Library Management System – Hackathon Level 2 Submission

Use Case Title:

Library Management System

Student Name:

Guruji.R

Register Number:

C2S27510

Institution:

Theni Kammavar Sangam College of Arts and Science

Department:

BCA

Date of Submission:

19- 04 - 2025

1. Problem Statement

Libraries require a robust system to manage book lending, track borrowed books, monitor due dates, and maintain accurate records. The challenge is to create an efficient Library Management System using SQLite 3 to streamline these operations.

2. Database Design & Implementation

2.1 Database Creation & Tables

The Library Management System will use the following SQL queries to create tables:

SQL Queries for Table Creation:

```
CREATE TABLE Books (
    BookID INTEGER PRIMARY KEY AUTOINCREMENT,
    Title TEXT NOT NULL,
    Author TEXT NOT NULL,
    Genre TEXT,
    ISBN TEXT UNIQUE,
    AvailabilityStatus TEXT CHECK(AvailabilityStatus IN ('Available', 'Issued'))
);
```

```
CREATE TABLE Users (
    UserID INTEGER PRIMARY KEY AUTOINCREMENT,
    Name TEXT NOT NULL,
    ContactInfo TEXT,
    MembershipType TEXT CHECK(MembershipType IN ('Student', 'Faculty', 'Guest'))
);
```

```
CREATE TABLE Transactions (
    TransactionID INTEGER PRIMARY KEY AUTOINCREMENT,
    BookID INTEGER,
    UserID INTEGER,
    IssueDate DATE,
    ReturnDate DATE,
    Status TEXT CHECK(Status IN ('Issued', 'Returned')),
    FOREIGN KEY(BookID) REFERENCES Books(BookID),
    FOREIGN KEY(UserID) REFERENCES Users(UserID)
);
```

2.2 ER Diagram (Reverse Engineered)

The ER diagram for the Library Management System represents the relationships between books, users, and transactions. The diagram visually illustrates the structure of the database.

3. Queries for Data Management

3.1 Insert Sample Data

SQL Queries for Sample Data Insertion:

```
INSERT INTO Books (Title, Author, Genre, ISBN, AvailabilityStatus)
VALUES
('The Great Gatsby', 'F. Scott Fitzgerald', 'Fiction', '9780743273565', 'Available'),
('1984', 'George Orwell', 'Dystopian', '9780451524935', 'Available');
```

```
INSERT INTO Users (Name, ContactInfo, MembershipType)
VALUES
('John Doe', 'johndoe@example.com', 'Student'),
('Jane Smith', 'janesmith@example.com', 'Faculty');
```

```
INSERT INTO Transactions (BookID, UserID, IssueDate, ReturnDate, Status)
VALUES
(1, 1, '2025-04-01', '2025-04-15', 'Issued');
```

3.2 Retrieval Queries

SQL Queries for Data Retrieval:

```
SELECT * FROM Books WHERE AvailabilityStatus = 'Available';
```

```
SELECT Users.Name, Books.Title, Transactions.ReturnDate
FROM Transactions
JOIN Users ON Transactions.UserID = Users.UserID
JOIN Books ON Transactions.BookID = Books.BookID
WHERE Transactions.ReturnDate < DATE('now') AND Transactions.Status = 'Issued';
```

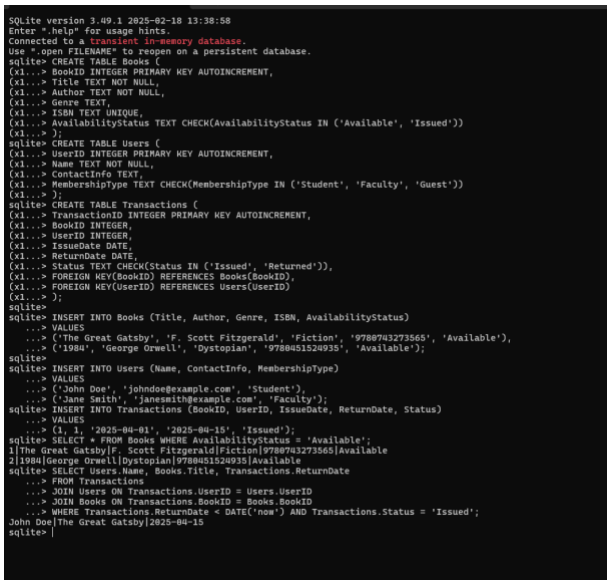
4. Implementation & Results

4.1 Execution Environment

The Library Management System was implemented using SQLite 3, and SQL queries were executed in SQLite Database Browser. Screenshots of query execution results have been attached.

4.2 Screenshots of Execution Results

Screenshots showing successful database creation, data insertion, and retrieval queries are attached in the GitHub repository.



5. GitHub Repository

5.1 Repository Link

https://github.com/mku60422sca010/Hackathon-SQL-Quires

5.2 Uploaded Files in Repository

The following files are available in the repository:

- SQL scripts for table creation and sample data insertion

- ER diagram of the database structure

- Screenshots of execution results