

Database Management System Final Project

Library Management System Database

University of Houston - Downtown

Final Phase

Tung Nguyen

Kinh Truong

Cooper Tran

Contribution Breakdown

Name	Contribution	Percentage
Cooper Tran	Insert Database Test Database Debug Report Drafting Format Report	33.33%
Kinh Truong	Make ER Diagram Insert Database Test Database Debug Report Drafting Error Document	33.33%
Tung Nguyen	Create Database Insert Database Test Database Debug Report Drafting Error Document	33.34%

Table of content

Table of content	3
1. Abstract.....	4
2. Mission statement.	5
3. Mission objectives.	6
4. ER diagram	7
5. Relational model.....	8
6. Use cases.....	13
7. Use case implementation	24
8. Test plan and records	33
9. Conclusion	56
10. References.....	58

1. Abstract

A database system for managing library resources at the University of Houston - Downtown has been developed collaboratively by Tung Nguyen, Kinh Truong, and Cooper Tran. This project aims to establish a Library Management System (LMS) that will assist librarians in performing multiple tasks simultaneously, such as organizing and sorting books, checking borrowing, and lending details, and collecting fines.

Through the LMS, all library processes could be managed in one place, allowing users to borrow or return books quickly without waiting in long lines. Librarians could easily add, view, update, or delete books and student information in the database using a single PC. Through the LMS, students and faculty can also locate books quickly on the shelves, making it easier to access library materials. As a library management system, the project offers all the essential features, and librarians can modify any data in the database once they are logged in. As a result of the LMS, the library management process could be streamlined and made more user-friendly, thereby saving both librarians and users time and effort.

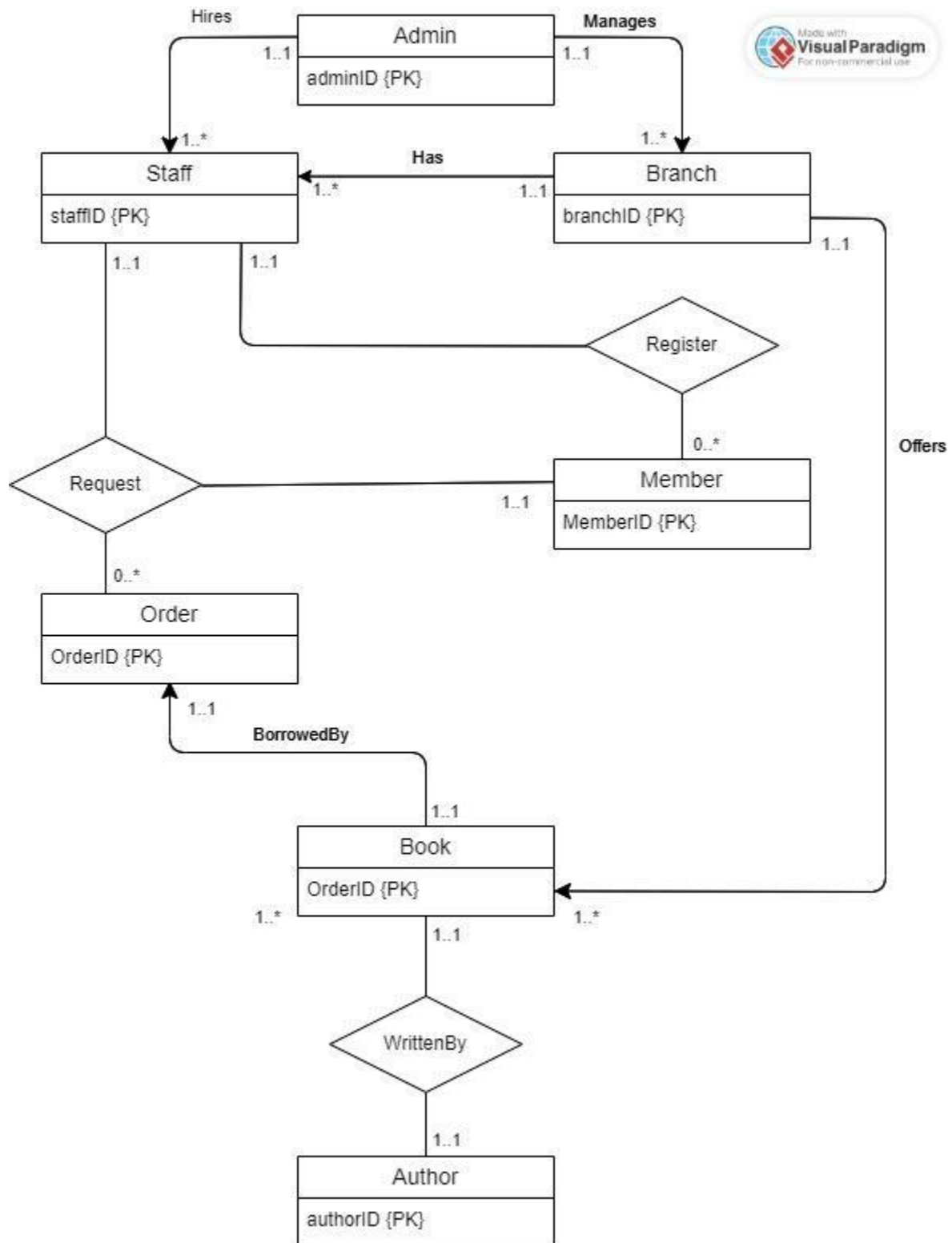
2. Mission statement.

A library database management system (DBMS) is essential for managing library resources. It facilitates librarians to easily create, update, and maintain records of library resources, and analyze data related to library usage and operations. The DBMS also increases the accuracy of tracking library items, ensuring that resources are available to patrons when needed, and provides quick access to information for users. By automating the process of checking out and returning books, the DBMS streamlines the circulation of library items, making it more organized and efficient. Overall, a library DBMS is crucial for maintaining a well-organized and effective library.

3. Mission objectives.

A Library Database Management System (LDMS) mission is to provide necessary tool for a typical library branch to manage its facilities, as well as its components and staff. Using SQL, our mission is to provide quick, accurate and easy maintenance to IT staff of the library branch. For the first user of LDMS, staff members will be uniquely identified by StaffID to categorize whether they are managing staffs, floor staffs, or warehouse staffs. For each staff, they can access full LDMS; however, only managing staffs can modify books' info and library's facilities. For next users, members, LDMS will identify each member with their unique memberID. LDMS will also hold members' information such as addresses, email, gender, email, phone numbers, and the amount of book loan they are currently under. All members' information can be accessed by all staff of the library branch along with their loanID to process. Members and staff will have the same ability to renew/cancel membership as well as request a book. For books, LDMS will identify each book by their ISBN along with their titles, authors, publishing data as well as genre. By implementing LDMS into a library, operations can turn smoothly with little to no data redundancy, providing staff and members accurate data of library content. More than that, LDMS is important for managing staff to manage the existing and future inventory.

4. ER diagram



5. Relational model

Branch (**branchID**, address, openHour, close Hour, openDay)

Branch
branchID {PK}
address
Open Hours
openHour
closeHour
openDay

Admin(**adminID**, title, lName, fName, age, sex, email, branchID)

Admin
adminID {PK}
title
Name
lName
fName
age
sex
email

Manage(**adminID**, branchID)

Manage
adminID {PK}
branchID {FK}

Normalization form/dependency: for Branch, Admin and Manage tables, there are dependencies between these tables. Manage table references a foreign key constraint from both Admin table and Branch table. However, all three tables are in 3NF as they have no transitive dependency.

Staff(staffID, title, lName, fName, age, sex, email, branchID, hiredBy)

Staff
staffID {PK}
title
Name
lName
fName
age
sex
email
branchID
{FK}

hiring(StaffID, adminID)

hiring
staffID {PK}
adminID {FK}

Normalization form/dependency: for Staff, Admin and hiring tables, there are dependencies when hiring columns consist foreign key constraints for other two tables. However, since there is no transitive dependency in them, they are all in 3NF

primaryBranch(StaffID, branchID)

primaryBranch
staffID {PK}
branchID {FK}

Normalization form/dependency: for Staff, Branch and primaryBranch tables, there are dependencies when primaryBranch columns consist foreign key

constraints from both Branch and Staff. However, since there is no transitive dependency in them, they are all in 3NF

Member(memberID, fName, lName, age, sex, DOB, email, status, hold)

Member
memberID
{PK}
name
lName
fName
age
sex
DOB
email
status
hold

Normalization form/dependency: There is no dependency in the Member table as there is no foreign key within the table. Member table is in 3NF.

Book(ISBN, genre, title, status, publishedBy, writtenBy, language, shelf)

Book
isbn {PK}
genre
location
status
publishedBy
language
shelf
writtenBy {FK}

offer(**ISBN**, branchID)

offer
isbn {PK}
branchID {FK}

Normalization form/dependency: There are dependencies in Book, offer and Branch as Offer contain foreign key constraints for other two tables. However, there is no transitive dependency within Book and Branch tables, they are in 3NF.

