Exercise Week 8 (Mentoring 4)

SQL & Relational Database

By : Oktavio Reza Putra

A. PROJECT OBJECTIVES

The design of the E-Library Database is essential for the smooth operation of the library system, ensuring that data is managed efficiently, that resources are allocated correctly, and that users have a positive experience when interacting with the system. With its emphasis on user limits, inventory management, and operational efficiency, the system supports the long-term sustainability and growth of the library network while ensuring data integrity, security, and compliance with organizational policies.

Objective 1: **Improved Operational Efficiency**

A well-structured database ensures that all library operations, from managing book inventories to tracking user activities, are handled efficiently. This reduces errors and allows for better decision-making.

Objective 2: **Better User Experience**

The database design ensures that users have a smooth experience when interacting with the library system, such as easily borrowing and returning books, placing holds, and receiving notifications about due dates and available books.

Objective 3: **Resource Allocation and Management**

By maintaining accurate records of each book's availability across different libraries, the database helps libraries optimize their resources, ensuring that books are stocked where they are most needed and managing stock levels effectively.

Objective 4: **Scalability**

The database design allows for easy expansion, whether that's adding more libraries, increasing the number of books, or accommodating more users. It supports future growth without requiring a significant overhaul of the underlying architecture.

Objective 5: **Security and Data Integrity**

The database enforces security measures, such as unique email addresses for users and checks on borrow and hold limits, which protect both the users and the library from potential misuse or errors.

Objective 6: **Accurate Record Keeping**

The system provides a way to keep track of all interactions with library resources (e.g., books borrowed, books held, books available) in a consistent and transparent manner, ensuring that historical data is preserved and easily accessible.

Objective 7: **Compliance with Library Policies**

The database design enforces library policies, such as limiting the number of books a user can borrow or hold, tracking overdue books, and ensuring that users cannot perform actions outside their rights.

B. MISSION STATEMENT

The mission of the E-Library Database System is to provide a robust, efficient, and user-friendly platform for managing library operations, ensuring that users have seamless access to a wide range of books, while maintaining accurate and up-to-date records of borrowing, holding, and book availability. We aim to enhance the overall library experience by enabling streamlined management of library resources, empowering users to discover and engage with knowledge in an organized and sustainable way. Our system is committed to supporting libraries in providing equitable, accessible, and effective services, fostering lifelong learning, and promoting the value of reading and education within communities.

C. TABLE STRUCTURES

1. Identifying the Objects Needed for the Database

The database for the e-library system requires several entities to manage books, users, libraries, book availability, borrowing, and holds. Based on the requirements, here are the main objects (tables) that need to be created:

a. **Library Managers**
b. **Libraries**
c. **Books**
d. **Library Book List (Stock)** - Linking books to libraries with quantities
e. **Users (User Registration)**
f. **Books Borrowed (Loans)**
g. **Books on Hold**

2. Table Descriptions, Fields, and Keys

- **Library Managers Table**

```sql
-- Create Library_Managers table
create table if not exists library_managers (
    manager_id serial primary key,
    first_name varchar(50) not null,
    last_name varchar(50) not null,
    phone_number varchar(20) not null unique,
    email varchar(100) not null unique,
    hire_date date not null
);
```

1. **Description**: This table stores information about the managers in charge of libraries.
2. **Fields**: manager_id, first_name, last_name, phone_number, email, hire_date
3. **Primary Key**: manager_id
4. **Candidate Keys**: phone_number, email

- **Libraries Table**

```sql
-- Create Table : Libraries
create table if not exists libraries (
    library_id serial primary key,
    library_name varchar(100) not null unique,
    library_address varchar(255) not null,
    phone_number varchar(20),
    manager_id int not null check(manager_id > 0),
    library_email varchar(100) not null unique,
    library_website varchar(100),
    membership_fee decimal(10, 2) check(membership_fee >= 0),
    library_established_date date,
    constraint fk_manager_lib
        foreign key(manager_id) references library_managers(manager_id)
);
```

1. **Description**: This table stores information about the libraries.
2. **Fields**: library_id, library_name, library_address, phone_number, manager_id, library_email, library_website, membership_fee, library_established_date
3. **Primary Key**: library_id
4. **Candidate Keys**: library_name, library_email
5. **Foreign Key**: manager_id references library_managers(manager_id)

- **Books Table**

```sql
-- Create Table : Books
create table if not exists books (
    book_id serial primary key,
    title varchar(255) not null,
    author varchar(100) not null,
    category varchar(50) not null check(category in (
        'Self-Improvement',
        'Biography',
        'Fantasy',
        'Romance',
        'Science Fiction',
        'History',
        'Mystery',
        'Non-Fiction',
        'Children',
        'Horror'
    )),
    available_quantity int not null check(available_quantity >= 0), -- Total Book Quantity from all library that restore the Book
    published_date date,
    isbn varchar(20) unique
);
```

1. **Description**: This table stores the details of the books available in the library system and overall books quantity in all library.
2. **Fields**: book_id, title, author, category, available_quantity (total overall books quantity from quantity in each library), published_date, isbn
3. **Primary Key**: book_id
4. **Candidate Keys**: isbn
5. **Constraints**:
    - Category should be one of the predefined options (e.g., 'Self-Improvement', 'Biography', etc.).
    - available_quantity must be >= 0.

