Simple E-Library Application Database System

Part 0: Introduction, Requirements, and Objectives

This is a report required to fulfills Pacmann JPP Max Program, SQL & Relational Database class project week 8. The overall goal of this project is divided into three parts, such as database design, populating database, and analyzing the e-library database system.

There are however, requirements for this database system, as mentioned below:

- **Manage multiple libraries**

  The application manages multiple libraries, each housing a diverse collection of books with varying quantities.

- **Book Collection**

  The database needs to store information about the diverse collection of books, including titles, authors, and available quantities. To make searching easier for users, books are also divided into categories such as: self-improvement, biography, Fantasy, Romance, Science Fiction, etc.

- **User Registration (Basically Handled by Application)**

  Users can register on the e-library platform. Registered users can interact with the platform by borrowing books, placing holds, and managing their account.

- **Loan and Hold System (Basically Handled by Application)**
  - Users can borrow books from any library in this application if the book is available.
  - The loan period is 2 weeks. Users can return books earlier than the due date.
  - Books will be automatically returned when they exceed the due date.
  - Users can only borrow 2 books at a time.
  - The platform keeps track of loan transactions, including loan dates, due dates, and return dates.
  - Users can place holds on books that are currently unavailable.
  - The library maintains a hold queue, and when a book becomes available, it can be borrowed by the customer at the front of the queue. Additionally, if a customer doesn't borrow a held book within one week, the book is released for other users to borrow.
  - Users can only hold 2 books at the same time.

As a side note, not all business rules can be enforced directly in the database. Some requirements or rules are implemented through the application or backend logic. This task is to provide tables that can assist the backend in enforcing those rules.

Therefore the objectives of this e-library database project is to design, populate, and analyze a relational database system that supports efficient management and accessibility of digital library resources. By implementing the database, the project aims to:

1. Optimize User Experience: Provide users with easy access to books, categorized by genre, author, and availability across different libraries.
2. Improve Inventory Management: Track the availability and circulation of books across multiple library branches, ensuring efficient resource distribution and timely updates for high-demand items.
3. Enable In-Depth Analysis: Gather insights into user engagement, popular genres, and demand patterns, helping the library make data-driven decisions about acquisitions, promotions, and service improvements.

Part 1: Database Design

Section 1. Mission Statement

The initial step in this project is to determine mission statement, with this mission statement we can also determine the scope, general description of its features, and its limitations. With this in mind, the mission statement for this database is as follows, “The database purpose is to provide a relational database for managing multi-library book collections from authors and publishers, user borrow and hold requests, and also supporting efficient backend operations for e-library platform”.

Based on the mission statement above, we can determine the limitations of this database system. One limitation to notice is that the database won’t include any currency transaction, hence making this e-library an open e-library with free registration and free to borrow books from library with limited access duration, as indicated by a loans table below.

Section 2. Table Structures

2.1 Fields, Keys, Business Rules, and Relations

With the mentioned mission statement on the previous section, each table needs to be structured to reflect the attributes and relationships of its corresponding entities. Based on the mission statement, we can determine the table that will be used for the database system, shown on the table below.

Table 1.1 Table’s Desription

Table Name	Description
Users	Stores information about registered user including email and username used for registration.
Libraries	Stores information about each libraries including its name and address (server location or url since it's e-library)

Categories	Stores information about book's genre and its id
Authors	Stores information about book's author and its id
Publishers	Stores information about book's publisher and its id
Books	Stores detailed information about each books. Including the title, genre/category, author, publisher, and its quantity in a library
Loans	Stores information used for tracking loan transactions including loan date, due date, and return date
Holds	Stores information used for managing hold requests for books currently available or for user that want to loan.

For each table, there needs to be fields, however each fields also needs to be considered to fulfill the business rule, constraints, and its relationship to each of the tables. Furthermore, each table's fields also needs to be able to accommodate the dummy data generation (which will be explained further in part 2) meant for system testing and analysis. Below is the table that will explain fields (and its reasonings), keys, and its business rules or constraints.

Table 1.2 Fields, Keys, and Business Rules

Users		
user_id (SERIAL)	PK/CK	A unique identifier for each user is critical for managing user accounts, linking users to loans, and tracking individual borrowing activity.
username (VARCHAR NOT NULL UNIQUE)	PK/CK	Including usernames allows users to log in, recognize their account in interfaces, and link activity within the system.
email (VARCHAR NOT NULL UNIQUE)		Each user's email ensures a unique point of contact and is useful for sending notifications or recovery information, as well as verifying account uniqueness.
Libraries		
library_id (VARCHAR)	PK/CK	We use library_id to uniquely identify each library branch within the system. The VARCHAR type provides flexibility for unique naming conventions.
library_name (VARCHAR NOT NULL UNIQUE)	PK/CK	Essential for identifying each library by name, which is important for user recognition and interface display.
location (VARCHAR)		Since this is an e-library, storing a URL or server location is necessary for digital access, allowing users or systems to connect to specific branches virtually.
Categories		
category_id (VARCHAR)	PK/CK	This field allows each genre or category to have a unique identifier. VARCHAR provides

		flexibility for naming while avoiding overlap with other IDs in the database.
category_name (VARCHAR NOT NULL UNIQUE)		Necessary for listing genre names in user interfaces, enabling users to filter or search by genre and allowing genre-based analysis.
Authors		
author_id (VARCHAR)	PK/CK	Each author needs a unique identifier for linking with books and analysis. The VARCHAR type offers flexibility for naming conventions.
author_name (VARCHAR NOT NULL)		Essential for identifying authors in the system, allowing users to search or browse by author and supporting analysis of popular authors.
Publishers		
publisher_id (VARCHAR)	PK/CK	Using a unique identifier for each publisher enables tracking and linking with books. The VARCHAR type provides flexibility for defining custom publisher IDs.
publisher_name (VARCHAR NOT NULL)		Necessary for identifying and displaying publisher information to users and for conducting publisher-based analyses.
Books		
book_id (VARCHAR)	PK/CK	Each book requires a unique identifier to track its details and transactions. Using VARCHAR allows us to customize IDs as needed.
title (VARCHAR NOT NULL)	PK/CK	The title is crucial for identifying books in searches and interfaces, as well as for managing records of borrowed books.
category_id (VARCHAR NOT NULL, FK to Categories)	FK	By linking each book to a category, we enable users to find books by genre, and we can analyze book circulation by category.
library_id (VARCHAR NOT NULL, FK to Libraries)	FK	Associating books with specific libraries allows for location-based inventory management and supports the ability to track availability at different branches.
publisher_id (VARCHAR NOT NULL, FK to Publishers)	FK	This field is essential to identify which publisher produced the book, supporting publisher-based analytics and tracking popular publishers.
author_id (VARCHAR NOT NULL, FK to Authors)	FK	Including author_id allows us to track books by author, supporting analyses on popular authors and ensuring that each book has an author associated with it.
quantity (INT CHECK (quantity >= 0))		This field represents the number of copies of each book available, which is necessary for

		inventory management, tracking book availability, and supporting recommendations to acquire additional copies if demand exceeds supply.
Loans		
loan_id (SERIAL)	PK/CK	Each loan transaction requires a unique identifier for tracking purposes, generated automatically to manage sequential records.
user_id (INT NOT NULL, FK to Users)	FK	Linking loans to users is essential for tracking borrowing behavior, allowing the library to see which users are most active.
book_id (VARCHAR NOT NULL, FK to Books)	FK	Tracking which book was borrowed is fundamental for inventory management and understanding book popularity.
library_id (VARCHAR NOT NULL, FK to Libraries)	FK	Including the library's identifier shows where the book was borrowed, supporting branch-specific analyses of loan activity.
loan_date (DATE NOT NULL)		This field captures when a book was borrowed, which is crucial for tracking loan duration and overdue books.
due_date (DATE NOT NULL)		The due date is necessary to set return expectations and calculate whether a book is overdue, supporting reminders and overdue policies.
return_date (DATE)		Tracking the return date is essential for managing loan completion and analyzing overdue trends, helping refine loan policies.
Holds		
hold_id (SERIAL)	PK/CK	Each hold request needs a unique identifier to track individual requests, generated automatically for convenience.
user_id (INT NOT NULL, FK to Users)	FK	Associating holds with users is essential to see who requested the hold and to notify the user when the book is available.
book_id (VARCHAR NOT NULL, FK to Books)	FK	Tracking which book is held helps understand demand for specific titles and manage the queue for popular books.
library_id (VARCHAR NOT NULL, FK to Libraries)	FK	Identifying the library where the hold was placed helps in location-based inventory management and ensures that the right branch fulfills the hold request.

hold_date (DATE NOT NULL)		This field tracks when a hold was requested, allowing for analysis of demand trends and hold durations.
release_date (DATE)		Recording the release date helps manage the hold queue, showing when a book became available for the next user or if the hold was canceled.

