Use Case Title: [**Library Management System Scenario**]

Student Name: [A.V.MULLAI NATHI]

Register Number: [22529ER033]

Institution: [NADAR SARASWATHI COLLEGE OF ARTS AND SCIENCE]

Department: [INFORMATION TECHNOLOGY]

Date of Submission: [13/04/2025]

**Library Management System Scenario**

**1. Problem Statement**

Libraries often face challenges in efficiently managing their vast collection of books, tracking loans, monitoring overdue books, and handling user information. The lack of a robust system can lead to inefficiencies, lost books, and frustration for both staff and patrons. Traditional methods, such as manual record-keeping, are prone to errors and can be time-consuming.

Real-World Challenge: In today's digital age, libraries need a more streamlined and efficient approach to managing their operations. The challenge is to develop a Library Management System (LMS) using SQL that allows library staff to add new books, track loans, monitor overdue books, and manage users effectively. This solution will improve the overall efficiency of library operations, enhance user satisfaction, and ensure that books are easily accessible to users**.**

**2. Database Design & Implementation**

**Design**

The design phase involves planning the structure, features, and flow of the system. The key components include:

1. Database Schema:
   * Entities:
     + Books
     + Users
     + Loans
   * Relationships:
     + Books ↔ Loans (One-to-Many): A book can be loaned multiple times.
     + Users ↔ Loans (One-to-Many): A user can borrow multiple books.
2. User Interface (UI):
   * If you're integrating with an application, design pages/screens for:
     + Adding/Updating/Deleting Books
     + Registering/Managing Users
     + Issuing and Returning Books
     + Viewing Reports for Overdue Books
3. System Features:
   * Book Inventory Management
   * User Registration
   * Loan Processing
   * Overdue Notification System
   * Fine Calculation (optional)
4. Workflows:
   * Book Borrowing Workflow: Search Book → Verify Availability → Record Loan
   * Return Workflow: Verify Book and User → Update Return Date

**Implementation**

This step involves turning the design into reality. Using SQL, here's a detailed implementation:

1. Database Creation

Create a database and tables (already provided in the previous steps).

2. Data Insertion

Populate your database with sample data:

* Books:
* INSERT INTO Books (Title, Author, Genre, PublicationYear, CopiesAvailable)
* VALUES
* ('The Catcher in the Rye', 'J.D. Salinger', 'Fiction', 1951, 3),
* ('To Kill a Mockingbird', 'Harper Lee', 'Fiction', 1960, 5);
* Users:
* INSERT INTO Users (Name, Email, Phone, MembershipDate)
* VALUES
* ('John Doe', 'john.doe@example.com', '1234567890', '2025-01-10'),
* ('Jane Smith', 'jane.smith@example.com', '0987654321', '2024-12-05');

3. Basic Operations

* Track Available Copies: Update the CopiesAvailable field when a book is borrowed or returned.
* UPDATE Books
* SET CopiesAvailable = CopiesAvailable - 1
* WHERE BookID = 1;
* Monitor Overdue Books:
* SELECT u.Name, b.Title, l.DueDate
* FROM Loans l
* JOIN Users u ON l.UserID = u.UserID
* JOIN Books b ON l.BookID = b.BookID
* WHERE l.DueDate < CURDATE() AND l.ReturnDate IS NULL;

**2.1 Database Creation & Tables**

SQL queries to create the database and necessary tables based on the selected use case.

**Step 1: Create the Database**

CREATE DATABASE LibraryDB;

**Step 2: Use the Database**

USE LibraryDB;

**Step 3: Create the Tables**

**Table: books**

CREATE TABLE Books (

BookID INT PRIMARY KEY AUTO\_INCREMENT,

Title VARCHAR(255) NOT NULL,

Author VARCHAR(255),

Genre VARCHAR(100),

PublicationYear INT,

CopiesAvailable INT NOT NULL

);

**Table: user**

CREATE TABLE Users (

UserID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(255) NOT NULL,

Email VARCHAR(255) UNIQUE NOT NULL,

Phone VARCHAR(15),

MembershipDate DATE NOT NULL

);

**Table: loans**

CREATE TABLE Loans (

LoanID INT PRIMARY KEY AUTO\_INCREMENT,

BookID INT,

UserID INT,

LoanDate DATE NOT NULL,

DueDate DATE NOT NULL,

ReturnDate DATE,

FOREIGN KEY (BookID) REFERENCES Books(BookID),

FOREIGN KEY (UserID) REFERENCES Users(UserID)

