Library Management System with SQL

Overview

This project implements a **Library Management System** using SQL. It involves designing and managing a relational database, executing CRUD operations, and working with advanced SQL queries. The system is built to efficiently manage information about library branches, employees, members, books, and their issuance/return status.

Objectives

- **Database Creation**: Set up a relational database to manage library operations.
- **Data Manipulation**: Perform Create, Read, Update, and Delete (CRUD) operations on various tables.
- **Table Transformations**: Use SQL features like `CREATE TABLE AS SELECT (CTAS)` to generate derived tables.
- **Query Development**: Write advanced SQL queries to retrieve and analyze data for practical scenarios

Database Details

Database Name: `library_management`

Tables Created:

1. **Branch**: Stores details about library branches, including manager and contact information.
2. **Employees**: Tracks employee data such as position, salary, and branch assignment.
3. **Members**: Maintains records of library members, including registration dates.
4. **Books**: Catalogs library books with attributes like category, rental price, and author details.
5. **Issue Status**: Manages information about issued books, linked to members and employees.
6. **Return Status**: Tracks returned books and their corresponding issued records.

Sample Table Creation:

```
CREATE DATABASE library_db;

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)
);
```

Key Functionalities

1. CRUD Operations

- **Create**: Add new records, such as books or members.
- **Read**: Retrieve data for analysis or reporting.
- **Update**: Modify existing records, e.g., member addresses or employee details.
- **Delete**: Remove records, like outdated entries or resolved issued statuses.

Example Query:
Retrieve books issued by a specific employee:

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

---

2. CTAS (Create Table As Select)

Generate summary tables for better insights:

- **Example**: Create a table to store the count of books issued per book:

```
CREATE TABLE book_issued_count 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;
```

---

3. Advanced SQL Queries

- **Find Members with Overdue Books**: Identify members whose book returns are overdue by more than 30 days:

```
SELECT
    ist.issued_member_id,
    m.member_name,
    bk.book_title,
    ist.issued_date,
    CURRENT_DATE - ist.issued_date AS overdue_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;
```

- **Calculate Rental Income by Category**:t

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

Project Highlights

- **Database Design**: Organized tables with appropriate relationships, ensuring data integrity through primary and foreign keys.

- **Real-World Use Cases**: Queries address practical problems like overdue books, income calculations, and branch-wise operations.
- **Scalable Structure**: The design supports future extensions like digital resources or new member types.

Future Scope

- Add features to track library events and schedules.
- Integrate reporting tools for dynamic data visualization.
- Automate overdue notifications for members.

This project demonstrates practical SQL skills, including database management, query optimization, and real-world problem-solving for a library system.
