Mini project report on

Exam Centre Management System

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering

UE22CS351A – DBMS Project

Submitted by:

AAKANKSH SEELIN	**PES2UG22CS003**
ADITI ROOPESH MIRJI	**PES2UG22CS032**

Under the guidance of
Dr. Suja
PES University
AUG - DEC 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

CERTIFICATE

This is to certify that the mini project entitled

Exam Centre Management System

is a bonafide work carried out by

Aakanksh Seelin	**PES2UG22CS003**
Aditi Roopesh Mirji	**PES2UG22CS032**

In partial fulfilment for the completion of fifth semester DBMS Project (UE22CS351A) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2024 – DEC. 2024. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5th semester academic requirements in respect of project work.

Signature
Dr Suja

DECLARATION

We hereby declare that the DBMS Project entitled **Exam Centre Management System** has been carried out by us under the guidance of **Dr Suja** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – DEC 2024.

Aakanksh Seelin	**PES2UG22CS003**	**Aakanksh.S**
Aditi Roopesh Mirji	**PES2UG22CS032**	**Armirji**

ABSTRACT

The Exam Centre Management System is a comprehensive system that aims to streamline a university library operation, manage resources effectively and provide an interactive interface for users. This system aims to replace traditional methods with an efficient, user-friendly digital solution to simplify the day-to-day functioning of an exam centre. The Exam Management System project is a practical integration of Database Management Systems (DBMS) and Software Engineering concepts, designed to address the real-world challenges of managing a library. The primary focus of this project is to create a software application that manages library operations, with an emphasis on database management through CRUD (Create, Read, Update, Delete) operations. The system will be developed as a web-based application that interacts with a relational database.

The objectives of the project are:

• Implement and manage a relational database for storing and retrieving information related to books, members, and transactions.

• Enable seamless creation, reading, updating, and deletion of database entries directly from the application interface.

• Provide functionalities for recording, issuing, and returning books.

• Create a user-friendly interface that simplifies interactions with the database for both library staff and members.

• Ensure the integrity and security of data through verified access controls.

TABLE OF CONTENTS

1)INTRODUCTION

The Exam Centre Management System will provide a user-friendly interface for managing a library's book inventory, member registrations , and borrowing/returning activities. The system will allow library administrators to perform CRUD operations on the database tables that store information about books, members, and transactions. The system is a multi-user system which will be primarily used by library administrators , but can be accessed by members as well.

The features of the project are:
• Verified Login – Authorized access to the application. Admin, or existing members can seamlessly login to the application. Existing option to add a new member and create a new profile. Primary Login for Admins Only.

• Book Management- Operations to add, search, update, and delete book records in the database.

• Member registration and profile management- Features to manage member records, including registration, updating details, and deletion.

• Book loan, return, and reservation functionality- Track the status of each book. Note down date of borrowing and expected date of return.

• Overdue alerts and fine calculation- If any borrowed book is not returned within the expected date , then display an overdue alert and calculate the expected fine based on the number of exceeded days.

2) **PROBLEM DEFINITION WITH USER REQUIREEMNT SPECIFICATIONS**

The Exam Centre Management System is a comprehensive system that aims to streamline a university library operation, manage resources effectively and provide an interactive interface for users. This system aims to replace traditional methods with an efficient, user-friendly digital solution to simplify the day-to-day functioning of an exam centre.

1) Functional Requirements:

1.1)Authentication

- The system shall allow authorized users to login to the application.
- The system shall restrict access to administrative functionalities to only authorized admins.

1.2)Book Management

- The system shall allow administrators to add new book records to the database.
- The system shall allow administrators and members to view a list of all books in the library.
- The system shall allow administrators to update the details of existing book records.
- The system shall allow administrators to delete book records from the database.
- The system shall allow members to borrow and return books.

1.3)Member Management

- The system shall allow administrators to add new member records to the database.
- The system shall allow administrators to view a list of all members.
- The system shall allow administrators to update the details of existing member records.
- The system shall allow administrators to delete member records from the database

1.4)Transaction Management

- The system shall allow administrators to record the borrowing of a book by a member.
- The system shall allow administrators to record the return of a borrowed book.
- The system shall allow administrators to view a list of all borrowing and return transactions.
- The system shall allow admins to calculate fines in case of delay in return of book.

2)Non-Functional Requirements:

2.1. Usability

- The system shall provide a simple and user-friendly graphical user interface (GUI) that can be easily navigated by users with basic computer skills.

2.2. Security

- The system shall restrict access to administrative functionalities to authorized users only.
- The system shall require user authentication for access to the application.
- The system shall restrict access to the member portal to verified members only.

2.3. Maintainability

- The system shall be designed in a modular manner, allowing easy updates and maintenance of individual components.

2.4. Reliability

- The system shall ensure data integrity and reliability, particularly during database transactions, to prevent data loss or corruption.

3)LIST OF SOFTWARES/TOOLS/PROGRAMMING LANGUAGES USED

- **Planning Tool:** Google Sheets- for initial planning, task distribution, and creating simple timelines.

- **Design Tool:** smartdraw – for creating flowcharts, ER diagrams, and other design diagrams; Figma – for creating a wireframe of the UI if required.

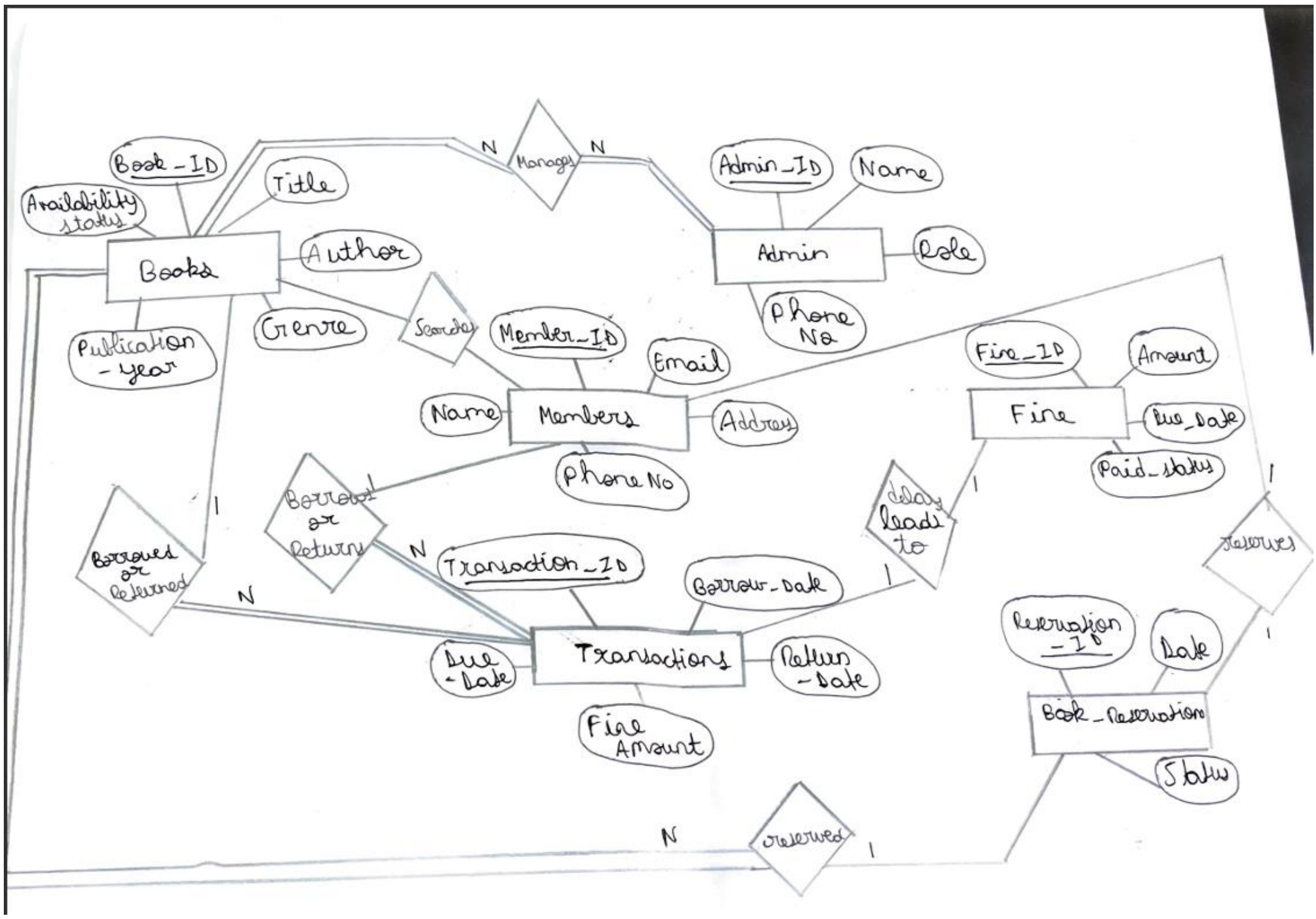
- **Version Control:** Git & GitHub- for managing our source code, maintaining versions, and collaboration.

- **Development Tool:** Python (in VS Code) with Streamlit for quick iterations on the frontend; MySQL for backend development.

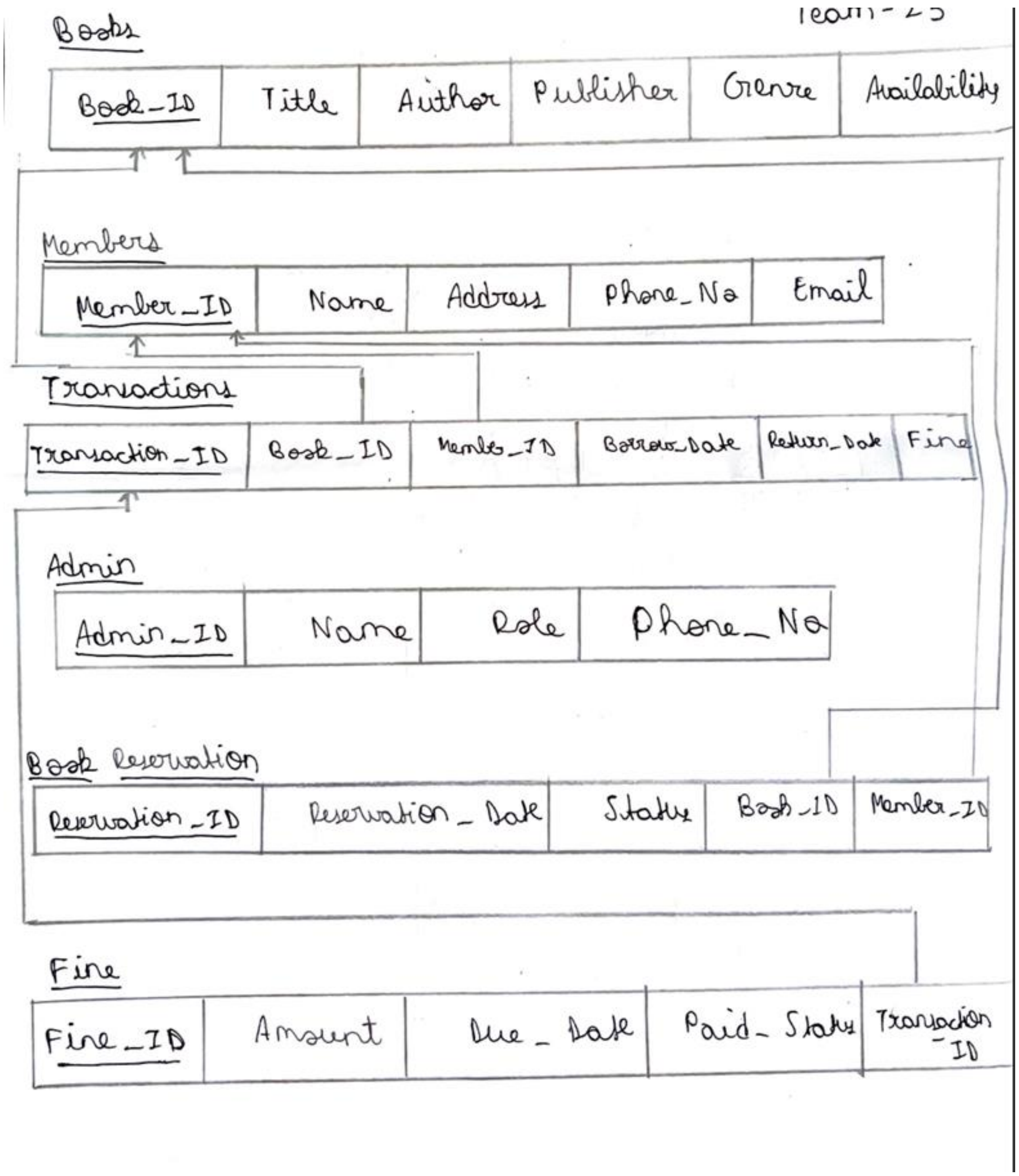
- **Bug Tracking:** GitHub Issues (for tracking bugs and feature requests as they emerge).

- **Testing Tool:** PyTest for unit testing in Python; Selenium for web application testing (if required).

4) ER MODEL


Book_ID
Title
Availability status
Books
Author
Genre
Publication_Year
N
Manages
N
Admin_ID
Name
Admin
Role
Phone No
Searches
Member_ID
Email
Name
Members
Address
Phone No
Fine_ID
Amount
Fine
Due_Date
Paid_status
1
Borrows or Returns
1
Borrowed or Returned
N
N
delay leads to
1
1
reserves
1
Transaction_ID
Borrow_Date
Transactions
Due_Date
Return_Date
Fine Amount
Reservation_ID
Date
Book_Reservation
Status
N
reserved
1

5) ER TO RELATIONAL MAPPING



6)DDL STATEMENTS

The DDL commands have been used in 27 different instances in the project.

DROP DATABASE IF EXISTS lib_mgmt;

- This command drops the lib_mgmt database if it already exists.

CREATE DATABASE lib_mgmt;

- This command creates a new database named lib_mgmt.

USE lib_mgmt;

- This command sets the current database to lib_mgmt.

CREATE TABLE Authors (...)

- This command creates a table named Authors with the specified columns and constraints.

CREATE TABLE Categories (...)

- This command creates a table named Categories with the specified columns and constraints.

CREATE TABLE Books (...)

- This command creates a table named Books with the specified columns and constraints.

CREATE TABLE Administrators (...)

- This command creates a table named Administrators with the specified columns and constraints.

CREATE TABLE Members (...)

- This command creates a table named Members with the specified columns and constraints.

CREATE TABLE MemberTransactions (...)

- This command creates a table named MemberTransactions with the specified columns and constraints.

CREATE TABLE AdminTransactions (...)

- This command creates a table named AdminTransactions with the specified columns and constraints.

CREATE PROCEDURE CheckOutBook(...)

- This command creates a stored procedure named CheckOutBook to handle the book checkout process.

CREATE PROCEDURE ReturnBook(...)

- This command creates a stored procedure named ReturnBook to handle the book return process.

CREATE PROCEDURE BorrowBook(...)

- This command creates a stored procedure named BorrowBook to handle the member book borrowing process.

CREATE PROCEDURE ReturnBook(...)

- This command creates a stored procedure named ReturnBook to handle the member book return process.

CREATE OR REPLACE VIEW BookListView AS ...

- This command creates a view named BookListView that provides a consolidated view of book information.

CREATE OR REPLACE VIEW TransactionDetailsView AS ...

- This command creates a view named TransactionDetailsView that provides a consolidated view of transaction details.

CREATE PROCEDURE DeleteBook(...)

- This command creates a stored procedure named DeleteBook to handle the deletion of books.

CREATE TABLE MemberBorrowingSummary (...)

- This command creates a table named MemberBorrowingSummary to store member borrowing statistics.

CREATE TRIGGER after_member_insert ...

- This command creates a trigger named after_member_insert that is executed after a new member is inserted into the Members table.

CREATE TRIGGER after_transaction_insert ...

- This command creates a trigger named after_transaction_insert that is executed after a new transaction is inserted into the MemberTransactions table.

CREATE TRIGGER after_transaction_update ...

- This command creates a trigger named after_transaction_update that is executed after a transaction in the MemberTransactions table is updated.

CREATE TRIGGER before_borrow_check ...

- This command creates a trigger named before_borrow_check that is executed before a new transaction is inserted into the MemberTransactions table.

CREATE TABLE BookStatusLog (...)

- This command creates a table named BookStatusLog to keep track of changes in book availability status.

CREATE TRIGGER after_book_status_change ...

- This command creates a trigger named after_book_status_change that is executed after a book's availability status is updated in the Books table.

CREATE FUNCTION CalculateTotalFines(...)

- This command creates a function named CalculateTotalFines to calculate the total fines for a member.

