Analysis of

Library Management System Using SQL

Meet the Librarian

Mr. WALEED ILYAS

Welcome to the Library!

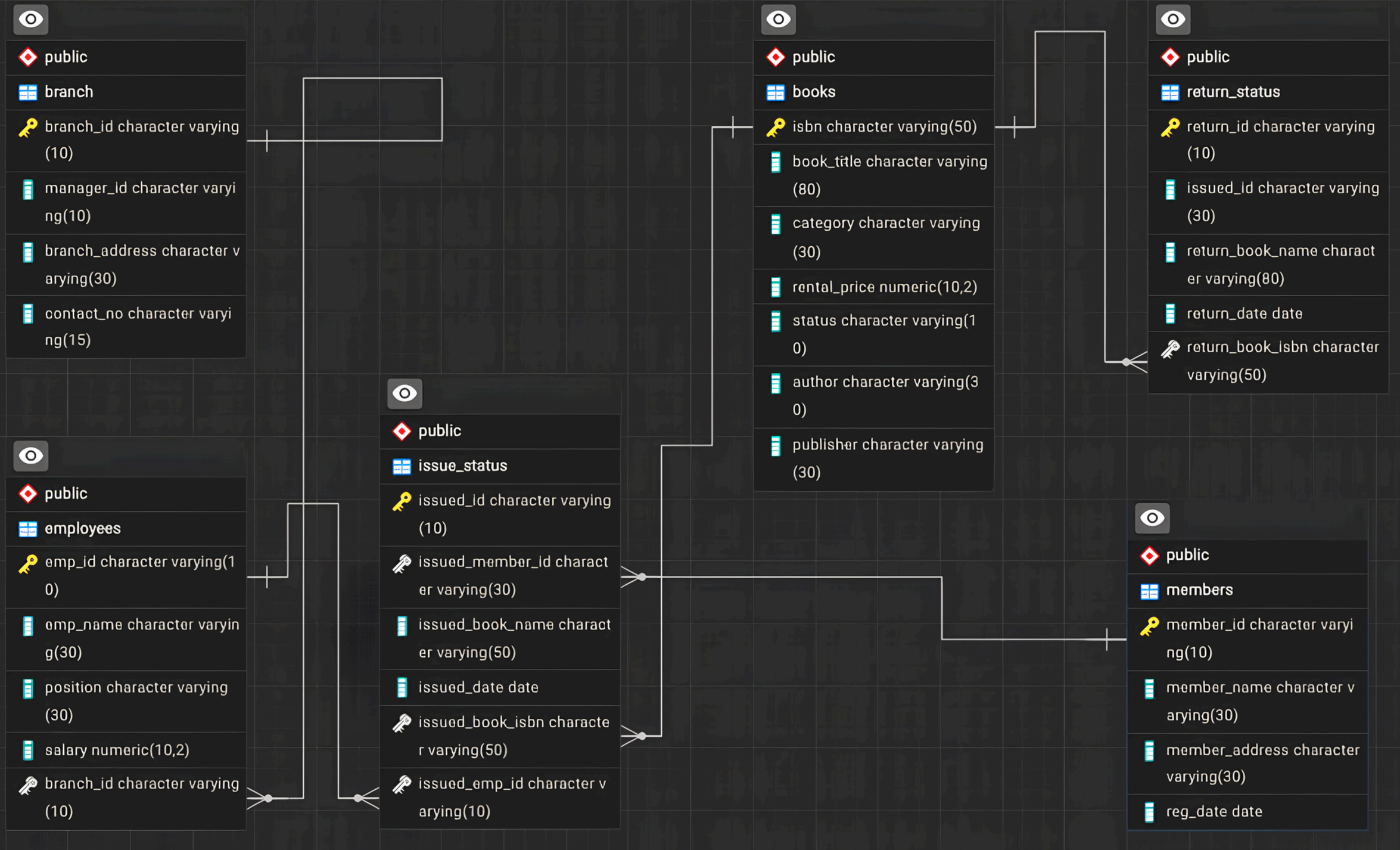
Project Overview

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.

