DBMS MINI PROJECT
ON

Title: Library Management System
Group No.: 01

Submitted By:

Rohit Agarwal(23CS8001)
Sandipto Roy(23CS8002)
Prachi Yadav(23CS8003)
Pranav Verma(23CS8004)
Siddardha Reddy(23CS8005)

Date of submission: 8th April 2025 **Teacher's signature:**

TABLE OF CONTENTS:

Acknowledgment

We extend our sincere and heartfelt thanks to our esteemed guide Mr. Prasenjit sir and the Department of Computer Science and Engineering for providing us with the right guidance and their support.We would also like to thank our peers and friends who helped us during this project.

ABSTRACT

The Library Management System is a software application developed to support librarians in efficiently managing the operations of a university library. It provides a wide range of features, including adding and updating member profiles, managing the catalog of books, and handling book issue and return transactions. The system is built in alignment with the client's requirements, focusing on functionality, security, and usability.

As a typical management information system, it involves both front-end and back-end development. The back-end ensures data consistency, integrity, and strong security practices, while leveraging robust libraries and frameworks. The front-end is designed to be intuitive, responsive, and easy to navigate, ensuring a smooth experience for users.

The database design includes three primary tables—Books, Members, and Transactions—forming the backbone of the system.

INTRODUCTION

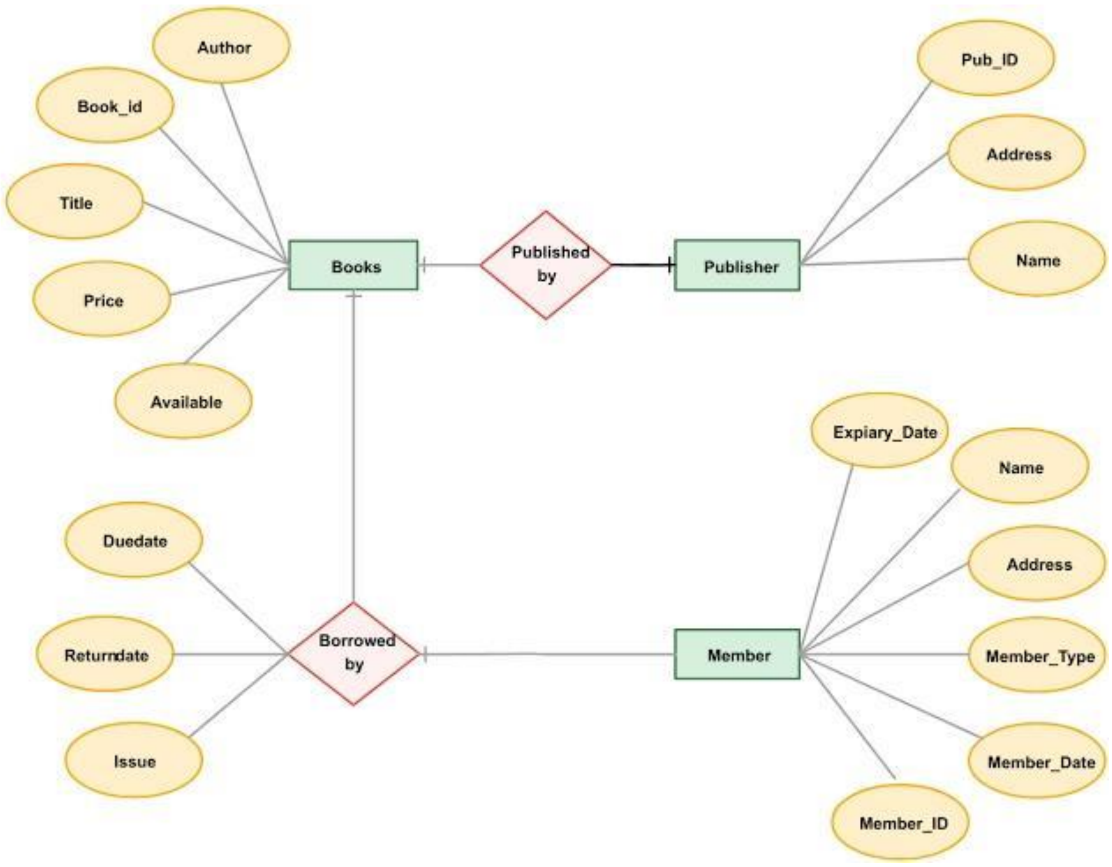
Library Management System is an application which refers to library systems which are generally small or medium in size. It is used by librarian to manage the library using a computerized system where he/she can record various transactions like issue of books, return of books, addition of new books, addition of new students etc. Books and student maintenance modules are also included in this system which would keep track of the students using the library and also a detailed description about the books a library contains. With this computerized system there will be no loss of book record or member record which generally happens when a non computerized system is used.

Objectives

PROJECT AIMS AND OBJECTIVES :

- Online book issue
- Request column for librarian for providing new books .
- A separate column for digital library
- Student login page where student can find books issued by him/her and date of return.
- A search column to search availability of books
- A teacher login page where teacher can add any events being organized in the college and important suggestions regarding books.
- Online notice board about the workshop.

ER Diagram



Author
Book_id
Title
Price
Available
Books
Published by
Publisher
Pub_ID
Address
Name
Expiary_Date
Name
Address
Member_Type
Member_Date
Member_ID
Member
Borrowed by
Duedate
Returndate
Issue

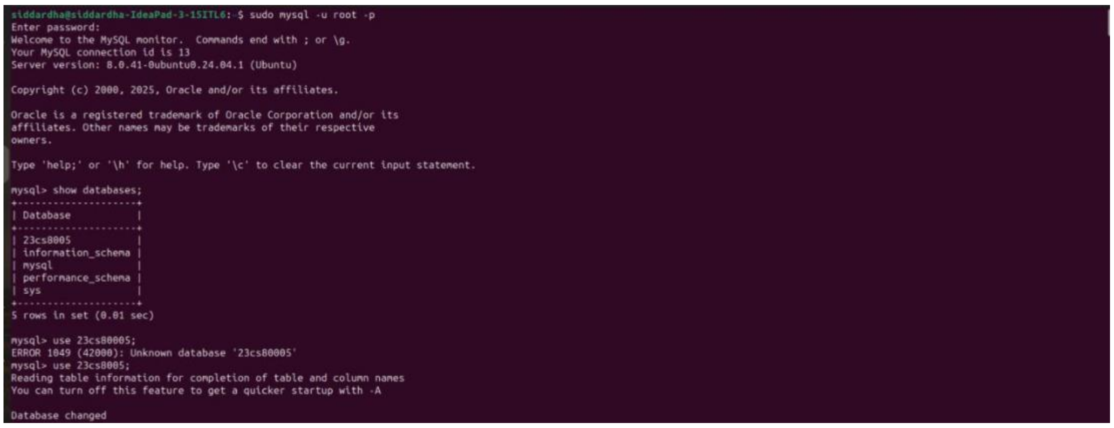
Technologies Used

OPERATION ENVIRONMENT:

PROCESSOR	INTEL CORE PROCESSOR
OPERATING SYSTEM	UBUNTU
MEMORY	1GB RAM OR MORE
HARD DISK SPACE	MINIMUM 3 GB FOR DATABASE USAGE FOR FUTURE
DATABASE	MY SQL
Query Language	SQL
Platform	Localhost

SQL Code

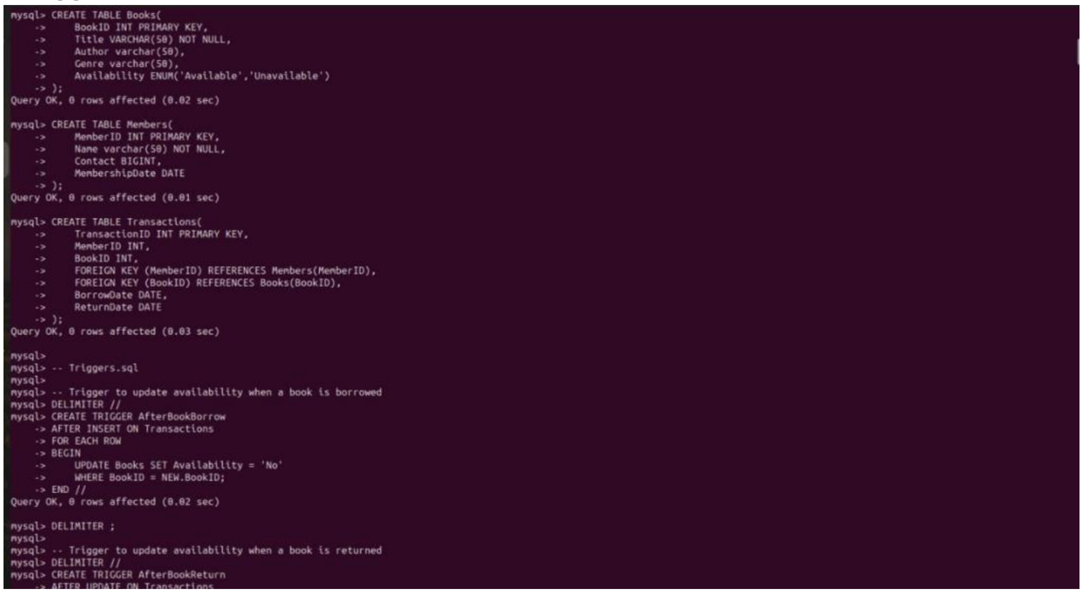
Opening the test bench :-



```
siddardha@siddardha-IdeaPad-3-15ITL6:~$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.41-0ubuntu0.24.04.1 (Ubuntu)

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| 23cs8005           |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.01 sec)

mysql> use 23cs80005;
ERROR 1049 (42000): Unknown database '23cs80005'
mysql> use 23cs8005;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

```
-- • Books (BookID, Title, Author, Genre, Availability)
-- • Members (MemberID, Name, Contact, MembershipDate)
-- • Transactions (TransactionID, MemberID, BookID, BorrowDate,
ReturnDate)
CREATE TABLE Books(
    BookID INT PRIMARY KEY,
    Title VARCHAR(50) NOT NULL,
    Author varchar(50),
    Genre varchar(50),
    Availability ENUM('Yes','No')
);
CREATE TABLE Members(
    MemberID INT PRIMARY KEY,
    Name varchar(50) NOT NULL,
    Contact BIGINT,
    MembershipDate DATE
);
CREATE TABLE Transactions(
    TransactionID INT PRIMARY KEY,
    MemberID INT,
    BookID INT,
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    BorrowDate DATE,
    ReturnDate DATE
);
-- Triggers.sql

-- Trigger to update availability when a book is borrowed
DELIMITER //
CREATE TRIGGER AfterBookBorrow
AFTER INSERT ON Transactions
FOR EACH ROW
BEGIN
    UPDATE Books SET Availability = 'No'
    WHERE BookID = NEW.BookID;
END //
DELIMITER ;

-- Trigger to update availability when a book is returned
DELIMITER //
CREATE TRIGGER AfterBookReturn
AFTER UPDATE ON Transactions
FOR EACH ROW
BEGIN
    IF NEW.ReturnDate IS NOT NULL THEN
        UPDATE Books SET Availability = 'Yes'
        WHERE BookID = NEW.BookID;
    END IF;
END //
DELIMITER ;
```

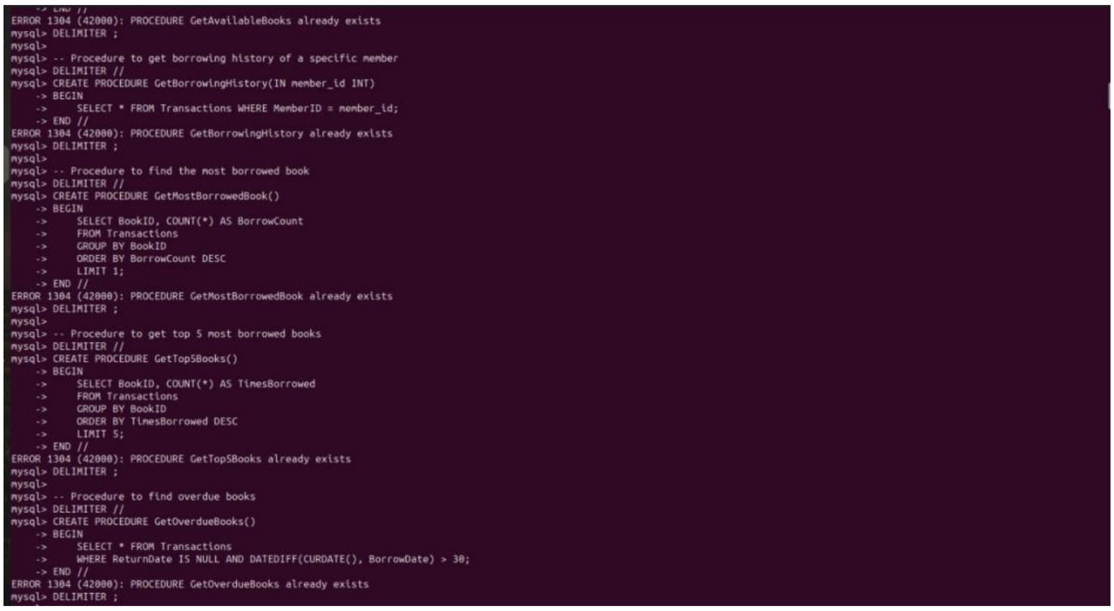
Trigger in test-bench:



