CityLibrary Management System Project

Name: Saurav Ganguly
Date: 09 Nov 2025

Project Description

It demonstrates the core concepts of database design, data manipulation, stored procedures, triggers, transactions, and query optimizations. The system supports:

- Members management
- Book catalogue and copies
- Loan and return management
- Fine calculation
- Event registration
- Audit logs for security and accountability

Completed Goals

Required Goals (1–8)

1. **Database Schema Design**
    - Created tables: members, books, book_copies, loans, fines, authors, events, event_registrations, audit_log.
    - Implemented appropriate data types, primary and foreign keys, and constraints.
2. **Data Insertion & Sample Data**
    - Inserted sample members, books, copies, and authors for testing.
    - Verified integrity using SELECT queries.
3. **Basic Queries**
    - Retrieved members, active loans, overdue fines, and events.
    - Used aggregate functions and filtering conditions.
4. **Joins & Relationships**

   - Implemented INNER JOIN, LEFT JOIN, RIGHT JOIN queries for combined data retrieval.

5. **Views**
   - Created views for active loans and overdue books.
6. **Stored Procedures**
   - CheckoutBook, ReturnBook, GenerateMemberReport, CalculateFineDays.
7. **Functions**
   - CalculateFineDays to calculate overdue days.
8. **User Input Handling**
   - Stored procedures accept parameters and return messages for user-friendly feedback.

Optional / Bonus Goals (9–12)

9. **Triggers for Data Integrity**
   - after_member_update: Logs member updates.
   - before_loan_insert: Prevents loans for suspended members.
   - before_loan_insert_due_date: Auto-calculates due date.
   - after_loan_return: Auto-creates fines for overdue returns.
   - before_book_delete: Prevents deletion of books with active loans.
10. **Query Performance Optimization**
    - Added indexes for loans, books, members, fines, events, and event_registrations.
    - Optimized inefficient queries using LEFT JOIN and CASE statements.
    - Verified query execution using EXPLAIN.
11. **Transaction Management**
    - Demonstrated safe checkout, fine payment, batch return, and rollback scenarios.
    - Used START TRANSACTION, COMMIT, and ROLLBACK to maintain data consistency.

12. **Error Handling & ACID Compliance**
    - Ensured all multi-step operations either complete fully or revert completely.
    - Logged actions to audit_log for traceability.

Challenges Faced & Solutions

Challenge	**Solution**
IF statements inside transactions failed	Wrapped logic inside stored procedures
Negative fine amounts	Used GREATEST(0, DATEDIFF(...))
Multiple subqueries slowing queries	Optimized using LEFT JOIN and SUM(CASE...)
Trigger errors due to missing columns	Updated to match actual column names (status, condition_status)

What I Learned

- Designing normalized database schemas with multiple relationships.
- Implementing stored procedures, functions, and triggers for automation and data integrity.
- Query optimization using indexes and efficient joins.
- Handling transactions, rollbacks, and multi-step operations safely.
- Logging and auditing changes for security and traceability.