- **Library Book List Table**

```sql
-- Create Table Linking: Library Book Stock
-- to track book availability in each library
create table if not exists library_book_list (
    library_book_list_id serial primary key,
    library_id int,
    book_id int,
    library_book_quantity int check(library_book_quantity >= 0),
    constraint fk_library_id_list
        foreign key(library_id) references libraries(library_id),
    constraint fk_book_id_list
        foreign key(book_id) references books(book_id)
);
```

1. **Description**: This table keeps track of the availability of books in each library.
2. **Fields**: library_book_list_id, library_id, book_id, library_book_quantity
3. **Primary Key**: library_book_list_id
4. **Foreign Keys**:
    - library_id references libraries(library_id)
    - book_id references books(book_id)

- **User Registration Table**

```sql
-- Create Table : User Registration
create table if not exists user_registration(
    user_id serial primary key,
    first_name varchar(50) not null,
    last_name varchar(50) not null,
    email varchar(100) unique not null,
    registration_date date
);
```

1. **Description**: This table stores information about registered users.
2. **Fields**: user_id, first_name, last_name, email, registration_date
3. **Primary Key**: user_id
4. **Candidate Keys**: email

- **Books Borrowed (Loan) Table**

```sql
-- Create Table : Books Borrow (Loan)
create table if not exists books_borrow (
    book_borrow_id serial primary key,
    user_id int check(user_id > 0),
    library_book_list_id int check(library_book_list_id > 0),
    borrow_date date,
    borrow_book_quantity int not null check(borrow_book_quantity > 0),
    books_returned boolean default false, -- if user borrow 2 same books and only 1 returned books_returned is False
    due_date date not null check(due_date = borrow_date + interval '14 days'),
    return_date date,
    constraint fk_user_id_borrow
        foreign key(user_id) references user_registration(user_id),
    constraint fk_library_book_list_id_borrow
        foreign key(library_book_list_id) references library_book_list(library_book_list_id)
);
```

1. **Description**: This table keeps track of books borrowed by users and ensure that no more than 2 books can be borrowed on the same day .Users can return books earlier than the due date. Books will be automatically returned when they exceed the due date
2. **Fields**: book_borrow_id, user_id, library_book_list_id, borrow_date, borrow_book_quantity, books_returned, due_date, return_date
3. **Primary Key**: book_borrow_id
4. **Foreign Keys**:
    - user_id references user_registration(user_id
    - library_book_list_id reference library_book_list(library_book_list_id)
5. **Constraint :**
    - borrow_book_quantity must be greater than 0.
    - books_returned defaults to false.

- **Books on Hold Table**

```sql
-- Create Table : Books Holds
create table if not exists book_holds (
    book_hold_id serial primary key,
    user_id int check(user_id > 0),
    library_book_list_id int check(library_book_list_id > 0),
    book_hold_quantity int not null check(book_hold_quantity > 0),
    hold_date date,
    expiry_date date not null check(expiry_date = hold_date + interval '7 days'),
    is_active boolean default true,
    constraint fk_user_id_holds
        foreign key(user_id) references user_registration(user_id),
    constraint fk_library_book_list_id_holds
        foreign key(library_book_list_id) references library_book_list(library_book_list_id)
);
```

1. **Description**: This table stores information about books users have placed on hold and Ensure that users can only hold up to 2 books on the same day.
2. **Fields**: book_hold_id, user_id, library_book_list_id, book_hold_quantity, hold_date, expiry_date, is_active
3. **Primary Key**: book_hold_id
4. **Foreign Keys**:
    - user_id references user_registration(user_id)
    - library_book_list_id references library_book_list(library_book_list_id)
- **Constraints**:
    - book_hold_quantity must be greater than 0.
    - is_active is set to true by default.

D. TABLE RELATIONSHIP

1. One to Many Relationship

- Library Managers to Libraries: A manager can oversee multiple libraries, but each library has only one manager.

1. Relationship: One-to-Many (library_managers.manager_id → libraries.manager_id)

- Libraries to Library Book List: A library can have many books in stock, but each book stock record belongs to only one library.
    1. Relationship: One-to-Many (libraries.library_id → library_book_list.library_id)
- Books to Library Book List: A book can be available in multiple libraries, but each library records its stock of that book.
    1. Relationship: One-to-Many (books.book_id → library_book_list.book_id)
- Library Book List to Books Borrowed (Loan): A Library Book List ID can be available in multiple Book Borrows
    1. Relationship: One-to-Many (library_book_list.library_book_list_id → books_borrow.library_book_list_id)
- User Registration to Books Borrowed (Loan): A user can borrow multiple books, but each borrow record is associated with a single user.
    1. Relationship: One-to-Many (user_registration.user_id → books_borrow.user_id)
- Library Book List to Book Holds (Loan): A Library Book List ID can be available in multiple Book Holds
    1. Relationship: One-to-Many (library_book_list.library_book_list_id → book_holds.library_book_list_id)
- User Registration to Book Holds: A user can place multiple books on hold, but each hold record is associated with a single user.
    1. Relationship: One-to-Many (user_registration.user_id → book_holds.user_id)

E. BUSINESS RULES AND CONSTRAINT

1. Library Managers Table:
    - Business Rules:
        1. The manager must have a unique phone number and email for identification and communication.
        2. The hire date must be specified, ensuring the manager's tenure in the system is recorded.

    - Constraints:
        1. manager_id is the Primary Key, automatically unique and indexed.
        2. phone_number is unique and NOT NULL to ensure no duplicates.
        3. email is unique and NOT NULL to ensure managers can be contacted uniquely.
        4. hire_date is NOT NULL to ensure that the hire date is provided.

2. Libraries Table:
    - Business Rules:
        1. Each library must have a unique name and address for identification.
        2. The manager_id should reference an existing manager from the library_managers table, meaning that each library is managed by one manager.

3. The membership fee, if applicable, must be a non-negative value.

- Constraints:
    1. library_id is the Primary Key, ensuring each library is uniquely identified.
    2. library_name and library_address are NOT NULL for proper identification.
    3. manager_id is a Foreign Key referencing library_managers(manager_id), establishing a one-to-many relationship between library_managers and libraries.
    4. library_email is unique to ensure each library has a distinct email.
    5. membership_fee is checked to ensure it's non-negative (i.e., CHECK(membership_fee >= 0)).

3. Books Table:
    - Business Rules:
        1. Books must have a title, author, and category.
        2. The available quantity of each book should be non-negative and represent the total available stock in all libraries.
        3. The ISBN should be unique for each book to avoid duplicates.

    - Constraints:
        1. book_id is the Primary Key, ensuring unique identification of each book.
        2. title, author, and category are NOT NULL to ensure the book has these essential attributes.
        3. category is CHECKED to ensure it falls within valid values such as 'Self-Improvement', 'Biography', etc.
        4. available_quantity is CHECKED to ensure it's non-negative (i.e., CHECK(available_quantity >= 0)).
        5. isbn is unique to avoid duplication of books based on ISBN.