WrittenBy(**isbn** authorID)

WrittenBy
isbn {PK}
authorID {FK}

Author(**authorID**, fName, lName)

Register
authorID {PK}
name
fName
lName

Normalization form/dependency: There are dependencies in Book, WrittenBy, and Author tables since the WrittenBy table contains foreign key constraints from Book and Author table in case there are multiple authors in one book. However, Book and Author have no transitive dependencies. Therefore, they are in 3NF.

Request(**requestID**, memberID, status, staffID, date, time, branchID)

Request
requestID {PK}
memberID {FK}
status
staffID {FK}

date
time
branchID {FK}

Order(orderID, requestID, isbn, borrowDate, returnDate, status, dueDate, accruedFine)

Order
orderID {PK}
requestID {FK}
isbn
borrowDate
returnDate
status
dueDate
accruedFine

Normalization form/dependency: Order and Request tables have foreign constraints from Staff, Member and Book, dependencies exist. Both in Request and Order also have transitive dependencies within tables; therefore they are in 2NF

Register(username, password, assignedBy, staffID, memberID)

Register
username {PK}
password
assignedBy
staffID
memberID

Normalization form/dependency: Register table have foreign key constraints from Staff and Member tables as there are transitive dependencies among them; therefore, they are in 2NF.

6. Use cases

1/ Use case name: Appoint a new Administrator (Manager)

Actor: Administrator (Admin)

Steps:

Admin clicks on “Appoint New Manager”.

Filling form appears with a unique adminID.

Admin fill first name, last name, gender, title, date of birth and branchID where administrator will work primarily.

Admin clicks on “Appoint” to confirm.

Popup displays to confirm all details.

Admin clicks “Confirm” to appoint new administrators.

2/ Use case name: Delete an Administrator information

Actor: Administrator (Admin)

Steps:

Admin clicks on “Delete a Manager Information”.

Search for an administrator with a unique adminID.

Admin clicks on “Delete” to confirm.

Popup displays to confirm all details.

Admin clicks “Confirm” to delete all the information.

3/ Use case name: Update an Administrator’s information

Actor: Administrator (Admin)

Steps:

Admin clicks on “Update an Manager information”.

Search for an administrator with a unique adminID.

A filing form appears to update a desired section.

Once done, the admin clicks on “Update” to confirm.

Popup displays to confirm all details.

Admin clicks “Confirm” to update all the information.

4/ Use case name: Appoint new staffs

Actor: Administrator (Admin), staffs

Steps:

Admin clicks on “Appoint New Staff”.

Filling form appears with a unique staffID.

Admin fill first name, last name, gender, title, date of birth and branchID.

Admin clicks on “Appoint” to confirm.

Popup displays to confirm all details.

Admin clicks “Confirm” to appoint new staffs

5/ Use case name: Lay off a staff

Actor: Administrator (Admin), staffs

Steps:

Admin clicks on “Delete a Staff”.

The admin searches for staff with a unique staffID.

Admin clicks on “Delete” to confirm.

Popup displays to confirm the decision.

Admin clicks “Confirm” to complete the task.

6/ Use case name: Update a staff’s information

Actor: Administrator (Admin), staffs

Steps:

Admin clicks on “Update a staff’s information”.

Search for a staff with a unique staffID.

A filing form appears to update a desired section.

Once done, the admin clicks on “Update” to confirm.

Popup displays to confirm all details.

Admin clicks “Confirm” to update all the information.

7/ Use case name: Add new book

Actor: Staffs

Steps:

Staff clicks on “Add new book”.

Filling form appears.

Staff fill out a book ISBN, title, author, publisher, genre and date.

Staff clicks on “Add” to confirm.

Popup appears to validate Staffs.

Staff clicks “Confirm” to add new books.

8/ Use case name: Delete a book’s information

Actor: Staffs

Steps:

Staff clicks on “Delete a book’s information”

Staff looks up the book’s information using ISBN.

Staff clicks on “Delete” to confirm.

Popup displays to confirm the decision.

Staff clicks “Confirm” to complete the task.

9/ Use case name: Update a book’s information

Actor: Staffs

Steps:

Staff clicks on “Update a book’s information”.

Staff search for a book with its ISBN number.

A filing form appears to update a desired section.

Once done, staff clicks on “Update” to confirm.

Popup displays to confirm all details.

Staff clicks “Confirm” to update all the information.

10/ Use case name: Register new member

Actor: Staffs

Steps:

Staff clicks on “Register new member”.

Filling form appears with a unique memberID.

Staff fill first name, last name, DOB, gender, email, phone numbers, expiration date and status.

Staff clicks “Confirm” to add new members.

11/ Use case name: Request a book loan

Actor: Members:

Steps:

Members login with their unique credentials (memberID)

Members search for books using ISBN or titles.

Once a book is found, members click on “Request borrowing this book”.

A popup appears to confirm members want to borrow this book.

Members click “Confirm request” to send out the request to staff.

12/ Use case name: Approve/Deny book loan request from members

Actor: Staffs

Steps:

Staff click “Request Report” to check new requests.

Staff clicks on desired requests.

Popup appears with memberID, first name, last name, status, book status and loanID.

Staff verify all the information and book status.

Staff click “Approve” or “Deny” processing desired loan requests.

Book status will change to “Unavailable” if approved otherwise stay “Available” if denied.

13/ Use case name: Renew/Cancel membership

Actor: Members

Steps:

Members login using their unique login credential (memberID)

Members click on “View info”.

Members check if their expiration date is near or passed.

Members click “Renew” or “Cancel” based on their choice.

Popup appears to confirm.

Members click “Confirm”.

14/ Use case name: Insert Member

Actor: Staffs

Steps:

Staff click on “Insert Member”.

Filling form appears with a unique memberID.

Staff fill first name, last name, DOB, gender, email, phone numbers, expiration date and status.

Staff click “Confirm” to add new members.

15/ Use case name: Delete Member

Actor: Staffs

Steps:

Staff click on “Delete Member”.

Staffs search for a member with a unique memberID.

Staff click on “Delete” to confirm.

Popup displays to confirm the decision.

Staff click “Confirm” to complete the task.

16/ Use case name: Update Member

Actor: Staffs

Steps:

Staff click on “Update Member”.

Staffs search for a member with a unique memberID.

A filing form appears to update a desired section.

Once done, staffs click on “Update” to confirm.

Popup displays to confirm all details.

Staffs click “Confirm” to update all the information.

17/ Use case name: Insert Branch

Actor: Administrator (Admin)

Steps:

Admin clicks on “Insert Branch”.

Filling form appears with a unique branchID.

Admin fills branch name, address, city, and postal code.

Admin clicks “Confirm” to add a new branch.

18/ Use case name: Delete Branch

Actor: Administrator (Admin)

Steps:

- Admin clicks on “Delete Branch”.
- Admin searches for a branch with a unique branchID.
- Admin clicks on “Delete” to confirm.
- Popup displays to confirm the decision.
- Admin clicks “Confirm” to complete the task.

19/ Use case name: Update Branch

Actor: Administrator (Admin)

Steps:

- Admin clicks on “Update Branch”.
- Admin searches for a branch with a unique branchID.
- A filing form appears to update a desired section.
- Once done, the admin clicks on “Update” to confirm.
- Popup displays to confirm all details.
- Admin clicks “Confirm” to update all the information

20/ Use case name: Insert Request Book

Actor: Members

Steps:

- Members login with their unique credentials (memberID).
- Members search for a book using ISBN or titles.
- Once the book is found, members click on “Request Borrowing This Book”.
- A popup appears to confirm members want to borrow the book.
- Members click “Confirm Request” to send out the request

21/ Use case name: Delete Request Book

Actor: Members

Steps:

- Members login with their unique credentials (memberID).
- Members click on “View My Requests”.
- Members search for a request using the book's ISBN or title.
- Once the request is found, members click on “Delete Request”.
- Popup displays to confirm the decision.

Members click “Confirm” to complete the task

22/ Use case name: Update Request Book

Actor: Members

Steps:

Members login with their unique credentials (memberID).

Members click on “View My Requests”.

Members search for a request using the book's ISBN or title.

Once the request is found, members click on “Update Request”.

A filing form appears to update the desired section.

Once done, members click on “Update” to confirm.

Popup displays to confirm all details.

Members click “Confirm” to update the request.

23/ Use case name: Insert Order

Actor: Staffs

Steps:

Staff click on “Insert Order”.

Filling form appears with a unique orderID.

Staff fill out the ISBN, supplier, quantity, and order date.

Staff click on “Confirm” to add a new order.

24/ Use case name: Delete Order

Actor: Staffs

Steps:

Staff click on “Delete Order”.

Staffs search for an order with a unique orderID.

Staff click on “Delete” to confirm.

Popup displays to confirm the decision.

Staff click “Confirm” to complete the task.

25/ Use case name: Update Order

Actor: Staffs

Steps:

Staff click on “Update Order”.

Staffs search for an order with a unique orderID.

A filing form appears to update a desired section.

Once done, staff click on “Update” to confirm.

Popup displays to confirm all details.

Staff click “Confirm” to update the order.

