SCHOOLOFCOMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENTOFCOMPUTER SCIENCE ENGINEERING	
ProgramName:B. Tech		**AssignmentType: Lab**	**AcademicYear:**2025-2026
CourseCoordinatorName		Venkataramana Veeramsetty	
Instructor(s)Name		Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)	
CourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
DateandDay of Assignment	Week8 - WednesDay	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	
AssignmentNumber:16.3(Presentassignmentnumber)/**24**(Totalnumberofassignments)			

Q.No.	**Question**	***ExpectedTime to complete***
1	**Lab 16 – Database Design and Queries: Schema Design and SQL Generation** **Lab Objectives** • To practice basic SQL query generation with AI assistance. • To analyze AI-suggested queries for correctness and efficiency.	Week5 - Monday

- To understand how AI can help in documenting and improving database logic.

Learning Outcomes

After completing this lab, students will be able to:

1. Use AI tools to design a simple ER diagram / schema for a given scenario.
2. Generate CREATE TABLE statements using AI.
3. Write and refine basic SQL queries (SELECT, INSERT, UPDATE, DELETE).
4. Validate correctness and efficiency of AI-generated SQL.
5. Compare AI-generated vs manually written queries.

Task Description #1 – Schema Generation

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

SQL Code

```sql
CREATE TABLE Members (
    member_id INT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    join_date DATE
);

CREATE TABLE Books (
    book_id INT PRIMARY KEY,
    title VARCHAR(200),
    author VARCHAR(100),
    available BOOLEAN
);

CREATE TABLE Loans (
    loan_id INT PRIMARY KEY,
    member_id INT,
    book_id INT,
    loan_date DATE,
    return_date DATE,
    FOREIGN KEY (member_id) REFERENCES Members(member_id),
    FOREIGN KEY (book_id) REFERENCES Books(book_id)
);
```

Task Description #2 – Error Insert Data
Task: Ask AI to generate INSERT INTO queries for the schema above (3 sample records per table).

Task Description #3 – Basic Queries
Task: Use AI to generate a query to list all books borrowed by a specific member

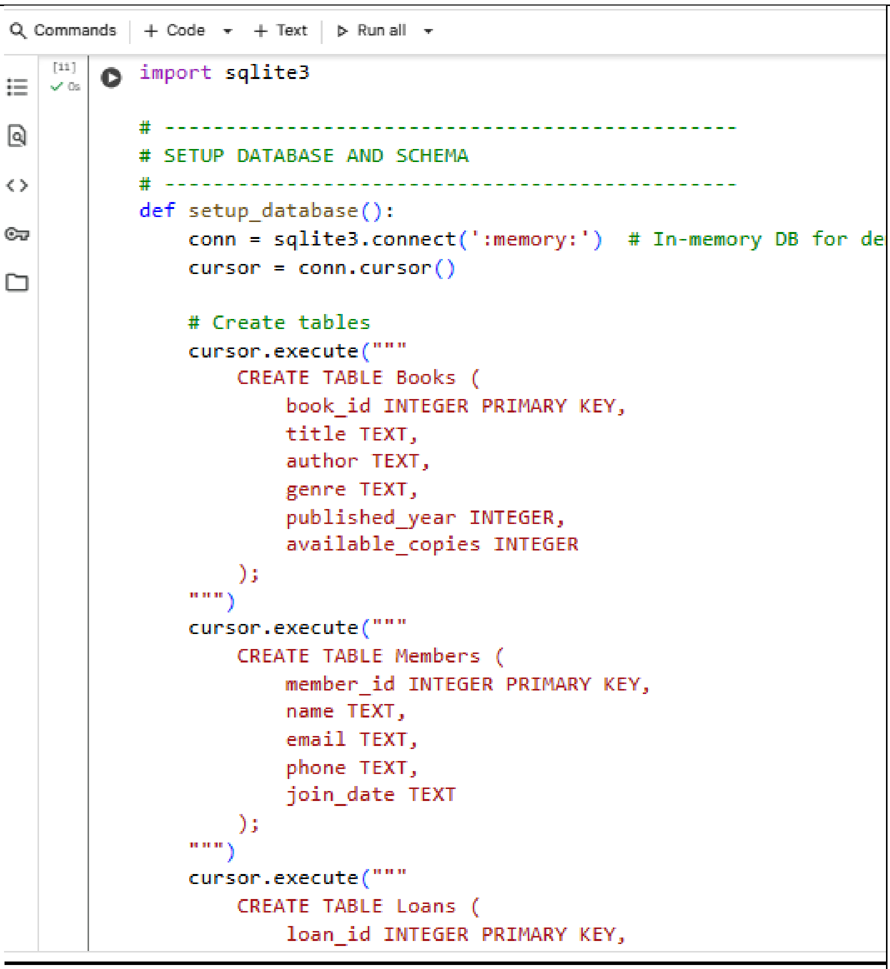
Task Description #4 – Update and Delete Queries
Task: Generate queries with AI for:

- Updating a book’s availability to FALSE when borrowed.
- Deleting a member record safely.

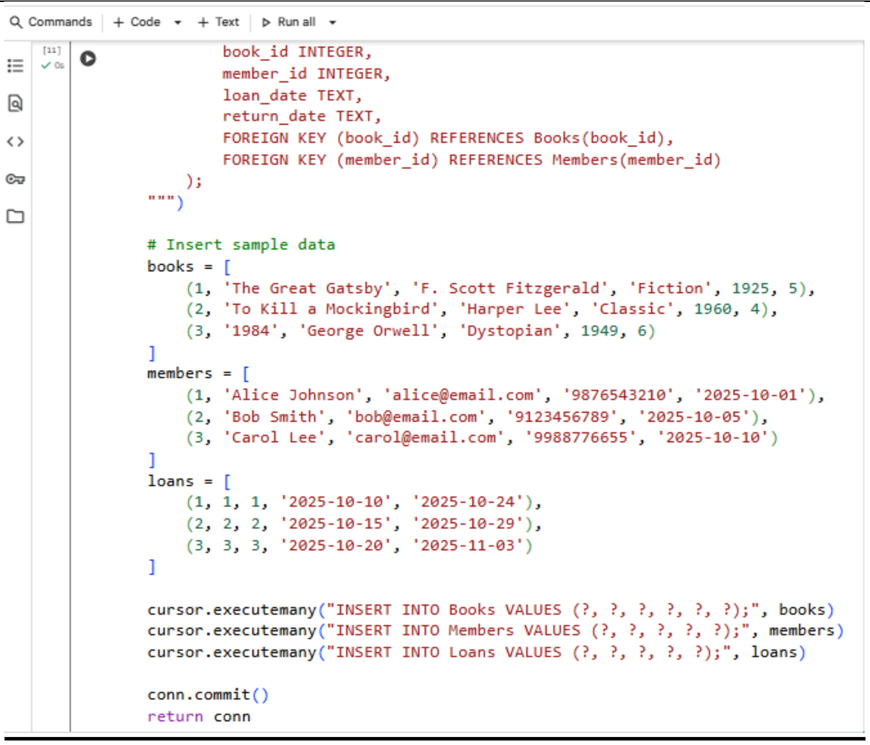
Prompt:

Write a menu-driven Python program using SQLite for a Library Management System with Books, Members, and Loans tables, including sample data, and features to display tables, list books borrowed by a member, borrow a book (update availability), and delete a member safely.

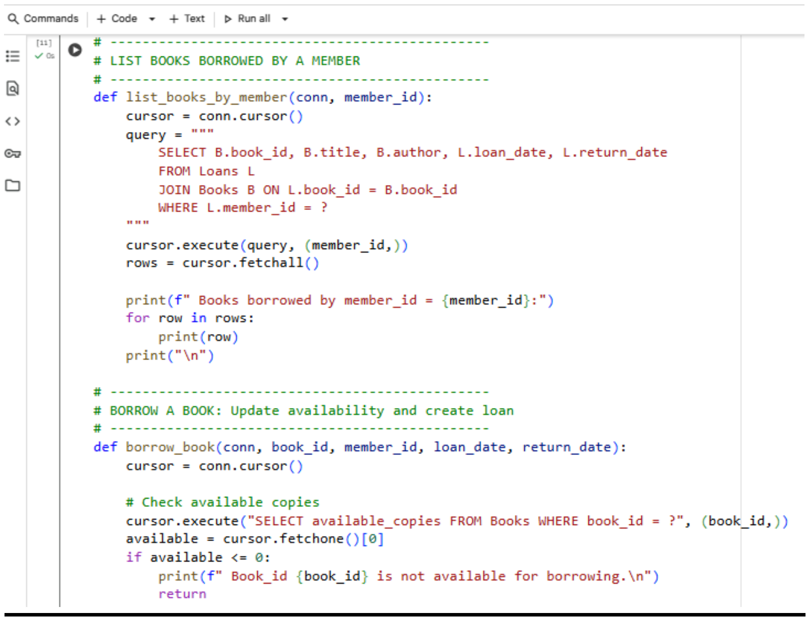
Code:



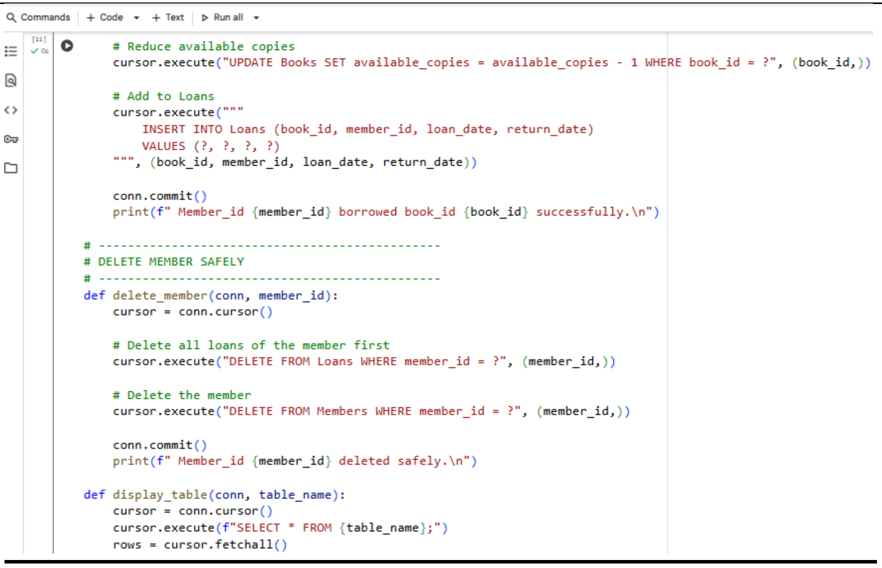
```
import sqlite3

# ------------------------------------------------
# SETUP DATABASE AND SCHEMA
# ------------------------------------------------
def setup_database():
    conn = sqlite3.connect(':memory:')  # In-memory DB for de
    cursor = conn.cursor()

    # Create tables
    cursor.execute("""
        CREATE TABLE Books (
            book_id INTEGER PRIMARY KEY,
            title TEXT,
            author TEXT,
            genre TEXT,
            published_year INTEGER,
            available_copies INTEGER
        );
    """)
    cursor.execute("""
        CREATE TABLE Members (
            member_id INTEGER PRIMARY KEY,
            name TEXT,
            email TEXT,
            phone TEXT,
            join_date TEXT
        );
    """)
    cursor.execute("""
        CREATE TABLE Loans (
            loan_id INTEGER PRIMARY KEY,
```



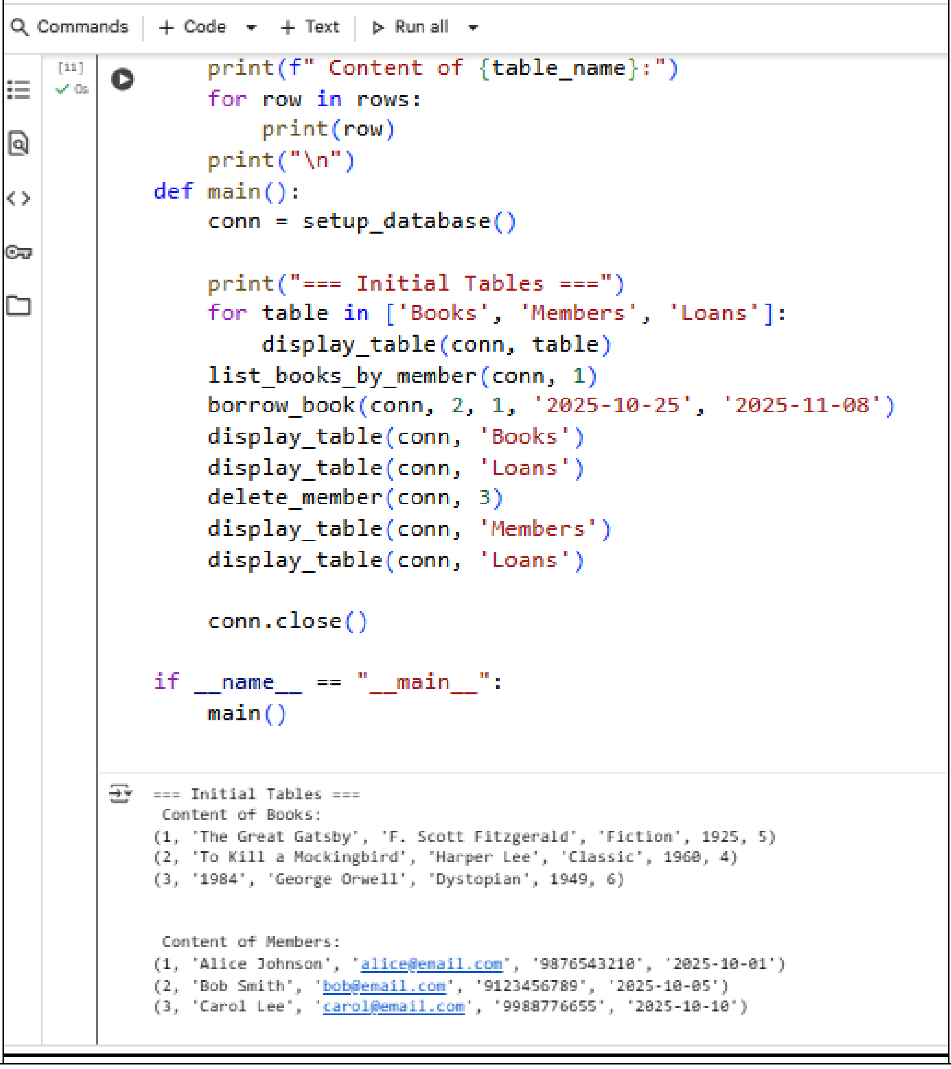
```
            book_id INTEGER,
            member_id INTEGER,
            loan_date TEXT,
            return_date TEXT,
            FOREIGN KEY (book_id) REFERENCES Books(book_id),
            FOREIGN KEY (member_id) REFERENCES Members(member_id)
        );
    """)

    # Insert sample data
    books = [
        (1, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Fiction', 1925, 5),
        (2, 'To Kill a Mockingbird', 'Harper Lee', 'Classic', 1960, 4),
        (3, '1984', 'George Orwell', 'Dystopian', 1949, 6)
    ]
    members = [
        (1, 'Alice Johnson', 'alice@email.com', '9876543210', '2025-10-01'),
        (2, 'Bob Smith', 'bob@email.com', '9123456789', '2025-10-05'),
        (3, 'Carol Lee', 'carol@email.com', '9988776655', '2025-10-10')
    ]
    loans = [
        (1, 1, 1, '2025-10-10', '2025-10-24'),
        (2, 2, 2, '2025-10-15', '2025-10-29'),
        (3, 3, 3, '2025-10-20', '2025-11-03')
    ]

    cursor.executemany("INSERT INTO Books VALUES (?, ?, ?, ?, ?, ?);", books)
    cursor.executemany("INSERT INTO Members VALUES (?, ?, ?, ?, ?);", members)
    cursor.executemany("INSERT INTO Loans VALUES (?, ?, ?, ?, ?);", loans)

    conn.commit()
    return conn
```



```
# ---------------------------------------------
# LIST BOOKS BORROWED BY A MEMBER
# ---------------------------------------------
def list_books_by_member(conn, member_id):
    cursor = conn.cursor()
    query = """
        SELECT B.book_id, B.title, B.author, L.loan_date, L.return_date
        FROM Loans L
        JOIN Books B ON L.book_id = B.book_id
        WHERE L.member_id = ?
    """
    cursor.execute(query, (member_id,))
    rows = cursor.fetchall()

    print(f" Books borrowed by member_id = {member_id}:")
    for row in rows:
        print(row)
    print("\n")

# ---------------------------------------------
# BORROW A BOOK: Update availability and create loan
# ---------------------------------------------
def borrow_book(conn, book_id, member_id, loan_date, return_date):
    cursor = conn.cursor()

    # Check available copies
    cursor.execute("SELECT available_copies FROM Books WHERE book_id = ?", (book_id,))
    available = cursor.fetchone()[0]
    if available <= 0:
        print(f" Book_id {book_id} is not available for borrowing.\n")
        return
```

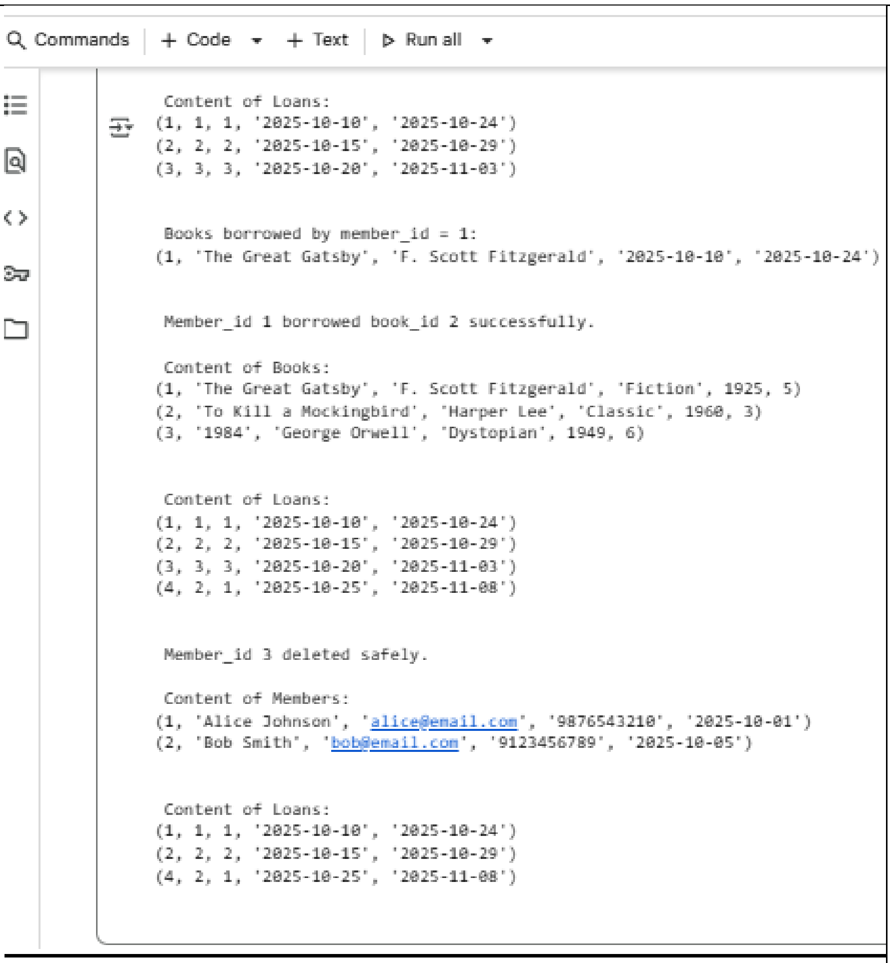


```python
    # Reduce available copies
    cursor.execute("UPDATE Books SET available_copies = available_copies - 1 WHERE book_id = ?", (book_id,))

    # Add to Loans
    cursor.execute("""
        INSERT INTO Loans (book_id, member_id, loan_date, return_date)
        VALUES (?, ?, ?, ?)
    """, (book_id, member_id, loan_date, return_date))

    conn.commit()
    print(f" Member_id {member_id} borrowed book_id {book_id} successfully.\n")

# -----------------------------------------------
# DELETE MEMBER SAFELY
# -----------------------------------------------
def delete_member(conn, member_id):
    cursor = conn.cursor()

    # Delete all loans of the member first
    cursor.execute("DELETE FROM Loans WHERE member_id = ?", (member_id,))

    # Delete the member
    cursor.execute("DELETE FROM Members WHERE member_id = ?", (member_id,))

    conn.commit()
    print(f" Member_id {member_id} deleted safely.\n")

def display_table(conn, table_name):
    cursor = conn.cursor()
    cursor.execute(f"SELECT * FROM {table_name};")
    rows = cursor.fetchall()
```



```python
    print(f" Content of {table_name}:")
    for row in rows:
        print(row)
    print("\n")
def main():
    conn = setup_database()

    print("=== Initial Tables ===")
    for table in ['Books', 'Members', 'Loans']:
        display_table(conn, table)
    list_books_by_member(conn, 1)
    borrow_book(conn, 2, 1, '2025-10-25', '2025-11-08')
    display_table(conn, 'Books')
    display_table(conn, 'Loans')
    delete_member(conn, 3)