4. Library Book List Table:
    - Business Rules:
        1. Each library should have a specific stock of books that could be checked out or reserved.
        2. The quantity of books in each library must be tracked and cannot be negative.

    - Constraints:
        1. library_book_list_id is the Primary Key, ensuring each entry in the library book list is unique.
        2. library_id is a Foreign Key referencing libraries(library_id), establishing a many-to-one relationship (many books in one library).
        3. book_id is a Foreign Key referencing books(book_id), linking each entry to a specific book.
        4. library_book_quantity is CHECKED to ensure it’s non-negative (i.e., CHECK(library_book_quantity >= 0)).

5. User Registration Table:
    - Business Rules:
        1. Each library should have a specific stock of books that could be checked out or reserved.
        2. The quantity of books in each library must be tracked and cannot be negative.

    - Constraints:
        1. user_id is the Primary Key, ensuring each user is uniquely identified.
        2. first_name, last_name, and email are NOT NULL for proper registration.
        3. email is unique to avoid multiple registrations with the same email address.
        4. registration_date is date type.
6. Books Borrowed Table:
    - Business Rules:
        1. Users must borrow books from a specific library and track the quantity borrowed.
        2. A borrow record must include the borrow date and due date.
        3. A user can only place a limited number of borrows.

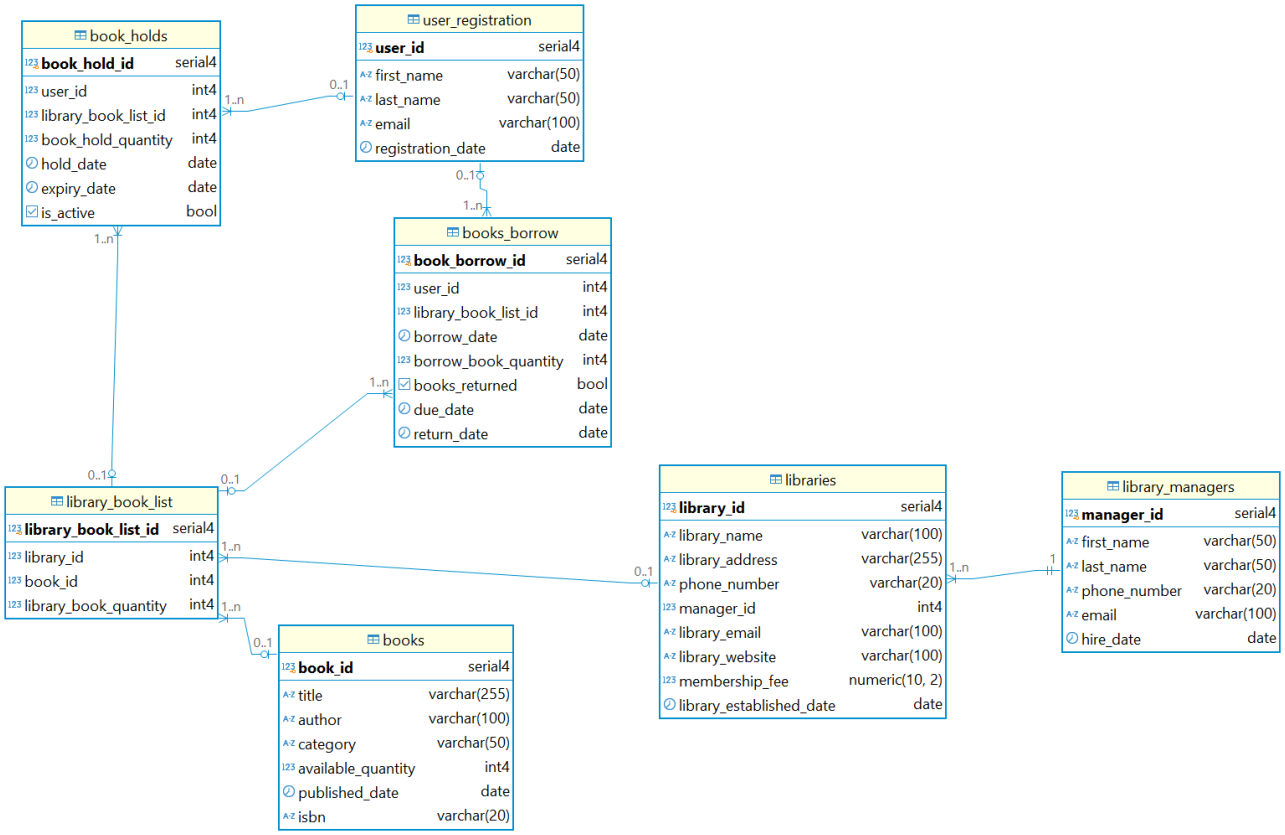
    - Constraints:
        1. book_borrow_id is the Primary Key, ensuring each borrow entry is unique.
        2. user_id is a Foreign Key referencing user_registration(user_id), creating a many-to-one relationship between users and borrowed books.
        3. library_book_list_id is a Foreign Key referencing library_book_list(library_book_list_id), linking each borrow entry to a specific library book.
        4. borrow_book_quantity is CHECKED to ensure it's a positive value.
        5. due_date is CHECKED to ensure it is 14 days after the borrow date (via CHECK(due_date = borrow_date + interval '14 days')).
        6. books_returned is a boolean indicating whether the borrowed book(s) have been returned.

7. Books Hold Table:
    - Business Rules:
        1. Users can place a hold on books, but the hold must expire after a certain period (7 days).
        2. A user can only place a limited number of holds.

    - Constraints:
        1. book_hold_id is the Primary Key, ensuring each hold is unique.
        2. user_id is a Foreign Key referencing user_registration(user_id), establishing a many-to-one relationship.
        3. library_book_list_id is a Foreign Key referencing library_book_list(library_book_list_id), linking the hold to a specific book in a specific library.

4. book_hold_quantity is CHECKED to ensure it's positive.
5. expiry_date is CHECKED to ensure it is exactly 7 days after the hold date (via CHECK(expiry_date = hold_date + INTERVAL '7 days')).

