Library-Management-System-SQL-based Database-Design-and-Analysis

Introduction:

Libraries are important these days. With so many people visiting and lots of things happening in libraries, it's important to have a good system to keep everything organized. This system helps to store and manage all the information, making it easier for libraries to understand what people need and how they use the library. By using this system, libraries can make better decisions and improve how they help people. As libraries get busier, having a good way to manage all the stuff in them becomes super important for making sure things run smoothly.

Database Design:

Our goal is to create a database for a library that handles books across different categories. The library has various branches in different locations. Each branch has a team of students responsible for managing the library resources and assisting visitors. Every student working in the library has a unique ID, and their personal information like name, phone number, and email is recorded.

The library houses different types of books categorized by genres like fiction, non-fiction, science, etc. Each book has specific details such as its title, description, category, and the user who added it to the library collection.

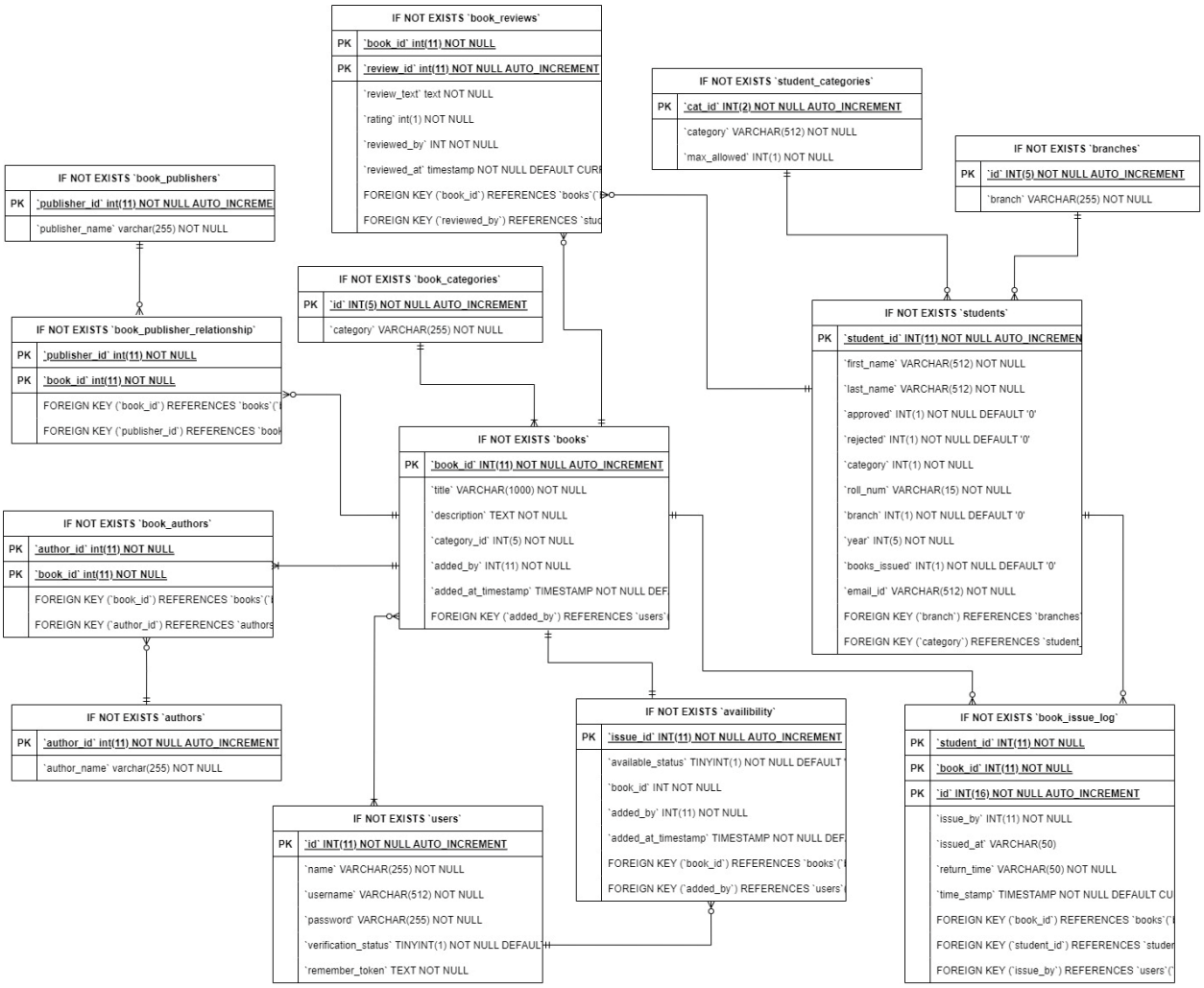
Students visiting the library can borrow books, and these transactions are recorded in a book issue log. This log keeps track of which student borrowed which book, when it was issued, and when it was returned. There's also a table that shows the availability status of each book, indicating if it's currently available for borrowing.

