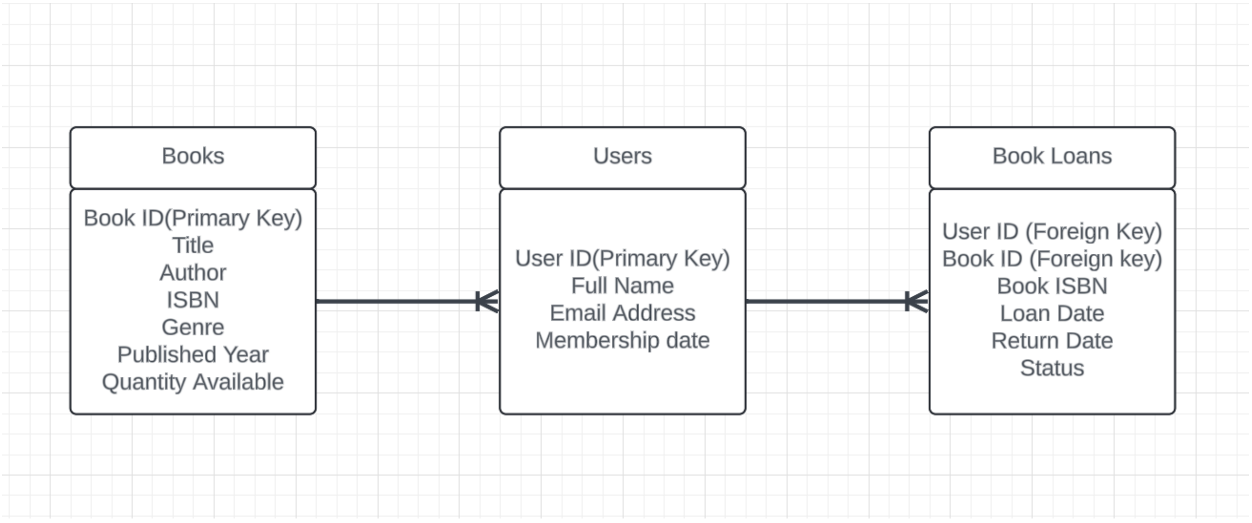
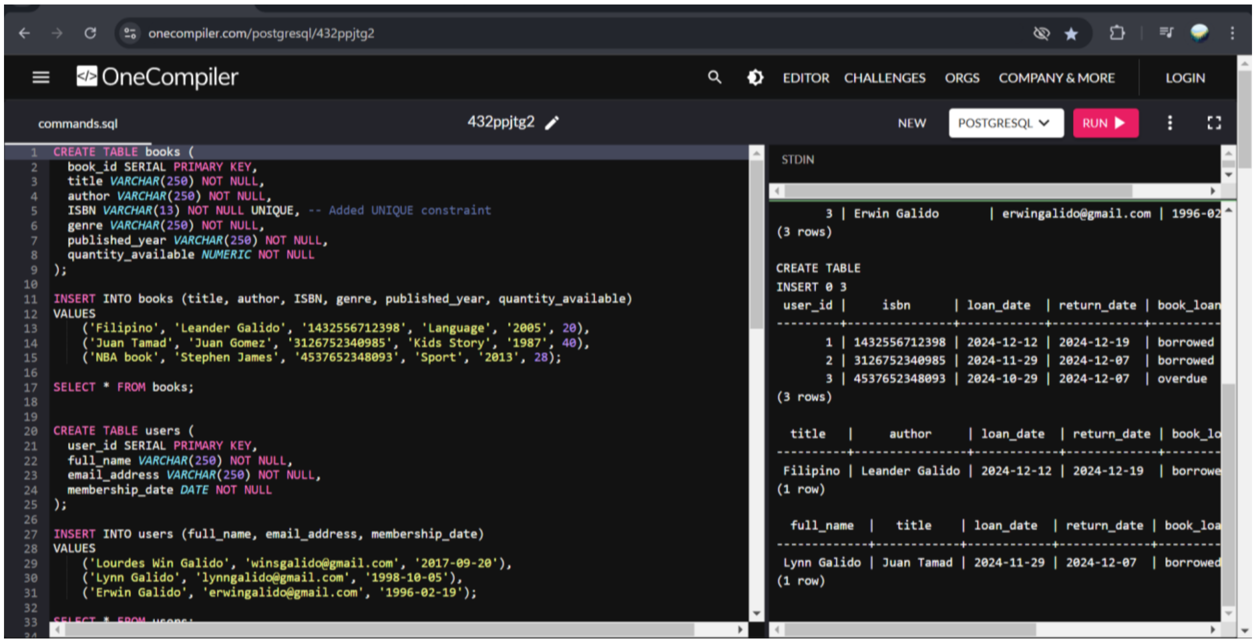
Part I.

