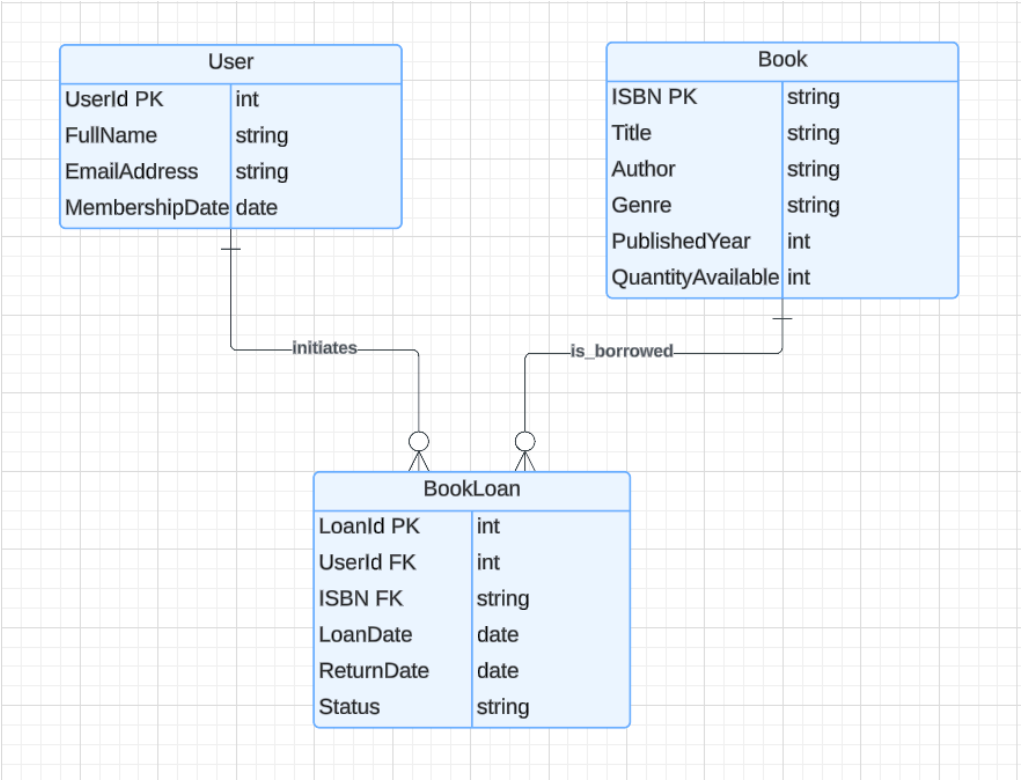
Elmor John Cortez **BSSE-2 SE2141**

Part 1: Conceptual Design



Part 2: Logical Design

Create Table For Users

```
CREATE TABLE Users (
    UserID SERIAL PRIMARY KEY,
    FullName VARCHAR(255) NOT NULL,
    EmailAddress VARCHAR(255) UNIQUE NOT NULL,
    MembershipDate DATE NOT NULL
);
```

Results Chart Export Source Primary Database postgres Run CTRL

```
Success. No rows returned
```

Create Table for Books

```sql
CREATE TABLE Books (
    ISBN VARCHAR(13) PRIMARY KEY,
    Title VARCHAR(255) NOT NULL,
    Author VARCHAR(255) NOT NULL,
    Genre VARCHAR(50),
    PublishedYear INT,
    QuantityAvailable INT NOT NULL CHECK (QuantityAvailable >= 0)
);
```

Results Chart Export Source Primary Database role: postgres Run CTRL

```
Success. No rows returned
```

Create Table for LoanBooks

```sql
CREATE TABLE BookLoans (
    LoanID SERIAL PRIMARY KEY,
    UserID INT NOT NULL,
    BookISBN VARCHAR(13) NOT NULL,
    LoanDate DATE NOT NULL,
    ReturnDate DATE,
    Status VARCHAR(20) NOT NULL,
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (BookISBN) REFERENCES Books(ISBN)
);
```

Results Chart Export Source Primary Database role: postgres Run CTRL

```
Success. No rows returned
```

Part 3: SQL Queries

a. Insert a new book into the library with a quantity of 5.

```sql
INSERT INTO Books (ISBN, Title, Author, Genre, PublishedYear, QuantityAvailable)
VALUES ('091234567890', 'Grokking Algorithms', 'Aditya Y Bhargava
', 'computer science', 2015, 5);
```

Results Chart Export Source Primary Database role postgres Run CTRL

Success. No rows returned

Filter Sort Insert RLS disabled role postgres Realtime off

isbn varchar	title varchar	author varchar	genre varchar	publishedyear int4	quantityavailable int4
091234567890	Grokking Algorithms	Aditya Y Bhargava	computer science	2015	5

b. Add a new user to the system.

```sql
INSERT INTO Users (FullName, EmailAddress, MembershipDate)
VALUES ('Carmine Tuden', 'Carmine@example.com', '2024-12-11');
```

Results Chart Export Source Primary Database role postgres Run CTRL

Success. No rows returned

Filter Sort Insert

use... i...	fullname varchar	emailaddress varchar	membershipd... d...
1	Carmine Tuden	Carmine@example.com	2024-12-11

c. Record a book loan for a user.

```sql
INSERT INTO BookLoans (UserID, BookISBN, LoanDate, Status)
VALUES (1, '091234567890', '2024-12-11', 'borrowed');
```

Results Chart Export Source Primary Database role postgres Run CTRL

Success. No rows returned

Filter Sort Insert RLS

loa... i...	use... i...	bookisbn varchar	loandate date	returndate date	status varchar
1	1	091234567890	2024-12-11	NULL	borrowed

d. Find all books borrowed by a specific user.

```sql
SELECT Books.Title, Books.Author, BookLoans.LoanDate
FROM BookLoans
JOIN Books ON BookLoans.BookISBN = Books.ISBN
WHERE BookLoans.UserID = 1;
```

Results Chart Export Source Primary Database role postgres Run CTRL

title	author	loandate
"Grokking Algorithms'	"Aditya Y Bhargava\r\	"2024-12-11"

e. List all overdue loans.

```
SELECT Users.FullName, Books.Title, BookLoans.LoanDate, BookLoans.ReturnDate
FROM BookLoans
JOIN Users ON BookLoans.UserID = Users.UserID
JOIN Books ON BookLoans.BookISBN = Books.ISBN
WHERE BookLoans.Status = 'overdue';
```

Results Chart Export Source Primary Database role postgres Run CTRL

```
Success. No rows returned
```

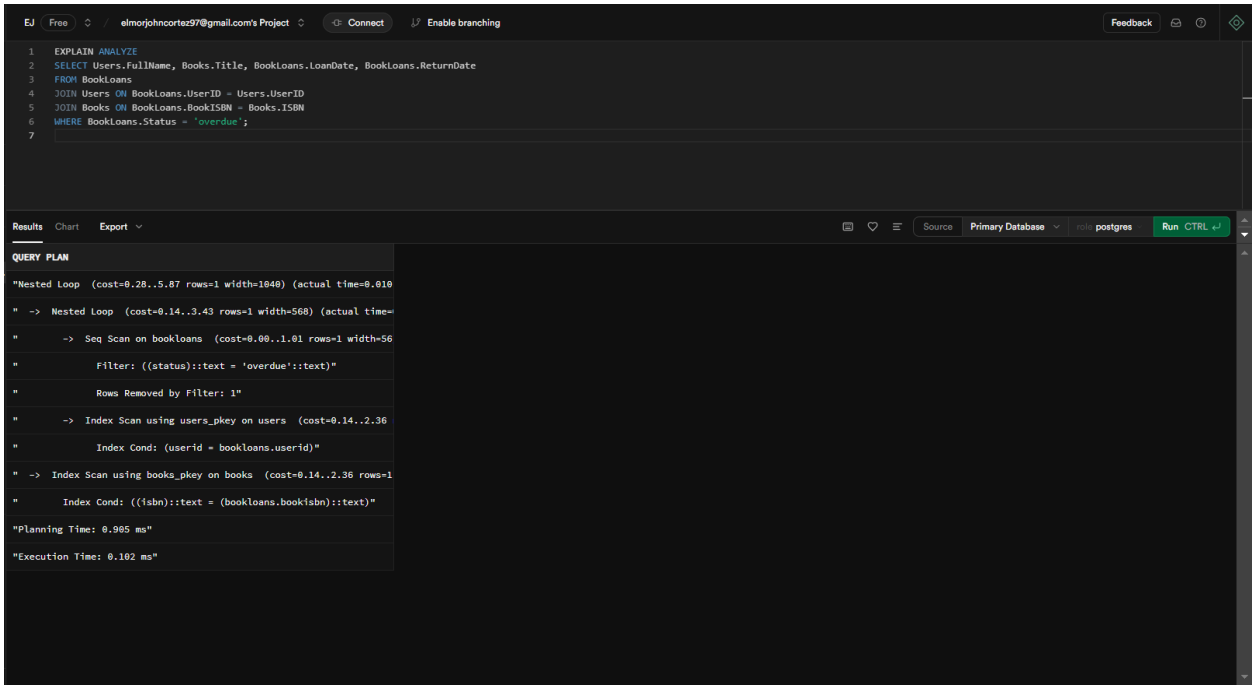
Part 4: Data Integrity and Optimization

First, Check Book Availability, You need a way to check how many copies of a book are available before a borrowing request is allowed. This can be done by looking up the book's details in the system, specifically the QuantityAvailable field, which tells you how many copies of that book are still available for borrowing.

Second, Function to Perform the Check, Create a function that performs this check whenever a new borrowing request a new entry in the BookLoans table is made. The function should look up the book by its unique ISBN and retrieve the number of available copies . If the available quantity is 0 or less, the function raises an exception, preventing the borrowing action from going through.

Lastly, Trigger to Automatically Run the Function, A trigger is used to automatically execute the function before any new record is inserted into the BookLoans table. The trigger checks the availability of the book right before a borrowing request is recorded. If the book is unavailable, it stops the process and prevents the loan from being made.

Fast retrieval of overdue loans.



Part 5: Reflection

5. What challenges might arise when scaling this database to handle millions of users and books? Suggest one solution for each challenge.

- First, handling millions of users and books can create significant database performance challenges. High query volumes can slow down operations considerably. To address this, one effective solution is to implement indexing on frequently queried fields, such as ISBN or UserID. Additionally, caching mechanisms can be used to temporarily store commonly accessed data, reducing the load on the database.

- Second, scalability is another challenge. Storing and backing up massive amounts of data becomes a critical issue as the library grows. A practical solution is horizontal scaling, which involves distributing data across multiple servers using distributed databases. This approach not only increases storage capacity but also improves performance.

- Lastly, data integrity is crucial. Ensuring consistency when multiple users access or modify data simultaneously can be problematic. For instance, two users attempting to

borrow the same book at the same time could cause errors. To prevent this, database transaction controls and locks can be implemented to ensure that only one process modifies data at a time, maintaining consistency.