```
mysql> CREATE TABLE Books(
    ->     BookID INT PRIMARY KEY,
    ->     Title VARCHAR(50) NOT NULL,
    ->     Author varchar(50),
    ->     Genre varchar(50),
    ->     Availability ENUM('Available','Unavailable')
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Members(
    ->     MemberID INT PRIMARY KEY,
    ->     Name varchar(50) NOT NULL,
    ->     Contact BIGINT,
    ->     MembershipDate DATE
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE Transactions(
    ->     TransactionID INT PRIMARY KEY,
    ->     MemberID INT,
    ->     BookID INT,
    ->     FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
    ->     FOREIGN KEY (BookID) REFERENCES Books(BookID),
    ->     BorrowDate DATE,
    ->     ReturnDate DATE
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> -- Triggers.sql
mysql>
mysql> -- Trigger to update availability when a book is borrowed
mysql> DELIMITER //
mysql> CREATE TRIGGER AfterBookBorrow
    -> AFTER INSERT ON Transactions
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE Books SET Availability = 'No'
    ->     WHERE BookID = NEW.BookID;
    -> END //
Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;
mysql>
mysql> -- Trigger to update availability when a book is returned
mysql> DELIMITER //
mysql> CREATE TRIGGER AfterBookReturn
    -> AFTER UPDATE ON Transactions
```

```
-- Stored_procedures.sql

-- Sample: Procedure to retrieve available books
DELIMITER //
CREATE PROCEDURE GetAvailableBooks()
BEGIN
    SELECT * FROM Books WHERE Availability = 'Yes';
END //
DELIMITER ;

-- Procedure to get borrowing history of a specific member
DELIMITER //
CREATE PROCEDURE GetBorrowingHistory(IN member_id INT)
BEGIN
    SELECT * FROM Transactions WHERE MemberID = member_id;
END //
DELIMITER ;

-- Procedure to find the most borrowed book
DELIMITER //
CREATE PROCEDURE GetMostBorrowedBook()
BEGIN
    SELECT BookID, COUNT(*) AS BorrowCount
    FROM Transactions
    GROUP BY BookID
    ORDER BY BorrowCount DESC
    LIMIT 1;
END //
DELIMITER ;

-- Procedure to get top 5 most borrowed books
DELIMITER //
CREATE PROCEDURE GetTop5Books()
BEGIN
    SELECT BookID, COUNT(*) AS TimesBorrowed
    FROM Transactions
    GROUP BY BookID
    ORDER BY TimesBorrowed DESC
    LIMIT 5;
END //
DELIMITER ;

-- Procedure to find overdue books
DELIMITER //
CREATE PROCEDURE GetOverdueBooks()
BEGIN
    SELECT * FROM Transactions
    WHERE ReturnDate IS NULL AND DATEDIFF(CURDATE(), BorrowDate) > 30;
END //
DELIMITER ;

-- Add more procedures as needed for the rest of the operations
```

Stored procedural in test- bench:



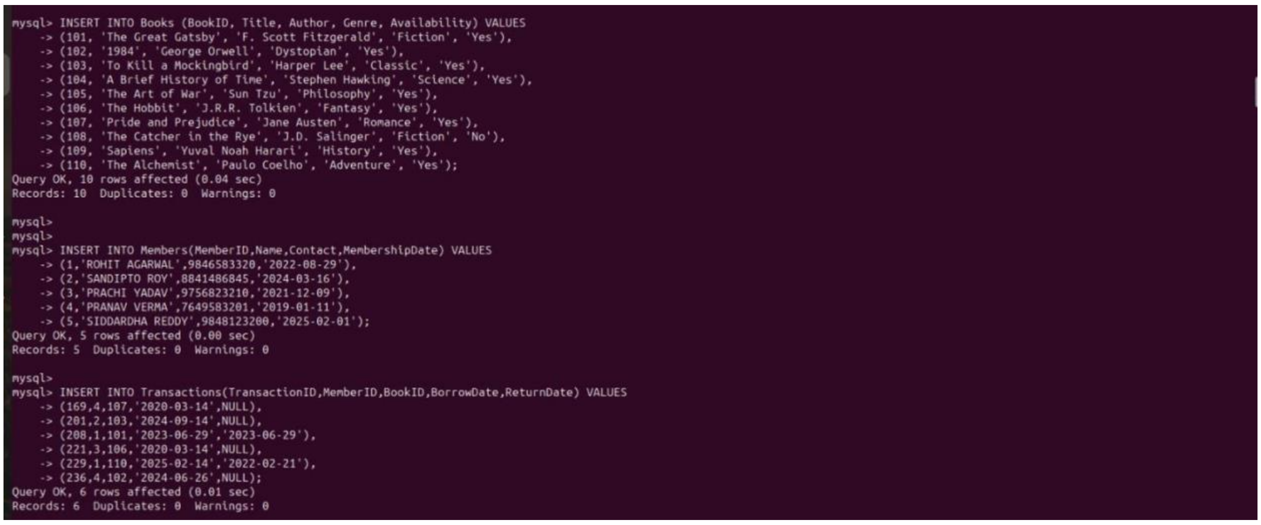
```
ERROR 1304 (42000): PROCEDURE GetAvailableBooks already exists
mysql> DELIMITER ;
mysql>
mysql> -- Procedure to get borrowing history of a specific member
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetBorrowingHistory(IN member_id INT)
    -> BEGIN
    ->     SELECT * FROM Transactions WHERE MemberID = member_id;
    -> END //
ERROR 1304 (42000): PROCEDURE GetBorrowingHistory already exists
mysql> DELIMITER ;
mysql>
mysql> -- Procedure to find the most borrowed book
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetMostBorrowedBook()
    -> BEGIN
    ->     SELECT BookID, COUNT(*) AS BorrowCount
    ->     FROM Transactions
    ->     GROUP BY BookID
    ->     ORDER BY BorrowCount DESC
    ->     LIMIT 1;
    -> END //
ERROR 1304 (42000): PROCEDURE GetMostBorrowedBook already exists
mysql> DELIMITER ;
mysql>
mysql> -- Procedure to get top 5 most borrowed books
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetTop5Books()
    -> BEGIN
    ->     SELECT BookID, COUNT(*) AS TimesBorrowed
    ->     FROM Transactions
    ->     GROUP BY BookID
    ->     ORDER BY TimesBorrowed DESC
    ->     LIMIT 5;
    -> END //
ERROR 1304 (42000): PROCEDURE GetTop5Books already exists
mysql> DELIMITER ;
mysql>
mysql> -- Procedure to find overdue books
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetOverdueBooks()
    -> BEGIN
    ->     SELECT * FROM Transactions
    ->     WHERE ReturnDate IS NULL AND DATEDIFF(CURDATE(), BorrowDate) > 30;
    -> END //
ERROR 1304 (42000): PROCEDURE GetOverdueBooks already exists
mysql> DELIMITER ;
```

