Library Management System - SQL Project Documentation

Overview

The Library Management System (LMS) is a structured SQL project designed to manage core library operations such as maintaining book records, managing members, tracking book issues and returns, and generating analytical reports. The system includes multiple interconnected tables that simulate a real-world library workflow using SQL queries ranging from basic CRUD operations to advanced JOINs, aggregate functions, and CTAS (Create Table As Select).

Objectives

- Design a normalized database structure for library operations.
- Insert realistic sample data for all tables.
- Perform CRUD operations to understand data manipulation.
- Use JOIN operations to link related tables and extract meaningful insights.
- Use aggregate functions to analyze performance metrics.
- Generate summary and analytical reports using CTAS.
- Track overdue books and identify member/employee patterns.
- Build a foundation for a scalable SQL-based library system.

Database Schema

```
CREATE DATABASE library;
USE library;
CREATE TABLE branch
( branch_id VARCHAR(10) PRIMARY KEY,
  manager_id VARCHAR(10),
  branch_address VARCHAR(30),
  contact_no VARCHAR(15));

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
(member_id VARCHAR(10) PRIMARY KEY,
 member_name VARCHAR(30),
 member_address VARCHAR(30),
 reg_date DATE);

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 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 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));

select * from branch;
select * from employees;
select * from members;
select * from books;
select * from issued_status;
select * from return_status;

INSERT INTO members(member_id, member_name, member_address, reg_date) VALUES
('C101', 'Alice Johnson', '123 Main St', '2021-05-15'),
('C102', 'Bob Smith', '456 Elm St', '2021-06-20'),
('C103', 'Carol Davis', '789 Oak St', '2021-07-10'),
('C104', 'Dave Wilson', '567 Pine St', '2021-08-05'),
('C105', 'Eve Brown', '890 Maple St', '2021-09-25'),
('C106', 'Frank Thomas', '234 Cedar St', '2021-10-15'),
('C107', 'Grace Taylor', '345 Walnut St', '2021-11-20'),
('C108', 'Henry Anderson', '456 Birch St', '2021-12-10'),
('C109', 'Ivy Martinez', '567 Oak St', '2022-01-05'),
('C110', 'Jack Wilson', '678 Pine St', '2022-02-25'),
('C118', 'Sam', '133 Pine St', '2024-06-01'),
('C119', 'John', '143 Main St', '2024-05-01');
SELECT * FROM members;

INSERT INTO branch(branch_id, manager_id, branch_address, contact_no) VALUES
('B001', 'E109', '123 Main St', '+919099988676'),
('B002', 'E109', '456 Elm St', '+919099988677'),
('B003', 'E109', '789 Oak St', '+919099988678'),
('B004', 'E110', '567 Pine St', '+919099988679'),
('B005', 'E110', '890 Maple St', '+919099988680');
SELECT * FROM branch;

INSERT INTO employees(emp_id, emp_name, position, salary, branch_id) VALUES
('E101', 'John Doe', 'Clerk', 60000.00, 'B001'),
('E102', 'Jane Smith', 'Clerk', 45000.00, 'B002'),
('E103', 'Mike Johnson', 'Librarian', 55000.00, 'B001'),
