

HACKATHON SUBMISSION (LEVEL-2-SOLUTION)

Use Case Title: Library Management System.

Student Name: M. Kalaiyarasi.

Register Number: U22CSE32237.

Institution: Sri Meenakshi Govt Arts college for Women (Autonomous),
Madurai. Department: Computer Science.

1. PROBLEM STATEMENT

Traditional library systems focus primarily on book lending and catalog management, which often leads to inefficiencies such as:

- Difficulty for users to find books matching their interests.
- Overwhelming book choices without personalized guidance.
- Lack of user engagement with available resources.
- No intelligent system to analyze reading patterns or preferences.

A **Library Management System** solves these issues by:

- Analyzing users reading history, preferences, and behavior.
- Recommending books based on genres, authors, and past borrowing trends.
- Helping users discover new books aligned with their interests.
- Enhancing user satisfaction and library engagement through data-driven suggestions.
- Supporting librarians with insights into popular books and reading trends.

2. DATABASE DESIGN & IMPLEMENTATION

2.1 DATABASE CREATION & TABLES:

Database creation:

```
CREATE DATABASE libraryDB;
```

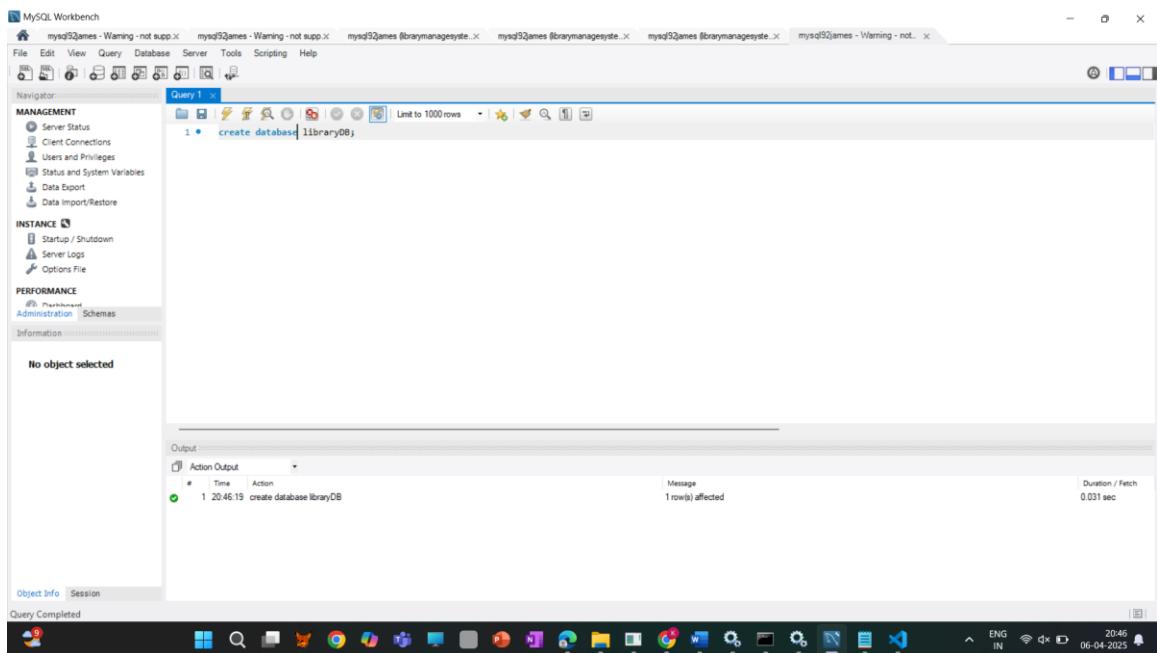


TABLE CREATION:

Genres table:

```
CREATE TABLE Genres (
    genre_id INT AUTO_INCREMENT PRIMARY KEY,
    genre_name VARCHAR(50) NOT NULL,
    description TEXT
);
```

Users table:

```
CREATE TABLE Users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    membership_date DATE,
    status VARCHAR(20)
);
```

Books table:

```
CREATE TABLE Books (
    book_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    author VARCHAR(100),
    genre_id INT,
```

```
year_published INT,  
isbn VARCHAR(20) UNIQUE,  
copies_available INT DEFAULT 1,  
FOREIGN KEY (genre_id) REFERENCES Genres(genre_id)  
);
```

Borrowing History table:

```
CREATE TABLE Borrowing_History (  
    borrow_id INT AUTO_INCREMENT PRIMARY KEY,  
    user_id INT,  
    book_id INT,  
    borrow_date DATE,  
    due_date DATE,  
    return_date DATE,  
    fine_amount DECIMAL(5,2) DEFAULT 0.00,  
    FOREIGN KEY (user_id) REFERENCES Users(user_id),  
    FOREIGN KEY (book_id) REFERENCES Books(book_id)  
);
```

Book Reviews table:

```
CREATE TABLE Book_Reviews (  
    review_id INT AUTO_INCREMENT PRIMARY KEY,  
    user_id INT,  
    book_id INT,  
    rating INT CHECK (rating BETWEEN 1 AND 5),  
    review_text TEXT,  
    review_date DATE,  
    FOREIGN KEY (user_id) REFERENCES Users(user_id),  
    FOREIGN KEY (book_id) REFERENCES Books(book_id)  
);
```

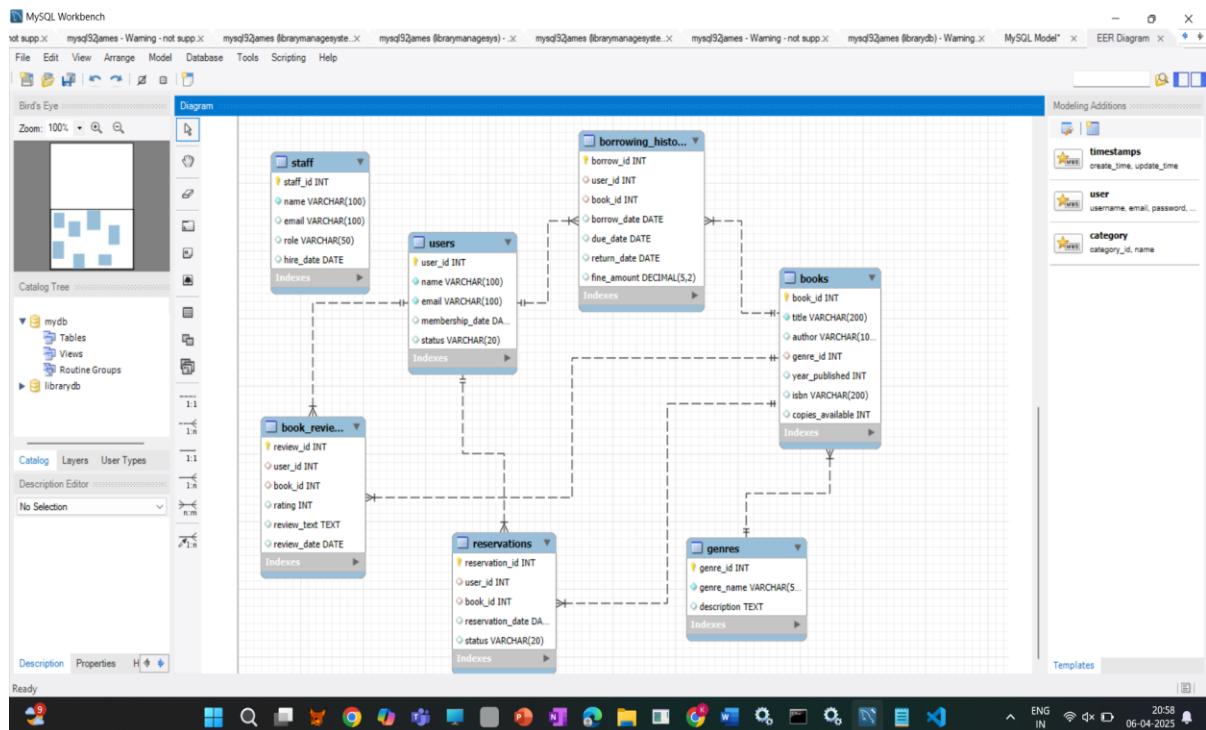
Staff table:

```
CREATE TABLE Staff (  
    staff_id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(100) NOT NULL,  
    email VARCHAR(100),  
    role VARCHAR(50),  
    hire_date DATE  
);
```