```
INSERT INTO Books (BookID, Title, Author, Genre, Availability) VALUES
(101, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Fiction', 'Yes'),
(102, '1984', 'George Orwell', 'Dystopian', 'Yes'),
(103, 'To Kill a Mockingbird', 'Harper Lee', 'Classic', 'Yes'),
(104, 'A Brief History of Time', 'Stephen Hawking', 'Science', 'Yes'),
(105, 'The Art of War', 'Sun Tzu', 'Philosophy', 'Yes'),
(106, 'The Hobbit', 'J.R.R. Tolkien', 'Fantasy', 'Yes'),
(107, 'Pride and Prejudice', 'Jane Austen', 'Romance', 'Yes'),
(108, 'The Catcher in the Rye', 'J.D. Salinger', 'Fiction', 'No'),
(109, 'Sapiens', 'Yuval Noah Harari', 'History', 'Yes'),
(110, 'The Alchemist', 'Paulo Coelho', 'Adventure', 'Yes');
```

```
INSERT INTO Members(MemberID,Name,Contact,MembershipDate) VALUES
(1,'ROHIT AGARWAL',9846583320,'2022-08-29'),
(2,'SANDIPTO ROY',8841486845,'2024-03-16'),
(3,'PRACHI YADAV',9756823210,'2021-12-09'),
(4,'PRANAV VERMA',7649583201,'2019-01-11'),
(5,'SIDDARDHA REDDY',9848123200,'2025-02-01');
```

```
INSERT INTO
Transactions(TransactionID,MemberID,BookID,BorrowDate,ReturnDate)
VALUES
(169,4,107,'2020-03-14',NULL),
(201,2,103,'2024-09-14',NULL),
(208,1,101,'2023-06-29','2023-06-29'),
(221,3,106,'2020-03-14',NULL),
(229,1,110,'2025-02-14','2022-02-21'),
(236,4,102,'2024-06-26',NULL);
```

Inserting data into the table using test-bench:

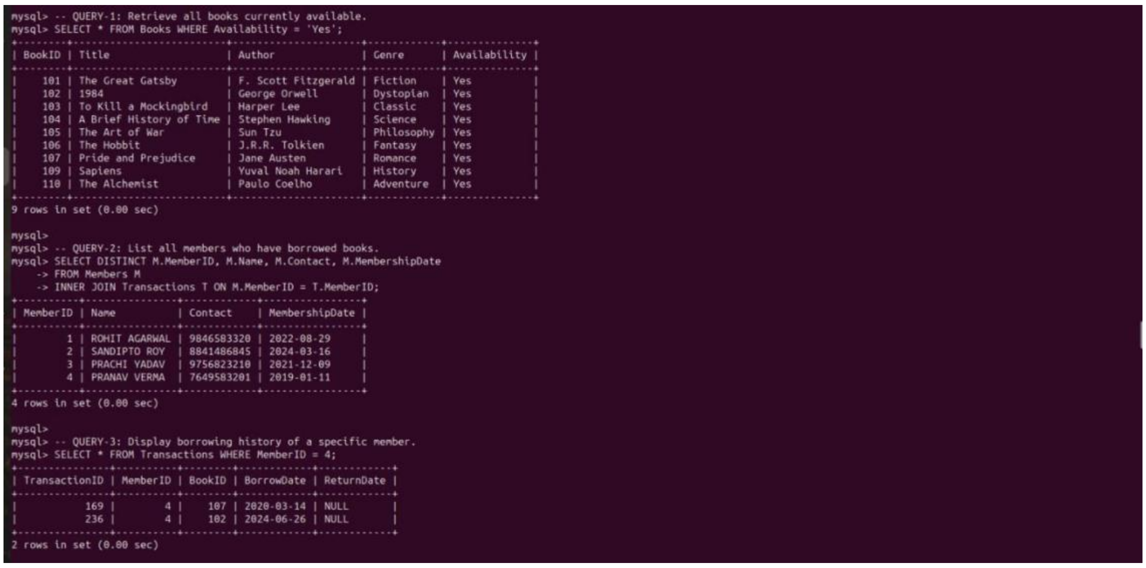


```
mysql> INSERT INTO Books (BookID, Title, Author, Genre, Availability) VALUES
    -> (101, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Fiction', 'Yes'),
    -> (102, '1984', 'George Orwell', 'Dystopian', 'Yes'),
    -> (103, 'To Kill a Mockingbird', 'Harper Lee', 'Classic', 'Yes'),
    -> (104, 'A Brief History of Time', 'Stephen Hawking', 'Science', 'Yes'),
    -> (105, 'The Art of War', 'Sun Tzu', 'Philosophy', 'Yes'),
    -> (106, 'The Hobbit', 'J.R.R. Tolkien', 'Fantasy', 'Yes'),
    -> (107, 'Pride and Prejudice', 'Jane Austen', 'Romance', 'Yes'),
    -> (108, 'The Catcher in the Rye', 'J.D. Salinger', 'Fiction', 'No'),
    -> (109, 'Sapiens', 'Yuval Noah Harari', 'History', 'Yes'),
    -> (110, 'The Alchemist', 'Paulo Coelho', 'Adventure', 'Yes');
Query OK, 10 rows affected (0.04 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql>
mysql>
mysql> INSERT INTO Members(MemberID,Name,Contact,MembershipDate) VALUES
    -> (1,'ROHIT AGARWAL',9846583320,'2022-08-29'),
    -> (2,'SANDIPTO ROY',8841486845,'2024-03-16'),
    -> (3,'PRACHI YADAV',9756823210,'2021-12-09'),
    -> (4,'PRANAV VERMA',7649583201,'2019-01-11'),
    -> (5,'SIDDARDHA REDDY',9848123200,'2025-02-01');
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
mysql> INSERT INTO Transactions(TransactionID,MemberID,BookID,BorrowDate,ReturnDate) VALUES
    -> (169,4,107,'2020-03-14',NULL),
    -> (201,2,103,'2024-09-14',NULL),
    -> (208,1,101,'2023-06-29','2023-06-29'),
    -> (221,3,106,'2020-03-14',NULL),
    -> (229,1,110,'2025-02-14','2022-02-21'),
    -> (236,4,102,'2024-06-26',NULL);
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