('E104', 'Emily Davis', 'Assistant', 40000.00, 'B001'),

('E105', 'Sarah Brown', 'Assistant', 42000.00, 'B001'),
('E106', 'Michelle Ramirez', 'Assistant', 43000.00, 'B001'),
('E107', 'Michael Thompson', 'Clerk', 62000.00, 'B005'),
('E108', 'Jessica Taylor', 'Clerk', 46000.00, 'B004'),
('E109', 'Daniel Anderson', 'Manager', 57000.00, 'B003'),
('E110', 'Laura Martinez', 'Manager', 41000.00, 'B005'),
('E111', 'Christopher Lee', 'Assistant', 65000.00, 'B005');
SELECT * FROM employees;

INSERT INTO books(isbn, book_title, category, rental_price, status, author, publisher) VALUES
('978-0-553-29698-2', 'The Catcher in the Rye', 'Classic', 7.00, 'yes', 'J.D. Salinger', 'Little, Brown and Company'),
('978-0-330-25864-8', 'Animal Farm', 'Classic', 5.50, 'yes', 'George Orwell', 'Penguin Books'),
('978-0-14-118776-1', 'One Hundred Years of Solitude', 'Literary Fiction', 6.50, 'yes', 'Gabriel Garcia Marquez', 'Penguin Books'),
('978-0-525-47535-5', 'The Great Gatsby', 'Classic', 8.00, 'yes', 'F. Scott Fitzgerald', 'Scribner'),
('978-0-141-44171-6', 'Jane Eyre', 'Classic', 4.00, 'yes', 'Charlotte Bronte', 'Penguin Classics'),
('978-0-307-37840-1', 'The Alchemist', 'Fiction', 2.50, 'yes', 'Paulo Coelho', 'HarperOne'),
('978-0-679-76489-8', 'Harry Potter and the Sorcerers Stone', 'Fantasy', 7.00, 'yes', 'J.K. Rowling', 'Scholastic'),
('978-0-7432-4722-4', 'The Da Vinci Code', 'Mystery', 8.00, 'yes', 'Dan Brown', 'Doubleday'),
('978-0-09-957807-9', 'A Game of Thrones', 'Fantasy', 7.50, 'yes', 'George R.R. Martin', 'Bantam'),
('978-0-393-05081-8', 'A Peoples History of the United States', 'History', 9.00, 'yes', 'Howard Zinn', 'Harper Perennial'),
('978-0-19-280551-1', 'The Guns of August', 'History', 7.00, 'yes', 'Barbara W. Tuchman', 'Oxford University Press'),
('978-0-307-58837-1', 'Sapiens: A Brief History of Humankind', 'History', 8.00, 'no', 'Yuval Noah Harari', 'Harper Perennial'),
('978-0-375-41398-8', 'The Diary of a Young Girl', 'History', 6.50, 'no', 'Anne Frank', 'Bantam'),
('978-0-14-044930-3', 'The Histories', 'History', 5.50, 'yes', 'Herodotus', 'Penguin Classics'),
('978-0-393-91257-8', 'Guns, Germs, and Steel: The Fates of Human Societies', 'History', 7.00, 'yes', 'Jared Diamond', 'W. W. Norton & Company'),
('978-0-7432-7357-1', '1491: New Revelations of the Americas Before Columbus', 'History', 6.50, 'no', 'Charles C. Mann', 'Vintage Books'),
('978-0-679-64115-3', '1984', 'Dystopian', 6.50, 'yes', 'George Orwell', 'Penguin Books'),
('978-0-14-143951-8', 'Pride and Prejudice', 'Classic', 5.00, 'yes', 'Jane Austen', 'Penguin Classics'),
('978-0-452-28240-7', 'Brave New World', 'Dystopian', 6.50, 'yes', 'Aldous Huxley', 'Harper Perennial'),
('978-0-670-81302-4', 'The Road', 'Dystopian', 7.00, 'yes', 'Cormac McCarthy', 'Knopf'),
('978-0-385-33312-0', 'The Shining', 'Horror', 6.00, 'yes', 'Stephen King', 'Doubleday'),
('978-0-451-52993-5', 'Fahrenheit 451', 'Dystopian', 5.50, 'yes', 'Ray Bradbury', 'Ballantine Books'),
('978-0-345-39180-3', 'Dune', 'Science Fiction', 8.50, 'yes', 'Frank Herbert', 'Ace'),