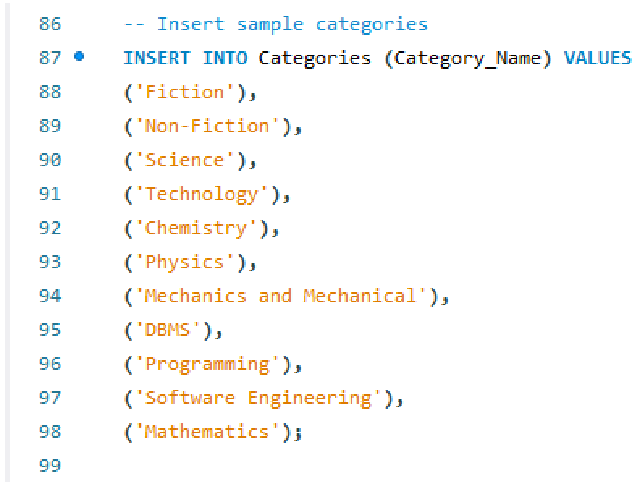
CREATE FUNCTION GetBookAvailabilityDetails(...)

- This command creates a function named GetBookAvailabilityDetails to get the availability status of a book with additional details.

7.DML STATEMENTS

1)**INSERT INTO Categories (Category_Name) VALUES** (...)

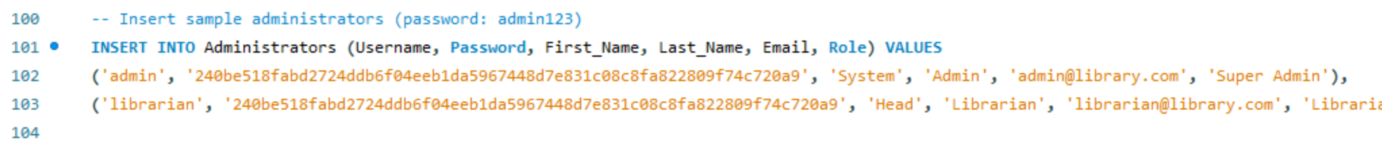
- This statement inserts sample category data into the Categories table.



```sql
-- Insert sample categories
INSERT INTO Categories (Category_Name) VALUES
('Fiction'),
('Non-Fiction'),
('Science'),
('Technology'),
('Chemistry'),
('Physics'),
('Mechanics and Mechanical'),
('DBMS'),
('Programming'),
('Software Engineering'),
('Mathematics');

```

2) **INSERT INTO Administrators (Username, Password, First_Name, Last_Name, Email, Role) VALUES (...)**

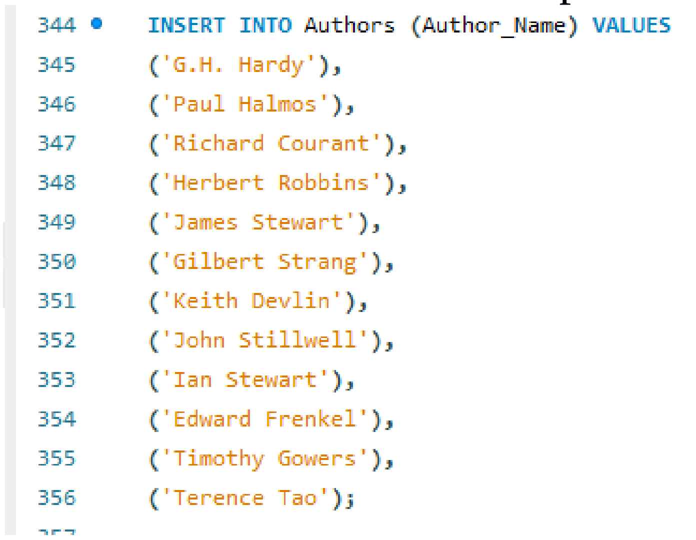
- This statement inserts sample administrator data into the Administrators table.



```sql
-- Insert sample administrators (password: admin123)
INSERT INTO Administrators (Username, Password, First_Name, Last_Name, Email, Role) VALUES
('admin', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'System', 'Admin', 'admin@library.com', 'Super Admin'),
('librarian', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'Head', 'Librarian', 'librarian@library.com', 'Libraria

```

3) **INSERT INTO Authors (Author_Name) VALUES (...)**

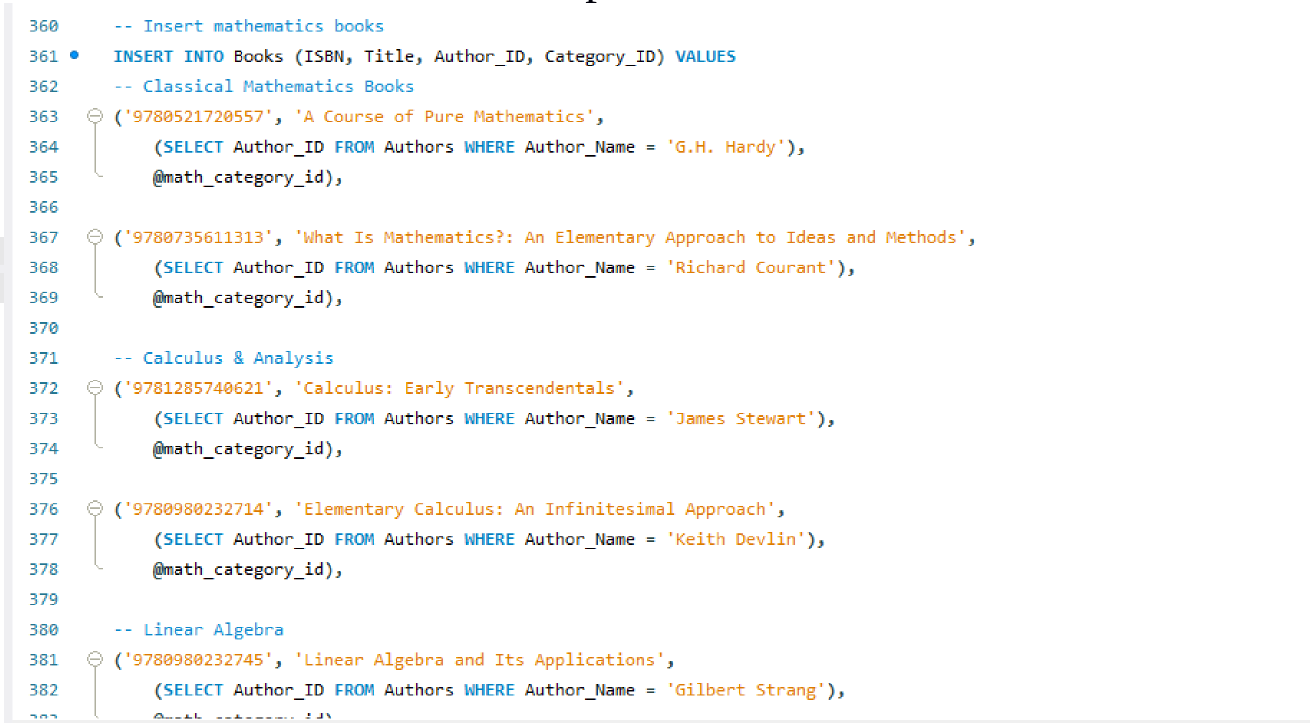
This statement inserts sample author data into the Authors table.



```sql
INSERT INTO Authors (Author_Name) VALUES
('G.H. Hardy'),
('Paul Halmos'),
('Richard Courant'),
('Herbert Robbins'),
('James Stewart'),
('Gilbert Strang'),
('Keith Devlin'),
('John Stillwell'),
('Ian Stewart'),
('Edward Frenkel'),
('Timothy Gowers'),
('Terence Tao');
```

4) **INSERT INTO Books (ISBN, Title, Author_ID, Category_ID) VALUES (...)**

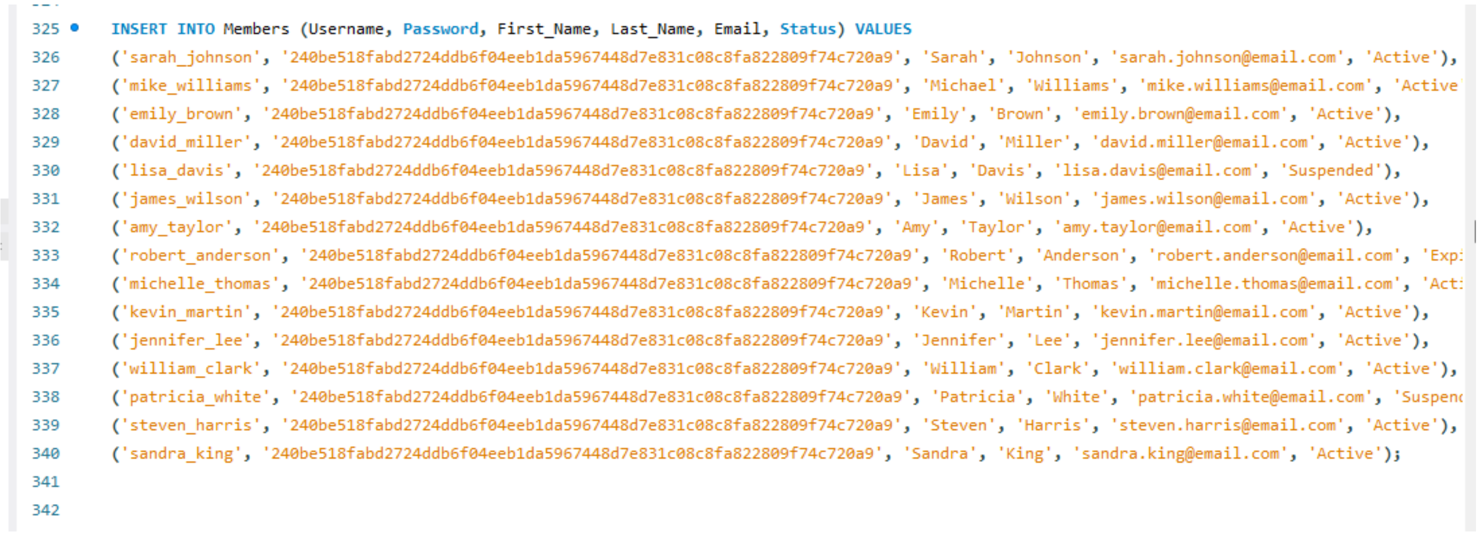
- This statement inserts sample book data into the Books table



```sql
-- Insert mathematics books
INSERT INTO Books (ISBN, Title, Author_ID, Category_ID) VALUES
-- Classical Mathematics Books
('9780521720557', 'A Course of Pure Mathematics',
    (SELECT Author_ID FROM Authors WHERE Author_Name = 'G.H. Hardy'),
    @math_category_id),

('9780735611313', 'What Is Mathematics?: An Elementary Approach to Ideas and Methods',
    (SELECT Author_ID FROM Authors WHERE Author_Name = 'Richard Courant'),
    @math_category_id),

-- Calculus & Analysis
('9781285740621', 'Calculus: Early Transcendentals',
    (SELECT Author_ID FROM Authors WHERE Author_Name = 'James Stewart'),
    @math_category_id),

('9780980232714', 'Elementary Calculus: An Infinitesimal Approach',
    (SELECT Author_ID FROM Authors WHERE Author_Name = 'Keith Devlin'),
    @math_category_id),

-- Linear Algebra
('9780980232745', 'Linear Algebra and Its Applications',
    (SELECT Author_ID FROM Authors WHERE Author_Name = 'Gilbert Strang'),
```

5) **INSERT INTO Members (Username, Password, First_Name, Last_Name, Email, Status) VALUES (...)**

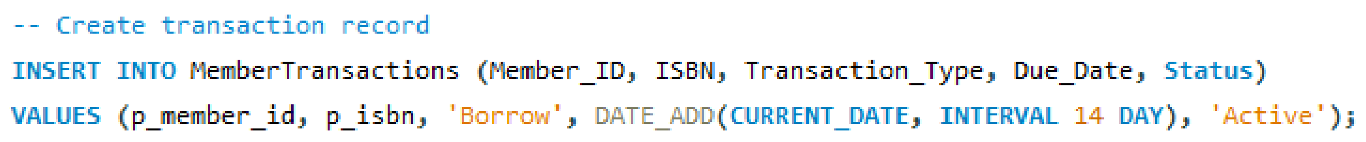
- This statement inserts sample member data into the Members table



```sql
INSERT INTO Members (Username, Password, First_Name, Last_Name, Email, Status) VALUES
('sarah_johnson', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'Sarah', 'Johnson', 'sarah.johnson@email.com', 'Active'),
('mike_williams', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'Michael', 'Williams', 'mike.williams@email.com', 'Active
('emily_brown', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'Emily', 'Brown', 'emily.brown@email.com', 'Active'),
('david_miller', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'David', 'Miller', 'david.miller@email.com', 'Active'),
('lisa_davis', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'Lisa', 'Davis', 'lisa.davis@email.com', 'Suspended'),
('james_wilson', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'James', 'Wilson', 'james.wilson@email.com', 'Active'),
('amy_taylor', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'Amy', 'Taylor', 'amy.taylor@email.com', 'Active'),
('robert_anderson', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'Robert', 'Anderson', 'robert.anderson@email.com', 'Exp
('michelle_thomas', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'Michelle', 'Thomas', 'michelle.thomas@email.com', 'Act
('kevin_martin', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'Kevin', 'Martin', 'kevin.martin@email.com', 'Active'),
('jennifer_lee', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'Jennifer', 'Lee', 'jennifer.lee@email.com', 'Active'),
('william_clark', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'William', 'Clark', 'william.clark@email.com', 'Active'),
('patricia_white', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'Patricia', 'White', 'patricia.white@email.com', 'Suspen
('steven_harris', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'Steven', 'Harris', 'steven.harris@email.com', 'Active'),
('sandra_king', '240be518fabd2724ddb6f04eeb1da5967448d7e831c08c8fa822809f74c720a9', 'Sandra', 'King', 'sandra.king@email.com', 'Active');
```

6) **INSERT INTO MemberTransactions (Member_ID, ISBN, Transaction_Type, Due_Date, Status) VALUES (...)**

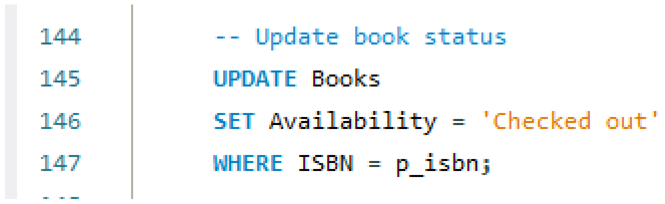
- This statement inserts sample member transaction data into the MemberTransactions table.



```sql
-- Create transaction record
INSERT INTO MemberTransactions (Member_ID, ISBN, Transaction_Type, Due_Date, Status)
VALUES (p_member_id, p_isbn, 'Borrow', DATE_ADD(CURRENT_DATE, INTERVAL 14 DAY), 'Active');
```

7) **UPDATE Books SET Availability = 'Checked out' WHERE ISBN = p_isbn**

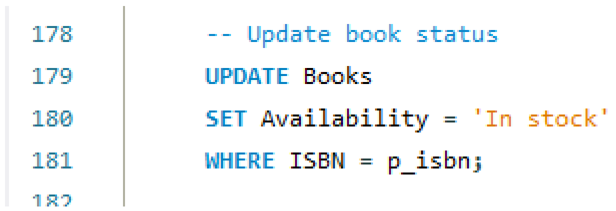
- This statement updates the availability status of a book in the Books table.



```
        -- Update book status
        UPDATE Books
        SET Availability = 'Checked out'
        WHERE ISBN = p_isbn;
```

8) **UPDATE Books SET Availability = 'In stock' WHERE ISBN = p_isbn**

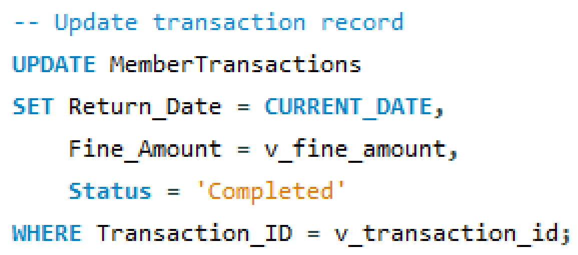
- This statement updates the availability status of a book in the Books table.



```
        -- Update book status
        UPDATE Books
        SET Availability = 'In stock'
        WHERE ISBN = p_isbn;
```

9) **UPDATE MemberTransactions SET Return_Date = CURRENT_DATE, Fine_Amount = v_fine_amount, Status = 'Completed' WHERE Transaction_ID = v_transaction_id**

