CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

UNIVERSITY INSTITUTE OF COMPUTING

CASE STUDY REPORT
ON
LIBRARY MANAGEMENT SYSTEM

Program Name: BCA

Subject Name/Code: Database Management System (23CAT-251)

Submitted By:

Name: Sehajpreet Singh

UID: 23BCA10497

Section: 4-'B'

Submitted To:

Name: Mr.Arvinder Singh

Designation: Assistant Professor

INTRODUCTION

A **Library Management System (LMS)** is a software application designed to handle the day-to-day operations of a library efficiently. It provides a systematic way to manage books, members, borrowing, returning, and fines. Traditional libraries face challenges such as manual record-keeping, delayed book tracking, and inefficient data retrieval. An LMS overcomes these challenges by digitizing the entire library process, reducing paperwork, and ensuring faster access to information.

The objective of this project is to create a relational database for a library that stores and manages information related to books, members, borrowing records, staff, fines, and suppliers. The database facilitates easy monitoring of book inventory, member activity, overdue returns, and financial transactions related to fines. The system is built using MySQL and includes various SQL queries for data manipulation and reporting.

This project demonstrates the use of structured query language (SQL), relational database design, normalization, and key operations like joins, aggregations, filtering, and sorting. It aims to make library management more streamlined, efficient, and accessible.

TECHNIQUES

The primary technology used in this project is MySQL, an open-source relational database management system. The following techniques have been implemented:

- **Entity-Relationship Modeling** for data structure visualisation.
- **Normalisation** to organise data efficiently and remove redundancy.
- **SQL Queries** for data manipulation and retrieval.
- **Use of Constraints** like PRIMARY KEY, FOREIGN KEY to enforce relationships.
- **Join operations** to combine data from multiple tables.
- **Aggregate Functions** to summarize and analyze data.
- **Filtering and Sorting** to extract meaningful insights from the dataset.
- **Stored Procedures and Views** (optional enhancements) for automation.

The goal is to simulate a real-time cinema database with multiple users accessing the system concurrently. Though our current system is simplified, it lays the foundation for large-scale enterprise software.

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

SYSTEM CONFIGURATION

Hardware Requirements

- **Processor:** Intel i5 / Ryzen 5 or higher
- **RAM:** 8 GB minimum
- **Storage:** 256 GB SSD / 500 GB HDD
- **Display:** 14" or larger

Software Requirements

- **OS:** Windows 10/11 or Ubuntu 20.04+
- **DBMS:** MySQL Server 8.0+
- **Interface Tool:** MySQL Workbench / phpMyAdmin
- **ER Tool:** Draw.io / dbdiagram.io
- **Editor:** VS Code / Notepad++

Database Details

- **Name:** vansh_db
- **Tables:** Customers, Orders, OrderDetails, Products, Suppliers, Employees
- **Relations:** Primary & Foreign Keys, Constraints for integrity

INPUT

The Library database receives input from various entities involved in daily supermarket operations. These inputs are collected through forms, employee entries, or automated systems and are stored in structured tables within the database.