26/ Use case name: Insert Register

Actor: Staffs

Steps:

Staffs click on “Insert Register”.

Filling form appears with a unique registerID.

Staffs fill in the date, memberID, staffID, and branchID.

Staffs click on “Confirm” to add new register.

27/ Use case name: Delete Register

Actor: Staffs

Steps:

Staffs click on “Delete Register”.

Staffs search for a register with a unique registerID.

Staffs click on “Delete” to confirm.

Popup displays to confirm the decision.

Staffs click “Confirm” to complete the task.

28/ Use case name: Update Register

Actor: Staffs

Steps:

Staffs click on “Update Register”.

Staffs search for a register with a unique registerID.

A filing form appears to update a desired section.

Once done, staff click on “Update” to confirm.

Popup displays to confirm all details.

Staff click “Confirm” to update the register.

29/ Use case name: Determine where the book locates and the status of that book

Actor: Staffs, Members

Steps:

Staffs or members click on “Search for Book”

Staffs and members use the book’s unique ISBN to locate the book.

Staff and members should be able to see a popup display with the ISBN, title, location and the status of the book.

Staffs and members can either check out or search for another book.

Book’s status will change to On loan if members or staffs check out.

30/ Use case name: Determine who hires staffs to coordinate staffs

Actor: Administrator (Manager)

Steps:

Admin clicks on Manage Staffs

A popup will display a list of staffs with their information.

Administrators can check staff’s history including when they got hired and who hired them.

If an administrator wants to coordinate staff in case of staff shortage, the administrator can check the location using the staff location function.

31/ Use case name: Check who wrote books in the library inventory

Actor: Staffs, members

Steps:

Staff and members can search for their desired book using its unique ISBN.

Staff and members will click on “Get more information” icon to expand book’s author section

A popup will display the book's author.

32/ Use case name: Report the average age of staff for census

Actor: Administrator (Manager)

Steps:

Administrator clicks on “Manage Staff”.

Administrator highlights “Age” column.

Administrator choose “Export to Excel file”

The Excel file will contain the average age for Administrator.

33/ Use case name: Print our total open hour of each branch for advertisement

Actor: Staffs

Steps:

Staff click on “Library Info”

Staff click on “Print Open Hour”

Software will print out the total open hour of each branch in an Excel File.

34/ Use case name: Determine how many books there are in the library

Actor: Staffs, Administrators (Manager)

Steps:

Staff, Administrators click on “Inventory”.

Staff, Administrators click on “Book”.

Staff, Administrators should be able to see the total of books within the UI of the library software.

35/ Use case name: Report the number of females and males administrator within the library system for a census

Actor: Administrators (Manager)

Steps:

Administrator clicks on “Manage Roster”.

Administrator highlights “Sex” column.

Administrator choose “Export to Excel file”

The Excel file will contain how many female and male administrators are in the library system.

36/ Use case name: Check the most expensive accrued fine in the library system

Actor: Staff

Steps:

Staff clicks on “Request and Order”

Staff clicks on “Generate an order report”

Report will contain the highest accrued fine in the system along with the member's name, what order they requested.

37/ Use case name: Determine total request by status inspecific period of time

Actor: Administrators

Steps:

Administrators click on “Report”

Administrators click on “Order”

Administrators choose “count request by status”

Administrators inputs period of time

Administrators should see the number of request in different status in the chosen period of time

38/ Use case name: Determine total author

Actor: Administrators

Steps:

Administrators click on “Report”

Administrators click on “Book”

Administrators choose “count number of authors”

Administrators should see the total amount of author

39/ Use case name: Determine which admin manage which library branch

Actor: Administrators

Steps:

Administrators click on “Report”

Administrators click on “Admin”

Administrators choose “view admin with branch”

Administrators should see list of admin, with the address of library that they manage

40/ Use case name: Determine total of staff with specific title in each branch

Actor: Administrators

Steps:

Administrators click on “Report”

Administrators click on “Staff”

Administrators choose “view total of employee at each branch by title”

Administrators should see the total of employee in each position

41/ Use case name: View order detail

Actor: Staffs, Members, Administrators

Steps:

Staffs, Members, Administrators click on “Order”

Staffs, Members, Administrators click on “View Order”

Staffs, Members, Administrators click on specific order to view order detail

Staffs, Members, Administrators should be able to see the order in detail

7. Use case implementation

1/ Use case name: Appoint a new Administrator (Manager)

Implementation:

```
INSERT INTO Admin (adminID, title, lName, fName, age, sex, email,  
branchID)
```

```
VALUES
```

```
('Ad01', 'General Manager', 'Johnson', 'Jessica', 35, 'F',  
'jessica.johnson@library.com', 'B01');
```

2/ Use case name: Delete an Administrator information

Implementation:

```
DELETE FROM Admin
```

```
WHERE adminID = 'Ad01';
```

3/ Use case name: Update an Administrator’s information (in this case: Job Title)

Implementation:

```
UPDATE Admin
```

```
SET title = 'Executive Manager'
```

```
WHERE adminID = 'Ad01';
```

4/ Use case name: Hire a new staff

Implementation:

```
INSERT INTO Staff (staffID, title, lName, fName, age, sex, email,  
branchID, hiredBy)
```

```
VALUES
```



```
('S01', 'Floor Staff', 'Lee', 'James', 25, 'M', 'james.lee@library.com',  
'B01','Ad01');
```

5/ Use case name: = Lay off a staff

Implementation:

```
DELETE FROM Staff  
WHERE staffID = 'S01';
```

6/ Use case name: Update a staff's information (in this case: Last name)

Implementation:

```
UPDATE Staff  
SET IName = 'Holland'  
WHERE staffID = 'S01';
```

7/ Use case name: Appoint a new book

Implementation:

```
INSERT INTO Book (isbn, genre, title, branchID, status, publishedBy,  
language, shelf)  
VALUES  
( '9781408894705', 'Fiction', 'The Testaments', 'B01', 'Available', 'Random  
House', 'English', 'Fiction A-Z');
```

8/ Use case name: Delete a book

Implementation:

```
DELETE FROM book  
WHERE isbn = '9781408894705';
```

9/ Use case name: Update an Book's information (in this case: Book's genre)

Implementation:

```
UPDATE Book  
SET genre = 'Historical Fiction'
```

WHERE isbn = '9781408894705';

10/ Use case name: Register new member

Implementation:

```
INSERT INTO Member (memberID, lName, fName, age, sex, DOB, email,
status, hold)
VALUES ('M01', 'Smith', 'John', 30, 'M', '1998-01-15', '555-1234',
'john.smith@email.com', 'Active' '0');
```

11/ Use case name: Request a book loan

Implementation:

```
INSERT INTO Request (requestID, memberID, status, staffID, date, time,
branchID)
VALUES ('R01', 'M02', 'Pending', 'S03', '2023-01-20', '00:00:13', 'B05');
```

12/ Use case name: Approve/Deny book loan request from members

Implementation:

```
UPDATE Request
SET status = 'Approved'
WHERE requestID = 'R01';
```

13/ Use case name: Renew/Cancel membership

Implementation:

```
UPDATE Member
SET membershipStatus = 'Expired'
WHERE memberID = 'M01';
```

14/ Use case name: Insert Member

Implementation:

```
INSERT INTO Member (memberID, lName, fName, age, sex, DOB, email,
status, hold)
```

```
VALUES ('M02', 'Doe', 'Jane', 30, 'F', '1993-04-19', 'janedoe@email.com',  
'Active', 'False');
```

15/ Use case name: Delete Member

Implementation:

```
DELETE FROM Member  
WHERE memberID = 'M02';
```

16/ Use case name: Update Member

Implementation:

```
UPDATE Member  
SET age= 29  
WHERE `= 'M01';
```

17/ Use case name: Insert Branch

Implementation:

```
INSERT INTO Branch (branchID, address, openHour, closeHour)  
VALUES ('B02', '456 Oak St, Houston, USA', '10:00:00', '18:00:00', 'Mon-Sat');
```

18/ Use case name: Delete Branch

Implementation:

```
DELETE FROM Branch  
WHERE branchID = 'B02';
```

19/ Use case name: Update Branch

Implementation:

```
UPDATE Branch  
SET address = '789 Oak Ave, Houston, USA'  
WHERE branchID = 'B02';
```

20/ Use case name: Insert Request Book

Implementation:

```
INSERT INTO Request (requestID, memberID, status, staffID, date, time,
branchID)
VALUES ('R02', M13, Pending, 'S01', '2023-01-21', '00:00:15', 'B05');
```

21/ Use case name: Delete Request Book

Implementation:

```
DELETE FROM Request
WHERE requestID = 'R02';
```

22/ Use case name: Update Request Book

Implementation:

```
UPDATE Request
SET status = 'Approved'
WHERE requestID = 'R01';
```

23/ Use case name: Insert Order

Implementation:

```
INSERT INTO Order (requestID, isbn, borrowDate, returnDate, status,
orderID, accruedFine)
VALUES ('C02', 'R03', '9780061231327', '2023-01-22', '2023-01-23',
'Returned', '2023-01-29', '0.00');
```

