Database System for an e-library Application

SQL and Relational Database - Job Preparation Program - Pacmann AI

Created by : Iqbal Al Fayyedh
JPP DE Batch 2b

Task Description

Your assignment is to design a database system for an e-library application. The application oversees multiple libraries, each hosting a diverse collection of books with varying quantities available for borrowing. Users can borrow or place holds on books (when the book is not immediately available for borrowing).
Below are the key points and requirements for the e-library database system:

- Manages multiple libraries
  The application manages multiple libraries, each housing a diverse collection of books with varying quantities available for borrowing.
- 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
  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
  - 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

Your task involves designing the database schema, ensuring that the relationships between tables are well-defined, and foreign keys are used appropriately. Include any additional attributes or tables necessary to support the described functionalities.

Your design should reflect a comprehensive understanding of the e-library's requirements and provide an effective solution to manage books, holds, and loans within a multi-library environment.

The Tools

- PostgreSQL
- Google Colab

Summary

Part 1: Designing The Database

1. **Mission Statement**
   E-library application adalah aplikasi yang dirancang untuk memudahkan pecinta buku untuk meminjam buku secara online. Dibutuhkan database yang mampu me-manage data untuk aplikasi tersebut.

   Terdapat sejumlah libraries yang memiliki berbagai macam koleksi buku. Setiap buku hanya dimiliki oleh satu library.

   Setiap user dapat meminjam maksimal 2 buku pada waktu yang sama dengan masa peminjaman tidak lebih dari 14 hari. Buku dapat dikembalikan sebelum tenggat waktu tercapai.

   Jika buku sedang tidak tersedia, user dapat melakukan booking dengan maksimal jumlah buku yang dapat di-booking adalah 2 buku pada waktu yang sama.

   Setiap user dapat mengakses peminjaman buku, melakukan booking, dan personalisasi profil pada platform ini.

   Platform senantiasa menyimpan aktivitas peminjaman ataupun booking.

   Limitations :
   -. Pada fitur “Setiap user dapat meninjam maksimal 2 buku” dan “Setiap user dapat menunggu maksimal 2 buku”, pada data base yang saya buat fitur ini tidak dibangun menggunakan logic python. Namun pada saat dataset sudah di-import ke database, dilakukan query untuk melihat apakah ada user yang tercatat meminjam/menunggu lebih dari 2 buku. Jika ada, raw data nya akan dimodifikasi agar memenuhi kriteria tersebut untuk kepentingan analisis. (penjelasan pada appendix)
   -. Pada tabel loans dan books ataupun holds dan books, mungkin ditemukan jumlah buku yang terpinjam/ditunggu melebihi jumlah buku yang tersedia karena data tersebut di-generate secara random.

-. Buku dengan id yang sama pada tabel loans dan holds ketika dijumlahkan pada periode “*active loan*” dapat melebihi jumlah buku yang tersedia.
-. Loan-hold system “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. “ tidak terdefinisi pada database ini.

2. Creating Table Structures

Dalam membuat table structure, kita perlu mengidentifikasi entity apa saja yang perlu dimasukkan ke dalam database.
Entity biasanya berkaitan dengan orang, aktivitas, ataupun benda.

Berdasarkan mission statement, kita melihat terdapat subjek user, libraries, books, and transactions (loan and booking) yang memungkinkan untuk dijadikan enitity pada database ini.

Berikut kandidat enitity yang akan dibuat:
-. Users
-. Libraries
-. Books
-. Loans
-. Holds

Selanjutnya, dapat kita definisikan deskripsi dan attribute untuk setiap entity beserta keys untuk setiap entity-nya..

libraries			books		
Store library information			Store detail about book collections		
- library_id - library_name	CK CK	**PK**	- book_id - library_id - title - author - book_quantity - category	CK CFK	**PK** FK

users		
Store user information		
- user_id - first_name - last_name - username - email - phone_number	CK CCK1 CCK1	**PK** CAK1 CAK1

loans		
Records loan transactions		
- loan_id - book_id - user_id - loan_date - due_date - return_date	CK CFK CFK	**PK** FK FK

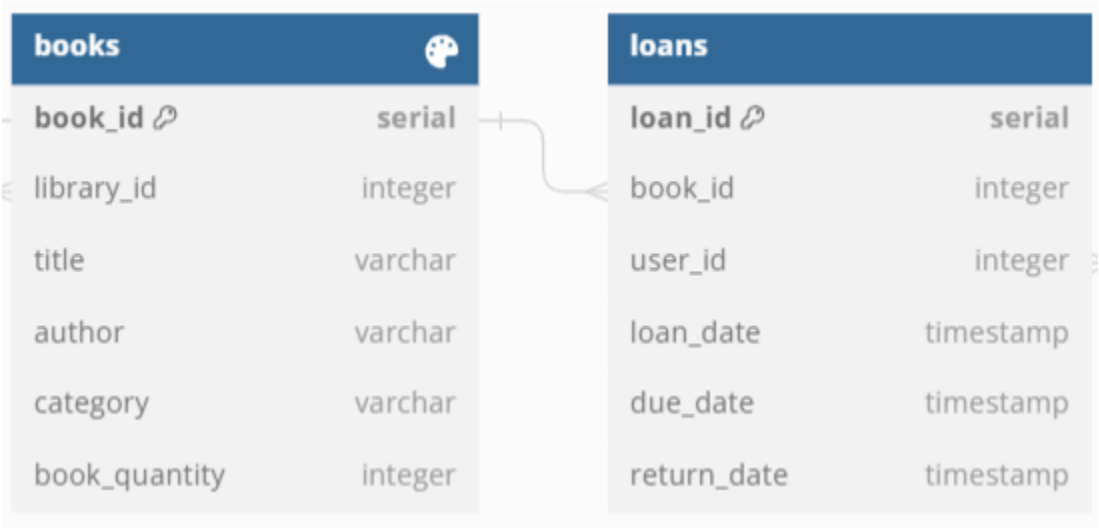
holds		
Manage holds placed by user		
- hold_id - book_id - user_id - hold_date - end_hold_date	CK CFK CFK	**PK** FK FK

3. **Determine Table Relationships**
   Selanjutnya, kita akan tentukan relationship antara tabel-tabel yang sudah dibuat.

