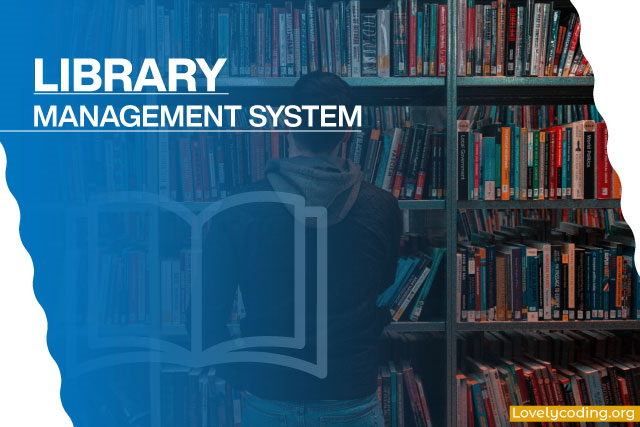
Employee ID: SP 11213

Employee Name: Nidhin Nandakumar

Project Name: Library Management System



**Project Background**

The Library Management System (LMS) project is aimed at providing an efficient and user-friendly solution to automate and streamline the operations of a modern library. In today's digital age, libraries face new challenges and opportunities in managing their collections, improving patron services, and adapting to the changing landscape of information management. This project addresses these challenges and leverages technology to enhance the library experience.

**Project Objective**

The primary objective of the Library Management System (LMS) project is to design, develop, and implement a comprehensive software solution that enhances the efficiency and effectiveness of library operations. The system aims to modernize and streamline the management of library resources, improve patron services, and provide data-driven insights for library administration.

The project seeks to achieve the following specific goals:

**Automation and Cataloging:** Implement a robust cataloging system (using the "Book," "Author," "Publisher," and "Genre" tables) to efficiently manage and track library resources. This includes automating the entry and maintenance of book records, author details, publisher information, and genre categorization.

**Patron Services:** Enhance patron services by creating a user-friendly interface for patrons to search and access library resources. The system (utilizing the "Patron," "Checkout," "Reservation," and "Library\_Card" tables) will enable patrons to borrow, return, and reserve items, and provide them with essential information, such as due dates and fine tracking.

**Fine Management:** Implement a fine tracking system (utilizing the "Fine" table) to accurately calculate and manage fines for overdue materials. Patrons will be informed about accrued fines, and library staff can efficiently handle fine collection and record-keeping.

**Staff Management:** Create a user management system (with the "Staff" table) to support library staff in their roles. Staff members will be able to access and manage the system securely and efficiently.

**Activities:**  
**Creating Tables using Queries**

Creating Tables

**Book Table**

CREATE TABLE Book (

ISBN VARCHAR(20) PRIMARY KEY,

Title VARCHAR(255),

Author\_ID INT,

Patron\_ID INT,

Publisher\_ID INT,

Genre\_ID INT,

Total\_Copies INT

);

Insertion of value example:

INSERT INTO Book (ISBN, Title, Author\_ID, Publisher\_ID, Genre\_ID, Total\_Copies)

VALUES('ISBN001', 'The Secret Garden', 1, 1, 1, 5)

**Patron Table**

CREATE TABLE Patron (

Patron\_ID INT PRIMARY KEY,

First\_Name VARCHAR(50),

Last\_Name VARCHAR(50),

Address TEXT,

Phone\_Number VARCHAR(15),

Email VARCHAR(100)

);

Insertion of value example:

INSERT INTO Patron (Patron\_ID, First\_Name, Last\_Name,Address,Phone\_Number,Email,)

VALUES

(100, 'Roger', 'Williams', '123 Main St, City, State, Zip', '555-345-6789', 'roger.williams@example.com')

**Checkout Table**

CREATE TABLE Checkout (

Checkout\_ID INT PRIMARY KEY,

Patron\_ID INT,

ISBN VARCHAR(20),

Checkout\_Date DATE,

Due\_Date DATE,

Return\_Date DATE

);