1. **Conceptual Design**

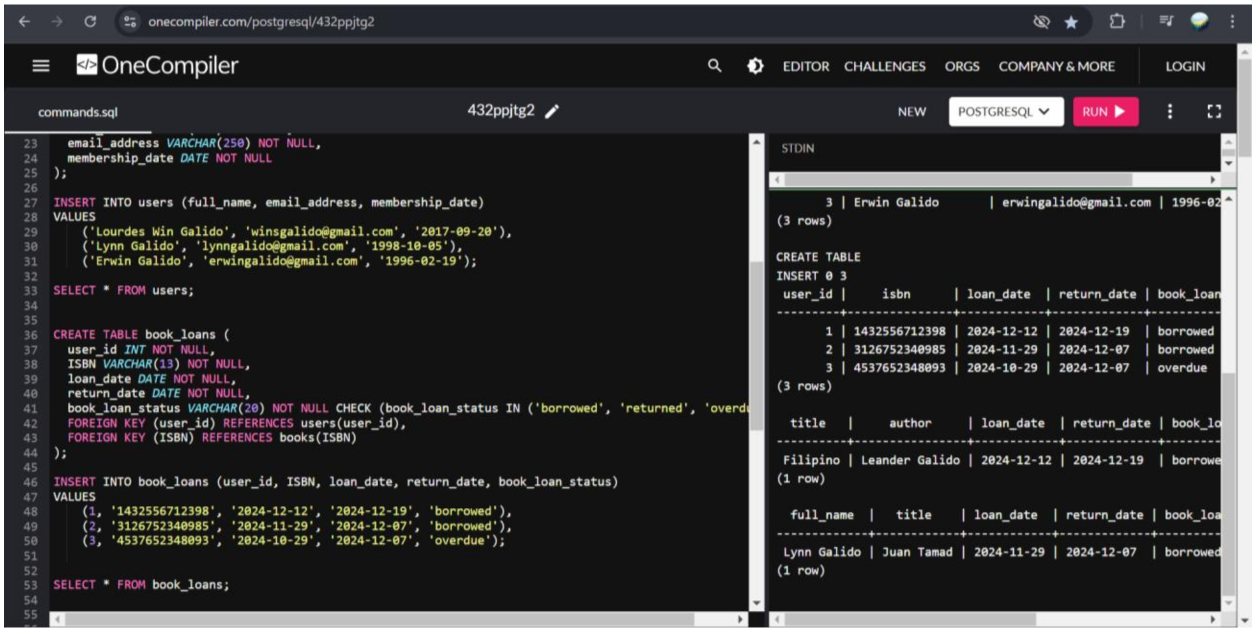


2. TABLES



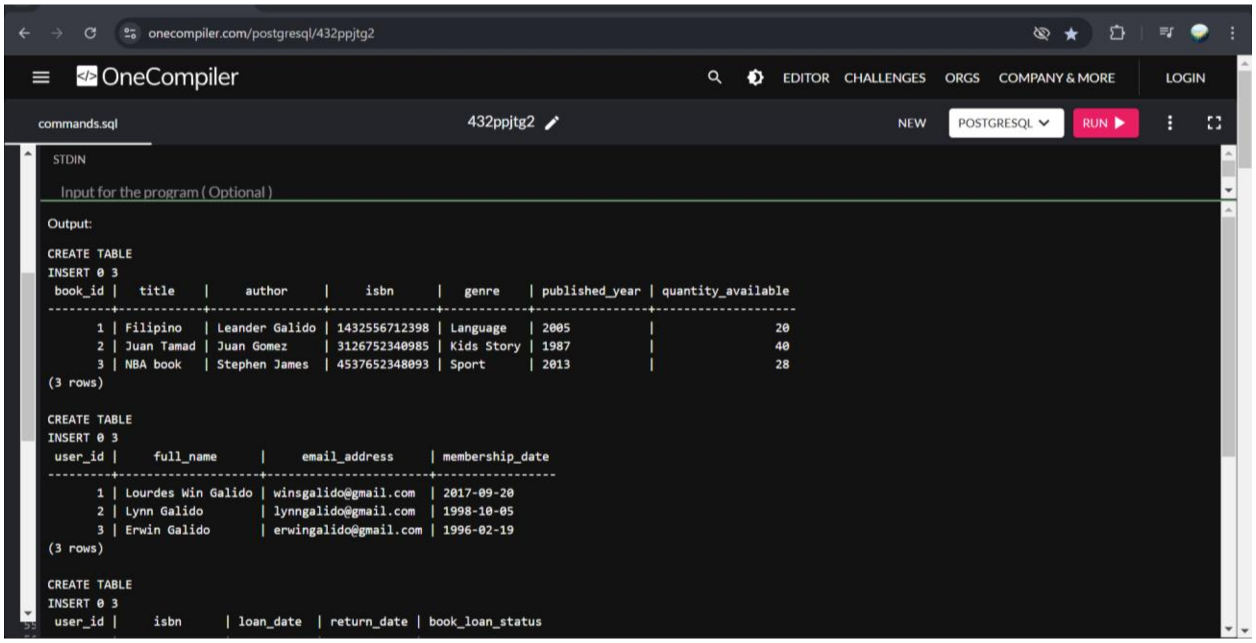
```sql
CREATE TABLE books (
  book_id SERIAL PRIMARY KEY,
  title VARCHAR(250) NOT NULL,
  author VARCHAR(250) NOT NULL,
  ISBN VARCHAR(13) NOT NULL UNIQUE, -- Added UNIQUE constraint
  genre VARCHAR(250) NOT NULL,
  published_year VARCHAR(250) NOT NULL,
  quantity_available NUMERIC NOT NULL
);

INSERT INTO books (title, author, ISBN, genre, published_year, quantity_available)
VALUES
    ('Filipino', 'Leander Galido', '1432556712398', 'Language', '2005', 20),
    ('Juan Tamad', 'Juan Gomez', '3126752340985', 'Kids Story', '1987', 40),
    ('NBA book', 'Stephen James', '4537652348093', 'Sport', '2013', 28);

SELECT * FROM books;


CREATE TABLE users (
  user_id SERIAL PRIMARY KEY,
  full_name VARCHAR(250) NOT NULL,
  email_address VARCHAR(250) NOT NULL,
  membership_date DATE NOT NULL
);

INSERT INTO users (full_name, email_address, membership_date)
VALUES
    ('Lourdes Win Galido', 'winsgalido@gmail.com', '2017-09-20'),
    ('Lynn Galido', 'lynngalido@gmail.com', '1998-10-05'),
    ('Erwin Galido', 'erwingalido@gmail.com', '1996-02-19');

SELECT * FROM users;
```

```
       3 | Erwin Galido       | erwingalido@gmail.com | 1996-02
(3 rows)

CREATE TABLE
INSERT 0 3
 user_id |     isbn      | loan_date  | return_date | book_loan
---------+---------------+------------+-------------+----------
       1 | 1432556712398 | 2024-12-12 | 2024-12-19  | borrowed
       2 | 3126752340985 | 2024-11-29 | 2024-12-07  | borrowed
       3 | 4537652348093 | 2024-10-29 | 2024-12-07  | overdue
(3 rows)

  title   |     author     | loan_date  | return_date | book_lo
----------+----------------+------------+-------------+--------
 Filipino | Leander Galido | 2024-12-12 | 2024-12-19  | borrowe
(1 row)

  full_name  |   title    | loan_date  | return_date | book_loa
-------------+------------+------------+-------------+---------
 Lynn Galido | Juan Tamad | 2024-11-29 | 2024-12-07  | borrowed
(1 row)
```