Furthermore, there's information on authors of the books stored in the database. Each author has an ID and a name. The relationship between books and authors is maintained in a separate table to link books with their respective authors.

Additionally, students can provide reviews for books they've read. These reviews contain the book's ID, the review text, the rating given by the student, and the student's ID who reviewed it.

Lastly, details about book publishers are also included. Each publisher has an ID and a name. The relationship between books and publishers is maintained in a table to link books with their respective publishers.

Entity-Relationship Diagram:



Relational Model:

Book_Categories (id, category)
id (Primary Key)

Branches (id, branch)
id (Primary Key)

Student_Categories (cat_id, category, max_allowed)
cat_id (Primary Key)

Users (id, name, username, password, verification_status, remember_token)
id (Primary Key)

Students (student_id, first_name, last_name, approved, rejected, category, roll_num, branch, year, books_issued, email_id)
student_id (Primary Key)
category (Foreign Key references Student_Categories' cat_id)
branch (Foreign Key references Branches' id)

Books (book_id, title, description, category_id, added_by, added_at_timestamp)
book_id (Primary Key)
category_id (Foreign Key references Book_Categories' id)
added_by (Foreign Key references Users' id)

Book_Authors (book_id, author_id)
book_id (Primary Key, Foreign Key references Books' book_id)
author_id (Primary Key, Foreign Key references Authors' author_id)

Authors (author_id, author_name)
author_id (Primary Key)

Book_Reviews (review_id, book_id, review_text, rating, reviewed_by, reviewed_at)
review_id (Primary Key)
book_id (Foreign Key references Books' book_id)
reviewed_by (Foreign Key references Students' student_id)

Book_Publishers (publisher_id, publisher_name)
publisher_id (Primary Key)

Book_Publisher_Relationship (book_id, publisher_id)
book_id (Primary Key, Foreign Key references Books' book_id)
publisher_id (Primary Key, Foreign Key references Book_Publishers' publisher_id)

Book_Issue_Log (id, book_id, student_id, issue_by, issued_at, return_time, time_stamp)
id (Primary Key)
book_id (Foreign Key references Books' book_id)
student_id (Foreign Key references Students' student_id)
issue_by (Foreign Key references Users' id)

Availability (issue_id, book_id, available_status, added_by, added_at_timestamp)
issue_id (Primary Key)
book_id (Foreign Key references Books' book_id)
added_by (Foreign Key references Users' id)

SQL Queries:

1. Rank books by their review counts:

```sql
SELECT
    b.title,
    COUNT(br.review_id) AS review_count,
    DENSE_RANK() OVER (ORDER BY COUNT(br.review_id) DESC) AS review_rank
FROM books b
LEFT JOIN book_reviews br ON b.book_id = br.book_id
GROUP BY b.book_id
ORDER BY review_count DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	review_count	review_rank
Bossypants	5	1
Wuthering Heights	4	2
Sapiens: A Brief History of Humankind	4	2
Educated	4	2
Open Book	4	2
Foundation	3	3
Snow Crash	3	3

2. Calculate the average rating for each book:

```sql
SELECT
    b.title,
    AVG(br.rating) AS avg_rating,
    DENSE_RANK() OVER (ORDER BY AVG(br.rating) DESC) AS rating_rank
FROM books b
LEFT JOIN book_reviews br ON b.book_id = br.book_id
GROUP BY b.book_id
ORDER BY avg_rating DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	avg_rating	rating_rank
Neuromancer	5.0000	1
Sense and Sensibility	5.0000	1
Anna Karenina	5.0000	1
1984	5.0000	1
Moby-Dick	5.0000	1
Life of Pi	5.0000	1
The Hunger Games	5.0000	1