Books (BookID, Title, Author, Genre, YearPublished, ISBN, Copies),

Members (MemberID, Name, Email, Phone, MembershipDate, Address),

Borrowing (BorrowID, BookID, MemberID, BorrowDate, DueDate, ReturnDate),

Staff (StaffID, Name, Position, Email, Contact),

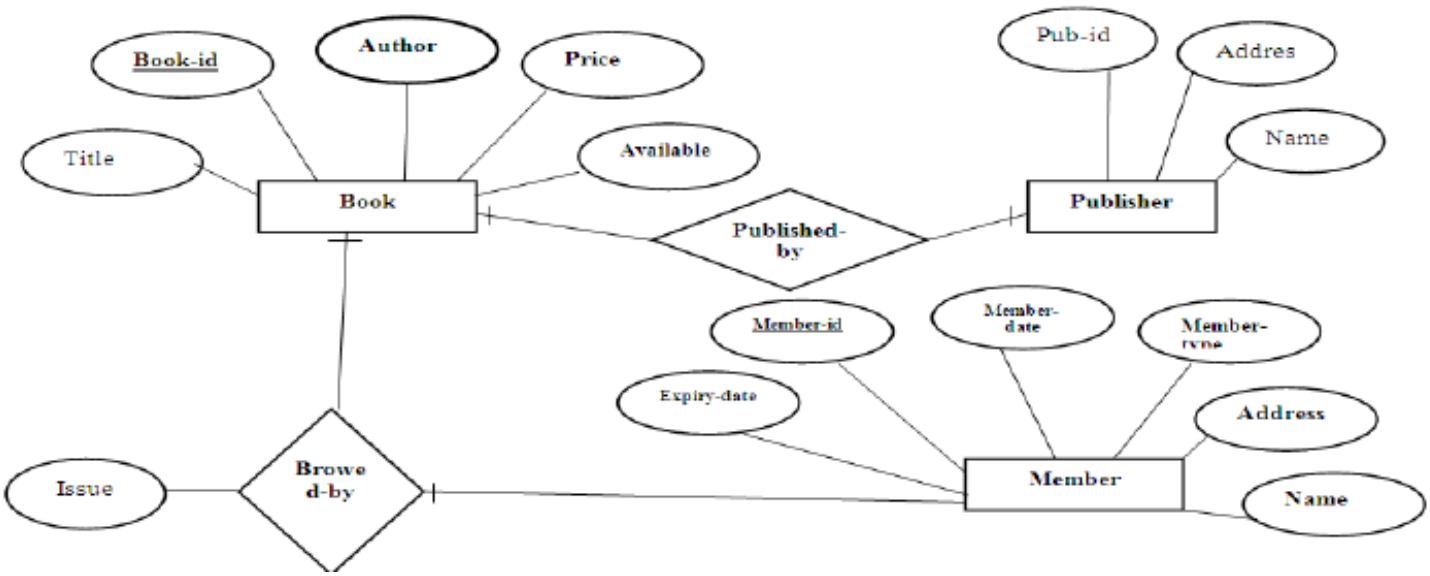
Fines (FineID, BorrowID, Amount, PaidStatus),

Suppliers (SupplierID, Name, ContactPerson, Phone, Address).

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

ENTITY-RELATIONSHIP DIAGRAM



The Entity-Relationship (ER) diagram outlines the structure and relationships among different entities of the hospital. It forms the blueprint for the actual database schema.

Each entity has clearly defined attributes and is connected using appropriate relationships like one-to-many and many-to-one, ensuring normalization and avoiding data redundancy.

RELATIONSHIP BETWEEN TABLES

These relationships ensure that the relational database mirrors real-world interactions within Library.

No.	Relationship Type	Parent Table	Child Table	Foreign key	Description
1	One-to-many	Members	Borrowings	MemberID	member can borrow multiple books
2	One-to-many	Books	Borrowings	BookID	One book can be borrowed multiple times
3	One-to-many	Borrowing	Fines	BorrowID	One borrowing record can generate one fine
4	One-to-many	Staff	Borrowings	StaffID	One staff handle multiple borrowings
5	One-to-many	Suppliers	Books	SupplierID	One supplier supply multiple books

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

TABULAR FORMAT (SCHEMA)

Table Name	Primary Key	Foreign Key	Description
Books	BookID	—	Stores books info
Members	MemberID	—	Records of member
Borrowing	BorrowID	BookID,MemberID	Store borrow details
Staff	StaffID	—	Stores staff info
Suppliers	SupplierID	BookID	Store supplier info
Fines	FineID	BorrowID	Store fine details

TABLE CREATION

1. Books Table:

```
CREATE TABLE Books (
  BookID INT PRIMARY KEY,
  Title VARCHAR(100),
  Author VARCHAR(100),
  Publisher VARCHAR(100),
  YearPublished YEAR,
  ISBN VARCHAR(20),
  Genre VARCHAR(50),
  CopiesAvailable INT
);
```

```
INSERT INTO Books VALUES
(1, 'The Alchemist', 'Paulo Coelho', 'HarperOne', 1988, '9780061122415', 'Fiction', 5),
(2, '1984', 'George Orwell', 'Secker & Warburg', 1949, '9780451524935', 'Dystopian', 4),
(3, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Scribner', 1925, '9780743273565', 'Classic', 6),
(4, 'To Kill a Mockingbird', 'Harper Lee', 'J.B. Lippincott', 1960, '9780061120084', 'Drama', 3),
(5, 'The Hobbit', 'J.R.R. Tolkien', 'George Allen', 1937, '9780547928227', 'Fantasy', 7),
(6, 'Pride and Prejudice', 'Jane Austen', 'T. Egerton', 1813, '9780141199078', 'Romance', 5),
(7, 'The Catcher in the Rye', 'J.D. Salinger', 'Little, Brown', 1951, '9780316769488', 'Classic', 2),
(8, 'Moby Dick', 'Herman Melville', 'Harper & Brothers', 1851, '9781503280786', 'Adventure', 4);
```

