DataBaseManagementSystem

ASSIGNMENT – 2

Y. Marthanda Teja

192311285

**1Q)** Develop a database for managing library operations, including books, authors, members, and transactions. - Model tables for books, authors, members, and transactions. - Write stored procedures for borrowing and returning books. - Implement triggers to update book availability in real-time. - Write SQL queries to generate reports on popular books and overdue transactions.

### **A) 1. Database Schema Design**

**Books Table**

Stores information about books.

| **Column** | **Type** | **Details** |
| --- | --- | --- |
| book\_id | INT (Primary Key) | Unique identifier for the book |
| title | VARCHAR(255) | Title of the book |
| author\_id | INT (Foreign Key) | Links to the Authors.author\_id |
| published\_year | INT | Year the book was published |
| total\_copies | INT | Total number of copies available |
| available\_copies | INT | Number of copies currently available |
| genre | VARCHAR(100) | Genre of the book |

**Authors Table**

Stores information about authors.

| **Column** | **Type** | **Details** |
| --- | --- | --- |
| author\_id | INT (Primary Key) | Unique identifier for authors |
| name | VARCHAR(255) | Full name of the author |
| birth\_year | INT | Year of birth |

**Members Table**

Stores information about library members.

| **Column** | **Type** | **Details** |
| --- | --- | --- |
| member\_id | INT (Primary Key) | Unique identifier for members |
| name | VARCHAR(255) | Name of the member |
| email | VARCHAR(255) | Email address of the member |
| phone | VARCHAR(15) | Contact number |
| join\_date | DATE | Date when the member joined |

**Transactions Table**

Logs borrowing and returning of books.

| **Column** | **Type** | **Details** |
| --- | --- | --- |
| transaction\_id | INT (Primary Key) | Unique identifier for the transaction |
| book\_id | INT (Foreign Key) | Links to Books.book\_id |
| member\_id | INT (Foreign Key) | Links to Members.member\_id |
| borrow\_date | DATE | Date the book was borrowed |
| due\_date | DATE | Date the book is due for return |
| return\_date | DATE (Nullable) | Date the book was returned |

**2. Stored Procedures**

**Borrow a Book**

This procedure handles borrowing a book, ensuring there are available copies.

sql

Copy code

DELIMITER //

CREATE PROCEDURE BorrowBook(

IN p\_book\_id INT,

IN p\_member\_id INT,

IN p\_days\_due INT

)

BEGIN

DECLARE available INT;

-- Check availability

SELECT available\_copies INTO available FROM Books WHERE book\_id = p\_book\_id;

IF available > 0 THEN

-- Insert transaction

INSERT INTO Transactions (book\_id, member\_id, borrow\_date, due\_date, return\_date)

VALUES (p\_book\_id, p\_member\_id, CURDATE(), DATE\_ADD(CURDATE(), INTERVAL p\_days\_due DAY), NULL);

-- Update book availability

UPDATE Books SET available\_copies = available\_copies - 1 WHERE book\_id = p\_book\_id;

ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'No copies available for borrowing';

END IF;

END //

DELIMITER ;

**Return a Book**

This procedure handles returning a book.

sql

Copy code

DELIMITER //

CREATE PROCEDURE ReturnBook(

IN p\_transaction\_id INT

)

BEGIN

DECLARE p\_book\_id INT;

-- Get the book ID from the transaction

SELECT book\_id INTO p\_book\_id FROM Transactions WHERE transaction\_id = p\_transaction\_id;

-- Update transaction with return date

UPDATE Transactions SET return\_date = CURDATE() WHERE transaction\_id = p\_transaction\_id;

-- Increment available copies

UPDATE Books SET available\_copies = available\_copies + 1 WHERE book\_id = p\_book\_id;

END //

DELIMITER ;

**3. Triggers**

**Update Availability on Transaction Delete**

Ensures that availability is adjusted when a transaction is removed.

sql

Copy code

DELIMITER //

CREATE TRIGGER UpdateAvailabilityOnTransactionDelete

AFTER DELETE ON Transactions

FOR EACH ROW

BEGIN

-- Increment available copies

UPDATE Books SET available\_copies = available\_copies + 1 WHERE book\_id = OLD.book\_id;

END //

DELIMITER ;

**4. Analytical Queries**

**Query 1: Popular Books**

Identify the most borrowed books.

sql

Copy code

SELECT b.title, COUNT(t.transaction\_id) AS borrow\_count

FROM Books b

JOIN Transactions t ON b.book\_id = t.book\_id

GROUP BY b.book\_id, b.title

ORDER BY borrow\_count DESC

LIMIT 10;

**Query 2: Overdue Transactions**

List transactions where the return date is past the due date.

sql

Copy code

SELECT t.transaction\_id, m.name AS member\_name, b.title AS book\_title, t.due\_date, DATEDIFF(CURDATE(), t.due\_date) AS days\_overdue

FROM Transactions t

JOIN Members m ON t.member\_id = m.member\_id

JOIN Books b ON t.book\_id = b.book\_id

WHERE t.return\_date IS NULL AND t.due\_date < CURDATE()

ORDER BY days\_overdue DESC;