Another consideration in determining the tables and its fields above is to reduce redundancy, thus, the tables above were normalized into its 3NF before design finalization. With the aforementioned foreign key (marked by FK) and primary key (marked by PK) above, we can then create a relationship between tables, as shown in diagram below.

Table 1.3 Table's Relations

	users	**libraries**	**categories**	**authors**	**publishers**	**books**	**loans**	**holds**
users							1 - N	1 - N
libraries						1 - N	1 - N	1 - N
categories						1 - N		
authors						1 - N		
publishers						1 - N		
books		1 - N	1 - N	1 - N	1 - N		1 - N	1 - N
loans	1 - N	1 - N				1 - N		
holds	1 - N	1 - N				1 - N		

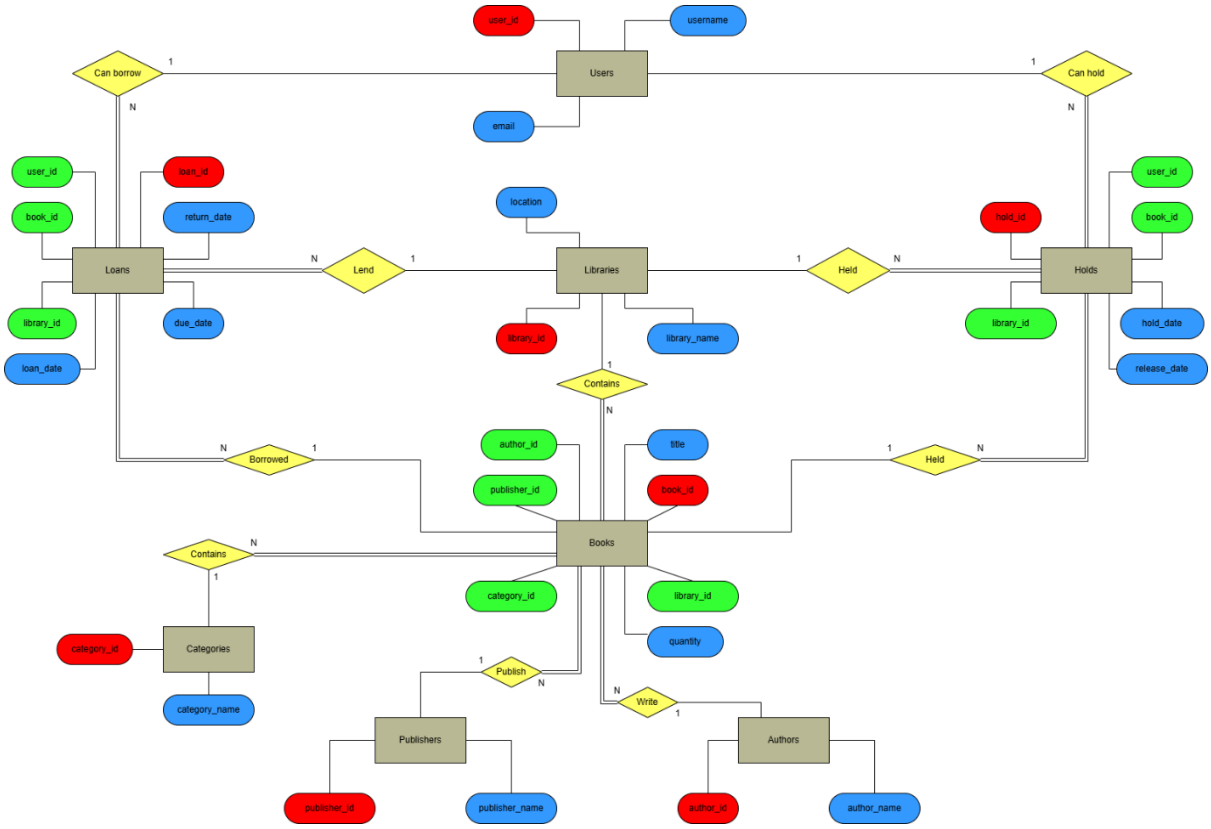
Based on the diagram above, all the relation from each tables are one to many, however below are the reasoning behind the table's relationships.

- **Books and Categories:** Each book is assigned to a single category (genre) through a foreign key relationship with the Categories table. This relationship allows books to be organized by genre, supporting both user browsing and genre-specific analysis.
- **Books and Libraries:** The Books table has a foreign key to the Libraries table, associating each book with a specific library branch. This relationship enables location-based tracking of book availability and supports queries that examine circulation rates at different branches.
- **Books and Authors:** Each book is linked to a single author through a foreign key to the Authors table. This relationship allows tracking of popular authors and supports searches and analysis based on authorship.
- **Books and Publishers:** The Books table links to Publishers to track the origin of each book. This relationship enables insights into publisher popularity and helps identify publisher-specific trends in borrowing.
- **Users and Loans:** Each user can have multiple loans, with the Loans table recording each borrowing transaction and linking back to Users via a foreign key. This

relationship is critical for tracking user engagement and managing individual borrowing records.

- **Users and Holds:** Users can place multiple holds, tracked in the Holds table, with a foreign key linking back to Users. This relationship helps manage demand for checked-out books and allows the library to notify users when their requested books are available.
- **Books and Loans:** The Loans table includes a foreign key to Books, allowing each book's borrowing history to be tracked. This relationship supports analysis of book popularity and enables efficient management of book returns and overdue notifications.
- **Books and Holds:** Each book can have multiple hold requests, recorded in the Holds table with a foreign key link to Books. This setup is crucial for tracking demand for unavailable books and prioritizing availability based on user holds.

With the table relationships above, we can model the Entity Relationship Diagram, as shown below.