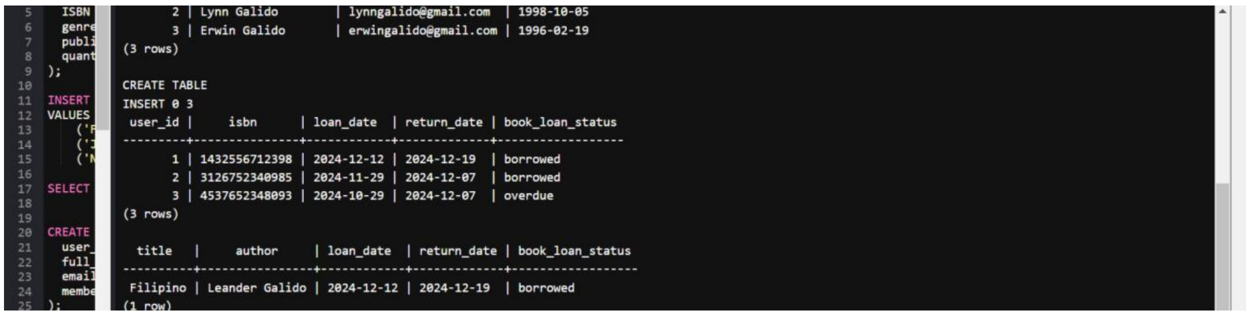


```sql
  email_address VARCHAR(250) NOT NULL,
  membership_date DATE NOT NULL
);

INSERT INTO users (full_name, email_address, membership_date)
VALUES
    ('Lourdes Win Galido', 'winsgalido@gmail.com', '2017-09-20'),
    ('Lynn Galido', 'lynngalido@gmail.com', '1998-10-05'),
    ('Erwin Galido', 'erwingalido@gmail.com', '1996-02-19');

SELECT * FROM users;


CREATE TABLE book_loans (
  user_id INT NOT NULL,
  ISBN VARCHAR(13) NOT NULL,
  loan_date DATE NOT NULL,
  return_date DATE NOT NULL,
  book_loan_status VARCHAR(20) NOT NULL CHECK (book_loan_status IN ('borrowed', 'returned', 'overdu
  FOREIGN KEY (user_id) REFERENCES users(user_id),
  FOREIGN KEY (ISBN) REFERENCES books(ISBN)
);

INSERT INTO book_loans (user_id, ISBN, loan_date, return_date, book_loan_status)
VALUES
    (1, '1432556712398', '2024-12-12', '2024-12-19', 'borrowed'),
    (2, '3126752340985', '2024-11-29', '2024-12-07', 'borrowed'),
    (3, '4537652348093', '2024-10-29', '2024-12-07', 'overdue');


SELECT * FROM book_loans;
```

```
       3 | Erwin Galido       | erwingalido@gmail.com | 1996-02
(3 rows)

CREATE TABLE
INSERT 0 3
 user_id |     isbn      | loan_date  | return_date | book_loan
---------+---------------+------------+-------------+----------
       1 | 1432556712398 | 2024-12-12 | 2024-12-19  | borrowed
       2 | 3126752340985 | 2024-11-29 | 2024-12-07  | borrowed
       3 | 4537652348093 | 2024-10-29 | 2024-12-07  | overdue
(3 rows)

  title   |     author     | loan_date  | return_date | book_lo
----------+----------------+------------+-------------+--------
 Filipino | Leander Galido | 2024-12-12 | 2024-12-19  | borrowe
(1 row)

  full_name  |   title    | loan_date  | return_date | book_loa
-------------+------------+------------+-------------+---------
 Lynn Galido | Juan Tamad | 2024-11-29 | 2024-12-07  | borrowed
(1 row)
```