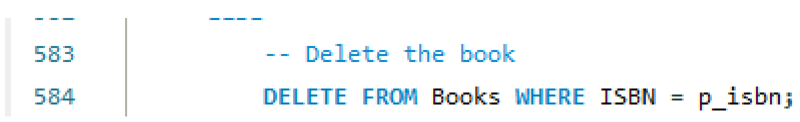
- This statement updates the member transaction record in the MemberTransactions table when a book is returned.



```
-- Update transaction record
UPDATE MemberTransactions
SET Return_Date = CURRENT_DATE,
    Fine_Amount = v_fine_amount,
    Status = 'Completed'
WHERE Transaction_ID = v_transaction_id;
```

10) **DELETE FROM Books WHERE ISBN = p_isbn**

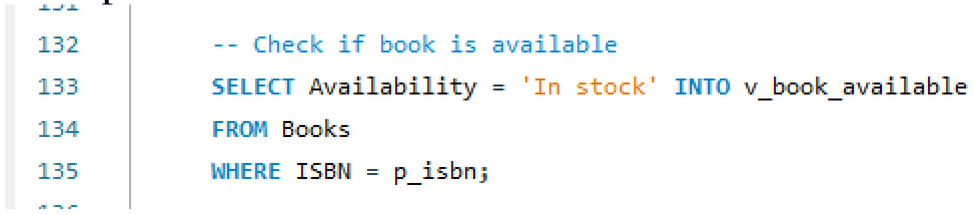
- This statement deletes a book from the Books table.



```
        -- Delete the book
        DELETE FROM Books WHERE ISBN = p_isbn;
```

11) **SELECT Availability = 'In stock' INTO v_book_available FROM Books WHERE ISBN = p_isbn;**

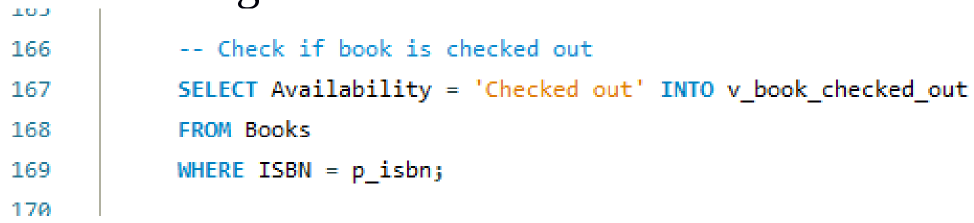
- This statement checks if the book with the given ISBN is available (in stock) and stores the result in the v_book_available variable. It's used in the CheckOutBook procedure to ensure the book is available before checking it out.



```
    -- Check if book is available
    SELECT Availability = 'In stock' INTO v_book_available
    FROM Books
    WHERE ISBN = p_isbn;
```

12) **SELECT Availability = 'Checked out' INTO v_book_checked_out FROM Books WHERE ISBN = p_isbn;**

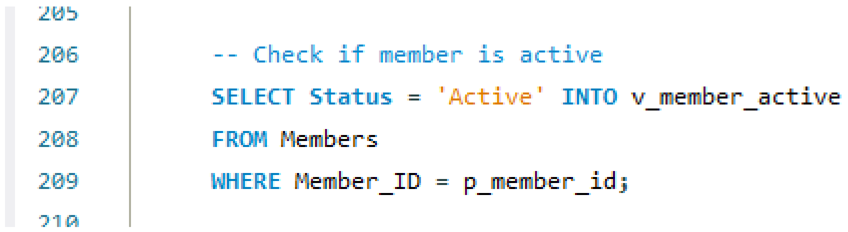
- This statement checks if the book with the given ISBN is currently checked out and stores the result in the v_book_checked_out variable. It's used in the ReturnBook procedure to ensure the book is currently checked out before allowing it to be returned.



```
    -- Check if book is checked out
    SELECT Availability = 'Checked out' INTO v_book_checked_out
    FROM Books
    WHERE ISBN = p_isbn;
```

13) **SELECT Status = 'Active' INTO v_member_active FROM Members WHERE Member_ID = p_member_id;**

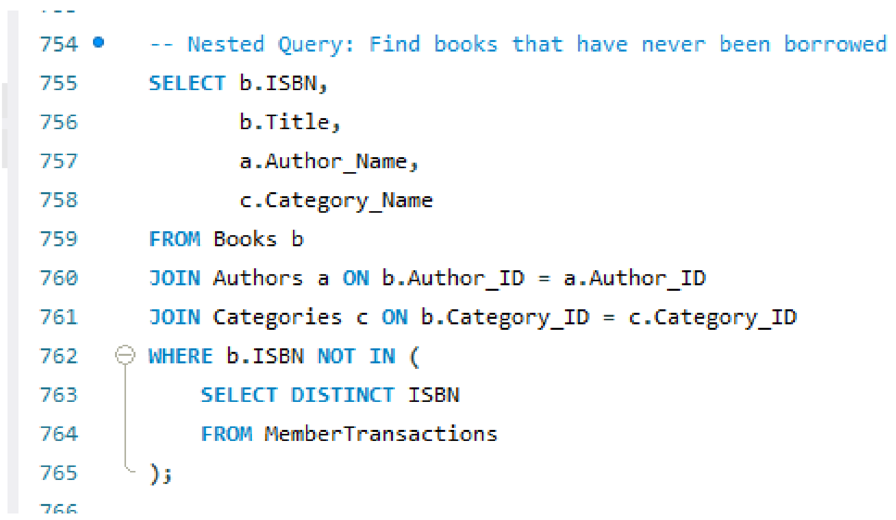
- This statement checks if the member with the given ID has an active status and stores the result in the v_member_active variable. It's used in the BorrowBook procedure to ensure the member is active before allowing them to borrow a book.



```
    -- Check if member is active
    SELECT Status = 'Active' INTO v_member_active
    FROM Members
    WHERE Member_ID = p_member_id;
```

8) QUERIES (JOIN QUERY, AGGREGATE FUNCTION QUERIES AND NESTED QUERY)

<u>Nested JOIN Query:</u>

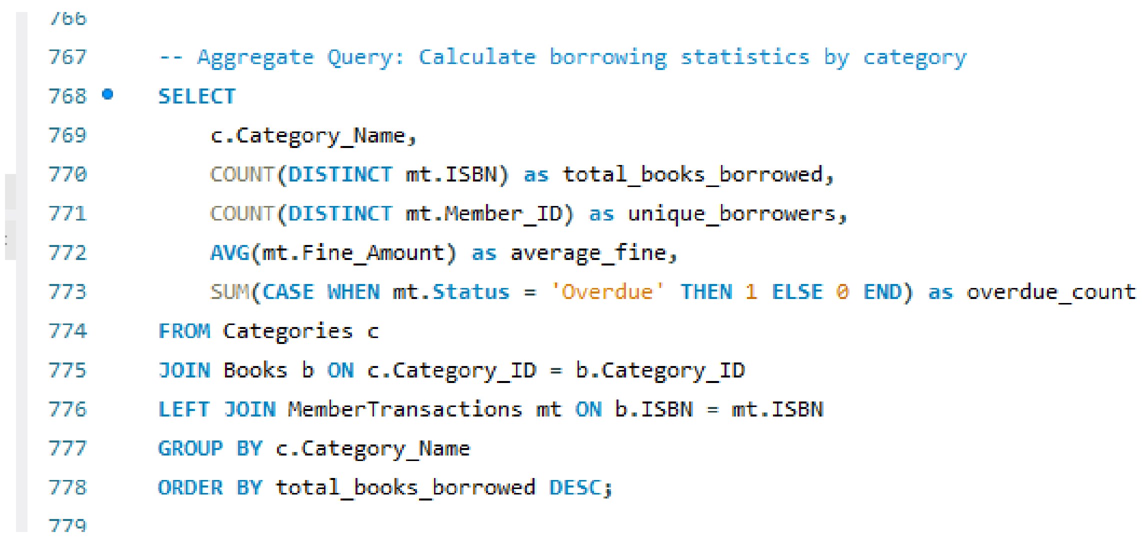


```sql
-- Nested Query: Find books that have never been borrowed
SELECT b.ISBN,
       b.Title,
       a.Author_Name,
       c.Category_Name
FROM Books b
JOIN Authors a ON b.Author_ID = a.Author_ID
JOIN Categories c ON b.Category_ID = c.Category_ID
WHERE b.ISBN NOT IN (
    SELECT DISTINCT ISBN
    FROM MemberTransactions
);
```

This query finds the books that have never been borrowed by members.

- The main query selects the ISBN, Title, Author Name, and Category Name of the books.
- It joins the Books, Authors, and Categories tables to get the necessary information about the books.
- The WHERE clause uses a nested query to check for books that do not have any associated transactions in the MemberTransactions table.
- The nested query selects the distinct ISBNs from the MemberTransactions table. This gives us a list of all the books that have been borrowed at least once.
- The main query then filters the books to only include those whose ISBNs are not present in the nested query's result. This gives us the books that have never been borrowed.

Aggregate Query:

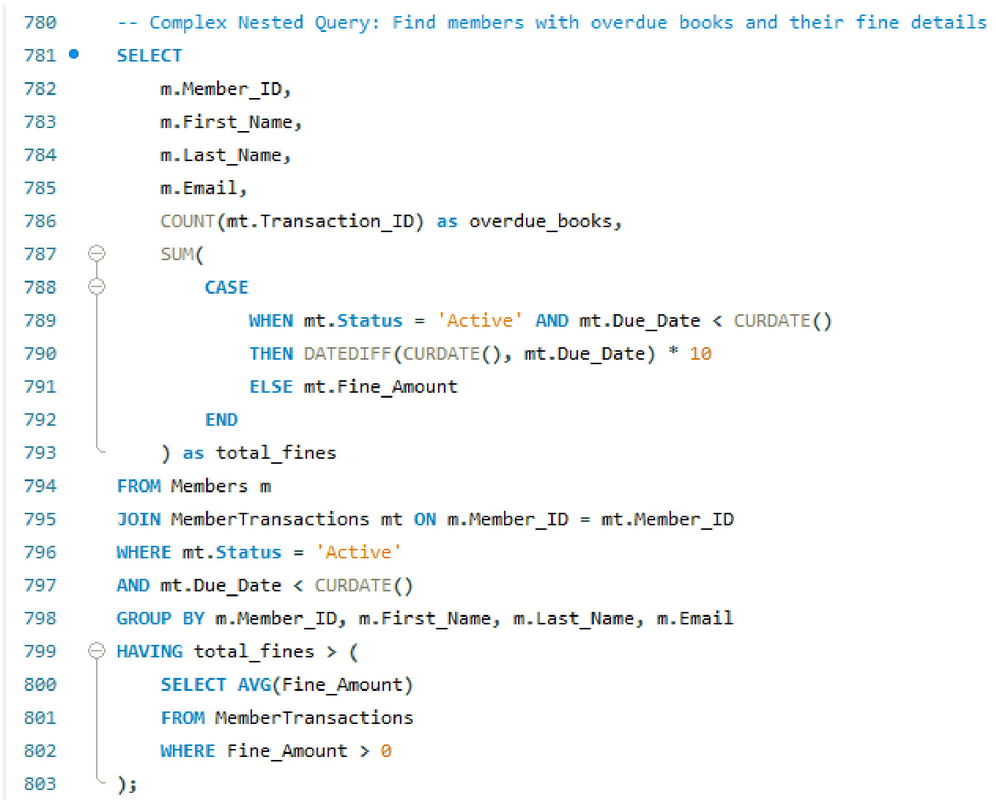


```
-- Aggregate Query: Calculate borrowing statistics by category
SELECT
    c.Category_Name,
    COUNT(DISTINCT mt.ISBN) as total_books_borrowed,
    COUNT(DISTINCT mt.Member_ID) as unique_borrowers,
    AVG(mt.Fine_Amount) as average_fine,
    SUM(CASE WHEN mt.Status = 'Overdue' THEN 1 ELSE 0 END) as overdue_count
FROM Categories c
JOIN Books b ON c.Category_ID = b.Category_ID
LEFT JOIN MemberTransactions mt ON b.ISBN = mt.ISBN
GROUP BY c.Category_Name
ORDER BY total_books_borrowed DESC;
```

This query calculates various borrowing statistics for each category of books.

- The main query selects the Category Name, total books borrowed, unique borrowers, average fine, and the count of overdue books.
- It joins the Categories, Books, and MemberTransactions tables to get the necessary data.
- The LEFT JOIN with the MemberTransactions table ensures that we include all categories, even if they don't have any associated transactions.
- The GROUP BY clause groups the results by the Category Name.
- For each category, it calculates:
    - total_books_borrowed: The count of distinct books borrowed (using DISTINCT mt.ISBN).
    - unique_borrowers: The count of distinct members who have borrowed books (using DISTINCT mt.Member_ID).
    - average_fine: The average fine amount.
    - overdue_count: The count of overdue transactions (using a CASE statement to check the Status column).
- The results are ordered by the total_books_borrowed column in descending order.

Complex Nested Query:



```sql
-- Complex Nested Query: Find members with overdue books and their fine details
SELECT
    m.Member_ID,
    m.First_Name,
    m.Last_Name,
    m.Email,
    COUNT(mt.Transaction_ID) as overdue_books,
    SUM(
        CASE
            WHEN mt.Status = 'Active' AND mt.Due_Date < CURDATE()
            THEN DATEDIFF(CURDATE(), mt.Due_Date) * 10
            ELSE mt.Fine_Amount
        END
    ) as total_fines
FROM Members m
JOIN MemberTransactions mt ON m.Member_ID = mt.Member_ID
WHERE mt.Status = 'Active'
AND mt.Due_Date < CURDATE()
GROUP BY m.Member_ID, m.First_Name, m.Last_Name, m.Email
HAVING total_fines > (
    SELECT AVG(Fine_Amount)
    FROM MemberTransactions
    WHERE Fine_Amount > 0
);
```

This complex nested query finds members who have overdue books and calculates the total fines they owe.

- The main query selects the Member ID, First Name, Last Name, Email, the count of overdue books, and the total fines owed.
- It joins the Members and MemberTransactions tables to get the necessary information about the members and their transactions.
- The WHERE clause filters to only include active transactions that are overdue.
- The GROUP BY clause groups the results by the member's identification columns (Member ID, First Name, Last Name, Email).
- The COUNT aggregate function is used to get the count of overdue books for each member.
- The SUM with a CASE statement is used to calculate the total fines owed by each member. If the transaction is active and overdue, it

calculates the fine based on the number of days overdue . Otherwise, it simply uses the Fine_Amount column.

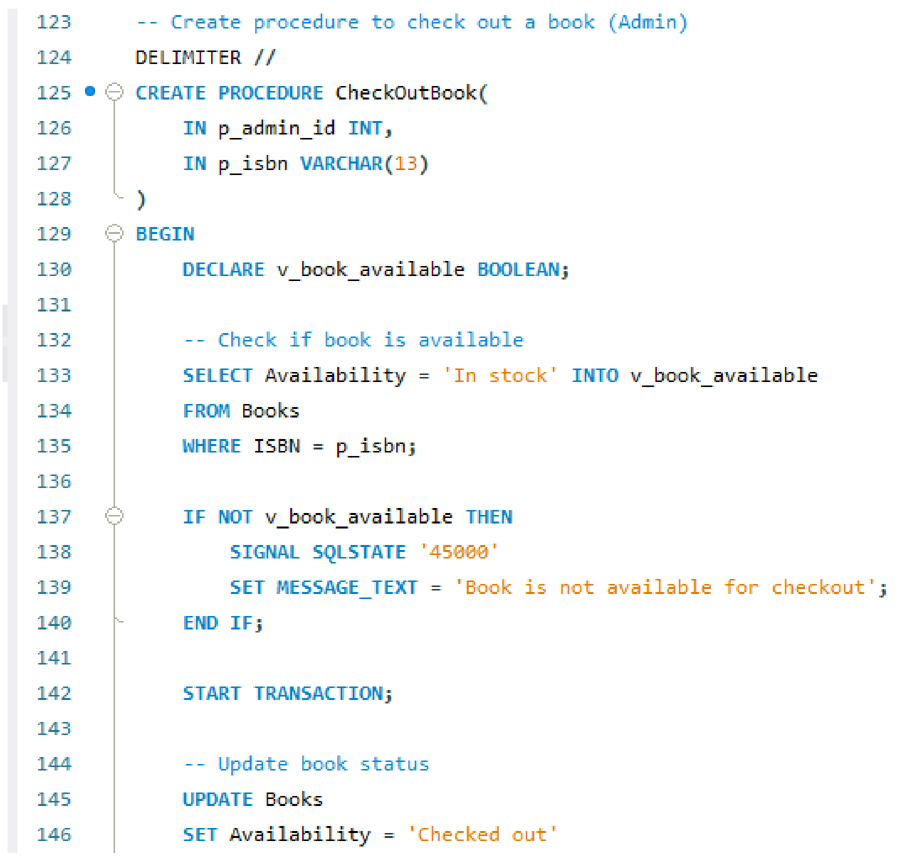
- The HAVING clause further filters the results to only include members whose total fines are above the average fine amount.

