

Lab 16 – Database Design and Queries: Schema Design and SQL Generation

Task Description #1 – Schema Generation

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

PROMPT : design a schema for a library management system where the tables are books, members, loans

Here's a possible schema for a library management system with tables for `books`, `members`, and `loans`:

`books` table:

- `book_id` (Primary Key, Integer): Unique identifier for each book.
- `title` (VARCHAR): The title of the book.
- `author` (VARCHAR): The author of the book.
- `available_copies` (Integer): Number of available copies for loan.

`members` table:

- `member_id` (Primary Key, Integer): Unique identifier for each member.
- `first_name` (VARCHAR): The member's first name.
- `last_name` (VARCHAR): The member's last name.
- `address` (VARCHAR): The member's address.
- `phone_number` (VARCHAR): The member's phone number.
- `email` (VARCHAR): The member's email address.
- `membership_date` (DATE): The date the member joined the library.

`loans` table:

- `loan_id` (Primary Key, Integer): Unique identifier for each loan.
- `book_id` (Foreign Key, Integer): References the `book_id` in the `books` table.
- `member_id` (Foreign Key, Integer): References the `member_id` in the `members` table.
- `loan_date` (DATE): The date the book was loaned out.
- `due_date` (DATE): The date the book is due to be returned.
- `return_date` (DATE): The actual date the book was returned (can be NULL if not returned yet).
- `status` (VARCHAR): The current status of the loan (e.g., 'borrowed', 'returned', 'overdue').

Relationships:

- One book can have many loans (one-to-many relationship between `books` and `loans`).
- One member can have many loans (one-to-many relationship between `members` and `loans`).

This schema provides a basic structure for managing books, members, and loans in a library system. You can further expand on this by adding more tables for things like authors, publishers, fines, reservations, etc., depending on the complexity of the system you need.

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 insert into queries for the above schema with atleast 3 sample values

```
-- Insert data into the books table
INSERT INTO books (book_id, title, author, available_copies) VALUES
(1, 'The Hitchhiker's Guide to the Galaxy', 'Douglas Adams', 5),
(2, 'Pride and Prejudice', 'Jane Austen', 3),
(3, '1984', 'George Orwell', 2);

-- Insert data into the members table
INSERT INTO members (member_id, first_name, last_name, address, phone_number, email) VALUES
(101, 'Arthur', 'Dent', '42 Wallaby Way', '555-1234', 'arthur.dent@example.com'),
(102, 'Elizabeth', 'Bennet', 'Longbourn', '555-5678', 'elizabeth.b@example.com'),
(103, 'Winston', 'Smith', 'Airstrip One', '555-8765', 'winston.s@example.com');

-- Insert data into the loans table
INSERT INTO loans (loan_id, book_id, member_id, loan_date, due_date, return_date, status) VALUES
(1001, 1, 101, '2023-04-01', '2023-04-15', NULL, 'borrowed'),
(1002, 2, 102, '2023-04-05', '2023-04-19', '2023-04-18', 'returned'),
(1003, 3, 103, '2023-04-10', '2023-04-24', NULL, 'borrowed');
```

Task Description #3 – Basic Queries

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

PROMPT: generate a query to list all the books borrowed by a certain member

```
SELECT
    b.title,
    b.author,
    l.loan_date,
    l.due_date
FROM
```

```
        loans l
JOIN
        books b ON l.book_id = b.book_id
WHERE
        l.member_id = [member_id]; -- Replace [member_id] with the actual member ID
```

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 queries with AI for: • Updating a book's availability to FALSE when borrowed. • Deleting a member record safely

```
-- Query to update a book's availability when borrowed
UPDATE books
SET available_copies = available_copies - 1 -- Assuming available_copies is decremented
WHERE book_id = [book_id]; -- Replace [book_id] with the ID of the borrowed book
```

```
-- Query to safely delete a member record
-- This is a basic delete. For a truly "safe" deletion in a production system,
-- you would typically need to handle related records in the 'loans' table first
-- (e.g., setting their member_id to NULL if allowed, archiving, or deleting the loans)
DELETE FROM members
WHERE member_id = [member_id]; -- Replace [member_id] with the ID of the member
```