);

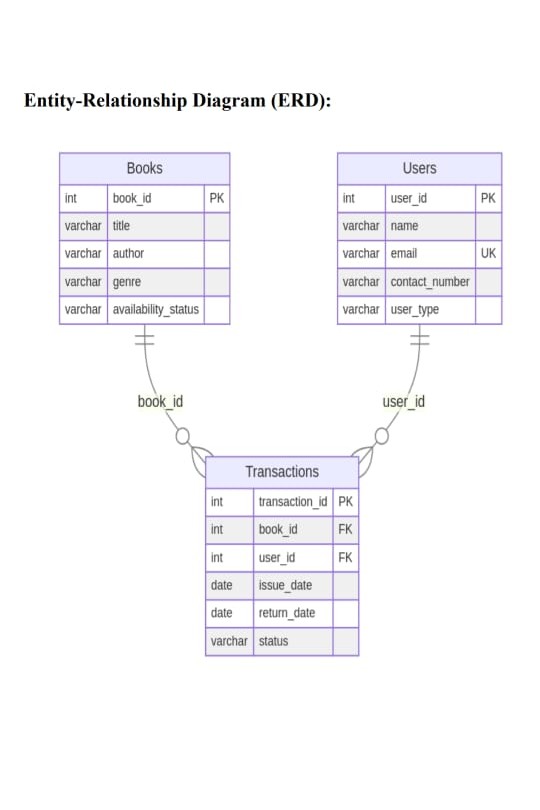
**2.2. ER Diagram (Reverse Engineered)**

**Using MySQL Workbench:**

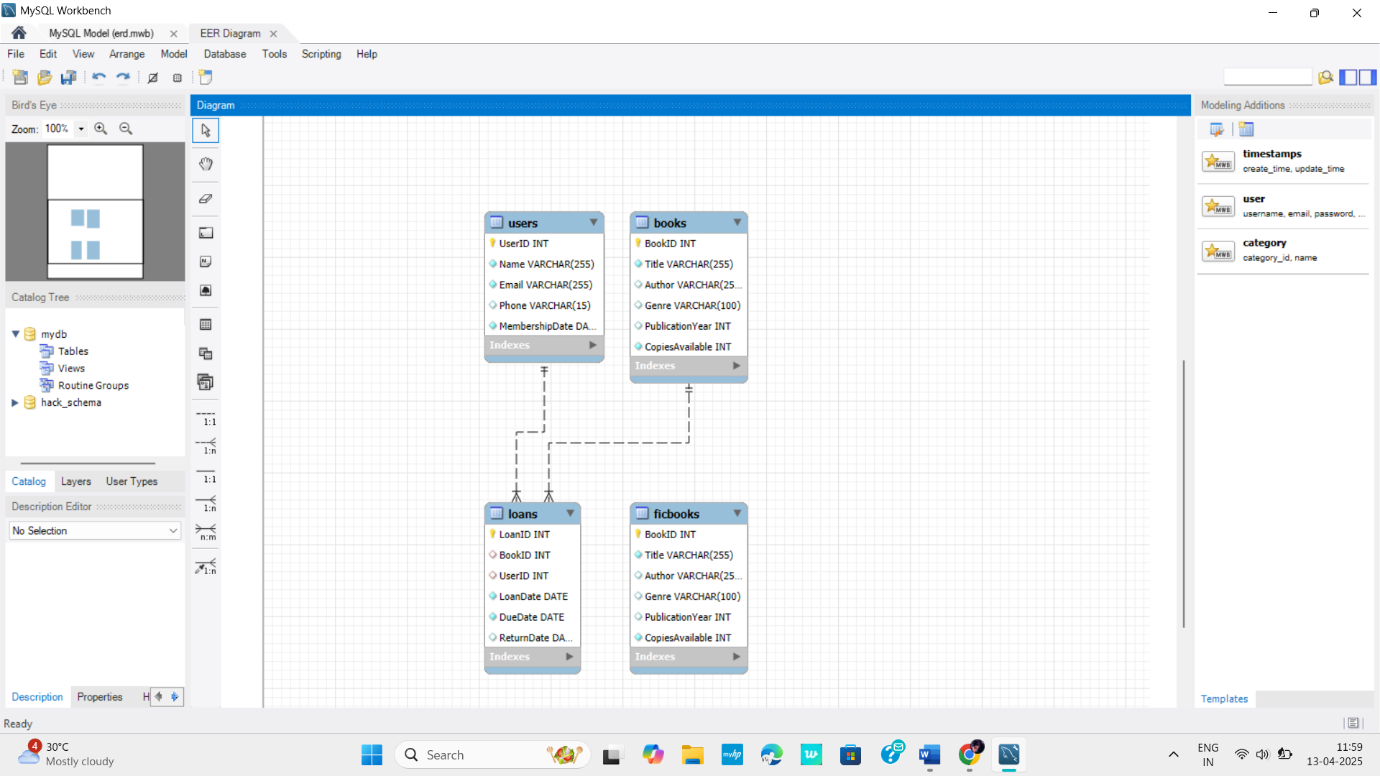
1. **Database Setup**: Ensure you've created the LibraryDB database and the Books, Users, and Loans tables with proper relationships.
2. **Reverse Engineering**:
   * Open MySQL Workbench.
   * Go to **Database > Reverse Engineer**.
   * Select your database connection and follow the prompts to import tables and relationships.
3. **Generate the Diagram**:
   * Once imported, go to **Models > EER Diagram** to view the auto-generated diagram.
4. **Save the Diagram**: Export it as an image or PDF.