This query provides a comprehensive view of the members with overdue books and the associated fine details, allowing the library management system to identify and follow up with members who need to return overdue books and pay their fines.

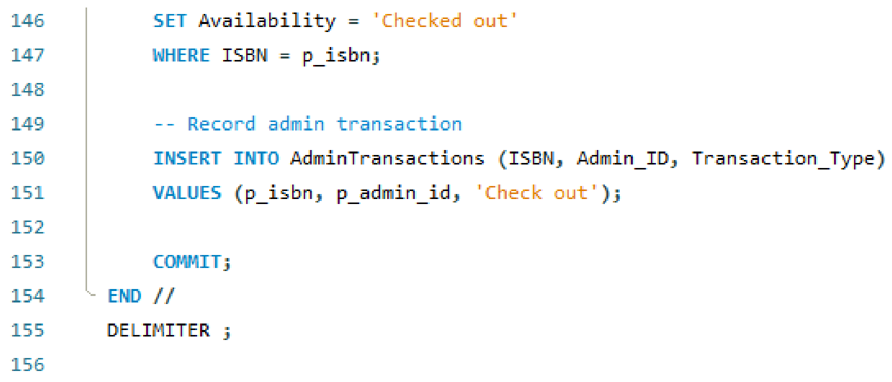
9) STORED PROCEDURE, FUNCTIONS AND TRIGGERS

Stored Procedures

CheckOutBook



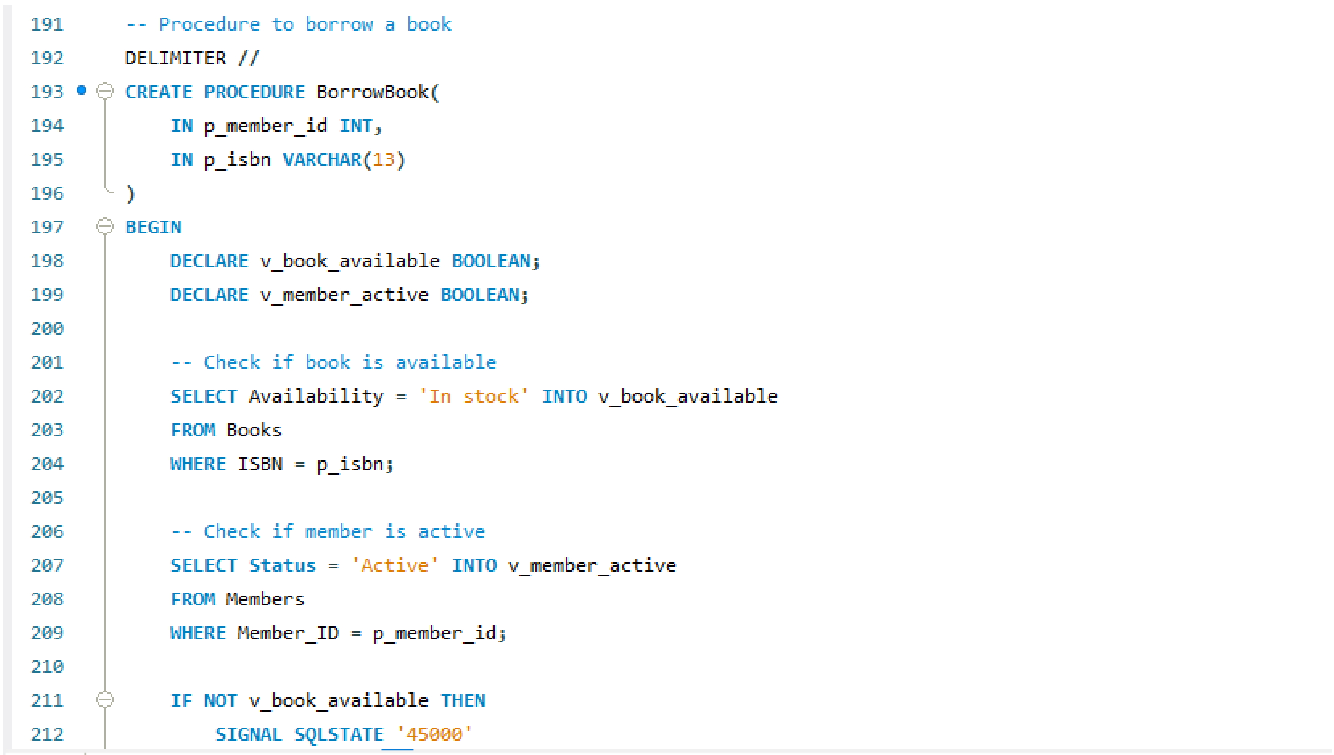
```sql
-- Create procedure to check out a book (Admin)
DELIMITER //
CREATE PROCEDURE CheckOutBook(
    IN p_admin_id INT,
    IN p_isbn VARCHAR(13)
)
BEGIN
    DECLARE v_book_available BOOLEAN;

    -- Check if book is available
    SELECT Availability = 'In stock' INTO v_book_available
    FROM Books
    WHERE ISBN = p_isbn;

    IF NOT v_book_available THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Book is not available for checkout';
    END IF;

    START TRANSACTION;

    -- Update book status
    UPDATE Books
    SET Availability = 'Checked out'
```



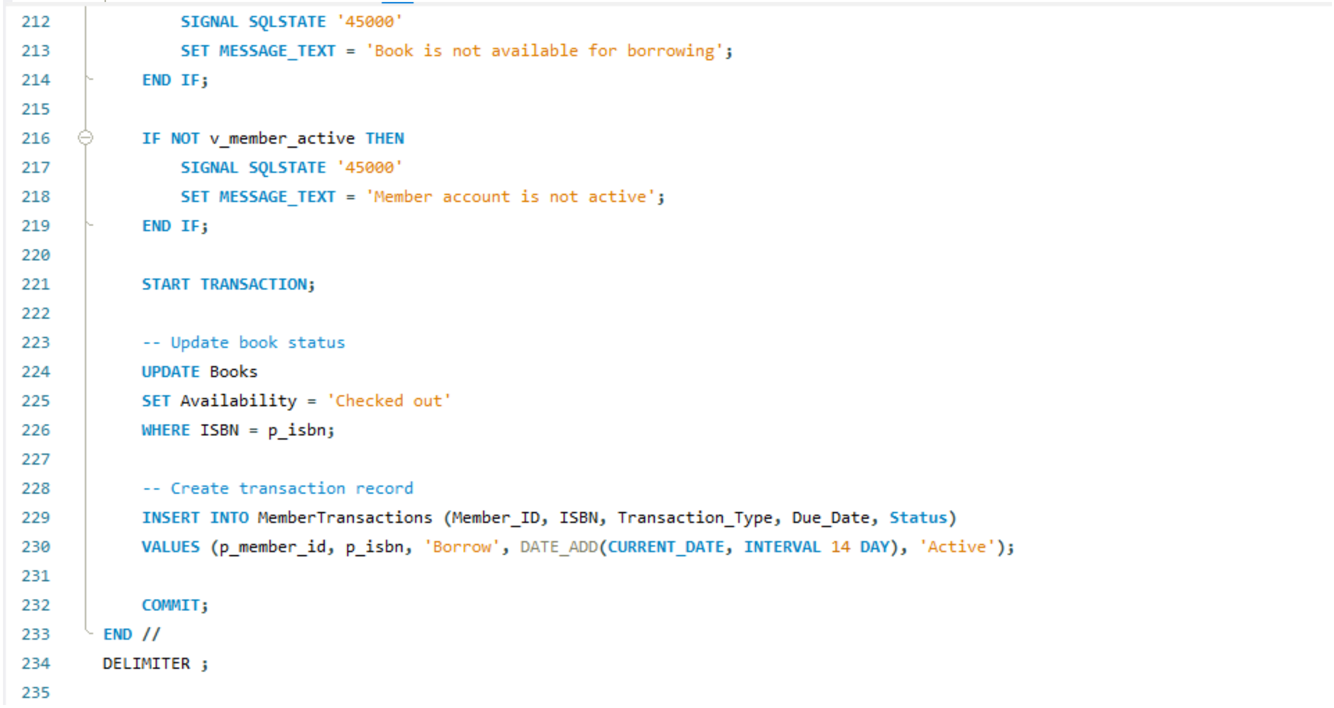
```sql
    SET Availability = 'Checked out'
    WHERE ISBN = p_isbn;

    -- Record admin transaction
    INSERT INTO AdminTransactions (ISBN, Admin_ID, Transaction_Type)
    VALUES (p_isbn, p_admin_id, 'Check out');

    COMMIT;
END //
DELIMITER ;
```

- Checks if the requested book is available for checkout.
- Updates the book's availability status to "Checked out".
- Records the checkout transaction in the AdminTransactions table

Borrow Book



```sql
-- Procedure to borrow a book
DELIMITER //
CREATE PROCEDURE BorrowBook(
    IN p_member_id INT,
    IN p_isbn VARCHAR(13)
)
BEGIN
    DECLARE v_book_available BOOLEAN;
    DECLARE v_member_active BOOLEAN;

    -- Check if book is available
    SELECT Availability = 'In stock' INTO v_book_available
    FROM Books
    WHERE ISBN = p_isbn;

    -- Check if member is active
    SELECT Status = 'Active' INTO v_member_active
    FROM Members
    WHERE Member_ID = p_member_id;

    IF NOT v_book_available THEN
        SIGNAL SQLSTATE '45000'
```

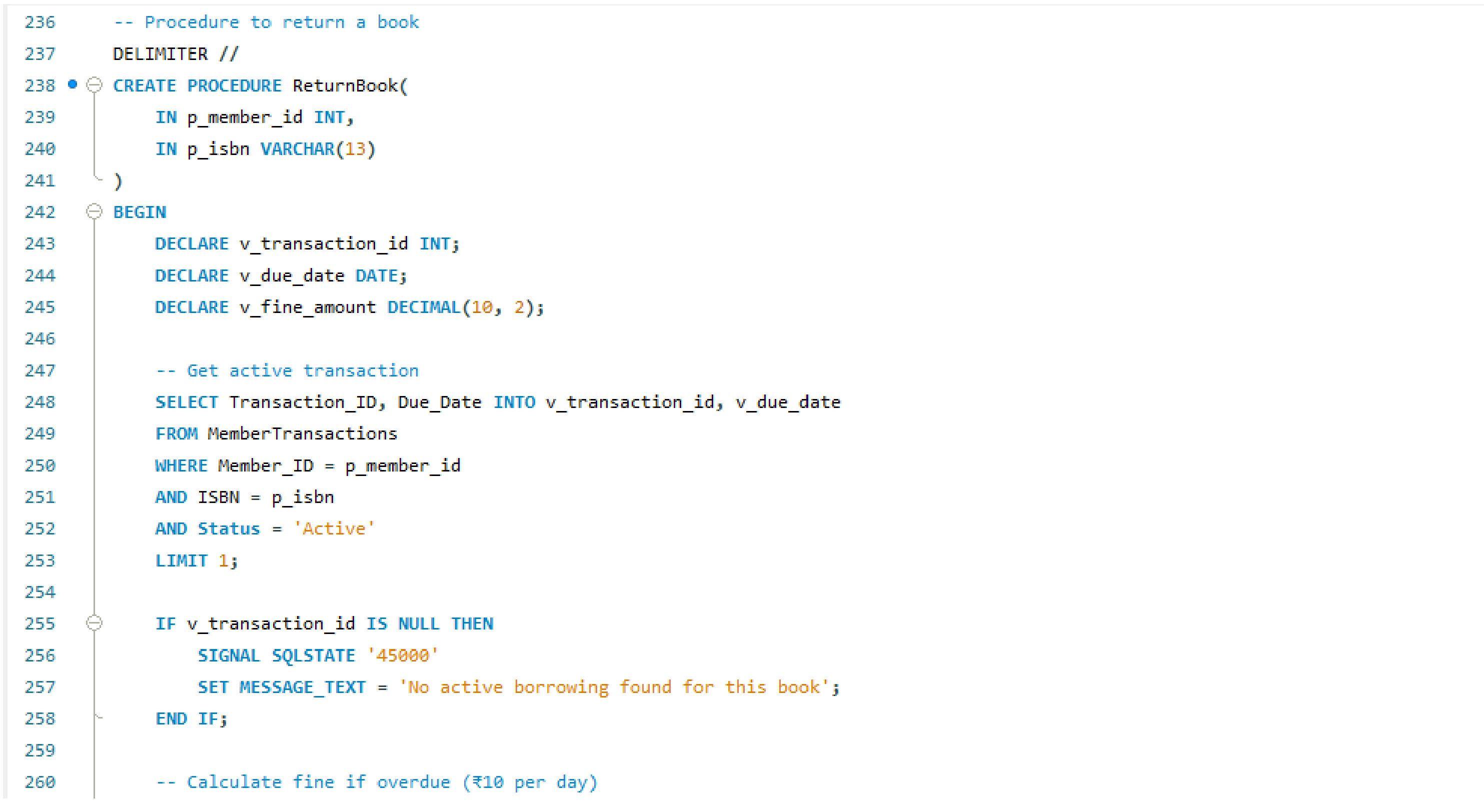


SIGNAL SQLSTATE '45000'

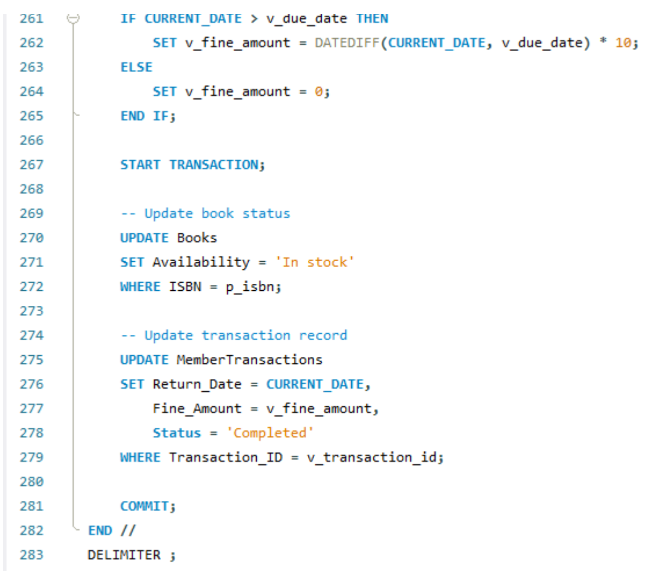
```sql
        SET MESSAGE_TEXT = 'Book is not available for borrowing';
    END IF;

    IF NOT v_member_active THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Member account is not active';
    END IF;

    START TRANSACTION;

    -- Update book status
    UPDATE Books
    SET Availability = 'Checked out'
    WHERE ISBN = p_isbn;

    -- Create transaction record
    INSERT INTO MemberTransactions (Member_ID, ISBN, Transaction_Type, Due_Date, Status)
    VALUES (p_member_id, p_isbn, 'Borrow', DATE_ADD(CURRENT_DATE, INTERVAL 14 DAY), 'Active');

    COMMIT;
END //
DELIMITER ;
```

- Checks if the requested book is available and if the member's account is active.
- Updates the book's availability status to "Checked out".
- Creates a new transaction record in the MemberTransactions table with a due date 14 days from the current date.