('978-0-375-50167-0', 'The Road', 'Dystopian', 7.00, 'yes', 'Cormac McCarthy', 'Vintage'),
('978-0-06-025492-6', 'Where the Wild Things Are', 'Children', 3.50, 'yes', 'Maurice Sendak', 'HarperCollins'),
('978-0-06-112241-5', 'The Kite Runner', 'Fiction', 5.50, 'yes', 'Khaled Hosseini', 'Riverhead Books'),
('978-0-06-440055-8', 'Charlotte''s Web', 'Children', 4.00, 'yes', 'E.B. White', 'Harper & Row'),
('978-0-679-77644-3', 'Beloved', 'Fiction', 6.50, 'yes', 'Toni Morrison', 'Knopf'),
('978-0-14-027526-3', 'A Tale of Two Cities', 'Classic', 4.50, 'yes', 'Charles Dickens', 'Penguin Books'),
('978-0-7434-7679-3', 'The Stand', 'Horror', 7.00, 'yes', 'Stephen King', 'Doubleday'),
('978-0-451-52994-2', 'Moby Dick', 'Classic', 6.50, 'yes', 'Herman Melville', 'Penguin Books'),
('978-0-06-112008-4', 'To Kill a Mockingbird', 'Classic', 5.00, 'yes', 'Harper Lee', 'J.B. Lippincott & Co.'),
('978-0-553-57340-1', '1984', 'Dystopian', 6.50, 'yes', 'George Orwell', 'Penguin Books'),
('978-0-7432-4722-5', 'Angels & Demons', 'Mystery', 7.50, 'yes', 'Dan Brown', 'Doubleday'),

('978-0-7432-7356-4', 'The Hobbit', 'Fantasy', 7.00, 'yes', 'J.R.R. Tolkien', 'Houghton Mifflin Harcourt');
select * from books;

INSERT INTO issued_status(issued_id, issued_member_id, issued_book_name, issued_date, issued_book_isbn, issued_emp_id) VALUES
('IS106', 'C106', 'Animal Farm', '2024-03-10', '978-0-330-25864-8', 'E104'),
('IS107', 'C107', 'One Hundred Years of Solitude', '2024-03-11', '978-0-14-118776-1', 'E104'),
('IS108', 'C108', 'The Great Gatsby', '2024-03-12', '978-0-525-47535-5', 'E104'),
('IS109', 'C109', 'Jane Eyre', '2024-03-13', '978-0-141-44171-6', 'E105'),
('IS110', 'C110', 'The Alchemist', '2024-03-14', '978-0-307-37840-1', 'E105'),
('IS111', 'C109', 'Harry Potter and the Sorcerers Stone', '2024-03-15', '978-0-679-76489-8', 'E105'),
('IS112', 'C109', 'A Game of Thrones', '2024-03-16', '978-0-09-957807-9', 'E106'),
('IS113', 'C109', 'A Peoples History of the United States', '2024-03-17', '978-0-393-05081-8', 'E106'),
('IS114', 'C109', 'The Guns of August', '2024-03-18', '978-0-19-280551-1', 'E106'),
('IS115', 'C109', 'The Histories', '2024-03-19', '978-0-14-044930-3', 'E107'),
('IS116', 'C110', 'Guns, Germs, and Steel: The Fates of Human Societies', '2024-03-20', '978-0-393-91257-8', 'E107'),
('IS117', 'C110', '1984', '2024-03-21', '978-0-679-64115-3', 'E107'),
('IS118', 'C101', 'Pride and Prejudice', '2024-03-22', '978-0-14-143951-8', 'E108'),
('IS119', 'C110', 'Brave New World', '2024-03-23', '978-0-452-28240-7', 'E108'),
('IS120', 'C110', 'The Road', '2024-03-24', '978-0-670-81302-4', 'E108'),
('IS121', 'C102', 'The Shining', '2024-03-25', '978-0-385-33312-0', 'E109'),
('IS122', 'C102', 'Fahrenheit 451', '2024-03-26', '978-0-451-52993-5', 'E109'),
('IS123', 'C103', 'Dune', '2024-03-27', '978-0-345-39180-3', 'E109'),
('IS124', 'C104', 'Where the Wild Things Are', '2024-03-28', '978-0-06-025492-6', 'E110'),
('IS125', 'C105', 'The Kite Runner', '2024-03-29', '978-0-06-112241-5', 'E110'),