3. List students with the highest number of books issued:

```
SELECT
    CONCAT(s.first_name, ' ', s.last_name) AS student_name,
    s.books_issued,
    DENSE_RANK() OVER (ORDER BY s.books_issued DESC) AS issuance_rank
FROM students s
ORDER BY s.books_issued DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_name	books_issued	issuance_rank
Olivia Lee	3	1
Mia Smith	3	1
Amelia Robinson	3	1
Evelyn Rivera	3	1
Abigail Mitchell	3	1
Luna Stewart	3	1
Penelope Morris	3	1
Layla Hill	3	1

4. Retrieve the top 3 authors based on the number of books they've written:

```
SELECT
    a.author_name,
    COUNT(DISTINCT ba.book_id) AS book_count,
    RANK() OVER (ORDER BY COUNT(DISTINCT ba.book_id) DESC) AS author_rank
FROM authors a
INNER JOIN book_authors ba ON a.author_id = ba.author_id
GROUP BY a.author_id
ORDER BY book_count DESC
LIMIT 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

author_name	book_count	author_rank
Jane Austen	4	1
Brandon Sanderson	4	1
Gillian Flynn	4	1

5. Display the total count of books published by each publisher:

```sql
SELECT
    bp.publisher_name,
    COUNT(DISTINCT br.book_id) AS total_books_published,
    ROW_NUMBER() OVER (ORDER BY COUNT(DISTINCT br.book_id) DESC) AS publisher_rank
FROM book_publishers bp
INNER JOIN book_publisher_relationship bpr ON bp.publisher_id = bpr.publisher_id
INNER JOIN books br ON bpr.book_id = br.book_id
GROUP BY bp.publisher_id
ORDER BY total_books_published DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

publisher_name	total_books_published	publisher_rank
FutureWorld Press	6	1
MissingPerson Chronicles	6	2
SurvivalMemoirs Press	5	3
InvestigativeJournal Books	5	4
EpicVerse Publishing House	4	5
FantasyRealms Press	4	6
KnightlySaga Publications	4	7
ApocalypticComedy Books	4	8

6. Identifying Maximum Books Issued in a Year by Branch:

```sql
SELECT branch, year, MAX(issued_count) AS max_books_issued
FROM (
    SELECT s.branch, s.year, COUNT(*) AS issued_count,
           RANK() OVER (PARTITION BY s.branch, s.year ORDER BY COUNT(*) DESC) AS yearly_rank
    FROM students s
    JOIN book_issue_log bi ON s.student_id = bi.student_id
    GROUP BY s.branch, s.year
) AS yearly_stats
WHERE yearly_rank = 1
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

branch	year	max_books_issued
1	2023	57

7. Top N Books per Category:

```sql
SELECT
    book_id,
    title,
    category_id,
    rating
FROM (
    SELECT
        B.book_id,
        B.title,
        B.category_id,
        BR.rating,
        ROW_NUMBER() OVER (PARTITION BY B.category_id ORDER BY BR.rating DESC) AS rating_rank
    FROM books B
    INNER JOIN book_reviews BR ON B.book_id = BR.book_id
) ranked_books
WHERE rating_rank <= 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

book_id	title	category_id	rating
39	The Lies of Locke Lamora	1	5
35	Harry Potter and the Sorcerer's Stone	1	5
4	Good Omens	1	4
7	Dune	2	5
6	Neuromancer	2	5
42	Hyperion	2	5

Features:

- The dataset encompasses various tables such as **books**, **students**, **book_issue_log**, **book_reviews**, etc., enabling the management of book details, student information, book issuing, reviews, and more within the library system.
- Realtionships features a well-structured relational database model with established relationships between tables like **books** and **authors**, **book_reviews** and **students**, etc., facilitating efficient data retrieval and maintenance.
- Incorporates a trigger named **update_book_rating** that automatically computes and updates the average book rating in the **books** table whenever new reviews are added in the **book_reviews** table, ensuring real-time book rating updates.
- The presence of a stored procedure **IssueBook** helps manage book issuance effectively by checking book availability, updating book issue logs, and adjusting book availability status based on issued or returned books, enhancing the overall library operations.

Future Scope:

- Expand user profiles to include borrowing history, favorite genres, ratings, and reviews, providing a personalized experience and aiding in better book recommendations.
- Develop a recommendation feature based on users' borrowing history or reviews, suggesting similar books or genres of interest.