



LIBRARY SYSTEM MANAGEMENT SQL PROJECT

Presented by: Diwakar Kumar

PROJECT OVERVIEW

DATABASE: LIBRARY_DB

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

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



Tables Created:

- Branch
- Employees
- Members
- Books
- IssueStatus
- ReturnStatus

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

```
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)
);
```

```
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)
);
```

CRUD OPERATIONS

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.')"

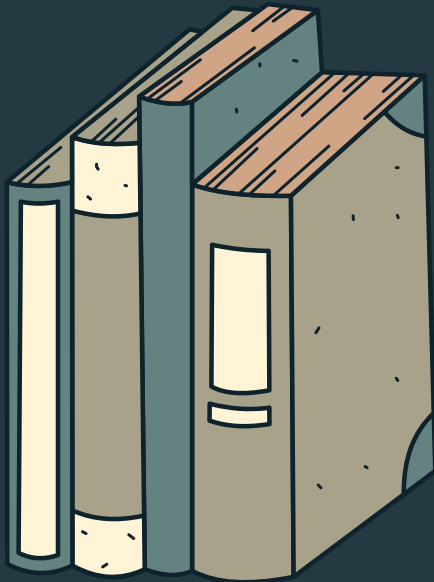
```
INSERT INTO books(isbn, book_title, category, rental_price, status, author, publisher)
VALUES
('978-1-60129-456-2', 'To Kill a Mockingbird', 'Classic', 6.00, 'yes', 'Harper Lee', 'J.B. Lippincott & Co.');
```

TASK 2: UPDATE AN EXISTING MEMBER'S ADDRESS

	member_id [PK] character varying (10)	member_name character varying (30)	member_address character varying (30)	reg_date date
1	C101	Alice Johnson	123 Main St	2021-05-15

```
UPDATE members
SET member_address = '125 Main St'
WHERE member_id = 'C101';
```

	[PK] character varying (10)	character varying (30)	character varying (30)	date
	C101	Alice Johnson	125 Main St	2021-05-15









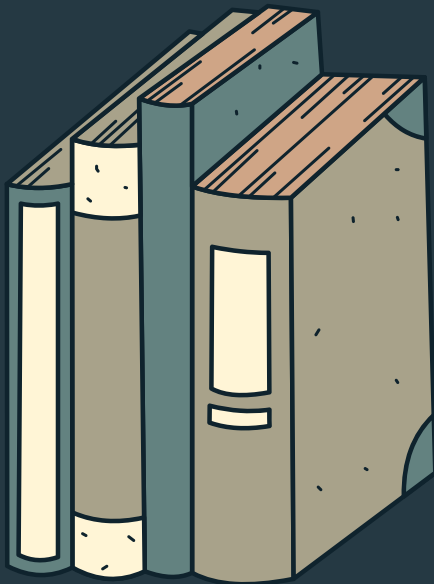
TASK 3: DELETE A RECORD FROM THE ISSUED STATUS TABLE --
OBJECTIVE: DELETE THE RECORD WITH ISSUED_ID = 'IS121' FROM THE ISSUED_STATUS TABLE.

```
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';
```

	issued_id [PK] character varying (10) 	issued_member_id character varying (30) 	issued_book_name character varying (80) 	issued_date date 	issued_book_isbn character varying (50) 	issued_emp_id character varying (10) 
1	IS130	C106	Moby Dick	2024-04-03	978-0-451-52994-2	E101
2	IS131	C106	To Kill a Mockingbird	2024-04-04	978-0-06-112008-4	E101









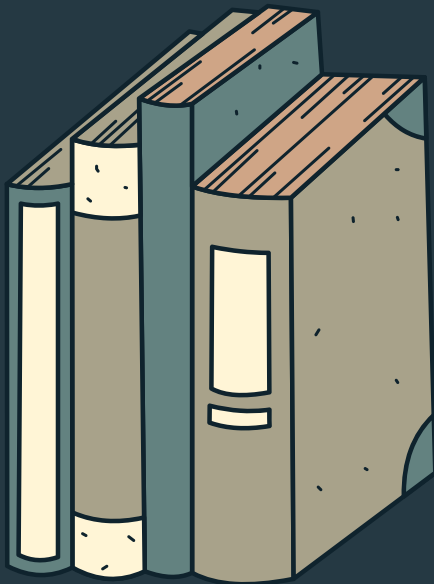
TASK 3: DELETE A RECORD FROM THE ISSUED STATUS TABLE --
OBJECTIVE: DELETE THE RECORD WITH ISSUED_ID = 'IS121' FROM THE ISSUED_STATUS TABLE.

```
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';
```

	issued_id [PK] character varying (10) 	issued_member_id character varying (30) 	issued_book_name character varying (80) 	issued_date date 	issued_book_isbn character varying (50) 	issued_emp_id character varying (10) 
1	IS130	C106	Moby Dick	2024-04-03	978-0-451-52994-2	E101
2	IS131	C106	To Kill a Mockingbird	2024-04-04	978-0-06-112008-4	E101