Reservations table:

```
CREATE TABLE Reservations (
    reservation_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    book_id INT,
    reservation_date DATE,
    status VARCHAR(20),
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (book_id) REFERENCES Books(book_id)
);
```

2.2 ER DIAGRAM (REVERSE ENGINEERED)



3. QUERIES FOR DATA MANAGEMENT

3.1 INSERT SAMPLE DATA:

GENRES TABLE:

```
INSERT INTO Genres (genre_name, description) VALUES
('Science Fiction', 'Fictional books based on futuristic science and technology'),
```

('Mystery', 'Books involving crime or puzzle-solving'),
('Romance', 'Books focusing on love stories'),
('History', 'Historical events and biographies'),
('Fantasy', 'Books involving magical or imaginary worlds');

USERS TABLE:

```
INSERT INTO Users (name, email, membership_date, status) VALUES  
('Alice Johnson', 'alice@gmail.com', '2023-01-15', 'Active'),  
('Bob Smith', 'bob@yahoo.com', '2022-11-22', 'Active'),  
('Clara White', 'clara@hotmail.com', '2023-05-03', 'Inactive'),  
('David Brown', 'david@gmail.com', '2021-08-30', 'Active'),  
('Ella Green', 'ella@rediffmail.com', '2024-02-14', 'Active');
```

BOOKS TABLE:

```
INSERT INTO Books (title, author, genre_id, year_published, isbn, copies_available)  
VALUES ('The Time Machine', 'H.G. Wells', 1, 1895, 'ISBN1234567890', 3),  
('Sherlock Holmes', 'Arthur Conan Doyle', 2, 1887, 'ISBN2345678901', 2),  
('Pride and Prejudice', 'Jane Austen', 3, 1813, 'ISBN3456789012', 4),  
('World War II', 'Winston Churchill', 4, 1948, 'ISBN4567890123', 1),  
('Harry Potter', 'J.K. Rowling', 5, 1997, 'ISBN5678901234', 5);
```

BORROWING HISTORY TABLE:

```
INSERT INTO Borrowing_History (user_id, book_id, borrow_date, due_date,  
return_date, fine_amount) VALUES  
(1, 1, '2024-01-01', '2024-01-15', '2024-01-14', 0.00),  
(2, 2, '2024-02-01', '2024-02-14', '2024-02-20', 15.00),  
(3, 3, '2024-03-05', '2024-03-19', NULL, 0.00),  
(4, 5, '2024-04-10', '2024-04-24', '2024-04-25', 5.00),  
(5, 4, '2024-03-01', '2024-03-15', '2024-03-12', 0.00);
```

BOOK REVIEWS TABLE:

```
INSERT INTO Book_Reviews (user_id, book_id, rating, review_text, review_date)
VALUES
(1, 1, 5, 'Amazing sci-fi classic!', '2024-01-16'),
(2, 2, 4, 'Intriguing mystery and great deduction.', '2024-02-21'),
(3, 3, 3, 'Romantic but a bit slow.', '2024-03-20'),
(4, 5, 5, 'Magical and thrilling.', '2024-04-26'),
(5, 4, 4, 'Informative and detailed.', '2024-03-13');
```

STAFF TABLE:

```
INSERT INTO Staff (name, email, role, hire_date) VALUES
('Ravi Kumar', 'ravi@library.com', 'Librarian', '2020-05-10'),
('Sneha Sharma', 'sneha@library.com', 'Assistant Librarian', '2021-07-15'),
('Anil Mehta', 'anil@library.com', 'Technician', '2022-01-20'),
('Priya Das', 'priya@library.com', 'Data Entry', '2023-03-01'),
('Mohit Verma', 'mohit@library.com', 'Clerk', '2024-01-12');
```

RESERVATIONS TABLE:

```
INSERT INTO Reservations (user_id, book_id, reservation_date, status) VALUES
(1, 2, '2024-04-01', 'Pending'),
(2, 1, '2024-04-02', 'Completed'),
(3, 5, '2024-04-03', 'Cancelled'),
(4, 3, '2024-04-04', 'Pending'),
(5, 4, '2024-04-05', 'Completed');
```

3.2 RETRIEVAL QUERIES:

1. Books by Genre

```
SELECT b.title, b.author, g.genre_name
FROM Books b
JOIN Genres g ON b.genre_id = g.genre_id
WHERE g.genre_name = 'Science Fiction';
```

2. Books Borrowed by a Specific User

```
SELECT u.name, b.title, bh.borrow_date, bh.due_date  
FROM Borrowing_History bh  
JOIN Users u ON bh.user_id = u.user_id  
JOIN Books b ON bh.book_id = b.book_id  
WHERE u.name = 'Alice Johnson';
```

3. List All Available Books

```
SELECT title, author, copies_available  
FROM Books  
WHERE copies_available > 0;
```

4. Books with Average Rating

```
SELECT b.title, AVG(br.rating) AS average_rating  
FROM Books b  
JOIN Book_Reviews br ON b.book_id = br.book_id  
GROUP BY b.title  
ORDER BY average_rating DESC;
```

5. Reservations for a Book

```
SELECT r.reservation_id, u.name AS user_name, b.title,  
r.reservation_date, r.status  
FROM Reservations r  
JOIN Users u ON r.user_id = u.user_id  
JOIN Books b ON r.book_id = b.book_id  
WHERE b.title = 'The Time Machine';
```

6. Overdue Books

```
SELECT u.name, b.title, bh.due_date, bh.return_date  
FROM Borrowing_History bh  
JOIN Users u ON bh.user_id = u.user_id
```

```

JOIN Books b ON bh.book_id = b.book_id
WHERE bh.return_date IS NULL AND bh.due_date < CURDATE();

```

7. Users with Outstanding Fines

```

SELECT u.name, SUM(bh.fine_amount) AS total_fines
FROM Borrowing_History bh
JOIN Users u ON bh.user_id = u.user_id
WHERE bh.fine_amount > 0
GROUP BY u.user_id;