DATA MODELING


public
branch
branch_id character varying (10)
manager_id character varying(10)
branch_address character varying(30)
contact_no character varying(15)
public
books
isbn character varying(50)
book_title character varying (80)
category character varying (30)
rental_price numeric(10,2)
status character varying(10)
author character varying(30)
publisher character varying (30)
public
return_status
return_id character varying (10)
issued_id character varying (30)
return_book_name character varying(80)
return_date date
return_book_isbn character varying(50)
public
issue_status
issued_id character varying (10)
issued_member_id character varying(30)
issued_book_name character varying(50)
issued_date date
issued_book_isbn character varying(50)
issued_emp_id character varying(10)
public
employees
emp_id character varying(10)
emp_name character varying(30)
position character varying (30)
salary numeric(10,2)
branch_id character varying (10)
public
members
member_id character varying(10)
member_name character varying(30)
member_address character varying(30)
reg_date date

1. Table Creation

Created tables for branches, employees, members, books, issued status, and return status. Each table includes relevant columns and relationships.

```sql
-- Creating table "Branch"
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)
);
```

```sql
-- 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)
);
```

```sql
-- 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
);
```

```sql
-- 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)
);
```

```sql
-- 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)
);
```

```sql
-- 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 employee's table.
- **Delete:** Records were removed 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.'"

```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;
```

TASK # 2

Update an Existing Member's Address

```sql
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.

```sql
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)
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

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
```

issued_emp_id character varying (10)	count bigint
E105	4
E101	2
E107	3
E102	2
E110	6
E109	2
E104	4
E106	6
E108	4

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;