2. Members Table:

```
CREATE TABLE Members (
    MemberID INT PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100),
    Phone VARCHAR(20),
    Address VARCHAR(150),
    MembershipDate DATE
);
```

```
INSERT INTO Members VALUES
(1, 'Alice Sharma', 'alice@example.com', '9876543210', 'Delhi', '2023-01-15'),
(2, 'Bob Verma', 'bob@example.com', '9876543211', 'Chandigarh', '2023-02-10'),
(3, 'Cathy Gill', 'cathy@example.com', '9876543212', 'Mumbai', '2023-03-05'),
(4, 'David Roy', 'david@example.com', '9876543213', 'Kolkata', '2023-04-20'),
(5, 'Eva Singh', 'eva@example.com', '9876543214', 'Pune', '2023-05-12'),
(6, 'Frank Thomas', 'frank@example.com', '9876543215', 'Hyderabad', '2023-06-01'),
(7, 'Grace Kaur', 'grace@example.com', '9876543216', 'Amritsar', '2023-07-22'),
(8, 'Harry Dutt', 'harry@example.com', '9876543217', 'Bangalore', '2023-08-18');
```

3. Borrowing Table

```sql
CREATE TABLE Borrowing (
  BorrowID INT PRIMARY KEY,
  MemberID INT,
  BookID INT,
  BorrowDate DATE,
  DueDate DATE,
  ReturnDate DATE,
  FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
  FOREIGN KEY (BookID) REFERENCES Books(BookID)
);
```

```sql
INSERT INTO Borrowing VALUES
(1, 1, 2, '2024-03-01', '2024-03-10', '2024-03-08'),
(2, 2, 3, '2024-03-02', '2024-03-12', NULL),
(3, 3, 5, '2024-03-05', '2024-03-15', '2024-03-14'),
(4, 4, 1, '2024-03-10', '2024-03-20', NULL),
(5, 5, 7, '2024-03-11', '2024-03-21', '2024-03-19'),
(6, 6, 6, '2024-03-13', '2024-03-23', '2024-03-21'),
(7, 7, 4, '2024-03-15', '2024-03-25', NULL),
(8, 8, 8, '2024-03-18', '2024-03-28', '2024-03-27');
```

4. Staff Table

```sql
CREATE TABLE Staff (
  StaffID INT PRIMARY KEY,
  Name VARCHAR(100),
  Position VARCHAR(50),
  Email VARCHAR(100),
  Phone VARCHAR(20)
);
```

```sql
INSERT INTO Staff VALUES
(1, 'Rita Mehra', 'Librarian', 'rita@example.com', '9123456780'),
(2, 'Sahil Bansal', 'Assistant', 'sahil@example.com', '9123456781'),
(3, 'Pooja Yadav', 'Admin', 'pooja@example.com', '9123456782'),
(4, 'Amit Rana', 'Clerk', 'amit@example.com', '9123456783'),
(5, 'Neha Roy', 'Technician', 'neha@example.com', '9123456784'),
(6, 'Manoj Das', 'Security', 'manoj@example.com', '9123456785'),
(7, 'Sunita Paul', 'Receptionist', 'sunita@example.com', '9123456786'),
(8, 'Rakesh Rao', 'Inventory Manager', 'rakesh@example.com', '9123456787');
```

5. Fines Table

```sql
CREATE TABLE Fines (
    FineID INT PRIMARY KEY,
    BorrowID INT,
    Amount DECIMAL(6,2),
    PaidStatus VARCHAR(10),
    PaymentDate DATE,
    FOREIGN KEY (BorrowID) REFERENCES Borrowing(BorrowID)
);
```

```sql
INSERT INTO Fines VALUES
(1, 1, 0.00, 'Paid', '2024-03-08'),
(2, 2, 10.00, 'Unpaid', NULL),
(3, 3, 0.00, 'Paid', '2024-03-14'),
(4, 4, 5.00, 'Unpaid', NULL),
(5, 5, 0.00, 'Paid', '2024-03-19'),
(6, 6, 0.00, 'Paid', '2024-03-21'),
(7, 7, 15.00, 'Unpaid', NULL),
(8, 8, 0.00, 'Paid', '2024-03-27');
```