24/ Use case name: Delete Order

Implementation:

```
DELETE FROM Order
WHERE orderID = 'C02';
```

25/ Use case name: Update Order

Implementation:

```
UPDATE Order
```

```
SET accruedFine= 40.00  
WHERE orderID = 'C01';
```

26/ Use case name: Insert Register

Implementation:

```
INSERT INTO Register (username, password, staffID, memberID)  
VALUES ('Adams.James.456', 'MemberJames60', NULL, 'M20')
```

27/ Use case name: Delete Register

Implementation:

```
DELETE FROM Register  
WHERE memberID= 'M20';
```

28/ Use case name: Update Register

Implementation:

```
UPDATE Register  
SET passowrd= 'MemberJames4698!'  
WHERE memberID= 'M20';
```

29/ Use case name: Determine where the book locates and the status of that book

Implementation:

```
SELECT Book.isbn, book.title, book.status, Branch.branchID AS availableAt,  
Branch.address  
FROM Offer  
INNER JOIN Book ON Offer.isbn = Book.isbn  
INNER JOIN Branch ON Offer.branchID = Branch.branchID;
```

30/ Use case name: Determine who hires staffs to coordinate staffs

Implementation:

```
SELECT Admin.adminID, Admin.title, Admin.lName, Admin.fName,  
Staff.staffID AS hiredBy, Staff.title AS hiredTitle, Staff.lName, Staff.fName
```

```
FROM Hiring
INNER JOIN Staff ON Hiring.staffID = Staff.staffID
INNER JOIN Admin ON Hiring.adminID = Admin.adminID;
```

31/ Use case name: Check who wrote books in the library inventory

Implementation:

```
SELECT Book.isbn, Book.title, Author.authorID, Author.lName,
Author.fName
FROM WrittenBy
INNER JOIN Book ON WrittenBy.isbn = Book.isbn
INNER JOIN Author ON WrittenBy.authorID = Author.authorID;
```

32/ Use case name: Report the average age of staff for a census

Implementation:

```
SELECT AVG(age) AS average_age_staff
FROM Staff;
```

33/ Use case name: Print our total open hour of each branch for advertisement

Implementation:

```
SELECT branchID, closeHour, openHour, SEC_TO_TIME( SUM(
TIME_TO_SEC( closeHour - openHour ) ) ) AS total_Open_Hour
FROM Branch
GROUP BY branchID;
```

34/ Use case name: Determine how many books there are in the library

Implementation:

```
SELECT COUNT(isbn) AS total_book
FROM Book;
```

35/ Use case name: Report the number of females and males administrator within the library system for a census

Implementation:

```
SELECT sex, COUNT(*) AS total  
FROM Admin  
GROUP BY sex;
```

36/ Use case name: Check the most expensive accrued fine in the library system

Implementation:

```
SELECT orderID, requestID, accruedFine  
FROM `Order`  
WHERE accruedFine = (SELECT MAX(accuredFine) FROM `Order`);
```

37/ Use case name: Determine total request by status inspecific period of time

Implementation:

```
SELECT status, COUNT(*) as count  
FROM Request  
WHERE date >= '2023-01-01' AND date <= '2023-03-31'  
GROUP BY status;
```

38/ Use case name: Determine total author

Implementation:

```
SELECT COUNT(DISTINCT CONCAT(lName, fName)) as TotalAuthor  
FROM Author;
```

39/ Use case name: Determine which admin manage which library branch

Implementation:

```
SELECT  
    m.adminID,  
    a.fName AS adminFristName,  
    a.lName AS adminLastName,  
    a.title AS title,  
    m.branchID,
```

```
        b.address AS branchAddress
FROM
    Manage m
    INNER JOIN Admin a ON m.adminID = a.adminID
    INNER JOIN Branch b on m.branchID = b.branchID
```

40/ Use case name: Determine total of staff with specific title in each branch

Implementation:

```
SELECT
    p.branchID,
    b.address AS branchAddress,
    s.title as staffTitle,
    COUNT(*) AS numberOfEmployee
FROM
    primaryBranch p
    INNER JOIN Staff s ON p.staffID = s.staffID
    INNER JOIN Branch b on p.branchID = b.branchID
GROUP BY p.branchID, s.title;
```

41/ Use case name: View order detail

Implementation:

```
SELECT
    o.orderID,
    r.requestID,
    br.address AS LibraryAddress,
    s.staffID,
    s.fName AS staffFristName,
    s.lName AS staffLastName,
    m.memberID,
    m.fName AS memberFristName,
    m.lName AS memberLastName,
```



```

r.date AS requestDate,
r.time AS requestTime,
b.isbn AS bookISBN,
b.title AS bookTitle,
o.borrowDate,
o.dueDate,
o.returnDate,
o.status AS orderStatus,
o.accuredFine
FROM `Order` o
INNER JOIN Request r on o.requestID = r.requestID
INNER JOIN Book b on o.isbn = b.isbn
INNER JOIN Member m on r.memberID = m.memberID
INNER JOIN Staff s on r.staffID = s.staffID
INNER JOIN Branch br on r.branchID = br.branchID;

```

8. Test plan and records

1/ **Use case name:** Appoint a new Administrator (Manager)

Implementation:

```

INSERT INTO Admin (adminID, title, lName, fName, age, sex, email,
branchID)
VALUES
('Ad01', 'General Manager', 'Johnson', 'Jessica', 35, 'F',
'jessica.johnson@library.com', 'B01');

```

Expected Output: Database Management System (DBS) should be able to take in new information like below figure

	adminID	title	lName	fName	age	sex	email
▶	Ad01	General Manager	Johnson	Jessica	35	F	jessica.johnson@library.com

Result Output: As expected, mySQL Workbench executed scripts successfully with above output.

2/ **Use case name:** Delete an Administrator information

Implementation:

```
DELETE FROM Admin
```

```
WHERE adminID = 'Ad01';
```

Expected Output: Database Management System (DBS) should be able to delete row associating “Ad01”

	adminID	title	lName	fName	age	sex	email
	Ad01	General Manager	Johnson	Jessica	35	F	jessica.johnson@library.com
	Ad02	General Manager	Smith	John	40	M	john.smith@library.com

Result Output: As expected, mySQL Workbench executed scripts successfully in deleting row associating “Ad01”

	adminID	title	lName	fName	age	sex	email
	Ad02	General Manager	Smith	John	40	M	john.smith@library.com

3/ Use case name: Update an Administrator’s information (in this case: Job Title)

Implementation:

```
UPDATE Admin
```

```
SET title = 'Executive Manager'
```

```
WHERE adminID = 'Ad01';
```

Expected Output: Database Management System (DBS) should be able to take in new information to replace own information (“General Manager” of “Ad01” changes to “Executive Manager”

	adminID	title	lName	fName	age	sex	email
▶	Ad01	General Manager	Johnson	Jessica	35	F	jessica.johnson@library.com

Result Output: As expected, mySQL Workbench executed scripts successfully in replacing job title of “Ad01” to “Executive Manager” as shown below.

	adminID	title	lName	fName	age	sex	email
	Ad01	Executive Manager	Johnson	Jessica	35	F	jessica.johnson@library.com

4/ Use case name: Hire a new staff

Implementation:

```

INSERT INTO Staff (staffID, title, lName, fName, age, sex, email,
branchID,hiredBy)
VALUES
('S01', 'Floor Staff', 'Lee', 'James', 25, 'M', 'james.lee@library.com',
'B01','Ad01');

```

Expected Output: DBS should be able to take in new information for Staff table

Result Output: DBS executes script successfully with new information associating with “S01” as shown below.

staffID	title	lName	fName	age	sex	email
S01	Floor Staff	Lee	James	25	M	james.lee@library.com

5/ Use case name: = Lay off a staff

Implementation:

```
DELETE FROM Staff
```

```
WHERE staffID = 'S01';
```

Expected Output: DBS should be able to delete row associating “S01”

staffID	title	lName	fName	age	sex	email
S01	Floor Staff	Lee	James	25	M	james.lee@library.com
S02	IT Staff	Wang	Linda	30	F	linda.wang@library.com

Result Output: As expected, DBS executed scripts successfully in deleting row associating “S01”

staffID	title	lName	fName	age	sex	email
S02	IT Staff	Wang	Linda	30	F	linda.wang@library.com

6/ Use case name: Update a staff’s information (in this case: Last name)

Implementation:

```
UPDATE Staff
```

```
SET lName = 'Holland'
```

```
WHERE staffID = 'S01';
```

Expected Output: DBS should be able to take in new information for staff with staffID “S01”

staffID	title	lName	fName	age	sex	email
S01	Floor Staff	Lee	James	25	M	james.lee@library.com

Result Output: As expected, DBS executes scripts successfully with new information in lName for staffID “S01”

staffID	title	lName	fName	age	sex	email
S01	Floor Staff	Holland	James	25	M	james.lee@library.com

7/ Use case name: Add a new book

Implementation:

INSERT INTO Book (isbn, genre, title, branchID, status, publishedBy,
language, shelf)

VALUES

('9781408894705', 'Fiction', 'The Testaments', 'B01', 'Available', 'Random
House', 'English', 'Fiction A-Z');

Expected Output: DBS should be able to take in new information for the Book table.

