📚 Library Management System: SQL Project

📋 Project Introduction

This project is a comprehensive Data Analysis and Database Management solution designed for a modern Library Management System. Utilizing **Microsoft SQL Server (T-SQL)**, this project simulates a real-world database environment where data is used to track inventory, manage member activities, process rentals, and generate financial insights.

The core objective of this project was not just to store data, but to solve **20 specific business interaction problems**, ranging from simple record management to complex automated workflows using Stored Procedures and advanced reporting using CTAS (Create Table As Select) and CTEs (Common Table Expressions).

🚀 Project Objectives

The primary goals of this SQL project were to:

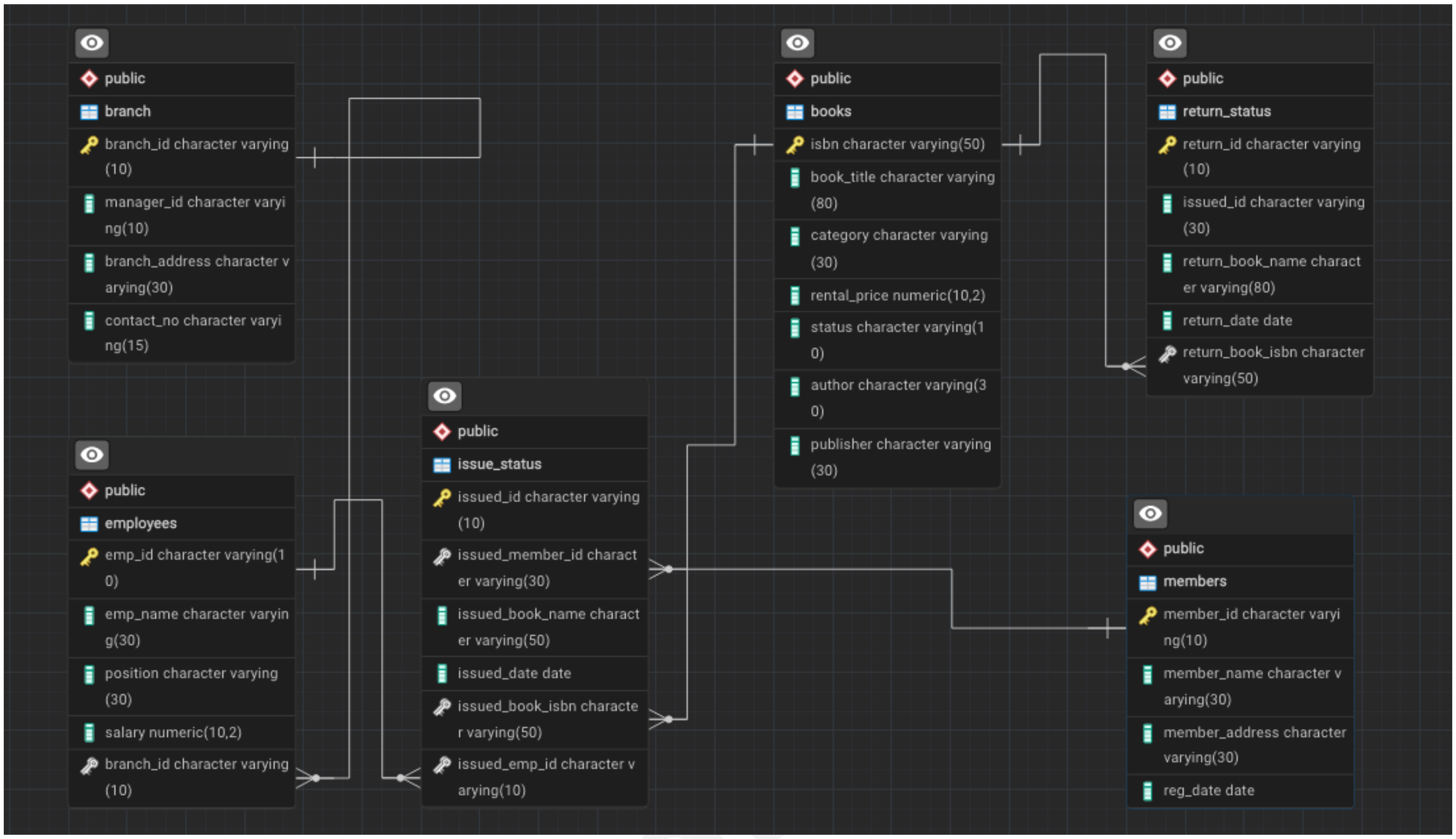
1. **Normalize and Organize Data**: Establish a relational schema connecting Books, Members, Employees, and Branches.
2. **Automate Workflows**: Use Stored Procedures to handle book issuance and return status updates automatically.
3. **Analyze Business Performance**: Track rental revenue, identify top-performing branches, and calculate overdue fines.
4. **Monitor Member Activity**: Identify active members and those with overdue or damaged books.

📂 Database Schema & Architecture

The system is built on a robust relational database schema consisting of the following tables:

- **BooksNew**: The central inventory table containing ISBN, title, category, rental price, status ('yes'/'no'), author, and publisher.
- **Branch**: Stores branch details, including manager information and contact details.
- **Employees**: Tracks staff data, including position, salary, and assigned branch.
- **Members**: Maintains customer profiles, registration dates, and address information.
- **Issued_Status**: A transactional table recording every book issuance, linking members to books and employees.
- **Return_Status**: Tracks returned books, return dates, and the condition of the book (Good/Damaged).

