# Reports:

## **Objectives**

The objectives of the system are to:

1. Design and maintain a library database
2. Include library branches, books, users, and book lending records
3. Allow users to place holds on unavailable books and manage the hold queue

## Part 1: Designing the Database

1. **Mission Statement**

Mission Statements: to create a database for E-Library System

* The database should manage multiple libraries
* Each library can host its book collections and quantities
* Each book should have information about title, author, and categories
* Its user can borrow books, at most 2 at a time
* If the book is currently unavailable, then the user can hold the book (request for when available)
* If the book became available, the user in front of the queue will be given chance in the duration of a week to borrow the book
* If the user doesn’t borrow the book, the chance is given to the next in queue
* User can hold at most 2 books

1. **Creating Table Structures**

By the statement above, we need table to hold information for library, book, user, and book circulations.

We will create tables

* 1. Libraries
  2. Books
  3. Users

Then, to supplement book information, we create

* 1. Authors
  2. Categories

To store library book collection, we need a junction table between library and books, we call this table as

* 1. LibraryBooks

Then to store book circulations and to comply with business rules, we add

* 1. Loans
  2. Holds

A book's category, author, and a user's favorite genres (categories) can have multiple values, creating a many-to-many relationship. For simplicity, this report does not create a junction table (cross-reference) for these relationships. Instead, we handle them as one-to-many relationships.

1. **Determine Table Relationships**

Shown in the ERD’s

1. **Determine Business Rules**

Some of the business rules will be implemented in the backends.

Notable rules that implemented in the database are :

* 1. Library name, book title, and user name should be not null.
  2. Quantity of book collections should be nonnegative
  3. Gender will be stored in boolean. 0 for female, and 1 for male.
  4. Due date for borrowing a book is calculated automatically, 2 weeks after the loan date

1. **Implementing The Design**

The result of this Database Design is an Entity Relationship Diagram (ERD). We create the ERD as shown in Figure 1 Entity Relationship Diagram. There are 8 tables and its relations.

We create a SQL script (a DDL) to create the necessary table in the database. The script is stored in <https://github.com/Arasy/E-Library-System/blob/main/scripts/table_create.sql>

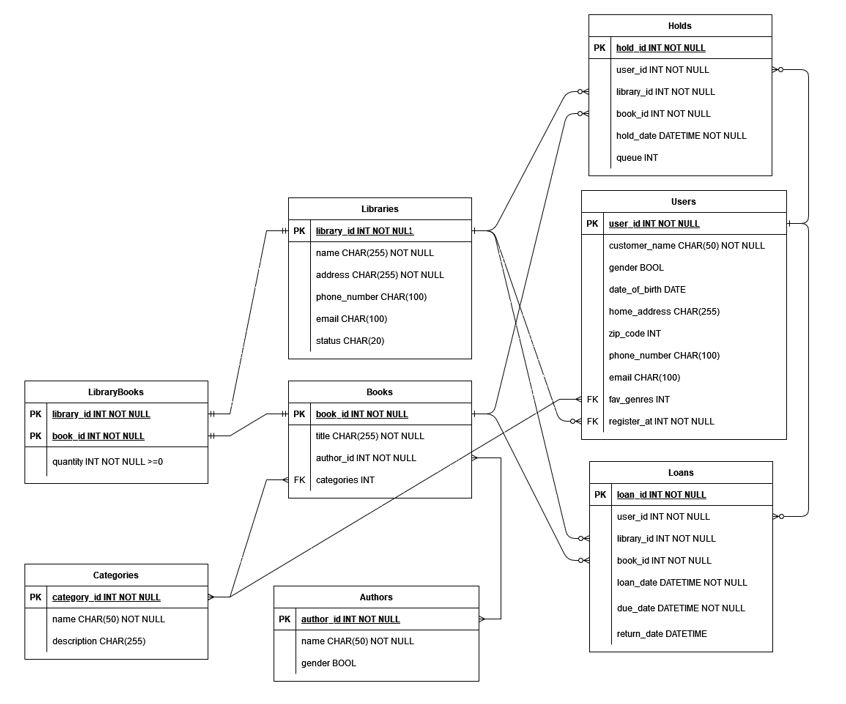
****

Figure 1 Entity Relationship Diagram

## Part 2: Populating Dummy Dataset

After implementing the database design, we create a dummy dataset for each table. The process is documented in the Python notebook file

<https://github.com/Arasy/E-Library-System/blob/main/scripts/data_dummy.ipynb> .

1. **Create Dummy Dataset**

Dummy datasets are created using Python and the Faker library, based on the design shown in the ERD (Entity-Relationship Diagram). The created datasets can be exported to JSON or CSV (optional). We include both formats in this report, and they are stored at <https://github.com/Arasy/E-Library-System/tree/main/dummy_data>

1. **Input Dummy Dataset into the Database:**

The datasets were inserted into the database using DBeaver as the interface. Some issues related to constraints on the due date column of the loans table occurred but have since been resolved. Notes on how we resolve the problem are recorded in the Python notebook file.

