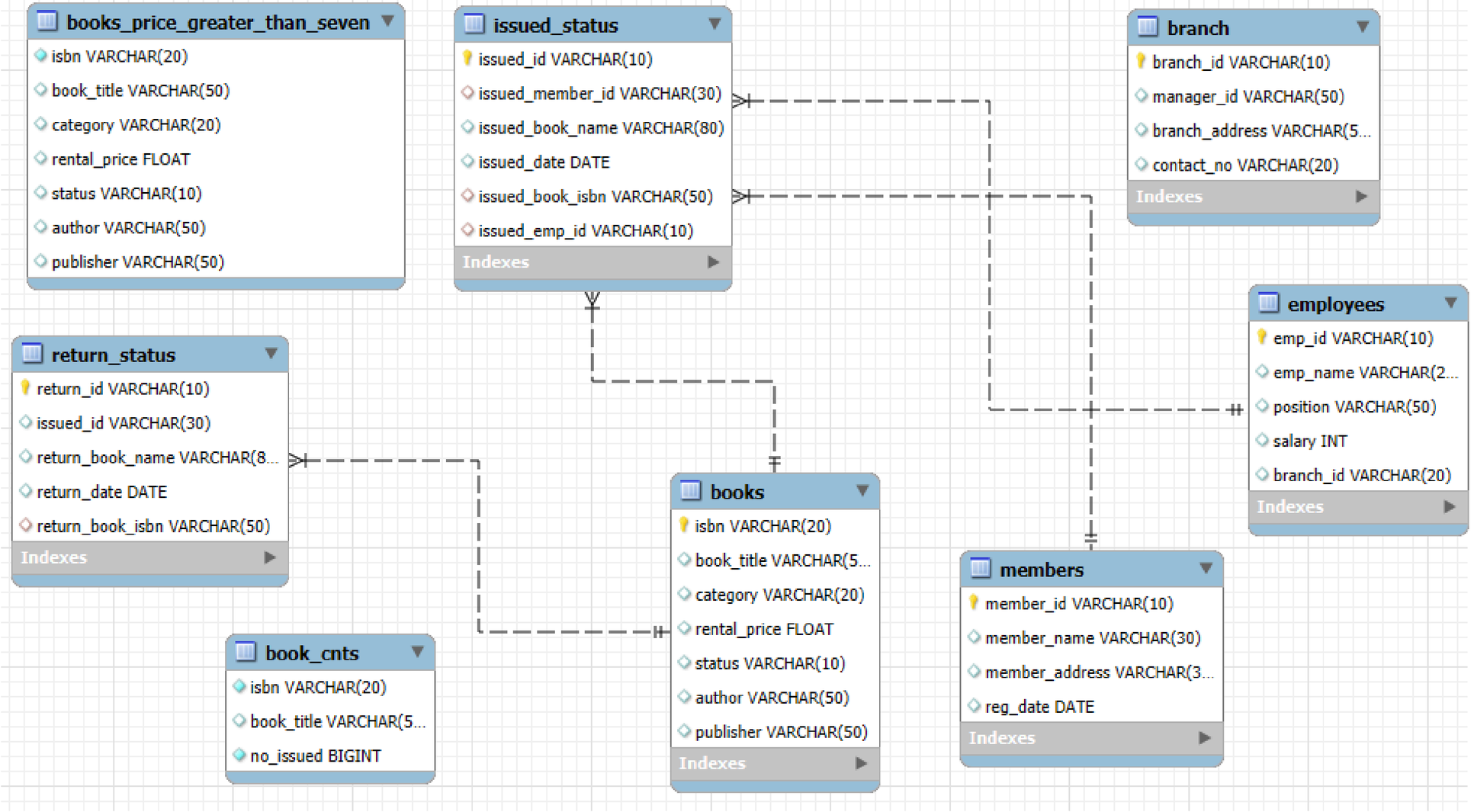
LIBRARY MANAGEMENT SYSTEM

JYOTI BAWA

Introduction

This Library Management System project is built using SQL to manage a library's core operations like managing books, employees, members, and book issue/return records. The project demonstrates how relational databases help in organizing and retrieving information efficiently using real-world queries.