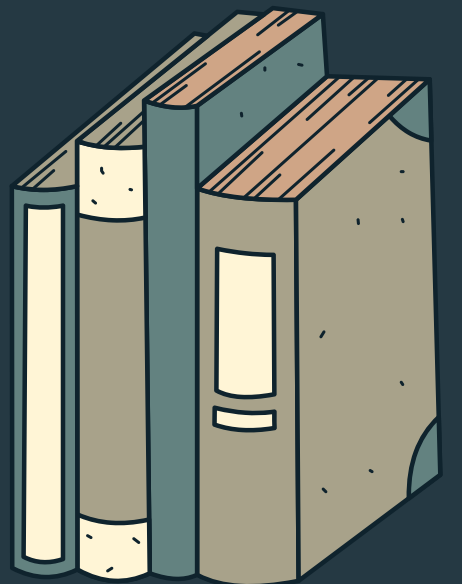


TASK 5: LIST MEMBERS WHO HAVE ISSUED MORE THAN ONE BOOK -- OBJECTIVE: USE GROUP BY TO FIND MEMBERS WHO HAVE ISSUED MORE THAN ONE BOOK.

```
SELECT
    ist.issued_emp_id,
    e.emp_name
    -- COUNT(*)
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
```

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;
```



DATA ANALYSIS & FINDINGS

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 books as b  
JOIN  
issued_status as ist  
ON ist.issued_book_isbn = b.isbn  
GROUP BY 1
```

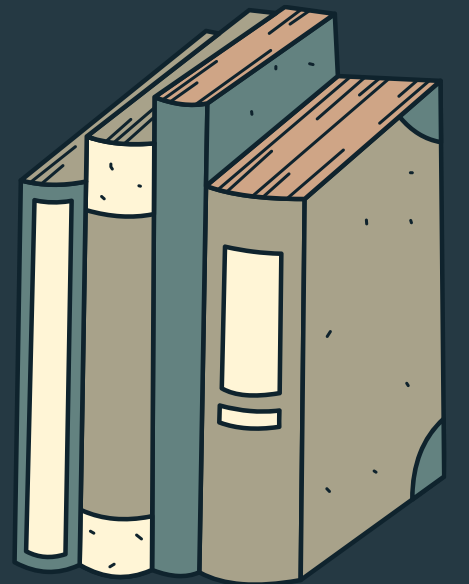


TASK 9: LIST MEMBERS WHO REGISTERED IN THE LAST 180 DAYS:

```
SELECT * FROM members
WHERE reg_date >= CURRENT_DATE - INTERVAL '180 days'
```

TASK 10: LIST EMPLOYEES WITH THEIR BRANCH MANAGER'S NAME AND THEIR BRANCH DETAILS:

```
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
```



TASK 11: CREATE A TABLE OF BOOKS WITH RENTAL PRICE ABOVE A CERTAIN THRESHOLD:

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

TASK 12: 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 ist.issued_id = rs.issued_id
WHERE rs.return_id IS NULL
```



ADVANCED SQL OPERATIONS

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,
    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 14: 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).

```
-- Store Procedures
CREATE OR REPLACE PROCEDURE add_return_records(p_return_id VARCHAR(10), p_issued_id VARCHAR(10), p_book_quality VARCHAR(10))
LANGUAGE plpgsql
AS $$

DECLARE
    v_isbn VARCHAR(50);
    v_book_name VARCHAR(80);

BEGIN
    -- all your logic and code
    -- inserting into returns based on users input
    INSERT INTO return_status(return_id, issued_id, return_date, book_quality)
    VALUES
    (p_return_id, p_issued_id, CURRENT_DATE, p_book_quality);

    SELECT
        issued_book_isbn,
        issued_book_name
    INTO
        v_isbn,
        v_book_name
    FROM issued_status
    WHERE issued_id = p_issued_id;

    UPDATE books
    SET status = 'yes'
    WHERE isbn = v_isbn;

    RAISE NOTICE 'Thank you for returning the book: %', v_book_name;

END;
$$
```

```
-- Testing FUNCTION add_return_records

issued_id = IS135
ISBN = WHERE isbn = '978-0-307-58837-1'

