**LIBRARY MANAGEMENT SYSTEM SQL QUERIES**

PART 1 - Create and populate the database with tables for branches, employees, members, books, issued\_status and return\_status -

CREATE TABLE books

    (

            isbn VARCHAR(50) PRIMARY KEY,

            book\_title VARCHAR(80),

            category VARCHAR(30),

            rental\_price FLOAT,

            status VARCHAR(10),

            author VARCHAR(30),

            publisher VARCHAR(30)

    );

    CREATE TABLE members

    (

            member\_id VARCHAR(10) PRIMARY KEY,

            member\_name VARCHAR(25),

            member\_address VARCHAR(75),

            reg\_date DATE

    );

    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)

    );

    CREATE TABLE return\_status

    (

            return\_id VARCHAR(10) PRIMARY KEY,

            issued\_id VARCHAR(10),

            return\_book\_name VARCHAR(75),

            return\_date DATE,

            return\_book\_isbn VARCHAR(50)

    );

    CREATE TABLE branch

    (

            branch\_id VARCHAR(10),

            manager\_id VARCHAR(10),

            branch\_address VARCHAR(30),

            contact\_no VARCHAR(15)

    );

    CREATE TABLE IF NOT EXISTS public.employees

    (

            emp\_id VARCHAR(10),

            emp\_name VARCHAR(25),

            position VARCHAR(15),

            branch\_id VARCHAR(25)

    );

        -- FOREIGN KEY

        ALTER TABLE issued\_status

        ADD CONSTRAINT fk\_members

        FOREIGN KEY(issued\_member\_id)

        REFERENCES members(member\_id);

        ALTER TABLE issued\_status

        ADD CONSTRAINT fk\_books

        FOREIGN KEY(issued\_book\_isbn)

        REFERENCES books(isbn);

        ALTER TABLE issued\_status

        ADD CONSTRAINT fk\_employees

        FOREIGN KEY(issued\_emp\_id)

        REFERENCES employees(emp\_id);

        ALTER TABLE employees

        ADD CONSTRAINT fk\_branch

        FOREIGN KEY(branch\_id)

        REFERENCES branch(branch\_id);

        ALTER TABLE return\_status

        ADD CONSTRAINT fk\_issued\_status

        FOREIGN KEY(issued\_id)

        REFERENCES issued\_status(issued\_id);

PART 2: CRUD Operations –

1. Create a new book record:

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.');

1. Update an existing member’s address:

UPDATE members

SET member\_address = '125 Main St'

WHERE member\_id = 'C101';

1. Delete a Record from the Issued Status Table:

UPDATE members

SET member\_address = '125 Main St'

WHERE member\_id = 'C101';

1. Retrieve All Books Issued by a Specific Employee:

SELECT \* FROM issued\_status

WHERE issued\_emp\_id = 'E101'

1. List Members Who Have Issued More Than One Book:

SELECT issued\_emp\_id,

COUNT(issued\_id) AS total\_books\_issued

FROM issued\_status

GROUP BY 1

HAVING COUNT(issued\_id) > 1;

1. Create Summary Tables - Used CTAS to generate new tables based on query results - each book and total book issued count:

CREATE TABLE bookcnts

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;

SELECT \* FROM bookcnts

**Queries to address specific questions –**

1. Retrieve All Books in a Specific Category:

SELECT

b.category,

SUM(b.rental\_price) AS total\_rental\_price,

COUNT(\*) AS total\_books\_issued

FROM books b

JOIN issued\_status ist

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

GROUP BY 1

ORDER BY 2 DESC;

1. Find Total Rental Income by Category:

SELECT

b.category,

SUM(b.rental\_price) AS total\_rental\_price,

COUNT(\*) AS total\_books\_issued

FROM books b

JOIN issued\_status ist

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

GROUP BY 1

ORDER BY 2 DESC;

1. List Members Who Registered in the Last 180 Days:

SELECT \* FROM members

WHERE reg\_date >= CURRENT\_DATE - INTERVAL '180 days';

1. 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

1. 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;

SELECT \* FROM expensive\_books;

1. Retrieve the List of Books Not Yet Returned:

SELECT DISTINCT ist.issued\_book\_name

FROM issued\_status as ist

LEFT JOIN

return\_status as rs

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

WHERE rs.return\_id IS NULL;

1. Identify members who have overdue books (assume a 220-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,

CURRENT\_DATE - ist.issued\_date AS overdues\_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 > 220

ORDER BY 1;

1. 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 b.branch\_id = e.branch\_id

LEFT JOIN return\_status AS rs

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

JOIN books as bk

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

GROUP BY 1, 2;

SELECT \* FROM branch\_reports;

1. Create a TABLE AS (CTAS) statement to create a new table active\_members containing members who have issued at least one book in the last 7 months:

CREATE TABLE active\_members

AS

SELECT \* FROM members

WHERE member\_id IN

(SELECT DISTINCT issued\_member\_id

FROM issued\_status

WHERE issued\_date >= CURRENT\_DATE - INTERVAL'7 month');

SELECT \* FROM active\_members;

1. Find Employees with the Most Book Issues Processed:

SELECT

e.emp\_name,

b.\*,

COUNT(ist.issued\_id) AS books\_issued

FROM issued\_status AS ist

JOIN employees AS e

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

JOIN branch AS b

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

GROUP BY 1, 2;