F. IMPLEMENTING THE DESIGN



Implementing ERD Process

1. **Define Database Schema, Business Rules, Constraint and Table Structures**
   - **a.** Create Schemas: If needed, create a schema to group related tables and objects in the database (e.g., CREATE SCHEMA my_schema;).
   - **b.** Define Tables: For each entity in the ERD, create a table. The table structure should include:
     - Table Name: Should match the entity in the ERD.
     - Columns in the table corresponding to the entity's attributes in the ERD.
     - Data Types: Assign appropriate data types (e.g., VARCHAR, INT, DATE, BOOLEAN).
     - Constraints: Define constraints to enforce rules on the data (e.g., NOT NULL, UNIQUE, CHECK).

- Define a primary key for each table, typically based on the entity's unique identifier in the ERD.
- Define Foreign Key that have relation in other table

2. **Define Relationships Between Tables**
   a. One-to-Many Relationships: This is the most common relationship where the primary key of one table (the "one" side) is referenced as a foreign key in another table (the "many" side).

      Example: A library can have many books, so the books table would have a foreign key referencing the libraries table.

3. **Create Dummy Data for each Tables (Populating the Database Step 1)**
   Create Dummy Data using a combination of Python and Excel where Dummy Data is created using Python and Data Saved into PostgreSQL Folder 17 in CSV format

4. **Input Dummy Data into each Tables (Populating The Database Step 2)**
   Input Dummy Data on each Table from CSV Files that have been created and saved using Python to PostgreSQL using COPY

G. CREATE DUMMY DATA AND INPUT THE DATA INTO AVAILABLE TABLES

1. **Create Dummy Tables**

- **Import Libraries in Python**

```python
import pandas as pd
import random
from datetime import datetime, timedelta
```

- **Create Dummy Data for Library Managers Table**

```python
# Function to generate a random date within a range
def random_date(start_date, end_date):
    return start_date + timedelta(days=random.randint(0, (end_date - start_date).days))

# Set the start and end date for the range (e.g., between 5 to 10 years ago)
start_date = datetime(2013, 1, 1).date()  # Start date: January 1, 2013
end_date = datetime(2018, 12, 31).date()  # End date: December 31, 2018

# Generate dummy data for the Library_Managers table
library_managers_data = {
    'manager_id': range(1, 4),
    'first_name': [f'ManagerFirstName{i}' for i in range(1, 4)],
    'last_name': [f'ManagerLastName{i}' for i in range(1, 4)],
    'phone_number': [f'0274-1234{i}' for i in range(1, 4)],
    'email': [f'manager{i}@library.com' for i in range(1, 4)],
    'hire_date': [random_date(start_date, end_date) for _ in range(3)]  # Random hire_date
}

# Create the DataFrame
library_managers_df = pd.DataFrame(library_managers_data)

# Display the DataFrame
library_managers_df
```

	manager_id	first_name	last_name	phone_number	email	hire_date
0	1	ManagerFirstName1	ManagerLastName1	0274-12341	manager1@library.com	2013-06-10
1	2	ManagerFirstName2	ManagerLastName2	0274-12342	manager2@library.com	2017-04-12
2	3	ManagerFirstName3	ManagerLastName3	0274-12343	manager3@library.com	2018-10-19

- **Create Dummy Data for Library Data**

```python
# Generate dummy data for the Libraries table
libraries_data = {
    'library_id': range(1, 4),
    'library_name': ['Perpustakaan Pintar', 'Perpustakaan Cerdas', 'Perpustakaan Logika'],
    'library_address': ['Jl. Beo No 230, Bantul, Yogyakarta', 'Jl. Markisa No 240, Bantul, Yogyakarta', 'Jl. Apel No 235,
    'phone_number': ['0274-88032', '0274-90902', '0274-98783'],
    'manager_id': [1, 2, 3],
    'library_email': [f'{name.lower().replace(" ", "")}@library.com' for name in ['Perpustakaan Pintar', 'Perpustakaan Cer
    'library_website': [f'www.{name.lower().replace(" ", "")}.com' for name in ['Perpustakaan Pintar', 'Perpustakaan Cerda
    'membership_fee': [random.choice(range(10, 21)) * 1000 for _ in range(3)],  # Rounded thousands
    'library_established_date': [
        (datetime.now() - timedelta(days=random.randint(365*10, 365*50))).strftime('%Y-%m-%d')
        for _ in range(3)
    ]
}

# Create the DataFrame
libraries_df = pd.DataFrame(libraries_data)

# Display the DataFrame
libraries_df
```

	library_id	library_name	library_address	phone_number	manager_id	library_email	library_website	membership_fee	l
0	1	Perpustakaan Pintar	Jl. Beo No 230, Bantul, Yogyakarta	0274-88032	1	perpustakaanpintar@library.com	www.perpustakaanpintar.com	19000	
1	2	Perpustakaan Cerdas	Jl. Markisa No 240, Bantul, Yogyakarta	0274-90902	2	perpustakaancerdas@library.com	www.perpustakaancerdas.com	13000	
2	3	Perpustakaan Logika	Jl. Apel No 235, Sleman, Yogyakarta	0274-98783	3	perpustakaanlogika@library.com	www.perpustakaanlogika.com	14000	

- **Create Dummy Data for Library Book List Table**

```python
# Generate dummy data for the Library_Book_List table
library_book_list_data = {
    'library_book_list_id': range(1, 31),
    'library_id': random.choices([1, 2, 3], k=30),
    'book_id': random.choices(range(1, 11), k=30),
    'library_book_quantity': random.choices(range(20, 40), k=30)
}
library_book_list_df = pd.DataFrame(library_book_list_data)

# Group by book_id and library_id to show quantities for each library and book combination
book_totals = library_book_list_df.groupby(['library_book_list_id','library_id', 'book_id'])['library_book_quantity'].s
book_totals
```

2	3	3	8	29
3	4	3	1	28
4	5	1	5	23
5	6	3	3	34
6	7	3	2	28
7	8	2	1	25
8	9	1	7	27
9	10	3	7	36
10	11	3	6	37
11	12	3	10	29
12	13	1	1	38
13	14	3	2	20

```python
# Create Book Groups to calculate total number of books in each libraries
book_groups = book_totals.groupby('book_id')['library_book_quantity'].sum().reset_index()
book_groups
```

	book_id	library_book_quantity
0	1	91
1	2	159
2	3	159
3	4	33
4	5	77
5	6	119
6	7	99
7	8	58
8	9	60
9	10	29