```

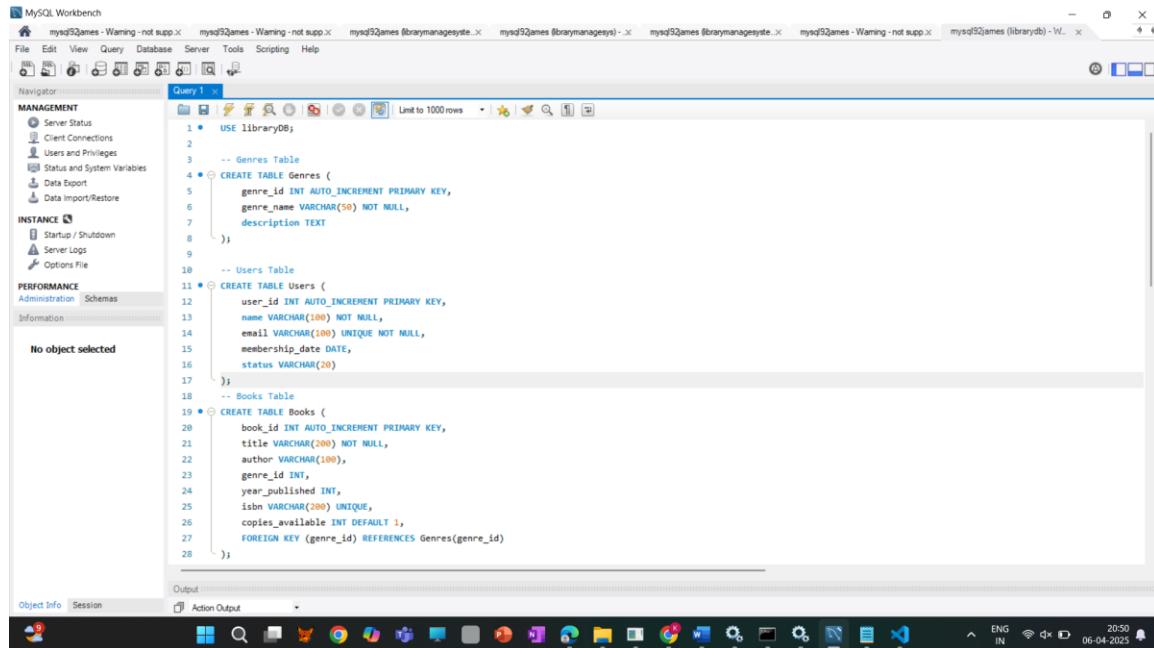
4. IMPLEMENTATION & RESULTS

4.1 EXECUTION ENVIRONMENT

The implementation was executed using MySQL Workbench 9.2 on Windows 11.

4.2 SCREENSHOTS OF EXECUTION RESULTS

1.Table Creation:



The screenshot shows the MySQL Workbench interface with the Query Editor tab active. The code in the editor creates three tables: Genres, Users, and Books. The Genres table has a primary key 'genre_id' and a unique constraint on 'genre_name'. The Users table has a primary key 'user_id' and unique constraints on 'name' and 'email'. The Books table has a primary key 'book_id' and a foreign key 'genre_id' referencing the 'genre_id' in the Genres table.

```

USE libraryDB;

-- Genres Table
CREATE TABLE Genres (
    genre_id INT AUTO_INCREMENT PRIMARY KEY,
    genre_name VARCHAR(50) NOT NULL,
    description TEXT
);

-- Users Table
CREATE TABLE Users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    membership_date DATE,
    status VARCHAR(20)
);

-- Books Table
CREATE TABLE Books (
    book_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    author VARCHAR(100),
    genre_id INT,
    year_published INT,
    isbn VARCHAR(200) UNIQUE,
    copies_available INT DEFAULT 1,
    FOREIGN KEY (genre_id) REFERENCES Genres(genre_id)
);

```

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: Query 1

```

29
30    -- Borrowing_History Table
31  • CREATE TABLE Borrowing_History (
32      borrow_id INT AUTO_INCREMENT PRIMARY KEY,
33      user_id INT,
34      book_id INT,
35      borrow_date DATE,
36      due_date DATE,
37      return_date DATE,
38      fine_amount DECIMAL(5,2) DEFAULT 0.00,
39      FOREIGN KEY (user_id) REFERENCES Users(user_id),
40      FOREIGN KEY (book_id) REFERENCES Books(book_id)
41  );
42
43    -- Book_Reviews Table
44  • CREATE TABLE Book_Reviews (
45      review_id INT AUTO_INCREMENT PRIMARY KEY,
46      user_id INT,
47      book_id INT,
48      rating INT CHECK (rating BETWEEN 1 AND 5),
49      review_text TEXT,
50      review_date DATE,
51      FOREIGN KEY (user_id) REFERENCES Users(user_id),
52      FOREIGN KEY (book_id) REFERENCES Books(book_id)
53  );
54
55
56

```

Object Info Session Action Output

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: Query 1

```

51
52    FOREIGN KEY (user_id) REFERENCES Users(user_id),
53    FOREIGN KEY (book_id) REFERENCES Books(book_id)
54
55
56
57    -- Staff Table
58  • CREATE TABLE Staff (
59      staff_id INT AUTO_INCREMENT PRIMARY KEY,
60      name VARCHAR(100) NOT NULL,
61      email VARCHAR(100),
62      role VARCHAR(50),
63      hire_date DATE
64  );
65
66
67
68    -- Reservations Table
69  • CREATE TABLE Reservations (
70      reservation_id INT AUTO_INCREMENT PRIMARY KEY,
71      user_id INT,
72      book_id INT,
73      reservation_date DATE,
74      status VARCHAR(20),
75      FOREIGN KEY (user_id) REFERENCES Users(user_id),
76      FOREIGN KEY (book_id) REFERENCES Books(book_id)
77  );
78

```

Object Info Session Action Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	20:49:57	USE libraryDB	0 row(s) affected	0.000 sec
2	20:49:57	CREATE TABLE Genres (genre_id INT AUTO_INCREMENT PRIMARY KEY, genre_name VARCHAR(50))	0 row(s) affected	0.047 sec
3	20:49:57	CREATE TABLE Users (user_id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, email VARCHAR(100), role VARCHAR(50), hire_date DATE)	0 row(s) affected	0.047 sec
4	20:49:57	CREATE TABLE Books (book_id INT AUTO_INCREMENT PRIMARY KEY, title VARCHAR(200) NOT NULL, author VARCHAR(100), publication_date DATE)	0 row(s) affected	0.047 sec
5	20:49:57	CREATE TABLE Borrowing_History (borrow_id INT AUTO_INCREMENT PRIMARY KEY, user_id INT, book_id INT, borrow_date DATE, due_date DATE, return_date DATE, fine_amount DECIMAL(5,2) DEFAULT 0.00, FOREIGN KEY (user_id) REFERENCES Users(user_id), FOREIGN KEY (book_id) REFERENCES Books(book_id))	0 row(s) affected	0.062 sec
6	20:49:57	CREATE TABLE Book_Reviews (review_id INT AUTO_INCREMENT PRIMARY KEY, user_id INT, book_id INT, rating INT CHECK (rating BETWEEN 1 AND 5), review_text TEXT, review_date DATE, FOREIGN KEY (user_id) REFERENCES Users(user_id), FOREIGN KEY (book_id) REFERENCES Books(book_id))	0 row(s) affected	0.047 sec
7	20:49:57	CREATE TABLE Staff (staff_id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, email VARCHAR(100), role VARCHAR(50), hire_date DATE)	0 row(s) affected	0.047 sec
8	20:49:58	CREATE TABLE Reservations (reservation_id INT AUTO_INCREMENT PRIMARY KEY, user_id INT, book_id INT, borrow_id INT, status VARCHAR(20))	0 row(s) affected	0.047 sec