Return Book



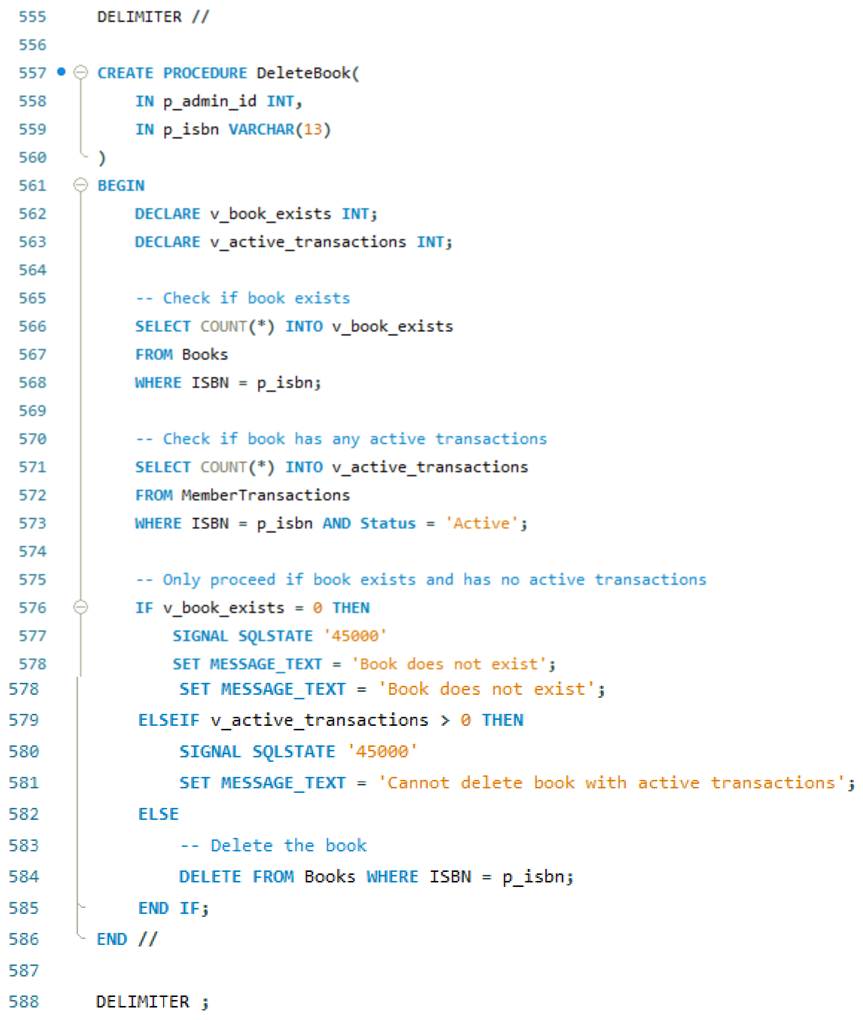
```sql
-- Procedure to return a book
DELIMITER //
CREATE PROCEDURE ReturnBook(
    IN p_member_id INT,
    IN p_isbn VARCHAR(13)
)
BEGIN
    DECLARE v_transaction_id INT;
    DECLARE v_due_date DATE;
    DECLARE v_fine_amount DECIMAL(10, 2);

    -- Get active transaction
    SELECT Transaction_ID, Due_Date INTO v_transaction_id, v_due_date
    FROM MemberTransactions
    WHERE Member_ID = p_member_id
    AND ISBN = p_isbn
    AND Status = 'Active'
    LIMIT 1;

    IF v_transaction_id IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No active borrowing found for this book';
    END IF;

    -- Calculate fine if overdue (₹10 per day)
```



```sql
    IF CURRENT_DATE > v_due_date THEN
        SET v_fine_amount = DATEDIFF(CURRENT_DATE, v_due_date) * 10;
    ELSE
        SET v_fine_amount = 0;
    END IF;

    START TRANSACTION;

    -- Update book status
    UPDATE Books
    SET Availability = 'In stock'
    WHERE ISBN = p_isbn;

    -- Update transaction record
    UPDATE MemberTransactions
    SET Return_Date = CURRENT_DATE,
        Fine_Amount = v_fine_amount,
        Status = 'Completed'
    WHERE Transaction_ID = v_transaction_id;

    COMMIT;
END //
DELIMITER ;
```

- Retrieves the active transaction for the given member and book.
- Calculates the fine amount based on the number of days the book is overdue.
- Updates the book's availability status to "In stock".
- Updates the transaction record in the MemberTransactions table with the return date and fine amount

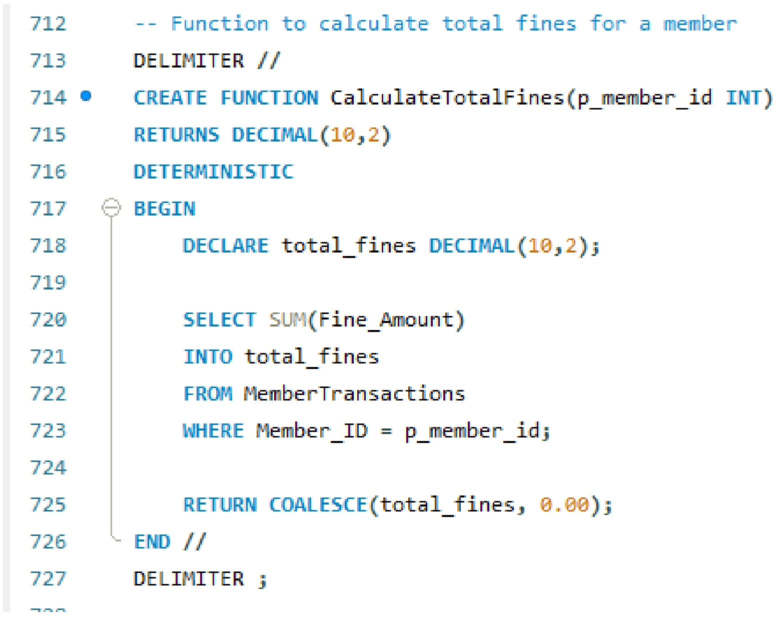
Delete Book



```sql
DELIMITER //

CREATE PROCEDURE DeleteBook(
    IN p_admin_id INT,
    IN p_isbn VARCHAR(13)
)
BEGIN
    DECLARE v_book_exists INT;
    DECLARE v_active_transactions INT;

    -- Check if book exists
    SELECT COUNT(*) INTO v_book_exists
    FROM Books
    WHERE ISBN = p_isbn;

    -- Check if book has any active transactions
    SELECT COUNT(*) INTO v_active_transactions
    FROM MemberTransactions
    WHERE ISBN = p_isbn AND Status = 'Active';

    -- Only proceed if book exists and has no active transactions
    IF v_book_exists = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Book does not exist';
        SET MESSAGE_TEXT = 'Book does not exist';
    ELSEIF v_active_transactions > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot delete book with active transactions';
    ELSE
        -- Delete the book
        DELETE FROM Books WHERE ISBN = p_isbn;
    END IF;
END //

DELIMITER ;
```

Functions

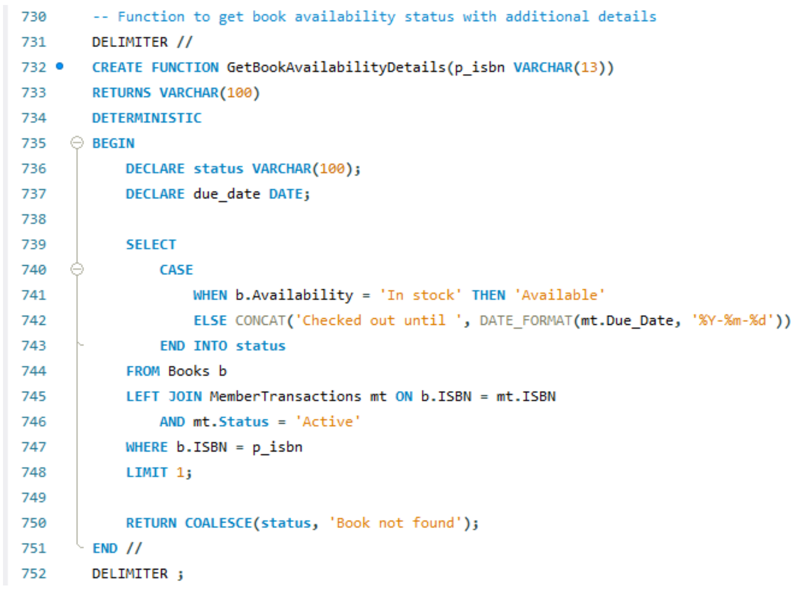
Calculate Total Fines



```sql
-- Function to calculate total fines for a member
DELIMITER //
CREATE FUNCTION CalculateTotalFines(p_member_id INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE total_fines DECIMAL(10,2);

    SELECT SUM(Fine_Amount)
    INTO total_fines
    FROM MemberTransactions
    WHERE Member_ID = p_member_id;

    RETURN COALESCE(total_fines, 0.00);
END //
DELIMITER ;
```

- Calculates the total fines owed by a given member by summing the Fine_Amount column in the MemberTransactions table.

GetBookAvailabilityDetails:

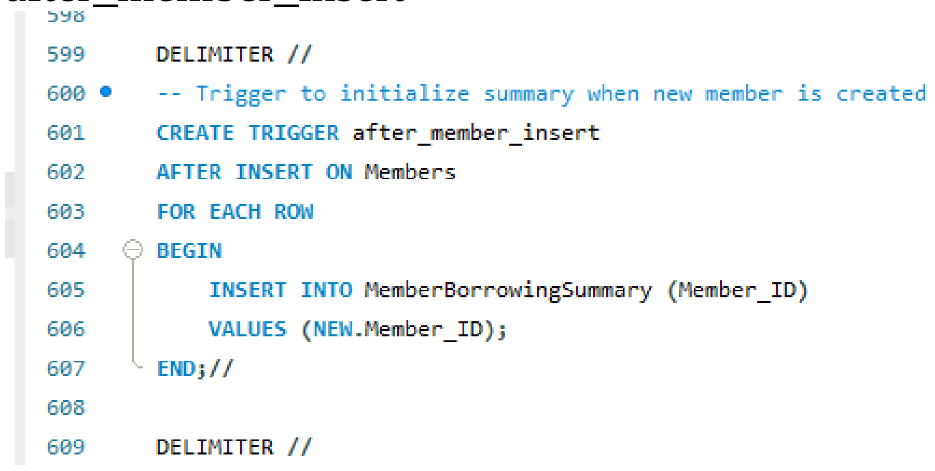


```sql
-- Function to get book availability status with additional details
DELIMITER //
CREATE FUNCTION GetBookAvailabilityDetails(p_isbn VARCHAR(13))
RETURNS VARCHAR(100)
DETERMINISTIC
BEGIN
    DECLARE status VARCHAR(100);
    DECLARE due_date DATE;

    SELECT
        CASE
            WHEN b.Availability = 'In stock' THEN 'Available'
            ELSE CONCAT('Checked out until ', DATE_FORMAT(mt.Due_Date, '%Y-%m-%d'))
        END INTO status
    FROM Books b
    LEFT JOIN MemberTransactions mt ON b.ISBN = mt.ISBN
        AND mt.Status = 'Active'
    WHERE b.ISBN = p_isbn
    LIMIT 1;

    RETURN COALESCE(status, 'Book not found');
END //
DELIMITER ;
```

- Retrieves the availability status of a given book, including the due date if the book is currently checked out.

Triggers:

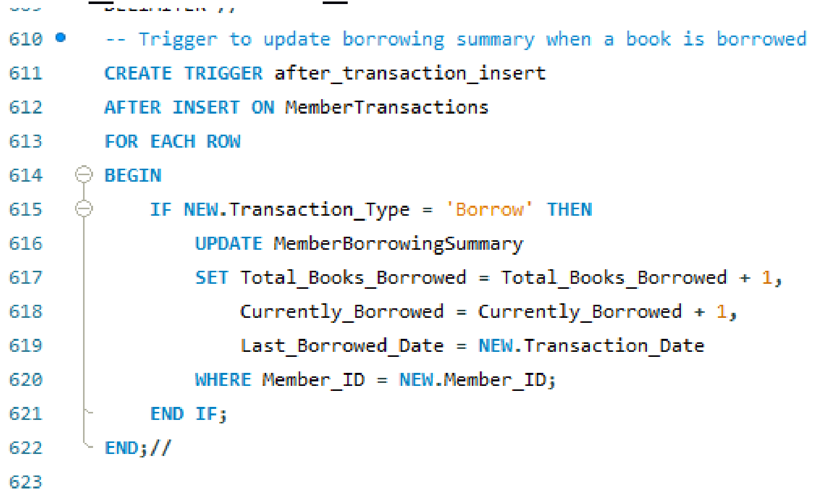
after_member_insert



```
DELIMITER //
-- Trigger to initialize summary when new member is created
CREATE TRIGGER after_member_insert
AFTER INSERT ON Members
FOR EACH ROW
BEGIN
    INSERT INTO MemberBorrowingSummary (Member_ID)
    VALUES (NEW.Member_ID);
END;//

DELIMITER //
```

- Initializes a new record in the MemberBorrowingSummary table when a new member is created.

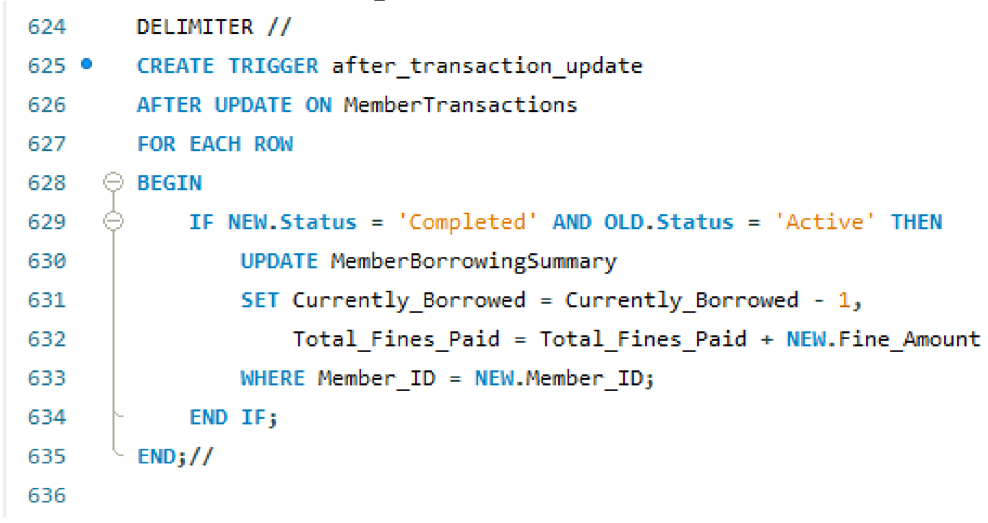
after_transaction_insert



```
-- Trigger to update borrowing summary when a book is borrowed
CREATE TRIGGER after_transaction_insert
AFTER INSERT ON MemberTransactions
FOR EACH ROW
BEGIN
    IF NEW.Transaction_Type = 'Borrow' THEN
        UPDATE MemberBorrowingSummary
        SET Total_Books_Borrowed = Total_Books_Borrowed + 1,
            Currently_Borrowed = Currently_Borrowed + 1,
            Last_Borrowed_Date = NEW.Transaction_Date
        WHERE Member_ID = NEW.Member_ID;
    END IF;
END;//
```

Updates the MemberBorrowingSummary table when a book is borrowed, incrementing the Total_Books_Borrowed and Currently_Borrowed columns.

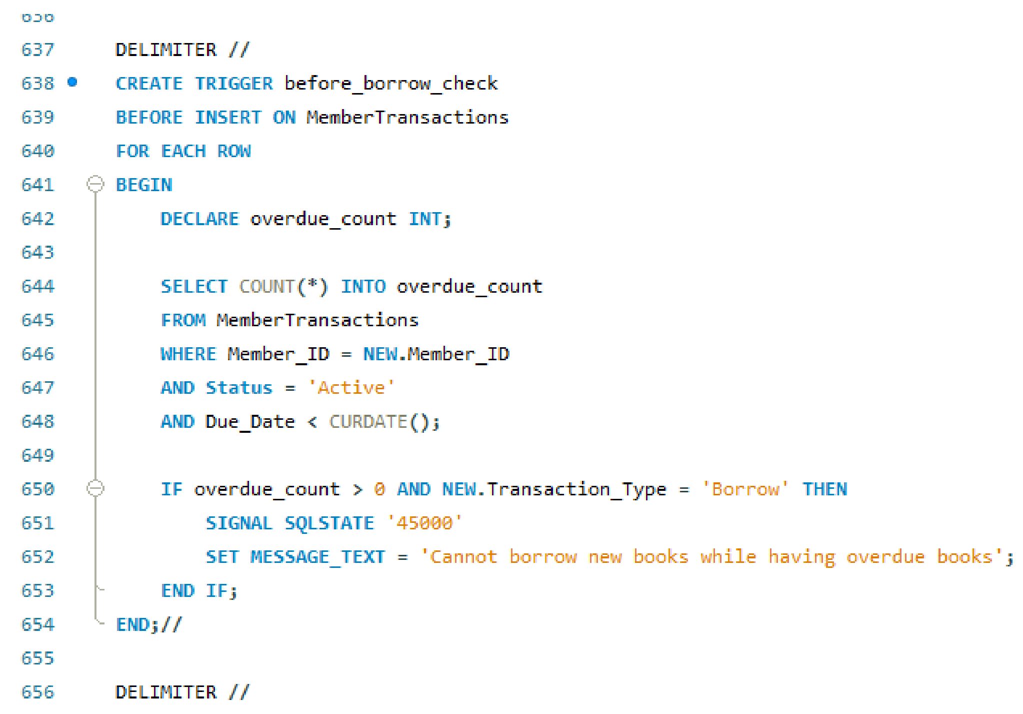
after_transaction_update



```
DELIMITER //
CREATE TRIGGER after_transaction_update
AFTER UPDATE ON MemberTransactions
FOR EACH ROW
BEGIN
    IF NEW.Status = 'Completed' AND OLD.Status = 'Active' THEN
        UPDATE MemberBorrowingSummary
        SET Currently_Borrowed = Currently_Borrowed - 1,
            Total_Fines_Paid = Total_Fines_Paid + NEW.Fine_Amount
        WHERE Member_ID = NEW.Member_ID;
    END IF;
END;//

```

- Updates the MemberBorrowingSummary table when a book is returned, decrementing the Currently_Borrowed column and adding the fine amount to the Total_Fines_Paid column.

