**Library Management System using SQL Project**

**Project Overview:**

**Project Title:** Library Management System  
**Database:** library

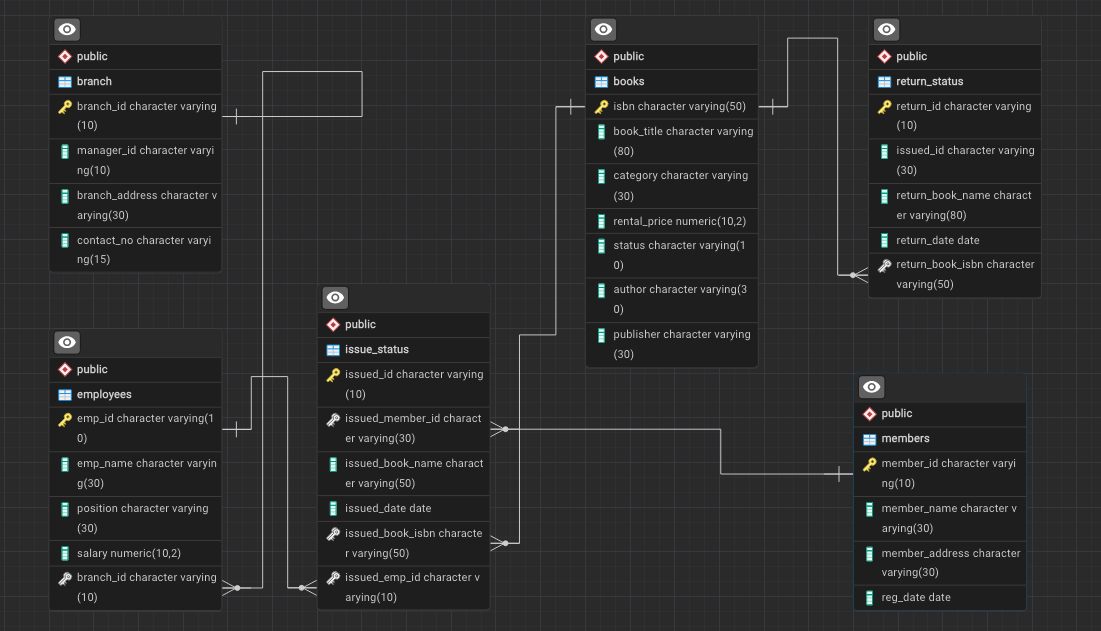
This project demonstrates the implementation of a Library Management System using SQL. It includes creating and managing tables, performing CRUD operations, and executing advanced SQL queries. The goal is to showcase skills in database design, manipulation, and querying.

**Objectives**

1. Set up the Library Management System Database: Create and populate the database with tables for branches, employees, members, books, issued status, and return status.
2. CRUD Operations: Perform Create, Read, Update, and Delete operations on the data.
3. CTAS (Create Table As Select): Utilize CTAS to create new tables based on query results.
4. Advanced SQL Queries: Develop complex queries to analyze and retrieve specific data.

**Project Structure**

**1. Database Setup**

**[](https://github.com/najirh/Library-System-Management---P2/blob/main/library_erd.png)**

* **Database Creation: Created a database named library.**
* **Table Creation: Created tables for branches, employees, members, books, issued status, and return status. Each table includes relevant columns and relationships.**

CREATE DATABASE library;

DROP TABLE IF EXISTS branch;

CREATE TABLE branch

(

branch\_id VARCHAR(10) PRIMARY KEY,

manager\_id VARCHAR(10),

branch\_address VARCHAR(30),

contact\_no VARCHAR(15)

);

**--Create table "Employee"**

DROP TABLE IF EXISTS employees;

CREATE TABLE employees

(

emp\_id VARCHAR(10) PRIMARY KEY,

emp\_name VARCHAR(30),

position VARCHAR(30),

salary DECIMAL(10,2),

branch\_id VARCHAR(10),

FOREIGN KEY (branch\_id) REFERENCES branch(branch\_id)

);

**-- Create table "Members"**

DROP TABLE IF EXISTS members;

CREATE TABLE members

(

member\_id VARCHAR(10) PRIMARY KEY,

member\_name VARCHAR(30),

member\_address VARCHAR(30),

reg\_date DATE

);

