHERE title = '1984';
2 2|1984|George Orwell|1949|Science Fiction
```

## Step 6: Delete Data

If we want to remove a book from the library, we can use the `DELETE` statement. For example, to delete "Pride and Prejudice":

```
1 DELETE FROM books WHERE title = 'Pride and Prejudice';
```

After running the command, you won't see any output, but you can verify the deletion by querying the data again:

```
1 sqlite> SELECT * FROM books;
2 1|To Kill a Mockingbird|Harper Lee|1960|Fiction
3 2|1984|George Orwell|1949|Science Fiction
4 3|The Great Gatsby|F. Scott Fitzgerald|1925|Classic
```

## Step 7: Drop the Table (Optional)

If you want to delete the entire table (including all its data), you can use the `DROP TABLE` command:

```
1 DROP TABLE books;
```

After running the command, you won't see any output, but you can verify that the table is gone by listing all tables:

```
1 sqlite> .tables
```

Since the table has been dropped, no tables will be listed.

## Step 8: Exit SQLite

Once you're done working with the database, you can exit the SQLite shell by typing:

```
1 .exit
```

The terminal will return to the normal command prompt:

```
1 sqlite> .exit
2 user@linux:~$
```

## Conclusion

In this tutorial, we created a simple SQLite database to manage a library system. We learned how to create a table, verify its schema, insert data, query data, update records, and delete records. SQLite is a lightweight, serverless database engine that is perfect for small applications, prototyping, and learning SQL.

Feel free to experiment with more complex queries and operations as you become more comfortable with SQLite!