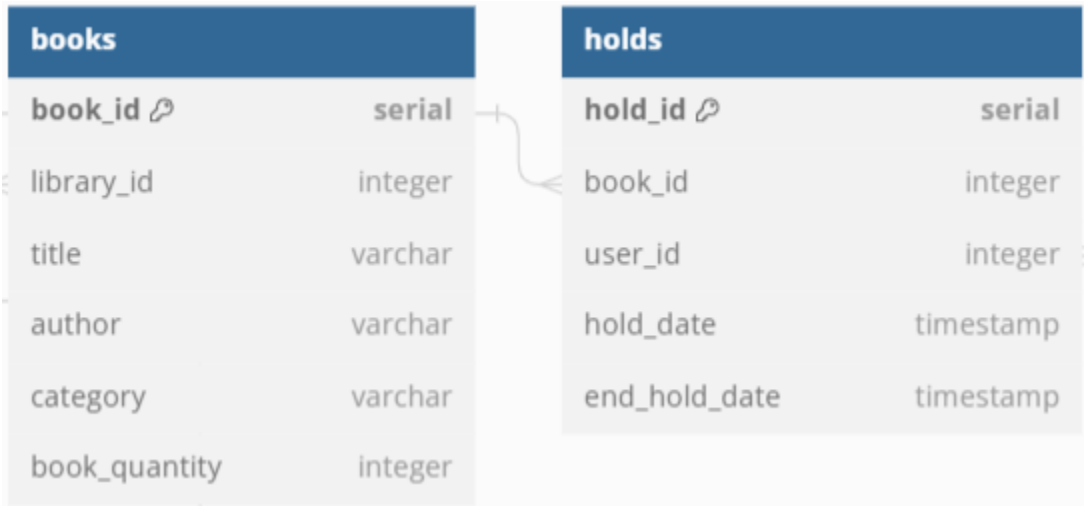
   Mapping the relationship:

   Dapat dikatakan bahwa library mempunyai koleksi buku. Dalam kasus saya, satu buku hanya bisa dimiliki oleh satu library. Sedangkan sebuah library dapat memiliki beberapa

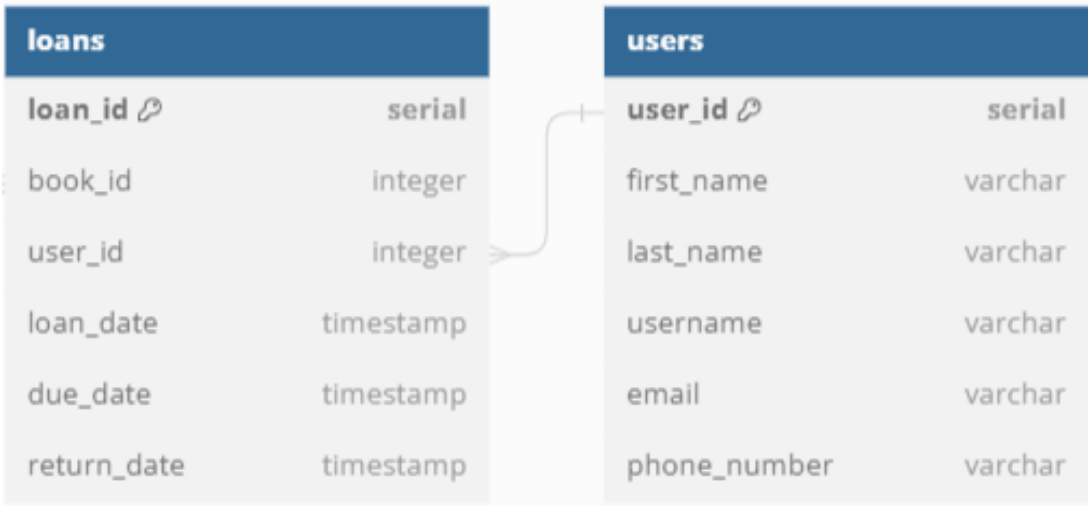
buku dalam koleksinya. Jadi kita tabel ini memiliki hubungan 1-to-N. Perhatikan bahwa field 'library_id' di tabel buku terhubung ke 'library_id' di tabel library, artinya atribut ini adalah foreign key, yang mereferensikan atribute 'library_id' di entity library.



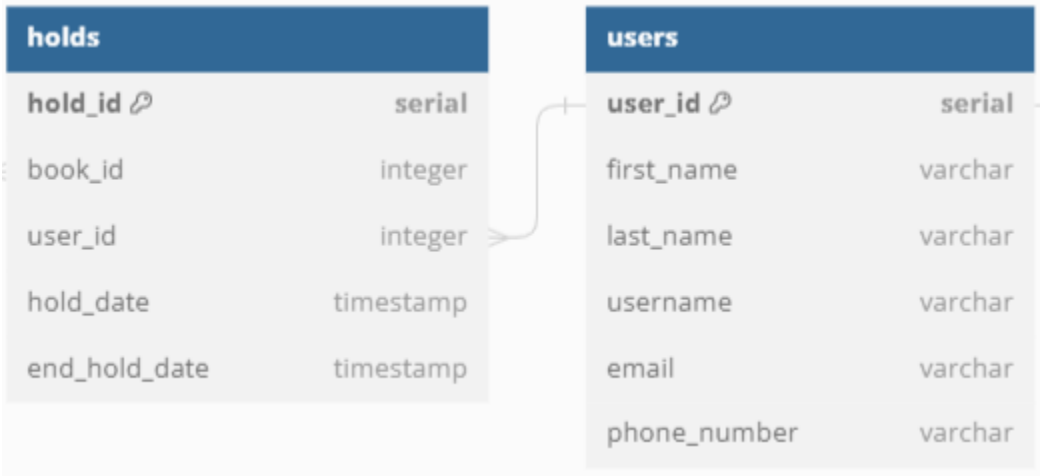
Ini adalah hubungan 1-ke-N dimana satu buku dapat berada dalam beberapa pinjaman, sedangkan satu pinjaman hanya dapat memiliki satu buku. Dan karena setiap pinjaman dapat memiliki satu buku saja, foreign key disimpan di tabel pinjaman.