Insertion of value example:

INSERT INTO Checkout (Checkout\_ID, Patron\_ID, ISBN, Checkout\_Date, Due\_Date, Return\_Date)

VALUES

(1, 100, 'ISBN001', '2023-04-01', '2023-04-15', '2023-04-10'),

**Author Table**

CREATE TABLE Author (

Author\_ID INT PRIMARY KEY,

Author\_Name VARCHAR(100)

);

Insertion of value example:

INSERT INTO Author (Author\_ID, Author\_Name)

VALUES (1, 'Jane Austen')

**Publisher Table**

CREATE TABLE Publisher (

Publisher\_ID INT PRIMARY KEY,

Publisher\_Name VARCHAR(100)

);

Insertion of value example:

INSERT INTO Publisher (Publisher\_ID, Publisher\_Name)

VALUES (1, 'Penguin Books')

**Genre Table**

CREATE TABLE Genre (

Genre\_ID INT PRIMARY KEY,

Genre\_Name VARCHAR(50)

);

Insertion of value example:

INSERT INTO Genre (Genre\_ID, Genre\_Name)

VALUES (1, 'Classics')

**Staff Table**

CREATE TABLE Staff (

Staff\_ID INT PRIMARY KEY,

First\_Name VARCHAR(50),

Last\_Name VARCHAR(50)

);

Insertion of value example:

INSERT INTO Staff (Staff\_ID, First\_Name, Last\_Name)

VALUES

(1, 'John', 'Smith')

**Reservation Table**

CREATE TABLE Reservation (

Reservation\_ID INT PRIMARY KEY,

Patron\_ID INT,

ISBN VARCHAR(20),

Reservation\_Date DATE

);

Insertion of value example:

INSERT INTO Reservation (Reservation\_ID, Patron\_ID, ISBN, Reservation\_Date)

VALUES

(1, 100, 'ISBN001', '2023-03-01')

**Library Table**

CREATE TABLE Library\_Card (

Card\_ID INT PRIMARY KEY,

Patron\_ID INT,

Issue\_Date DATE,

Staff\_ID INT,

Expiration\_Date DATE

);

Insertion of value example:

INSERT INTO Library\_Card (Card\_ID, Patron\_ID, Issue\_Date,Staff\_ID,Expiration\_Date)

VALUES

(1, 100, '2023-04-01',1, '2024-04-01')

**Fine Table**

CREATE TABLE Fine (

Fine\_ID INT PRIMARY KEY,

Patron\_ID INT,

Amount DECIMAL(10, 2)

);

Insertion of value example:

INSERT INTO Fine (Fine\_ID, Patron\_ID, Amount)

VALUES

(1, 100, 5.00)

**Transaction Table**

CREATE TABLE Transactions (

Transaction\_ID INT PRIMARY KEY,

Patron\_ID INT,

ISBN VARCHAR(20),

Transaction\_Date DATE,

Transaction\_Type VARCHAR(50)

);

Insertion of value example:

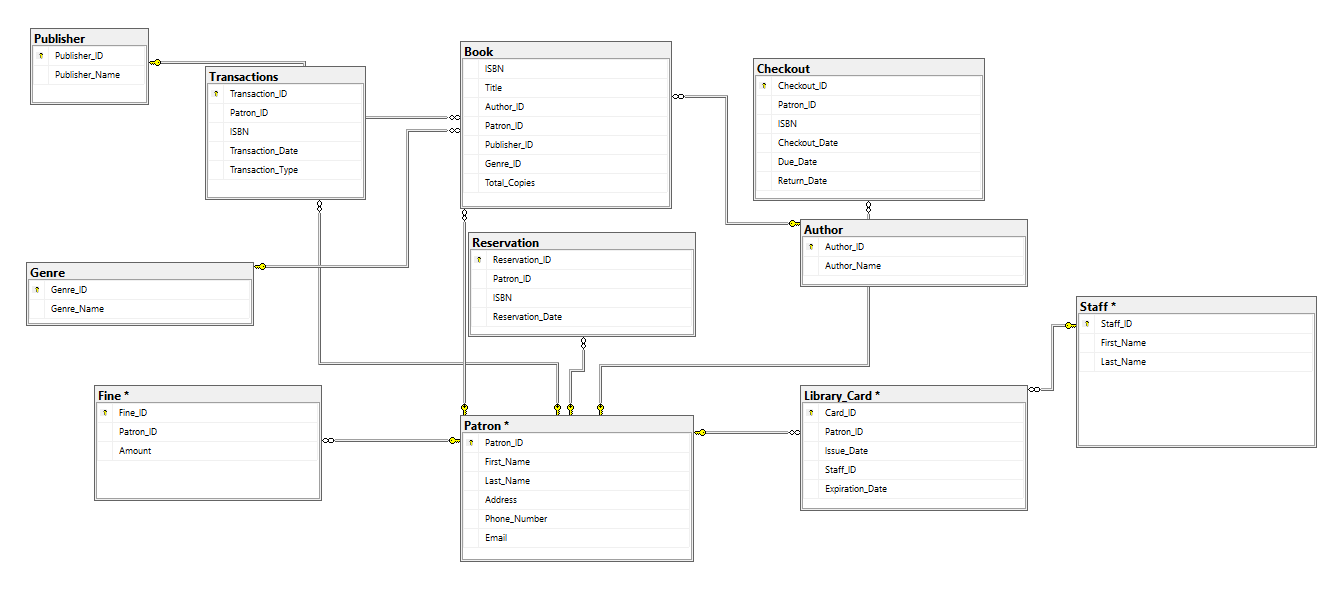
INSERT INTO Transactions (Transaction\_ID, Patron\_ID, ISBN,Transaction\_Date, Transaction\_Type) VALUES (1, 101, 'ISBN001', '2023-10-10', 'Check Out')