QUERIES:-

```
-- QUERY-1: Retrieve all books currently available.
SELECT * FROM Books WHERE Availability = 'Yes';

-- QUERY-2: List all members who have borrowed books.
SELECT DISTINCT M.MemberID, M.Name, M.Contact, M.MembershipDate
FROM Members M
INNER JOIN Transactions T ON M.MemberID = T.MemberID;

-- QUERY-3: Display borrowing history of a specific member.
SELECT * FROM Transactions WHERE MemberID = 4;
```

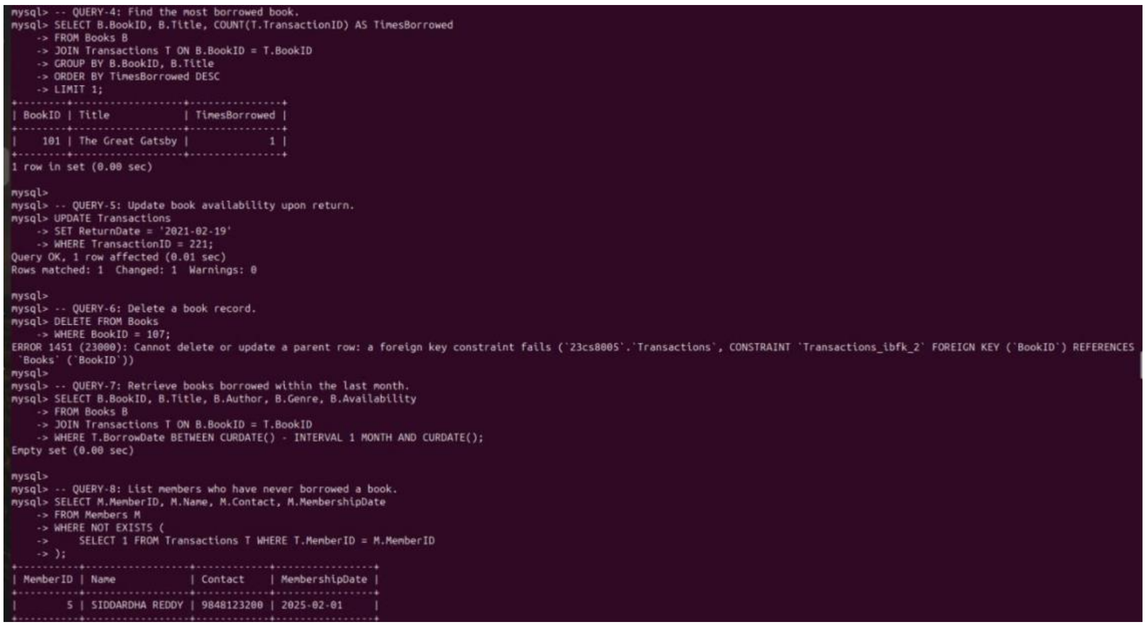
Output:



```
mysql> -- QUERY-1: Retrieve all books currently available.
mysql> SELECT * FROM Books WHERE Availability = 'Yes';
+--------+------------------------+--------------------+-------------+--------------+
| BookID | Title                  | Author             | Genre       | Availability |
+--------+------------------------+--------------------+-------------+--------------+
|    101 | The Great Gatsby       | F. Scott Fitzgerald | Fiction     | Yes          |
|    102 | 1984                   | George Orwell      | Dystopian   | Yes          |
|    103 | To Kill a Mockingbird  | Harper Lee         | Classic     | Yes          |
|    104 | A Brief History of Time | Stephen Hawking    | Science     | Yes          |
|    105 | The Art of War         | Sun Tzu            | Philosophy  | Yes          |
|    106 | The Hobbit             | J.R.R. Tolkien     | Fantasy     | Yes          |
|    107 | Pride and Prejudice    | Jane Austen        | Romance     | Yes          |
|    109 | Sapiens                | Yuval Noah Harari  | History     | Yes          |
|    110 | The Alchemist          | Paulo Coelho       | Adventure   | Yes          |
+--------+------------------------+--------------------+-------------+--------------+
9 rows in set (0.00 sec)

mysql>
mysql> -- QUERY-2: List all members who have borrowed books.
mysql> SELECT DISTINCT M.MemberID, M.Name, M.Contact, M.MembershipDate
    -> FROM Members M
    -> INNER JOIN Transactions T ON M.MemberID = T.MemberID;
+----------+---------------+------------+----------------+
| MemberID | Name          | Contact    | MembershipDate |
+----------+---------------+------------+----------------+
|        1 | ROHIT AGARWAL | 9846583320 | 2022-08-29     |
|        2 | SANDIPTO ROY  | 8841486845 | 2024-03-16     |
|        3 | PRACHI YADAV  | 9756823210 | 2021-12-09     |
|        4 | PRANAV VERMA  | 7649583201 | 2019-01-11     |
+----------+---------------+------------+----------------+
4 rows in set (0.00 sec)

mysql>
mysql> -- QUERY-3: Display borrowing history of a specific member.
mysql> SELECT * FROM Transactions WHERE MemberID = 4;
+---------------+----------+--------+------------+------------+
| TransactionID | MemberID | BookID | BorrowDate | ReturnDate |
+---------------+----------+--------+------------+------------+
|           169 |        4 |    107 | 2020-03-14 | NULL       |
|           236 |        4 |    102 | 2024-06-26 | NULL       |
+---------------+----------+--------+------------+------------+
2 rows in set (0.00 sec)
```

```
-- QUERY-4: Find the most borrowed book.
SELECT B.BookID, B.Title, COUNT(T.TransactionID) AS TimesBorrowed
FROM Books B
JOIN Transactions T ON B.BookID = T.BookID
GROUP BY B.BookID, B.Title
ORDER BY TimesBorrowed DESC
LIMIT 1;

-- QUERY-5: Update book availability upon return.
UPDATE Transactions
SET ReturnDate = '2021-02-19'
WHERE TransactionID = 221;

-- QUERY-6: Delete a book record.
DELETE FROM Books
WHERE BookID = 107;

-- QUERY-7: Retrieve books borrowed within the last month.
SELECT B.BookID, B.Title, B.Author, B.Genre, B.Availability
FROM Books B
JOIN Transactions T ON B.BookID = T.BookID
WHERE T.BorrowDate BETWEEN CURDATE() - INTERVAL 1 MONTH AND CURDATE();

-- QUERY-8: List members who have never borrowed a book.
SELECT M.MemberID, M.Name, M.Contact, M.MembershipDate
FROM Members M
WHERE NOT EXISTS (
    SELECT 1 FROM Transactions T WHERE T.MemberID = M.MemberID);
```