2. Sample Data Insertion:

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator
Query 1
1 • use libraryDB;
2
3 • INSERT INTO Genres (genre_name, description) VALUES
4 ('Science Fiction', 'Fictional books based on futuristic science and technology'),
5 ('Mystery', 'Books involving crime or puzzle-solving'),
6 ('Romance', 'Books focusing on love stories'),
7 ('History', 'Historical events and biographies'),
8 ('Fantasy', 'Books involving magical or imaginary worlds')
9
10 • INSERT INTO Users (name, email, membership_date, status) VALUES
11 ('Alice Johnson', 'alice@gmail.com', '2023-01-15', 'Active'),
12 ('Bob Smith', 'bob@yahoo.com', '2022-11-22', 'Active'),
13 ('Clara White', 'clara@hotmail.com', '2023-05-03', 'Inactive'),
14 ('David Brown', 'david@gmail.com', '2021-09-30', 'Active'),
15 ('Ella Green', 'ella@rediffmail.com', '2024-02-14', 'Active')
16
17 • INSERT INTO Books (title, author, genre_id, year_published, isbn, copies_available) VALUES
18 ('The Time Machine', 'H.G. Wells', 1, 1895, 'ISBN1234567890', 3),
19 ('Sherlock Holmes', 'Arthur Conan Doyle', 2, 1890, 'ISBN12345678901', 2),
20 ('Pride and Prejudice', 'Jane Austen', 3, 1813, 'ISBN123456789012', 4),
21 ('World War II', 'Winston Churchill', 4, 1948, 'ISBN1234567890123', 1),
22 ('Harry Potter', 'J.K. Rowling', 5, 1997, 'ISBN12345678901234', 5)
23
24 • INSERT INTO Borrowing_History (user_id, book_id, borrow_date, due_date, return_date, fine_amount) VALUES
25 (1, 1, '2024-01-15', '2024-01-14', 0.00),
26 (2, 2, '2024-02-01', '2024-02-20', 15.00),
27 (3, 3, '2024-03-05', '2024-03-19', NULL, 0.00),
28 (4, 5, '2024-04-10', '2024-04-24', '2024-04-25', 5.00),
29 (5, 4, '2024-03-01', '2024-03-15', '2024-03-12', 0.00)
30
31
32 • INSERT INTO Book_Reviews (user_id, book_id, rating, review_text, review_date) VALUES
33 (1, 1, 5, 'Amazing sci-fi classic!', '2024-01-16'),
34 (2, 2, 4, 'Intriguing mystery and great deduction.', '2024-02-21'),
35 (3, 3, 3, 'Romantic but a bit slow.', '2024-03-20'),
36 (4, 5, 5, 'Magical and thrilling.', '2024-04-26'),
37 (5, 4, 4, 'Informative and detailed.', '2024-03-13')
38
39 • INSERT INTO Staff (name, email, role, hire_date) VALUES
40 ('Ravi Kumar', 'ravi@library.com', 'Librarian', '2020-05-10'),
41 ('Sneha Sharma', 'sneha@library.com', 'Assistant Librarian', '2021-07-15'),
42 ('Anil Mehta', 'anil@library.com', 'Technician', '2022-01-20'),
43 ('Priya Das', 'priya@library.com', 'Data Entry', '2023-03-01'),
44 ('Mohit Verma', 'mohit@library.com', 'Clerk', '2024-01-12')
45
46 • INSERT INTO Reservations (user_id, book_id, reservation_date, status) VALUES
47 (1, 2, '2024-04-01', 'Pending'),
48 (2, 1, '2024-04-02', 'Completed'),
49 (3, 5, '2024-04-03', 'Cancelled'),
50 (4, 3, '2024-04-04', 'Pending'),
51 (5, 4, '2024-04-05', 'Completed')
52

```

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator
Query 1
24 • INSERT INTO Borrowing_History (user_id, book_id, borrow_date, due_date, return_date, fine_amount) VALUES
25 (1, 1, '2024-01-01', '2024-01-15', '2024-01-14', 0.00),
26 (2, 2, '2024-02-01', '2024-02-14', '2024-02-20', 15.00),
27 (3, 3, '2024-03-05', '2024-03-19', NULL, 0.00),
28 (4, 5, '2024-04-10', '2024-04-24', '2024-04-25', 5.00),
29 (5, 4, '2024-03-01', '2024-03-15', '2024-03-12', 0.00)
30
31
32 • INSERT INTO Book_Reviews (user_id, book_id, rating, review_text, review_date) VALUES
33 (1, 1, 5, 'Amazing sci-fi classic!', '2024-01-16'),
34 (2, 2, 4, 'Intriguing mystery and great deduction.', '2024-02-21'),
35 (3, 3, 3, 'Romantic but a bit slow.', '2024-03-20'),
36 (4, 5, 5, 'Magical and thrilling.', '2024-04-26'),
37 (5, 4, 4, 'Informative and detailed.', '2024-03-13')
38
39 • INSERT INTO Staff (name, email, role, hire_date) VALUES
40 ('Ravi Kumar', 'ravi@library.com', 'Librarian', '2020-05-10'),
41 ('Sneha Sharma', 'sneha@library.com', 'Assistant Librarian', '2021-07-15'),
42 ('Anil Mehta', 'anil@library.com', 'Technician', '2022-01-20'),
43 ('Priya Das', 'priya@library.com', 'Data Entry', '2023-03-01'),
44 ('Mohit Verma', 'mohit@library.com', 'Clerk', '2024-01-12')
45
46 • INSERT INTO Reservations (user_id, book_id, reservation_date, status) VALUES
47 (1, 2, '2024-04-01', 'Pending'),
48 (2, 1, '2024-04-02', 'Completed'),
49 (3, 5, '2024-04-03', 'Cancelled'),
50 (4, 3, '2024-04-04', 'Pending'),
51 (5, 4, '2024-04-05', 'Completed')
52