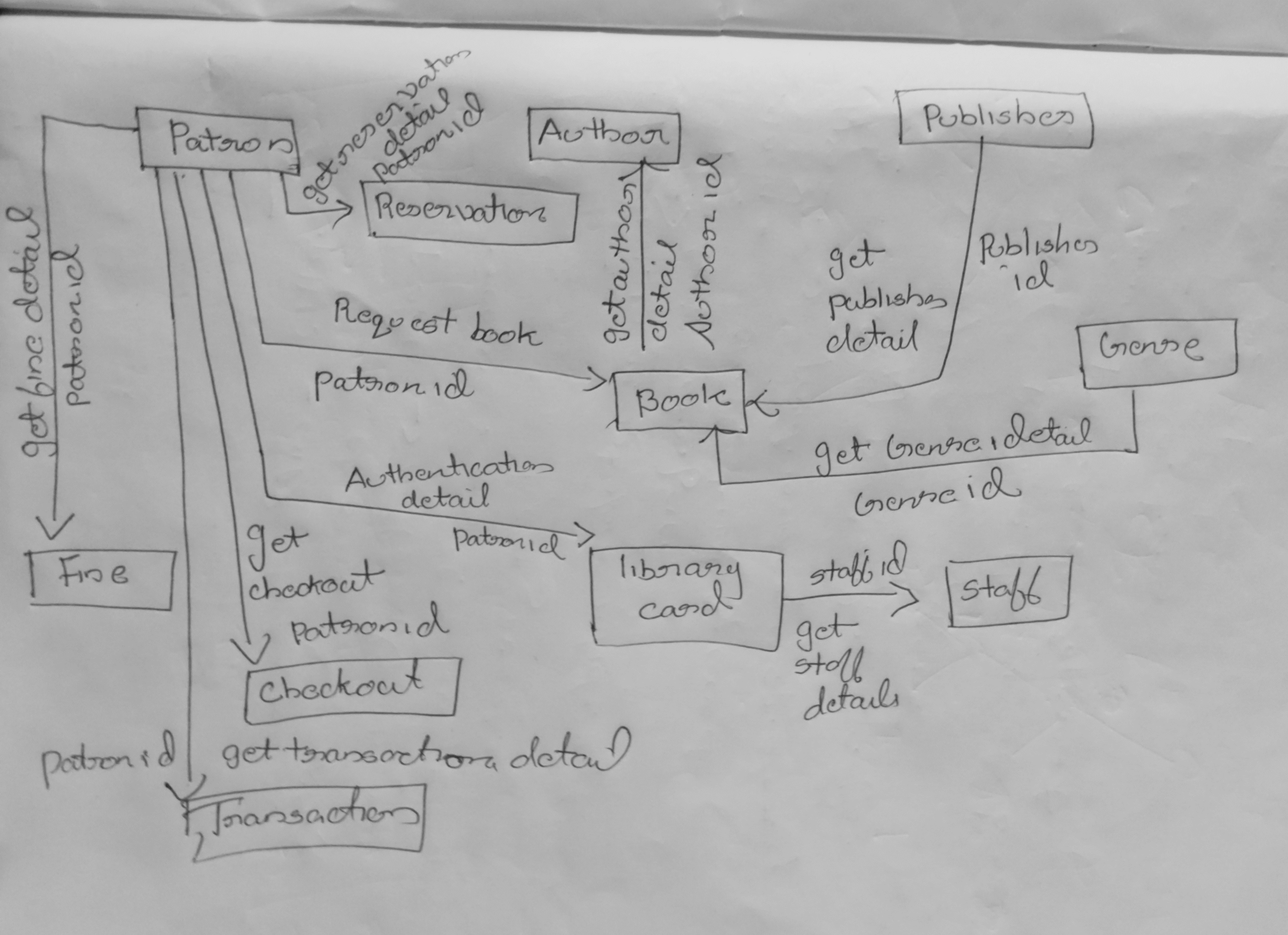
For setting foreign key here is the example of foreign key set in Library\_Card table from table Patron.

FOREIGN KEY (Patron\_ID) REFERENCES Patron(Patron\_ID)

This is how the foreign keys are set in the corresponding tables as per schema diagram.

**Schema Diagram and the Data-Flow Diagram**





**Entering Data in tables using queries and triggers**

Here the trigger is used to calculate fine amount in the fine table on adding data in the checkout table ,if the book is delayed to return.

CREATE TRIGGER CalculateFine

ON Checkout

AFTER INSERT

AS

BEGIN

UPDATE Fine

SET Amount = DATEDIFF(DAY, i.Due\_Date, i.Return\_Date) \* 5

FROM Fine

JOIN Inserted AS i ON Fine.Patron\_ID = i.Patron\_ID

WHERE i.Return\_Date > i.Due\_Date;

END;

Here is an example how value inserted to checkout table so that the trigger will be invoked.

insert into checkout values(26,103,'ISBN022','3-16-2023','4-08-2023','4-16-2023');

**or**

Here is an example for the update query that can be used to update the fine values

UPDATE Fine

SET Amount = DATEDIFF(DAY, Checkout.Due\_Date, Checkout.Return\_Date) \* 5

FROM Fine

JOIN Checkout ON Fine.Patron\_ID = Checkout.Patron\_ID

WHERE Checkout.Return\_Date > Checkout.Due\_Date;

**Creating Views**

**View 1**

Here the View named CheckoutDetails is created to list the name of book,their patrons and their following checkout dates.

CREATE VIEW CheckoutDetails AS

SELECT Book.Title, Patron.First\_Name, Patron.Last\_Name, Checkout.Checkout\_Date

FROM Book

INNER JOIN Checkout ON Book.ISBN = Checkout.ISBN

INNER JOIN Patron ON Checkout.Patron\_ID = Patron.Patron\_ID;

SELECT \* FROM CheckoutDetails;

**View 2**

Here the view named AuthorInfo shows authors and the books they've written in a single, easy-to-access list. It simplifies the process of finding out which author wrote which books.