Result Output: DBS executes script successfully with new information associating with ISBN “9780060883287” as shown below.

isbn	genre	title	status	publishedBy	language	shelf
9780060883287	Fiction	One Hundred Years of Solitude	Available	Harper Perennial	English	Fiction

8/ Use case name: Delete a book

Implementation:

DELETE FROM book

WHERE isbn = '9781408894705';

Expected Output: DBS should be able to delete row associating “9780060883287” off the inventory

isbn	genre	title	status	publishedBy	language	shelf
9780060883287	Fiction	One Hundred Years of Solitude	Available	Harper Perennial	English	Fiction
9780061120084	Fiction	To Kill A Mockingbird	On Loan	HarperCollins	English	Fiction

Result Output: As expected, DBS executed scripts successfully in deleting row associating “9780060883287” off the book inventory

isbn	genre	title	status	publishedBy	language	shelf
9780061120084	Fiction	To Kill A Mockingbird	On Loan	HarperCollins	English	Fiction

9/ Use case name: Update an Book’s information (in this case: Book’s genre)

Implementation:

UPDATE Book

SET genre = 'Historical Fiction'

WHERE isbn = '9781408894705';

Expected Output: DBS should be able to take in new information for Book table with ISBN “9781408894705”

isbn	genre	title	status	publishedBy	language	shelf
9780060883287	Fiction	One Hundred Years of Solitude	Available	Harper Perennial	English	Fiction

Result Output: As expected, DBS executes scripts successfully with new book’s genre as “Historical Fiction” for ISBN “9781408894705”

isbn	genre	title	status	publishedBy	language	shelf
9780060883287	Historical Fiction	One Hundred Years of Solitude	Available	Harper Perennial	English	Fiction

10/ Use case name: Register new member

Implementation:

INSERT INTO Member (memberID, lName, fName, age, sex, DOB, email, status, hold)

VALUES ('M01', 'Smith', 'John', 30, 'M', '1998-01-15', '555-1234',

john.smith@email.com, ‘Active’ ‘0’);

Expected Output: DBS should be able to insert a new member into the Member table.

Result Output: DBS executes the script successfully, inserting the new member with memberID "M01" as shown below.

memberID	lName	fName	age	sex	DOB	email	status	hold
M01	Smith	John	25	M	1998-01-15	johnsmith@email.com	Active	0

11/ Use case name: Request a book loan

Implementation:

```
INSERT INTO Request (requestID, memberID, status, staffID, date, time,  
branchID)
```

```
VALUES ('R01', 'M02', 'Pending', 'S03', '2023-01-20', '00:00:13', 'B05');
```

Expected Output: DBS should be able to insert a new request into the Request table.

Result Output: DBS executes the script successfully, inserting the new request with requestID "R01" as shown below.

Result Grid

Filter Rows:

Edit:

Export/Import:

requestID	memberID	status	staffID	date	time	branchID
R01	M02	Pending	S03	2023-01-20	00:00:13	B05

12/ Use case name: Approve/Deny book loan request from members

Implementation:

```
UPDATE Request
```

```
SET status = 'Approved'
```

```
WHERE requestID = 'R01';
```

Expected Output: DBS should be able to update the status of the request with requestID "R01" to "Approved".

Result Output: As expected, DBS executes the script successfully, updating the status of the request with requestID "R01" to "Approved" as shown below.

Result Grid

Filter Rows:

Edit:

Export/Import

	requestID	memberID	status	staffID	date	time	branchID
	R01	M02	Approved	S03	2023-01-20	00:00:13	B05

13/ Use case name: Renew/Cancel membership

Implementation:

```
UPDATE Member
```

```
SET membershipStatus = 'Expired'
```

```
WHERE memberID = 'M01';
```

Expected Output: DBS should be able to update the membership status of the member with memberID "M01" to "Expired".

Result Output: As expected, DBS executes the script successfully, updating the membership status of the member with memberID "M01" to "Expired" as shown below.

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

memberID	lName	fName	age	sex	DOB	email	status	hold
M01	Smith	John	25	M	1998-01-15	johnsmith@email.com	Expired	0

14/ Use case name: Insert Member

Implementation:

```
INSERT INTO Member (memberID, lName, fName, age, sex, DOB, email, status, hold)
```

```
VALUES ('M02', 'Doe', 'Jane', 30, 'F', '1993-04-19', 'janedoe@email.com', 'Active', 'False');
```

Expected Output: DBS should be able to insert a new member into the Member table.

Result Output: DBS executes the script successfully, inserting the new member with memberID "M02" as shown below.

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Conte

	memberID	lName	fName	age	sex	DOB	email	status	hold
	M01	Smith	John	25	M	1998-01-15	johnsmith@email.com	Expired	0
	M02	Doe	Jane	30	F	1993-04-19	janedoe@email.com	Active	0

15/ Use case name: Delete Member

Implementation:

```
DELETE FROM Member
```

```
WHERE memberID = 'M02';
```

Expected Output: DBS should be able to delete the member with memberID "M02" from the Member table.

Result Output: As expected, DBS executes the script successfully, deleting the member with memberID "M02" from the Member table as shown below.

Result Grid		Filter Rows:		Edit:		Export/Import:		Wrap Cell Content:	
	memberID	lName	fName	age	sex	DOB	email	status	hold
	M01	Smith	John	25	M	1998-01-15	johnsmith@email.com	Expired	0
	M03	Williams	James	42	M	1981-07-27	jameswilliams@email.com	Active	0

16/ Use case name: Update Member

Implementation:

UPDATE Member

SET age= 29

WHERE `= 'M01';

Expected Output: DBS should be able to update the age of the member with memberID "M01" to "29".

Result Output: As expected, DBS executes the script successfully, updating the age of the member with memberID "M01" to "29" as shown below.

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Cont

	memberID	lName	fName	age	sex	DOB	email	status	hold
	M01	Smith	John	29	M	1998-01-15	johnsmith@email.com	Expired	0

17/ Use case name: Insert Branch






Implementation:

INSERT INTO Branch (branchID, address, openHour, closeHour)

VALUES ('B02','456 Oak St, Houston, USA', 10:00:00', '18:00:00', 'Mon-Sat);

Expected Output: DBS should be able to insert a new branch into the Branch table.

Result Output: DBS executes the script successfully, inserting the new branch with branchID "B02" as shown below.

Result Grid			Filter Rows:	<input type="text"/>	Edit:				Export/Import:
	branchID	address	openHour	closeHour	openDay				
▶	B01	123 Main St, Houston, USA	09:00:00	17:00:00	Mon-Fri				
	B02	456 Oak Ave, Houston, USA	10:00:00	18:00:00	Mon-Sat				

18/ Use case name: Delete Branch

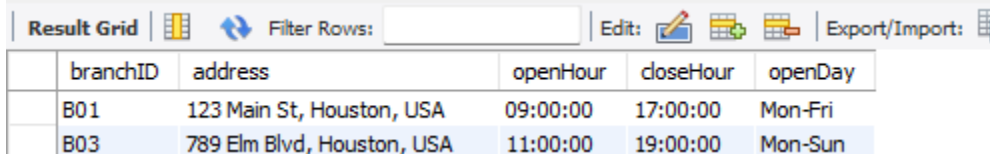
Implementation:

DELETE FROM Branch

WHERE branchID = 'B02';

Expected Output: DBS should be able to delete the branch with branchID "B02" from the Branch table.

Result Output: As expected, DBS executes the script successfully, deleting the branch with branchID "B02" from the Branch table as shown below.



branchID	address	openHour	closeHour	openDay
B01	123 Main St, Houston, USA	09:00:00	17:00:00	Mon-Fri
B03	789 Elm Blvd, Houston, USA	11:00:00	19:00:00	Mon-Sun

19/ Use case name: Update Branch

Implementation:

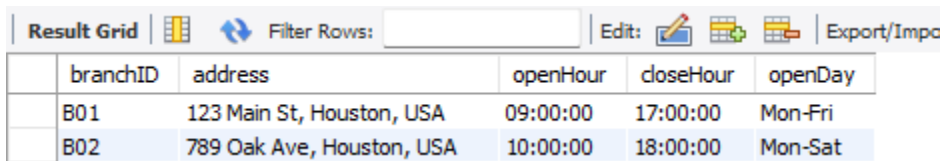
UPDATE Branch

SET address = '789 Oak Ave, Houston, USA'

WHERE branchID = 'B02';

Expected Output: DBS should be able to update the address of the branch with branchID "B02" to "789 Oak Ave, Houston, USA".

Result Output: As expected, DBS executes the script successfully, updating the address of the branch with branchID "B02" to "789 Oak Ave, Houston, USA" as shown below.



branchID	address	openHour	closeHour	openDay
B01	123 Main St, Houston, USA	09:00:00	17:00:00	Mon-Fri
B02	789 Oak Ave, Houston, USA	10:00:00	18:00:00	Mon-Sat

20/ Use case name: Insert Request Book

Implementation:

INSERT INTO Request (requestID, memberID, status, staffID, date, time, branchID)

VALUES ('R02', M13, Pending, 'S01', '2023-01-21', '00:00:15', 'B05');

Expected Output: DBS should be able to insert a new request into the Request table.

Result Output: DBS executes the script

Result Grid							
Filter Rows:							
	requestID	memberID	status	staffID	date	time	branchID
	R01	M02	Approved	S03	2023-01-20	00:00:13	B05
	R02	M13	Pending	S01	2023-01-21	00:00:15	B05

21/ Use case name: Delete Request Book

Implementation:

DELETE FROM Request

WHERE requestID = 'R02';

Expected Output: DBS should be able to delete the request with requestID "R02" from the Request table.

Result Output: As expected, DBS executes the script successfully, deleting the request with requestID "R02" from the Request table as shown below.

Result Grid							
Filter Rows:							
	requestID	memberID	status	staffID	date	time	branchID
	R01	M02	Approved	S03	2023-01-20	00:00:13	B05
	R03	M03	Approved	S02	2023-01-22	00:00:10	B02

22/ Use case name: Update Request Book

Implementation:

UPDATE Request

SET status = 'Approved'

WHERE requestID = 'R01';

Expected Output: DBS should be able to update the status of the request with requestID "R01" to "Approved".

Result Output: As expected, DBS executes the script successfully, updating the status of the request with requestID "R01" to "Approved" as shown below.

Result Grid							
Filter Rows:							
	requestID	memberID	status	staffID	date	time	branchID
	R01	M02	Approved	S03	2023-01-20	00:00:13	B05

23/ Use case name: Insert Order

Implementation:

```
INSERT INTO Order (requestID, isbn, borrowDate, returnDate, status,
orderID, accruedFine)
VALUES ('C02', 'R03', '9780061231327', '2023-01-22', '2023-01-23',
'Returned', '2023-01-29', '0.00');
```

Expected Output: DBS should be able to insert a new order into the Order table.

Result Output: DBS executes the script successfully, inserting the new order with orderID "C02" as shown below.

Result Grid								
Filter Rows: <input type="text"/>								
Edit: Export/Import: Wrap Cell Content:								
	orderID	requestID	isbn	borrowDate	returnDate	status	dueDate	accuredFine
	C01	R01	9780743273565	2023-01-20	2023-01-29	Returned	2023-01-27	20.00
	C02	R03	9780061231327	2023-01-22	2023-01-23	Returned	2023-01-29	0.00

24/ Use case name: Delete Order

Implementation:

```
DELETE FROM Order
WHERE orderID = 'C02';
```

Expected Output: DBS should be able to delete the order with orderID "C02" from the Order table.

Result Output: As expected, DBS executes the script successfully, deleting the order with orderID "C02" from the Order table as shown below.

Result Grid								
Filter Rows: <input type="text"/>								
Edit: Export/Import: Wrap Cell Content:								
	orderID	requestID	isbn	borrowDate	returnDate	status	dueDate	accuredFine
▶	C01	R01	9780743273565	2023-01-20	2023-01-29	Returned	2023-01-27	20.00
	C03	R05	9780199216819	2023-02-10	2023-02-24	Returned	2023-02-17	70.00

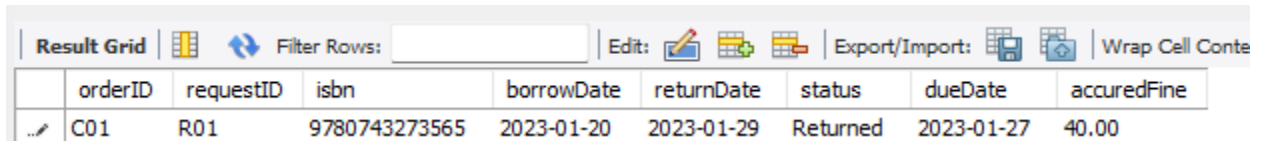
25/ Use case name: Update Order

Implementation:

```
UPDATE Order
SET accruedFine= 40.00
WHERE orderID = 'C01';
```

Expected Output: DBS should be able to update the quantity of the order with orderID "C01" to 40.00.

Result Output: As expected, DBS executes the script successfully, updating the quantity of the order with orderID "C01" to 40.00 as shown below.



orderID	requestID	isbn	borrowDate	returnDate	status	dueDate	accruedFine
C01	R01	9780743273565	2023-01-20	2023-01-29	Returned	2023-01-27	40.00

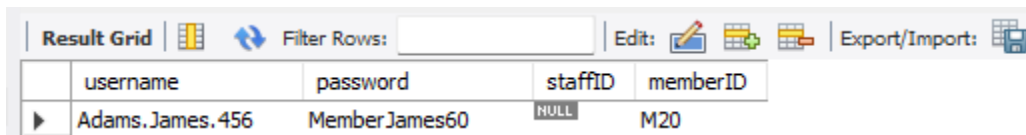
26/ Use case name: Insert Register

Implementation:

```
INSERT INTO Register (username, password, staffID, memberID)
VALUES ('Adams.James.456', 'MemberJames60', NULL, 'M20')
```

Expected Output: DBS should be able to insert a new register into the Register table.

Result Output: DBS executes the script successfully, inserting the new register with as shown below.



username	password	staffID	memberID
Adams.James.456	MemberJames60	NULL	M20

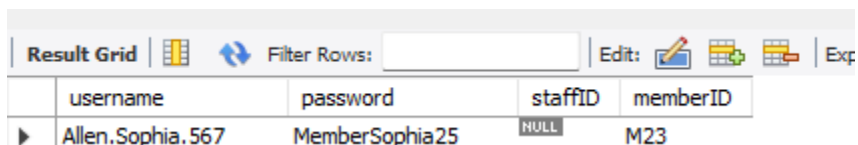
27/ Use case name: Delete Register

Implementation:

```
DELETE FROM Register
WHERE memberID= 'M20';
```

Expected Output: DBS should be able to delete the register with memberID= 'M20' from the Register table.

Result Output: As expected, DBS executes the script successfully, deleting the register with memberID= 'M20' from the Register table as shown below.



username	password	staffID	memberID
Allen.Sophia.567	MemberSophia25	NULL	M23

28/ Use case name: Update Register

Implementation:

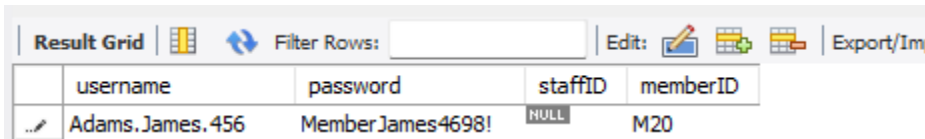
```
UPDATE Register
```

```
SET passowrd= 'MemberJames4698!'
```

```
WHERE memberID= 'M20';
```

Expected Output: DBS should be able to update the password with memberID= 'M20' to 'MemberJames4698!'

Result Output: As expected, DBS executes the script successfully, updating the password with memberID= 'M20' to 'MemberJames4698!'



The screenshot shows a database management tool interface. At the top, there is a toolbar with buttons for 'Result Grid', 'Filter Rows', 'Edit', and 'Export/Import'. Below the toolbar is a table with the following data:

username	password	staffID	memberID
Adams.James.456	MemberJames4698!	NULL	M20

29/ Use case name: Determine where the book locates and the status of that book

Implementation:

```
SELECT Book.isbn, book.title, book.status, Branch.branchID AS availableAt,
```

```
Branch.address
```

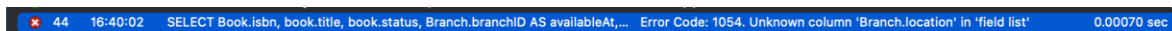
```
FROM Offer
```

```
INNER JOIN Book ON Offer.isbn = Book.isbn
```

```
INNER JOIN Branch ON Offer.branchID = Branch.branchID;
```

Expected Output: DBS should be able to return books information along with its location, this can be done in large batches or a specific book.

Result Output: DBS was not able to return correct information with below error



The screenshot shows a database error message with the following text: "44 16:40:02 SELECT Book.isbn, book.title, book.status, Branch.branchID AS availableAt, ... Error Code: 1054. Unknown column 'Branch.location' in 'field list' 0.00070 sec".

Solution: Since Branch.location is not in the field list, it should be Branch.address.

Solution Output: As expected, the error was correct. DBS execute scripts successfully with below results.

