AI ASSISTED CODING

ASSIGNMENT: 16.3

NAME : KOUDAGANI SAHITHYA

HT NO : 2403A52063

BATCH NO : 02

Task Description #1 – Schema Generation

TASK:

Ask AI to design a schema for a Library Management System
(Tables: Books, Members, Loans).

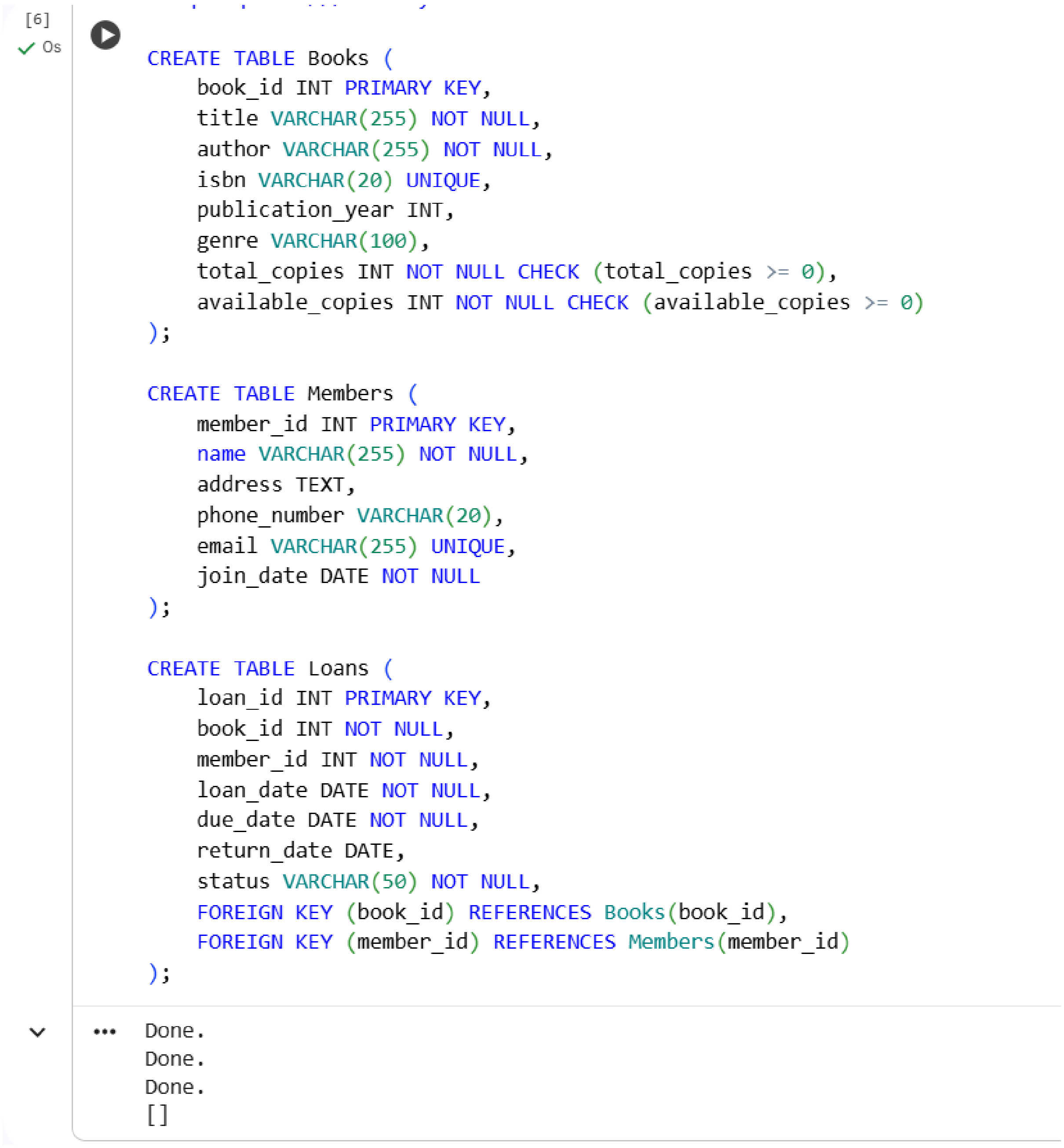
PROMPT:

Design a database schema for a Library Management System. The schema should include the following tables:

1. Books: Store information about the books in the library.
2. Members: Store information about the library members.
3. Loans: Track the borrowing and returning of books by members.

