

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	Academic Year: 2025-2026
Course Coordinator Name		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)		
Course Code	24CS002PC215	Course Title	AI Assisted Coding
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment	Week 8 - Wednesday	Time(s)	
Duration	2 Hours	Applicable to Batches	
Assignment Number: 16.3 (Present assignment number)/24 (Total number of assignments)			
Q.No.	Question		Expected Time to complete
1	Lab 16 – Database Design and Queries: Schema Design and SQL Generation Lab Objectives <ul style="list-style-type: none">To practice basic SQL query generation with AI assistance.To analyze AI-suggested queries for correctness and efficiency.		Week 5 - 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.

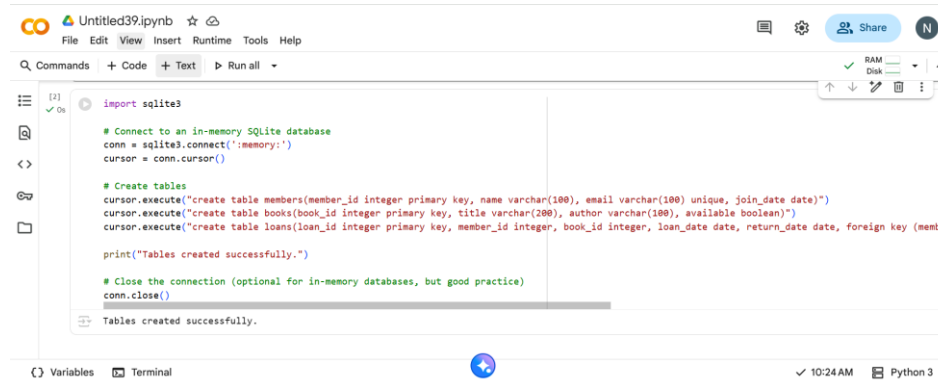
Task Description #1 – Schema Generation

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

SQL Code

```
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)  
);
```

PROMPT1: Design a database schema for a Library Management System of Books, Members, and Loans. Include primary and foreign keys.



```
import sqlite3

# Connect to an in-memory SQLite database
conn = sqlite3.connect(':memory:')
cursor = conn.cursor()

# Create tables
cursor.execute("create table members(member_id integer primary key, name varchar(100), email varchar(100) unique, join_date date)")
cursor.execute("create table books(book_id integer primary key, title varchar(200), author varchar(100), available boolean)")
cursor.execute("create table loans(loan_id integer primary key, member_id integer, book_id integer, loan_date date, return_date date, foreign key (member_id) references members(member_id))")

print("Tables created successfully.")

# Close the connection (optional for in-memory databases, but good practice)
conn.close()

Tables created successfully.
```

EXPLANATION:


We need to import sql library and the methods then only it will run properly. `import sqlite3`: Imports the `sqlite3` library, which provides an interface to work with SQLite databases. `conn = sqlite3.connect(':memory:')`: Establishes a connection to an in-memory SQLite database. This means the database exists only in the computer's RAM and will be lost when the program finishes. Using `:memory:` is useful for temporary databases or testing. `cursor = conn.cursor()`: Creates a cursor object. Cursors are used to execute SQL commands and fetch results from the database.

Task Description #2 – Error Insert Data

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

PROMPT1: Insert atleast 3 records(rows) of each table (members ,books and loan)

CODE:



```
# Sample INSERT INTO statements
insert_members_sql = """
INSERT INTO members (member_id, name, email, join_date) VALUES
(1, 'Alice Smith', 'alice.smith@example.com', '2023-01-15'),
(2, 'Bob Johnson', 'bob.johnson@example.com', '2023-02-20'),
(3, 'Charlie Brown', 'charlie.brown@example.com', '2023-03-10');
"""

insert_books_sql = """
INSERT INTO books (book_id, title, author, available) VALUES
(101, 'The Great Gatsby', 'F. Scott Fitzgerald', 1),
(102, 'To Kill a Mockingbird', 'Harper Lee', 0),
(103, '1984', 'George Orwell', 1);
"""

insert_loans_sql = """
INSERT INTO loans (loan_id, member_id, book_id, loan_date, return_date) VALUES
(1001, 1, 101, '2023-04-01', '2023-04-15'),
(1002, 2, 102, '2023-04-05', NULL),
(1003, 3, 103, '2023-04-10', '2023-04-25');
"""

print("Sample INSERT INTO queries generated.")

Sample INSERT INTO queries generated.
```

EXPLANATION:

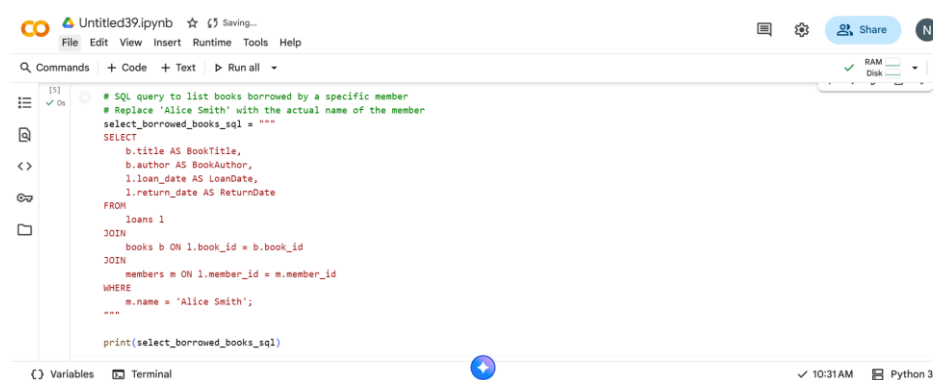
In this Python code defines three multi-line strings, each containing an SQL INSERT INTO query. These queries are designed to insert sample data into the members, books, and loans tables respectively, with three sample records provided for each table. The code concludes by printing a confirmation message indicating that these sample queries have been generated and stored in variables.

Task Description #3 – Basic Queries

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

PROMPT: Write an SQL query to list all books borrowed by a specific member.


CODE:

The screenshot shows a Jupyter Notebook window titled 'Untitled39.ipynb'. The code cell contains a multi-line string for an SQL query, preceded by a comment: '# SQL query to list books borrowed by a specific member # Replace 'Alice Smith' with the actual name of the member'. The query is a SELECT statement that joins the 'loans' table with the 'books' and 'members' tables. It filters for the member named 'Alice Smith' and selects the book title, author, loan date, and return date. The code is wrapped in a Python print statement.

```
[5]: # SQL query to list books borrowed by a specific member
# Replace 'Alice Smith' with the actual name of the member
select_borrowed_books_sql = """
SELECT
    b.title AS BookTitle,
    b.author AS BookAuthor,
    l.loan_date AS LoanDate,
    l.return_date AS ReturnDate
FROM
    loans l
JOIN
    books b ON l.book_id = b.book_id
JOIN
    members m ON l.member_id = m.member_id
WHERE
    m.name = 'Alice Smith';
"""

print(select_borrowed_books_sql)
```

OUTPUT:

The output shows the SQL query that was generated, formatted with indentation for readability.

```
SELECT
    b.title AS BookTitle,
    b.author AS BookAuthor,
    l.loan_date AS LoanDate,
    l.return_date AS ReturnDate
FROM
    loans l
JOIN
    books b ON l.book_id = b.book_id
JOIN
    members m ON l.member_id = m.member_id
WHERE
    m.name = 'Alice Smith';
```

EXPLANATION:

This SQL query is designed to retrieve information about books borrowed by a specific member from a library database. It works by joining the loans, books, and members tables together based on their related book_id and member_id columns. The SELECT statement specifies the columns to retrieve: the book's title and author from the books table, and the loan and return dates from the loans table.

Finally, the WHERE clause filters the results to include only the loans associated with the member whose name matches 'Alice Smith' (or the name you specify), effectively listing all books borrowed by that particular member.

Task Description #4 – Update and Delete Queries

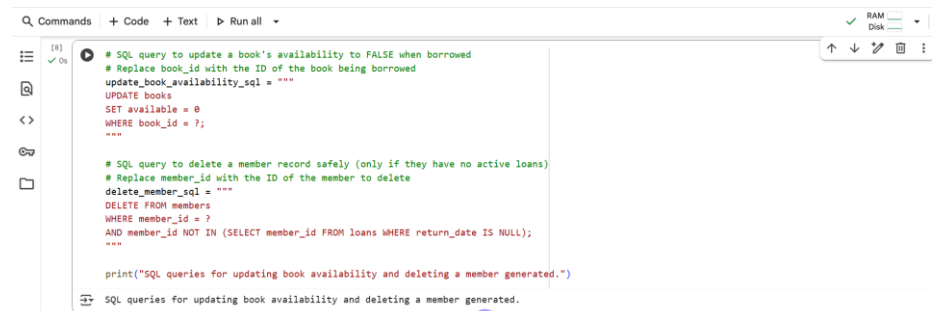
Task: Generate queries with AI for:

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

PROMPT1:write a query on updating a books availability to false

PROMPT2:delete the member record

CODE:



```
[N] ✓ On
Q Commands + Code + Text ▶ Run all
# SQL query to update a book's availability to FALSE when borrowed
# Replace book_id with the ID of the book being borrowed
update_book_availability_sql = """
UPDATE books
SET available = 0
WHERE book_id = ?;
"""

# SQL query to delete a member record safely (only if they have no active loans)
# Replace member_id with the ID of the member to delete
delete_member_sql = """
DELETE FROM members
WHERE member_id = ?
AND member_id NOT IN (SELECT member_id FROM loans WHERE return_date IS NULL);
"""

print("SQL queries for updating book availability and deleting a member generated.")
SQL queries for updating book availability and deleting a member generated.
```

EXPLANATION:

This code contains two SQL queries designed for managing book availability and member records. The first query, `update_book_availability_sql`, updates the `books` table to set the available status to false (represented by 0) for a specific book, identified by its `book_id`. This is useful when a book is borrowed. The second query, `delete_member_sql`, safely removes a record from the `members` table for a specific `member_id`, but only if that member has no active loans (meaning all their borrowed books have been returned, indicated by a non-null `return_date` in the `loans` table).