    display_table(conn, 'Members')
    display_table(conn, 'Loans')

    conn.close()

if __name__ == "__main__":
    main()
```

```
=== Initial Tables ===
 Content of Books:
(1, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Fiction', 1925, 5)
(2, 'To Kill a Mockingbird', 'Harper Lee', 'Classic', 1960, 4)
(3, '1984', 'George Orwell', 'Dystopian', 1949, 6)


 Content of Members:
(1, 'Alice Johnson', 'alice@email.com', '9876543210', '2025-10-01')
(2, 'Bob Smith', 'bob@email.com', '9123456789', '2025-10-05')
(3, 'Carol Lee', 'carol@email.com', '9988776655', '2025-10-10')
```

```
 Content of Loans:
(1, 1, 1, '2025-10-10', '2025-10-24')
(2, 2, 2, '2025-10-15', '2025-10-29')
(3, 3, 3, '2025-10-20', '2025-11-03')


 Books borrowed by member_id = 1:
(1, 'The Great Gatsby', 'F. Scott Fitzgerald', '2025-10-10', '2025-10-24')


 Member_id 1 borrowed book_id 2 successfully.

 Content of Books:
(1, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Fiction', 1925, 5)
(2, 'To Kill a Mockingbird', 'Harper Lee', 'Classic', 1960, 3)
(3, '1984', 'George Orwell', 'Dystopian', 1949, 6)


 Content of Loans:
(1, 1, 1, '2025-10-10', '2025-10-24')
(2, 2, 2, '2025-10-15', '2025-10-29')
(3, 3, 3, '2025-10-20', '2025-11-03')
(4, 2, 1, '2025-10-25', '2025-11-08')


 Member_id 3 deleted safely.

 Content of Members:
(1, 'Alice Johnson', 'alice@email.com', '9876543210', '2025-10-01')
(2, 'Bob Smith', 'bob@email.com', '9123456789', '2025-10-05')


 Content of Loans:
(1, 1, 1, '2025-10-10', '2025-10-24')
(2, 2, 2, '2025-10-15', '2025-10-29')
(4, 2, 1, '2025-10-25', '2025-11-08')
```