Ini adalah hubungan 1-ke-N dimana satu buku dapat berada dalam beberapa booking, sedangkan satu booking hanya dapat memiliki satu buku. Dan karena setiap pinjaman dapat memiliki satu buku saja, foreign key disimpan di tabel pinjaman.



Masuk akal jika seorang user boleh memiliki lebih dari satu pinjaman (dalam hal ini maksimal 2 pcs), sedangkan satu loan_id dikaitkan hanya dengan satu user. Jadi hubungannya adalah 1-ke-N dan foreign key disimpan dalam tabel pinjaman.

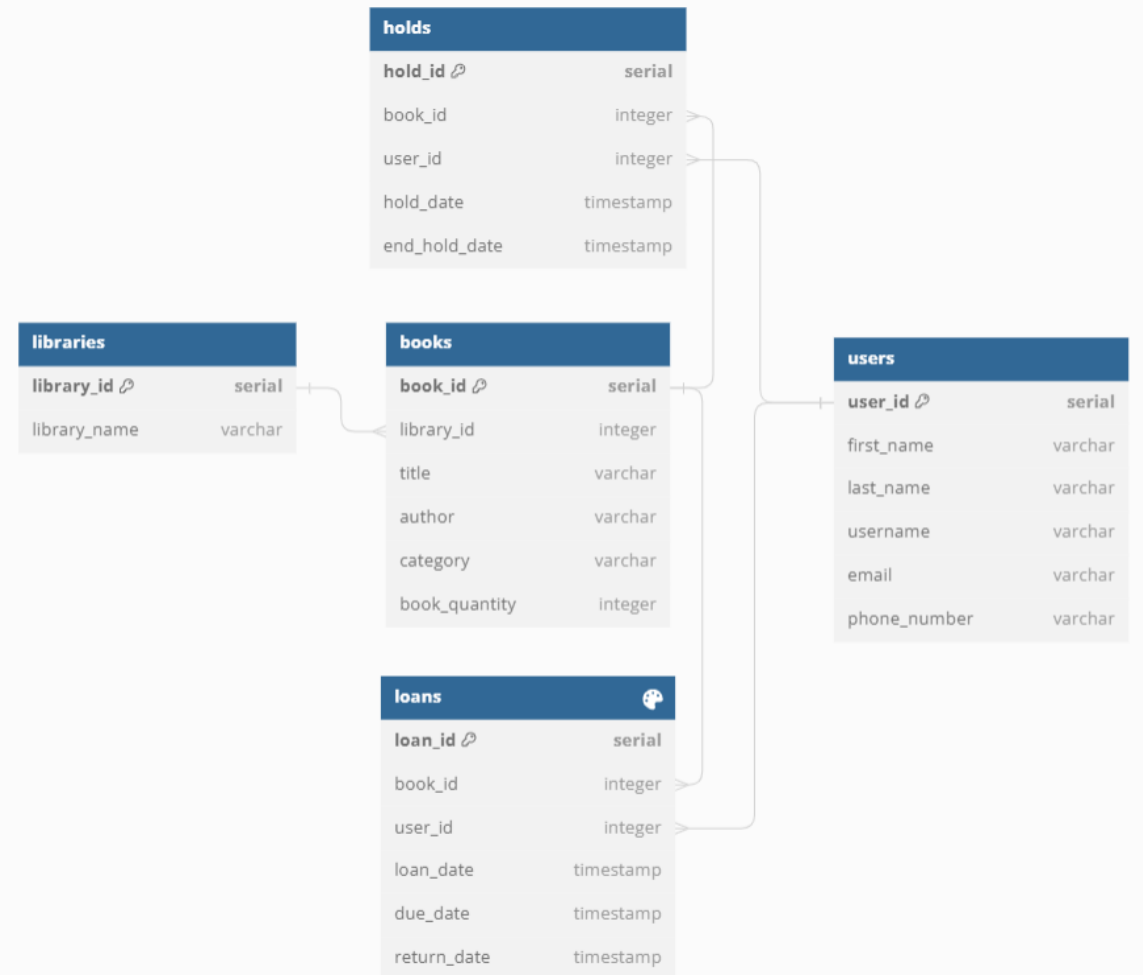


Kita dapat melakukan hal yang sama untuk memodelkan hubungan antara users dan holds. Seorang user boleh memiliki lebih dari satu booking (dalam hal ini maksimal 2 pcs), sedangkan satu hold_id dikaitkan hanya dengan satu user. Jadi hubungannya adalah 1-ke-N dan foreign key disimpan di tabel hold.

Relationship table:

	libraries	books	users	loans	holds
libraries		1:N			
books				1:N	1:N
users				1:N	1:N
loans					
holds					

Full Entity Relationship Diagram (ERD):



4. **Determine Business Rules**
   Selanjutnya, kita akan mendefinisikan rules pada setiap attribute.

libraries			books		
Store library information			Store detail about book collections		
- Library_id : int - Library_name : varchar	Not Null Not Null	**PK**	- book_id : int - library_id : int - title : varchar - author : varchar - book_quantity : int - category : varchar	Not Null Not Null Not Null Not Null Not Null Not Null	**PK** FK

users			loans		
Store user information			Records loan transactions		
- user_id : int - first_name : varchar - last_name : varchar - username : varchar - email : varchar - phone_number : varchar	Not Null Not Null Not Null Not Null Not Null Not Null	**PK** CAK1 CAK1	- loan_id : int - book_id : int - user_id : int - loan_date : timestamp - due_date : timestamp - return_date : timestamp	Not Null Not Null Not Null Not Null Not Null	**PK** FK FK

holds		
Manage holds placed by user		
- hold_id : int - book_id : int - user_id : int - hold_date : timestamp - End_hold_date : timestamp	Not Null Not Null Not Null Not Null	**PK** FK FK

5. **Implementing The Design**
   Setelah kita membuat table structure, membangun relationship, dan membuat Entity Relationship Diagram (ERD), saatnya untuk kita mengimplementasikannya pada PostgreSQL dan merumuskan Data Definition Language (DDL).