Output:

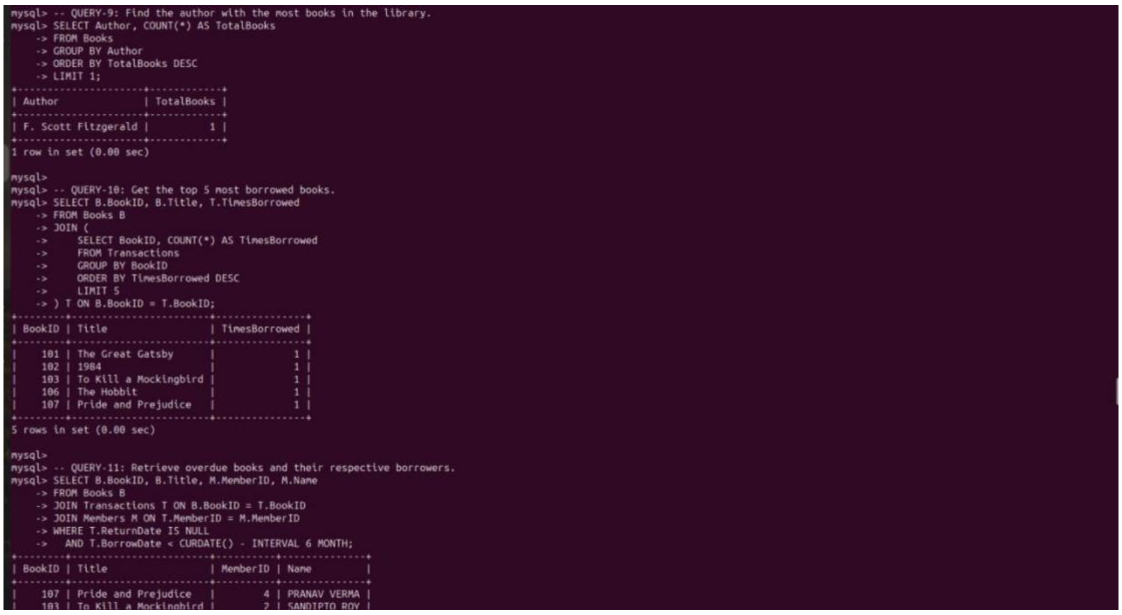


```
mysql> -- QUERY-4: Find the most borrowed book.
mysql> SELECT B.BookID, B.Title, COUNT(T.TransactionID) AS TimesBorrowed
    -> FROM Books B
    -> JOIN Transactions T ON B.BookID = T.BookID
    -> GROUP BY B.BookID, B.Title
    -> ORDER BY TimesBorrowed DESC
    -> LIMIT 1;
+--------+------------------+---------------+
| BookID | Title            | TimesBorrowed |
+--------+------------------+---------------+
|    101 | The Great Gatsby |             1 |
+--------+------------------+---------------+
1 row in set (0.00 sec)

mysql>
mysql> -- QUERY-5: Update book availability upon return.
mysql> UPDATE Transactions
    -> SET ReturnDate = '2021-02-19'
    -> WHERE TransactionID = 221;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> -- QUERY-6: Delete a book record.
mysql> DELETE FROM Books
    -> WHERE BookID = 107;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`23cs8005`.`Transactions`, CONSTRAINT `Transactions_ibfk_2` FOREIGN KEY (`BookID`) REFERENCES
 `Books` (`BookID`))
mysql>
mysql> -- QUERY-7: Retrieve books borrowed within the last month.
mysql> SELECT B.BookID, B.Title, B.Author, B.Genre, B.Availability
    -> FROM Books B
    -> JOIN Transactions T ON B.BookID = T.BookID
    -> WHERE T.BorrowDate BETWEEN CURDATE() - INTERVAL 1 MONTH AND CURDATE();
Empty set (0.00 sec)

mysql>
mysql> -- QUERY-8: List members who have never borrowed a book.
mysql> SELECT M.MemberID, M.Name, M.Contact, M.MembershipDate
    -> FROM Members M
    -> WHERE NOT EXISTS (
    ->     SELECT 1 FROM Transactions T WHERE T.MemberID = M.MemberID
    -> );
+----------+-----------------+------------+----------------+
| MemberID | Name            | Contact    | MembershipDate |
+----------+-----------------+------------+----------------+
|        5 | SIDDARDHA REDDY | 9848123200 | 2025-02-01     |
+----------+-----------------+------------+----------------+
```

```
-- QUERY-9: Find the author with the most books in the library.
SELECT Author, COUNT(*) AS TotalBooks
FROM Books
GROUP BY Author
ORDER BY TotalBooks DESC
LIMIT 1;

-- QUERY-10: Get the top 5 most borrowed books.
SELECT B.BookID, B.Title, T.TimesBorrowed
FROM Books B
JOIN (
    SELECT BookID, COUNT(*) AS TimesBorrowed
    FROM Transactions
    GROUP BY BookID
    ORDER BY TimesBorrowed DESC
    LIMIT 5
) T ON B.BookID = T.BookID;

-- QUERY-11: Retrieve overdue books and their respective borrowers.
SELECT B.BookID, B.Title, M.MemberID, M.Name
FROM Books B
JOIN Transactions T ON B.BookID = T.BookID
JOIN Members M ON T.MemberID = M.MemberID
```

```
WHERE T.ReturnDate IS NULL
  AND T.BorrowDate < CURDATE() - INTERVAL 6 MONTH;
```