## Part 3: Questions

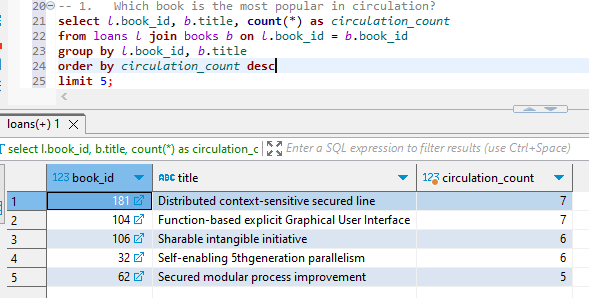
As we already have a library system and its (dummy) data in place, we are intrigued to answer several questions:

1. Which book is the most popular in circulation?
2. Which library lends books the most frequently?
3. Who are the top 5 borrowers?
4. Which genre has the highest circulation rate?
5. What is the percentage of borrowed books returned on time?

Those question will show us the big picture of our library. It can show us measures of effective library policies, book circulations, and user engagements. With proper follow up, it can contribute to effective resource management and service provision.

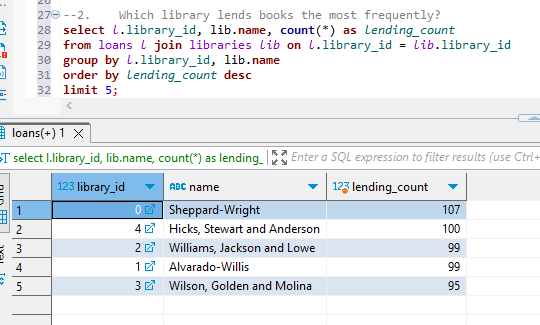
The query used to answer those questions can be accessed from <https://github.com/Arasy/E-Library-System/blob/main/scripts/questions.sql> . Below is a snippet of the code and the results:

1.

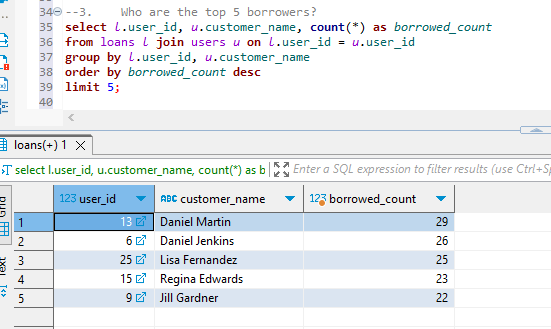


The most popular book is book id 181, titled “Distributed context-sensitive secured line”, having been borrowed a total of 7 times.

2.

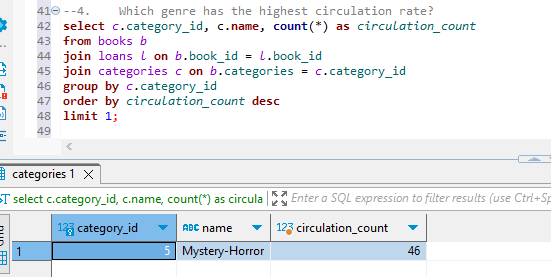


Sheppard-Wright Library is the most frequently lending books. With lending count 107 times.



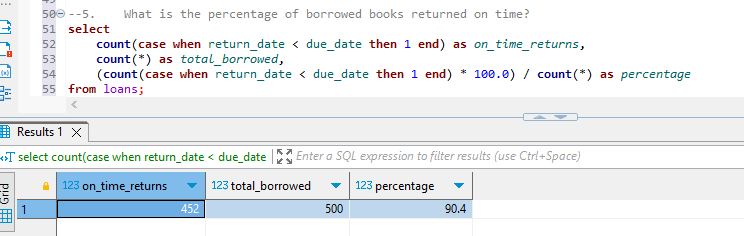
Daniel Martin (user\_id 13), Daniel Jenkins (user\_id 6), Lisa Fernandez (user\_id 25), Regina Edwards (user\_id 15), and Jill Gardner (user\_id 9) are the top borrowers.

4.



Mystery-Horror is the most popular genre (category). Having been borrowed 46 times.

5.



The percentage of book returned on time (return date < due date) is 90.4%. The formula is using exclusive symbol because the book is automatically returned after 2 weeks (equal condition), so it considered as not on time returns.

# Future Works

* + - 1. Connecting python script for generating dummy data directly into the database using sqlalchemy
      2. Implementing cross-reference tables for many-to-many relationships that occurred
      3. Creating view tables for simplified data analysis
      4. Creating backend and user interface, using Django or similar framework

# References

<https://www.drawio.com/doc/faq/add-rows>

<https://stackoverflow.com/questions/18992653/entity-relationship-diagram-how-does-the-is-a-relationship-translate-into-table>

<https://www.postgresql.org/docs/current/sql-createtable.html>

<https://stackoverflow.com/questions/21503658/prevent-less-than-zero-values-in-postgresql>

<https://stackoverflow.com/questions/9789736/how-to-implement-a-many-to-many-relationship-in-postgresql>

<https://faker.readthedocs.io/en/master/>

<https://www.geeksforgeeks.org/python-faker-library/>

<https://docs.sqlalchemy.org/en/20/orm/quickstart.html>

<https://stackoverflow.com/questions/3996904/generate-random-integers-between-0-and-9>

<https://stackoverflow.com/questions/55198378/faker-python-set-custom-word-list-for-faker-word>

<https://stackoverflow.com/questions/11875770/how-can-i-overcome-datetime-datetime-not-json-serializable>

<https://stackoverflow.com/questions/553303/generate-a-random-date-between-two-other-dates>

<https://dbeaver.com/docs/dbeaver/Data-transfer/#importing-data-from-csv-file>

<https://www.geeksforgeeks.org/connecting-postgresql-with-sqlalchemy-in-python/>

<https://stackoverflow.com/questions/24929735/how-to-calculate-date-difference-in-postgresql>