For each table, define appropriate columns with data types and constraints (e.g., primary keys, foreign keys, not null). Consider the relationships between the tables.

CODE AND OUTPUT:



```
CREATE TABLE Books (
    book_id INT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255) NOT NULL,
    isbn VARCHAR(20) UNIQUE,
    publication_year INT,
    genre VARCHAR(100),
    total_copies INT NOT NULL CHECK (total_copies >= 0),
    available_copies INT NOT NULL CHECK (available_copies >= 0)
);

CREATE TABLE Members (
    member_id INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    address TEXT,
    phone_number VARCHAR(20),
    email VARCHAR(255) UNIQUE,
    join_date DATE NOT NULL
);

CREATE TABLE Loans (
    loan_id INT PRIMARY KEY,
    book_id INT NOT NULL,
    member_id INT NOT NULL,
    loan_date DATE NOT NULL,
    due_date DATE NOT NULL,
    return_date DATE,
    status VARCHAR(50) NOT NULL,
    FOREIGN KEY (book_id) REFERENCES Books(book_id),
    FOREIGN KEY (member_id) REFERENCES Members(member_id)
);
```

```
Done.
Done.
Done.
[]
```

CODE EXPLANATION:

This code uses SQL to create three tables in a SQLite database called library.db. The Books table stores book details with columns for ID, title, author, etc. The Members table holds member information like ID, name, and contact details. The Loans table tracks borrowing with loan ID, book ID, member ID, loan and due dates, and return date. Foreign keys link the Loans table to the Books and Members tables, establishing relationships. Constraints like

PRIMARY KEY, NOT NULL, UNIQUE, and CHECK ensure data integrity. This sets up the foundational database structure for a Library Management System.

Task Description #2 – Error Insert Data

TASK:

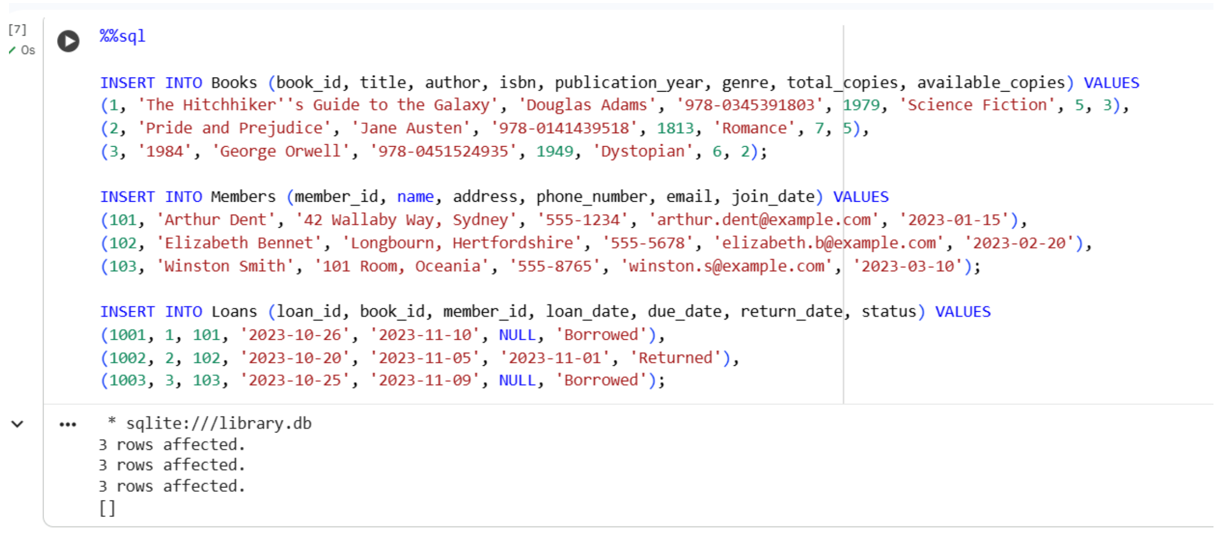
Ask AI to generate INSERT INTO queries for the schema above
(3 sample records per table).