6. Suppliers Table

```
CREATE TABLE Suppliers (
    SupplierID INT PRIMARY KEY,
    Name VARCHAR(100),
    Contact VARCHAR(20),
    Email VARCHAR(100),
    Address VARCHAR(150)
);
```

```
INSERT INTO Suppliers VALUES
(1, 'Book World', '9988776655', 'bookworld@example.com', 'Delhi'),
(2, 'ReadMore Ltd.', '9988776656', 'readmore@example.com', 'Mumbai'),
(3, 'Pages & Co.', '9988776657', 'pagesco@example.com', 'Bangalore'),
(4, 'Global Publishers', '9988776658', 'globalpub@example.com', 'Hyderabad'),
(5, 'LitSupply', '9988776659', 'litsupply@example.com', 'Kolkata'),
(6, 'Wordsmiths', '9988776660', 'wordsmiths@example.com', 'Chennai'),
(7, 'BookBarn', '9988776661', 'bookbarn@example.com', 'Pune'),
(8, 'Educators Hub', '9988776662', 'educators@example.com', 'Ahmedabad');
```

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

SQL QUERIES (13 Queries)

- ```
  SELECT b.Title, m.Name AS BorrowedBy
  FROM Books b
  JOIN Borrowing br ON b.BookID = br.BookID
  JOIN Members m ON br.MemberID = m.MemberID;
  ```

Title	BorrowedBy
The Alchemist	David Roy
1984	Alice Sharma
The Great Gatsby	Bob Verma
To Kill a Mockingbird	Grace Kaur
The Hobbit	Cathy Gill
Pride and Prejudice	Frank Thomas
The Catcher in the Rye	Eva Singh
Moby Dick	Harry Dutt

- ```
  SELECT Genre, COUNT(*) AS TotalBooks
  FROM Books
  GROUP BY Genre;
  ```

Genre	TotalBooks
Fiction	1
Dystopian	1
Classic	2
Drama	1
Fantasy	1
Romance	1
Adventure	1

```sql
SELECT s.Name AS SupplierName, COUNT(b.BookID) AS TotalSupplied
FROM Suppliers s
JOIN Books b ON s.SupplierID = b.BookID
GROUP BY s.Name;
```

SupplierName	TotalSupplied
Book World	1
ReadMore Ltd.	1
Pages & Co.	1
Global Publishers	1
LitSupply	1
Wordsmiths	1
BookBarn	1
Educators Hub	1

```sql
SELECT m.Name, AVG(f.Amount) AS AverageFine
FROM Members m
JOIN Borrowing br ON m.MemberID = br.MemberID
JOIN Fines f ON br.BorrowID = f.BorrowID
GROUP BY m.Name;
```

Name	AverageFine
Alice Sharma	0.000000
Bob Verma	10.000000
Cathy Gill	0.000000
David Roy	5.000000
Eva Singh	0.000000
Frank Thomas	0.000000
Grace Kaur	15.000000
Harry Dutt	0.000000

```sql
SELECT *
FROM Staff
ORDER BY Position ASC;
```

StaffID	Name	Position	Email	Phone
3	Pooja Yadav	Admin	pooja@example.com	9123456782
2	Sahil Bansal	Assistant	sahil@example.com	9123456781
4	Amit Rana	Clerk	amit@example.com	9123456783
8	Rakesh Rao	Inventory Manager	rakesh@example.com	9123456787
1	Rita Mehra	Librarian	rita@example.com	9123456780
7	Sunita Paul	Receptionist	sunita@example.com	9123456786
6	Manoj Das	Security	manoj@example.com	9123456785
5	Neha Roy	Technician	neha@example.com	9123456784
NULL	NULL	NULL	NULL	NULL

```sql
SELECT m.Name, f.Amount, f.PaidStatus
FROM Fines f
JOIN Borrowing br ON f.BorrowID = br.BorrowID
JOIN Members m ON br.MemberID = m.MemberID
WHERE f.PaidStatus = 'Unpaid';
```

Name	Amount	PaidStatus
Bob Verma	10.00	Unpaid
David Roy	5.00	Unpaid
Grace Kaur	15.00	Unpaid

```
SELECT b.Title, COUNT(br.MemberID) AS TotalBorrowed
FROM Books b
JOIN Borrowing br ON b.BookID = br.BookID
GROUP BY b.Title;
```

Title	TotalBorrowed
The Alchemist	1
1984	1
The Great Gatsby	1
To Kill a Mockingbird	1
The Hobbit	1
Pride and Prejudice	1
The Catcher in the Rye	1
Moby Dick	1

```
SELECT m.Name, b.Title, br.DueDate
FROM Borrowing br
JOIN Books b ON br.BookID = b.BookID
JOIN Members m ON br.MemberID = m.MemberID
WHERE br.ReturnDate IS NULL AND br.DueDate < CURDATE();
```