- **Create Dummy Data for Books Table**

```python
# Generate dummy data for the Books table
# Function to generate a random date within a range
def random_date(start_date, end_date):
    return start_date + timedelta(days=random.randint(0, (end_date - start_date).days))

# Set the start and end date for the range (e.g., between 1 to 20 years ago)
start_date = datetime(2003, 1, 1).date()  # Start date: January 1, 2003
end_date = datetime(2023, 1, 1).date()  # End date: January 1, 2023

books_data = {
    'book_id': range(1, 11),
    'title': [f'Book Title {i}' for i in range(1, 11)],
    'author': [f'Author {i}' for i in range(1, 11)],
    'category': random.choices(
        ['Self-Improvement', 'Biography', 'Fantasy', 'Romance', 'Science Fiction', 'History', 'Mystery', 'Non-Fiction',
    ),
    'published_date': [random_date(start_date, end_date) for _ in range(10)],  # Random published_date,
    'isbn': [f'ISBN{i:013}' for i in range(1, 11)]
}
books_df = pd.DataFrame(books_data)

# Merge the total available quantities into the books data
books_df = books_df.merge(book_groups, on='book_id', how='left')
books_df['library_book_quantity'].fillna(0, inplace=True)
books_df.rename(columns={'library_book_quantity': 'available_quantity'}, inplace=True)
books_df['available_quantity'] = books_df['available_quantity'].astype(int)

# Display the DataFrame
books_df = books_df[['book_id', 'title', 'author', 'category','available_quantity','published_date','isbn']]
books_df
```

	book_id	title	author	category	available_quantity	published_date	isbn
0	1	Book Title 1	Author 1	Biography	91	2017-01-17	ISBN0000000000001
1	2	Book Title 2	Author 2	Mystery	159	2018-06-06	ISBN0000000000002
2	3	Book Title 3	Author 3	Self-Improvement	159	2013-07-22	ISBN0000000000003
3	4	Book Title 4	Author 4	Biography	33	2017-12-22	ISBN0000000000004
4	5	Book Title 5	Author 5	Non-Fiction	77	2012-12-21	ISBN0000000000005
5	6	Book Title 6	Author 6	Fantasy	119	2012-04-20	ISBN0000000000006
6	7	Book Title 7	Author 7	Horror	99	2008-04-23	ISBN0000000000007
7	8	Book Title 8	Author 8	Horror	58	2004-07-27	ISBN0000000000008
8	9	Book Title 9	Author 9	History	60	2005-09-06	ISBN0000000000009
9	10	Book Title 10	Author 10	History	29	2018-11-08	ISBN0000000000010

- **Create Dummy Data for User Registration**

```python
# Define the date range for registration_date (before 1 January 2023)
start_date = datetime(2021, 1, 1)
end_date = datetime(2022, 12, 31)

# Generate a list of random dates within the specified range
date_range = pd.date_range(start=start_date, end=end_date).to_list()

# Generate dummy data for the User_Registration table
users_data = {
    'user_id': range(1, 21),
    'first_name': [f'UserFirstName{i}' for i in range(1, 21)],
    'last_name': [f'UserLastName{i}' for i in range(1, 21)],
    'email': [f'user{i}@example.com' for i in range(1, 21)],
    'registration_date': random.sample(date_range, k=20)  # Sample random dates from the date_range
}

# Create the DataFrame
users_df = pd.DataFrame(users_data)

# Display the DataFrame
users_df
```

	user_id	first_name	last_name	email	registration_date
0	1	UserFirstName1	UserLastName1	user1@example.com	2022-08-02
1	2	UserFirstName2	UserLastName2	user2@example.com	2022-12-24
2	3	UserFirstName3	UserLastName3	user3@example.com	2021-11-06
3	4	UserFirstName4	UserLastName4	user4@example.com	2021-02-08
4	5	UserFirstName5	UserLastName5	user5@example.com	2022-02-15
5	6	UserFirstName6	UserLastName6	user6@example.com	2021-09-12
6	7	UserFirstName7	UserLastName7	user7@example.com	2021-11-14
7	8	UserFirstName8	UserLastName8	user8@example.com	2022-06-17
8	9	UserFirstName9	UserLastName9	user9@example.com	2021-11-05
9	10	UserFirstName10	UserLastName10	user10@example.com	2022-05-05
10	11	UserFirstName11	UserLastName11	user11@example.com	2022-03-09
11	12	UserFirstName12	UserLastName12	user12@example.com	2021-03-04
12	13	UserFirstName13	UserLastName13	user13@example.com	2021-02-25
13	14	UserFirstName14	UserLastName14	user14@example.com	2022-09-26
14	15	UserFirstName15	UserLastName15	user15@example.com	2022-07-19
15	16	UserFirstName16	UserLastName16	user16@example.com	2021-10-29
16	17	UserFirstName17	UserLastName17	user17@example.com	2022-05-01
17	18	UserFirstName18	UserLastName18	user18@example.com	2022-12-30
18	19	UserFirstName19	UserLastName19	user19@example.com	2021-09-09
19	20	UserFirstName20	UserLastName20	user20@example.com	2021-06-22