5.1 Merumuskan Data Definition Language (DDL)

DDL adalah bagian dari SQL yang digunakan untuk mendefinisikan dan mengelola struktur database. Statement DDL berfungsi untuk membuat, mengubah, dan menghapus objek database.

5.1.1 Creating Database Object

a. Libraries Table

```
CREATE TABLE libraries (
  library_id serial PRIMARY KEY,
  library_name varchar not null
);
```

b. Books Table

```
CREATE TABLE books (
  book_id serial PRIMARY KEY,
  library_id integer not null,
  title varchar not null,
  author varchar not null,
  category varchar not null,
  book_quantity integer not null check(book_quantity >= 0),
  CONSTRAINT fk_libraries
    FOREIGN KEY (library_id) REFERENCES libraries(library_id)
);
```

c. Users Table

```
CREATE TABLE users (
  user_id serial PRIMARY KEY,
  first_name varchar not null,
  last_name varchar not null,
  username varchar not null unique,
  email varchar not null unique,
  phone_number varchar not null
);
```

d. Loans Table

```
CREATE TABLE loans (
  loan_id serial PRIMARY KEY,
  book_id integer not null,
  user_id integer not null,
  loan_date timestamp not null DEFAULT CURRENT_TIMESTAMP,
  due_date timestamp not null GENERATED ALWAYS AS (loan_date + INTERVAL '14 days') STORED,
  return_date timestamp,
  CONSTRAINT fk_book
    FOREIGN KEY (book_id) REFERENCES books(book_id),
  CONSTRAINT fk_user
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

*return_date tidak not null karena value akan null jika buku belum dikembalikan

e. Holds Table

```
CREATE TABLE holds (
  hold_id serial PRIMARY KEY,
  book_id integer not null,
  user_id integer not null,
  hold_date timestamp not null DEFAULT CURRENT_TIMESTAMP,
  end_hold_date timestamp,
  CONSTRAINT fk_book
    FOREIGN KEY (book_id) REFERENCES books(book_id),
  CONSTRAINT fk_user
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

*end_hold_date tidak not null karena value akan null jika buku belum masih belum available

5.1.2 Deleting Database Object

Kita dapat menggunakan syntax berikut ini untuk menghapus table yang telah terbuat.

Syntax :

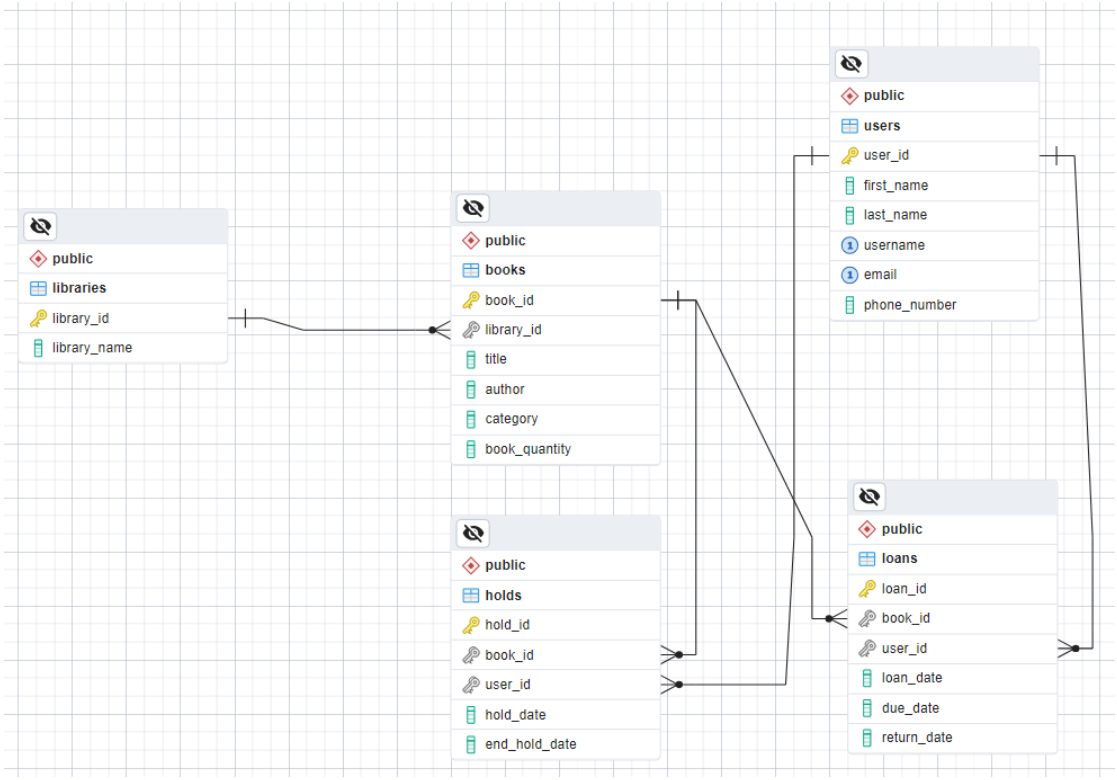
```
DROP TABLE IF EXISTS libraries;
DROP TABLE IF EXISTS books;
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS loans;
DROP TABLE IF EXISTS holds;
```

5.1.3 Altering Database Object

Statement *Alter* tidak digunakan dalam membangun database ini karena struktur database sudah diatur dari awal.

5.2 Entity Relationship Diagram (ERD)

Berikut adalah ERD yang di-generate oleh PostgreSQL setelah menjalankan DDL.



Part 2: Populating Dummy Dataset

1. **Create Dummy Dataset**

   Dalam membuat dataset untuk database ini, saya menggunakan Python untuk meng-generate dummy datanya.

