

# Database

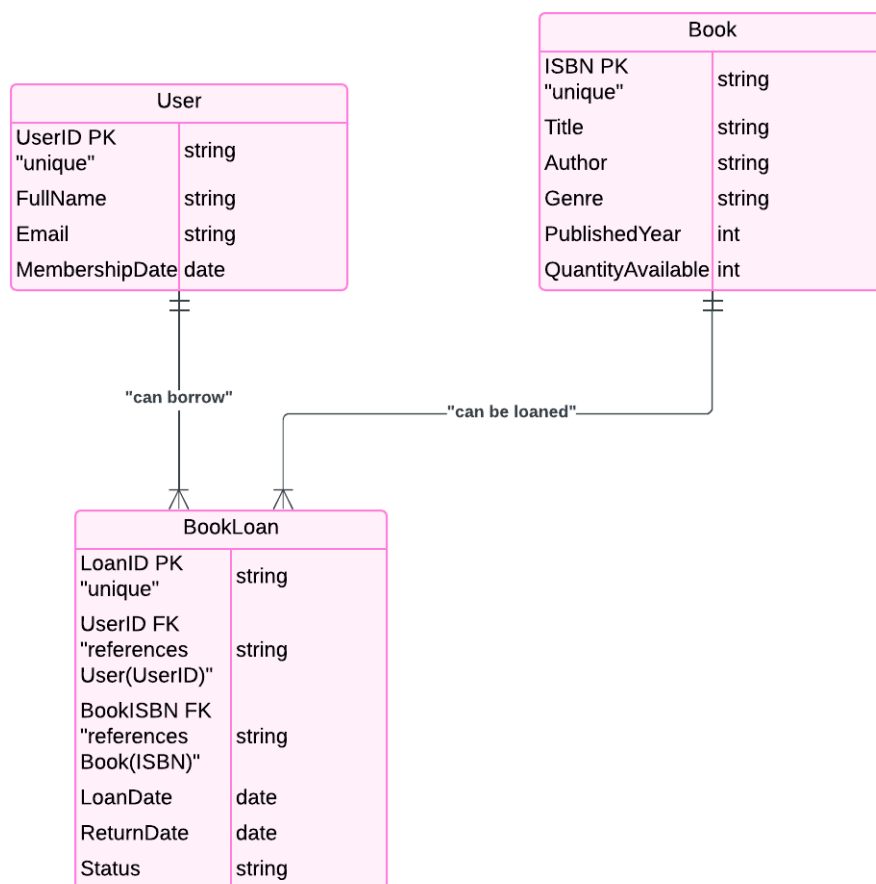
## Lab4

Name: Nelissa Tuden BSSE - 2

## Entity-Relationship (ER) Diagram

### Part 1: Conceptual Design

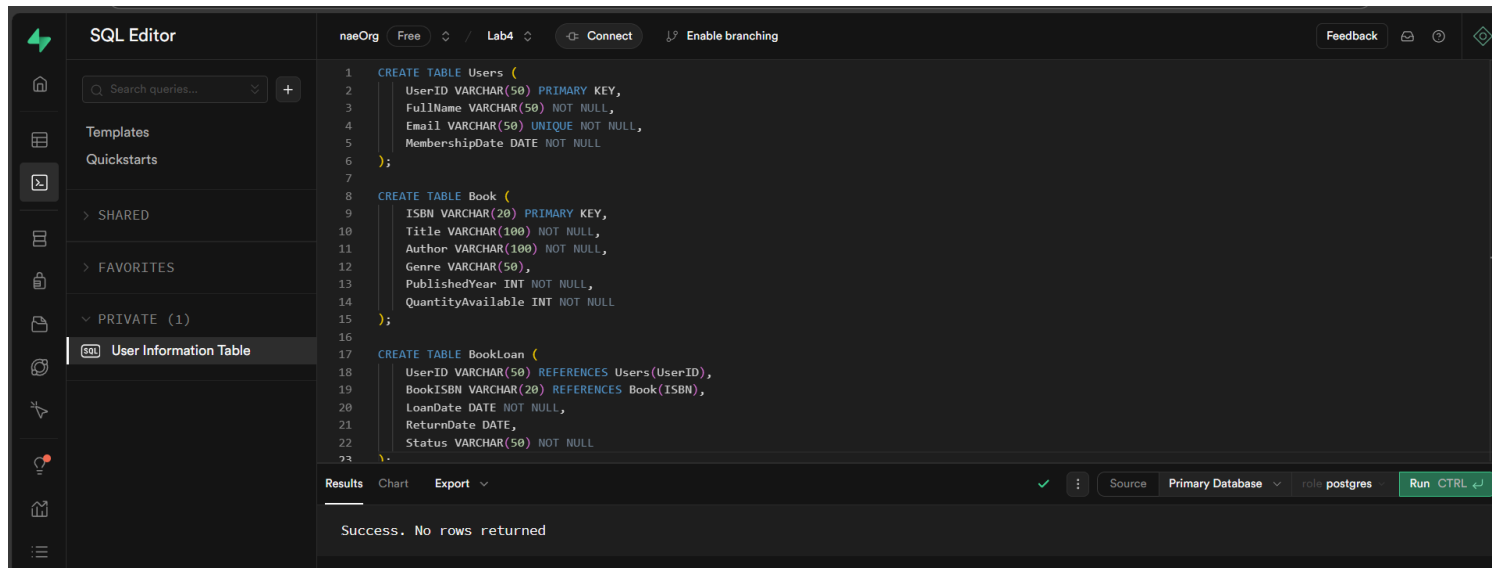
#### Conceptual Design



A user can borrow many books, but a book can only have a one book loan.

## Part 2: Logical Design

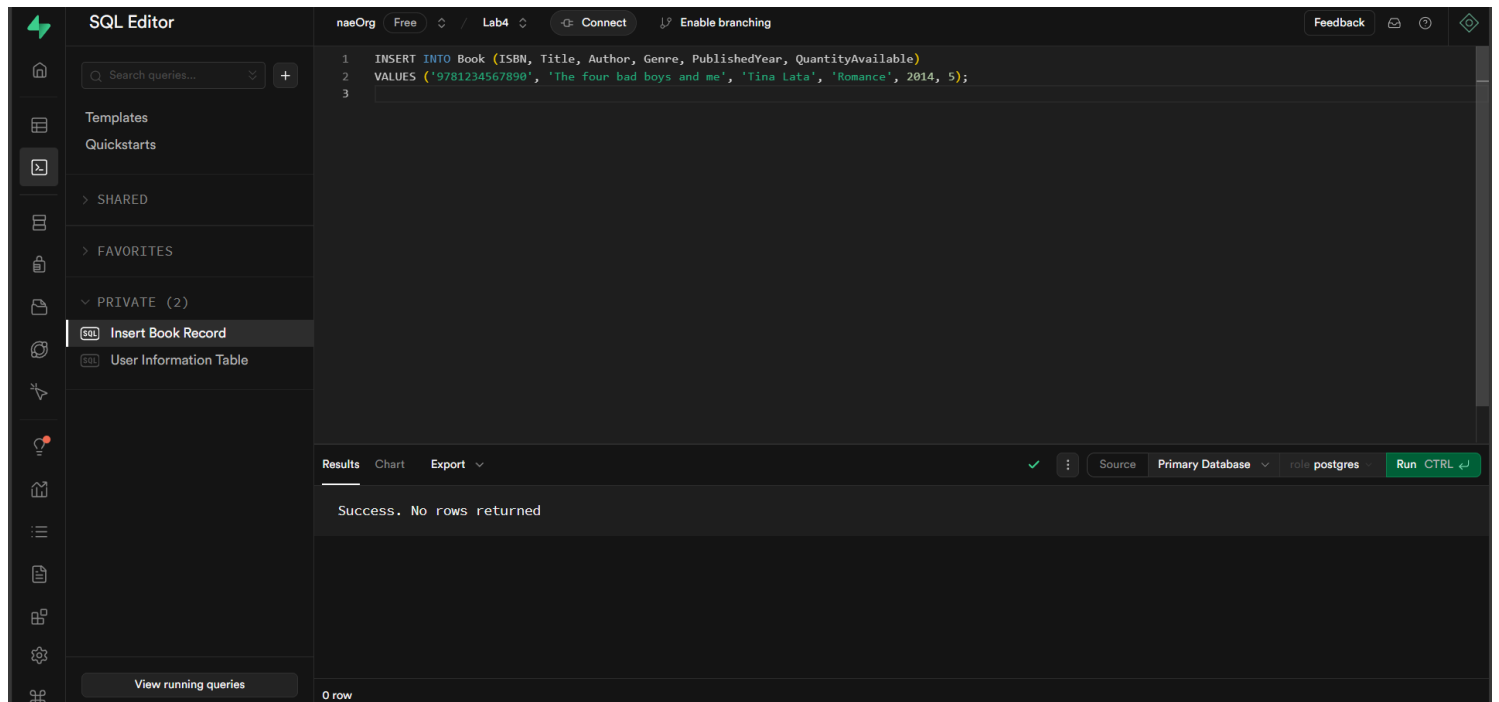
### Logical Design



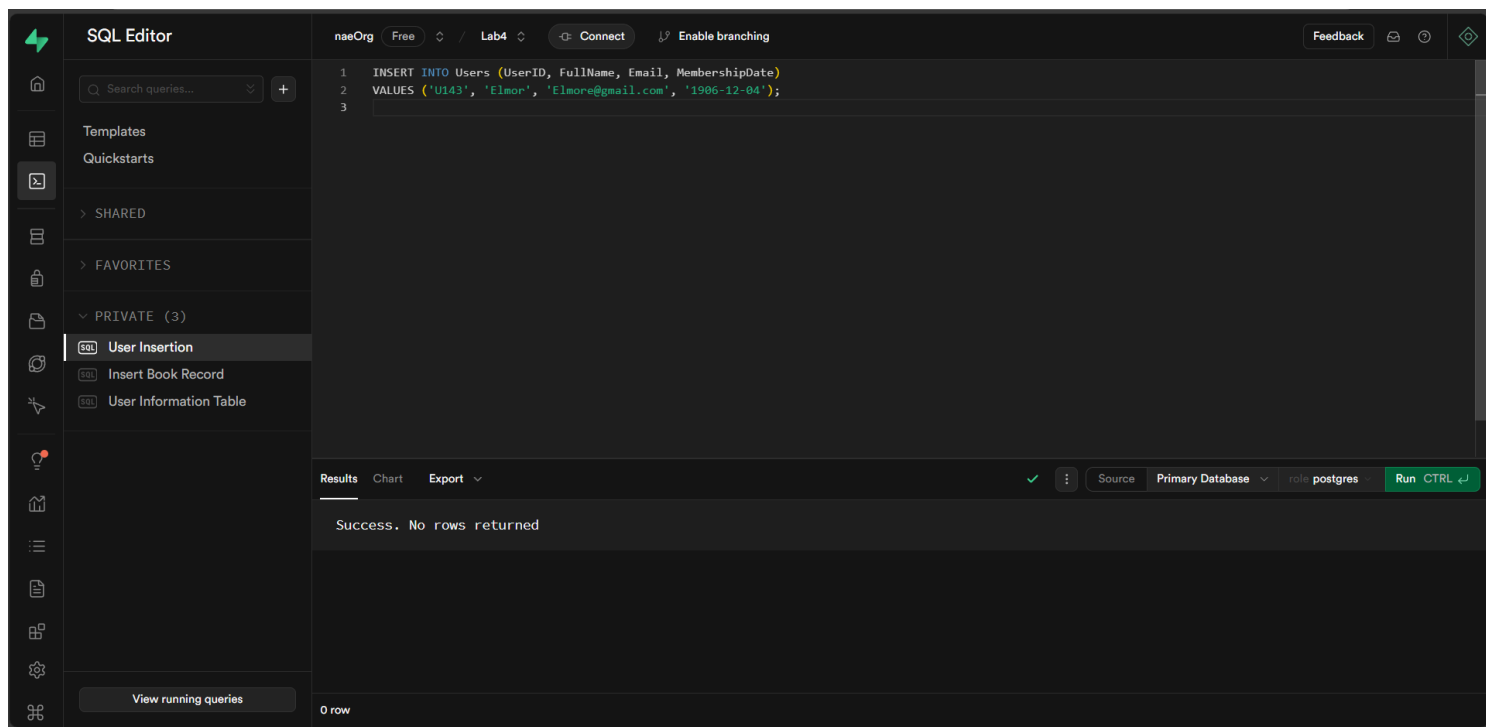
Creating tables for the users, book and bookloan.

## Part 3: SQL Queries

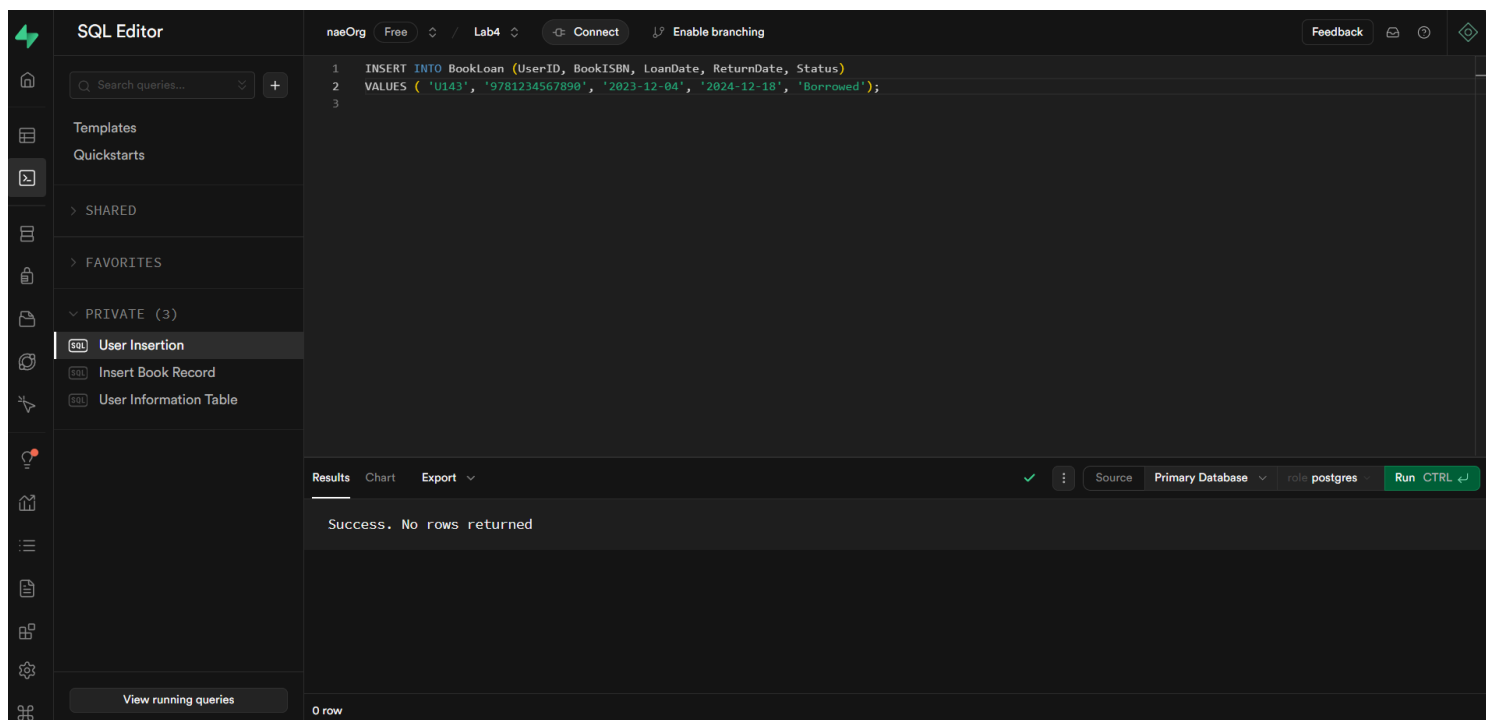
- Insert a new book into the library with a quantity of 5.



- Add a new user to the system.



- Record a book loan for a user.



- Find all books borrowed by a specific user.

The SQL Editor interface shows a query in the editor pane:

```

1 SELECT b.Title, b.Author, b.PublishedYear, bl.LoanDate, bl.ReturnDate
2 FROM Book b
3 JOIN BookLoan bl ON b.ISBN = bl.BookISBN
4 WHERE bl.UserID = 'U143' AND bl.Status = 'Borrowed';
5

```

The results pane displays the following data:

title	author	publishedyear	loandate	returndate
"The four bad boys and me"	"Tina Lata"	2014	"2023-12-04"	"2024-12-18"

The interface includes a sidebar with a file explorer showing 'User Insertion', 'Insert Book Record', and 'User Information Table'. The bottom status bar indicates '1 row (Limited to only 100 rows)' and 'Limit results to: 100 rows'.

- List all overdue loans.

The SQL Editor interface shows a query in the editor pane:

```

1 SELECT bl.userid, b.title, bl.loandate, bl.returndate
2 FROM BookLoan bl
3 JOIN Book b ON bl.BookISBN = b.ISBN
4 WHERE bl.returndate < CURRENT_DATE AND bl.status = 'Borrowed';
5

```

The results pane displays the following data:

userid	title	loandate	returndate
"U143"	"The four bad boys and me"	"2024-10-08"	"2024-11-10"

The interface includes a sidebar with a file explorer showing 'Overdue Book Loans', 'User Insertion', 'Insert Book Record', and 'User Information Table'. The bottom status bar indicates '1 row (Limited to only 100 rows)' and 'Limit results to: 100 rows'.

## Part 4: Data Integrity and Optimization

- The prevention of borrowing books when no copies are available.

The screenshot shows the Supabase SQL Editor interface. The left sidebar contains a search bar and a list of queries under 'PRIVATE (5)'. The main editor area contains the following SQL code:

```

1
2 SELECT QuantityAvailable
3 FROM Book
4 WHERE ISBN = '9781234567890';
5
6 UPDATE Book
7 SET QuantityAvailable = QuantityAvailable - 1
8 WHERE ISBN = '9781234567890' AND QuantityAvailable > 0;
9
10
11 INSERT INTO BookLoan ( UserID, BookISBN, LoanDate, ReturnDate, Status)
12 VALUES ('U143', '9781234567890', '2024-10-08', '2024-11-10', 'Borrowed');
13

```

Below the editor, the 'Results' tab is active, showing a single row of data:

quantityavailable
5

The bottom status bar indicates '1 row'.

- Fast retrieval of overdue loans.

The screenshot shows the Supabase SQL Editor interface. The left sidebar contains a search bar and a list of queries under 'PRIVATE (6)'. The main editor area contains the following SQL code:

```

1
2 CREATE INDEX idx_return_date ON BookLoan (ReturnDate);
3
4 SELECT bl.UserID, b.Title, bl.LoanDate, bl.ReturnDate
5 FROM BookLoan bl
6 JOIN Book b ON bl.BookISBN = b.ISBN
7 WHERE bl.ReturnDate < CURRENT_DATE AND bl.Status = 'Borrowed';
8

```

Below the editor, the 'Results' tab is active, showing a table of data:

userid	title	loandate	returndate
"U143"	"The four bad boys and me"	"2024-10-08"	"2024-11-10"

## Part 5: Reflection

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

The main challenges that might arise when scaling this database to handle millions of users and books is that it will slow the performance of this

database. As the load increases, performance slow down. I am not that pro in database, but I can suggest this solution. We could split the database into smaller, easier-to-manage parts, so that the database would be able to manage the data given. Also, will use NoSQL database because NoSQL can manage a large amount of data and can scale it easily. Finally, horizontal scaling adds more servers to share the workload. Using these methods together will help keep performance strong as your user base grows.