

ASSIGNMENT – 16.3

NAME : BHAWWANA SHRE

HALLTICKECT NO : 2403A52311

BATCH NUMBER : 01

COURSE CODE : 24CS002PC215

PROGRAM NAME : B.TECH

YEAR/SEM : 2ND AND 3RD

TASK 1 :

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

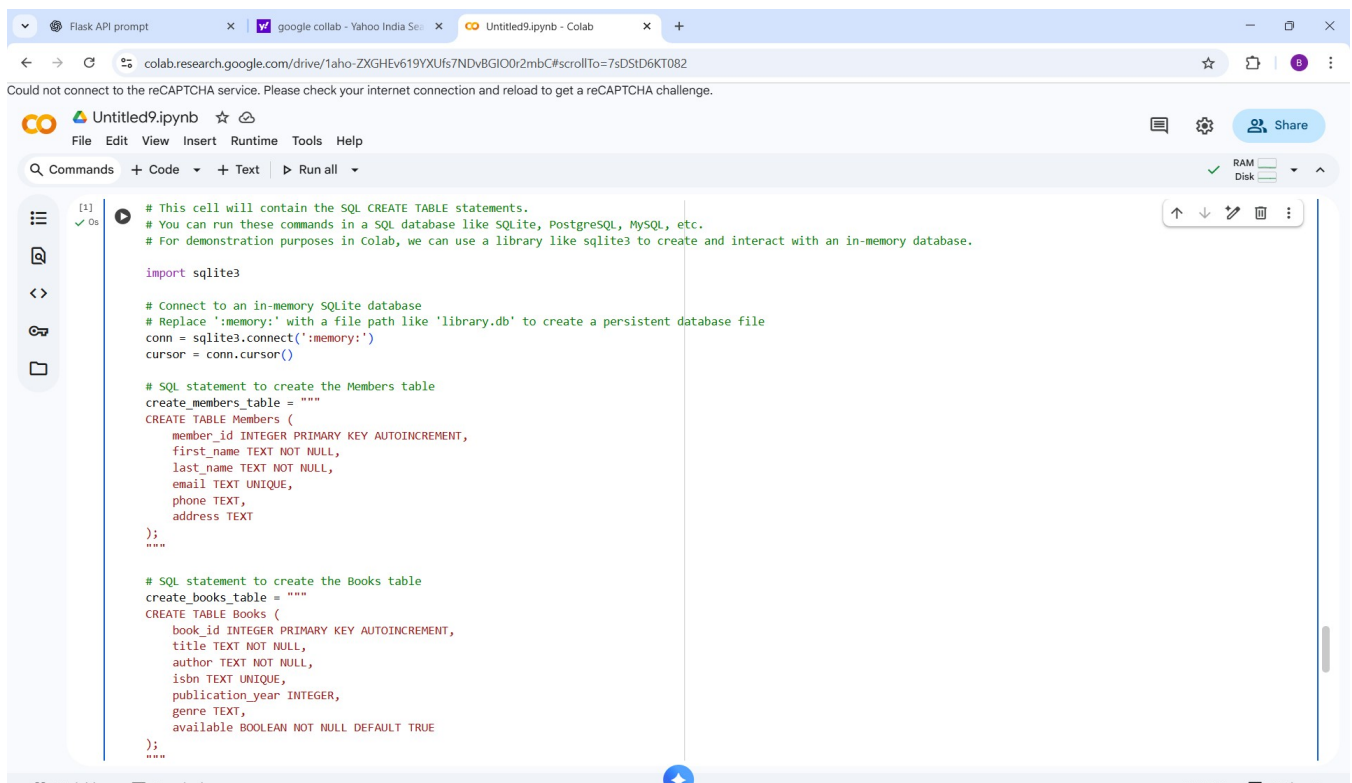
SQL Code

```
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)  
);
```

PROMPT :

“Design a Database Schema for a Library Management System with tables: Members, Books, and Loans — including primary keys, foreign keys, and necessary attributes. Provide SQL CREATE TABLE statements.”

