**SE 2141 – Laboratory 4**

**Online Library Management System**

**Name**: Kimly John Vergara

**Course & Year**: BSSE – 2

**Date:** December 11, 2024

**Part 1:** Draw anEntity-Relationship (ER) Diagramfor the system based on the given

requirements.

A screenshot of a computer

Description automatically generated

*Diagram 1: Online Library Management System*

The Entity-Relationship (ER) Diagram illustrates the structure of the library management system, highlighting its entities, attributes, primary keys, and relationships. Key entities include Books (with attributes like ISBN, Title, and Quantity Available), Users (with attributes like ID, Full Name, and Email Address), and Book Loans (with attributes like Loan ID, Loan Date, and Status). The diagram shows a one-to-many relationship between Users and Book Loans (a user can have multiple loans) and between Books and Book Loans (a book can be borrowed multiple times). These relationships and attributes form the foundation for implementing the database.

**Part 2:** Translate the ER diagram into relational tables.

**A computer code with text

Description automatically generated with medium confidence**

*Diagram 2.1: Create Books Table*

The Books table stores details about each book in the library, including Book\_ID (Primary Key), Title, Author, ISBN, Genre, Published\_Year, and Quantity\_Available. The Book\_ID is unique, ensuring each book can be identified distinctly. The other fields are mandatory, with Quantity\_Available tracking the number of available copies for borrowing. Constraints ensure that each book has a unique identifier and all relevant information is provided.

A close-up of a computer code

Description automatically generated

*Diagram 2.2: Create Users Table*

The Users table contains information about library members, including User\_ID (Primary Key), Full\_Name, Email\_Address, and Membership\_Date. The User\_ID uniquely identifies each user, while Email\_Address is unique and required for communication purposes. The Membership\_Date helps track when the user joined the library. Constraints enforce uniqueness for User\_ID and Email\_Address, ensuring no duplicates.

A computer screen shot of text

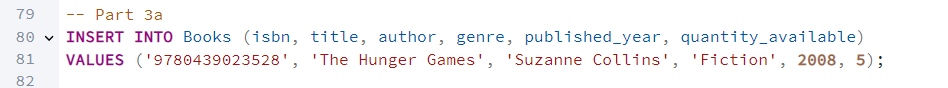
Description automatically generated

*Diagram 2.3:* *Create Book Loans Table*

The Book\_Loans table records the details of books borrowed by users, including Loan\_ID (Primary Key), User\_ID (Foreign Key), Book\_ISBN (Foreign Key), Loan\_Date, Return\_Date, and Status. The Loan\_ID uniquely identifies each loan transaction, while User\_ID and Book\_ISBN create relationships with the Users and Books tables. The Status tracks the loan's current state, and the Return\_Date is updated when the book is returned. Constraints ensure data integrity and accurate tracking of loans.

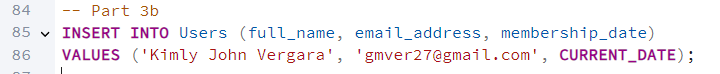
**Part 3:** Write SQL queries for the following scenarios.

* A. Insert a new book into the library with a quantity of 5.



This query inserts a new book record into the Books table with all necessary details, including the book's Title, Author, ISBN, Genre, Published\_Year, and a Quantity\_Available of 5. The ISBN ensures that the book is uniquely identified in the library's collection.

* B. Add a new user to the system.



The query adds a new user to the Users table, including the user's Full\_Name, Email\_Address, and Membership\_Date. The Email\_Address is required and must be unique, ensuring no duplicate accounts in the system.

* C. Record a book loan for a user.

A close-up of a computer code

Description automatically generated

This query records a book loan in the Book\_Loans table by associating a user with a borrowed book. It includes the User\_ID, Book\_ISBN, Loan\_Date, and an initial Status of "borrowed". The loan status is essential for tracking whether the book has been returned or is overdue.

* D. Find all books borrowed by a specific user.

A screenshot of a computer

Description automatically generated

This query retrieves all books borrowed by a specific user by joining the Book\_Loans and Books tables. It uses the User\_ID to filter the results and returns information about the books, such as the title, author, and loan status.

* E. List all overdue loans.

A computer screen shot of a code

Description automatically generated

This query lists all overdue loans by selecting records from the Book\_Loans table where the Return\_Date is past the current date and the loan Status is "borrowed". It helps track overdue books that need attention.

**Part 4:** Data Integrity and Optimization

* Explain how you would ensure the prevention of borrowing books when no copies are available.

A computer screen shot of text

Description automatically generated

This query defines a trigger function prevent\_borrowing\_unavailable\_books that ensures a user cannot borrow a book if no copies are available. Before a new book loan is recorded in the Book\_Loans table, the function checks if the quantity\_available for the specified book (isbn) is greater than zero. If not, it raises an exception, preventing the loan. If copies are available, the quantity is decreased by 1 to reflect the borrowed book. The trigger before\_borrowing is set to execute the function before each insert operation on the Book\_Loans table.

* Explain how you would ensure fast retrieval of overdue loans.

A screenshot of a computer

Description automatically generated

This query creates an index on the Return\_Date and Status columns in the Book\_Loans table using the CREATE INDEX statement. The index, named idx\_overdue\_loans, optimizes the retrieval of overdue loans by allowing faster lookups based on these two columns. The query that follows retrieves overdue loans by joining the Book\_Loans, Users, and Books tables. It filters for loans where the status is 'overdue' and orders the results by the loan\_date in ascending order. This approach ensures that overdue loans are retrieved efficiently, even with large datasets, by utilizing the created index for faster query execution.

**Part 5:** Reflection.

When scaling the database to accommodate millions of users and books, a few challenges might come up. First, there’s the issue of performance and query speed—retrieving large amounts of data, like overdue loans or book details, might slow things down. To fix this, indexing frequently searched columns like User\_ID, ISBN, Return\_Date, and Status can help speed up query execution and make data retrieval faster. Another concern is the database size and storage—storing tons of data can eventually lead to performance problems. A solution for this would be data partitioning, like splitting the Book\_Loans table by date or region, which helps spread out the data and improves performance. Concurrency and data integrity might also be an issue, especially when lots of users are borrowing books at once. Using transaction management and locking mechanisms, like optimistic or pessimistic locking, can help avoid conflicts and keep the data consistent. As the system grows, maintaining availability is key, so implementing database replication and load balancing can ensure the system stays up and running without interruptions. Lastly, backup and recovery processes need to be efficient to handle all the growing data. Automating incremental backups and using cloud-based solutions for disaster recovery can help ensure a fast recovery and reduce data loss.

**Assumptions made:**

* I assumed that the Primary Key of the Books table is the ISBN since it is unique in every book.