- **Create Dummy Data for Books Borrow**

```python
# Generate a list of random dates within a certain period (e.g., last year)
start_date = datetime(2023, 1, 1)
end_date = datetime(2024, 1, 1)
date_range = pd.date_range(start=start_date, end=end_date).to_list()

# Generate dummy data for the Books_Borrow table
books_borrow_data = {
    'book_borrow_id': range(1, 21),
    'user_id': random.choices(range(1, 21), k=20),
    'library_book_list_id': random.choices(range(1, 21), k=20),
    'borrow_date': random.sample(date_range, k=20),  # Sample random dates from the date_range
    'borrow_book_quantity': random.choices(range(1, 3), k=20),
    'books_returned': [True if random.random() > 0.5 else False for _ in range(20)]
}

# Create the DataFrame
books_borrow_df = pd.DataFrame(books_borrow_data)

# Sort the borrow_date column in ascending order
books_borrow_df['borrow_date'] = books_borrow_df['borrow_date'].sort_values().values

# Set the due_date to 14 days after the borrow_date
books_borrow_df['due_date'] = books_borrow_df['borrow_date'] + timedelta(days=14)

# Set the return_date randomly or as None
books_borrow_df['return_date'] = [
    None if random.random() > 0.5 else row['borrow_date'] + timedelta(days=random.randint(1, 14))
    for _, row in books_borrow_df.iterrows()
]

# Display the DataFrame
books_borrow_df
```

	book_borrow_id	user_id	library_book_list_id	borrow_date	borrow_book_quantity	books_returned	due_date	return_date
0	1	2	18	2023-01-13	2	True	2023-01-27	NaT
1	2	5	12	2023-02-05	2	True	2023-02-19	NaT
2	3	3	18	2023-02-23	1	False	2023-03-09	2023-03-05
3	4	12	6	2023-03-04	2	False	2023-03-18	2023-03-13
4	5	12	12	2023-03-06	1	True	2023-03-20	2023-03-13
5	6	15	19	2023-04-05	1	False	2023-04-19	NaT
6	7	8	16	2023-05-22	2	False	2023-06-05	2023-06-01
7	8	4	3	2023-05-27	1	True	2023-06-10	NaT
8	9	6	15	2023-06-13	2	False	2023-06-27	2023-06-27
9	10	19	18	2023-07-04	2	True	2023-07-18	2023-07-16
10	11	10	4	2023-07-14	2	True	2023-07-28	NaT
11	12	1	8	2023-07-15	1	True	2023-07-29	NaT
12	13	14	5	2023-07-19	1	False	2023-08-02	2023-07-22
13	14	14	16	2023-09-02	1	True	2023-09-16	2023-09-09
14	15	13	4	2023-09-15	2	True	2023-09-29	NaT

- **Create Dummy Data for Book Holds**

```python
# Set a fixed end date for the random generation
end_date = datetime(2024, 1, 1)

# Generate random hold_date within the last 14 days, but order them from earliest to latest
book_holds_data = {
    'book_hold_id': range(1, 21),
    'user_id': random.choices(range(1, 21), k=20),
    'library_book_list_id': random.choices(range(1, 21), k=20),
    'book_hold_quantity': random.choices(range(1, 3), k=20),
    'hold_date': sorted([end_date - timedelta(days=random.randint(1, 14)) for _ in range(20)])  # Sort the random dates
}

# Create the DataFrame
book_holds_df = pd.DataFrame(book_holds_data)

# Set expiry_date to 7 days after the hold_date
book_holds_df['expiry_date'] = book_holds_df['hold_date'] + timedelta(days=7)

# Set is_active randomly (with 70% chance being True)
book_holds_df['is_active'] = [True if random.random() > 0.3 else False for _ in range(20)]

# Display the DataFrame
book_holds_df
```

	book_hold_id	user_id	library_book_list_id	book_hold_quantity	hold_date	expiry_date	is_active
0	1	3	12	1	2023-12-18	2023-12-25	True
1	2	14	18	1	2023-12-20	2023-12-27	False
2	3	19	5	1	2023-12-20	2023-12-27	True
3	4	2	1	2	2023-12-21	2023-12-28	True
4	5	15	4	1	2023-12-22	2023-12-29	True
5	6	7	10	1	2023-12-22	2023-12-29	True
6	7	3	10	1	2023-12-23	2023-12-30	True
7	8	8	5	1	2023-12-24	2023-12-31	True
8	9	1	6	2	2023-12-26	2024-01-02	True
9	10	9	11	1	2023-12-26	2024-01-02	True
10	11	10	3	2	2023-12-26	2024-01-02	True

- **Save the Data into CSV Format**

```
#Save to CSV format
library_managers_df.to_csv(r'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\library_managers.
libraries_df.to_csv(r'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\libraries.csv', index=Fa
users_df.to_csv(r'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\user_registration.csv', inde

books_df.to_csv(r'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\books_df.csv', index=False)
book_totals.to_csv(r'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\library_book_list.csv', i
books_borrow_df.to_csv(r'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\books_borrow.csv' ,in
book_holds_df.to_csv(r'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\book_holds.csv', index=
```

2. **Input Data into Available Tables**
   - **Input Dummy Data into Library Managers Table**

```
-- Input Library Managers Table
copy library_managers(manager_id, first_name, last_name, phone_number, email, hire_date)
from 'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\library_managers.csv'
delimiter ','
csv header;

select * from library_managers;
```

manager_id	first_name	last_name	phone_number	email	hire_date
1	ManagerFirstName1	ManagerLastName1	0274-12341	manager1@library.com	2013-06-10
2	ManagerFirstName2	ManagerLastName2	0274-12342	manager2@library.com	2017-04-12
3	ManagerFirstName3	ManagerLastName3	0274-12343	manager3@library.com	2018-10-19

   - **Input Dummy Data into Libraries Table**

```
-- Input Libraries Table
copy libraries(library_id, library_name, library_address, phone_number, manager_id, library_email,
               library_website, membership_fee, library_established_date)
from 'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\libraries.csv'
delimiter ','
csv header;

select * from libraries;
```

library_id	library_name	library_address	phone_number	manager_id	library_email	library_website
1	Perpustakaan Pintar	Jl. Beo No 230, Bantul, Yogyakarta	0274-88032	1	perpustakaanpintar@library.com	www.perpustakaanpintar.com
2	Perpustakaan Cerdas	Jl. Markisa No 240, Bantul, Yogyakarta	0274-90902	2	perpustakaancerdas@library.com	www.perpustakaancerdas.com
3	Perpustakaan Logika	Jl. Apel No 235, Sleman, Yogyakarta	0274-98783	3	perpustakaanlogika@library.com	www.perpustakaanlogika.com

   - **Input Dummy Data into Library Book List Table**

```
-- Input Library Book List Table
copy library_book_list(library_book_list_id, library_id, book_id, library_book_quantity)
from 'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\library_book_list.csv'
delimiter ','
csv header;

select * from library_book_list;
```

library_book_list_id	library_id	book_id	library_book_quantity
1	2	2	31
2	1	6	34
3	3	8	29
4	3	1	28
5	1	5	23
6	3	3	34
7	3	2	28
8	2	1	25
9	1	7	27
10	3	7	36
11	3	6	37
12	3	10	29
13	1	1	38
14	3	2	20
15	1	3	33
16	3	6	27
17	1	5	30
18	3	4	33
19	2	2	22
20	3	7	36

- **Input Dummy Data into Books Table**

```
-- Input Books Table
copy books(book_id, title, author, category, available_quantity, published_date, isbn)
from 'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\books_df.csv'
delimiter ','
csv header;

select * from books;
```

book_id	title	author	category	available_quantity	published_date	isbn
1	Book Title 1	Author 1	Biography	91	2017-01-17	ISBN0000000000001
2	Book Title 2	Author 2	Mystery	159	2018-06-06	ISBN0000000000002
3	Book Title 3	Author 3	Self-Improvement	159	2013-07-22	ISBN0000000000003
4	Book Title 4	Author 4	Biography	33	2017-12-22	ISBN0000000000004
5	Book Title 5	Author 5	Non-Fiction	77	2012-12-21	ISBN0000000000005
6	Book Title 6	Author 6	Fantasy	119	2012-04-20	ISBN0000000000006

- **Input Dummy Data into User Registration Table**

```
-- Input User Registration Table
copy user_registration(user_id, first_name, last_name, email, registration_date)
from 'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\user_registration.csv'
delimiter ','
csv header;

select * from user_registration;
```

user_id	first_name	last_name	email	registration_date
1	UserFirstName1	UserLastName1	user1@example.com	2022-08-02
2	UserFirstName2	UserLastName2	user2@example.com	2022-12-24
3	UserFirstName3	UserLastName3	user3@example.com	2021-11-06
4	UserFirstName4	UserLastName4	user4@example.com	2021-02-08
5	UserFirstName5	UserLastName5	user5@example.com	2022-02-15
6	UserFirstName6	UserLastName6	user6@example.com	2021-09-12

- **Input Dummy Data into Books Borrow Table**

```
-- Input Books Borrow Table
copy books_borrow(book_borrow_id, user_id, library_book_list_id, borrow_date, borrow_book_quantity, books_returned, due_date, return_date)
from 'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\books_borrow.csv'
delimiter ','
csv header;

select * from books_borrow;
```

k_borrow_id	user_id	library_book_list_id	borrow_date	borrow_book_quantity	books_returned	due_date	return_date
1	2	18	2023-01-13	2	[v]	2023-01-27	[NULL]
2	5	12	2023-02-05	2	[v]	2023-02-19	[NULL]
3	3	18	2023-02-23	1	[ ]	2023-03-09	2023-03-05
4	12	6	2023-03-04	2	[ ]	2023-03-18	2023-03-13
5	12	12	2023-03-06	1	[v]	2023-03-20	2023-03-13

- **Input Dummy Data into Book Holds Table**

```
-- Input Books Hold Table
copy book_holds(book_hold_id, user_id, library_book_list_id, book_hold_quantity, hold_date, expiry_date, is_active)
from 'C:\Program Files\PostgreSQL\17\data\File Exercise Week 8 Pacmann Academy\book_holds.csv'
delimiter ','
csv header;

select * from book_holds;
```

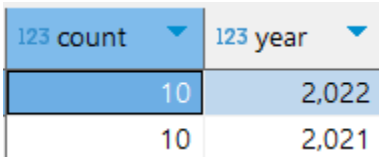
book_hold_id	user_id	library_book_list_id	book_hold_quantity	hold_date	expiry_date	is_active
1	3	12	1	2023-12-18	2023-12-25	[v]
2	14	18	1	2023-12-20	2023-12-27	[ ]
3	19	5	1	2023-12-20	2023-12-27	[v]
4	2	1	2	2023-12-21	2023-12-28	[v]
5	15	4	1	2023-12-22	2023-12-29	[v]
6	7	10	1	2023-12-22	2023-12-29	[v]

H. BUSINESS QUESTION

1. **Number of User in each Year**

   Determine how many unique users registered or used the library services each year. This helps track the growth of the user base over time and provides insight into trends, such as peak periods of new user acquisition or user retention.

```
select
    count(user_id),
    extract(year from registration_date) as year
from
    user_registration
group by
    year;
```



123 count	123 year
10	2,022
10	2,021

   - Analysis: Each year there are only 10 users each who register. In 2023 there were no users who registered due to maintenance on the E-Library.
   - Recommendations:
     1. Provide new users with free e-book downloads or access to popular titles upon registration.
     2. Create a reward system where users earn points for activities like borrowing books or referring others, which can be redeemed for services or physical rewards.
     3. Collaborate with educational institutions to make e-library access part of their curriculum or offer student and teacher accounts.
     4. Partner with local businesses to offer employees library memberships as part of their benefits.
     5. Collect feedback on what features or content would attract new users and keep current ones engaged.

2. **Users with Most Active Holds**

Identify which users have the highest number of active book holds. This can inform the library about its most engaged users and help in recognizing users who may need special attention or promotions. It also assists in understanding the demand and user behavior for reserved books.

```sql
-- Case 2: Users with Most Active Holds
select
    t1.user_id,
    user_name,
    email,
    cnt_hold,
    qty_book_hold
from
    (select
        user_id,
        count(book_hold_id) as cnt_hold,
        sum(book_hold_quantity) as qty_book_hold
    from
        book_holds
    group by
        user_id
    order by
        qty_book_hold desc) t1
join
    (
        select
            user_id,
            concat(first_name,'',last_name) as user_name,
            email
        from
            user_registration
    ) t2
on
    t1.user_id = t2.user_id
limit 5;
```

user_id	user_name	email	cnt_hold	qty_book_hold
15	UserFirstName15UserLastName15	user15@example.com	3	4
2	UserFirstName2UserLastName2	user2@example.com	2	4
1	UserFirstName1UserLastName1	user1@example.com	2	3
18	UserFirstName18UserLastName18	user18@example.com	1	2
10	UserFirstName10UserLastName10	user10@example.com	1	2

- Analysis: Most Borrowed Holds only 4 Books during 2023
- Recommendations:
    1. Diversify the book selection to cater to a wider range of interests. Include trending genres, new releases, or books from popular authors to attract different user segments.
    2. Promote best-sellers and books with high demand as "top holds" on your platform. These books will have a natural appeal and can inspire users to place holds quickly.
    3. Introduce a loyalty or reward system where users earn points for each book they place on hold. These points can be redeemed for rewards, such as extended borrowing periods, free event tickets, or exclusive library access.

3. **Users That Most Frequently Borrowed Books**

Find out which usershave been borrowed the most times.

```sql
-- Case 3:Most Frequently Borrowed Books
select
    t1.user_id,
    user_name,
    email,
    cnt_borrow,
    qty_book_borrow
from
    (select
        user_id,
        count(book_borrow_id) as cnt_borrow,
        sum(borrow_book_quantity) as qty_book_borrow
    from
        books_borrow
    group by
        user_id
    order by
        qty_book_borrow desc) t1
join
    (
        select
            user_id,
            concat(first_name,'',last_name) as user_name,
            email
        from
            user_registration
    ) t2
on
    t1.user_id = t2.user_id
limit 5;
```

user_id	user_name	email	cnt_borrow	qty_book_borrow
12	UserFirstName12UserLastName12	user12@example.com	4	6
2	UserFirstName2UserLastName2	user2@example.com	2	3
13	UserFirstName13UserLastName13	user13@example.com	2	3
10	UserFirstName10UserLastName10	user10@example.com	1	2
6	UserFirstName6UserLastName6	user6@example.com	1	2

- Analysis: The most books borrowed is only 6 books and the most books borrowed is only 4 times during 2023.
- Recommendations:
    1. Add books from diverse genres or topics that cater to different interests (e.g., sci-fi, self-help, history, technology, etc.). Introduce collections on trending topics or global issues like sustainability, AI, and mental health
    2. Make sure to acquire popular new releases or upcoming titles in various genres, ensuring that the collection stays fresh and relevant to users' evolving preferences.
    3. Create monthly or quarterly themes, such as "Science Fiction Month" or "Mindfulness and Well-being Week", and promote specific books that align with these themes. Offer discounted or free access to books related to the theme to encourage borrowing.
    4. Introduce gamified elements like badges or leaderboards for the most active borrowers or for completing specific reading challenges. This can motivate users to hold more books to achieve rewards.

4. **Identify First Borrow and Last Borrow Books by Users**

Determine the first and last instances of book borrowing by each user. This provides insights into user engagement, retention, and borrowing patterns over time. It can also help identify long-term users and track the lifecycle of a user's activity within the library.

```
with
    final_borrow
as
    (
        with
            first_last_borrow
        as
            (
                select
                    user_id,
                    first_value(borrow_date) over(partition by user_id order by borrow_date asc) as first_borrow_date,
                    first_value(borrow_date) over(partition by user_id order by borrow_date desc) as last_borrow_date
                from
                    books_borrow
            )
    select
        user_id,
        first_borrow_date,
        last_borrow_date,
        last_borrow_date - first_borrow_date as borrow_first_last_day_diff
    from
        first_last_borrow
    )
select
    distinct user_id,
    first_borrow_date,
    last_borrow_date,
    borrow_first_last_day_diff
from
    final_borrow
where
    borrow_first_last_day_diff > 0
order by
    borrow_first_last_day_diff desc;
```

user_id	first_borrow_date	last_borrow_date	borrow_first_last_day_diff
2	2023-01-13	2023-11-17	308
12	2023-03-04	2023-12-02	273
13	2023-09-15	2023-12-29	105
14	2023-07-19	2023-09-02	45

- Analysis: The maximum book loan duration is 308 days. This shows that the user borrowed the first book and borrowed the book again in the next book borrowing is 308 days.
- Recommendations:
    1. Introduce a “Frequent Borrower” Reward Program
    2. Use users' borrowing history to recommend books in genres or topics they frequently engage with. Personalized suggestions can trigger interest and prompt quick returns and re-borrowing.
    3. For frequently borrowed books, allow users to borrow them for a short, predefined period (e.g., 5-7 days). This ensures that users don’t hold onto books for too long and encourages faster borrowing cycles.
    4. If someone reserves a popular book, prioritize them after the previous borrower returns it. Make it clear to the current borrower that they should return it soon to allow others to borrow.

5. **Most Borrowed Books in the Library**

Identify which books have been borrowed the most across all users and libraries. This helps in understanding overall book demand and usage. Libraries can use this information to prioritize stocking, promote certain books, or plan for replacements if popular books are worn out or damaged.

```sql
with
    most_borrow_books
as
    (
        select
            *
        from
            (
                select
                    library_book_list_id,
                    sum(borrow_book_quantity) as borrows_qty
                from
                    books_borrow
                group by
                    library_book_list_id
            ) t1
        join
            (
                select
                    library_book_list_id,
                    book_id
                from
                    library_book_list
            ) t2
        using
            (library_book_list_id)
        order by
            borrows_qty desc
    )
select
    book_id,
    title,
    author,
    category,
    borrows_qty
from
    most_borrow_books mbs
join
    books b
using
    (book_id);
```

book_id	title	author	category	borrows_qty
4	Book Title 4	Author 4	Biography	7
1	Book Title 1	Author 1	Biography	4
10	Book Title 10	Author 10	History	3
6	Book Title 6	Author 6	Fantasy	3
1	Book Title 1	Author 1	Biography	3
3	Book Title 3	Author 3	Self-Improvement	2
3	Book Title 3	Author 3	Self-Improvement	2
7	Book Title 7	Author 7	Horror	1
5	Book Title 5	Author 5	Non-Fiction	1
6	Book Title 6	Author 6	Fantasy	1
2	Book Title 2	Author 2	Mystery	1
8	Book Title 8	Author 8	Horror	1

- Analysis: The most borrowed book is Book Title 4 with 7 books.

I. REFERENCES

1. https://medium.com/@danishman/creating-a-modern-library-database-b7dff4313f28
2. https://vertabelo.com/blog/database-for-library-system/