Picture 1.1 Modeled Table's ERD

2.2 Data Definition Language (DDL) and Generated ERD

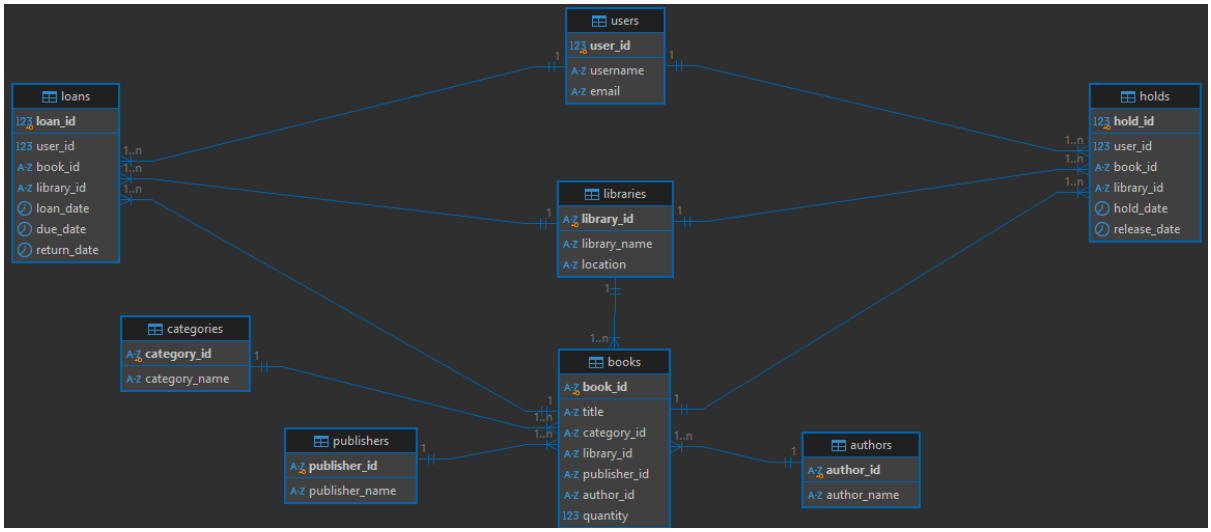
With the modeled relationship above we can start create the DDL for generating tables and its relationship within the database. However, as a side note we need to be careful with the order of table creations, below is the DDL.

```sql
--Create libraries table
CREATE TABLE Libraries (
    library_id VARCHAR(50) PRIMARY KEY,
    library_name VARCHAR(100) NOT NULL UNIQUE,
    location VARCHAR(100)
);

--Create categories table
CREATE TABLE Categories (
    category_id VARCHAR(50) PRIMARY KEY,
    category_name VARCHAR(50) NOT NULL UNIQUE
);

--Create users table
CREATE TABLE Users (
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE
);

--Create authors table
CREATE TABLE Authors (
    author_id VARCHAR(50) PRIMARY KEY,
    author_name VARCHAR(100) NOT NULL
);

--Create publishers table
CREATE TABLE Publishers (
    publisher_id VARCHAR(50) PRIMARY KEY,
    publisher_name VARCHAR(100) NOT NULL
);

--Create books table
CREATE TABLE Books (
    book_id VARCHAR(50) PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    category_id VARCHAR(50) NOT NULL,
    library_id VARCHAR(50) NOT NULL,
    publisher_id VARCHAR(50) NOT NULL,
    author_id VARCHAR(50) NOT NULL,
    quantity INT CHECK (quantity >= 0), -- Ensures no negative quantities
    CONSTRAINT
       fk_categories
       FOREIGN KEY (category_id)
       REFERENCES Categories(category_id),
    CONSTRAINT
       fk_libraries
       FOREIGN KEY (library_id)
       REFERENCES Libraries(library_id),
    CONSTRAINT
       fk_publishers
       FOREIGN KEY (publisher_id)
       REFERENCES Publishers(publisher_id),
    CONSTRAINT
       fk_authors
       FOREIGN KEY (author_id)
       REFERENCES Authors(author_id)
);
```

```sql
--Create loans table
CREATE TABLE Loans (
    loan_id SERIAL PRIMARY KEY,
    user_id INT NOT NULL,
    book_id VARCHAR(50) NOT NULL,
    library_id VARCHAR(50) NOT NULL,
    loan_date DATE NOT NULL,
    due_date DATE NOT NULL,
    return_date DATE, --Nullable until returned
    CONSTRAINT
       fk_users
       FOREIGN KEY (user_id)
       REFERENCES Users(user_id),
    CONSTRAINT
       fk_books
       FOREIGN KEY (book_id)
       REFERENCES Books(book_id),
    CONSTRAINT
       fk_libraries
       FOREIGN KEY (library_id)
       REFERENCES Libraries(library_id)
);

--Create holds table
CREATE TABLE Holds (
    hold_id SERIAL PRIMARY KEY,
    user_id INT NOT NULL,
    book_id VARCHAR(50) NOT NULL,
    library_id VARCHAR(50) NOT NULL,
    hold_date DATE NOT NULL,
    release_date DATE, --Nullable until nol longer held
    CONSTRAINT
       fk_users
       FOREIGN KEY (user_id)
       REFERENCES Users(user_id),
    CONSTRAINT
       fk_books
       FOREIGN KEY (book_id)
       REFERENCES Books(book_id),
    CONSTRAINT
       fk_libraries
       FOREIGN KEY (library_id)
       REFERENCES Libraries(library_id)
);
```

As mentioned before, we must be careful with the order of the table's creation. In this case the DDL create the table of "outer layer", which means the parent table that will give its "1-to" relation onto its "to-many" tables. In this case parents table are the Users, Libraries, Publishers, Authors, and Categories table. However there is another reasoning behind this order, it's because of the children table that needs to referencing its parents table with its "REFERENCES" clause.

After the DDL is executed, then the tables are generated. With the tables generated, we can finally see the actual ERD that generated by the system. The generated ERD is shown below.



Picture 1.2 Generated ERD

With the tables created and relationship establised, what left are populating the database with dummy data, and creating case analysis based on the populated data. As a side note, the tables might look so simple or even too small and unvaried, in this case the next part will discuss about matters still related to part one, and also answering questions such as, “why there are no table to handle multiple categories and authors ?”.

Part 2: Populating The Database

Section 1. Data Source and Dummy Data

1.1 Reasoning

For the small amount of data in books table, namely book title and its categories, data scraping method was used. The reasoning behind this is coincidentally I still have the scraped data for the previous mini project, however there is more important reason as the data generated by faker can be really weird such as the category cannot be something like Self – Improvement or Fiction, so using scraping will ensure that the categories data realistic. Instead Faker will generate random words that are not specifically intended for book categories. Another reason is because the task objective is about e-library, I decided to look for data that are the most representative in the library, that are books.

This reasoning explains previous question, “is this database able to handle multiple category or author”. The answer to that is unfortunately no, due to the nature of Faker and the scraped data, in which from the scraped site one book can only have one category, not multiple category like Fiction and Mystery one book for example.

As for the reason why using scraping and Faker, the reason is because I want to implement the database system with automation in mind, so I decided to just using purely Python or the backend for generating dummies.

1.2 Data Scraping

This subsection will not explain how to scrape data for too far (the Faker subsection too), only the website source, tools, and steps. Because I want to keep the focus on the SQL and RDMS and not to extend the report for unnecessary topics. Instead, I will upload the script and the scraped result in the same folder as this report

Scraping site: "https://books.toscrape.com/"

As for the source, the website that created specifically for scraping books was used. This was chosen to ensure the accuracy of the title and category data, the titles in this site are more or less acurate to the real world title. The only data that were not available are publisher and author.

We can scrape these data using the BeautifulSoup (BS4) module in Python, since the website is a static website, therefore no driver needed. Beside the BS4 module, we can also use the requests module to send the HTTP request to the site and getting the content we want to convert it into the "soup" object for extracting its HTML tags or elements. Another tool is Pandas to easily and efficiently convert the content into DataFrames and furthermore into csv file.

Here are the step by steps in how to generate dataset from scraping:

1. Create request to http, assign its response to variable that we will use for soup object.
2. Extract the available categories into a list, however there were two category that undetermined, and I decided to remove it.
3. After that, we can extract every url of the corresponding category, the purpose of this is to automatically loop to the next url after the script finish extracting. Then, we can exclude the removed genre.
4. With this we can map genre and its url by combining extracted content from number 2 and 3.
5. The final step is to get content contained by the corresponding html tag inside each url that we loop, and store it inside a DataFrame and save it into a csv.