CREATE VIEW AuthorInfo AS

SELECT A.Author\_ID, A.Author\_Name,

B.ISBN, B.Title

FROM Author A

JOIN Book B ON A.Author\_ID = B.Author\_ID;

select \* from AuthorInfo

**View 3**

This view "PatronTransactions" view displays transaction details, including transaction ID, date, type, along with patron and book information, making it easy to track transactions associated with specific patrons and books.

CREATE VIEW PatronTransactions AS

SELECT T.Transaction\_ID, T.Transaction\_Date, T.Transaction\_Type,

P.Patron\_ID, P.First\_Name, P.Last\_Name,

B.ISBN, B.Title

FROM Transactions T

JOIN Patron P ON T.Patron\_ID = P.Patron\_ID

JOIN Book B ON T.ISBN = B.ISBN;

select \* from PatronTransactions

**Reports using queries**

Query to List all the books checked out by patrons(Users) with library cards issued by Library Staff Named 'John Smith':

SELECT Book.ISBN, Book.Title, Checkout.Checkout\_Date, Checkout.Due\_Date, Checkout.Return\_Date

FROM Book

INNER JOIN Checkout ON Book.ISBN = Checkout.ISBN

INNER JOIN Library\_Card ON Checkout.Patron\_ID = Library\_Card.Patron\_ID

INNER JOIN Staff ON Library\_Card.Staff\_ID = Staff.Staff\_ID

WHERE Staff.First\_Name = 'John' AND Staff.Last\_Name = 'Smith';

Query to calculate the total fine amount for each patron and find the patron’s with the highest total fine in descending order of the fine amount.

SELECT Patron.Patron\_ID, Patron.First\_Name, Patron.Last\_Name, SUM(Fine.Amount) AS Total\_Fines

FROM Patron

LEFT JOIN Fine ON Patron.Patron\_ID = Fine.Patron\_ID

GROUP BY Patron.Patron\_ID, Patron.First\_Name, Patron.Last\_Name

ORDER BY Total\_Fines DESC

Query to find the books that were checked out and returned late, along with the number of days they were overdue:

SELECT Book.ISBN, Book.Title, Checkout.Checkout\_Date, Checkout.Due\_Date, Checkout.Return\_Date,

DATEDIFF(day, Checkout.Due\_Date, Checkout.Return\_Date) AS Days\_Overdue

FROM Book

INNER JOIN Checkout ON Book.ISBN = Checkout.ISBN

WHERE Checkout.Return\_Date > Checkout.Due\_Date;

Query to calculate the average fine amount for each patron who has checked out books more than once, ordered by average fine :

SELECT Fine.Patron\_ID, Patron.First\_Name, Patron.Last\_Name, AVG(Fine.Amount) AS Avg\_Fine

FROM Patron

INNER JOIN Fine ON Patron.Patron\_ID = Fine.Patron\_ID

INNER JOIN Checkout ON Patron.Patron\_ID = Checkout.Patron\_ID

GROUP BY Fine.Patron\_ID, Patron.First\_Name, Patron.Last\_Name

HAVING COUNT(Checkout.Checkout\_ID) > 1

ORDER BY Avg\_Fine;

Query used to calculate and display the total fines collected each month, breaking down the results by year and month:

SELECT

YEAR(Transactions.Transaction\_Date) AS Collection\_Year,

MONTH(Transactions.Transaction\_Date) AS Collection\_Month,

SUM(Fine.Amount) AS Total\_Fines

FROM Transactions

INNER JOIN Fine ON Transactions.Patron\_ID = Fine.Patron\_ID

GROUP BY YEAR(Transactions.Transaction\_Date), MONTH(Transactions.Transaction\_Date)

ORDER BY YEAR(Transactions.Transaction\_Date), MONTH(Transactions.Transaction\_Date);