**Using Oracle SQL Developer:**

1. **Connect to the Database**:
   * Open Oracle SQL Developer and connect to your database.
2. **Generate Data Model**:
   * Navigate to **Tools > Data Modeler > Import Data Dictionary**.
   * Select the tables (Books, Users, Loans) and import them.
3. **View the Diagram**:
   * Oracle SQL Developer will create the ER diagram with all the relationships.
4. **Customize and Export**: You can rearrange, label, and save the diagram.



**WORKBENCH GENERATED ER DIAGRAM:**

****

**3. Queries for Data Management**

**3.1 Insert Sample Data**

INSERT INTO Books (Title, Author, Genre, PublicationYear, CopiesAvailable)

VALUES

('1984', 'George Orwell', 'Dystopian', 1949, 5),

('To Kill a Mockingbird', 'Harper Lee', 'Fiction', 1960, 3),

('The Great Gatsby', 'F. Scott Fitzgerald', 'Classics', 1925, 4),

('A Brief History of Time', 'Stephen Hawking', 'Science', 1988, 2),

('The Catcher in the Rye', 'J.D. Salinger', 'Fiction', 1951, 6);

**3.2 Retrieval Queries**

**1. Fetch Books by Genre**

SELECT Title, Author, PublicationYear, CopiesAvailable

FROM Books

WHERE Genre = 'Fiction';

**2. Get Overdue Books**

SELECT u.Name AS UserName, b.Title AS BookTitle, l.DueDate

FROM Loans l

JOIN Users u ON l.UserID = u.UserID

JOIN Books b ON l.BookID = b.BookID

WHERE l.DueDate < CURDATE() AND l.ReturnDate IS NULL;

**3. List All Available Books**

SELECT Title, Author, Genre, CopiesAvailable

FROM Books

WHERE CopiesAvailable > 0;

**4. Track Loans by a Specific User**

SELECT b.Title, l.LoanDate, l.DueDate

FROM Loans l

JOIN Books b ON l.BookID = b.BookID

WHERE l.UserID = 1 AND l.ReturnDate IS NULL;

**5. Count Total Books by Genre**

SELECT Genre, COUNT(\*) AS TotalBooks

FROM Books

GROUP BY Genre;

**6. List Users with Active Loans**

SELECT DISTINCT u.Name, u.Email

FROM Loans l

JOIN Users u ON l.UserID = u.UserID

WHERE l.ReturnDate IS NULL;

**7. Get Most Borrowed Books**

SELECT b.Title, COUNT(l.LoanID) AS BorrowCount

FROM Loans l

JOIN Books b ON l.BookID = b.BookID

GROUP BY b.BookID

ORDER BY BorrowCount DESC

LIMIT 5;