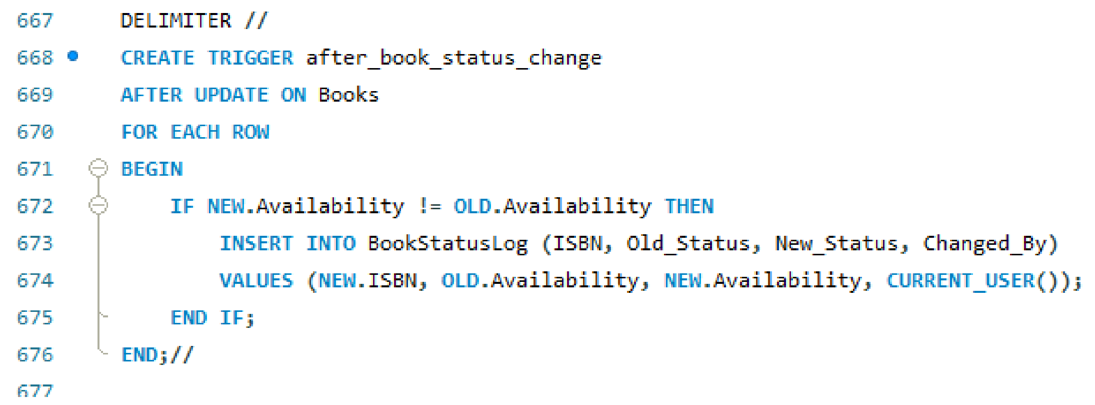
before_borrow_check



```

DELIMITER //
CREATE TRIGGER before_borrow_check
BEFORE INSERT ON MemberTransactions
FOR EACH ROW
BEGIN
    DECLARE overdue_count INT;

    SELECT COUNT(*) INTO overdue_count
    FROM MemberTransactions
    WHERE Member_ID = NEW.Member_ID
    AND Status = 'Active'
    AND Due_Date < CURDATE();

    IF overdue_count > 0 AND NEW.Transaction_Type = 'Borrow' THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot borrow new books while having overdue books';
    END IF;
END;//

DELIMITER //
```

- Checks if the member has any overdue books before allowing a new borrow transaction.
- Throws an error if the member has overdue books

after_book_status_change



```sql
DELIMITER //
CREATE TRIGGER after_book_status_change
AFTER UPDATE ON Books
FOR EACH ROW
BEGIN
    IF NEW.Availability != OLD.Availability THEN
        INSERT INTO BookStatusLog (ISBN, Old_Status, New_Status, Changed_By)
        VALUES (NEW.ISBN, OLD.Availability, NEW.Availability, CURRENT_USER());
    END IF;
END;//
```

- Logs changes in the book's availability status in the BookStatusLog table.

9) FRONT END DEVELOPMENT (FUNCTIONALITIES/FEATURES OF THE APPLICATION)

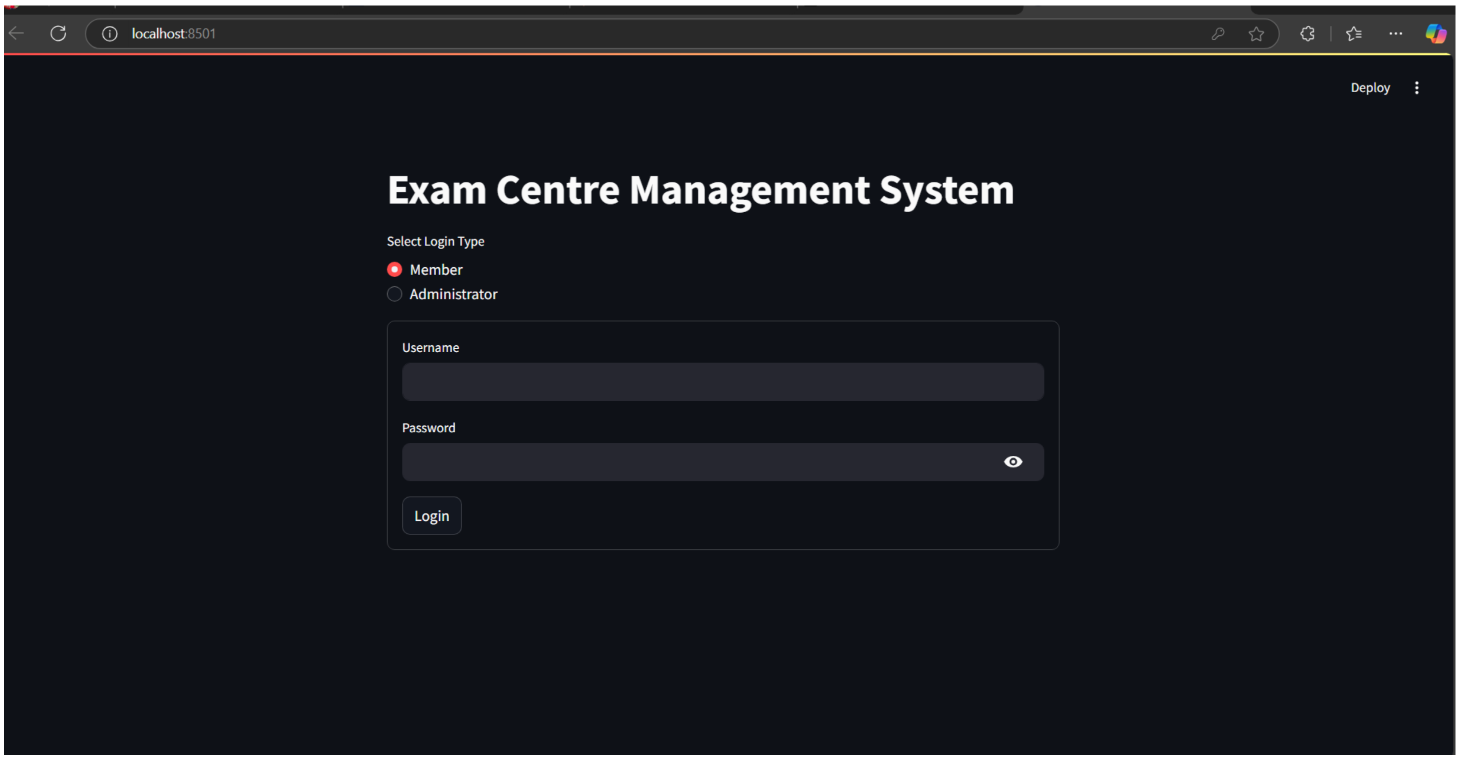
The frontend has been developed using Streamlit, a Python framework for creating web applications.

1. Main Structure:

- The application uses a single-page architecture managed by main() function
- It maintains user state using Streamlit's session state (st.session_state)
- Two main portals: Admin Portal and Member Portal
- Login page as the entry point.

2. Login Page (login_page()):

- Clean title "Exam Centre Management System"
- Radio button to select user type (Member/Administrator)
- Simple form with:
    - Username input
    - Password input (masked)
    - Login button
- Success/Error messages for login attempts



Exam Centre Management System

Select Login Type

[✓ Member] Administrator

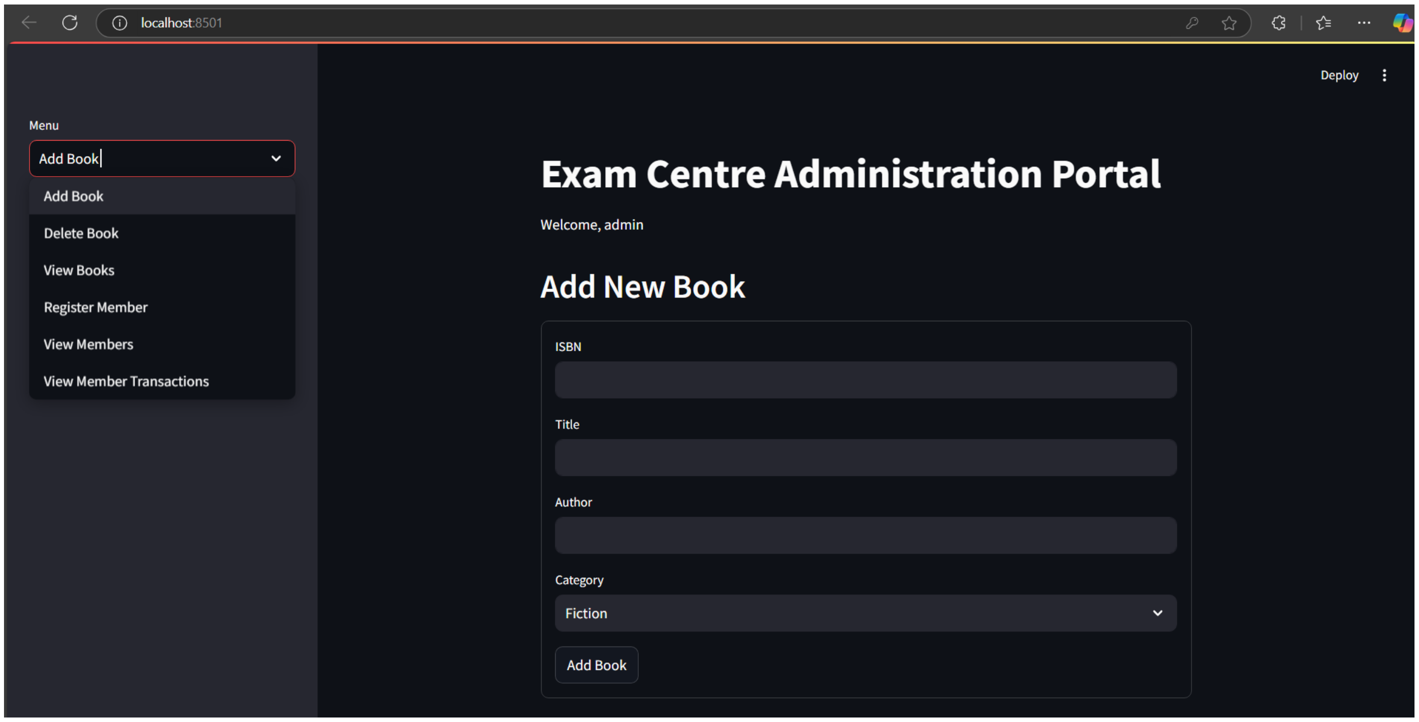
Username

Password

Login

3 <u>Admin Portal (admin_portal()):</u>

- Sidebar menu with options:
    - Add Book
    - Delete Book
    - View Books
    - Register Member
    - View Members
    - View Member Transactions
- Each menu option has its own section with relevant forms and tables
- Logout button in sidebar



Exam Centre Administration Portal

Welcome, admin

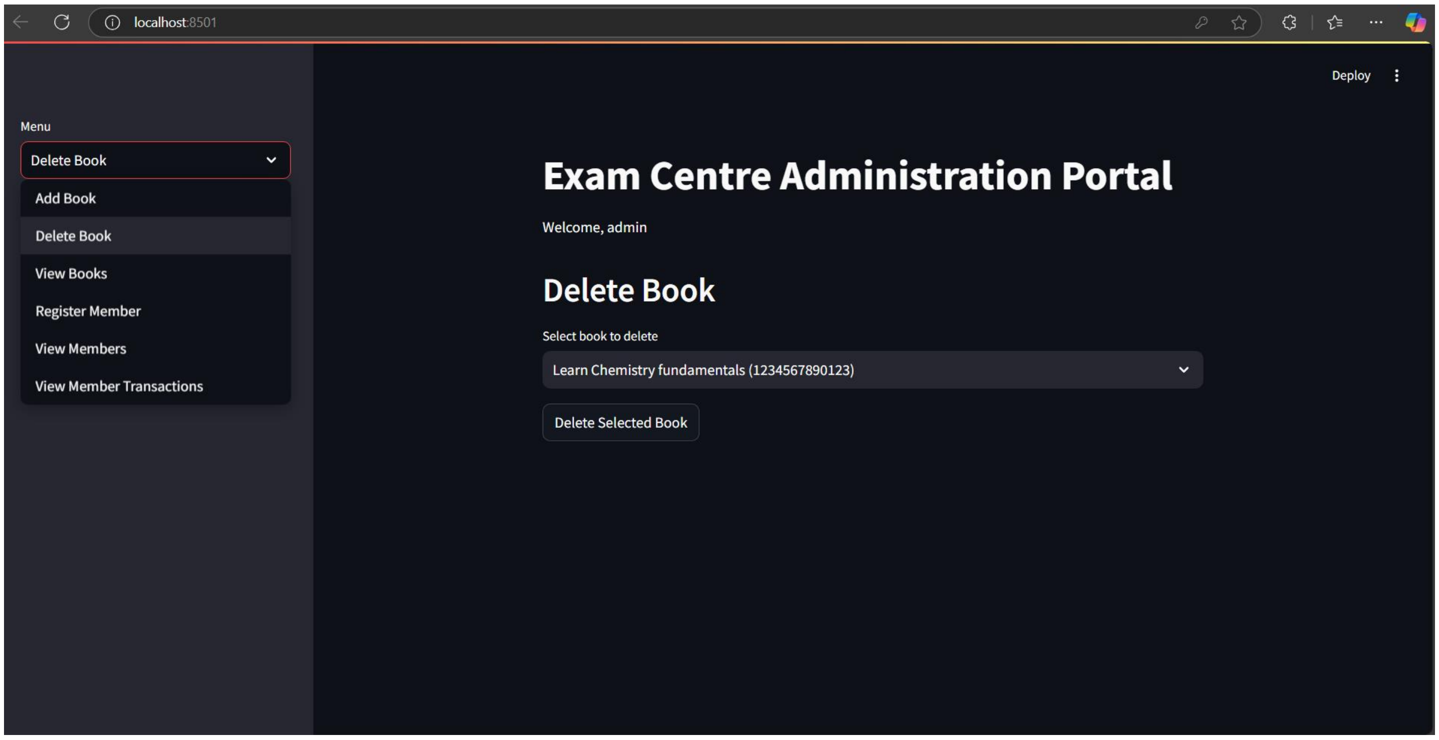
Add New Book

ISBN

Title

Author

Category: Fiction

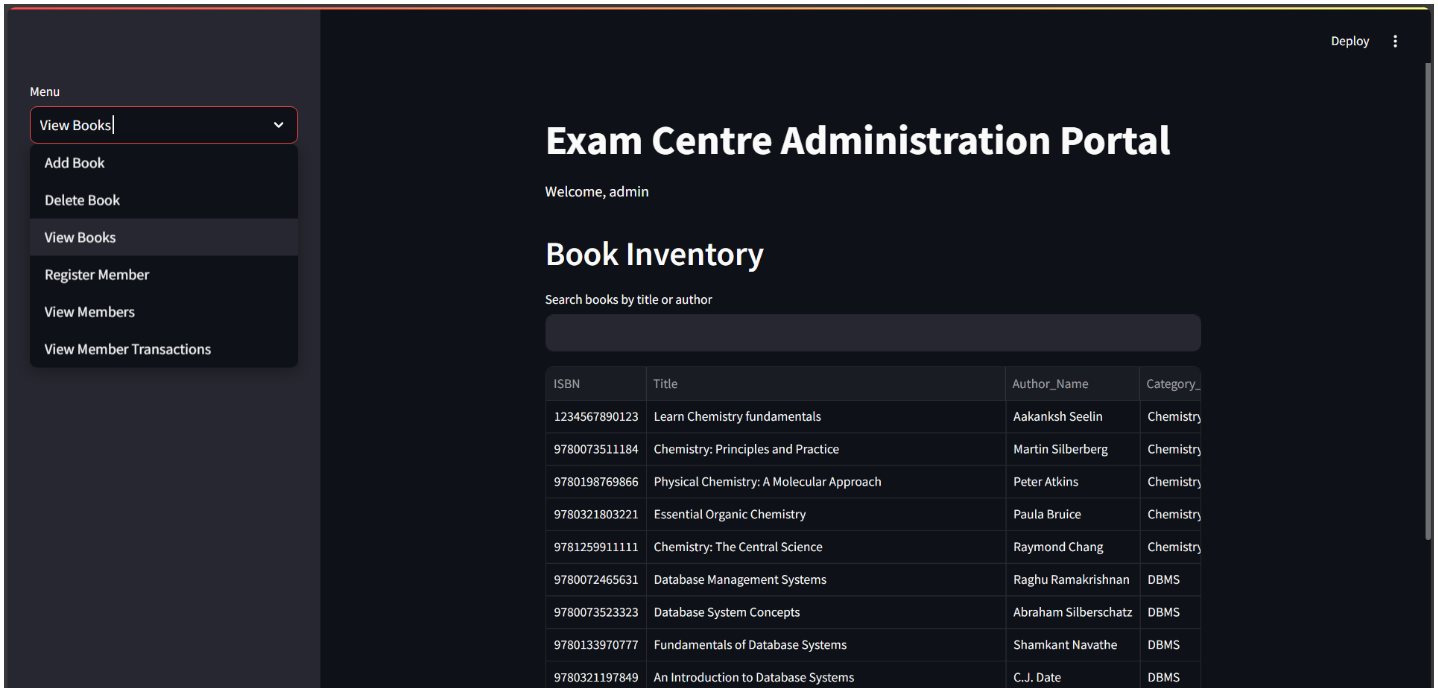


Exam Centre Administration Portal

Welcome, admin

Delete Book

Select book to delete: Learn Chemistry fundamentals (1234567890123)