   Berikut prosesnya.

a. Initialization

Instal Faker dan Tabulate library

```
!pip install Faker
!pip install tabulate
```

Importing Libraries

```
# Import Library yang akan digunakan
from faker import Faker
from tabulate import tabulate
import random
```

```
from datetime import datetime, timedelta
import csv
```

Define Faker localization

```
FAKER = Faker('id_ID')
```

Membuat fungsi untuk menampilkan data

```
def show_data(table):
    """
    Fungsi untuk menampilkan data

    arg:
        - table (dict) : data dictionary yang ingin ditampilkan

    return:
        None
    """

    tab = tabulate(tabular_data = table,
                   headers = table.keys(),
                   tablefmt = "psql",
                   numalign = "center")
    print(tab)
```

b. Membuat tabel users

Pertama, membuat fungsi untuk meng-generate nama.

```
def generate_name(n_name):
    """
    Fungsi untuk membuat nama dummy

    arg:
        - n_name (int) : jumlah data nama yang ingin dibuat

    return:
        names (list) : list nama yang sudah dibuat
    """

    names = list()

    while len(names) < n_name:
```

```
        first_name = FAKER.first_name()
        last_name = FAKER.last_name()

        full_name = (f'{first_name} {last_name}')
        if full_name not in names:
            names.append(full_name)

    return names
```

Kedua, membuat fungsi untuk meng-generate data pada setiap attribute.

```
def users_table(n_user, is_print):
    """
    Fungsi untuk membuat dummy data user table
    header:
        - user_id
        - first_name
        - last_name
        - username
        - e-mail
        - phone

    arg:
        - n_user (int)  : Jumlah user yang ingin dibuat
        - is_print  (bool) : Jika True akan menampilkan hasil data

    return:
        - table  (list) :
    """

    # Buat table
    table = {}
    table["user_id"] = [i+1 for i in range(n_user)]
    names = generate_name(n_user)
    table['first_name'] = [i.split(' ')[0] for i in names]
    table['last_name'] = [i.split(' ')[1] for i in names]
    table['username'] = [f"{name.lower().replace(' ', '')}" for name in names]
    table['email'] = [f"{name.lower().replace(' ', '')}@{FAKER.free_email_domain()}" \
                      for name in names]
    table['phone'] = [FAKER.phone_number() for i in range(n_user)]

    # Print table
    if is_print:
        show_data(table)

    return table
```

Terakhir, panggil fungsi untuk membuat tabel.

```
# membuat data table users
```

```
users_table = users_table(n_user = 1000,
                                    is_print = True)
```

c. Creating books table

Dalam membuat tabel buku, saya tidak menemukan provider Faker yang mampu menghasilkan data dummy untuk judul buku. Jadi, saya mendownload file csv dari Kaggle (https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset), lalu menggunakan judul dan author dari dataset tersebut.

Berikut prosesnya.

Pertama, buat fungsi untuk mengekstrak file csv ke list of dictionary.

```
def csv_to_dict(filename):
    """
    Fungsi untuk ekstrak file csv menjadi list of dictionary

    arg:
        - filename (str) : nama file csv yang akan dibuka
    return:
        - data  (list) :  list of dictionary
    """

    # buka file csv
    with open(f'{filename}', mode='r') as file:
        csv_reader = csv.DictReader(file)

        # simpan dalam bentuk list of dictionary
        data = {}
        for row in csv_reader:
            for key, value in row.items():
                # setdefault() untuk menambahkan key ke result_dict
                # value dari key diisi dengan empty list dulu
              # empty list diisi dengan method append per baris data
                data.setdefault(key, []).append(value)

    return data
```

```
# buka file review
books = csv_to_dict('Books.csv')
```

Kedua, membuat fungsi untuk meng-generate data pada setiap attribute.

```
def books_table(n_book, libraries_table, is_print):
    """
    Fungsi untuk membuat dummy data books table
    header:
        - book_id
        - library_id
        - title
        - author
        - category
        - book_quantity

    arg:
        - n_book (int)  : Jumlah buku yang ingin dibuat
        - is_print  (bool) : Jika True akan menampilkan hasil data

    return:
        - table  (list) :
    """
    # List of book category
    category_list = ['Romance', 'Sci-Fi', 'Adventure', 'Thriller',
                     'Comedy', 'Drama','History','Biography']

    # Buat table
    table = {}
    table["book_id"] = [i+1 for i in range(n_book)]
    table["library_id"] = [random.choice(libraries_table['library_id']) \
                           for i in range(n_book)]
    table['title'] = [books['Book-Title'][i] for i in range(n_book)]
    table['author'] = [books['Book-Author'][i] for i in range(n_book)]
    table['category'] = [random.choice(category_list) for i in range(n_book)]
    table['book_quantity'] = [FAKER.random_int(1, 5, 1) for i in range(n_book)]

    # Print table
    if is_print:
        show_data(table)

    return table
```

Terakhir, panggil fungsi untuk membuat tabel.

```
# membuat data table books
books_table = books_table(n_book = 1000, libraries_table = libraries_table, is_print =
True)
```

d. Creating libraries table

Dalam membangun tabel library, saya memutuskan untuk meng-generate data library_name menggunakan nama provinsi yang ada di Indonesia.

Pertama, buat fungsi untuk meng-generate data untuk setiap attribute. Karena library_name akan menggunakan nama provinsi di Indonesia, saya membuat daftar nama provinsi.

```
def libraries_table(n_lib, is_print):
    """
    Fungsi untuk membuat dummy data libraries table
    header:
        - library_id
        - library_name

    arg:
        - n_lib (int)  : Jumlah library yang ingin dibuat
        - is_print  (bool) : Jika True akan menampilkan hasil data

    return:
        - table  (list) :
    """

    # list of province
    province = ['Aceh',
                'Sumatera Utara',
                'Sumatera Barat',
                'Riau',
                'Jambi',
                'Sumatera Selatan',
                'Bengkulu',
                'Lampung',
                'Kepulauan Bangka Belitung',
                'Kepulauan Riau',
                'DKI Jakarta',
                'Jawa Barat',
                'Jawa Tengah',
                'DI Yogyakarta',
                'Jawa Timur',
                'Banten',
                'Bali',
                'Nusa Tenggara Barat',
                'Nusa Tenggara Timur',
                'Kalimantan Barat',
                'Kalimantan Tengah',
                'Kalimantan Selatan',
                'Kalimantan Timur',
                'Kalimantan Utara',
                'Sulawesi Utara',
                'Sulawesi Tengah',
                'Sulawesi Selatan',
                'Sulawesi Tenggara',
                'Gorontalo',
```

```
                'Sulawesi Barat',
                'Maluku',
                'Maluku Utara',
                'Papua',
                'Papua Barat']

    # Buat table
    table = {}
    table["library_id"] = [i+1 for i in range(n_lib)]
    table['library_name'] = [f"e-library {province[i]}" for i in range(n_lib)]

    # Print table
    if is_print:
        show_data(table)

    return table
```