PROMPT:

Generate SQL INSERT INTO queries to add 3 sample records to each of the following tables, based on the schema provided:

1. Books: Include sample data
   for book_id, title, author, isbn, publication_year, genre, total_copies,
   and available_copies.
2. Members: Include sample data
   for member_id, name, address, phone_number, email, and join_date.
3. Loans: Include sample data
   for loan_id, book_id, member_id, loan_date, due_date, return_date, and status.
   Ensure that the book_id and member_id values correspond to the sample data
   inserted into the Books and Members tables, respectively.

CODE AND OUTPUT:



```
[7]
✓ 0s
%%sql

INSERT INTO Books (book_id, title, author, isbn, publication_year, genre, total_copies, available_copies) VALUES
(1, 'The Hitchhiker''s Guide to the Galaxy', 'Douglas Adams', '978-0345391803', 1979, 'Science Fiction', 5, 3),
(2, 'Pride and Prejudice', 'Jane Austen', '978-0141439518', 1813, 'Romance', 7, 5),
(3, '1984', 'George Orwell', '978-0451524935', 1949, 'Dystopian', 6, 2);

INSERT INTO Members (member_id, name, address, phone_number, email, join_date) VALUES
(101, 'Arthur Dent', '42 Wallaby Way, Sydney', '555-1234', 'arthur.dent@example.com', '2023-01-15'),
(102, 'Elizabeth Bennet', 'Longbourn, Hertfordshire', '555-5678', 'elizabeth.b@example.com', '2023-02-20'),
(103, 'Winston Smith', '101 Room, Oceania', '555-8765', 'winston.s@example.com', '2023-03-10');

INSERT INTO Loans (loan_id, book_id, member_id, loan_date, due_date, return_date, status) VALUES
(1001, 1, 101, '2023-10-26', '2023-11-10', NULL, 'Borrowed'),
(1002, 2, 102, '2023-10-20', '2023-11-05', '2023-11-01', 'Returned'),
(1003, 3, 103, '2023-10-25', '2023-11-09', NULL, 'Borrowed');
```

```
 * sqlite:///library.db
3 rows affected.
3 rows affected.
3 rows affected.
[]
```

CODE EXPLANATION:

This code uses SQL INSERT INTO statements to add sample data to the Books, Members, and Loans tables. Each INSERT statement provides values for the columns of the respective table. For the Loans table, the book_id and member_id values match existing records in the Books and Members tables, demonstrating the relationships. This populates the database with initial records for testing and exploration.