('IS126', 'C105', 'Charlotte''s Web', '2024-03-30', '978-0-06-440055-8', 'E110'),
('IS127', 'C105', 'Beloved', '2024-03-31', '978-0-679-77644-3', 'E110'),
('IS128', 'C105', 'A Tale of Two Cities', '2024-04-01', '978-0-14-027526-3', 'E110'),
('IS129', 'C105', 'The Stand', '2024-04-02', '978-0-7434-7679-3', 'E110'),
('IS130', 'C106', 'Moby Dick', '2024-04-03', '978-0-451-52994-2', 'E101'),
('IS131', 'C106', 'To Kill a Mockingbird', '2024-04-04', '978-0-06-112008-4', 'E101'),
('IS132', 'C106', 'The Hobbit', '2024-04-05', '978-0-7432-7356-4', 'E106'),
('IS133', 'C107', 'Angels & Demons', '2024-04-06', '978-0-7432-4722-5', 'E106'),
('IS134', 'C107', 'The Diary of a Young Girl', '2024-04-07', '978-0-375-41398-8', 'E106'),
('IS135', 'C107', 'Sapiens: A Brief History of Humankind', '2024-04-08', '978-0-307-58837-1', 'E108'),
('IS136', 'C107', '1491: New Revelations of the Americas Before Columbus', '2024-04-09', '978-0-7432-7357-1', 'E102'),
('IS137', 'C107', 'The Catcher in the Rye', '2024-04-10', '978-0-553-29698-2', 'E103'),
('IS138', 'C108', 'The Great Gatsby', '2024-04-11', '978-0-525-47535-5', 'E104'),
('IS139', 'C109', 'Harry Potter and the Sorcerers Stone', '2024-04-12', '978-0-679-76489-8', 'E105'),
('IS140', 'C110', 'Animal Farm', '2024-04-13', '978-0-330-25864-8', 'E102');
select * from issued_status;

INSERT INTO return_status(return_id, issued_id, return_date) VALUES
('RS101', 'IS101', '2023-06-06'),
('RS102', 'IS105', '2023-06-07'),
('RS103', 'IS103', '2023-08-07'),
('RS104', 'IS106', '2024-05-01'),
('RS105', 'IS107', '2024-05-03'),
('RS106', 'IS108', '2024-05-05'),
('RS107', 'IS109', '2024-05-07'),
('RS108', 'IS110', '2024-05-09'),
('RS109', 'IS111', '2024-05-11'),

('RS110', 'IS112', '2024-05-13'),
('RS111', 'IS113', '2024-05-15'),
('RS112', 'IS114', '2024-05-17'),
('RS113', 'IS115', '2024-05-19'),
('RS114', 'IS116', '2024-05-21'),
('RS115', 'IS117', '2024-05-23'),
('RS116', 'IS118', '2024-05-25'),
('RS117', 'IS119', '2024-05-27'),
('RS118', 'IS120', '2024-05-29');
SELECT * FROM return_status;

-- 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.');

-- Task 2: Update an Existing Member's Address.
update members
set member_address = '999 New St, Suite 10'
where member_id = 'c119';

-- 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.
select issued_book_name from issued_status
where issued_emp_id = 'E104';

-- Task 5: List Members Who Have Issued More Than One Book
select issued_emp_id, count(*) as issued_book
from issued_status
group by issued_emp_id
having count(*) > 1;

-- 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 count_issued
from issued_status as ist
join books as b on ist.issued_book_isbn = b.isbn
group by b.isbn,b.book_title;
select * from book_issued_cnt;

-- Task 7. Retrieve All Books in a Specific Category:
select * from books
where category = 'History';

-- Task 8: Find Total Rental Income by Category:
select category,sum(rental_price) as total_renatl,
count(*) as book_count
from books as b
join issued_status as ist