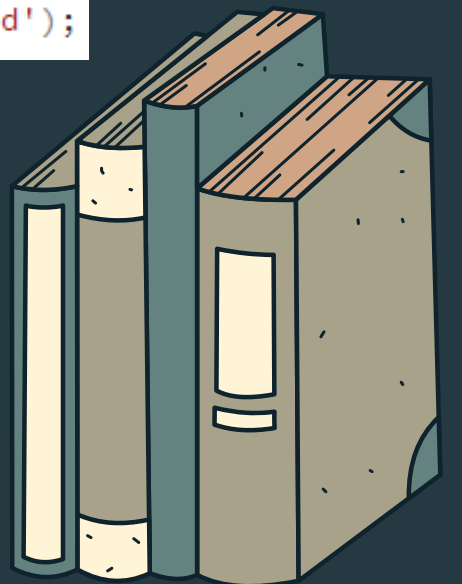
SELECT * FROM books
WHERE isbn = '978-0-307-58837-1';

SELECT * FROM issued_status
WHERE issued_book_isbn = '978-0-307-58837-1';

SELECT * FROM return_status
WHERE issued_id = 'IS135';

-- calling function
CALL add_return_records('RS138', 'IS135', 'Good');

-- calling function
CALL add_return_records('RS148', 'IS140', 'Good');
```



TASK 15: 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 16: 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 * FROM members
WHERE member_id IN (SELECT
                    DISTINCT issued_member_id
                    FROM issued_status
                    WHERE
                        issued_date >= CURRENT_DATE - INTERVAL '2 month'
                    )
;

SELECT * FROM active_members;
```



TASK 17: 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
```



TASK 18: STORED PROCEDURE

OBJECTIVE: CREATE A STORED PROCEDURE TO MANAGE THE STATUS OF BOOKS IN A LIBRARY SYSTEM.

DESCRIPTION: WRITE A STORED PROCEDURE THAT UPDATES THE STATUS OF A BOOK IN THE LIBRARY BASED ON ITS ISSUANCE.

THE PROCEDURE SHOULD FUNCTION AS FOLLOWS: THE STORED PROCEDURE SHOULD TAKE THE BOOK_ID AS AN INPUT PARAMETER.

THE PROCEDURE SHOULD FIRST CHECK IF THE BOOK IS AVAILABLE (STATUS = 'YES').

IF THE BOOK IS AVAILABLE, IT SHOULD BE ISSUED, AND THE STATUS IN THE BOOKS TABLE SHOULD BE UPDATED TO 'NO'.

IF THE BOOK IS NOT AVAILABLE (STATUS = 'NO'), THE PROCEDURE SHOULD RETURN AN ERROR MESSAGE INDICATING THAT THE BOOK IS CURRENTLY NOT AVAILABLE.



```

CREATE OR REPLACE PROCEDURE issue_book(p_issued_id VARCHAR(10), p_issued_member_id VARCHAR(30), p_issued_book_isbn VARCHAR(30), p_issued_emp_id VARCHAR(10))
LANGUAGE plpgsql
AS $$

DECLARE
-- all the variable
    v_status VARCHAR(10);

BEGIN
-- all the code
    -- checking if book is available 'yes'
    SELECT
        status
    INTO
        v_status
    FROM books
    WHERE isbn = p_issued_book_isbn;

    IF v_status = 'yes' THEN

        INSERT INTO issued_status(issued_id, issued_member_id, issued_date, issued_book_isbn, issued_emp_id)
        VALUES
        (p_issued_id, p_issued_member_id, CURRENT_DATE, p_issued_book_isbn, p_issued_emp_id);

        UPDATE books
            SET status = 'no'
        WHERE isbn = p_issued_book_isbn;

        RAISE NOTICE 'Book records added successfully for book isbn : %', p_issued_book_isbn;

    ELSE
        RAISE NOTICE 'Sorry to inform you the book you have requested is unavailable book_isbn: %', p_issued_book_isbn;
    END IF;

END;
$$

```

```

SELECT * FROM books;
-- "978-0-553-29698-2" -- yes
-- "978-0-375-41398-8" -- no
SELECT * FROM issued_status;

CALL issue_book('IS155', 'C108', '978-0-553-29698-2', 'E104');

CALL issue_book('IS156', 'C108', '978-0-375-41398-8', 'E104');

SELECT * FROM books
WHERE isbn = '978-0-375-41398-8'

```





**FOR MORE DETAILS AND TO
ACCESS THE FILES, CHECK OUT
MY GITHUB:
HLIBRARY-MANAGEMENT-
SYSTEM-SQ**