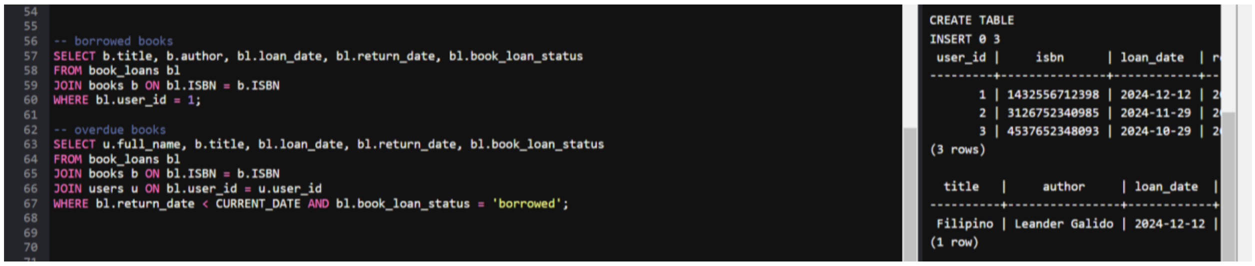
```
Output:

CREATE TABLE
INSERT 0 3
 book_id |   title    |     author     |     isbn      |   genre    | published_year | quantity_available
---------+------------+----------------+---------------+------------+----------------+-------------------
       1 | Filipino   | Leander Galido | 1432556712398 | Language   | 2005           |                 20
       2 | Juan Tamad | Juan Gomez     | 3126752340985 | Kids Story | 1987           |                 40
       3 | NBA book   | Stephen James  | 4537652348093 | Sport      | 2013           |                 28
(3 rows)

CREATE TABLE
INSERT 0 3
 user_id |     full_name      |     email_address     | membership_date
---------+--------------------+-----------------------+----------------
       1 | Lourdes Win Galido | winsgalido@gmail.com  | 2017-09-20
       2 | Lynn Galido        | lynngalido@gmail.com  | 1998-10-05
       3 | Erwin Galido       | erwingalido@gmail.com | 1996-02-19
(3 rows)

CREATE TABLE
INSERT 0 3
 user_id |     isbn      | loan_date  | return_date | book_loan_status
```



```
  ISBN
  genre
  publi
  quant
);

INSERT
VALUES
    ('F
    ('J
    ('N

SELECT


CREATE
  user_
  full_
  email
  membe
);
```

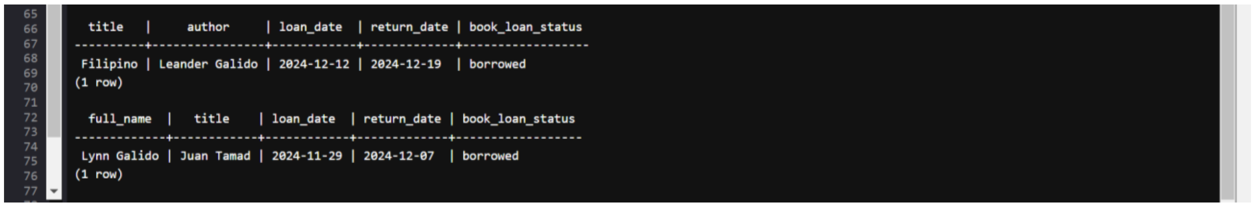
```
       2 | Lynn Galido        | lynngalido@gmail.com  | 1998-10-05
       3 | Erwin Galido       | erwingalido@gmail.com | 1996-02-19
(3 rows)

CREATE TABLE
INSERT 0 3
 user_id |     isbn      | loan_date  | return_date | book_loan_status
---------+---------------+------------+-------------+-----------------
       1 | 1432556712398 | 2024-12-12 | 2024-12-19  | borrowed
       2 | 3126752340985 | 2024-11-29 | 2024-12-07  | borrowed
       3 | 4537652348093 | 2024-10-29 | 2024-12-07  | overdue
(3 rows)

  title   |     author     | loan_date  | return_date | book_loan_status
----------+----------------+------------+-------------+-----------------
 Filipino | Leander Galido | 2024-12-12 | 2024-12-19  | borrowed
(1 row)
```

3. SQL Queries



```sql
-- borrowed books
SELECT b.title, b.author, bl.loan_date, bl.return_date, bl.book_loan_status
FROM book_loans bl
JOIN books b ON bl.ISBN = b.ISBN
WHERE bl.user_id = 1;

-- overdue books
SELECT u.full_name, b.title, bl.loan_date, bl.return_date, bl.book_loan_status
FROM book_loans bl
JOIN books b ON bl.ISBN = b.ISBN
JOIN users u ON bl.user_id = u.user_id
WHERE bl.return_date < CURRENT_DATE AND bl.book_loan_status = 'borrowed';
```