SELECT * FROM public.book_issued_cnt
```

Result of TASK # 6

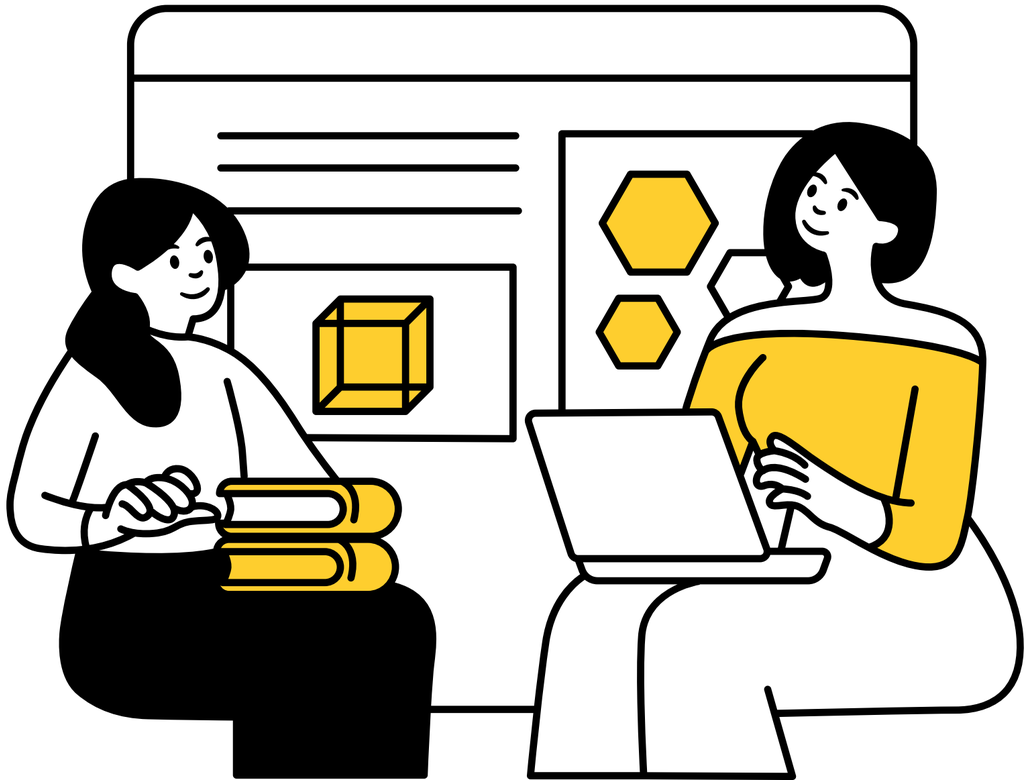
isbn character varying (50)	book_title character varying (80)	issue_count bigint
978-0-375-41398-8	The Diary of a Young Girl	1
978-0-393-91257-8	Guns, Germs, and Steel: The Fates of Human Societies	1
978-0-679-77644-3	Beloved	1
978-0-7432-7356-4	The Hobbit	1
978-0-679-76489-8	Harry Potter and the Sorcerers Stone	2
978-0-06-112241-5	The Kite Runner	1
978-0-7432-4722-5	Angels & Demons	1
978-0-330-25864-8	Animal Farm	2
978-0-452-28240-7	Brave New World	1
978-0-307-58837-1	Sapiens: A Brief History of Humankind	1
978-0-14-143951-8	Pride and Prejudice	1
978-0-14-118776-1	One Hundred Years of Solitude	1
978-0-525-47535-5	The Great Gatsby	2
978-0-06-112008-4	To Kill a Mockingbird	1
978-0-451-52994-2	Moby Dick	1
978-0-307-37840-1	The Alchemist	1

4. Data Analysis & Findings

TASK # 7

Retrieve All Books in a Classic Category

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



book_title character varying (80)	category character varying (30)
The Catcher in the Rye	Classic
Animal Farm	Classic
The Great Gatsby	Classic
Jane Eyre	Classic
Pride and Prejudice	Classic
A Tale of Two Cities	Classic
Moby Dick	Classic
To Kill a Mockingbird	Classic
To Kill a Mockingbird	Classic

TASK # 8

Find Total Rental Income by Category

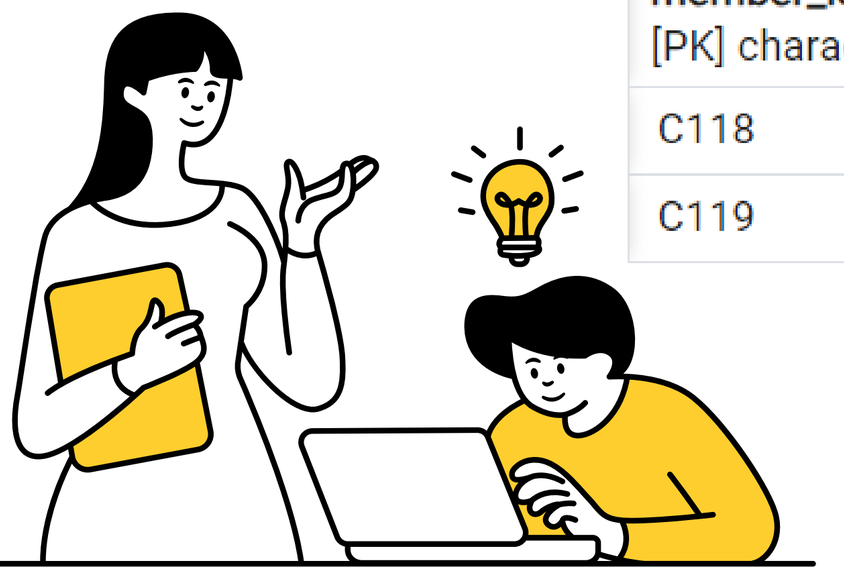
```sql
SELECT
    b.category,
    SUM(b.rental_price),
    COUNT(*)
FROM
issued_status as ist
JOIN
books as b
ON b.isbn = ist.issued_book_isbn
GROUP BY 1
```

category character varying (30)	sum numeric	count bigint
History	49.50	7
Fantasy	28.50	4
Dystopian	25.50	4
Horror	7.00	1
Literary Fiction	6.50	1
Mystery	7.50	1
Classic	59.00	10
Children	7.50	2
Science Fiction	8.50	1
Fiction	14.50	3

TASK # 9

List Members Who Registered in the Last 180 Days

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



member_id [PK] character varying (10)	member_name character varying (30)	member_address character varying (30)	reg_date date
C118	Sam	133 Pine St	2024-06-01
C119	John	143 Main St	2024-05-01

TASK # 10

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