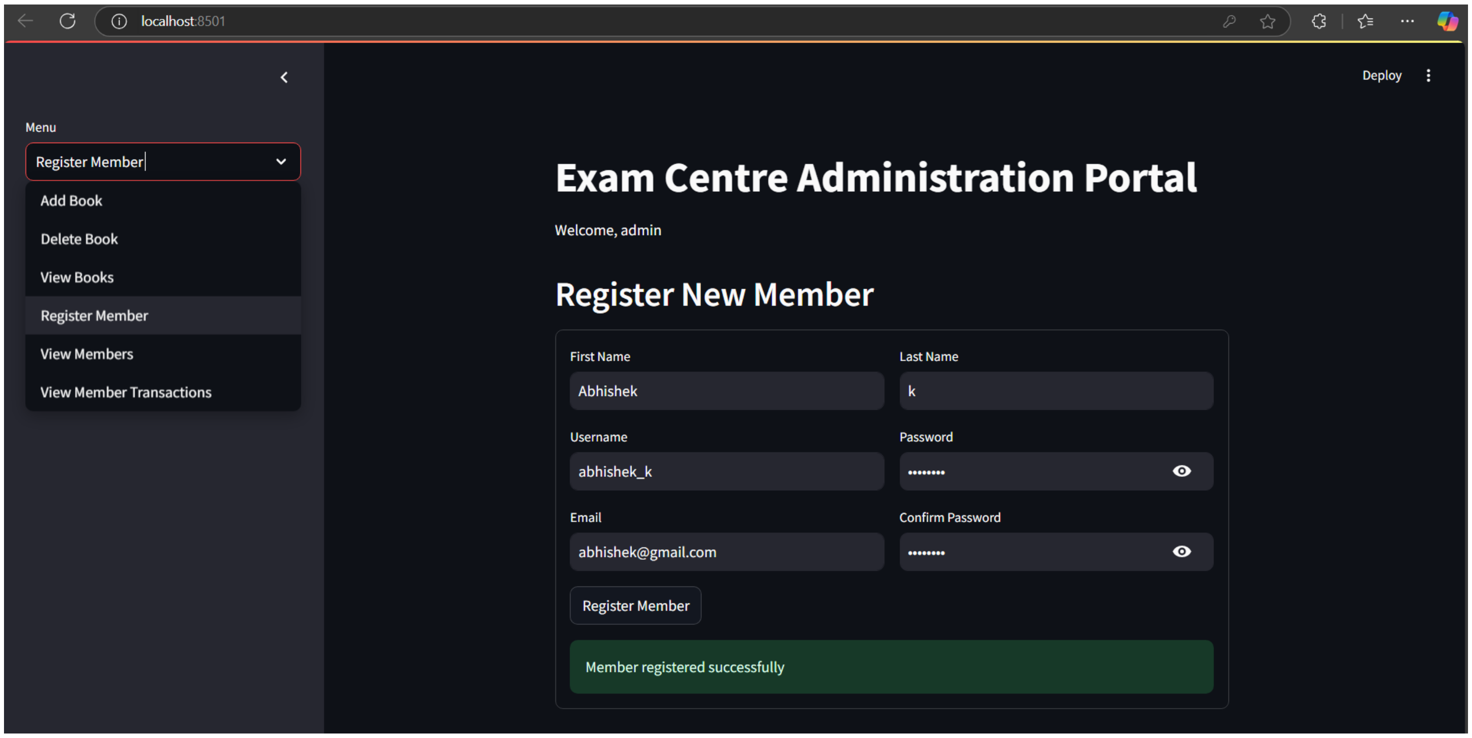
Exam Centre Administration Portal

Welcome, admin

Book Inventory

Search books by title or author

ISBN	Title	Author_Name	Category_
1234567890123	Learn Chemistry fundamentals	Aakanksh Seelin	Chemistry
9780073511184	Chemistry: Principles and Practice	Martin Silberberg	Chemistry
9780198769866	Physical Chemistry: A Molecular Approach	Peter Atkins	Chemistry
9780321803221	Essential Organic Chemistry	Paula Bruice	Chemistry
9781259911111	Chemistry: The Central Science	Raymond Chang	Chemistry
9780072465631	Database Management Systems	Raghu Ramakrishnan	DBMS
9780073523323	Database System Concepts	Abraham Silberschatz	DBMS
9780133970777	Fundamentals of Database Systems	Shamkant Navathe	DBMS
9780321197849	An Introduction to Database Systems	C.J. Date	DBMS



Exam Centre Administration Portal

Welcome, admin

Register New Member

First Name: Abhishek

Last Name: k

Username: abhishek_k

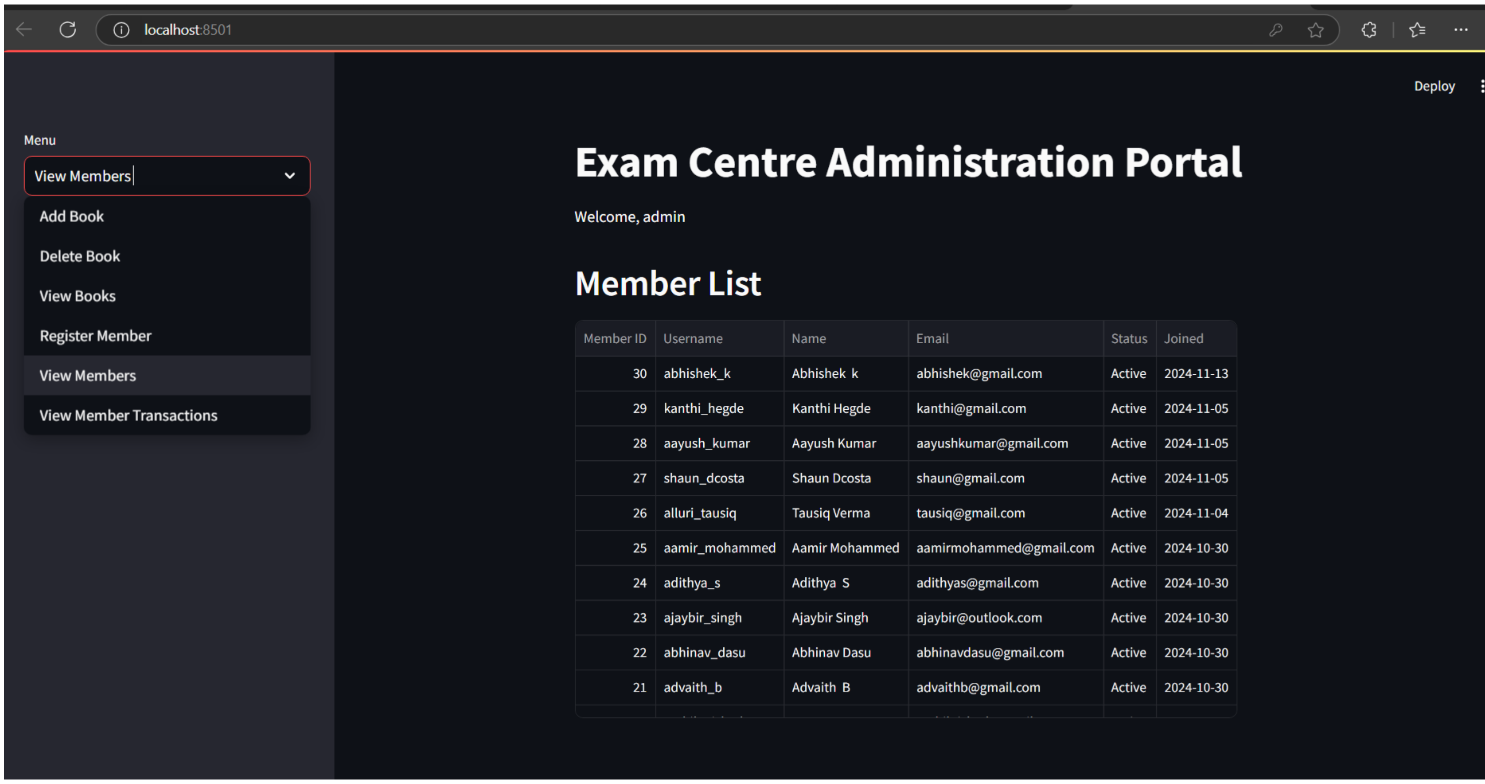
Password: ••••••••

Email: abhishek@gmail.com

Confirm Password: ••••••••

Register Member

Member registered successfully



Deploy

Menu

View Members

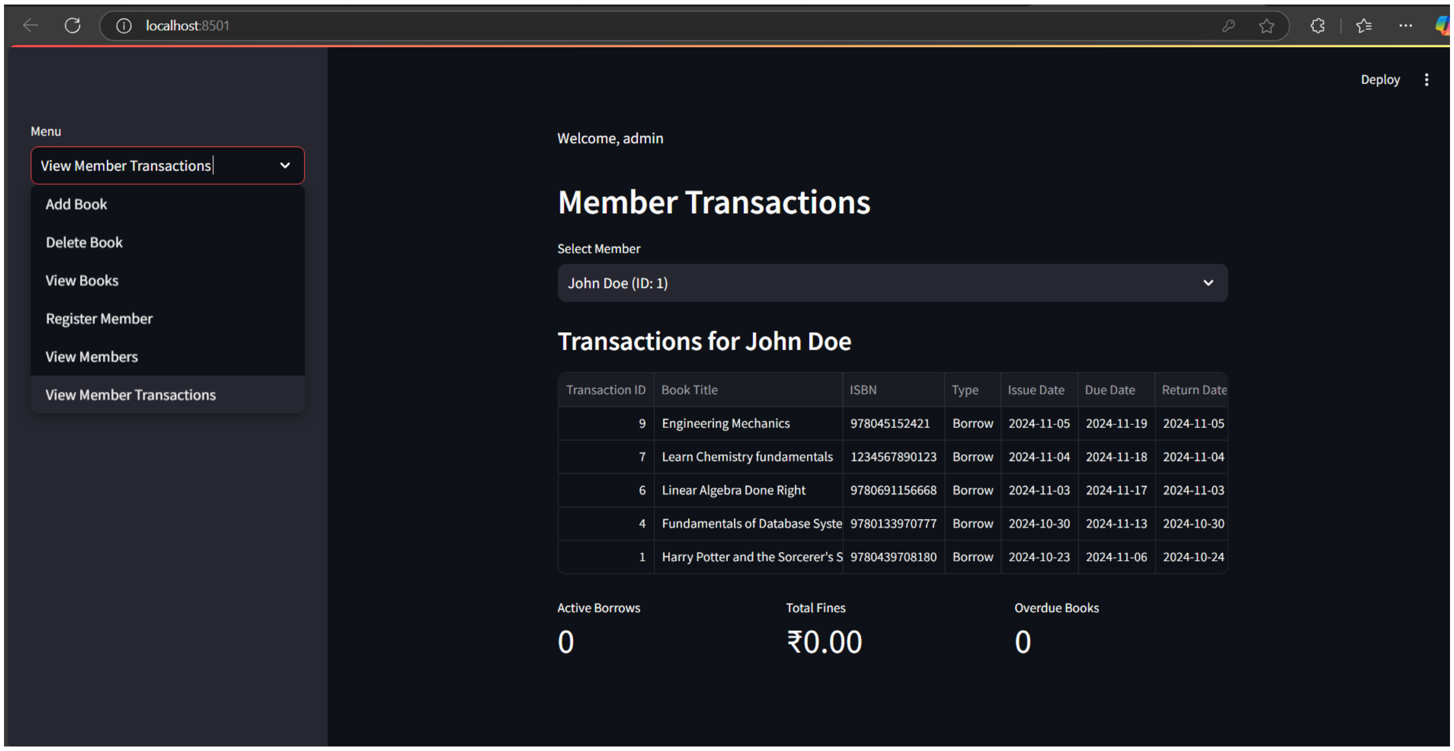
- Add Book
- Delete Book
- View Books
- Register Member
- View Members
- View Member Transactions

Exam Centre Administration Portal

Welcome, admin

Member List

Member ID	Username	Name	Email	Status	Joined
30	abhishek_k	Abhishek k	abhishek@gmail.com	Active	2024-11-13
29	kanthi_hegde	Kanthi Hegde	kanthi@gmail.com	Active	2024-11-05
28	aayush_kumar	Aayush Kumar	aayushkumar@gmail.com	Active	2024-11-05
27	shaun_dcosta	Shaun Dcosta	shaun@gmail.com	Active	2024-11-05
26	alluri_tausiq	Tausiq Verma	tausiq@gmail.com	Active	2024-11-04
25	aamir_mohammed	Aamir Mohammed	aamirmohammed@gmail.com	Active	2024-10-30
24	adithya_s	Adithya S	adithyas@gmail.com	Active	2024-10-30
23	ajaybir_singh	Ajaybir Singh	ajaybir@outlook.com	Active	2024-10-30
22	abhinav_dasu	Abhinav Dasu	abhinavdasu@gmail.com	Active	2024-10-30
21	advaith_b	Advaith B	advaithb@gmail.com	Active	2024-10-30



Deploy

Menu

View Member Transactions

- Add Book
- Delete Book
- View Books
- Register Member
- View Members
- View Member Transactions

Welcome, admin

Member Transactions

Select Member

John Doe (ID: 1)

Transactions for John Doe

Transaction ID	Book Title	ISBN	Type	Issue Date	Due Date	Return Date
9	Engineering Mechanics	978045152421	Borrow	2024-11-05	2024-11-19	2024-11-05
7	Learn Chemistry fundamentals	1234567890123	Borrow	2024-11-04	2024-11-18	2024-11-04
6	Linear Algebra Done Right	9780691156668	Borrow	2024-11-03	2024-11-17	2024-11-03
4	Fundamentals of Database Syste	9780133970777	Borrow	2024-10-30	2024-11-13	2024-10-30
1	Harry Potter and the Sorcerer's S	9780439708180	Borrow	2024-10-23	2024-11-06	2024-10-24