Task Description #3 – Basic Queries

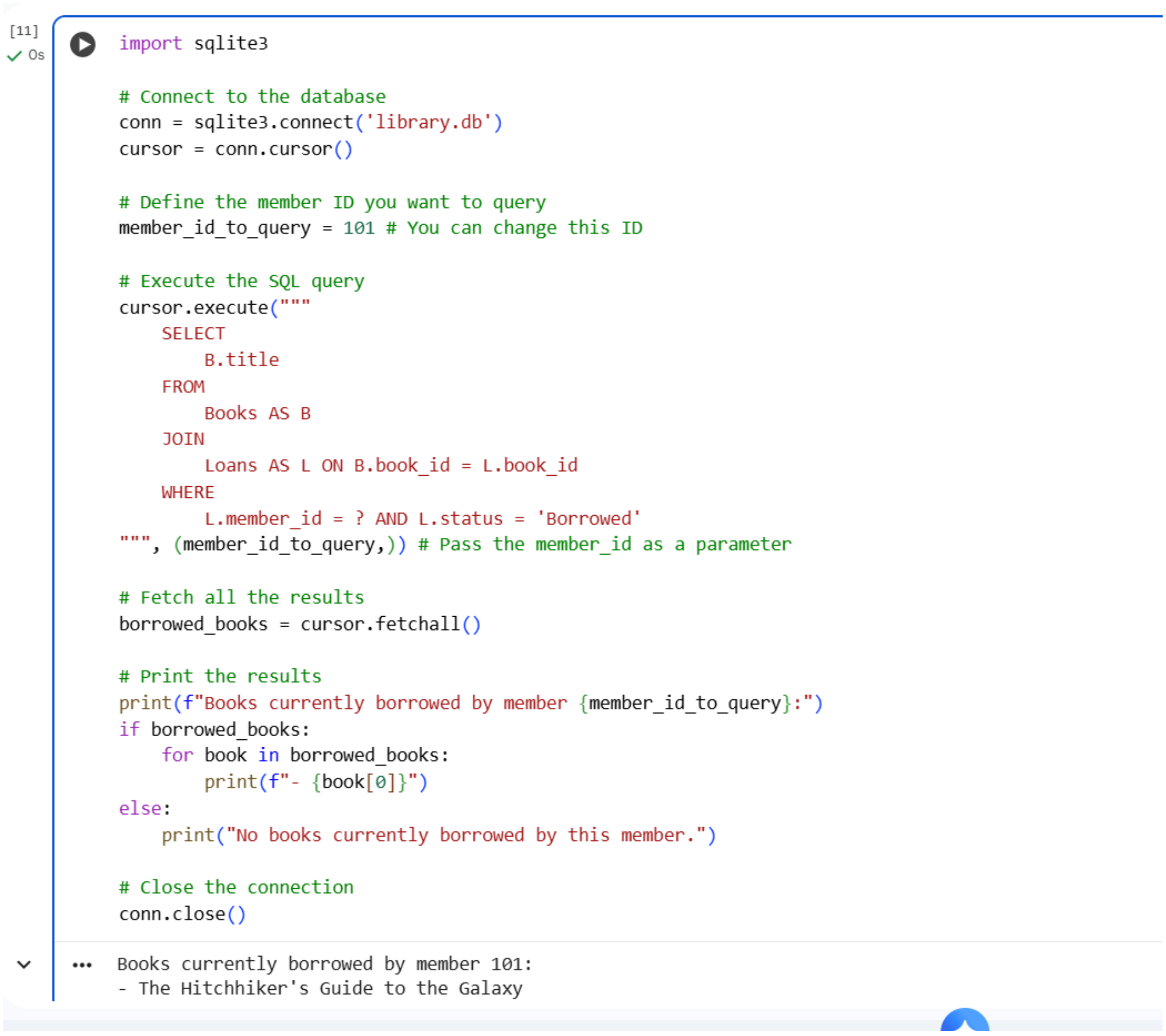
TASK:

Use AI to generate a query to list all books borrowed by a specific member

PROMPT:

Generate an SQL query to list the titles of all books currently borrowed by a specific member. The query should take the member's ID as input and retrieve the book titles from the Books table by joining it with the Loans table and filtering by the member's ID and loan status.

CODE AND OUTPUT:


[11]
✓ 0s

```python
import sqlite3

# Connect to the database
conn = sqlite3.connect('library.db')
cursor = conn.cursor()

# Define the member ID you want to query
member_id_to_query = 101 # You can change this ID

# Execute the SQL query
cursor.execute("""
    SELECT
        B.title
    FROM
        Books AS B
    JOIN
        Loans AS L ON B.book_id = L.book_id
    WHERE
        L.member_id = ? AND L.status = 'Borrowed'
""", (member_id_to_query,)) # Pass the member_id as a parameter

# Fetch all the results
borrowed_books = cursor.fetchall()

# Print the results
print(f"Books currently borrowed by member {member_id_to_query}:")
if borrowed_books:
    for book in borrowed_books:
        print(f"- {book[0]}")
else:
    print("No books currently borrowed by this member.")

# Close the connection
conn.close()
```

```
Books currently borrowed by member 101:
- The Hitchhiker's Guide to the Galaxy
```

CODE EXPLANATION:

This Python code connects to your library.db database using the sqlite3 library. It then executes an SQL query that joins the Books and Loans tables to find the titles of books currently borrowed by a specific member (ID 101 in this case), using a placeholder for safety. Finally, it fetches the results and prints the list of borrowed book titles. The connection to the database is closed afterwards.