Name	Title	DueDate
Bob Verma	The Great Gatsby	2024-03-12
David Roy	The Alchemist	2024-03-20
Grace Kaur	To Kill a Mockingbird	2024-03-25

```sql
SELECT m.Name, SUM(f.Amount) AS TotalPaid
FROM Members m
JOIN Borrowing br ON m.MemberID = br.MemberID
JOIN Fines f ON br.BorrowID = f.BorrowID
WHERE f.PaidStatus = 'Paid'
GROUP BY m.Name;
```

Name	TotalPaid
Alice Sharma	0.00
Cathy Gill	0.00
Eva Singh	0.00
Frank Thomas	0.00
Harry Dutt	0.00

```sql
SELECT b.Title, COUNT(br.BookID) AS TimesBorrowed
FROM Borrowing br
JOIN Books b ON br.BookID = b.BookID
GROUP BY b.Title
ORDER BY TimesBorrowed DESC
LIMIT 1;
```

Title	TimesBorrowed
The Alchemist	1

- ```
  SELECT Name
  FROM Members
  WHERE MemberID NOT IN (SELECT DISTINCT MemberID FROM Borrowing);
  ```

Name

- ```
  SELECT Title
  FROM Books
  WHERE BookID IN (SELECT DISTINCT BookID FROM Borrowing);
  ```

Title
The Alchemist
1984
The Great Gatsby
To Kill a Mockingbird
The Hobbit
Pride and Prejudice
The Catcher in the Rye
Moby Dick

```sql
SELECT m.Name, b.Title, br.BorrowDate
FROM Borrowing br
JOIN Books b ON br.BookID = b.BookID
JOIN Members m ON br.MemberID = m.MemberID
WHERE (br.BorrowDate, m.MemberID) IN (
  SELECT MAX(BorrowDate), MemberID
  FROM Borrowing
  GROUP BY MemberID
);
```

Name	Title	BorrowDate
Alice Sharma	1984	2024-03-01
Bob Verma	The Great Gatsby	2024-03-02
Cathy Gill	The Hobbit	2024-03-05
David Roy	The Alchemist	2024-03-10
Eva Singh	The Catcher in the Rye	2024-03-11
Frank Thomas	Pride and Prejudice	2024-03-13
Grace Kaur	To Kill a Mockingbird	2024-03-15
Harry Dutt	Moby Dick	2024-03-18

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

SUMMARY

The Library Management System is a structured database project developed to manage the day-to-day operations of a library efficiently. The system focuses on storing, retrieving, and managing data related to books, members, staff, borrowing records, fines, and suppliers. By implementing a relational database using MySQL, the system ensures data integrity, reduces redundancy, and improves overall operational efficiency.

Key functionalities of the system include book tracking, member management, borrowing and returning of books, fine calculation, and report generation through SQL queries. The use of SQL operations such as **JOINs, aggregation, filtering, grouping, and subqueries** enabled us to derive meaningful insights and support decision-making.

Through the creation of six interconnected tables and the execution of various queries, the project effectively demonstrates the real-world application of database concepts. It serves as a strong foundation for further development into a more comprehensive and user-friendly library management software.

CONCLUSION

Observations:

- The Hospital Management System database successfully demonstrates the organisation and management of hospital-related data such as patients, doctors, staff, departments, appointments, billing, and pharmacy.
- The use of SQL queries allows for effective data retrieval, patient tracking, appointment scheduling, and billing generation.
- Proper relational mapping using foreign keys ensures data consistency and integrity across all entities.
- The ER diagram and schema design provide a clear and normalised structure that supports both current hospital needs and future scalability.
- Complex queries like grouping, filtering, and joining across multiple tables have been efficiently implemented.

Limitations:

- This project is limited to backend database implementation and lacks a user-friendly frontend interface for hospital staff or patients.
- The pharmacy system is not directly linked to prescriptions in medical records, so medication usage tracking is not automated.
- There is no role-based access or login system for doctors, staff, or administrators.
- Real-time alerts for low medicine stock, upcoming appointments, or unpaid bills are not part of the current scope.
- Advanced features like data analytics, reporting dashboards, or integration with real-world hospital software (e.g., EHR systems) are not included.

Future Scope

- Develop a user-friendly web or desktop interface using front-end technologies (e.g., HTML, CSS, JavaScript) for ease of use.
- Integrate SMS or email notification systems to remind users about due dates and overdue books.
- Add advanced reporting features, such as graphical dashboards, using tools like Power BI or Tableau.
- Implement authentication and authorization to restrict access to sensitive operations.
- Expand the system to support digital libraries and e-book management.