Lalu, panggil fungsi untuk membuat tabel.

```
# membuat data table libraries
libraries_table = libraries_table(n_lib = 34, is_print = True)
```

e. Membuat tabel loans

Pertama, buat fungsi untuk meng-generate informasi terkait tanggal pinjaman.

```
def loan_generator():
    '''
    Fungsi untuk membuat informasi tanggal terkait peminjaman

    args:
        None

  return:
    - loan_date (str)     : tanggal peminjaman
    - return_date (str)   : tanggal pengembalian

    '''
    # definisikan awal tanggal
    start_date = datetime(2015, 1, 1)

    # definisikan akhir tanggal
    end_date = datetime(2023, 12, 31, 23, 59, 59)

    # membuat dummy loan date berdasarkan start_date dan end_date
    loan_date = FAKER.date_time_between(start_date = start_date,
```

```
                                        end_date = end_date)

    # banyak hari peminjaman
    end_delta = timedelta(days=(FAKER.random_int(1, 14, 1)))

    # membuat dummy return date
    return_date = datetime.combine(loan_date.date() + end_delta,
                                   FAKER.time_object())

    return loan_date, return_date
```

Kedua, membuat fungsi untuk meng-generate data pada setiap attribute.

```
def loans_table(n_loan, is_print):
    """
    Fungsi untuk membuat dummy data loans table
    header:
        - loan_id
        - book_id
        - user_id
        - loan_date
        - return_date

    arg:
        - n_loan (int)  : Jumlah loan yang ingin dibuat
        - is_print  (bool) : Jika True akan menampilkan hasil data

    return:
        - table  (list) :
    """
    # Buat table
    table = {}
    table["loan_id"] = [i+1 for i in range(n_loan)]
    table['book_id'] = [random.choice(books_table['book_id']) \
                            for i in range(n_loan)]
    table['user_id'] = [random.choice(users_table['user_id']) \
                            for i in range(n_loan)]
    date = [loan_generator() for i in range(n_loan)]
    table['loan_date'] = [date[i][0] for i in range(n_loan)]
    table['return_date'] = [date[i][1] for i in range(n_loan)]
```

```
    # Print table
    if is_print:
        show_data(table)

    return table
```

Terakhir, panggil fungsi untuk membuat tabel.

```
# membuat data tabel loans
loans_table = loans_table(n_loan = 5000, is_print = True)
```

f. Membuat tabel holds

Pertama, buat fungsi untuk menghasilkan informasi terkait tanggal booking.

```
def hold_generator():
    '''
    Fungsi untuk membuat tanggal reservasi

    args:
        None

  return:
    - hold_date (str)     : tanggal peminjaman
    - end_hold_date (str)      : tanggal antrian berakhir
    '''
    # definisikan awal tanggal
    start_date = datetime(2015, 1, 1)

    # definisikan akhir tanggal
    end_date = datetime(2023, 12, 31, 23, 59, 59)

    # membuat dummy berdasarkan start_date dan end_date
    hold_date = FAKER.date_time_between(start_date = start_date,
                                        end_date = end_date)

    # lama hari antrian
    end_delta = timedelta(days=(FAKER.random_int(1, 14, 1)))

    # timestamp antrian berakhir
    end_hold_date = datetime.combine(hold_date.date() + end_delta,
                                FAKER.time_object())

    # mengubah object datetime ke string
    # reservation_date = reservation_date.strftime('%d-%m-%Y %H-%M-%S')
```

```
    # start_date = start_date.strftime('%d-%m-%Y %H-%M-%S')
    # end_date = end_date.strftime('%d-%m-%Y %H-%M-%S')

    return hold_date, end_hold_date
```

Kedua, membuat fungsi untuk meng-generate data pada setiap attribute.

```
def holds_table(n_hold, is_print):
    """
    Fungsi untuk membuat dummy data holds table
    header:
        - hold_id
        - book_id
        - user_id
        - hold_date
        - end_hold_date

    arg:
        - n_hold (int)  : Jumlah hold yang ingin dibuat
        - is_print  (bool) : Jika True akan menampilkan hasil data

    return:
        - table  (list) :
    """
    # Buat table
    table = {}
    table["hold_id"] = [i+1 for i in range(n_hold)]
    table['book_id'] = [random.choice(books_table['book_id']) \
                            for i in range(n_hold)]
    table['user_id'] = [random.choice(users_table['user_id']) \
                            for i in range(n_hold)]
    date = [hold_generator() for i in range(n_hold)]
    table['hold_date'] = [date[i][0] for i in range(n_hold)]
    table['end_hold_date'] = [date[i][1] for i in range(n_hold)]

    # Print table
    if is_print:
        show_data(table)

    return table
```

Terakhir, panggil fungsi untuk membuat tabel.

```
# membuat data loans table
holds_table = holds_table(n_hold = 4500, is_print = True)
```

g. Setelah semua tabel dibuat, saatnya menyimpannya ke dalam file csv.

Buat fungsi untuk menyimpan data dummy ke dalam file csv.

```
def save_to_csv(data, nama_file):
    '''
    Fungsi untuk menyimpan data dummy ke csv

    args:
        - data (list)     : list of dictionary data yang akan dijadikan csv
        - nama_file (str) : nama untuk file csv

  return:
    - None
    '''

    # Membuat file csv
    with open(file = f"{nama_file}.csv", mode = 'w', newline = '') as csv_file:
        # Membuat writer csv
        writer = csv.writer(csv_file)

        # write header csv
        writer.writerow(list(data.keys()))

        # mengetahui panjang data
        len_data = len(list(data.items())[0][1])

        # write data ke file csv
        for i in range(len_data):
            row = []
            for key in data.keys():
                row.append(data[key][i])
            writer.writerow(row)
```

Panggil fungsi untuk menyimpan data.