```
on ist.issued_book_isbn = b.isbn
group by category;
```

-- Task 9. List Members Who Registered in the Last 180 Days:

```
select * from members
where reg_date = date_sub(curdate(),interval 180 day);
```

-- Task 10. List Employees with Their Branch Manager's Name and their branch details:

```
select e.emp_id as employee_id,
e.emp_name as employee_name,
b.branch_id as branch_id,
m.emp_name as manager
from employees  e
join branch  b
on e.branch_id = b.branch_id
join employees  m
on m.emp_id = b.manager_id;
```

-- Task 11. Create a Table of Books with Rental Price Above a Certain Threshold:

```
create table expense_books as
select * from books
where rental_price > 7.00;
select * from expense_books;
```

-- Task 12: Retrieve the List of Books Not Yet Returned

```
 select *from issued_status as  ist
left join return_status as rs
on ist.issued_id = rs.issued_id
where rs.return_id is null;
```

-- Task 13: 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,b.book_title,ist.issued_date,
date_sub(current_date(),interval 30 day) as over_due_days
from issued_status as ist
join members as m on m.member_id = ist.issued_member_id
join books as  b on b.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 date_sub(current_date(),interval 30 day)  > 30
order by ist.issued_member_id,m.member_name,b.book_title,ist.issued_date;
```

-- 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 b.branch_id,b.manager_id;
select * from branch_reports;
```

-- Task 15: CTAS: Create a Table of Active Members Use the CREATE TABLE AS (CTAS) statement to create a new table active_members containing members who have issued at least one book in the last 2 months.

```
create table active_members as
select distinct m.member_id,m.member_name,m.member_address,m.reg_date
from members as m
join issued_status as ist on m.member_id = ist.issued_member_id
where ist.issued_date >= date_sub(current_date,interval 2 month);
select * from active_members;
```

-- Task 16: 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.branch_address as branch,
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 e.emp_name,branch
order by count(ist.issued_id) desc limit 3;
```

-- Task 17: Identify Members Issuing High-Risk Books Write a query to identify members who have issued books more than twice with the status "damaged" in the books table. Display the member name, book title, and the number of times they've issued damaged books.

```
select m.member_name,b.book_title,
count(*) as damaged_issue_count
from issued_status as ist
join members as m on ist.issued_member_id = m.member_id
join books as b on ist.issued_book_isbn = b.isbn
where b.status = 'damaged'
group by m.member_name,b.book_title
having count(*) > 2;
```

Detailed Insights of All Tasks & Queries

Task 1: Create a New Book Record

This query inserts a new book into the books table using an INSERT statement. It ensures all required fields such as ISBN, category, rental price, author, and publisher are provided.

Task 2: Update an Existing Member's Address

Uses UPDATE with a WHERE clause to modify only the selected member's address, ensuring data integrity.

Task 3: Delete a Record from issued_status

DELETE removes a specific issued book entry using issued_id as the filter. This maintains accuracy in issued records.

Task 4: Retrieve All Books Issued by a Specific Employee
Uses a SELECT with a WHERE filter to list all books issued by employee 'E104'. Helps track employee performance.

Task 5: List Members Who Have Issued More Than One Book
Uses GROUP BY and HAVING to count books issued by each employee. HAVING ensures only those with more than one entry are shown.

Task 6: Create Summary Table Using CTAS (book_issued_cnt)
Uses CTAS + JOIN to count how many times each book was issued, producing a new summary table.

Task 7: Retrieve All Books in a Specific Category
Simple SELECT with WHERE to filter books by category (e.g., History).

Task 8: Find Total Rental Income by Category
Uses SUM and COUNT on rental_price grouped by category. This provides revenue analytics per book category.

Task 9: List Members Registered in Last 180 Days
Checks registration date using DATE_SUB and CURDATE. Helps identify recent members.

Task 10: Employees with Branch Manager Name & Branch Details
A self-join on employees maps employees to branch managers. Useful for HR and branch hierarchy reports.

Task 11: Create Table of Books with Rental Price > 7
Uses CTAS to create a new table expense_books for high-rental books. Helps identify premium books.

Task 12: Retrieve List of Books Not Yet Returned
LEFT JOIN issued_status with return_status and filters rows where return_id is NULL. Identifies pending returns.

Task 13: Identify Members with Overdue Books (30+ Days)
Compares issue date with CURRENT_DATE minus 30 days. Detects overdue books and displays member & book details.