To cut this subsection short, below is the csv result of the scraped data, however using scraping we are able to extract 782 books across 48 categories.

	title	genre
0	It's Only the Himalayas	Travel
1	Full Moon over Noahâ□□s Ark: An Odyssey to Mou...	Travel
2	See America: A Celebration of Our National Par...	Travel
3	Vagabonding: An Uncommon Guide to the Art of L...	Travel
4	Under the Tuscan Sun	Travel
5	A Summer In Europe	Travel
6	The Great Railway Bazaar	Travel
7	A Year in Provence (Provence #1)	Travel
8	The Road to Little Dribbling: Adventures of an...	Travel
9	Neither Here nor There: Travels in Europe	Travel

Picture 2.1 Scraped Data

1.3 Faker and Other Dummy Data

The Faker module was used to generate most of the dummy data inside the table. But here is the catch, for the sake of consistency we must treat the data generation as if it is creating DDL, which means there are table priorities. In this case, like the DDL we must prioritize the parents table first and then the children tables. A side note, by using Faker I made nine different functions, eight to generate data in each tables, and one for converting to csv.

As mentioned before, to keep this subsection short enough and not unnecessarily long, here we will only discuss the data generation related to the books table. This include books, libraries, categories, publishers, and authors.

- **Libraries Table**

  For libraries we can make a function that receive a number of data that we want to generate as an input, the function will return the DataFrame.

```
# Generate Libraries
def lib_gen(n_lib):

    libraries = {
        'library_id': [f"L-{i:02}" for i in range(1, n_lib + 1)],
        'library_name': [fake.unique.company() for _ in range(n_lib)],
        'location': [fake.country() for _ in range(n_lib)]
    }

    lib_df = pd.DataFrame(libraries)

    return lib_df
```

Faker is used for creating the dummy data for library_name and the location of the library, while the library_id is simply using range of the inputed number of data. Notice that the library_id is a string because on we used VARCHAR in the DDL.

- **Categories Table**

Categories table is one of the table in which we use the scraped data. The function to generate data in this table receive the scraped data itself.

```
# Generate Categories
def category_gen(scraped_data):

    read_data = pd.read_csv(scraped_data)
    unique_genres = read_data['genre'].unique()

    categories = {
        'category_id': [f"C-{i:02}" for i in range(1, len(unique_genres) +
1)],
        'category_name': unique_genres
    }

    category_df = pd.DataFrame(categories)

    return category_df
```

By reading the scraped data using Pandas, we can then extract only the unique genres, category_id is using the same method as library_id.

- **Publishers Table**

Publishers table generally uses the same method as Libraries table to generate its dummy data.

```
# Generate Publishers
def pub_gen(n_pub):

    publishers = {
        'publisher_id': [f"P-{i:02}" for i in range(1, n_pub + 1)],
        'publisher_name': [fake.company() for _ in range(n_pub)]
    }

    pub_df = pd.DataFrame(publishers)

    return pub_df
```

The only difference is in the id, notice the usage of "P", indicating the id for publisher.

- **Authors Table**

  Authors table still generally use the same method as the Libraries and Publishers table, the only difference is the id.

```
# Generate Authors
def authors_gen(n_authors):

    authors = {
        'author_id': [f"A-{i:02}" for i in range(1, n_authors + 1)],
        'author_name': [fake.name() for _ in range(n_authors)]
    }

    author_df = pd.DataFrame(authors)

    return author_df
```

- **Books Table**

  Books table is where the informations from previous tables are culminated, in this table we will use the scraped data, and the DataFrame from the previous generated dummy datasets.

```
# Generate Books
def books_gen(scraped_data, lib_df, pub_df, author_df):

    # Generate category data and read the scraped book data
    category_df = category_gen(scraped_data)
    read_data = pd.read_csv(scraped_data)

    # Create a mapping of genre to category_id
    genre_to_category_id = dict(zip(category_df['category_name'],
category_df['category_id']))

    # Generate book data with correct category_id
    books = {
        'book_id': [f"B-{i:03}" for i in range(1, len(read_data['title']) +
1)],
        'title': read_data['title'].tolist(),  # Use the titles from scraped
data
        'category_id': [genre_to_category_id[genre] for genre in
read_data['genre']],
        'library_id': [random.choice(lib_df['library_id']) for _ in
range(len(read_data['title']))],
```

```
        'publisher_id': [random.choice(pub_df['publisher_id']) for _ in
range(len(read_data['title']))],
        'author_id': [random.choice(author_df['author_id']) for _ in
range(len(read_data['title']))],
        'quantity': [random.randint(1, 10) for _ in
range(len(read_data['title']))]
    }

    book_df = pd.DataFrame(books)

    return book_df
```

Notice there is no category DataFrames inside the function's arguments, because we re-read that again using pandas and re-map it using dictionary and zip function to ensure that there are no duplicate keys or in this case no duplicate categories.

As for the other inputs, we can randomize it by using the random module and its choice method, so that the randomized choice would be randomly assigned to each corresponding fields.

- **.csv Conversion**

  To convert any dataset that we generated into the csv, we can use the function below.

```
# CSV converter
def csv_convert(df_tables, filename):

    csv_file = df_tables.to_csv(filename, index=False)

    return csv_file
```

This function receive any dummy dataset that we generated before and the filename that we want to save as.

By using the functions above, we can generate dummy data consistently and quite realistically. The only super unrealistic part is perhaps when the Harry Potter novel series was not written by J.K. Rowling, because of the reason mentioned above that the scraping website did not provide that information.

However, most of the function, especially the parents table require number of data that we want to generate as a input requirement, here is the number of data that I use:

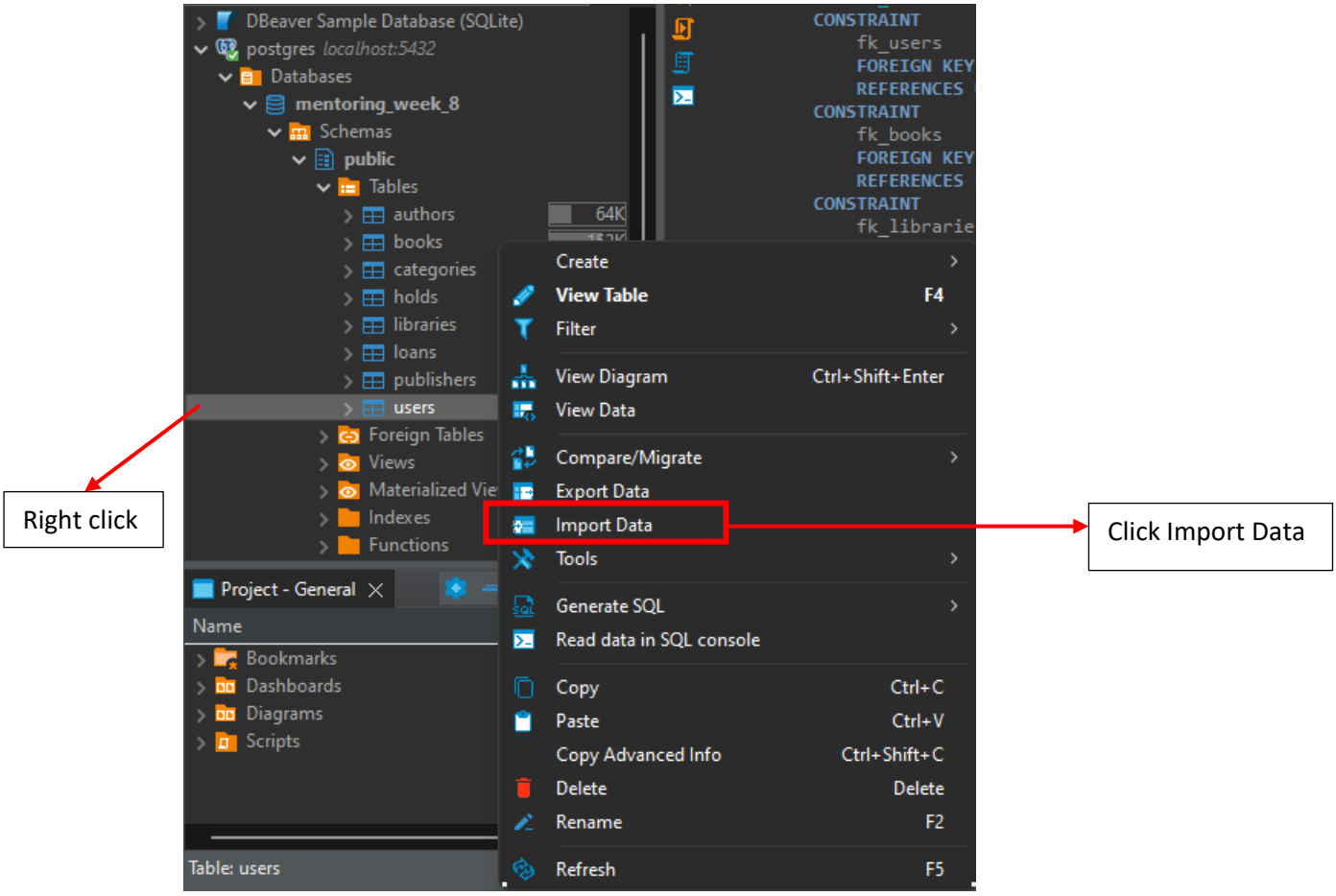
- Authors: 365 authors. This allows for the one authors can write many books statement that we saw in ERD.
- Publishers: 60 publishers, ensuring a balance between publisher diversity and practical analysis of publisher-based trends in borrowing and availability.