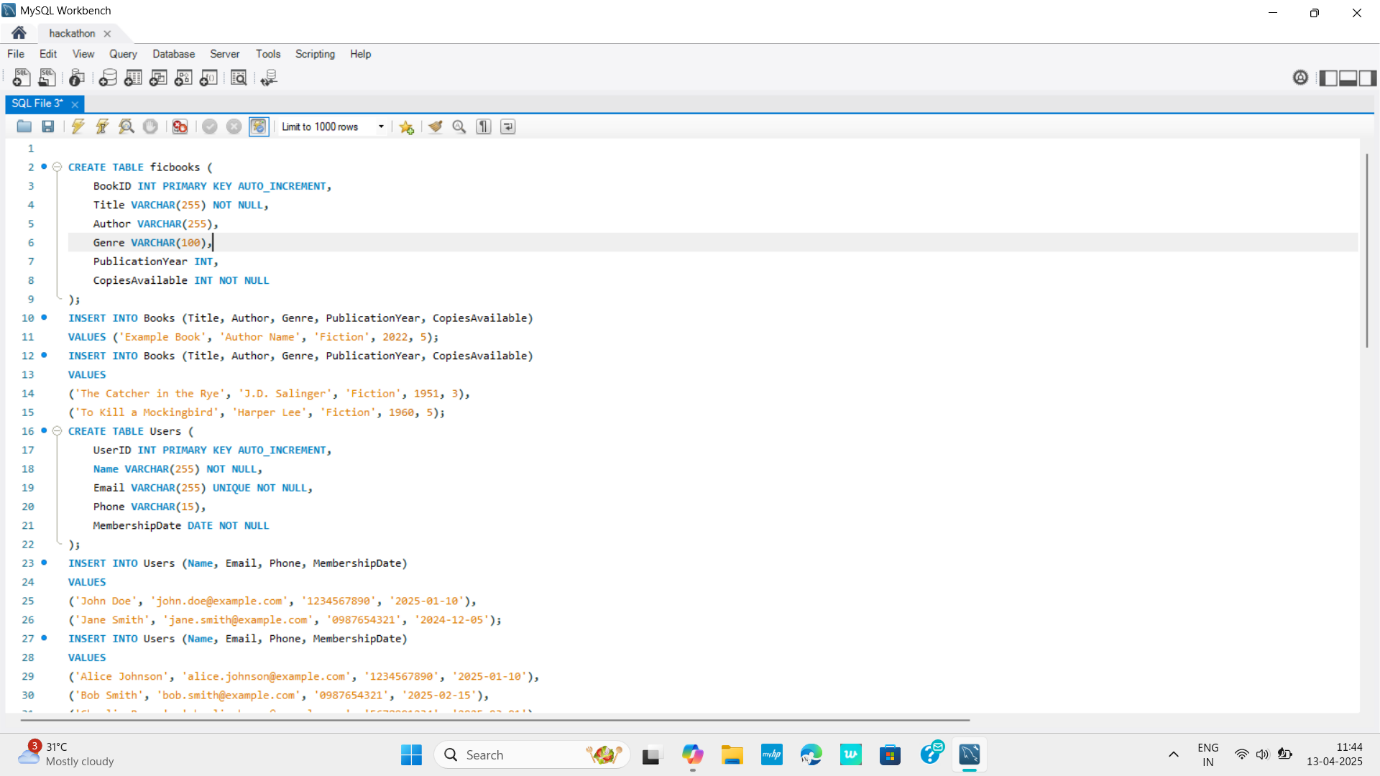
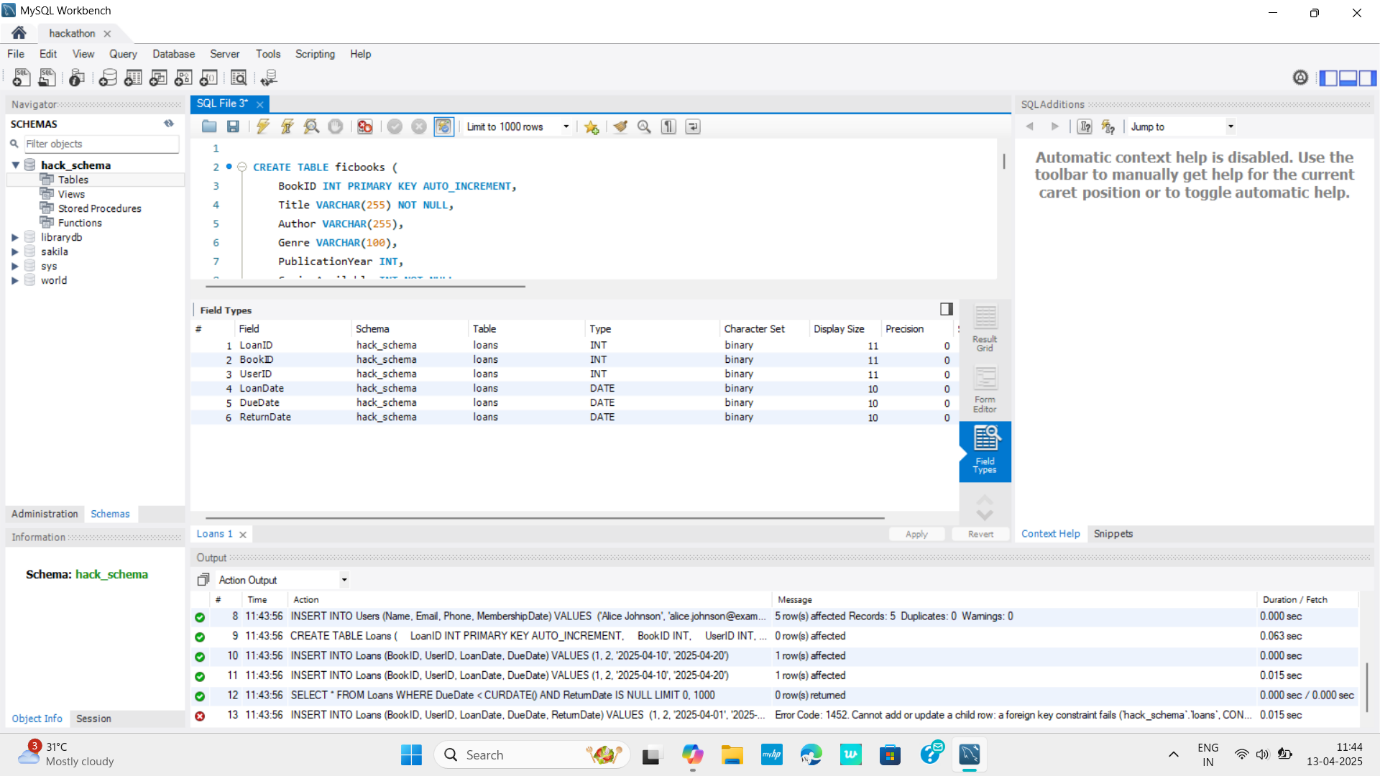