Query to find the most popular genre based on the total number of book copies:

SELECT Genre.Genre\_Name, SUM(Book.Total\_Copies) AS Total\_Copies

FROM Book

INNER JOIN Genre ON Book.Genre\_ID = Genre.Genre\_ID

GROUP BY Genre.Genre\_Name

ORDER BY Total\_Copies DESC

Query to find the number of books that are not reserved:

SELECT COUNT(\*) AS NumberOfBooksNotReserved

FROM Book b

LEFT JOIN Reservation r ON b.ISBN = r.ISBN

WHERE r.Reservation\_ID IS NULL;

Query to find all reservations made by a specific patron with their reservation dates:

SELECT Book.Title, Reservation.Reservation\_Date

FROM Book

JOIN Reservation ON Book.ISBN = Reservation.ISBN

WHERE Reservation.Patron\_ID = 120;

Query to list all transactions (checkouts, returns, etc.) for a specific patron within a date range:

SELECT ISBN, Transaction\_Date, Transaction\_Type

FROM Transactions

WHERE Patron\_ID = 100

AND Transaction\_Date BETWEEN '2023-01-01' AND '2023-12-31';

Query used to retrieve the latest transaction date for each transaction type in the library database while including the associated patron's first name, patron's last name, and book title:

SELECT Transactions.Transaction\_Type, MAX(Transactions.Transaction\_Date) AS Latest\_Transaction\_Date,

Patron.First\_Name, Patron.Last\_Name, Book.Title

FROM Transactions

JOIN Patron ON Transactions.Patron\_ID = Patron.Patron\_ID

JOIN Book ON Transactions.ISBN = Book.ISBN

GROUP BY Transactions.Transaction\_Type, Patron.First\_Name, Patron.Last\_Name, Book.Title;

Query to retrieve all transactions (of a specific type) for a group of patrons:

SELECT Patron.First\_Name, Patron.Last\_Name, Transactions.Transaction\_Type, Transactions.Transaction\_Date

FROM Patron

JOIN Transactions ON Patron.Patron\_ID = Transactions.Patron\_ID

WHERE Patron.Patron\_ID IN (100, 101, 102)

AND Transactions.Transaction\_Type = 'Check Out';

Query to find all patrons who have checked out books from a specific list of genres:

SELECT DISTINCT Patron.First\_Name, Patron.Last\_Name

FROM Patron

JOIN Checkout ON Patron.Patron\_ID = Checkout.Patron\_ID

JOIN Book ON Checkout.ISBN = Book.ISBN

JOIN Genre ON Book.Genre\_ID = Genre.Genre\_ID

WHERE Genre.Genre\_Name IN ('Mystery', 'Science Fiction', 'Romance');

Query to find the average number of days books are checked out by genre:

SELECT Genre.Genre\_Name, AVG(DATEDIFF(day, Checkout.Checkout\_Date, Checkout.Return\_Date)) AS Avg\_Days\_Checked\_Out

FROM Genre

JOIN Book ON Genre.Genre\_ID = Book.Genre\_ID

JOIN Checkout ON Book.ISBN = Checkout.ISBN

GROUP BY Genre.Genre\_Name;

Query to find the patrons who have both checked out and reserved books:

SELECT DISTINCT Patron.Patron\_ID, Patron.First\_Name, Patron.Last\_Name

FROM Patron

WHERE Patron.Patron\_ID IN (

SELECT Patron\_ID FROM Checkout

INTERSECT

SELECT Patron\_ID FROM Reservation

);

Query to list all the staff members and the number of library cards they have issued:

SELECT Staff.Staff\_ID, Staff.First\_Name, Staff.Last\_Name, COUNT(Library\_Card.Card\_ID) AS Cards\_Issued

FROM Staff

LEFT JOIN Library\_Card ON Staff.Staff\_ID = Library\_Card.Staff\_ID

GROUP BY Staff.Staff\_ID, Staff.First\_Name, Staff.Last\_Name;