9780060883287	One Hundred Years of Solitude	Available	B01	123 Main St, Houston, USA
9780061120084	To Kill A Mockingbird	On Loan	B01	123 Main St, Houston, USA
9780451527741	1984	On Loan	B01	123 Main St, Houston, USA
9780805301020	Molecular Biology of the Cell	Available	B01	123 Main St, Houston, USA
9781408894705	The Testaments	Available	B01	123 Main St, Houston, USA
9780199699326	The Oxford Handbook of Criminology	Available	B02	456 Oak Ave, Houston, USA
9780226104034	The Structure of Scientific Revolutions	Available	B02	456 Oak Ave, Houston, USA
9780316341686	The Girl Who Takes an Eye for an Eye	On Loan	B02	456 Oak Ave, Houston, USA
9780387943804	The Feynman Lectures on Physics	Available	B02	456 Oak Ave, Houston, USA
9780544003415	The Lord of the Rings	Available	B02	456 Oak Ave, Houston, USA
9780547928210	The Hobbit	Available	B02	456 Oak Ave, Houston, USA
9780747532743	Harry Potter and the Philosophers St...	Available	B02	456 Oak Ave, Houston, USA
9780878939640	Evolution: The Story of Life	Available	B02	456 Oak Ave, Houston, USA
9780061231327	A Brief History of Time	Available	B03	789 Elm Blvd, Houston, USA
9780063021909	The Midnight Library	Available	B03	789 Elm Blvd, Houston, USA
9780199216819	The Selfish Gene	Available	B03	789 Elm Blvd, Houston, USA
9780743273566	The Great Gatsby	Available	B03	789 Elm Blvd, Houston, USA
9780141439518	Pride and Prejudice	On Loan	B04	1010 Pine Rd, Houston, USA
9780231137108	Guns, Germs, and Steel	Available	B04	1010 Pine Rd, Houston, USA
9780743273565	The Power of Now: A Guide to Spirit...	Available	B04	1010 Pine Rd, Houston, USA
9780307277671	The Da Vinci Code	Available	B05	1111 Maple Dr, Houston, U...
9780041670040	The Catcher in the Rye	Available	B05	1111 Maple Dr, Houston, U...
Result 1				

30/ Use case name: Determine who hires staffs to coordinate staffs

Implementation:

```
SELECT Admin.adminID, Admin.title, Admin.lName, Admin.fName,
Staff.staffID AS hiredBy, Staff.title AS hiredTitle, Staff.lName, Staff.fName
FROM Hiring
INNER JOIN Staff ON Hiring.staffID = Staff.staffID
INNER JOIN Admin ON Hiring.adminID = Admin.adminID;
```

Expected Output: DBS should be able to show a table consisting of staff and who hired them with their names.

Result Output: As expected, DBS executes scripts successfully with below results showing staff and administration information.

adminID	title	adminLastName	adminFirstName	hiredBy	hiredTitle	staffLastNa...	staffFirstNa...
Ad01	General Manager	Johnson	Jessica	S01	Floor Staff	Lee	James
Ad01	General Manager	Johnson	Jessica	S06	Warehouse Staff	Andir	Holland
Ad01	General Manager	Johnson	Jessica	S07	Floor Staff	Smith	Key
Ad01	General Manager	Johnson	Jessica	S14	Warehouse Staff	Chen	Alec
Ad03	Executive Manager	Garcia	Maria	S05	IT Staff	Liu	Michael
Ad06	General Manager	Henderson	Tammie	S02	IT Staff	Wang	Linda
Ad06	General Manager	Henderson	Tammie	S03	Warehouse Staff	Kim	David
Ad06	General Manager	Henderson	Tammie	S10	Floor Staff	Le	Hong
Ad06	General Manager	Henderson	Tammie	S13	Floor Staff	Nguyen	Tracy
Ad06	General Manager	Henderson	Tammie	S15	IT Staff	Tran	Derik
Ad07	General Manager	Jordan	Morgan	S04	Floor Staff	Chen	Sophie
Ad07	General Manager	Jordan	Morgan	S08	IT Staff	Keyes	Vincent
Ad07	General Manager	Jordan	Morgan	S09	Warehouse Staff	Collins	Kayla
Ad07	General Manager	Jordan	Morgan	S11	It Staff	Huan	Tran
Ad07	General Manager	Jordan	Morgan	S12	Warehouse Staff	Willy	Linda

31/ Use case name: Check who wrote books in the library inventory

Implementation:

```
SELECT Book.isbn, Book.title, Author.authorID, Author.lName,
       Author.fName
```

```
FROM WrittenBy
```

```
INNER JOIN Book ON WrittenBy.isbn = Book.isbn
```

```
INNER JOIN Author ON WrittenBy.authorID = Author.authorID;
```

Expected Output: DBS should be able to show a table consisting of books along with their associating authors including multiple authors

Result Output: As expected, DBS executes scripts successfully with below results showing books and their information including their authors.

authorID	lName	fName	No_of_Books
A16	Tolkien	J.R.R.	2
A01	García Márquez	Gabriel	1
A02	Lee	Harper	1
A03	Hawking	Stephen	1
A04	Haig	Matt	1
A05	Austen	Jane	1
A06	Dawkins	Richard	1
A07	Maguire	Mike	1
A08	Kuhn	Thomas S.	1
A09	Diamond	Jared	1
A10	Brown	Dan	1
A11	Lagercrantz	David	1
A12	Salinger	J.D.	1
A13	Feynman	Richard P.	1
A14	Orwell	George	1
A15	Kandel	Eric R.	1
A18	Tolle	Eckhart	1
A19	Fitzgerald	F. Scott	1
A20	Rowling	J.K.	1
A21	Gladwell	Malcolm	1
A22	Alberts	Bruce	1
A23	Douglas Palmer	Douglas	1
A24	Atwood	Margaret	1
A25	Greene	Brian	1

32/ Use case name: Report the average age of staff for a census

Implementation:

```
SELECT AVG(age) AS average_age_staff  
FROM Staff;
```

Expected Output: DBS should be able to export data that can be exported to an .csv file.

Result Output: As expected, DBS executes scripts successfully with below results showing the average age of staff roster.

average_age_staff
34.6667

33/ Use case name: Print our total open hour of each branch for advertisement

Implementation:

```
SELECT branchID, closeHour, openHour, SEC_TO_TIME( SUM(  
TIME_TO_SEC( closeHour - openHour ) ) ) AS total_Open_Hour  
FROM Branch  
GROUP BY branchID;
```

Expected Output: DBS should be able to export data that can be exported to an .csv file.

Result Output: Error, DBS was not able to produce data for total open hours of each branch with syntax error.

Solution: Use this script to convert time to correct format.

```
SEC_TO_TIME( SUM( TIME_TO_SEC( closeHour - openHour ) ) ) AS total_Open_Hour
```

Solution Output: As expected, new scripts were carried out successfully exporting correct data.

branchID	closeHour	openHour	total_Open_Ho...
B01	17:00:00	09:00:00	08:00:00
B02	18:00:00	10:00:00	08:00:00
B03	19:00:00	11:00:00	08:00:00
B04	20:00:00	12:00:00	08:00:00
B05	21:00:00	13:00:00	08:00:00

34/ Use case name: Determine how many books there are in the library

Implementation:

```
SELECT COUNT(isbn) AS total_book
FROM Book;
```

Expected Output: DBS should be able to show the total books in the inventory.

Result Output: As expected, DBS executes scripts successfully with below results.

total_book
25

35/ Use case name: Report the number of females and males administrator within the library system for a census

Implementation:

```
SELECT sex, COUNT(*) AS total
FROM Admin
GROUP BY sex;
```

Expected Output: DBS should be able to show the total male and female in the administrator roster.

Result Output: As expected, DBS executes scripts successfully with below results.

sex	total
F	6
M	4

36/ Use case name: Check the most expensive accrued fine in the library system

Implementation:

```
SELECT orderID, requestID, accruedFine
FROM `Order`
WHERE accruedFine = (SELECT MAX(accuredFine) FROM `Order`);
```

Expected Output: DBS should be able to show which orderID has the highest accrued fine.

Result Output: Error. Instead of showing orderID with the highest accrued fine, DBS returns every member sorted by accrued fine.

Solution: Specify target by using this script.

orderID	requestID	accuredFine
C03	R05	70.00

37/ Use case name: Determine total request by status inspecific period of time

Implementation:

```
SELECT status, COUNT(*) as totalOrder
FROM Request
WHERE date >= '2023-01-01' AND date <= '2023-03-31'
GROUP BY status;
```

Expect Output: DBS should show the total number of orders base on status

Result Output: DBS does show the total number of orders base on status

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	status	totalOrder			
▶	Approved	6			
	Pending	2			
	Denied	2			

38/ Use case name: Determine total author

Implementation:

```
SELECT COUNT(DISTINCT CONCAT(lName, fName)) as TotalAuthor  
FROM Author;
```

Expect Output: DBS should show the total number of author

Result Output: DBS does show the total number of author

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	TotalAuthor				
▶	24				

39/ Use case name: Determine which admin manage which library branch

Implementation:

```
SELECT  
    m.adminID,  
    a.fName AS adminFristName,  
    a.lName AS adminLastName,  
    a.title AS title,  
    m.branchID,  
    b.address AS branchAddress  
FROM  
    Manage m  
    INNER JOIN Admin a ON m.adminID = a.adminID  
    INNER JOIN Branch b on m.branchID = b.branchID
```

Expect Output: DBS should show admins' name and the library branch that they work at

Result Output: DBS does show admins' name and the library branch that they work at

