

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. Mohammad 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	24CS002PC 215	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)			
1	Question		Expected Time to complete
	Lab 16 – Database Design and Queries: Schema Design and SQL Generation		Week 5 - Monday
	Lab Objectives		

- To practice basic SQL query generation with AI assistance.
- To analyze AI-suggested queries for correctness and efficiency.
- 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)  
);
```

OUTPUT:

```

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)
);

```

```

File "/tmp/ipython-input-2452795888.py", line 1
CREATE TABLE Members (
    ^
SyntaxError: invalid syntax

```

OBSERVATION:

The schema defines three tables: Members (stores member info with unique IDs), Books (stores book info with unique IDs and availability), and Loans (records who borrowed which book and when, linking members and books via foreign keys).

Task Description #2 – Error Insert Data

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

OUTPUT:

```

# Insert sample data into Members table
cursor.execute('''
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 sample data into Books table
cursor.execute('''
INSERT INTO Books (book_id, title, author, available) VALUES
(101, 'The Great Gatsby', 'F. Scott Fitzgerald', TRUE),
(102, 'To Kill a Mockingbird', 'Harper Lee', FALSE),
(103, '1984', 'George Orwell', TRUE);
''')

# Insert sample data into Loans table
cursor.execute('''
INSERT INTO Loans (loan_id, member_id, book_id, loan_date, return_date) VALUES
(1001, 1, 102, '2023-04-01', '2023-04-15'),
(1002, 2, 101, '2023-04-05', NULL),
(1003, 3, 103, '2023-04-10', NULL);
''')


# Commit the changes
conn.commit()

print("Database schema created and sample data inserted successfully.")

# You can now query the database, for example:
# cursor.execute("SELECT * FROM Members")
# print(cursor.fetchall())

# Close the connection when done
# conn.close()

```

 Database schema created and sample data inserted successfully.

OBSERVATION:

The INSERT queries in the successful cell populate the Members, Books, and Loans tables with sample data, allowing you to work with the database.


Task Description #3 – Basic Queries

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

OUTPUT:

✓ Query using member name

sql

 Copy code

```
SELECT
    b.book_id,
    b.title,
    b.author,
    l.loan_date,
    l.return_date
FROM
    Books b
JOIN
    Loans l ON b.book_id = l.book_id
JOIN
    Members m ON l.member_id = m.member_id
WHERE
    m.name = 'John Doe';
```

✓ Query using member_id

sql

Copy code

```
SELECT
    b.book_id,
    b.title,
    b.author,
    l.loan_date,
    l.return_date
FROM
    Books b
JOIN
    Loans l ON b.book_id = l.book_id
WHERE
    l.member_id = 1;
```

OBSERVATION:

The query retrieves all books borrowed by a specific member by joining the Members, Loans, and Books tables. It displays each book's title, author, and loan details based on the member's ID or name.

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.

OUTPUT:

```
-- Update a book's availability to FALSE when borrowed
UPDATE Books
SET available = FALSE
WHERE book_id = 101;

-- Safely delete a member record (automatically deletes their loans)
DELETE FROM Members
WHERE member_id = 5;
```

OBSERVATION:

- The query updates the available status to prevent multiple borrowings of the same book.
- The ON DELETE CASCADE ensures that when a member is deleted, their related loan records are automatically removed — maintaining data integrity.