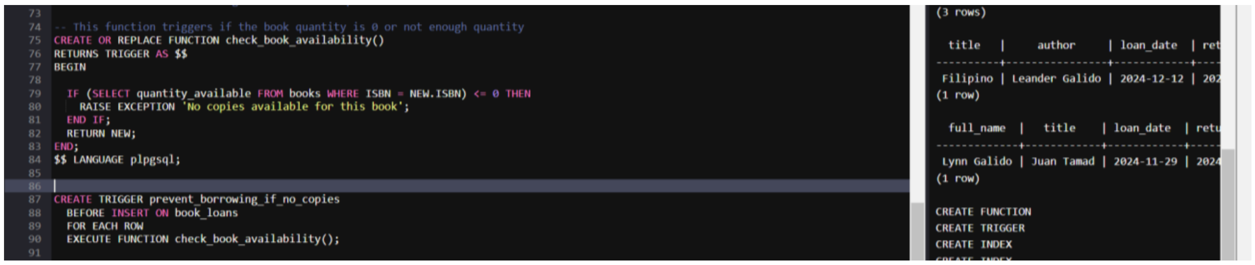
```
CREATE TABLE
INSERT 0 3
 user_id |     isbn      | loan_date  | r
---------+---------------+------------+--
       1 | 1432556712398 | 2024-12-12 | 2
       2 | 3126752340985 | 2024-11-29 | 2
       3 | 4537652348093 | 2024-10-29 | 2
(3 rows)

  title   |     author      | loan_date  |
----------+-----------------+------------+
 Filipino | Leander Galido  | 2024-12-12 |
(1 row)
```



```
  title   |     author     | loan_date  | return_date | book_loan_status
----------+----------------+------------+-------------+------------------
 Filipino | Leander Galido | 2024-12-12 | 2024-12-19  | borrowed
(1 row)

  full_name  |    title    | loan_date  | return_date | book_loan_status
-------------+-------------+------------+-------------+------------------
 Lynn Galido | Juan Tamad  | 2024-11-29 | 2024-12-07  | borrowed
(1 row)
```

4. Data Integrity and Optimization

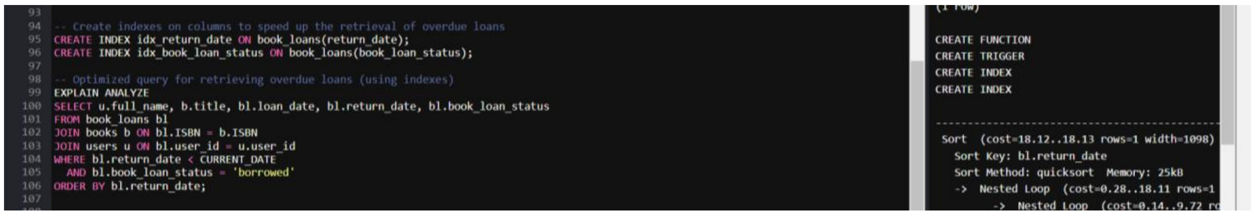
- **If the quantity of book is less than the quantity that user want to borrow, I will just put a function that displays a modal that the book quantity is not enough.**



```sql
-- This function triggers if the book quantity is 0 or not enough quantity
CREATE OR REPLACE FUNCTION check_book_availability()
RETURNS TRIGGER AS $$
BEGIN

  IF (SELECT quantity_available FROM books WHERE ISBN = NEW.ISBN) <= 0 THEN
    RAISE EXCEPTION 'No copies available for this book';
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER prevent_borrowing_if_no_copies
  BEFORE INSERT ON book_loans
  FOR EACH ROW
  EXECUTE FUNCTION check_book_availability();
```

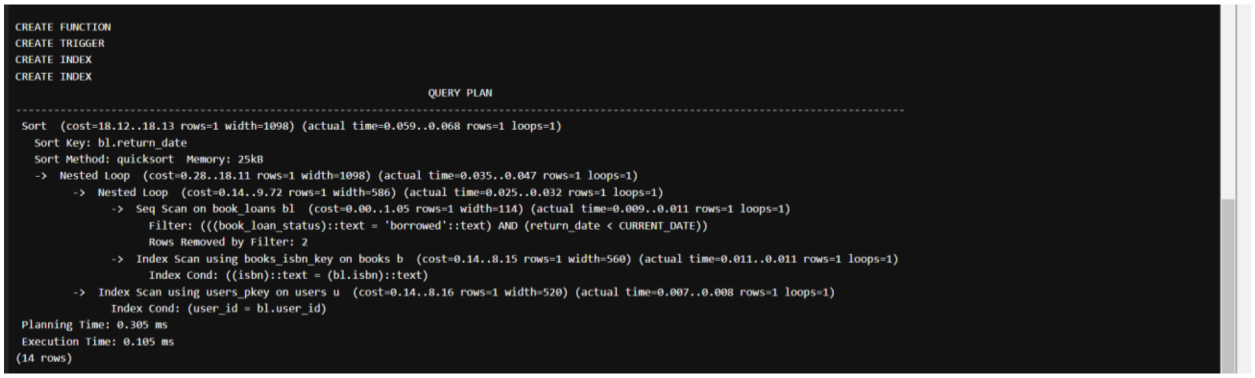
```
(3 rows)

  title   |     author     | loan_date  | ret
----------+----------------+------------+----
 Filipino | Leander Galido | 2024-12-12 | 202
(1 row)

  full_name  |   title    | loan_date  | retu
-------------+------------+------------+-----
 Lynn Galido | Juan Tamad | 2024-11-29 | 2024
(1 row)

CREATE FUNCTION
CREATE TRIGGER
CREATE INDEX
```

- **For fast data retrieval, indexes are first created on specific columns. When querying, the database uses these indexes to quickly locate relevant rows without scanning every element in the table, reducing the search time significantly.**



```sql
-- Create indexes on columns to speed up the retrieval of overdue loans
CREATE INDEX idx_return_date ON book_loans(return_date);
CREATE INDEX idx_book_loan_status ON book_loans(book_loan_status);

-- Optimized query for retrieving overdue loans (using indexes)
EXPLAIN ANALYZE
SELECT u.full_name, b.title, bl.loan_date, bl.return_date, bl.book_loan_status
FROM book_loans bl
JOIN books b ON bl.ISBN = b.ISBN
JOIN users u ON bl.user_id = u.user_id
WHERE bl.return_date < CURRENT_DATE
  AND bl.book_loan_status = 'borrowed'
ORDER BY bl.return_date;
```

```
CREATE FUNCTION
CREATE TRIGGER
CREATE INDEX
CREATE INDEX

-------------------------------------------------
 Sort  (cost=18.12..18.13 rows=1 width=1098)
   Sort Key: bl.return_date
   Sort Method: quicksort  Memory: 25kB
   ->  Nested Loop  (cost=0.28..18.11 rows=1
         ->  Nested Loop  (cost=0.14..9.72 ro
```



```
CREATE FUNCTION
CREATE TRIGGER
CREATE INDEX
CREATE INDEX
                                                            QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=18.12..18.13 rows=1 width=1098) (actual time=0.059..0.068 rows=1 loops=1)
   Sort Key: bl.return_date
   Sort Method: quicksort  Memory: 25kB
   ->  Nested Loop  (cost=0.28..18.11 rows=1 width=1098) (actual time=0.035..0.047 rows=1 loops=1)
         ->  Nested Loop  (cost=0.14..9.72 rows=1 width=586) (actual time=0.025..0.032 rows=1 loops=1)
               ->  Seq Scan on book_loans bl  (cost=0.00..1.05 rows=1 width=114) (actual time=0.009..0.011 rows=1 loops=1)
                     Filter: (((book_loan_status)::text = 'borrowed'::text) AND (return_date < CURRENT_DATE))
                     Rows Removed by Filter: 2
               ->  Index Scan using books_isbn_key on books b  (cost=0.14..8.15 rows=1 width=560) (actual time=0.011..0.011 rows=1 loops=1)
                     Index Cond: ((isbn)::text = (bl.isbn)::text)
         ->  Index Scan using users_pkey on users u  (cost=0.14..8.16 rows=1 width=520) (actual time=0.007..0.008 rows=1 loops=1)
               Index Cond: (user_id = bl.user_id)
 Planning Time: 0.305 ms
 Execution Time: 0.105 ms
(14 rows)
```

Part 5. Reflection

My experience in doing this activity is very useful it helps me to understand more on how to get/retrieve the data more faster than usual, it also improve my skills in using the postgresql because of the conditions applied into it like rules like you cannot borrow a book if it has zero quantity, at first it is somehow hard but with the help of searching and YouTube tutorials it help me finished this laboratory activity.