Result Grid		Filter Rows:		Export:		Wrap Cell Content: <input type="checkbox"/>	
	adminID	adminFristName	adminLastName	title	branchID	branchAddress	
▶	Ad01	Jessica	Johnson	General Manager	B01	123 Main St, Houston, USA	
	Ad08	Zelly	Gonzales	Order Manager	B01	123 Main St, Houston, USA	
	Ad09	Kenneth	Marvis	IT Manager	B01	123 Main St, Houston, USA	
	Ad10	Tommy	Nguyen	Relation Manager	B01	123 Main St, Houston, USA	
	Ad02	John	Smith	General Manager	B02	456 Oak Ave, Houston, USA	
	Ad03	Maria	Garcia	Executive Manager	B03	789 Elm Blvd, Houston, USA	
	Ad04	David	Brown	IT Manager	B04	1010 Pine Rd, Houston, USA	
	Ad06	Tammie	Henderson	General Manager	B04	1010 Pine Rd, Houston, USA	
	Ad07	Morgan	Jordan	General Manager	B04	1010 Pine Rd, Houston, USA	
	Ad05	Emily	Taylor	Bracnhes Manager	B05	1111 Maple Dr, Houston, USA	

40/ Use case name: Determine total of staff with specific title in each branch

implementation:

```
SELECT
    p.branchID,
    b.address AS branchAddress,
    s.title as staffTitle,
    COUNT(*) AS numberOfEmployee
FROM
    primaryBranch p
    INNER JOIN Staff s ON p.staffID = s.staffID
    INNER JOIN Branch b on p.branchID = b.branchID
GROUP BY p.branchID, s.title;
```

Expect Output: DBS should the number of staff with specific titles in a specific branch.

Result Output: DBS does the number of staff with specific titles in a specific branch.

Result Grid				
		Filter Rows:	Export:	Wrap Cell Content:
	branchID	branchAddress	staffTitle	numberOfEmployee
▶	B01	123 Main St, Houston, USA	Floor Staff	1
	B01	123 Main St, Houston, USA	IT Staff	1
	B01	123 Main St, Houston, USA	Warehouse Staff	1
	B02	456 Oak Ave, Houston, USA	Floor Staff	1
	B02	456 Oak Ave, Houston, USA	IT Staff	1
	B02	456 Oak Ave, Houston, USA	Warehouse Staff	1
	B03	789 Elm Blvd, Houston, USA	Floor Staff	1
	B03	789 Elm Blvd, Houston, USA	IT Staff	1
	B03	789 Elm Blvd, Houston, USA	Warehouse Staff	1
	B04	1010 Pine Rd, Houston, USA	Floor Staff	1
	B04	1010 Pine Rd, Houston, USA	It Staff	1
	B04	1010 Pine Rd, Houston, USA	Warehouse Staff	1
	B05	1111 Maple Dr, Houston, USA	Floor Staff	1
	B05	1111 Maple Dr, Houston, USA	Warehouse Staff	1
	B05	1111 Maple Dr, Houston, USA	IT Staff	1

41/ Use case name: View order detail

Implementation:

SELECT

o.orderID,
 r.requestID,
 br.address AS LibraryAddress,
 s.staffID,
 s.fName AS staffFristName,
 s.lName AS staffLastName,
 m.memberID,
 m.fName AS memberFristName,
 m.lName AS memberLastName,
 r.date AS requestDate,
 r.time AS requestTime,
 b.isbn AS bookISBN,
 b.title AS bookTitle,
 o.borrowDate,
 o.dueDate,
 o.returnDate,

```

o.status AS orderStatus,
o.accuredFine
FROM `Order` o
INNER JOIN Request r on o.requestID = r.requestID
INNER JOIN Book b on o.isbn = b.isbn
INNER JOIN Member m on r.memberID = m.memberID
INNER JOIN Staff s on r.staffID = s.staffID
INNER JOIN Branch br on r.branchID = br.branchID;

```

Expect Output: DBS should show order in specific details which are extract from foreign key

Result Output: DBS does show order in specific details which are extract from foreign key

orderID	requestID	LibraryAddress	staffID	staffFirstName	staffLastName	memberID	memberFirstName	memberLastName	requestDate	requestTime	bookISBN	bookTitle	borrowDate	dueDate	returnDate	orderStatus	accuredFine
C01	R01	1111 Maple Dr, Houston, USA	S03	David	Kim	M02	Jane	Doe	2023-01-20	00:00:13	9780743273565	The Power of Now: A Guide to Spiritual Enlighte...	2023-01-20	2023-01-27	2023-01-29	Returned	20.00
C02	R03	456 Oak Ave, Houston, USA	S02	Linda	Wang	M03	James	Williams	2023-01-22	00:00:10	9780061231327	A Brief History of Time	2023-01-22	2023-01-29	2023-01-23	Returned	0.00
C03	R05	123 Main St, Houston, USA	S01	James	Lee	M10	Wei	Wang	2023-02-10	00:00:09	9780199216819	The Selfish Gene	2023-02-10	2023-02-17	2023-02-24	Returned	70.00
C04	R06	1111 Maple Dr, Houston, USA	S03	David	Kim	M07	David	Lee	2023-01-03	00:00:13	9780387943804	The Feynman Lectures on Physics	2023-03-01	2023-03-08	2023-03-08	Returned	0.00
C05	R08	1111 Maple Dr, Houston, USA	S03	David	Kim	M04	Mary	Brown	2023-03-21	00:00:12	9780521186627	Principles of Neural Science	2023-03-21	2023-03-28	2023-03-24	Returned	0.00
C06	R10	1010 Pine Rd, Houston, USA	S01	James	Lee	M09	Raj	Gupta	2023-03-23	00:00:15	9780743273566	The Great Gatsby	2023-03-23	2023-03-30	2023-03-31	Returned	10.00
C07	R13	123 Main St, Houston, USA	S03	David	Kim	M10	Wei	Wang	2023-04-14	00:00:15	9780316341686	The Girl Who Takes an Eye for an Eye	2023-04-14	2023-04-21	2023-04-20	Pending	0.00
C08	R15	1010 Pine Rd, Houston, USA	S02	Linda	Wang	M25	Ava	Wright	2023-04-20	00:00:11	9780767908184	Blink: The Power of Thinking Without Thinking	2023-04-20	2023-04-27	2023-04-27	Active	0.00
C09	R16	789 Elm Blvd, Houston, USA	S03	David	Kim	M01	John	Smith	2023-04-21	00:00:09	9780451527741	1984	2023-04-21	2023-04-28	2023-04-28	Active	0.00

9. Conclusion

In conclusion, this comprehensive report has provided an in-depth analysis and documentation of the database design and implementation for the library management system. We have meticulously covered various aspects of the database, including defining the library's entities and relationships, creating tables with appropriate keys and constraints, and designing a robust and scalable database structure.

Throughout the development process, we have ensured the reliability and functionality of the database system by testing it using 41 specific use cases, which span a wide range of library management system features. These use cases include, but are not limited to, managing books and authors, member registration and management, staff hiring and scheduling, library branch management, and the tracking of various library activities like book requests, loans, and returns.

The successful execution of these use cases validates the database design's effectiveness and ability to handle complex operations and tasks. In cases where errors were encountered, we promptly identified the issues and provided appropriate solutions, ensuring that the final implementation performed as expected. The ability of the database system to handle these use cases demonstrates its adaptability and reliability in meeting the library's operational requirements.

Moreover, the library management system's database design is structured to promote easy maintenance, expansion, and scalability. As the library's needs evolve, the database system can be refined and enhanced to accommodate new features, services, and requirements. This adaptability ensures that the library management system remains relevant and effective in the face of changing demands and technological advancements.

In addition to the technical aspects, we have also addressed the importance of data integrity and security. The database design includes constraints and rules that help maintain the accuracy and consistency of the data. These measures ensure that the library staff can rely on the system to provide accurate, up-to-date, and trustworthy information for their day-to-day tasks and make informed decisions.

Finally, the user-friendly interface and the comprehensive functionality provided by the library management system will enhance the overall experience for both library staff and members. The system offers a streamlined and efficient way to manage and access library resources, ultimately fostering a positive learning, research, and personal growth environment.

In summary, the library management system's database design and implementation have proven to be efficient, reliable, and adaptable, making it an invaluable tool for library staff and members alike. As the library continues to grow and evolve, the database system will serve as a solid foundation for new features and enhancements, ensuring the library remains a vital and supportive resource for its community.

10. References

- Admin. (2021, October 22). *The Importance of the Library for Students*. Kitab Center.
<https://kitabcenter.com/the-importance-of-the-library-for-students/#:~:text=Libraries%20are%20the%20best%20place%20for%20children%20to>
- library database objective*. (n.d.). Bing. Retrieved April 28, 2023, from
<https://www.bing.com/search?q=library+database++objective&qs=n&form=QBRE&sp=-1&ghc=1&lq=0&pq=library+database++objectiv&sc=6-26&sk=&cvid=E205EACEC1F94FBBB43F3B1A66739C2D&ghsh=0&ghacc=0&ghpl=>
- W3Schools. (2019). *SQL Tutorial*. W3schools.com.
<https://www.w3schools.com/sql/default.asp>