Users table

```
# menyimpan data user dalam bentuk csv
save_to_csv(data = users_table,
            nama_file = 'users')
```

Books table

```
# menyimpan data buku dalam bentuk csv
save_to_csv(data = books_table,
            nama_file = 'books')
```

Library table

```
# menyimpan data library dalam bentuk csv
save_to_csv(data = libraries_table,
            nama_file = 'libraries')
```

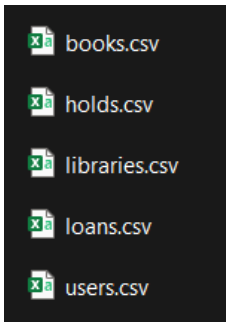
Loans table

```
# menyimpan data loan dalam bentuk csv
save_to_csv(data = loans_table,
            nama_file = 'loans')
```

Holds table

```
# menyimpan data buku dalam bentuk csv
save_to_csv(data = holds_table,
            nama_file = 'holds')
```

Output :



2. **Input Dummy Dataset into the Database:**
   Setelah file csv berhasil di export, saatnya memasukan dataset ke dalam database.
   Untuk dapat memasukkan dataset ke database, kita dapat menggunakan sytax berikut.

```
-- Import csv to database --
COPY users
FROM 'D:\3. Pacmann\2. SQL\Exercise\Week 6\csv files V2\users.csv'
DELIMITER ','
CSV
HEADER;

COPY libraries
FROM 'D:\3. Pacmann\2. SQL\Exercise\Week 6\csv files V2\libraries.csv'
DELIMITER ','
CSV
HEADER;

COPY books
FROM 'D:\3. Pacmann\2. SQL\Exercise\Week 6\csv files V2\books.csv'
DELIMITER ','
CSV
HEADER;

COPY loans
FROM 'D:\3. Pacmann\2. SQL\Exercise\Week 6\csv files V2\loans.csv'
DELIMITER ','
CSV
HEADER;

COPY holds
FROM 'D:\3. Pacmann\2. SQL\Exercise\Week 6\csv files V2\holds.csv'
DELIMITER ','
CSV
HEADER;
```

Part 3: Write 5 objectives/questions

1. Pada setiap tahun, pada bulan apa buku paling banyak dipinjam?

```
WITH monthly_loan_count AS (
    SELECT
        EXTRACT(YEAR FROM loan_date) AS loan_year,
        EXTRACT(MONTH FROM loan_date) AS loan_month,
        COUNT(*) AS total_loans,
        RANK() OVER (PARTITION BY EXTRACT(YEAR FROM loan_date) ORDER BY COUNT(*) DESC) AS month_rank
    FROM
        loans
    GROUP BY
        loan_year, loan_month
)

SELECT
    loan_year,
    loan_month,
    total_loans,
    month_rank
FROM
    monthly_loan_count
WHERE
    month_rank = 1;
```

loan_year numeric	loan_month numeric	total_loans bigint	month_rank bigint
2015	7	57	1
2016	9	50	1
2016	3	50	1
2017	3	57	1
2017	7	57	1
2018	10	60	1
2019	3	56	1
2020	7	63	1
2021	1	64	1
2022	4	58	1
2022	5	58	1
2023	3	59	1

Why is it important : Melihat trend peminjaman buku berdasarkan ranking bulan pada setiap tahunnya

Analysis **:** Di sejumlah tahun, terdapat 2 bulan yang berada pada rangking 1. Selain itu, jika dilihat dari distribusi bulannya, bulan 3 muncul paliing banyak.

Recommendation : Perlu dilihat buku apa saja yang paling banyak dipinjam pada bulan 3. Dapat dilakukan juga research lebih detail mengenai hubungan kategori paling banyak yang dipinjam pada bulan 3 dan

ada event apa pada bulan tersebut. Sehingga pola tersebut dapat diterapkan dibulan-bulan lain agar bisa menambah frekuensi peminjaman.

2. Berikan data top 10 buku yang paling sering dipinjam!

```
SELECT
    b.title,
    b.category,
    b.author,
    count(l.book_id) as total_loan
FROM loans l
JOIN books b
    ON l.book_id = b.book_id
GROUP BY 1,2,3
ORDER BY total_loan DESC, category ASC
LIMIT 10;
```

title character varying	category character varying	author character varying	total_loan bigint
Protect and Defend	Drama	Richard North Patterson	15
The Perfect Storm : A True Story of Men Against the Sea	Biography	Sebastian Junger	14
Cheaters	Biography	Eric Jerome Dickey	13
Life of Pi	Thriller	Yann Martel	13
The Rime of the Ancient Mariner and Other Poems	Drama	Samuel Taylor Coleridge	12
The Street Lawyer	Adventure	JOHN GRISHAM	11
New Perspectives: Runes	Adventure	Bernard King	11
Thieves of Light (Photon : the Ultimate Game on Planet Eart...	Adventure	Michael Hudson	11
The Women in His Life (G K Hall Large Print Book Series)	Biography	Barbara Taylor Bradford	11
Keep It Simple: And Get More Out of Life	Drama	Nick Page	11

Why is it important : Untuk melihat trend peminjaman buku

Analysis **:** dari top 10 buku yang paling sering dipinjam, kategori Drama, Biografi, dan Adventure terlihat mendominasi. Untuk lebih meyakinkan, dapat dilihat kategori apa yang paling banyak di-booking.

Recommendation : Terlepas dari kategori, buku-buku tersebut banyak peminat dan bisa dipertimbangkan untuk menambah quantity-nya.

3. Berikan data top 3 kategori yang paling banyak di booking!

```
SELECT
    b.category,
    count(h.book_id) as total_booking
FROM holds h
JOIN books b
    ON h.book_id = b.book_id
GROUP BY 1
ORDER BY total_booking DESC
LIMIT 3;
```

category character varying	total_booking bigint
Biography	693
Drama	618
History	600

Why is it important : Untuk mengkonfirmasi pertanyaan no.2

Analysis **:** Top 3 kategori yang paling banyak di-booking adalah Biography, Drama, dan History. Kategori Biography dan Drama juga menjadi kategori yang paling sering dipinjam.