CODE :



The screenshot shows a Google Colab notebook interface. The browser tabs at the top include 'Flask API prompt', 'google colab - Yahoo India Se...', and 'Untitled9.ipynb - Colab'. The address bar shows the Colab URL. A message at the top states: 'Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.' The notebook toolbar shows 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. The code editor contains the following Python code:

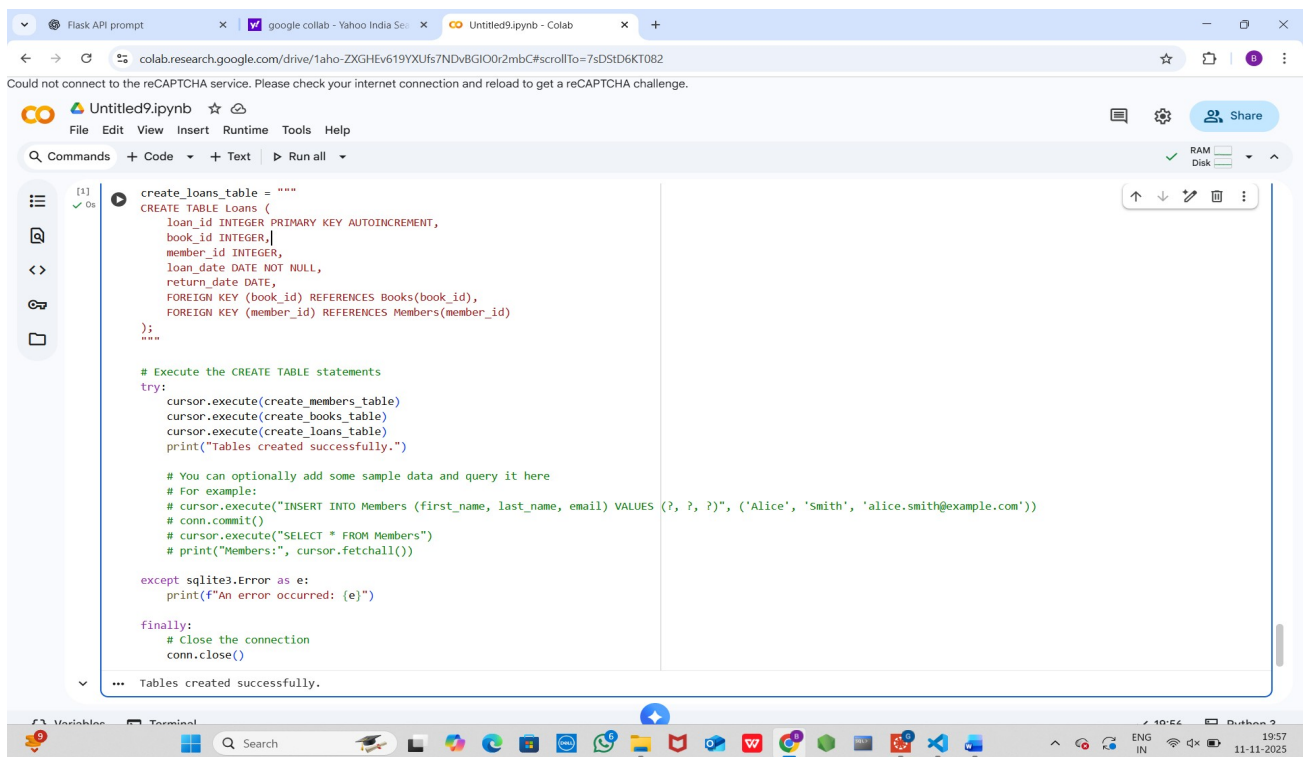
```
[1] ✓ 0s
# This cell will contain the SQL CREATE TABLE statements.
# You can run these commands in a SQL database like SQLite, PostgreSQL, MySQL, etc.
# For demonstration purposes in Colab, we can use a library like sqlite3 to create and interact with an in-memory database.

import sqlite3

# Connect to an in-memory SQLite database
# Replace ':memory:' with a file path like 'library.db' to create a persistent database file
conn = sqlite3.connect(':memory:')
cursor = conn.cursor()

# SQL statement to create the Members table
create_members_table = """
CREATE TABLE Members (
    member_id INTEGER PRIMARY KEY AUTOINCREMENT,
    first_name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    email TEXT UNIQUE,
    phone TEXT,
    address TEXT
);
"""

# SQL statement to create the Books table
create_books_table = """
CREATE TABLE Books (
    book_id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT NOT NULL,
    author TEXT NOT NULL,
    isbn TEXT UNIQUE,
    publication_year INTEGER,
    genre TEXT,
    available BOOLEAN NOT NULL DEFAULT TRUE
);
"""
```



The screenshot shows a Google Colab notebook titled 'Untitled9.ipynb'. The code in the cell defines a SQLite database schema with three tables: 'Members', 'Books', and 'Loans'. The 'Loans' table has foreign keys to 'Books' and 'Members'. The code then executes the CREATE TABLE statements, inserts sample data into the 'Members' table, and prints the results. The output at the bottom of the cell shows 'Tables created successfully.'.

```
create_loans_table = """
CREATE TABLE Loans (
    loan_id INTEGER PRIMARY KEY AUTOINCREMENT,
    book_id INTEGER,
    member_id INTEGER,
    loan_date DATE NOT NULL,
    return_date DATE,
    FOREIGN KEY (book_id) REFERENCES Books(book_id),
    FOREIGN KEY (member_id) REFERENCES Members(member_id)
);
"""

# Execute the CREATE TABLE statements
try:
    cursor.execute(create_members_table)
    cursor.execute(create_books_table)
    cursor.execute(create_loans_table)
    print("Tables created successfully.")

    # You can optionally add some sample data and query it here
    # For example:
    # cursor.execute("INSERT INTO Members (first_name, last_name, email) VALUES (?, ?, ?)", ('Alice', 'Smith', 'alice.smith@example.com'))
    # conn.commit()
    # cursor.execute("SELECT * FROM Members")
    # print("Members:", cursor.fetchall())

except sqlite3.Error as e:
    print(f"An error occurred: {e}")

finally:
    # Close the connection
    conn.close()

... Tables created successfully.
```

OUTPUT :

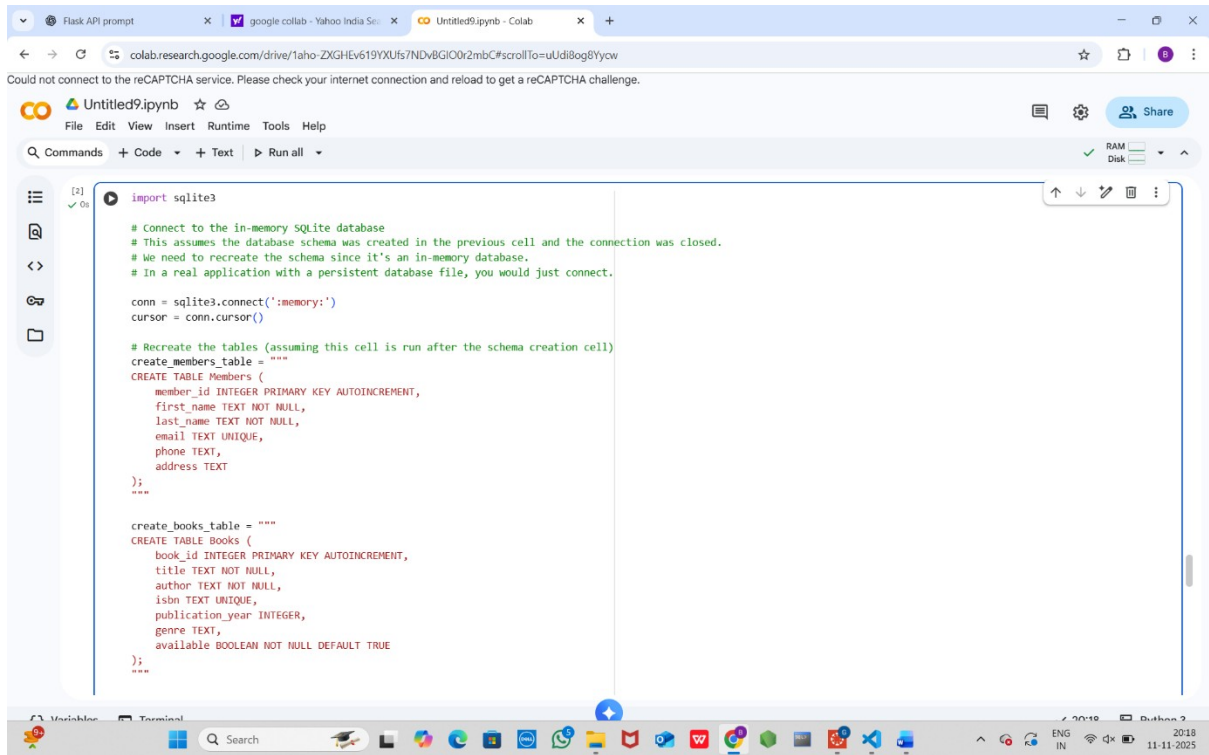
Tables created successfully.

TASK 2 :

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

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

CODE :



```
import sqlite3

# Connect to the in-memory SQLite database
# This assumes the database schema was created in the previous cell and the connection was closed.
# We need to recreate the schema since it's an in-memory database.
# In a real application with a persistent database file, you would just connect.

conn = sqlite3.connect(':memory:')
cursor = conn.cursor()

# Recreate the tables (assuming this cell is run after the schema creation cell)
create_members_table = """
CREATE TABLE Members (
    member_id INTEGER PRIMARY KEY AUTOINCREMENT,
    first_name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    email TEXT UNIQUE,
    phone TEXT,
    address TEXT
);
"""

create_books_table = """
CREATE TABLE Books (
    book_id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT NOT NULL,
    author TEXT NOT NULL,
    isbn TEXT UNIQUE,
    publication_year INTEGER,
    genre TEXT,
    available BOOLEAN NOT NULL DEFAULT TRUE
);
"""
```

OUTPUT :

Tables recreated for inserting data.

