# Library Management System SQL project

## Introduction

This project implements a **Library Management System** using a relational database. It simulates core library operations: managing authors, books, members, loans, and automated notifications. Advanced SQL features—such as **views**, **triggers**, **foreign keys**, and **stored constraints**—are leveraged to ensure data integrity and enable automated processes for loan reminders and overdue alerts.

## Abstract

The goal is to build a robust database schema that accurately tracks:

- **Authors** and their details
- **Books** along with metadata and stock availability
- **Members** and their registration information
- **Loans** of books to members, including borrowing and return statuses
- Automated **due-date notifications** for upcoming or overdue loans

Views summarize loan statuses at any point in time. Triggers monitor loan activity and populate a notifications table for timely alerts.

## Tools Used

- **PostgreSQL** (or MariaDB) – RDBMS for schema design, constraints, triggers, and SQL queries
- **SQL client** – e.g., MySQL Workbench, pgAdmin, phpMyAdmin, or CLI for executing DDL/DML
- **SQL functions** – AUTO_INCREMENT, FOREIGN KEY, ENUM, DATE_ADD, CURRENT_DATE, views, triggers with NEW row context, and INSERT IGNORE

## Steps Involved in Building the Project

1. **Requirement Analysis & Planning**

- Define clear goals: manage books, authors, members, loans, notifications.

### 2. Schema Design & Table Creation

- Define tables with appropriate fields, data types, primary keys, default values, and ENUMs (e.g., notification type).

### 3. Populating Initial Data

- Insert sample authors, books, members, and loan records.
- Populate the booksauthors bridge to reflect book-author relationships.

### 4. Creating Views for Reporting

- Implement borrowedbooks view to list active loans with due dates.
- Create overduebook view to identify loans past their due date.

### 5. Triggers for Automation & Reporting Queries

- Build AFTER INSERT and AFTER UPDATE triggers on loans to automatically insert UPCOMING (2 days before due) or OVERDUE notifications into due_notification.

## Conclusion

This Library Management System:

- Utilizes a **normalized** relational schema with enforced integrity via foreign keys
- Demonstrates **view-based reporting** to surface current and overdue loans
- Automates loan monitoring with **SQL triggers**, populating a notification queue
- Offers a foundation easily extended with features like fines, reservations, or full-text search