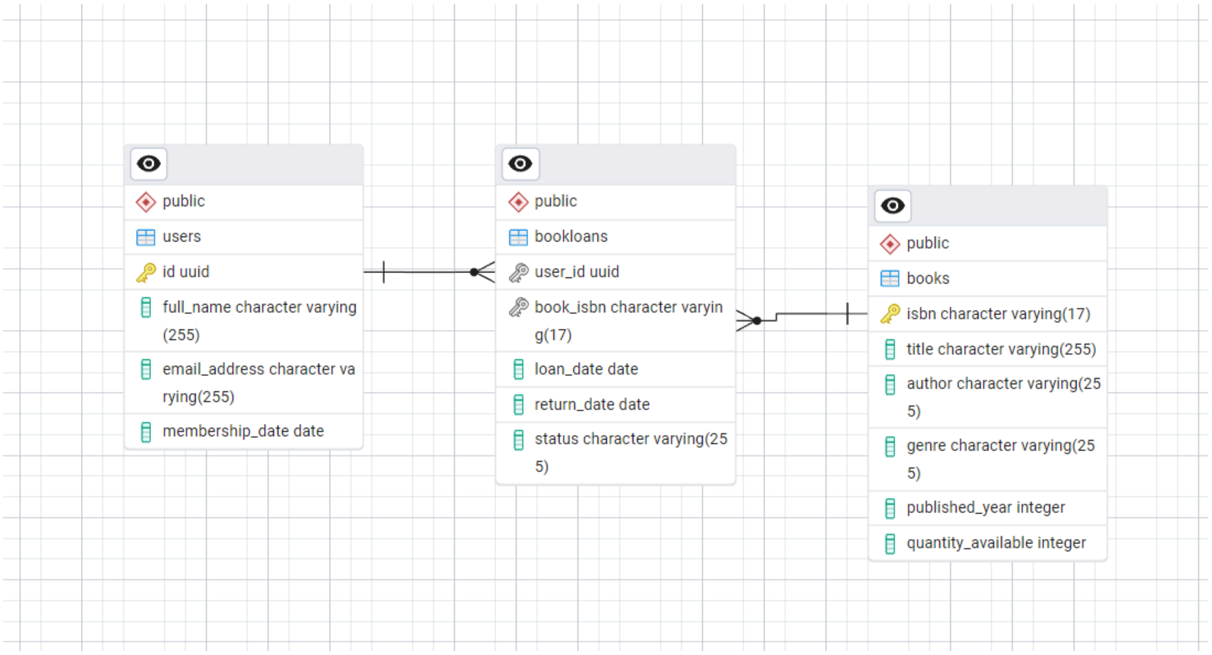
SE 2141 - LABORATORY 4

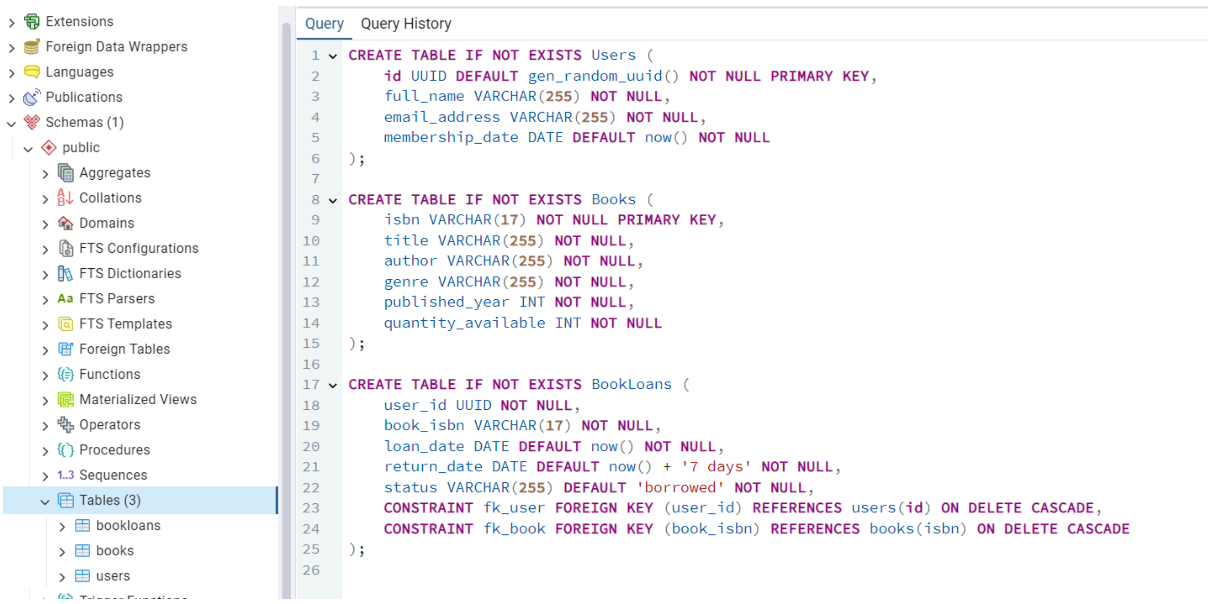
December 11, 2024

Name: John Rofer B. Casio **Course & Year:** BSSE - 2

Part 1:


public
users
id uuid
full_name character varying (255)
email_address character varying(255)
membership_date date
public
bookloans
user_id uuid
book_isbn character varying(17)
loan_date date
return_date date
status character varying(255)
public
books
isbn character varying(17)
title character varying(255)
author character varying(255)
genre character varying(255)
published_year integer
quantity_available integer

Part 2:



```sql
CREATE TABLE IF NOT EXISTS Users (
    id UUID DEFAULT gen_random_uuid() NOT NULL PRIMARY KEY,
    full_name VARCHAR(255) NOT NULL,
    email_address VARCHAR(255) NOT NULL,
    membership_date DATE DEFAULT now() NOT NULL
);

CREATE TABLE IF NOT EXISTS Books (
    isbn VARCHAR(17) NOT NULL PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255) NOT NULL,
    genre VARCHAR(255) NOT NULL,
    published_year INT NOT NULL,
    quantity_available INT NOT NULL
);

CREATE TABLE IF NOT EXISTS BookLoans (
    user_id UUID NOT NULL,
    book_isbn VARCHAR(17) NOT NULL,
    loan_date DATE DEFAULT now() NOT NULL,
    return_date DATE DEFAULT now() + '7 days' NOT NULL,
    status VARCHAR(255) DEFAULT 'borrowed' NOT NULL,
    CONSTRAINT fk_user FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    CONSTRAINT fk_book FOREIGN KEY (book_isbn) REFERENCES books(isbn) ON DELETE CASCADE
);
```

Part 3:

- This will be sent on the github with a file named “Table Schemas.sql”.

Part 4:

- I prevented the borrowing of books when no copies are available with the following codes:

  ```
  FROM Books b
  WHERE b.title = 'To Kill a Mockingbird'
    AND b.quantity_available > 0
  ```

  This is further shown in the All Queries SQL file in lines 35 to 53.

- I did the fast retrieval of books that are overdue by first updating by updating the quantity_available by adding the count of books loaned that are overdue by selecting all of them and using the count function to get how many there are and adding it to the current quantity_available.

  This is further shown in the All Queries SQL file in lines 122 to 141.

Part 5:

Reflection:

The first that would become a problem would probably be the JOIN operation, using this operation might make the retrieval slow with a massive dataset. A solution to this might be with the use of indexing which would most likely make the performance better.

The second would probably be the START TRANSACTION and COMMIT, as this locks the tables, which might prevent user access. To solve this, removing them completely can help but having them also has its pros.

Lastly, the subqueries to update the quantity_available is probably not ideal with a lot of datasets. A solution to this can be stored procedures to deal with bulk updates.