```sql
SELECT
    e1.emp_id,
    e1.emp_name,
    e1.position,
    b.branch_id,
    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;
```

Result of TASK # 10

emp_id character varying (10)	emp_name character varying (30)	position character varying (30)	branch_id character varying (10)	manager character varying (30)
E101	John Doe	Clerk	B001	Daniel Anderson
E102	Jane Smith	Clerk	B002	Daniel Anderson
E103	Mike Johnson	Librarian	B001	Daniel Anderson
E104	Emily Davis	Assistant	B001	Daniel Anderson
E105	Sarah Brown	Assistant	B001	Daniel Anderson
E106	Michelle Ramirez	Assistant	B001	Daniel Anderson
E107	Michael Thompson	Clerk	B005	Laura Martinez
E108	Jessica Taylor	Clerk	B004	Laura Martinez
E109	Daniel Anderson	Manager	B003	Daniel Anderson
E110	Laura Martinez	Manager	B005	Laura Martinez
E111	Christopher Lee	Assistant	B005	Laura Martinez

TASK # 11

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
```

book_title character varying (80)	category character varying (30)	rental_price numeric (10,2)
The Great Gatsby	Classic	8.00
The Da Vinci Code	Mystery	8.00
A Game of Thrones	Fantasy	7.50
A Peoples History of the United States	History	9.00
Sapiens: A Brief History of Humankind	History	8.00
Dune	Science Fiction	8.50
Angels & Demons	Mystery	7.50

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 rs.issued_id = ist.issued_id
WHERE rs.return_id IS NULL;
```

issued_book_name character varying (80)
1491: New Revelations of the Americas Before Columbus
A Tale of Two Cities
Angels & Demons
Animal Farm
Beloved
Charlotte's Web
Dune
Fahrenheit 451
Harry Potter and the Sorcerers Stone
Moby Dick
Sapiens: A Brief History of Humankind

ADVANCE 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.

```sql
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
```

Result of TASK # 13

issued_member_id character varying (30)	member_name character varying (30)	book_title character varying (80)	issued_date date	over_dues_days integer
C102	Bob Smith	Fahrenheit 451	2024-03-26	163
C103	Carol Davis	Dune	2024-03-27	162
C104	Dave Wilson	Where the Wild Things Are	2024-03-28	161
C105	Eve Brown	Charlotte's Web	2024-03-30	159
C105	Eve Brown	The Kite Runner	2024-03-29	160
C105	Eve Brown	Beloved	2024-03-31	158
C105	Eve Brown	A Tale of Two Cities	2024-04-01	157
C105	Eve Brown	The Stand	2024-04-02	156
C106	Frank Thomas	To Kill a Mockingbird	2024-04-04	154
C106	Frank Thomas	The Hobbit	2024-04-05	153
C106	Frank Thomas	Moby Dick	2024-04-03	155
C107	Grace Taylor	Sapiens: A Brief History of Humankind	2024-04-08	150
C107	Grace Taylor	1491: New Revelations of the Americas Before Columbus	2024-04-09	149
C107	Grace Taylor	The Catcher in the Rye	2024-04-10	148
C107	Grace Taylor	Angels & Demons	2024-04-06	152

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.

```sql
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;
```

Result of TASK # 14

branch_id character varying (10)	manager_id character varying (10)	number_book_issued bigint	number_of_book_return bigint	total_revenue numeric
B004	E110	4	3	26.50
B002	E109	2	0	12.00
B003	E109	2	0	14.00
B001	E109	17	9	111.50
B005	E110	9	3	50.00

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 * FROM members
WHERE member_id IN (SELECT
DISTINCT issued_member_id
FROM issued_status WHERE
issued_date >= CURRENT_DATE
- INTERVAL '5 month'
                    );

SELECT * FROM active_members;
```

member_id character varying (10)	member_name character varying (30)	member_address character varying (30)	reg_date date
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

TASK # 16

Find Employees with the Most Book Issues Processed

Write a query to find the top **6** employees who have processed the most book issues. Display the employee name, number of books processed, and their branch.

```sql
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
limit 6
```

Result of TASK # 16

emp_name character varying (30)	branch_id character varying (10)	manager_id character varying (10)	branch_address character varying (30)	contact_no character varying (15)	no_book_issued bigint
Jessica Taylor	B004	E110	567 Pine St	+919099988679	4
John Doe	B001	E109	123 Main St	+919099988676	2
Mike Johnson	B001	E109	123 Main St	+919099988676	1
Laura Martinez	B005	E110	890 Maple St	+919099988680	6
Michael Thompson	B005	E110	890 Maple St	+919099988680	3
Emily Davis	B001	E109	123 Main St	+919099988676	4

THANK YOU!!!

- **I appreciate your time and attention in reviewing my SQL project.**
- **If you have any questions or feedback, feel free to reach out!**

EMAIL
waleedilyas99@gmail.com

MOBILE NO.
+92 3176063654