Sample data inserted successfully.

--- Loans ---

[(1, 1, 1, '2023-10-26', None), (2, 2, 2, '2023-10-20', None), (3, 3, 1, '2023-10-25', None)]

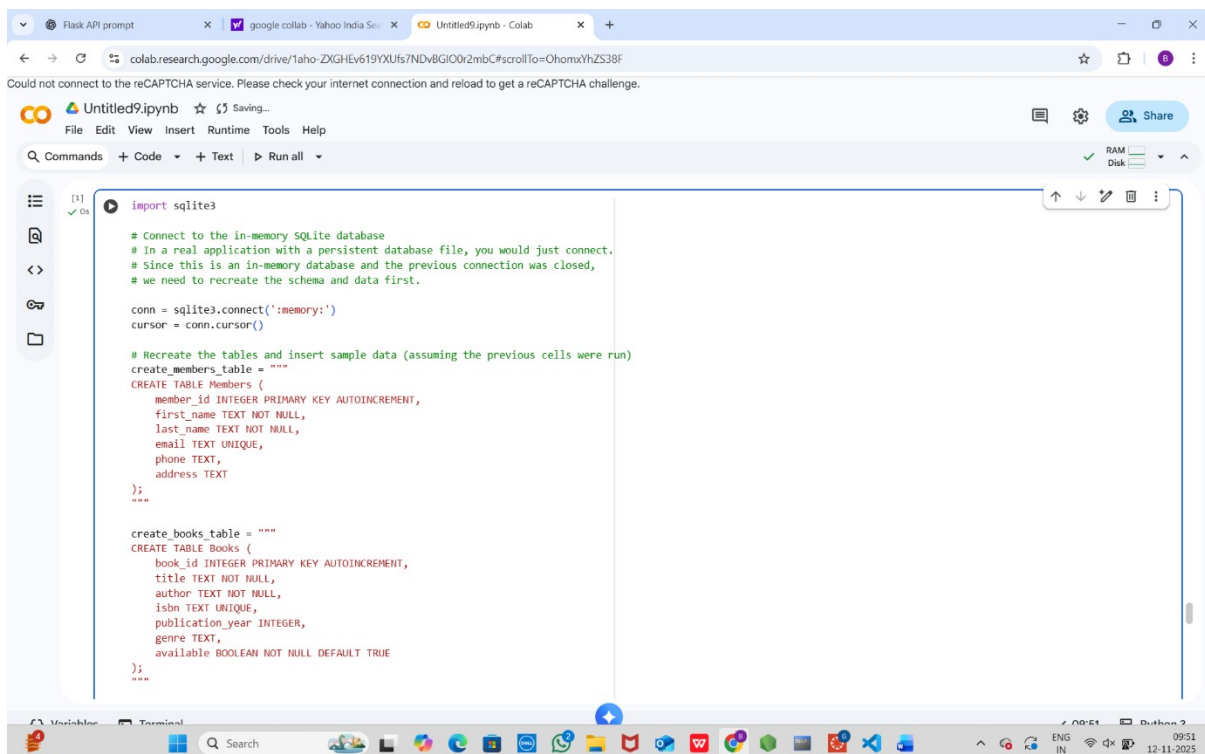
TASK 3 :

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

PROMPT :

“Generate an SQL query to list all books borrowed by a specific member using the Library Management System schema.”

CODE :



```
[1] ✓ C#
import sqlite3

# Connect to the in-memory SQLite database
# In a real application with a persistent database file, you would just connect.
# Since this is an in-memory database and the previous connection was closed,
# we need to recreate the schema and data first.

conn = sqlite3.connect(':memory:')
cursor = conn.cursor()

# Recreate the tables and insert sample data (assuming the previous cells were run)
create_members_table = """
CREATE TABLE Members (
    member_id INTEGER PRIMARY KEY AUTOINCREMENT,
    first_name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    email TEXT UNIQUE,
    phone TEXT,
    address TEXT
);
"""

create_books_table = """
CREATE TABLE Books (
    book_id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT NOT NULL,
    author TEXT NOT NULL,
    isbn TEXT UNIQUE,
    publication_year INTEGER,
    genre TEXT,
    available BOOLEAN NOT NULL DEFAULT TRUE
);
"""
```

OUTPUT :

Database schema and data recreated for querying.

--- Books borrowed by Member ID 1 ---

*('The Hitchhikers Guide to the Galaxy',
'Douglas Adams', '2023-10-26', None)*

('1984', 'George Orwell', '2023-10-25', None)