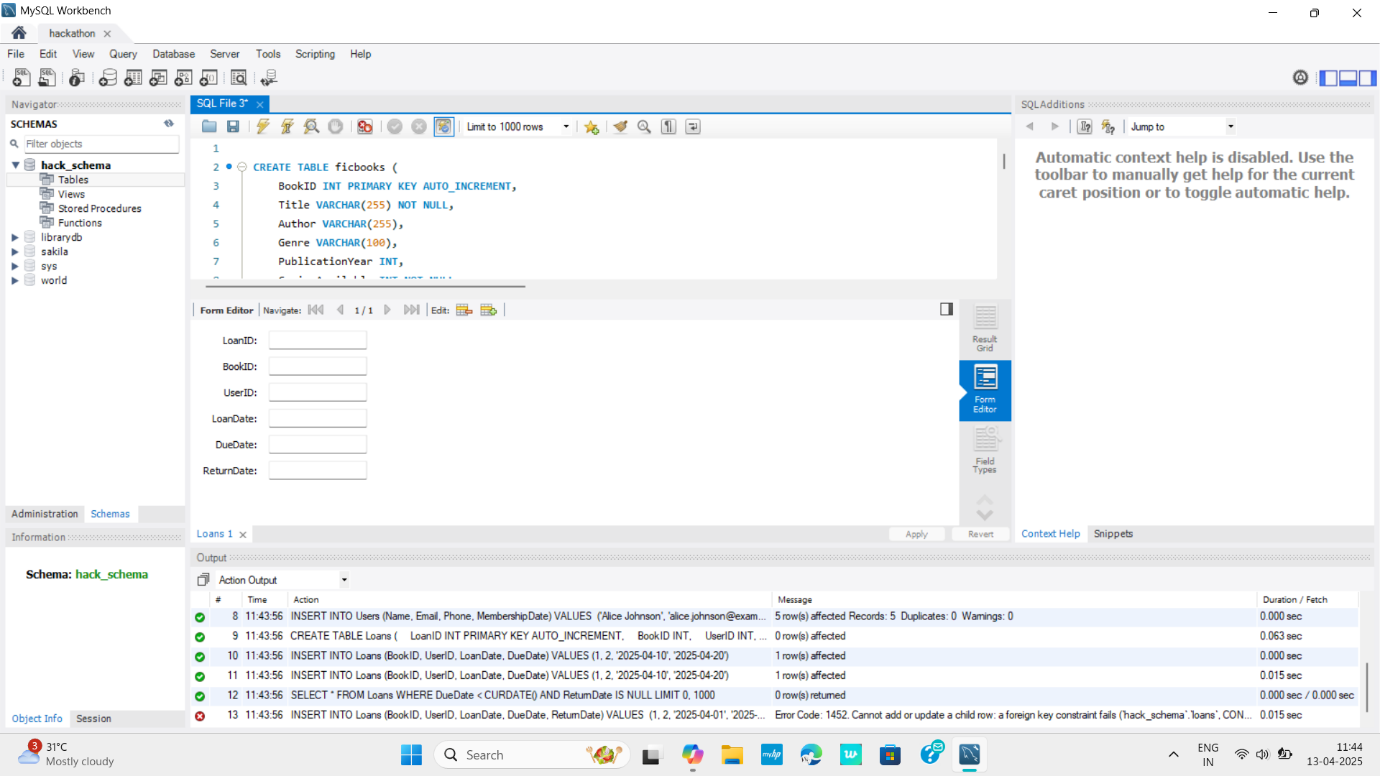