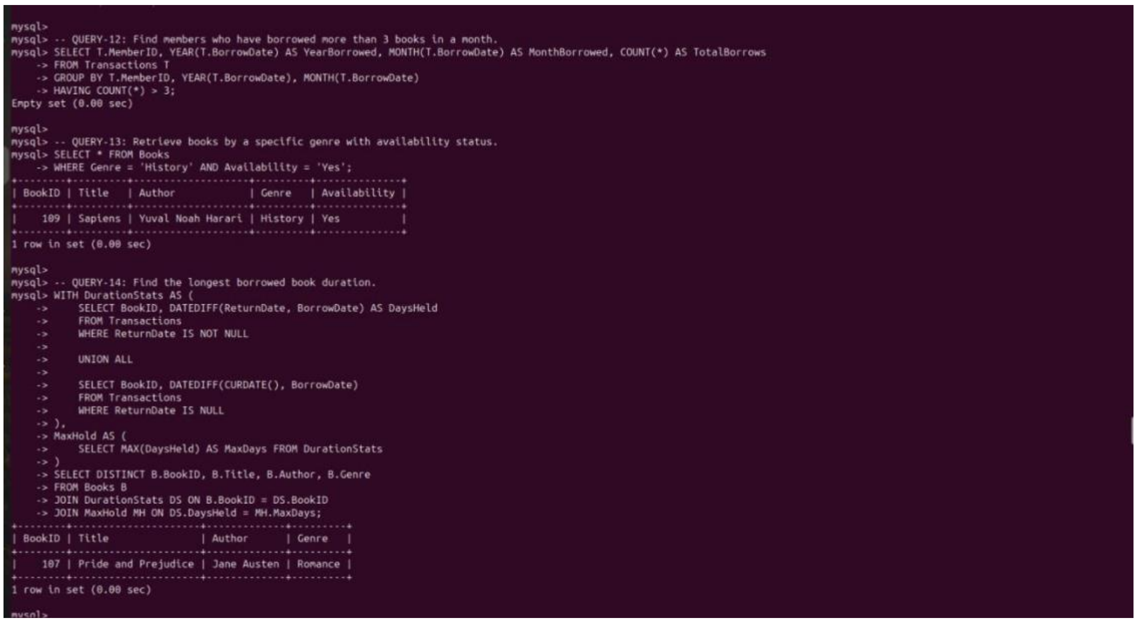
Output:



```
mysql> -- QUERY-9: Find the author with the most books in the library.
mysql> SELECT Author, COUNT(*) AS TotalBooks
    -> FROM Books
    -> GROUP BY Author
    -> ORDER BY TotalBooks DESC
    -> LIMIT 1;
+---------------------+------------+
| Author              | TotalBooks |
+---------------------+------------+
| F. Scott Fitzgerald |          1 |
+---------------------+------------+
1 row in set (0.00 sec)

mysql>
mysql> -- QUERY-10: Get the top 5 most borrowed books.
mysql> SELECT B.BookID, B.Title, T.TimesBorrowed
    -> FROM Books B
    -> JOIN (
    ->     SELECT BookID, COUNT(*) AS TimesBorrowed
    ->     FROM Transactions
    ->     GROUP BY BookID
    ->     ORDER BY TimesBorrowed DESC
    ->     LIMIT 5
    -> ) T ON B.BookID = T.BookID;
+--------+-----------------------+---------------+
| BookID | Title                 | TimesBorrowed |
+--------+-----------------------+---------------+
|    101 | The Great Gatsby      |             1 |
|    102 | 1984                  |             1 |
|    103 | To Kill a Mockingbird |             1 |
|    106 | The Hobbit            |             1 |
|    107 | Pride and Prejudice   |             1 |
+--------+-----------------------+---------------+
5 rows in set (0.00 sec)

mysql>
mysql> -- QUERY-11: Retrieve overdue books and their respective borrowers.
mysql> SELECT B.BookID, B.Title, M.MemberID, M.Name
    -> FROM Books B
    -> JOIN Transactions T ON B.BookID = T.BookID
    -> JOIN Members M ON T.MemberID = M.MemberID
    -> WHERE T.ReturnDate IS NULL
    ->   AND T.BorrowDate < CURDATE() - INTERVAL 6 MONTH;
+--------+-----------------------+----------+--------------+
| BookID | Title                 | MemberID | Name         |
+--------+-----------------------+----------+--------------+
|    107 | Pride and Prejudice   |        4 | PRANAV VERMA |
|    103 | To Kill a Mockingbird |        2 | SANDIPTO ROY |
```

```
-- QUERY-12: Find members who have borrowed more than 3 books in a month.
SELECT T.MemberID, YEAR(T.BorrowDate) AS YearBorrowed,
MONTH(T.BorrowDate) AS MonthBorrowed, COUNT(*) AS TotalBorrows
FROM Transactions T
GROUP BY T.MemberID, YEAR(T.BorrowDate), MONTH(T.BorrowDate)
HAVING COUNT(*) > 3;

-- QUERY-13: Retrieve books by a specific genre with availability status.
SELECT * FROM Books
WHERE Genre = 'History' AND Availability = 'Yes';

-- QUERY-14: Find the longest borrowed book duration.
WITH DurationStats AS (
    SELECT BookID, DATEDIFF(ReturnDate, BorrowDate) AS DaysHeld
    FROM Transactions
    WHERE ReturnDate IS NOT NULL

    UNION ALL

    SELECT BookID, DATEDIFF(CURDATE(), BorrowDate)
    FROM Transactions
    WHERE ReturnDate IS NULL
),
MaxHold AS (
    SELECT MAX(DaysHeld) AS MaxDays FROM DurationStats
)
SELECT DISTINCT B.BookID, B.Title, B.Author, B.Genre
FROM Books B
JOIN DurationStats DS ON B.BookID = DS.BookID
JOIN MaxHold MH ON DS.DaysHeld = MH.MaxDays;
```