- Libraries: 8 libraries, but the location cannot be unique because there are two libraries in a country, representing a multi-branch library system to enable location-specific inventory tracking and analysis.
- Users: 650 users, providing a sufficient user base for analyzing borrowing patterns and engagement trends.
- Loans: 2000 loan records, simulating a dynamic borrowing environment that allows for detailed circulation and overdue analysis.
- Holds: 455 holds, reflecting demand for high-interest books and supporting availability analysis.
- Loans and Holds Date: The generated dates for loan and hold start from 2023-06-01 to 2024-06-30, making it a year period.

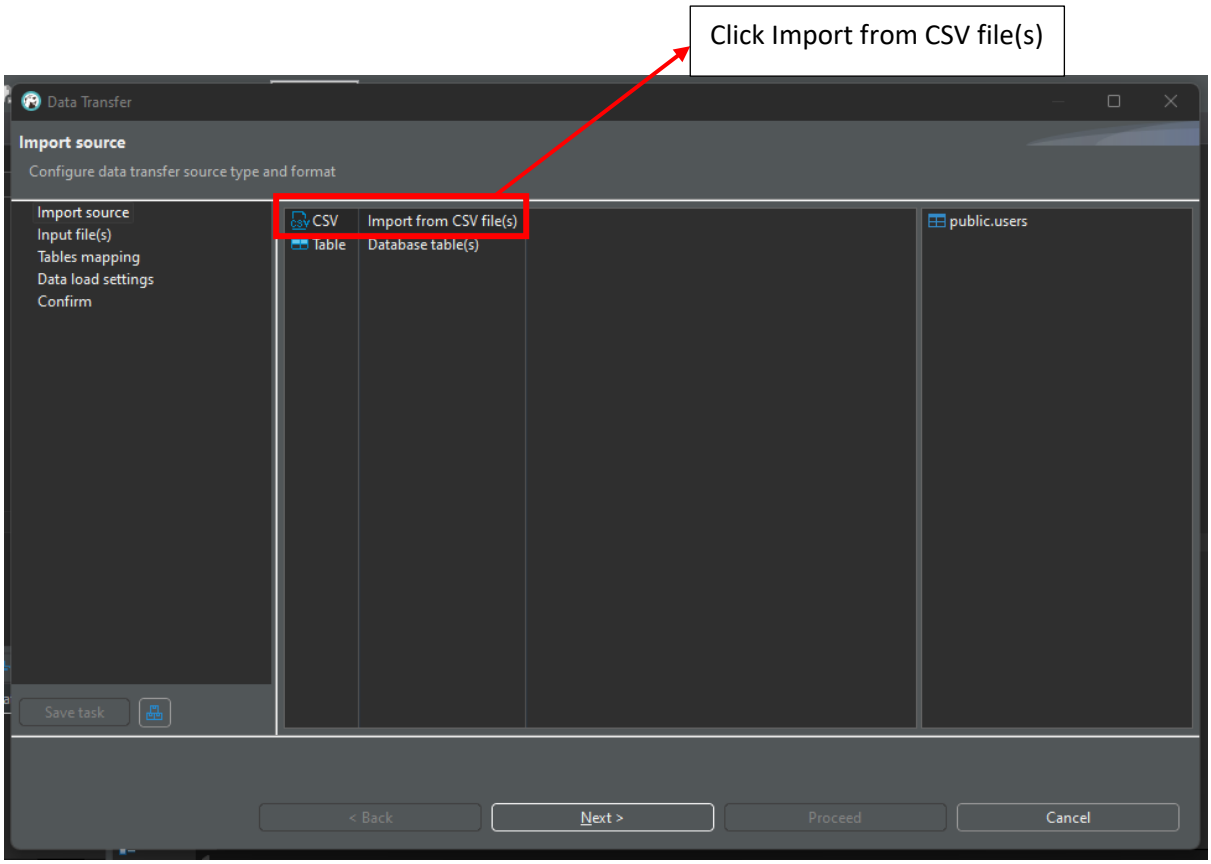
Section 2. Importing Data

For data imports, we have two choices here. One is to use a 'COPY' statement in the query or we can use any of our choice of GUI application. In this case we will use the built in function inside the DBeaver application. Here are the steps.

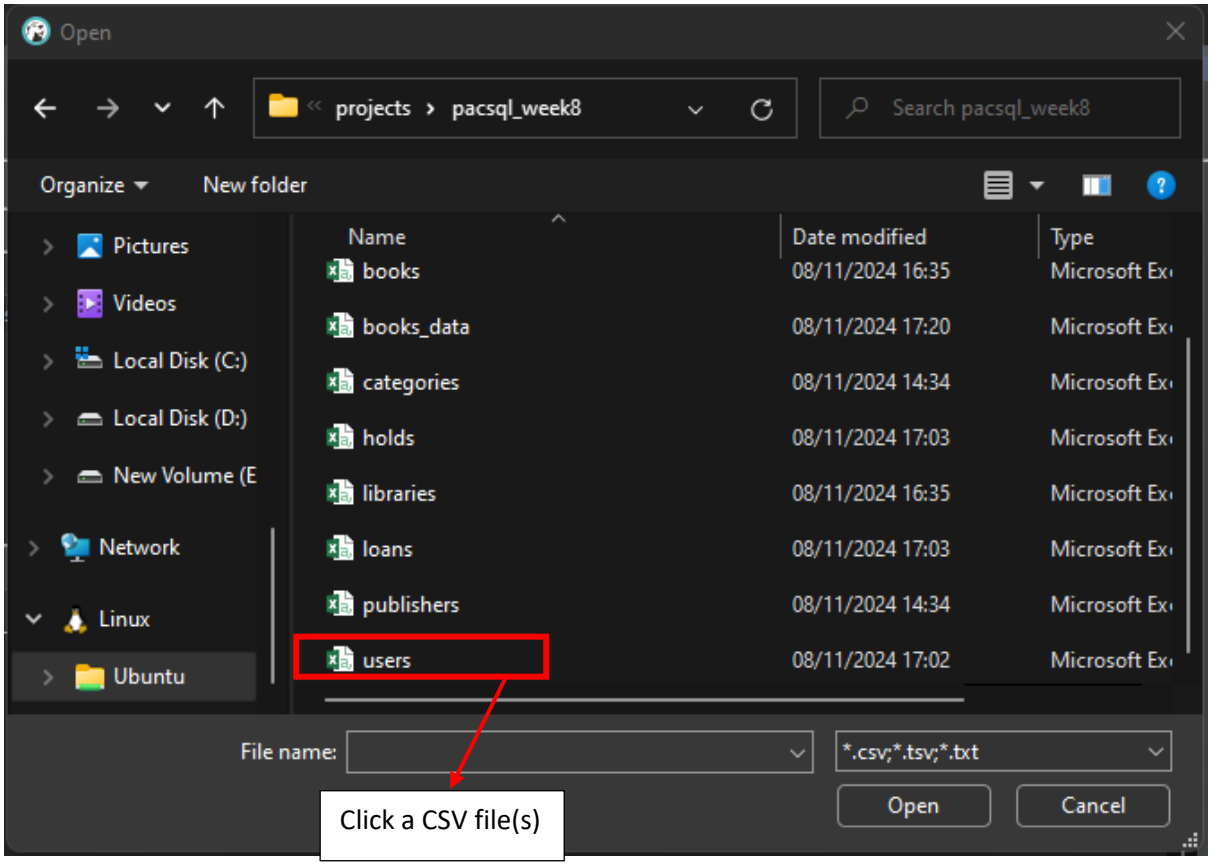
1. Right click any table, then click import.