```

#	Time	Action	Message	Duration / Fetch
1	21:04:05	use libraryDB	0 row(s) affected	0.000 sec
2	21:04:05	INSERT INTO Genres (genre_name, description) VALUES (Science Fiction', 'Fictional books based on futuristic sci...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
3	21:04:05	INSERT INTO Users (name, email, membership_date, status) VALUES (Alice Johnson', 'alice@gmail.com', '2023-01-01', ...)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
4	21:04:05	INSERT INTO Books (title, author, genre_id, year_published, isbn, copies_available) VALUES (The Time Machine', ...)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec
5	21:04:05	INSERT INTO Borrowing_History (user_id, book_id, borrow_date, due_date, return_date, fine_amount) VALUES (1, ...)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.015 sec
6	21:04:05	INSERT INTO Book_Reviews (user_id, book_id, rating, review_text, review_date) VALUES (1, 1, 5, 'Amazing sci fi cl...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec
7	21:04:05	INSERT INTO Staff (name, email, role, hire_date) VALUES (Ravi Kumar', 'ravi@library.com', 'Librarian', '2020-05-10')	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec
8	21:04:05	INSERT INTO Reservations (user_id, book_id, reservation_date, status) VALUES (1, 2, '2024-04-01', 'Pending')	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec

3.Data Retrieval Queries:

1. Books by genre:

The screenshot shows the MySQL Workbench interface. In the top-left corner, there are two tabs: "mysql92games - Warning - not supp..." and "mysql92games (librarydb) - W...". The left sidebar contains sections for MANAGEMENT, INSTANCE, PERFORMANCE, and Administration. Under Administration, the Schemas tab is selected. The main area has a "Query 1" tab open with the following SQL code:

```
1 • use libraryDB;
2
3 • SELECT b.title, b.author, g.genre_name
4   FROM Books b
5   JOIN Genres g ON b.genre_id = g.genre_id
6 WHERE g.genre_name = 'Science Fiction';
7
```

The "Result Grid" shows the results of the query:

title	author	genre_name
The Time Machine	H.G. Wells	Science Fiction

The "Result 1" tab shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	21:09:04	use libraryDB	0 row(s) affected	0.000 sec
2	21:09:04	SELECT b.title, b.author, g.genre_name FROM Books b JOIN Genres g ON b.genre_id = g.genre_id WHERE g....	1 row(s) returned	0.000 sec / 0.000 sec

2. Books Borrowed by a Specific User:

The screenshot shows the MySQL Workbench interface. In the top-left corner, there are two tabs: "mysql92games - Warning - not supp..." and "mysql92games (librarydb) - W...". The left sidebar contains sections for MANAGEMENT, INSTANCE, PERFORMANCE, and Administration. Under Administration, the Schemas tab is selected. The main area has a "Query 1" tab open with the following SQL code:

```
1 • use libraryDB;
2
3 • SELECT u.name, b.title, bh.borrow_date, bh.due_date
4   FROM Borrowing_History bh
5   JOIN Users u ON bh.user_id = u.user_id
6   JOIN Books b ON bh.book_id = b.book_id
7 WHERE u.name = 'Alice Johnson';
8
```

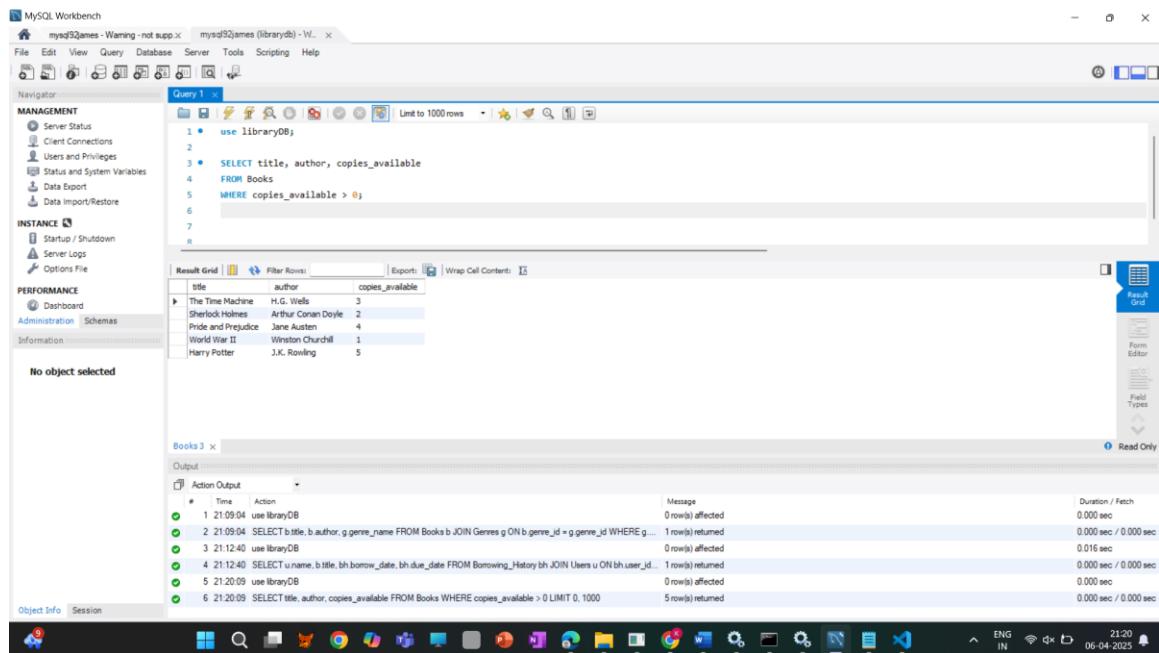
The "Result Grid" shows the results of the query:

name	title	borrow_date	due_date
Alice Johnson	The Time Machine	2024-01-01	2024-01-15

The "Result 2" tab shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	21:09:04	use libraryDB	0 row(s) affected	0.000 sec
2	21:09:04	SELECT u.name, b.title, bh.borrow_date, bh.due_date FROM Borrowing_History bh JOIN Users u ON bh.user_id = u.user_id WHERE u....	1 row(s) returned	0.000 sec / 0.000 sec
3	21:12:40	use libraryDB	0 row(s) affected	0.016 sec
4	21:12:40	SELECT u.name, b.title, bh.borrow_date, bh.due_date FROM Borrowing_History bh JOIN Users u ON bh.user_id = u.user_id WHERE u....	1 row(s) returned	0.000 sec / 0.000 sec

3. List all Available books:



```

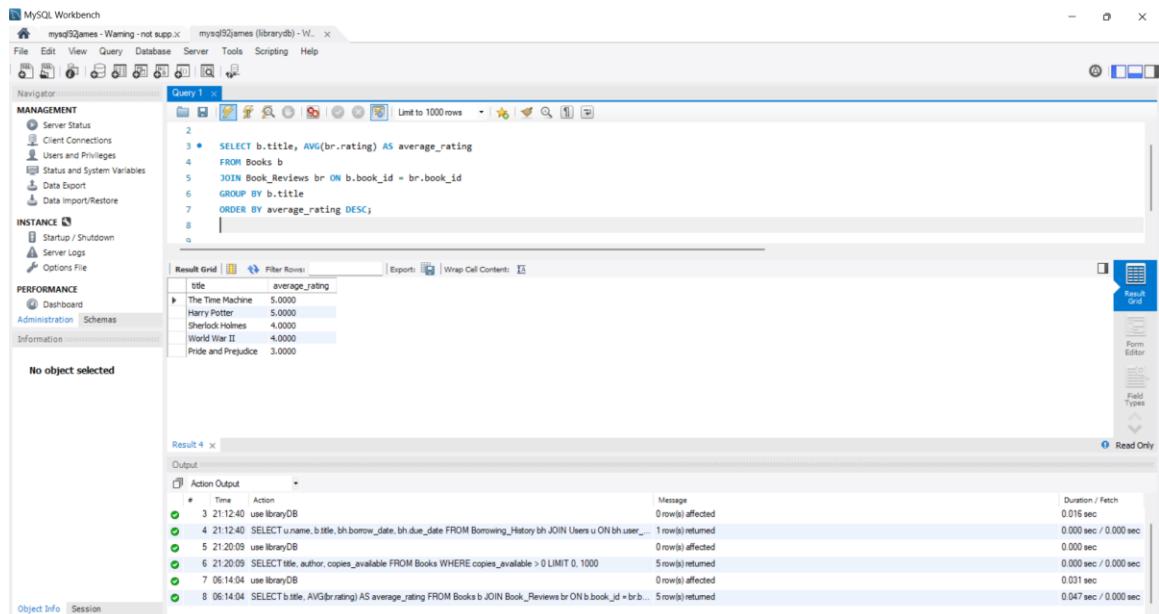
use libraryDB;
SELECT title, author, copies_available
FROM Books
WHERE copies_available > 0;