> **Note:** A detailed ER (Entity Relationship) Diagram is recommended to visualize the foreign key constraints between these tables.


public
branch
branch_id character varying(10)
manager_id character varying(10)
branch_address character varying(30)
contact_no character varying(15)
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
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
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
members
member_id character varying(10)
member_name character varying(30)
member_address character varying(30)
reg_date date

🧩 The 20 Problem Statements Solved

This project addresses **20 specific data challenges** split into three complexity levels:

🟢 Level 1: Basic Operations (CRUD & filtering)

1. **New Book Record**: Inserted new book data into the system.
2. **Update Member Info**: Updated addresses for specific members.
3. **Delete Records**: Removed erroneous records from the issued_status table.
4. **Retrieve Issued Books**: Selected all books issued by a specific employee.
5. **Filter by Category**: Retrieved all books classified as 'Classic'.
6. **Find Total Rental Income**: Calculated revenue generated by book category.
7. **List Recent Registrations**: Identified members who registered in the last 180 days.
8. **List Employees**: Retrieved employee details along with their branch manager's name.

🟡 Level 2: Intermediate Analysis (Joins & Aggregations)

9. **Member Activity**: Listed members who have issued more than one book.
10. **Branch Performance**: Created a report showing total books issued/returned and revenue per branch.

11. **CTAS - Book Counts**: Created a summary table book_issued_cnt tracking issuance frequency.
12. **CTAS - Expensive Books**: Generated a table expensive_books for items with rental price > $7.00.
13. **CTAS - Active Members**: Created a table active_members for users active in the last 2 months.
14. **Overdue Books Analysis**: Identified transactions where books were not returned within the 30-day window.

🔴 Level 3: Advanced SQL (Stored Procedures, Window Functions, CTEs)

15. **Branch Performance Table**: Used SELECT INTO to create a persistent performance report table.
16. **Top Employees**: Found the top 3 employees who processed the most book issues.
17. **High-Risk Members (CTE)**: Used a Common Table Expression to identify members who repeatedly return damaged books.
18. **Stored Procedure - Issue Book**: Created a procedure to verify book availability before issuing.
19. **Stored Procedure - Return Book**: Created a procedure to update book status upon return.
20. **Automated Fine Calculation**: Generated a comprehensive report listing overdue books and calculating fines ($0.50/day).

💻 Key SQL Implementations

Below are highlights of the advanced logic implemented in this project.

1. Automated Book Issuance (Stored Procedure)

This procedure ensures data integrity by checking if a book is actually available (status = 'yes') before allowing it to be issued.

```
CREATE OR ALTER 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)
)
AS
BEGIN
   DECLARE @v_status VARCHAR(10);

   -- Check status
   SELECT @v_status = status FROM BooksNew WHERE isbn = @p_issued_book_isbn;
```

```
    IF @v_status = 'yes'
    BEGIN
        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, GETDATE(), @p_issued_book_isbn,
@p_issued_emp_id);

        -- Update book status to 'no'
        UPDATE BooksNew SET status = 'no' WHERE isbn = @p_issued_book_isbn;
        PRINT 'Book issued successfully.';
    END
    ELSE
    BEGIN
        PRINT 'Book is currently unavailable.';
    END
END;
```

2. Overdue Fines Calculation (Data Analysis)

This query identifies members with overdue books and calculates the exact fine based on the number of days passed the 30-day limit.

```
SELECT
    m.member_id,
    COUNT(ist.issued_id) AS number_of_overdue_books,
    SUM((DATEDIFF(DAY, ist.issued_date, GETDATE()) - 30) * 0.50) AS total_fines
INTO overdue_books_summary
FROM issued_status ist
JOIN members m ON ist.issued_member_id = m.member_id
LEFT JOIN return_status rs ON ist.issued_id = rs.issued_id
WHERE rs.issued_id IS NULL -- Not returned
AND DATEDIFF(DAY, ist.issued_date, GETDATE()) > 30 -- Overdue
GROUP BY m.member_id;
```

📈 Business Summary & Insights

By analyzing the data generated through these SQL queries, the following business insights were derived:

1. **Revenue Optimization**: The analysis of rental income by category revealed which book genres are the most profitable, allowing for better budget allocation for future book purchases.
2. **Operational Efficiency**: The implementation of Stored Procedures for issuing and returning books reduced manual data entry errors and ensured real-time inventory accuracy.
3. **Risk Management**: By identifying members who frequently return damaged books or hold overdue items, the library can enforce stricter lending policies for high-risk accounts.
4. **Branch Performance**: The branch performance report highlighted disparities in workload and revenue, suggesting a need for resource redistribution between high-traffic and low-traffic branches.

🛠️ Technology Stack

- **Database Engine**: Microsoft SQL Server (T-SQL)
- **Development Tool**: SQL Server Management Studio (SSMS)
- **Key Concepts**: Stored Procedures, Triggers, CTAS, CTEs, Window Functions, Joins.

📝 Conclusion

This project demonstrates the capability to handle complex database requirements, from basic CRUD operations to advanced analytical reporting and automated process handling. It serves as a testament to the power of SQL in solving real-world business problems.

Created by [Sudipta Biswas]