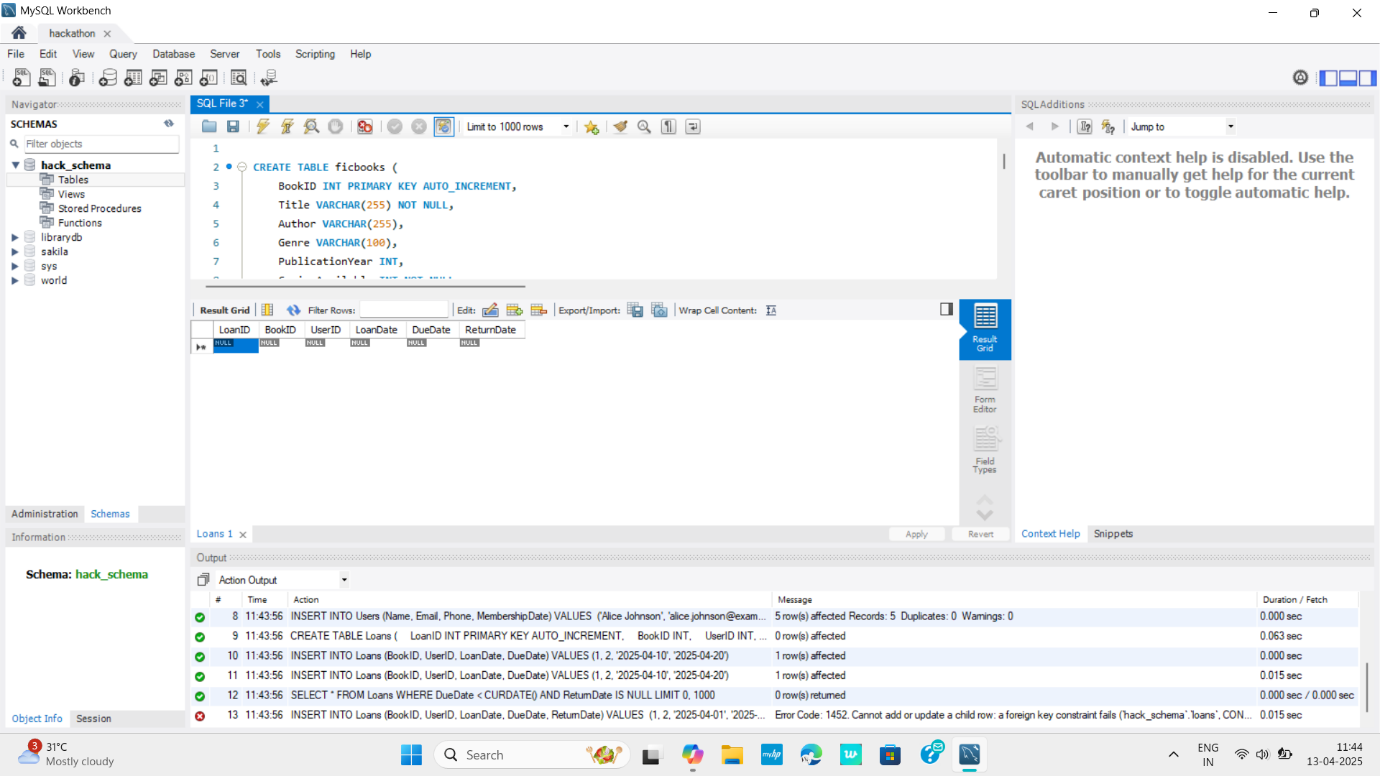
**4. IMPLEMENTATION & RESULTS**

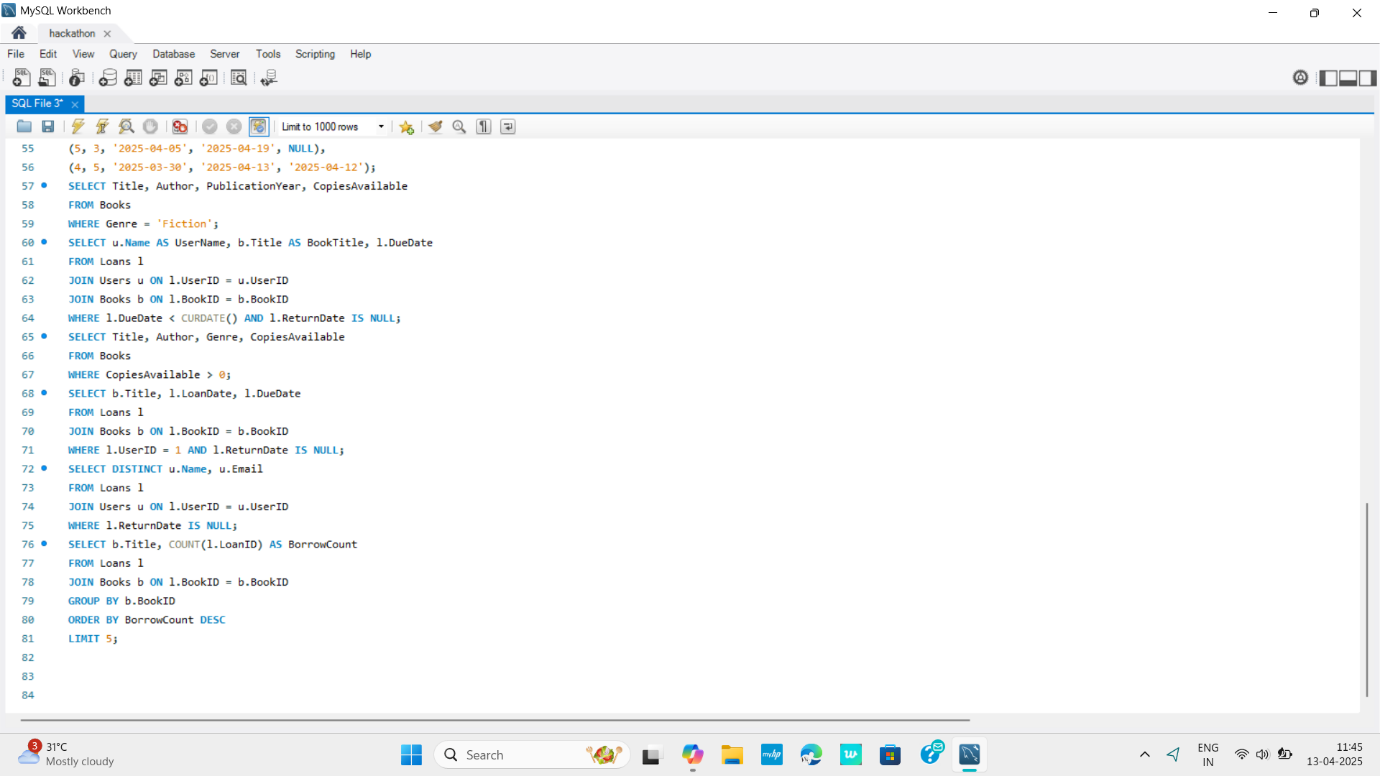
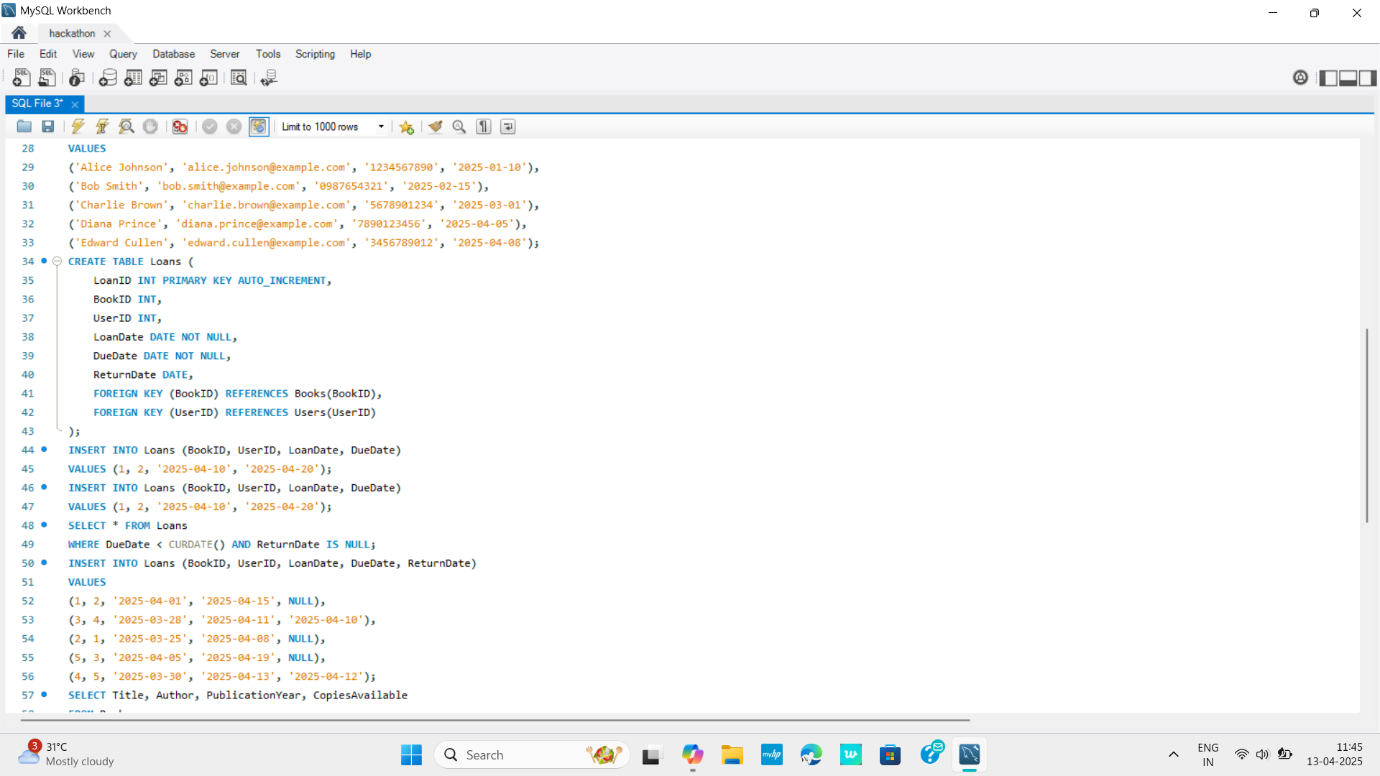
**4.1 Execution Environment:**

For the **Execution Environment**, you can specify the tools and platforms used to implement your Library Management System (LMS). Here's an example of how you might describe it:

1. **Database Management Tool**: MySQL Workbench (or Oracle SQL Developer)
   * Used for creating the database schema, inserting sample data, and executing SQL queries.
   * Enabled visualization of relationships and generation of the ER diagram.
2. **SQL Version**: MySQL 8.0 (or Oracle SQL if applicable)
   * Provided a robust platform for developing, testing, and managing the database.
3. **Operating System**: Windows 11
   * Served as the primary platform for running MySQL Workbench (or Oracle SQL Developer).
4. **Hardware**:
   * Processor: Intel Core i5 or equivalent
   * RAM: 8 GB (minimum recommended for smooth execution)
5. **Additional Tools** (if applicable):
   * A front-end application for connecting to the database (e.g., Python Flask/Django for testing or JavaScript with Node.js).

**4.2 Screenshots of Execution Results**

****



**5.GitHub Repository**

**5.1 & 5.2Git hub repository link:** [**oracle-hack/ at main · Nathi-devisor/oracle-hack**](https://github.com/Nathi-devisor/oracle-hack/tree/main)