```

The screenshot shows the MySQL Workbench interface. The 'Query Grid' pane displays the results of the SQL query. The results table has three columns: title, author, and copies_available. The data shows five books with their respective authors and available copy counts.

title	author	copies_available
The Time Machine	H.G. Wells	3
Sherlock Holmes	Arthur Conan Doyle	2
Pride and Prejudice	Jane Austen	4
World War II	Winston Churchill	1
Harry Potter	J.K. Rowling	5

4. Books with Average Rating



```

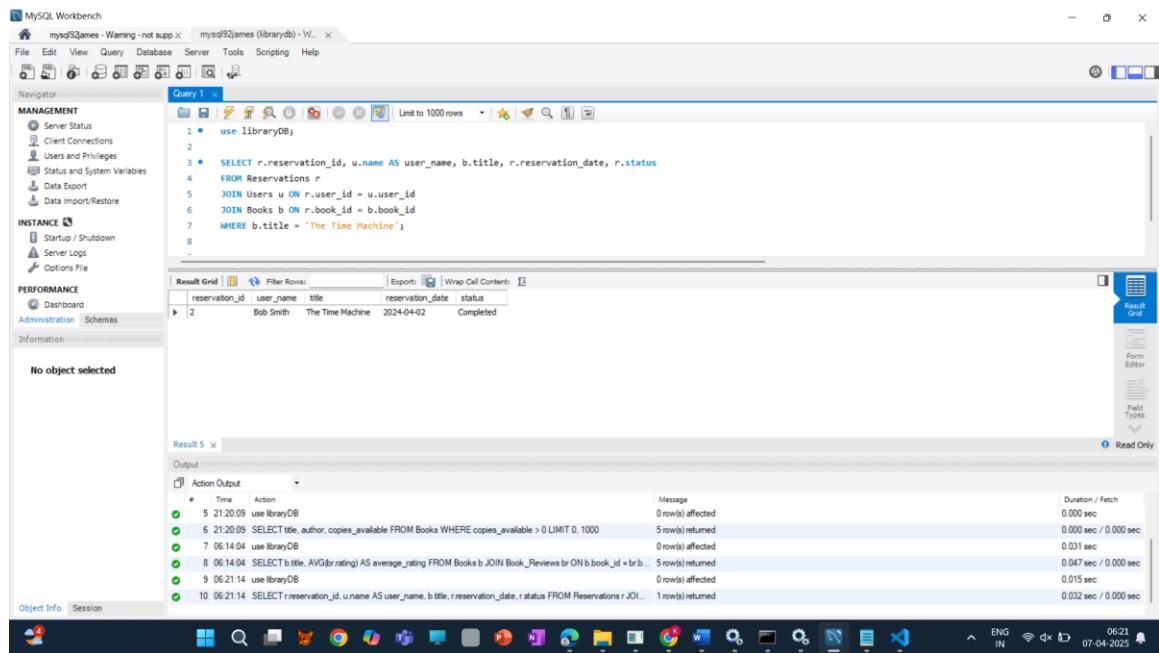
SELECT b.title, AVG(br.rating) AS average_rating
FROM Books b
JOIN Book_Reviews br ON b.book_id = br.book_id
GROUP BY b.title
ORDER BY average_rating DESC;

```

The screenshot shows the MySQL Workbench interface. The 'Query Grid' pane displays the results of the SQL query. The results table has two columns: title and average_rating. The data shows five books with their average ratings.

title	average_rating
The Time Machine	5.0000
Harry Potter	5.0000
Sherlock Holmes	4.0000
World War II	4.0000
Pride and Prejudice	3.0000

5. Reservations for a Book:



The screenshot shows the MySQL Workbench interface with a query editor and results grid.

Query Editor:

```
1 • use libraryDB;
2
3 •   SELECT r.reservation_id, u.name AS user_name, b.title, r.reservation_date, r.status
4   FROM Reservations r
5   JOIN Users u ON r.user_id = u.user_id
6   JOIN Books b ON r.book_id = b.book_id
7   WHERE b.title = 'The Time Machine';
8
```

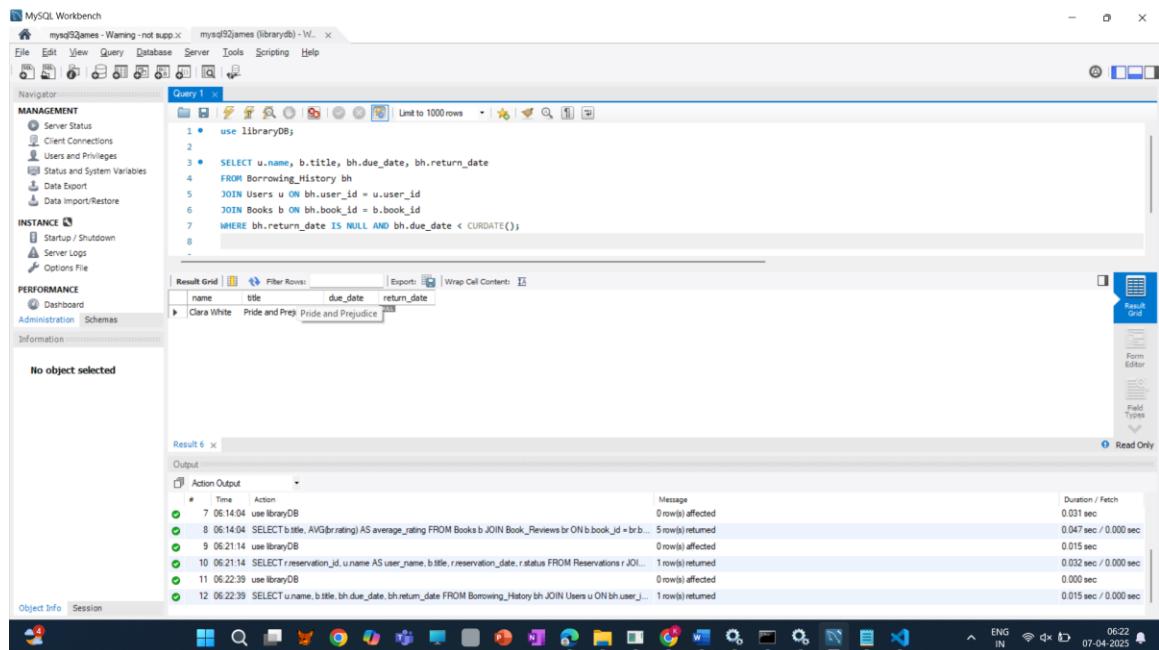
Result Grid:

reservation_id	user_name	title	reservation_date	status
2	Bob Smith	The Time Machine	2024-04-02	Completed