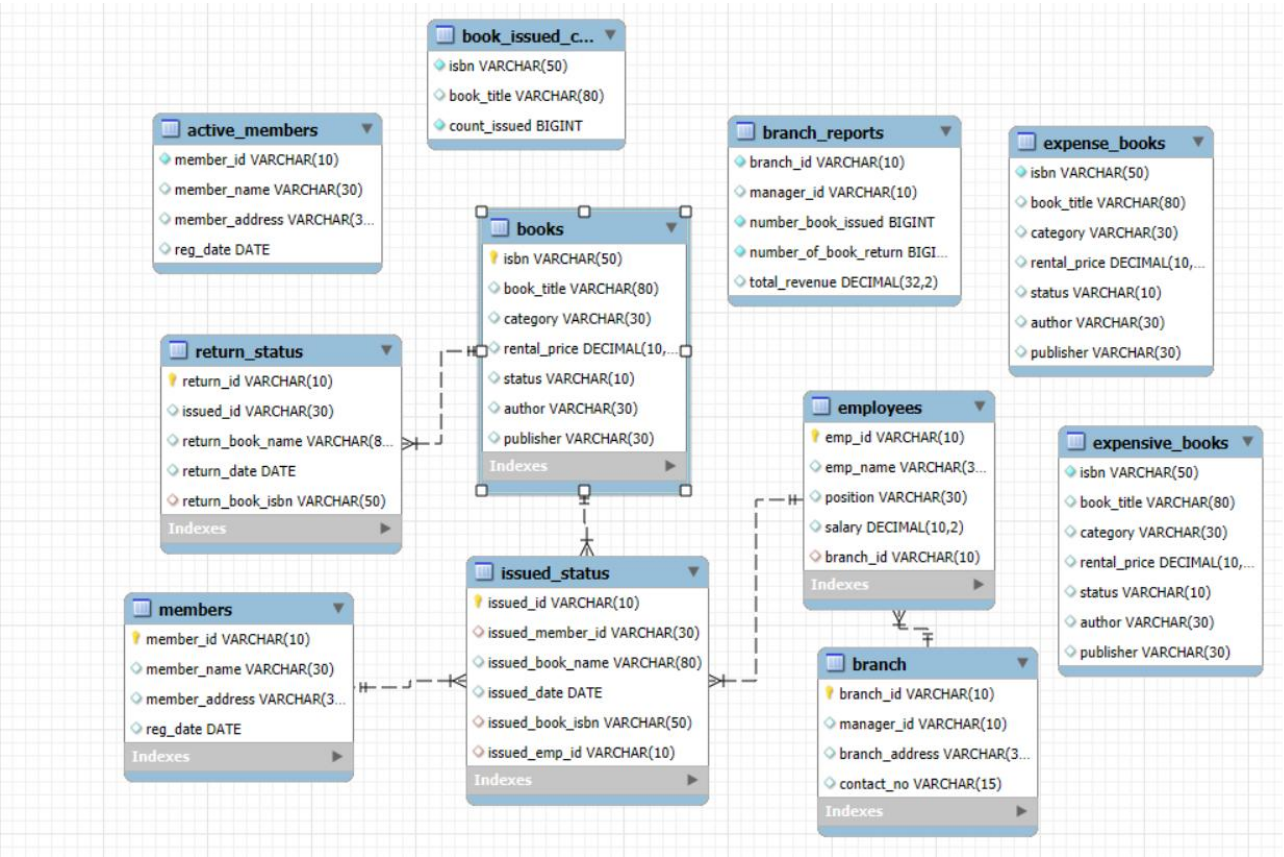
Task 14: Branch Performance Report
Combines multiple JOINs to calculate: issued books, returned books, and total rental revenue per branch. Creates a summary table branch_reports.

Task 15: CTAS to Create active_members Table
Identifies members who issued a book in last 2 months using DATE_SUB. Stores them in a new table active_members.

Task 16: Find Top 3 Employees with Most Book Issues
Ranks employees by number of processed issues using ORDER BY COUNT(*) DESC LIMIT 3.

Task 17: Identify Members Issuing Damaged Books More Than Twice
Uses GROUP BY and HAVING COUNT(*) > 2 to detect repeated issuers of damaged books. Helpful for risk management.

ER Daigram



Conclusion

This Library Management System project highlights the practical application of SQL in solving real-world data management challenges. The system demonstrates how relational databases can be used to efficiently manage books, members, employees, and operational workflows such as issuing and returning books.

Through the execution of 17 structured SQL tasks, this project explores advanced concepts such as JOIN operations, aggregate functions, subqueries, CTAS (Create Table As Select), overdue identification, and performance reporting.

Overall, this documentation provides a complete understanding of SQL database design and query implementation, showcasing strong analytical, database development, and problem-solving skills.