Database Schema Design



Create a New Book Record -- "978-1-60129-456-2', 'To Kill a Mockingbird', 'Classic', 6.00, 'yes', 'Harper Lee', 'J.B. Lippincott & Co.'

```sql
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;
```

Update an Existing Member's Address

```sql
UPDATE members
SET
    member_address = '125 Main St'
WHERE
    member_id = 'C101';
SELECT
    *
FROM
    members;
```

Delete a Record from the Issued Status Table

```sql
SELECT
    *
FROM
    issued_status
WHERE
    issued_id = 'IS121';

DELETE FROM issued_status
WHERE
    issued_id = 'IS121';
```

Retrieve All Books Issued by a Specific Employee

```sql
SELECT
    *
FROM
    issued_status
WHERE
    issued_emp_id = 'E101';
```

List Members Who Have Issued More Than One Book

```sql
SELECT
    ist.issued_emp_id, e.emp_name
FROM
    issued_status AS ist
        JOIN
    employees AS e ON e.emp_id = ist.issued_emp_id
GROUP BY 1 , 2
HAVING COUNT(ist.issued_id) > 1;
```

Create Summary Tables: Used CTAS to generate new tables based on query results - each book and total book_issued_cnt

```sql
CREATE TABLE book_cnts
AS
SELECT
    b.isbn,
    b.book_title,
    COUNT(ist.issued_id) as no_issued
FROM books as b
JOIN
issued_status as ist
ON ist.issued_book_isbn = b.isbn
GROUP BY 1, 2;

SELECT * FROM
book_cnts;
```

Retrieve All Books in a Specific Category

```sql
SELECT * FROM books
WHERE category = 'Classic';
```

Find Total Rental Income by Category

```sql
SELECT
    b.category, SUM(b.rental_price), COUNT(*)
FROM
    books AS b
        JOIN
    issued_status AS ist ON ist.issued_book_isbn = b.isbn
GROUP BY 1;
```

List Members Who Registered in the Last 180 Days

```sql
SELECT * FROM members
WHERE reg_date >= CURRENT_DATE -
 INTERVAL 180 day;
```

```sql
INSERT INTO members(member_id, member_name, member_address, reg_date)
VALUES
('C118', 'sam', '145 Main St', '2024-06-01'),
('C119', 'john', '133 Main St', '2024-05-01');
```

List Employees with Their Branch Manager's Name and their branch details

```sql
SELECT
    e1.*, b.manager_id, e2.emp_name AS manager
FROM
    employees AS e1
        JOIN
    branch AS b ON b.branch_id = e1.branch_id
        JOIN
    employees AS e2 ON b.manager_id = e2.emp_id;
```

Create a Table of Books with Rental Price Above a Certain Threshold 7USD

```
CREATE TABLE books_price_greater_than_seven
AS   (
SELECT * FROM Books
WHERE rental_price > 7);


SELECT * FROM
books_price_greater_than_seven;
```

Retrieve the List of Books Not Yet Returned

```sql
SELECT DISTINCT
    ist.issued_book_name
FROM
    issued_status AS ist
        LEFT JOIN
    return_status AS rs ON ist.issued_id = rs.issued_id
WHERE
    rs.return_id IS NULL;


SELECT * FROM return_status;
```

CONCLUSION

Throughout the project, I implemented various SQL techniques such as:

- JOINS – to combine data across multiple tables for meaningful insights.
- CTAS (Create Table As Select) – to generate summary tables from live data.
- GROUP BY & HAVING – to perform data aggregation and filtering.
- DML operations – including INSERT, UPDATE, and DELETE for data manipulation.
- Subqueries and Conditional Filters – for retrieving precise and actionable records.

These practical implementations helped me strengthen my understanding of database relationships, normalization, and query optimization.