Action Output:

#	Time	Action	Message	Duration / Fetch
1	5 21:20:09	use libraryDB	0 row(s) affected	0.000 sec
2	6 21:20:09	SELECT title, author, copies_available FROM Books WHERE copies_available > 0 LIMIT 0, 1000	5 rows returned	0.000 sec / 0.000 sec
3	7 06:14:04	use libraryDB	0 row(s) affected	0.031 sec
4	8 06:14:04	SELECT b.title, AVG(br.rating) AS average_rating FROM Books b JOIN Book_Reviews br ON b.book_id = br.book_id GROUP BY b.title	5 rows returned	0.047 sec / 0.000 sec
5	9 06:21:14	use libraryDB	0 row(s) affected	0.015 sec
6	10 06:21:14	SELECT r.reservation_id, u.name AS user_name, b.title, r.reservation_date, r.status FROM Reservations r JOIN Users u ON r.user_id = u.user_id JOIN Books b ON r.book_id = b.book_id WHERE b.title = 'The Time Machine'	1 row(s) returned	0.032 sec / 0.000 sec

6. Overdue Books:



The screenshot shows the MySQL Workbench interface with a query editor and results grid.

Query Editor:

```
1 • use libraryDB;
2
3 •   SELECT u.name, b.title, bh.due_date, bh.return_date
4   FROM Borrowing_History bh
5   JOIN Users u ON bh.user_id = u.user_id
6   JOIN Books b ON bh.book_id = b.book_id
7   WHERE bh.return_date IS NULL AND bh.due_date < CURDATE();
8
```

Result Grid:

name	title	due_date	return_date
Clara White	Pride and Prejudice	2024-04-05	

Action Output:

#	Time	Action	Message	Duration / Fetch
1	7 06:14:04	use libraryDB	0 row(s) affected	0.031 sec
2	8 06:14:04	SELECT b.title, AVG(br.rating) AS average_rating FROM Books b JOIN Book_Reviews br ON b.book_id = br.book_id GROUP BY b.title	5 rows returned	0.047 sec / 0.000 sec
3	9 06:21:14	use libraryDB	0 row(s) affected	0.015 sec
4	10 06:21:14	SELECT r.reservation_id, u.name AS user_name, b.title, r.reservation_date, r.status FROM Reservations r JOIN Users u ON r.user_id = u.user_id JOIN Books b ON r.book_id = b.book_id WHERE b.title = 'The Time Machine'	1 row(s) returned	0.032 sec / 0.000 sec
5	11 06:22:39	use libraryDB	0 row(s) affected	0.000 sec
6	12 06:22:39	SELECT u.name, b.title, bh.due_date, bh.return_date FROM Borrowing_History bh JOIN Users u ON bh.user_id = u.user_id JOIN Books b ON bh.book_id = b.book_id WHERE bh.return_date IS NULL AND bh.due_date < CURDATE()	1 row(s) returned	0.015 sec / 0.000 sec

7. User with outstanding fines:

The screenshot shows the MySQL Workbench interface. In the top-left pane, the 'MANAGEMENT' section is visible with various database management options like Server Status, Client Connections, and Data Export. The 'INSTANCE' section shows startup/shutdown status and server logs. The 'PERFORMANCE' section includes a dashboard and administration tools. The bottom-left pane says 'No object selected'. The main area has a 'Query 1' tab open with the following SQL query:

```
1 • use libraryDB;
2
3 • SELECT u.name, SUM(bh.fine_amount) AS total_fines
4   FROM Borrowing_History bh
5   JOIN Users u ON bh.user_id = u.user_id
6   WHERE bh.fine_amount > 0
7   GROUP BY u.user_id;
8
```

The 'Result Grid' shows the results:

name	total_fines
Bob Smith	15.00
David Brown	5.00

The 'Result 7' tab shows the execution history:

#	Time	Action	Message	Duration / Fetch
9	06:21:14	use libraryDB	0 row(s) affected	0.015 sec
10	06:21:14	SELECT reservation_id, u.name AS user_name, b.title, r.reservation_date, r.status FROM Reservations r JOIN Users u ON r.user_id = u.user_id JOIN Books b ON r.book_id = b.id JOIN Book_Statuses rs ON r.status = rs.id	1 row(s) returned	0.032 sec / 0.000 sec
11	06:22:39	use libraryDB	0 row(s) affected	0.000 sec
12	06:22:39	SELECT u.name, b.title, bh.due_date, bh.return_date FROM Borrowing_History bh JOIN Users u ON bh.user_id = u.user_id JOIN Books b ON bh.book_id = b.id	1 row(s) returned	0.015 sec / 0.000 sec
13	06:23:25	use libraryDB	0 row(s) affected	0.000 sec
14	06:23:25	SELECT u.name, SUM(bh.fine_amount) AS total_fines FROM Borrowing_History bh JOIN Users u ON bh.user_id = u.user_id GROUP BY u.user_id	2 row(s) returned	0.015 sec / 0.000 sec

4. Select Queries:

GENRES TABLE:

The screenshot shows the MySQL Workbench interface. The 'MANAGEMENT' and 'INSTANCE' sections are visible. The bottom-left pane says 'No object selected'. The main area has a 'Query 1' tab open with the following SQL query:

```
1 • use libraryDB;
2
3 • select*from genres;
4
```

The 'Result Grid' shows the results:

genre_id	genre_name	description
1	Science Fiction	Fictional books based on futuristic science and technology.
2	Mystery	Books involving crime or puzzle-solving.
3	Romance	Books focusing on love stories.
4	History	Historical events and biographies.
5	Fantasy	Books involving magical or imaginary worlds.

The 'genres 1' tab shows the execution history:

#	Time	Action	Message	Duration / Fetch
1	10:22:29	use libraryDB	0 row(s) affected	0.000 sec
2	10:22:29	select*from genres LIMIT 0, 1000	5 row(s) returned	0.015 sec / 0.000 sec

STAFF TABLE:

The screenshot shows the MySQL Workbench interface with the 'staff' table selected. The table has columns: staff_id, name, email, role, and hire_date. The data is as follows:

staff_id	name	email	role	hire_date
1	Ravi Kumar	ravi@library.com	Librarian	2020-05-10
2	Sneha Sharma	sneha@library.com	Assistant Librarian	2021-07-15
3	Anil Mehta	anil@library.com	Technician	2022-01-20
4	Priya Das	priya@library.com	Data Entry	2023-03-01
5	Mohit Verma	mohit@library.com	Clerk	2024-01-12

The 'Output' pane shows the execution history of the query:

#	Time	Action	Message	Duration / Fetch
1	10:22:29	use libraryDB	0 row(s) affected	0.000 sec
2	10:22:29	select*from staff	5 row(s) returned	0.015 sec / 0.000 sec
3	10:24:45	use libraryDB	0 row(s) affected	0.000 sec
4	10:24:45	select*from staff	5 row(s) returned	0.015 sec / 0.000 sec

USERS TABLE:

The screenshot shows the MySQL Workbench interface with the 'users' table selected. The table has columns: user_id, name, email, membership_date, and status. The data is as follows:

user_id	name	email	membership_date	status
1	Alice Johnson	alice@gmail.com	2023-01-15	Active
2	Bob Smith	bob@yahoo.com	2022-11-22	Active
3	Clara White	clara@hotmail.com	2023-05-03	Inactive
4	David Brown	david@gmail.com	2021-08-30	Active
5	Ella Green	ella@edffmail.com	2024-02-14	Active

The 'Output' pane shows the execution history of the query:

#	Time	Action	Message	Duration / Fetch
1	10:22:29	use libraryDB	0 row(s) affected	0.000 sec
2	10:22:29	select*from users	5 row(s) returned	0.015 sec / 0.000 sec
3	10:24:45	use libraryDB	0 row(s) affected	0.000 sec
4	10:24:45	select*from staff	5 row(s) returned	0.015 sec / 0.000 sec
5	10:25:31	use libraryDB	0 row(s) affected	0.000 sec
6	10:25:31	select*from users	5 row(s) returned	0.000 sec / 0.000 sec

BORROWING_HISTORY TABLE:

The screenshot shows the MySQL Workbench interface with the 'borrowing_history' table selected. The table has columns: borrow_id, user_id, book_id, borrow_date, due_date, return_date, and fine_amount. The data grid displays seven rows of borrowing records. The 'Output' pane shows the execution history for the query.

borrow_id	user_id	book_id	borrow_date	due_date	return_date	fine_amount
1	1	1	2024-01-01	2024-01-15	2024-01-14	0.00
2	2	2	2024-02-01	2024-02-14	2024-02-20	15.00
3	3	3	2024-03-05	2024-03-19		0.00
4	4	5	2024-04-10	2024-04-24	2024-04-25	5.00
5	5	4	2024-03-01	2024-03-15	2024-03-12	0.00
6	6	6	2024-04-15	2024-04-29		0.00
7	7	7	2024-05-01	2024-05-15		0.00

Action Output:

- # 3 10:24:45 use libraryDB Message: 0 row(s) affected Duration / Fetch: 0.000 sec
- # 4 10:24:45 select*from borrowing_history Message: 5 row(s) returned 0.015 sec / 0.000 sec
- # 5 10:25:31 use libraryDB Message: 0 row(s) affected 0.000 sec
- # 6 10:25:31 select*from users LIMIT 0, 1000 Message: 5 row(s) returned 0.000 sec / 0.000 sec
- # 7 10:27:29 use libraryDB Message: 0 row(s) affected 0.000 sec
- # 8 10:27:29 select*from borrowing_history LIMIT 0, 1000 Message: 5 row(s) returned 0.000 sec / 0.000 sec

BORROWING_REVIEWS TABLE:

The screenshot shows the MySQL Workbench interface with the 'book_reviews' table selected. The table has columns: review_id, user_id, book_id, rating, review_text, and review_date. The data grid displays five rows of review records. The 'Output' pane shows the execution history for the query.

review_id	user_id	book_id	rating	review_text	review_date
1	1	1	5	Amazing sci-fi classic!	2024-01-16
2	2	2	4	Intriguing mystery and great deduction.	2024-02-21
3	3	3	3	Romantic but a bit slow.	2024-03-20
4	4	5	5	Magical and thrilling.	2024-04-26
5	5	4	4	Informative and detailed.	2024-03-13

Action Output:

- # 5 10:25:31 use libraryDB Message: 0 row(s) affected Duration / Fetch: 0.000 sec
- # 6 10:25:31 select*from users LIMIT 0, 1000 Message: 5 row(s) returned 0.000 sec / 0.000 sec
- # 7 10:27:29 use libraryDB Message: 0 row(s) affected 0.000 sec
- # 8 10:27:29 select*from borrowing_history LIMIT 0, 1000 Message: 5 row(s) returned 0.000 sec / 0.000 sec
- # 9 10:28:22 use libraryDB Message: 0 row(s) affected 0.000 sec
- # 10 10:28:22 select*from book_reviews LIMIT 0, 1000 Message: 5 row(s) returned 0.000 sec / 0.000 sec

RESERVATIONS TABLE:

The screenshot shows the MySQL Workbench interface with the 'reservations' table selected. The table has columns: reservation_id, user_id, book_id, reservation_date, and status. The data shows five rows of reservations.

reservation_id	user_id	book_id	reservation_date	status
1	1	2	2024-04-01	Pending
2	2	4	2024-04-02	Canceled
3	3	5	2024-04-03	Cancelled
4	4	3	2024-04-04	Pending
5	5	4	2024-04-05	Completed

The 'Output' pane shows the history of actions performed on the database, including SELECT and USE statements.

BOOKS TABLE:

The screenshot shows the MySQL Workbench interface with the 'books' table selected. The table has columns: book_id, title, author, genre_id, year_published, isbn, and copies_available. The data shows five books.

book_id	title	author	genre_id	year_published	isbn	copies_available
1	The Time Machine	H.G. Wells	1	1895	ISBN1234567890	3
2	Sherlock Holmes	Arthur Conan Doyle	2	1887	ISBN2345678901	2
3	Pride and Prejudice	Jane Austen	3	1813	ISBN3456789012	4
4	World War II	Winston Churchill	4	1948	ISBN4567890123	1
5	Harry Potter	J.K. Rowling	5	1997	ISBN5678901234	5

The 'Output' pane shows the history of actions performed on the database, including SELECT and USE statements.

5. GITHUB REPOSITORY

5.1 REPOSITORY LINK

<https://github.com/kalai02082005/LibraryManagementsystem.git>

5.2 uploaded FILES IN REPOSITORY

The following files are included in the repository:

- **database_schema.sql** – SQL scripts for creating all required tables (Users, Books, Genres, Borrowing_History, Book_Reviews, Staff, Reservations).
- **insert_sample_data.sql** – SQL scripts for inserting sample records into each table.
- **retrieval_queries.sql** – SQL scripts for retrieving data such as books by genre, available books, borrowed books, top-rated books, etc.
- **ER_Diagram.png** – Entity-Relationship Diagram showing relationships between tables.
- **project_report.docx** – Final formatted documentation report (can be in .docx or .pdf).
- **screenshots/** – Folder containing screenshots of executed SQL queries and output from MySQL Workbench or any DBMS used.

6. Conclusion

The Library Management System simplifies library operations and enhances user engagement through personalized book suggestions. With a well-structured database and efficient retrieval queries, it ensures smooth management of users, books, and transactions. This project demonstrates effective use of SQL and ER modeling, laying the groundwork for future enhancements like smart recommendations and web integration.