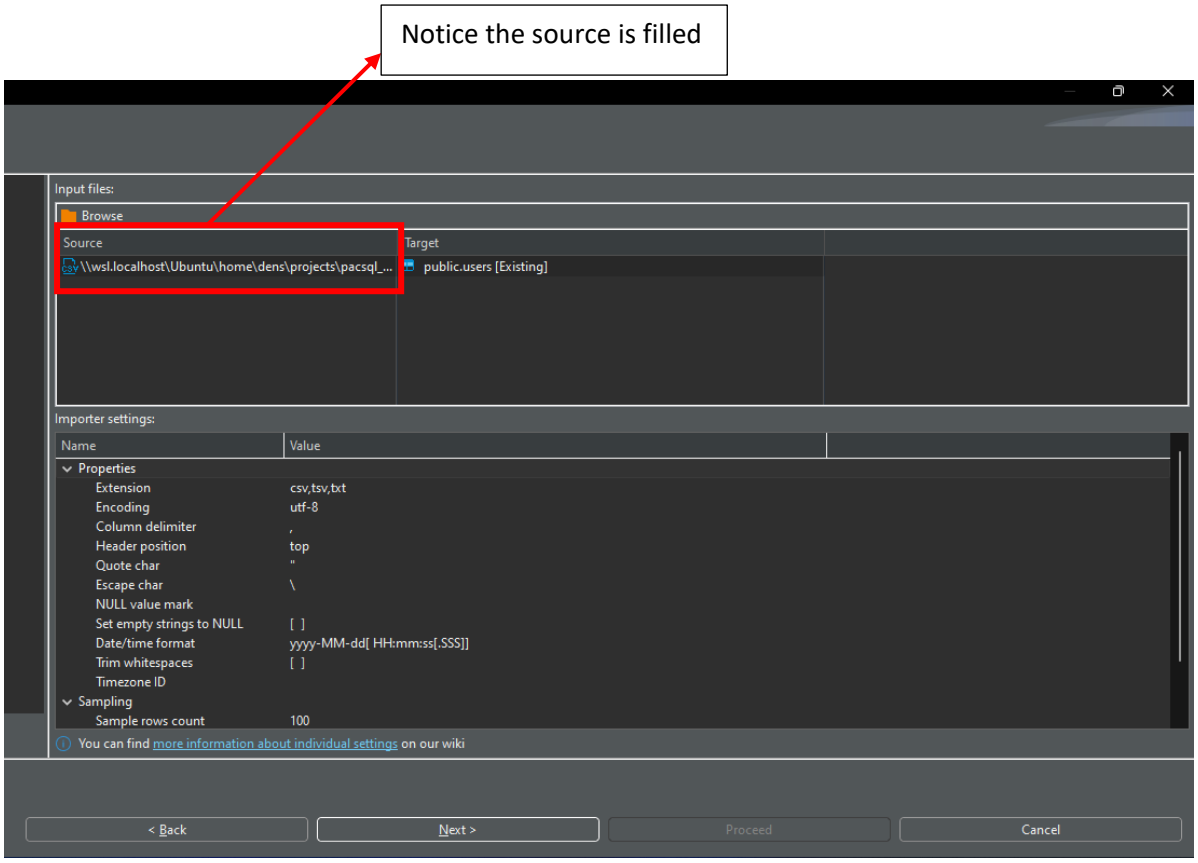


2. Click import from csv file, then Next >.

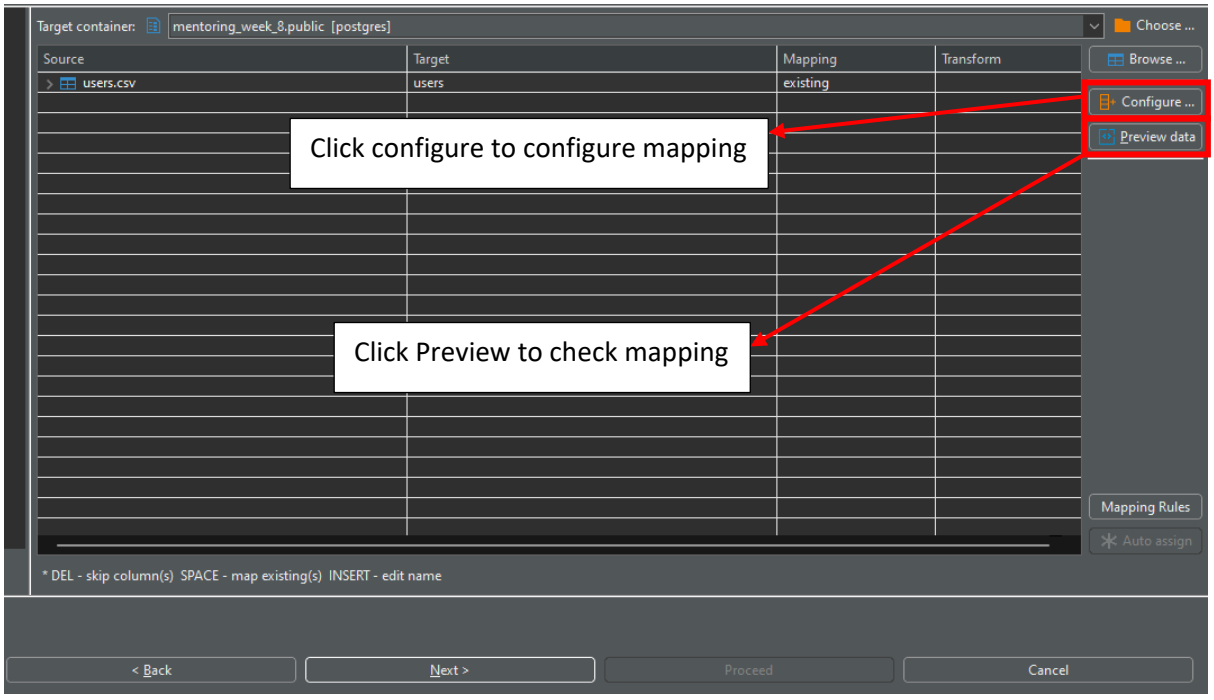


3. Click any csv file that we generate before, then Next >.

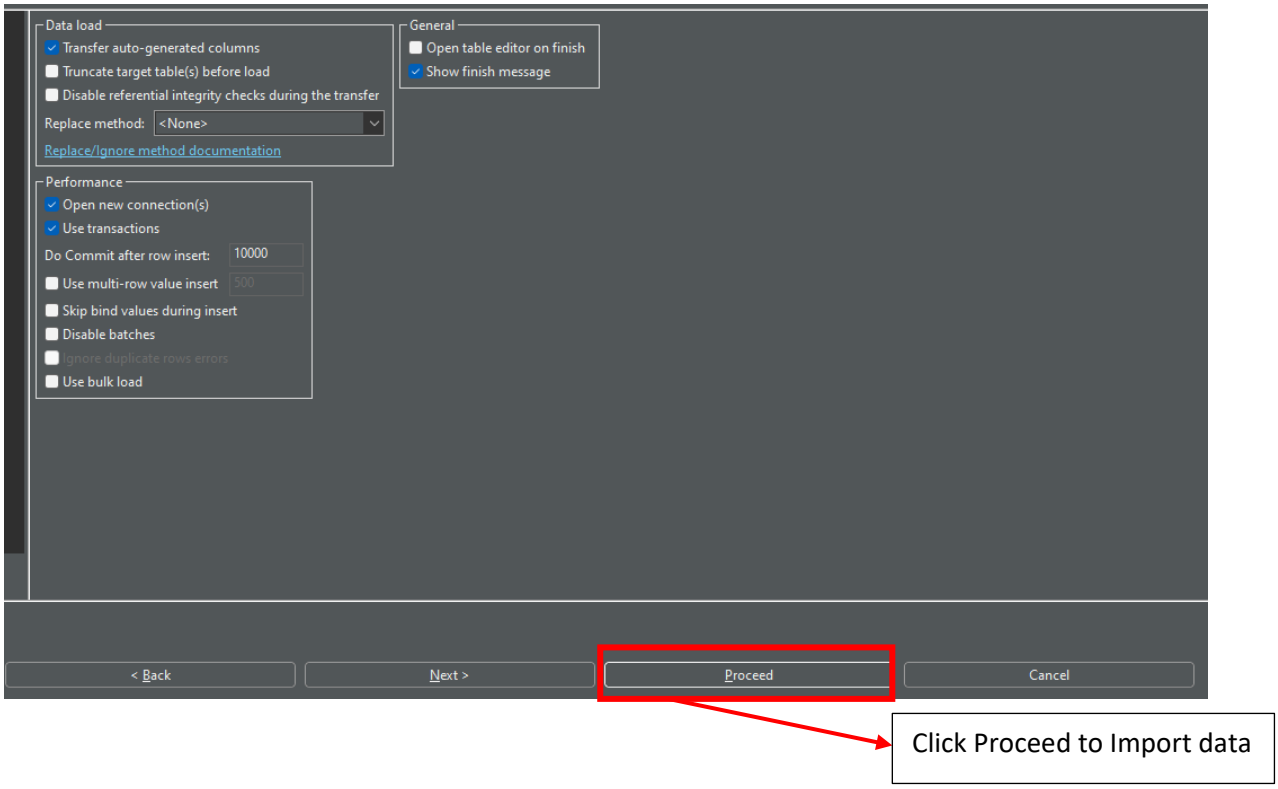




4. Map the data and its table, in this case we have already matched data and the table, including the fields order.



5. Click proceed



With this, the data is safely imported, we can fully skip the step 4 to map data, because we already following the database design up to data generation properly. But, in this case we still need to preview the data imports just to make sure.

Part 3: Case Analysis

Section 1. Case Questions and Importance

With database design finally established and the dummy data successfully generated, we can then create case questions for analysis, below are the proposed questions and its importance.

1. **Which books and authors are most frequently borrowed ?**

   Importance: Identifying the most popular books and authors helps the library make data-driven decisions about inventory, highlighting which books may need additional copies to meet demand. It also helps understand user preferences, which can guide future acquisitions**.**

2. **What are the most popular genres and which books in each genre have the highest borrowing rates ?**

   Importance: Understanding genre popularity ensures that the library's collection aligns with user interests, promoting higher engagement and satisfaction. Knowing which books are in demand within popular genres also allows the library to optimize book availability and reduce wait times.

3. **What is the average number of books borrowed per user, and how many users are repeat borrowers ?**

   Importance: Analyzing borrowing patterns provides insights into user behavior, helping the library determine how effectively it engages its users. It also helps identify which users are repeat borrowers, which can inform loyalty programs or targeted marketing efforts to boost engagement among infrequent borrowers.

4. **Which library has the highest circulation rate, and what is the average book quantity available per library ?**

   Importance: Understanding circulation rates across different library branches helps allocate resources effectively. High-circulation libraries may need larger collections, while lower-circulation locations might benefit from targeted promotions or special events to increase foot traffic and borrowing rates.

5. **How many books are overdue, and which libraries have the highest number of overdue loans ?**

   Importance: Tracking overdue loans is essential for managing inventory and maintaining circulation efficiency. Identifying libraries with high overdue rates allows the library to implement measures, like reminder notifications or policy adjustments, to reduce overdue instances and improve user accountability.

The questions above will be answered and analized in the next section. Insights and recommendations will also be provided in the next section.

Section 2. Analysis and Insights

Question 1. Which books and authors are most frequently borrowed ?

- **SQL Query**

```sql
-- Top 10 most borrowed books
SELECT
      b.book_id,
      b.title,
      COUNT(l.loan_id) AS borrow_count
FROM
      books b
JOIN
      loans l USING(book_id)
GROUP BY
      1, 2
ORDER BY
      borrow_count DESC
LIMIT 10;

-- Most frequently borrowed authors
SELECT
      a.author_id,
      a.author_name,
```

```
      COUNT(l.loan_id) AS borrow_count
FROM
      authors a
JOIN
      books b USING(author_id)
JOIN
      loans l USING(book_id)
GROUP BY
      1, 2
ORDER BY
      borrow_count DESC
LIMIT 10;
```

- **Query Result**

	book_id	title	borrow_count
1	B-553	Nightstruck: A Novel	10
2	B-290	Kitchens of the Great Midwest	9
3	B-747	Far From True (Promise Falls Trilogy #2)	8
4	B-679	Catherine the Great: Portrait of a Woman	8
5	B-732	Rework	7
6	B-079	Outcast, Vol. 1: A Darkness Surrounds Him (Outcast #1)	7
7	B-051	Love, Lies and Spies	7
8	B-153	Wuthering Heights	7
9	B-112	Death Note, Vol. 6: Give-and-Take (Death Note #6)	7
10	B-049	The Marriage of Opposites	7

Top 10 Frequently Borrowed Books

	author_id	author_name	borrow_count
1	A-286	Mariah Tran	24
2	A-318	Andrew Li	20
3	A-15	Mary Palmer	19
4	A-291	Joseph Ramsey	19
5	A-118	Jessica Brewer	19
6	A-243	Jared Craig	18
7	A-201	Jennifer Pierce	17
8	A-330	Mrs. Lisa Thompson	17
9	A-94	Jason Nelson	17
10	A-194	Anthony Adams	16

Top 10 Frequently Borrowed Books Based on Author

- **Description**

  The most borrowed book is titled Nightstruck: A Novel with 10 times borrowed, followed closely by book titled Kitchens of the Great Midwest with 9 times borrowed. While based on authors, books written by Mariah Tran borrowed 24 times, and followed by books written by Andrew Li borrowed 20 times.

Based on this findings, we should consider acquiring additional copies of high demand titles and promote the most popular authors through display of digital highlights.

Question 2. What are the most popular genres ?

- **SQL Query**

```sql
SELECT
	c.category_name,
	COUNT(l.loan_id) AS borrow_count
FROM
	categories c
JOIN
	books b USING(category_id)
JOIN
	loans l USING(book_id)
GROUP BY
	c.category_name
ORDER BY
	borrow_count DESC
LIMIT 10;
```

- **Query Result**

	category_name	borrow_count
1	Nonfiction	265
2	Sequential Art	194
3	Fiction	151
4	Young Adult	144
5	Fantasy	109
6	Romance	93
7	Mystery	87
8	Food and Drink	83
9	Childrens	77
10	Historical Fiction	65

- **Description**

Non-fiction genre is the most borrowed books with 265 times borrowed, followed by sequential art with 194 times borrowed. Based on this finding, we can consider to increase the availability of books in high-demand genres, and consider featuring genre-based recommendations to engage readers.

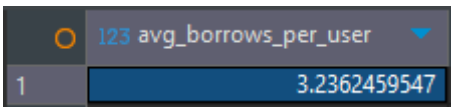
Question 3. What is the average number of books borrowed per user, and how many users are repeat borrowers ?

- **SQL Query**

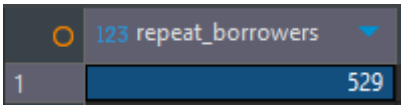
```sql
-- Average number of books borrowed per user
SELECT
```

```
        AVG(borrow_count) AS avg_borrows_per_user
FROM (
    SELECT
        user_id,
        COUNT(loan_id) AS borrow_count
    FROM
        loans
    GROUP BY
        user_id
) AS user_borrows;

-- Number of repeat borrowers
SELECT
        COUNT(user_id) AS repeat_borrowers
FROM (
    SELECT
        user_id,
        COUNT(loan_id) AS borrow_count
    FROM
        loans
    GROUP BY
        user_id
    HAVING
        COUNT(loan_id) > 1
) AS repeat_users;

-- Repeat borrowers with their usernames and borrow count
SELECT
        u.username,
        COUNT(l.loan_id) AS borrow_count
FROM
        Users u
JOIN
        Loans l ON u.user_id = l.user_id
GROUP BY 1
HAVING
        COUNT(l.loan_id) > 1
ORDER BY
        borrow_count DESC;
```

- **Query Result**



Average Number of Borrowed Books per User



Number of Repeat Borrowers

	username	borrow_count
1	heidi74	10
2	crystal38	9
3	rmitchell	8
4	lovebecky	8
5	lhenderson	8
6	rnixon	8
7	tylerbennett	8
8	dickersonmark	8
9	benjaminpowers	7
10	davidsonrobert	7

Top 10 Repeat Borrower

- **Description**

  Based on the findings, average number of books borrowed is roughly 3 to 4 books with the number of repeat borrowers of 529 from 650 users. This indicates that the majority of users are repeat borrowers. However, although the average borrowed books are 3 to 4 books, there are users who borrow exceeding that number, topping at 10 books by heidi74 followed closely by crystal38 at 9 books.

  With this information, we can consider to implement a loyalty program to encourage frequent borrowing and offer incentives for first-time users to borrow more frequently. We might also to start the donation links for the e-library maintenance needs, since it is a non profit e-library.

Question 4. Which library has the highest circulation rate, and what is the average book quantity available per library ?

- **SQL Query**

```sql
-- Library with the highest circulation rate
SELECT
	l.library_name,
	l.location,
	COUNT(lo.loan_id) AS circulation_rate
FROM
	libraries l
JOIN
	loans lo ON l.library_id = lo.library_id
GROUP BY
	1, 2
ORDER BY
	circulation_rate DESC;

-- Average quantity of books per library
SELECT
	l.library_name,
	l.location,
	AVG(b.quantity) AS avg_book_quantity
```

```
FROM
      libraries l
JOIN
      books b ON l.library_id = b.library_id
GROUP BY 1, 2;
```

- **Query Result**

	library_name	location	circulation_rate
1	Coleman, Rodriguez and Miller	Afghanistan	268
2	Rogers-Bird	Saudi Arabia	263
3	Brady Group	Rwanda	260
4	Hartman PLC	Algeria	255
5	Murphy Ltd	Anguilla	253
6	Conway-Henderson	Anguilla	248
7	Perez, Powell and Owens	Nicaragua	233
8	Mason Ltd	Antigua and Barbuda	220

Library's Circulation Rate

	library_name	location	avg_book_quantity
1	Hartman PLC	Algeria	5.8607594937
2	Perez, Powell and Owens	Nicaragua	5.5340909091
3	Mason Ltd	Antigua and Barbuda	6.1546391753
4	Conway-Henderson	Anguilla	5.41
5	Brady Group	Rwanda	5.6396396396
6	Rogers-Bird	Saudi Arabia	5.3979591837
7	Coleman, Rodriguez and Miller	Afghanistan	6.1904761905
8	Murphy Ltd	Anguilla	5.6116504854

Average Book Quantity in Libraries

- **Description**

  Some libraries like Coleman, Rogers-Bird, and Brady Group library have high circulation rate each at 268, 263, and 260. While some has lower circulation rate such as Mason Ltd library at 220. However, in terms of quantity, every library have an average of roughly about five to six or even seven books.

  Based on findings about circulation rate Afghanistan has the highest rate, this imply the high engagement rate in that country or in that server, since Afghanistan is the server location for the Coleman library. With information, we can consider to reallocate high-demand books to busy branches and promote low-circulation libraries with events or targeted marketing.

Question 5. How many books are overdue, and which libraries have the highest number of overdue loans ?

- **SQL Query**

```sql
-- Total count of overdue books
SELECT
	COUNT(*) AS overdue_books_count
FROM
	loans
WHERE
	return_date IS NULL AND due_date < CURRENT_DATE;

-- Libraries with the most overdue loans
SELECT
	l.library_name,
	l.location,
	COUNT(lo.loan_id) AS overdue_count
FROM
	libraries l
JOIN
	loans lo USING(library_id)
WHERE
	lo.return_date IS NULL AND lo.due_date < CURRENT_DATE
GROUP BY
	1, 2
ORDER BY
	overdue_count DESC;
```

- **Query Result**

	123 overdue_books_count
1	188

Overdue Books

	A-Z library_name	A-Z location	123 overdue_count
1	Rogers-Bird	Saudi Arabia	29
2	Murphy Ltd	Anguilla	28
3	Mason Ltd	Antigua and Barbuda	26
4	Brady Group	Rwanda	25
5	Coleman, Rodriguez and Miller	Afghanistan	23
6	Hartman PLC	Algeria	22
7	Perez, Powell and Owens	Nicaragua	18
8	Conway-Henderson	Anguilla	17

Overdue Counts in Libraries

- **Description**

  Rogers-Bird library has the highest overdue rates at 29 books, while Conway-Henderson library has the lowest overdue rates at 17 books. This suggest the difference in loan duration policies or reminder system.

Based on these findings we can consider to implement automated reminders for due dates and adjust loan duration policies based on overdue trends. We could also implement a more strict system such as limiting the loan duration or amount for those who have violate the terms of 14 days loan duration.

REFERENCES

- Mitchell, R. (2018). Web Scraping with Python (2nd ed.). O'Reilly Media.
- Towards Data Science. (2022, March 10). Coding and Implementing a Relational Database Using MySQL. Medium. Retrieved from https://towardsdatascience.com/coding-and-implementing-a-relational-database-using-mysql-d9bc69be90f5
- Towards Data Science. (2022, March 25). Designing a Relational Database and Creating an Entity Relationship Diagram. Medium. Retrieved from https://towardsdatascience.com/designing-a-relational-database-and-creating-an-entity-relationship-diagram-89c1c19320b2
- Books to Scrape. (n.d.). Retrieved October 31, 2023, from https://books.toscrape.com/