**-- Create table "Books"**

DROP TABLE IF EXISTS books;

CREATE TABLE books

(

isbn VARCHAR(50) PRIMARY KEY,

book\_title VARCHAR(80),

category VARCHAR(30),

rental\_price DECIMAL(10,2),

status VARCHAR(10),

author VARCHAR(30),

publisher VARCHAR(30)

);

**-- Create table "IssueStatus"**

DROP TABLE IF EXISTS issued\_status;

CREATE TABLE issued\_status

(

issued\_id VARCHAR(10) PRIMARY KEY,

issued\_member\_id VARCHAR(30),

issued\_book\_name VARCHAR(80),

issued\_date DATE,

issued\_book\_isbn VARCHAR(50),

issued\_emp\_id VARCHAR(10),

FOREIGN KEY (issued\_member\_id) REFERENCES members(member\_id),

FOREIGN KEY (issued\_emp\_id) REFERENCES employees(emp\_id),

FOREIGN KEY (issued\_book\_isbn) REFERENCES books(isbn)

);

**-- Create table "ReturnStatus"**

DROP TABLE IF EXISTS return\_status;

CREATE TABLE return\_status

(

return\_id VARCHAR(10) PRIMARY KEY,

issued\_id VARCHAR(30),

return\_book\_name VARCHAR(80),

return\_date DATE,

return\_book\_isbn VARCHAR(50),

FOREIGN KEY (return\_book\_isbn) REFERENCES books(isbn)

);

**2. CRUD Operations**

* Create: Inserted sample records into the books table.
* Read: Retrieved and displayed data from various tables.
* Update: Updated records in the employees table.
* Delete: Removed records from the members table as needed.
* **Task 1.** Create a New Book Record -- "978-1-60129-456-2', 'To Kill a Mockingbird', 'Classic', 6.00, 'yes', 'Harper Lee', 'J.B. Lippincott & Co.'"

INSERT INTO books(isbn, book\_title, category, rental\_price, status, author, publisher)

VALUES('978-1-60129-456-2', 'To Kill a Mockingbird', 'Classic', 6.00, 'yes', 'Harper Lee', 'J.B. Lippincott & Co.');

SELECT \* FROM books;

* **Task 2:** Update an Existing Member's Address

UPDATE members

SET member\_address = '125 Oak St'

WHERE member\_id = 'C103';

* **Task 3:** Delete a Record from the Issued Status Table

 -- Objective: Delete the record with issued\_id = 'IS121' from the issued\_status table.

DELETE FROM issued\_status

WHERE issued\_id = 'IS121';

* **Task 4:** Retrieve All Books Issued by a Specific Employee

-- Objective: Select all books issued by the employee with emp\_id = 'E101'.

SELECT \* FROM issued\_status

WHERE issued\_emp\_id = 'E101'

* **Task 5:** List Members Who Have Issued More Than One Book

-- Objective: Use GROUP BY to find members who have issued more than one book.

SELECT

issued\_emp\_id,

COUNT(\*)

FROM issued\_status

GROUP BY 1

HAVING COUNT(\*) > 1

**3. CTAS (Create Table As Select)**

* **Task 6:** Create Summary Tables: Used CTAS to generate new tables based on query results - each book and total book\_issued\_cnt\*\*

CREATE TABLE book\_issued\_cnt AS

SELECT b.isbn, b.book\_title, COUNT(ist.issued\_id) AS issue\_count

FROM issued\_status as ist

JOIN books as b

ON ist.issued\_book\_isbn = b.isbn

GROUP BY b.isbn, b.book\_title;

**4. Data Analysis & Findings**

**The following SQL queries were used to address specific questions:**

* **Task 7.** Retrieve All Books in a Specific Category:

SELECT \* FROM books

WHERE category = 'Classic';

Task 8: Find Total Rental Income by Category:

SELECT

b.category,

SUM(b.rental\_price),

COUNT(\*)

FROM

issued\_status as ist

JOIN

books as b

ON b.isbn = ist.issued\_book\_isbn

GROUP BY 1

* **Task 9.** List Employees with Their Branch Manager's Name and their branch details:

SELECT

e1.emp\_id,

e1.emp\_name,

e1.position,

e1.salary,

b.\*,

e2.emp\_name as manager

FROM employees as e1

JOIN

branch as b

ON e1.branch\_id = b.branch\_id

JOIN

employees as e2

ON e2.emp\_id = b.manager\_id

Task 10. Create a Table of Books with Rental Price Above a Certain Threshold:

CREATE TABLE expensive\_books AS

SELECT \* FROM books

WHERE rental\_price > 7.00;

Task 11: Retrieve the List of Books Not Yet Returned

SELECT \* FROM issued\_status as ist

LEFT JOIN

return\_status as rs

ON rs.issued\_id = ist.issued\_id

WHERE rs.return\_id IS NULL;

**Advanced SQL Operations**

* **Task 12:** Identify Members with Overdue Books  
  Write a query to identify members who have overdue books (assume a 30-day return period). Display the member's\_id, member's name, book title, issue date, and days overdue.

SELECT

ist.issued\_member\_id,

m.member\_name,

bk.book\_title,

ist.issued\_date,

-- rs.return\_date,

CURRENT\_DATE - ist.issued\_date as over\_dues\_days

FROM issued\_status as ist

JOIN

members as m

ON m.member\_id = ist.issued\_member\_id

JOIN

books as bk

ON bk.isbn = ist.issued\_book\_isbn

LEFT JOIN

return\_status as rs

ON rs.issued\_id = ist.issued\_id

WHERE

rs.return\_date IS NULL

AND

(CURRENT\_DATE - ist.issued\_date) > 30

ORDER BY 1;

* **Task 13**: Update Book Status on Return  
  Write a query to update the status of books in the books table to "Yes" when they are returned (based on entries in the return\_status table).

SELECT \* FROM issued\_status

WHERE issued\_book\_isbn = '978-0-451-52994-2';

-- IS104

SELECT \* FROM books

WHERE isbn = '978-0-451-52994-2';

UPDATE books

SET status = 'no'

WHERE isbn = '978-0-451-52994-2';

SELECT \* FROM return\_status

WHERE issued\_id = 'IS130';

UPDATE books

SET status = 'yes'

WHERE isbn = '978-0-451-52994-2';

--

INSERT INTO return\_status(return\_id, issued\_id, return\_date)

VALUES

('RS125', 'IS130', CURRENT\_DATE);

SELECT \* FROM return\_status

WHERE issued\_id = 'IS130';

* **Task 14:** Branch Performance Report  
  Create a query that generates a performance report for each branch, showing the number of books issued, the number of books returned, and the total revenue generated from book rentals.

CREATE TABLE branch\_reports

AS

SELECT

b.branch\_id,

b.manager\_id,

COUNT(ist.issued\_id) as number\_book\_issued,

COUNT(rs.return\_id) as number\_of\_book\_return,

SUM(bk.rental\_price) as total\_revenue

FROM issued\_status as ist

JOIN

employees as e

ON e.emp\_id = ist.issued\_emp\_id

JOIN

branch as b

ON e.branch\_id = b.branch\_id

LEFT JOIN

return\_status as rs

ON rs.issued\_id = ist.issued\_id

JOIN

books as bk

ON ist.issued\_book\_isbn = bk.isbn

GROUP BY 1, 2;

SELECT \* FROM branch\_reports;

**Task 15: Find Employees with the Most Book Issues Processed  
Write a query to find the top 3 employees who have processed the most book issues. Display the employee name, number of books processed, and their branch.**

**SELECT**

**e.emp\_name,**

**b.\*,**

**COUNT(ist.issued\_id) as no\_book\_issued**

**FROM issued\_status as ist**

**JOIN**

**employees as e**

**ON e.emp\_id = ist.issued\_emp\_id**

**JOIN**

**branch as b**

**ON e.branch\_id = b.branch\_id**

**GROUP BY 1, 2**

**Reports**

* **Database Schema: Detailed table structures and relationships.**
* **Data Analysis: Insights into book categories, employee salaries, member registration trends, and issued books.**
* **Summary Reports: Aggregated data on high-demand books and employee performance.**

**Conclusion**

**This project demonstrates the application of SQL skills in creating and managing a library management system. It includes database setup, data manipulation, and advanced querying, providing a solid foundation for data management and analysis.**

**Thank you for your interest in this project!**