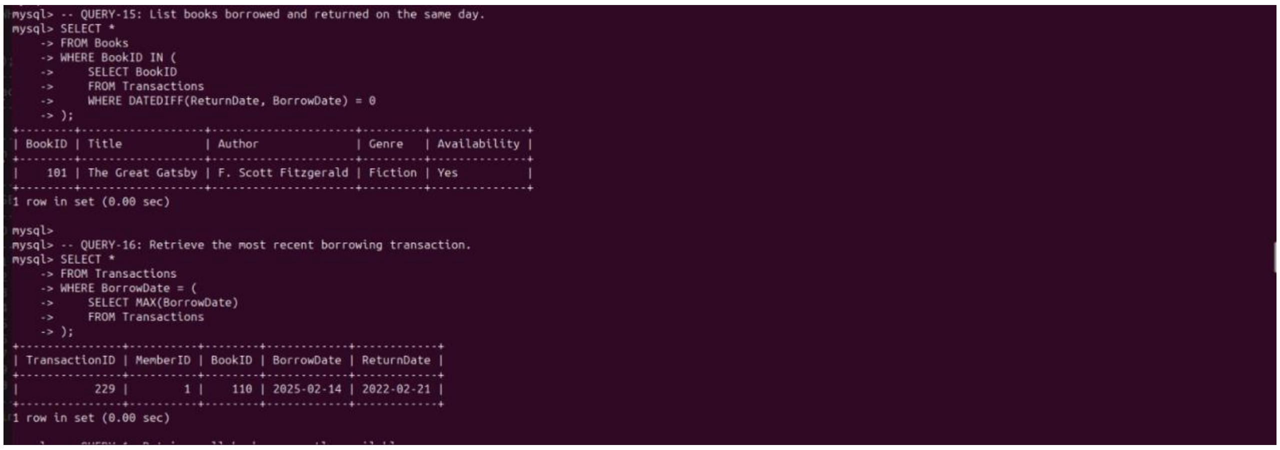
Output:



```
mysql>
mysql> -- QUERY-12: Find members who have borrowed more than 3 books in a month.
mysql> SELECT T.MemberID, YEAR(T.BorrowDate) AS YearBorrowed, MONTH(T.BorrowDate) AS MonthBorrowed, COUNT(*) AS TotalBorrows
    -> FROM Transactions T
    -> GROUP BY T.MemberID, YEAR(T.BorrowDate), MONTH(T.BorrowDate)
    -> HAVING COUNT(*) > 3;
Empty set (0.00 sec)

mysql>
mysql> -- QUERY-13: Retrieve books by a specific genre with availability status.
mysql> SELECT * FROM Books
    -> WHERE Genre = 'History' AND Availability = 'Yes';
+--------+---------+-------------------+---------+--------------+
| BookID | Title   | Author            | Genre   | Availability |
+--------+---------+-------------------+---------+--------------+
|    109 | Sapiens | Yuval Noah Harari | History | Yes          |
+--------+---------+-------------------+---------+--------------+
1 row in set (0.00 sec)

mysql>
mysql> -- QUERY-14: Find the longest borrowed book duration.
mysql> WITH DurationStats AS (
    ->     SELECT BookID, DATEDIFF(ReturnDate, BorrowDate) AS DaysHeld
    ->     FROM Transactions
    ->     WHERE ReturnDate IS NOT NULL
    ->
    ->     UNION ALL
    ->
    ->     SELECT BookID, DATEDIFF(CURDATE(), BorrowDate)
    ->     FROM Transactions
    ->     WHERE ReturnDate IS NULL
    -> ),
    -> MaxHold AS (
    ->     SELECT MAX(DaysHeld) AS MaxDays FROM DurationStats
    -> )
    -> SELECT DISTINCT B.BookID, B.Title, B.Author, B.Genre
    -> FROM Books B
    -> JOIN DurationStats DS ON B.BookID = DS.BookID
    -> JOIN MaxHold MH ON DS.DaysHeld = MH.MaxDays;
+--------+---------------------+-------------+---------+
| BookID | Title               | Author      | Genre   |
+--------+---------------------+-------------+---------+
|    107 | Pride and Prejudice | Jane Austen | Romance |
+--------+---------------------+-------------+---------+
1 row in set (0.00 sec)

mysql>
```

```
-- QUERY-15: List books borrowed and returned on the same day.
SELECT *
FROM Books
WHERE BookID IN (
    SELECT BookID
    FROM Transactions
    WHERE DATEDIFF(ReturnDate, BorrowDate) = 0
);

-- QUERY-16: Retrieve the most recent borrowing transaction.
SELECT *
FROM Transactions
WHERE BorrowDate = (
    SELECT MAX(BorrowDate)
    FROM Transactions
);
```

Output:



```
mysql> -- QUERY-15: List books borrowed and returned on the same day.
mysql> SELECT *
    -> FROM Books
    -> WHERE BookID IN (
    ->     SELECT BookID
    ->     FROM Transactions
    ->     WHERE DATEDIFF(ReturnDate, BorrowDate) = 0
    -> );
+--------+------------------+--------------------+---------+--------------+
| BookID | Title            | Author             | Genre   | Availability |
+--------+------------------+--------------------+---------+--------------+
|    101 | The Great Gatsby | F. Scott Fitzgerald | Fiction | Yes          |
+--------+------------------+--------------------+---------+--------------+
1 row in set (0.00 sec)

mysql>
mysql> -- QUERY-16: Retrieve the most recent borrowing transaction.
mysql> SELECT *
    -> FROM Transactions
    -> WHERE BorrowDate = (
    ->     SELECT MAX(BorrowDate)
    ->     FROM Transactions
    -> );
+---------------+----------+--------+------------+------------+
| TransactionID | MemberID | BookID | BorrowDate | ReturnDate |
+---------------+----------+--------+------------+------------+
|           229 |        1 |    110 | 2025-02-14 | 2022-02-21 |
+---------------+----------+--------+------------+------------+
1 row in set (0.00 sec)
```

CONCLUSION

The Library Management System project successfully provides a simple yet effective way to manage books, members, and transactions within a library. It allows librarians to track available books, manage member details, and monitor borrowing and returning activities.

Key features include:

Book Management: Adding, updating, and removing books, with availability tracking.

Member Management: Storing member information and tracking their borrowing activities.

Transaction Tracking: Recording when books are borrowed and returned, including overdue books.

Essentially, we built a straightforward library system that lets librarians easily handle books and members. It keeps track of who borrows what and when things are due, making daily library tasks much simpler.
We're happy with how it turned out, and we learned a lot about databases along the way.

References

1)GeeksforGeeks SQL - https://www.geeksforgeeks.org/sql-tutorial/

2) Class Notes and Lectures from the DBMS Course

3)Faculty Guidance and Peer Discussions