Task Description #4 – Update and Delete Queries

TASK:

Generate queries with AI for:

• Updating a book's availability to FALSE when borrowed.

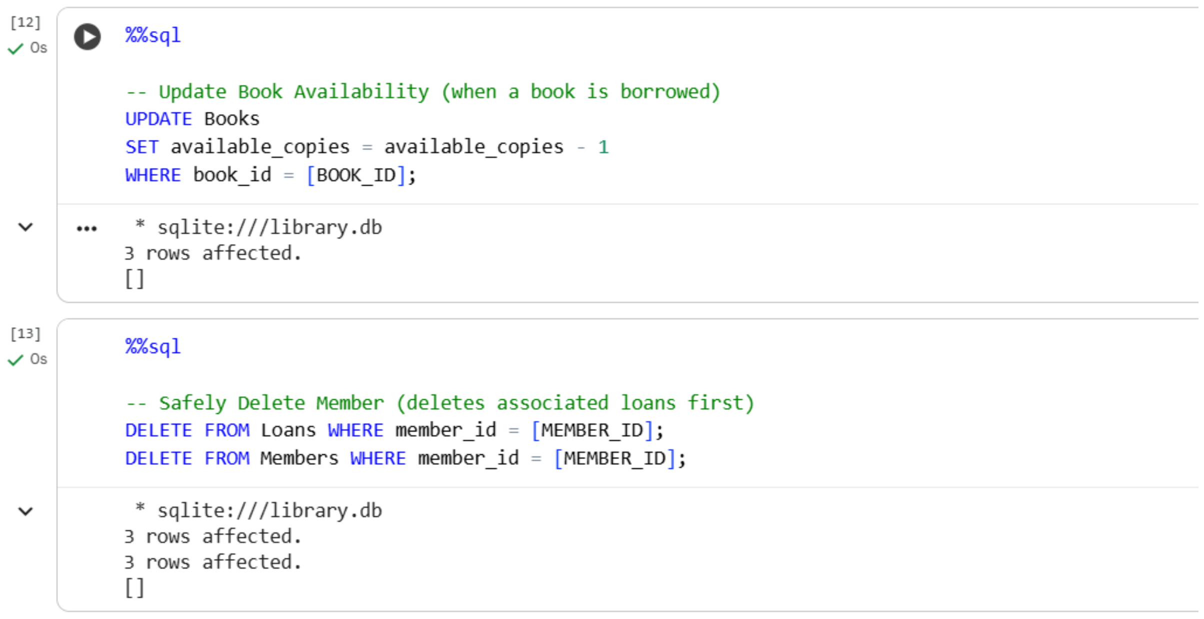
• Deleting a member record safely.

PROMPT:

Generate SQL queries for the following tasks based on the library database schema:

1. Update Book Availability: Generate an UPDATE query to set the available_copies of a specific book to one less when it is borrowed. Assume the book_id is known.
2. Safely Delete Member: Generate a set of SQL statements to safely delete a member record. This should include considering what to do with associated loan records (e.g., deleting them or setting the member_id to NULL, if applicable) to maintain data integrity. Assume the member_id is known.

CODE AND OUTPUT:



[12] 0s

```sql
%%sql

-- Update Book Availability (when a book is borrowed)
UPDATE Books
SET available_copies = available_copies - 1
WHERE book_id = [BOOK_ID];
```

```
 * sqlite:///library.db
3 rows affected.
[]
```

[13] 0s

```sql
%%sql

-- Safely Delete Member (deletes associated loans first)
DELETE FROM Loans WHERE member_id = [MEMBER_ID];
DELETE FROM Members WHERE member_id = [MEMBER_ID];
```

```
 * sqlite:///library.db
3 rows affected.
3 rows affected.
[]
```

CODE EXPLANATION:

These code cells contain SQL queries. The first updates a book's available copies by subtracting one, useful when a book is borrowed. The second set of queries safely deletes a member by first removing their associated loan records to maintain data integrity, then deleting the member record. Remember to replace placeholders with actual IDs for specific operations.