Active Borrows

0

Total Fines

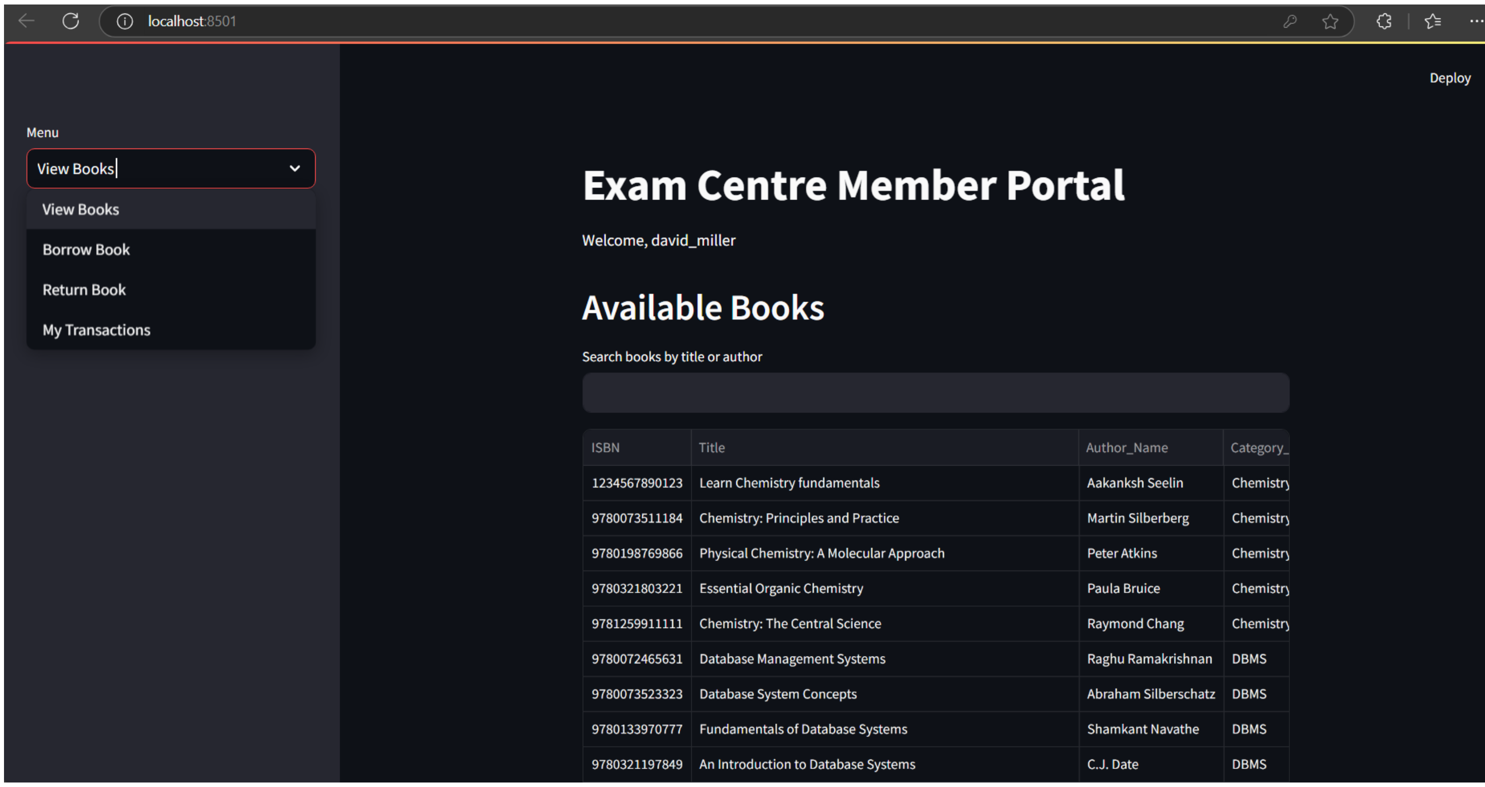
₹0.00

Overdue Books

0

4. Member Portal (member_portal()):

- Sidebar menu with options:
    - View Books
    - Borrow Book
    - Return Book
    - My Transactions
- Each section has appropriate forms and data displays
- Logout button in sidebar



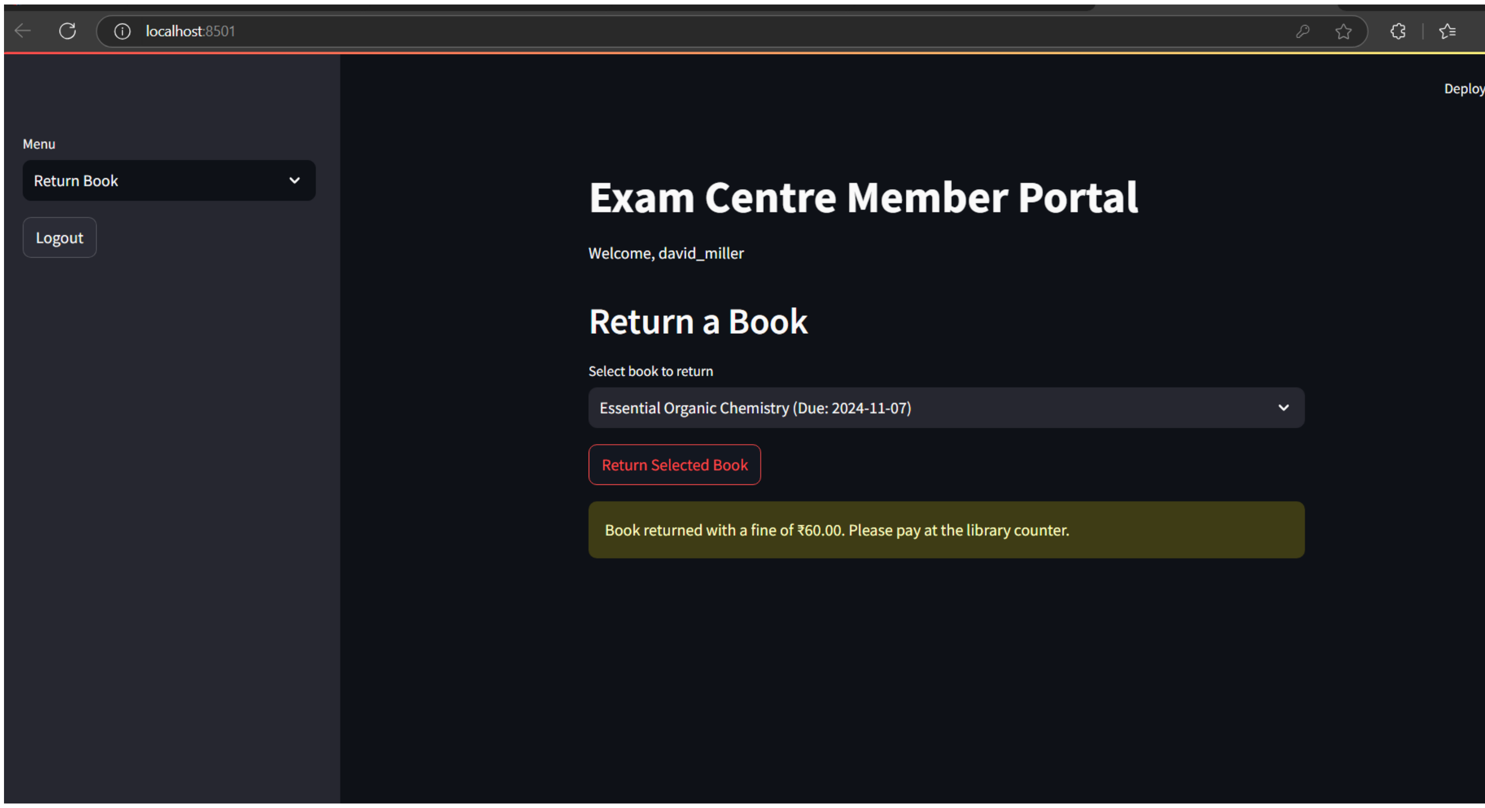
Exam Centre Member Portal

Welcome, david_miller

Available Books

Search books by title or author

ISBN	Title	Author_Name	Category_
1234567890123	Learn Chemistry fundamentals	Aakanksh Seelin	Chemistry
9780073511184	Chemistry: Principles and Practice	Martin Silberberg	Chemistry
9780198769866	Physical Chemistry: A Molecular Approach	Peter Atkins	Chemistry
9780321803221	Essential Organic Chemistry	Paula Bruice	Chemistry
9781259911111	Chemistry: The Central Science	Raymond Chang	Chemistry
9780072465631	Database Management Systems	Raghu Ramakrishnan	DBMS
9780073523323	Database System Concepts	Abraham Silberschatz	DBMS
9780133970777	Fundamentals of Database Systems	Shamkant Navathe	DBMS
9780321197849	An Introduction to Database Systems	C.J. Date	DBMS



Exam Centre Member Portal

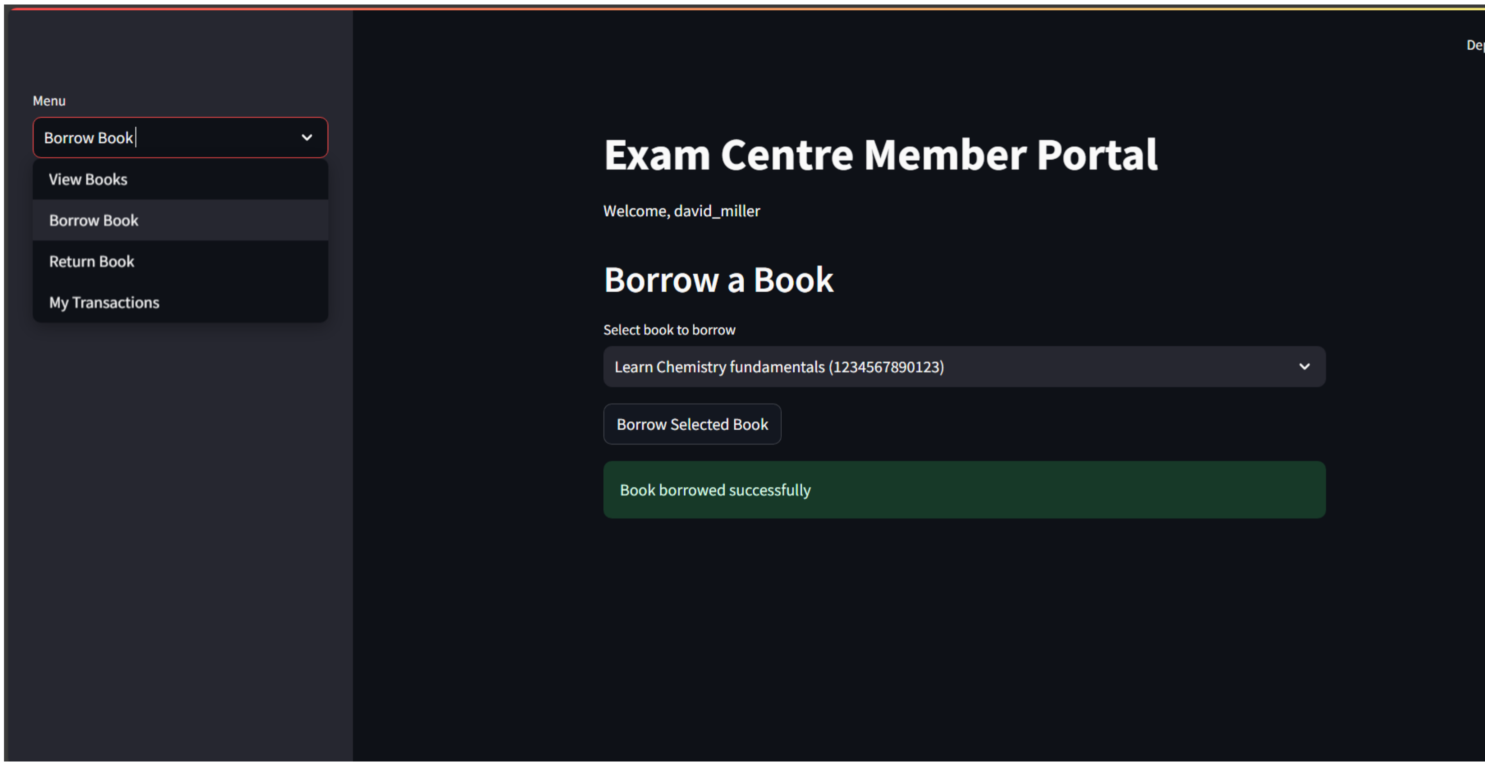
Welcome, david_miller

Return a Book

Select book to return

Essential Organic Chemistry (Due: 2024-11-07)

Book returned with a fine of ₹60.00. Please pay at the library counter.



Exam Centre Member Portal

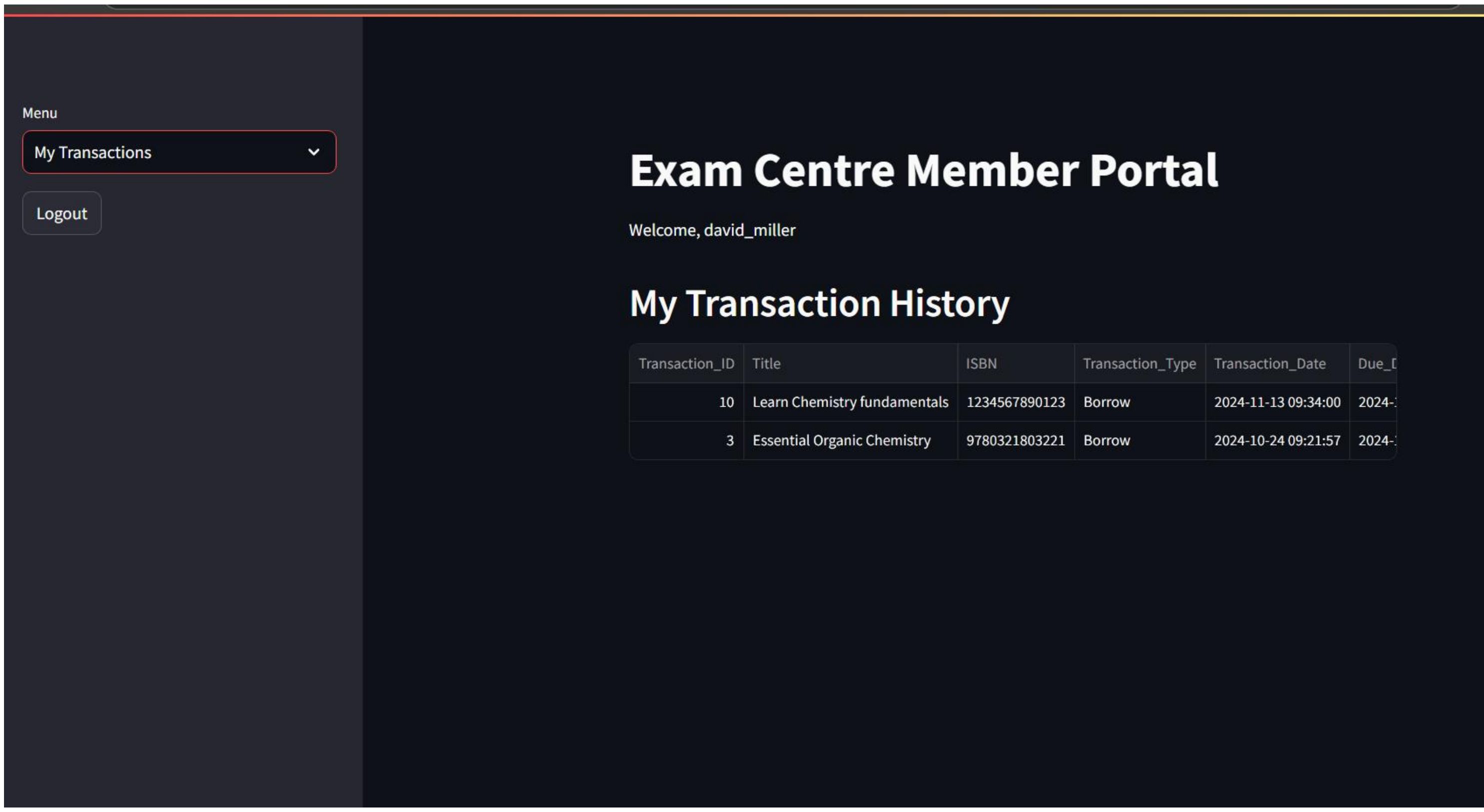
Welcome, david_miller

Borrow a Book

Select book to borrow

Learn Chemistry fundamentals (1234567890123)

Book borrowed successfully



Exam Centre Member Portal

Welcome, david_miller

My Transaction History

Transaction_ID	Title	ISBN	Transaction_Type	Transaction_Date	Due_D
10	Learn Chemistry fundamentals	1234567890123	Borrow	2024-11-13 09:34:00	2024-
3	Essential Organic Chemistry	9780321803221	Borrow	2024-10-24 09:21:57	2024-

6. <u>Key UI Features:</u>
   - Responsive layout with columns for form organization
   - Interactive data tables with search functionality
   - Clear success/error messages for user feedback
   - Metrics display for statistics
   - Dropdown menus for selections
   - Form validation and error handling

7. <u>Data Presentation:</u>
   - Books displayed in tabular format with search capability
   - Member transactions shown with formatted dates and currency
   - Status indicators for books (In stock/Borrowed)
   - Fine amounts displayed in Indian Rupees (₹)

8. Navigation:
   - Clear hierarchical menu structure
   - Sidebar for main navigation
   - Logical grouping of related functions
   - Easy logout access

This frontend design follows several good practices:

- Clean and intuitive interface
- Consistent layout across different sections
- Clear feedback for user actions
- Proper form validation
- Organized menu structure
- Responsive design elements
- Clear data presentation

REFERENCES/BIBLIOGRAPHY

- Database Management System principles and design
- Software Engineering concepts and practices.
- SQL commands and syntax.
- Github

SQL Syntax

Database Design in DBMS - GeeksforGeeks

Software Engineering Tutorial - GeeksforGeeks

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS

- SQL: Structured Query Language
- CRUD: Create, Read, Update, Delete
- EMS: Exam Centre Management System
- UI: User Interface
- Admin: Administrator of the EMS
- Member: A registered user of the library who can borrow books
- ISBN: International Standard Book Number