Recommendation : Ini menandakan buku-buku dengan kategori Bigraphy dan Drama merupakan buku favorit sehingga dapat dipertimbangkan untuk menambah quantity-nya.

4. Berapa lama peminjaman minimum, maximum, dan rata-rata untuk setiap kategori buku?

```sql
WITH loan_duration as (
    SELECT
        loan_id,
        (return_date - loan_date) as duration
    FROM
        loans
)

SELECT
    category,
    min(duration) as minimum_duration,
    max(duration) as maximum_duration,
    avg(duration) as average_duration
FROM
    loans
JOIN loan_duration using(loan_id)
JOIN books using(book_id)
GROUP BY 1
ORDER BY 1;
```

category character varying	minimum_duration interval	maximum_duration interval	average_duration interval
Adventure	01:23:18.922637	14 days 23:01:40.873813	6 days 31:09:05.963002
Biography	08:01:31.109926	14 days 20:42:25.409963	6 days 32:08:02.337994
Comedy	04:41:28.691613	14 days 20:32:20.308045	6 days 35:16:54.188171
Drama	03:00:53.476578	14 days 15:56:01.014646	6 days 35:44:15.096343
History	05:01:44.328071	14 days 14:26:33.849102	7 days 12:59:56.869691
Romance	05:45:50.836888	14 days 19:17:08.079667	6 days 29:22:26.875455
Sci-Fi	01:44:38.039087	14 days 21:56:33.838396	7 days 14:28:44.053342
Thriller	08:29:48.682873	14 days 19:42:59.124861	6 days 35:45:40.865078

Why is it important : Melihat apakah waktu maksimal peminjaman sudah optimal

Analysis **:** Dari seluruh kategory, rata-rata buku kembali setelah 6 hari. Dapat diasumsikan wkatu peminjaman selama 14 hari sudah optimal untuk saat ini.

Recommendation : tidak ada

5. Tampilkan data 10 user yang paling sering melakukan peminjaman!

```
SELECT
    CONCAT(u.first_name, ' ', u.last_name) as customer_name,
    u.email,
    u.phone_number,
    count(l.user_id) as loan_frequency
FROM loans l
JOIN users u
    ON l.user_id = u.user_id
GROUP BY 1,2,3
ORDER BY loan_frequency DESC, customer_name ASC
LIMIT 10;
```

customer_name text	email character varying	phone_number character varying	loan_frequency bigint
Juli Uyainah	juliuyainah@yahoo.com	+62-43-610-1011	12
Kasiran Mangunsong	kasiranmangunsong@yahoo.com	(034) 515 3002	12
Kezia Sihombing	keziasihombing@hotmail.com	(0233) 139 4672	12
Luwar Laksita	luwarlaksita@yahoo.com	+62 (966) 449 5592	12
Mursinin Siregar	mursininsiregar@gmail.com	0840713080	12
Reksa Wahyudin	reksawahyudin@yahoo.com	(002) 127 6769	12
Sarah Hassanah	sarahhassanah@gmail.com	(0280) 913-3496	12
Usman Sudiati	usmansudiati@yahoo.com	(091) 974 2475	12
Baktiadi Riyanti	baktiadiriyanti@gmail.com	0832408911	11
Galiono Maryati	galionomaryati@yahoo.com	+62 (45) 010-8301	11

Why is it important : Mempertimbangkan user yang bisa mendapat program khusus

Analysis **:** Rata-rata user meminjam buku sebanyak 12 kali.

Recommendation : Dapat dipertimbangkan untuk memberikan program khusus (ex. Loyalty program) dimana user yang memenuhi kriteria dapat meminjam buku hingga 3 pcs dalam waktu bersamaan dalam periode tertentu.

Appendix.

1. Documentation

Github : https://github.com/alfayyedh/e-library-database/tree/main

Medium :
https://medium.com/@alfayyedh/database-system-for-an-e-library-application-0aa0303e8dec

2. Dataset Verification
    1. Cek apakah ada user yang meminjam lebih dari 2 buku pada waktu bersamaan.
    Digunakan LAG function untuk menghitung rentang waktu peminjaman buku.
    Jika current_return_date - previous_return_date < 14 hari, artinya terdapat peminjaman yang overlap. Pada kasus ini, peminjaman overlap hanya dibatasi maksimal 2 pcs/user.
    Pada dataset yang di-generate menggunakan Faker, terdapat user yang memiliki peminjaman overlap sebanyak 3. Sehingga dataset perlu dimodifikasi.
    Modifikasi dilakukan secara manual pada raw data csv.

```sql
-- Cek apakah ada user yang meminjam lebih dari 2 buku pada saat bersamaan --
WITH lag_return as (
    SELECT
        user_id,
        loan_date,
        return_date,
        LAG(return_date) OVER (PARTITION BY user_id ORDER BY user_id,loan_date) AS prev_return_date
    FROM
        loans
)
SELECT
    user_id,
    count(user_id) as user_count
FROM
    lag_return
WHERE
    (return_date - prev_return_date) < '14 days'
GROUP BY 1
HAVING count(user_id) > 2
ORDER BY 1;
```

Sebelum modifikasi

	user_id integer	user_count bigint
1	78	3

Setelah modifikasi

2. Metode yang sama juga dilakukan pada holds table.

   Hasilnya tidak ada holds/booking yang overlap > 2pcs/user.

```sql
-- Cek apakah ada user yang booking lebih dari 2 buku pada saat bersamaan --
WITH lag_end_hold_date as (
    SELECT
        user_id,
        hold_date,
        end_hold_date,
        LAG(end_hold_date) OVER (PARTITION BY user_id ORDER BY user_id,hold_date) AS prev_end_hold_date
    FROM
        holds
)
SELECT
    user_id,
    count(user_id) as user_count
FROM
    lag_end_hold_date
WHERE
    (end_hold_date - prev_end_hold_date) < '14 days'
GROUP BY 1
HAVING count(user_id) > 2
ORDER BY 1;
```

   Result:

Reference

https://medium.com/towards-data-science/practical-sql-create-and-query-a-relational-database-8bac84d78703

https://towardsdatascience.com/designing-a-relational-database-and-creating-an-entity-relationship-diagram-89c1c19320b2

https://pacmann.io/course

https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset