Instructions:

1. Read the case study carefully and answer the questions based on the requirements described.
2. Use **ER diagrams**, **SQL schema definitions**, and written explanations where applicable.
3. Complete the exam by **12/11/2024 19:00**.

Case Study:

You have been tasked to design a database for an **Online Library Management System**. The system should keep track of books, users, and book loans. Below are the requirements:

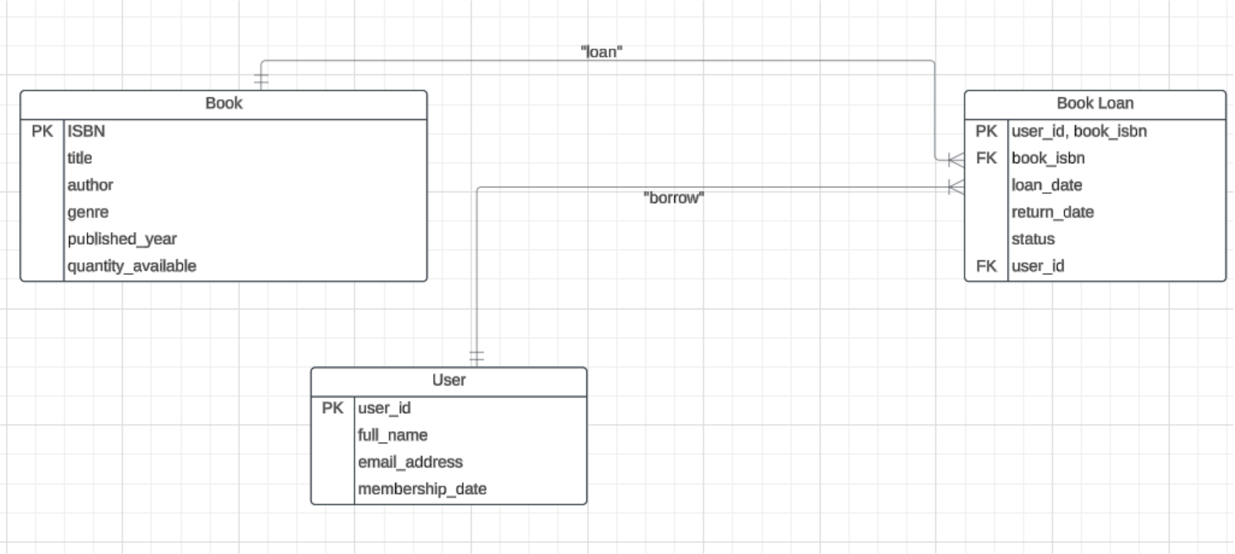
1. **Books**: The library has a collection of books. Each book has the following details:
   - Title
   - Author
   - ISBN (unique identifier)
   - Genre (e.g., Fiction, Non-Fiction)
   - Published Year
   - Quantity Available
2. **Users**: Users of the library can borrow books. Each user has:
   - A unique ID
   - Full Name
   - Email Address
   - Membership Date
3. **Book Loans**: Users can borrow books. Each loan should record:
   - User ID
   - Book ISBN
   - Loan Date
   - Return Date
   - Status (e.g., "borrowed", "returned", "overdue")
4. **Rules**:
   - A user can borrow multiple books, but the loan status must be updated when books are returned.
   - The library should not allow loans for unavailable books (i.e., if all copies of a book are borrowed).

Part 1: Conceptual Design - 25pts

1. Draw an **Entity-Relationship (ER) Diagram** for the system based on the given

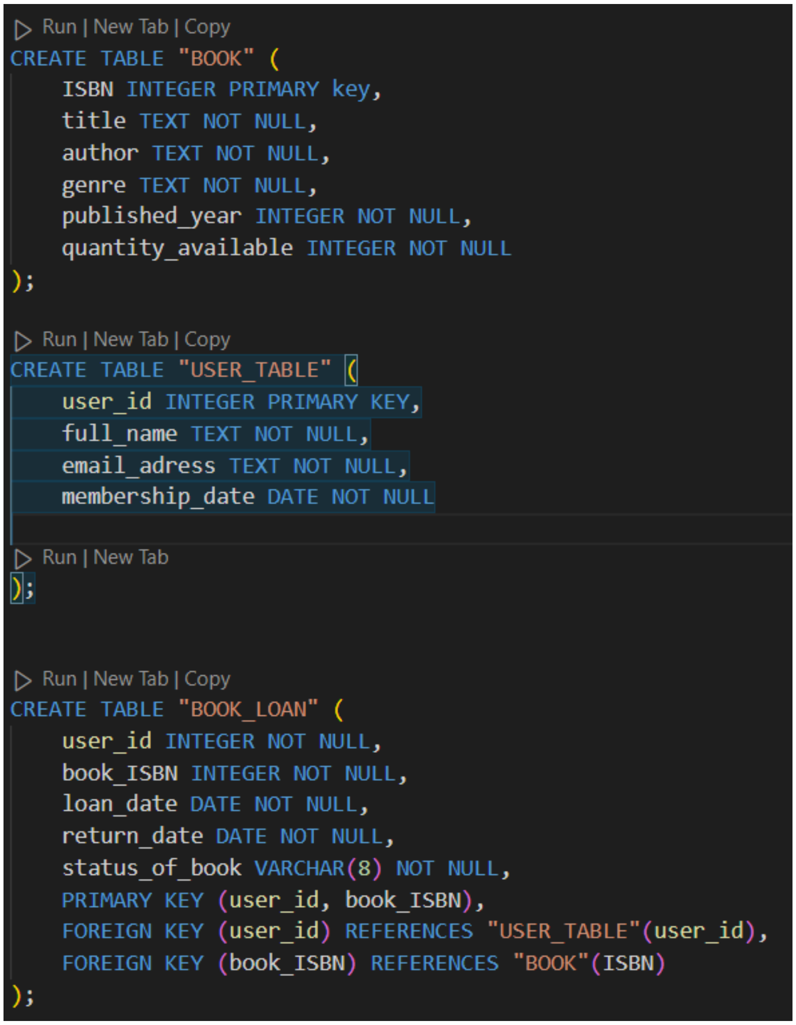
requirements. Ensure you specify:

- ○ Entities- Book, Book Loan, User
- ○ Attributes - title, author, genre, published_year, quantity_available, user_id, full_name, email_address, membership_date, loan_id, loan_date, return_date, status
- ○ Primary Keys - ISBN, user_id,
- ○ Relationships with cardinalities (e.g., one-to-many, many-to-many)



Part 2: Logical Design - 25pts

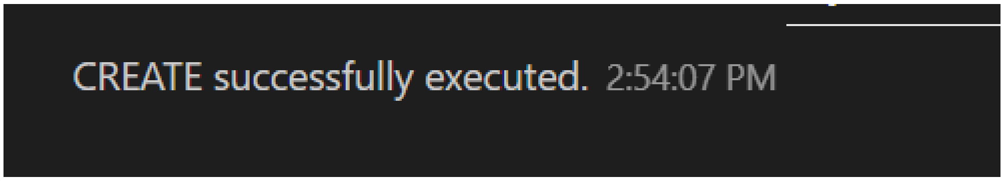
2. Translate the ER diagram into relational tables. Define:
   - ○ Table schemas (list all attributes, data types, and constraints such as primary keys, foreign keys, and NOT NULL).



```
CREATE TABLE "BOOK" (
    ISBN INTEGER PRIMARY key,
    title TEXT NOT NULL,
    author TEXT NOT NULL,
    genre TEXT NOT NULL,
    published_year INTEGER NOT NULL,
    quantity_available INTEGER NOT NULL
);

CREATE TABLE "USER_TABLE" (
    user_id INTEGER PRIMARY KEY,
    full_name TEXT NOT NULL,
    email_adress TEXT NOT NULL,
    membership_date DATE NOT NULL

);


CREATE TABLE "BOOK_LOAN" (
    user_id INTEGER NOT NULL,
    book_ISBN INTEGER NOT NULL,
    loan_date DATE NOT NULL,
    return_date DATE NOT NULL,
    status_of_book VARCHAR(8) NOT NULL,
    PRIMARY KEY (user_id, book_ISBN),
    FOREIGN KEY (user_id) REFERENCES "USER_TABLE"(user_id),
    FOREIGN KEY (book_ISBN) REFERENCES "BOOK"(ISBN)
);
```

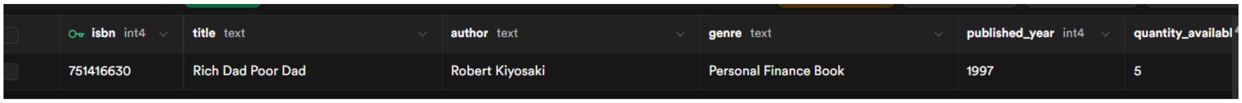
Part 3: SQL Queries

3. Write SQL queries for the following scenarios (15pts each):

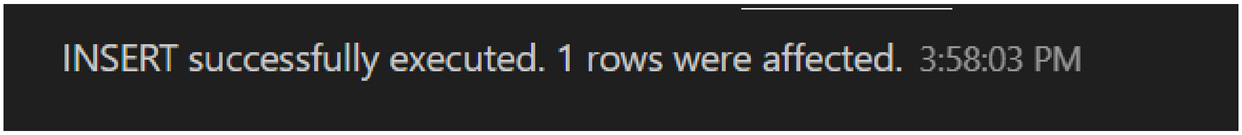


```
CREATE successfully executed. 2:54:07 PM
```

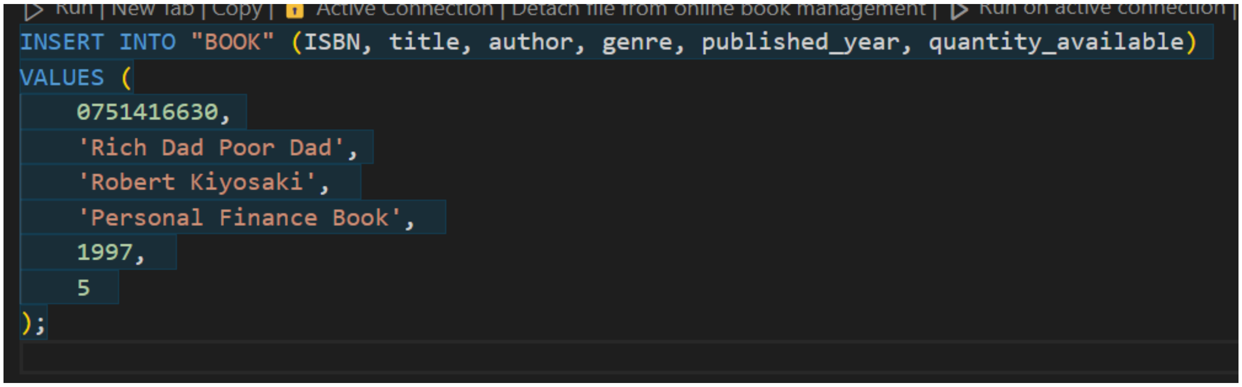
○ a. Insert a new book into the library with a quantity of 5.



isbn int4	title text	author text	genre text	published_year int4	quantity_availabl
751416630	Rich Dad Poor Dad	Robert Kiyosaki	Personal Finance Book	1997	5



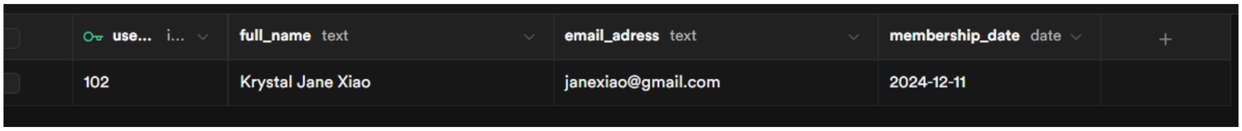
```
INSERT successfully executed. 1 rows were affected. 3:58:03 PM
```



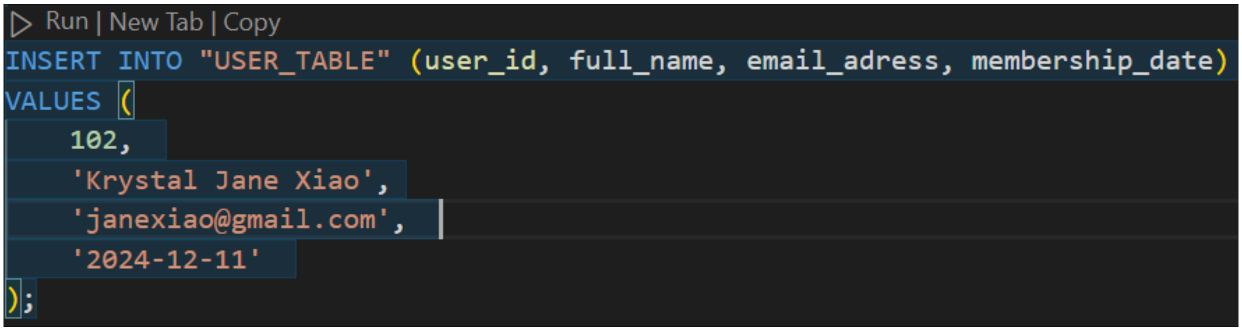
▷ Run | New Tab | Copy | Active Connection | Detach file from online book management | ▷ Run on active connection |

```sql
INSERT INTO "BOOK" (ISBN, title, author, genre, published_year, quantity_available)
VALUES (
    0751416630,
    'Rich Dad Poor Dad',
    'Robert Kiyosaki',
    'Personal Finance Book',
    1997,
    5
);
```

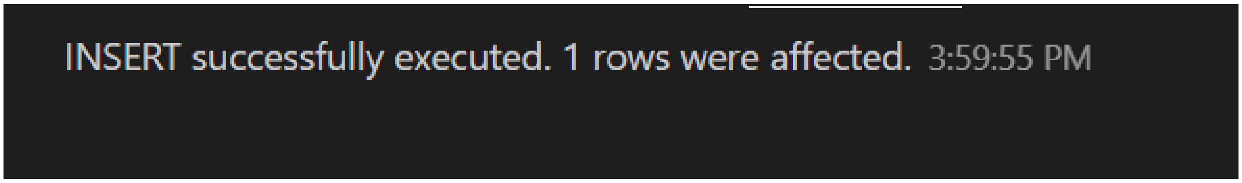
○ b. Add a new user to the system.



use... i...	full_name text	email_adress text	membership_date date
102	Krystal Jane Xiao	janexiao@gmail.com	2024-12-11

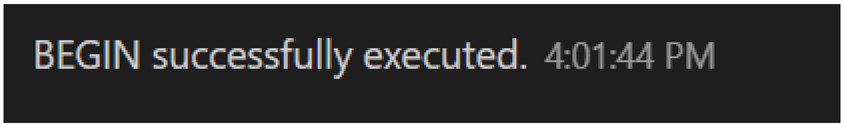


```sql
INSERT INTO "USER_TABLE" (user_id, full_name, email_adress, membership_date)
VALUES (
    102,
    'Krystal Jane Xiao',
    'janexiao@gmail.com',
    '2024-12-11'
);
```

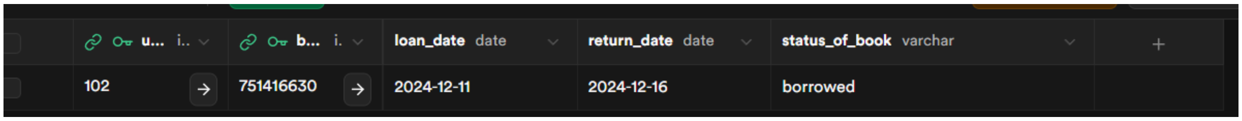


```
INSERT successfully executed. 1 rows were affected. 3:59:55 PM
```

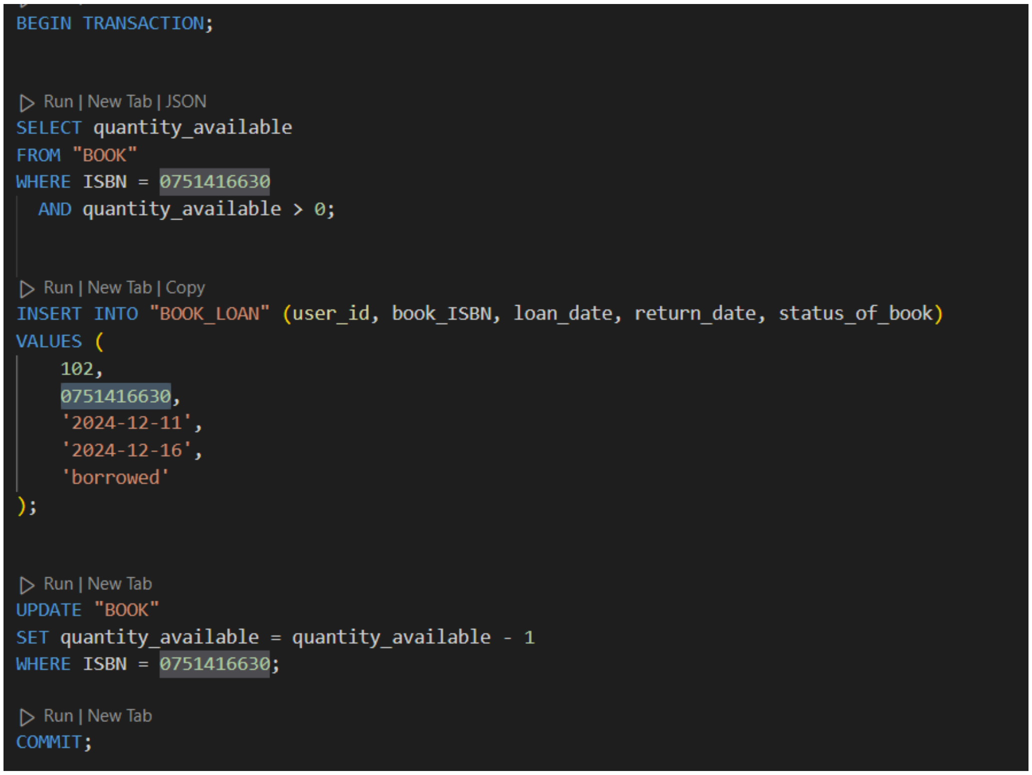
○ c. Record a book loan for a user.



```
BEGIN successfully executed. 4:01:44 PM
```

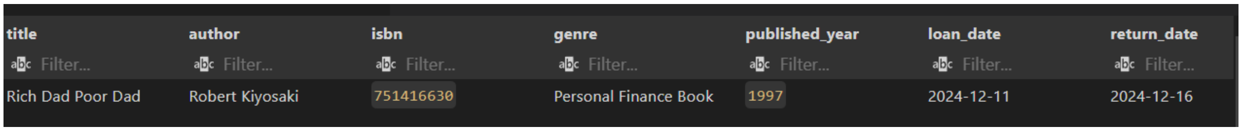


u... i..	b... i.	loan_date date	return_date date	status_of_book varchar
102	751416630	2024-12-11	2024-12-16	borrowed

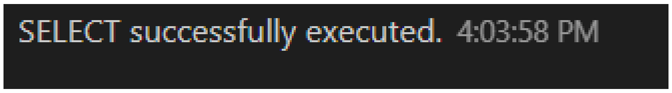


```
BEGIN TRANSACTION;

SELECT quantity_available
FROM "BOOK"
WHERE ISBN = 0751416630
  AND quantity_available > 0;

INSERT INTO "BOOK_LOAN" (user_id, book_ISBN, loan_date, return_date, status_of_book)
VALUES (
    102,
    0751416630,
    '2024-12-11',
    '2024-12-16',
    'borrowed'
);

UPDATE "BOOK"
SET quantity_available = quantity_available - 1
WHERE ISBN = 0751416630;

COMMIT;
```

○ d. Find all books borrowed by a specific user.

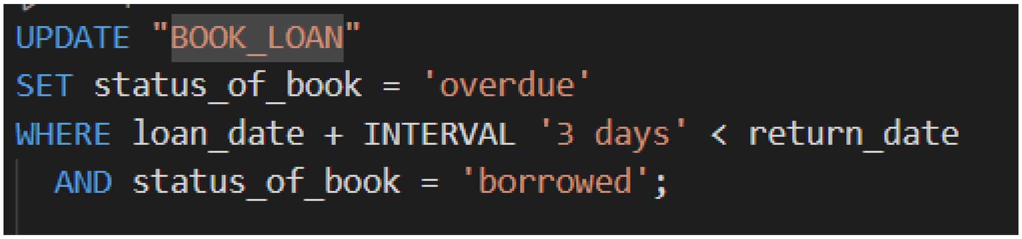


title	author	isbn	genre	published_year	loan_date	return_date
Rich Dad Poor Dad	Robert Kiyosaki	751416630	Personal Finance Book	1997	2024-12-11	2024-12-16

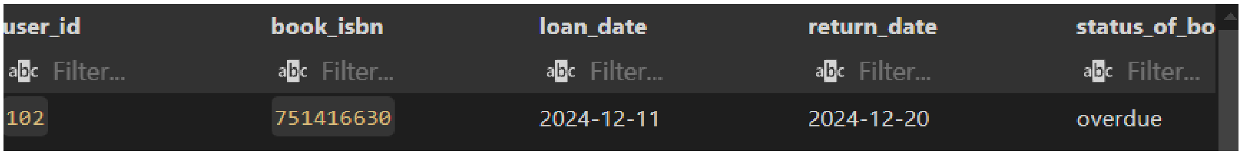


```
SELECT successfully executed. 4:03:58 PM
```

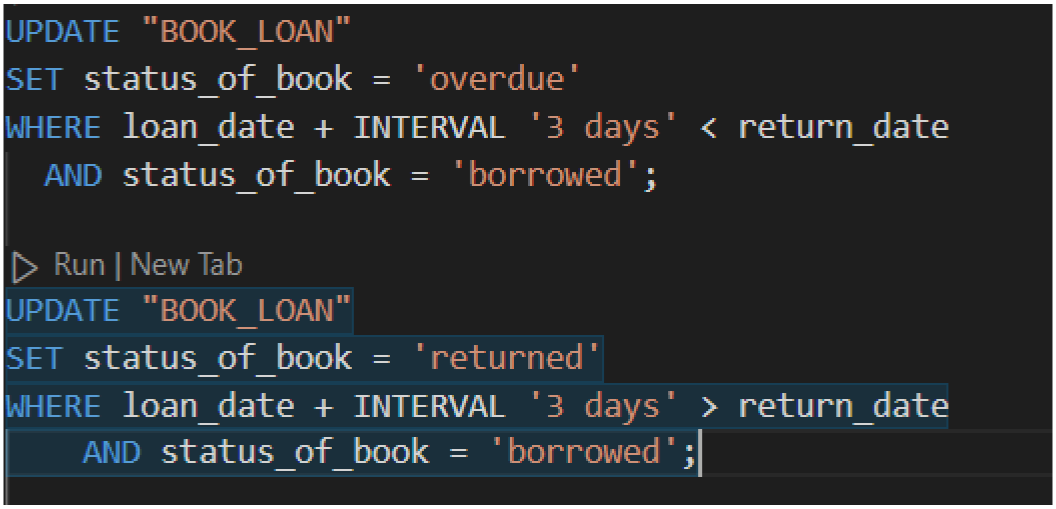
○ e. List all overdue loans.



```
UPDATE "BOOK_LOAN"
SET status_of_book = 'overdue'
WHERE loan_date + INTERVAL '3 days' < return_date
  AND status_of_book = 'borrowed';
```



user_id	book_isbn	loan_date	return_date	status_of_bo
Filter...	Filter...	Filter...	Filter...	Filter...
102	751416630	2024-12-11	2024-12-20	overdue

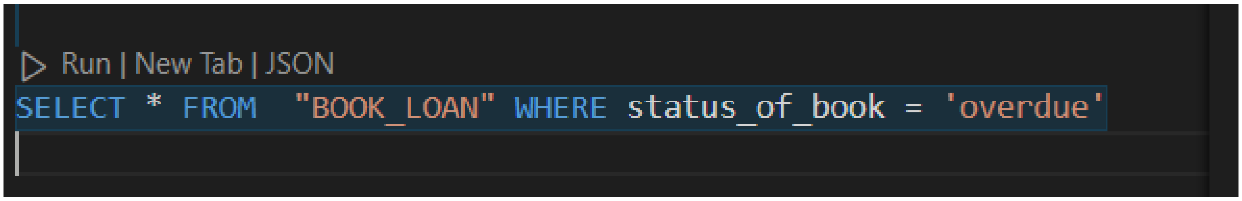


```
UPDATE "BOOK_LOAN"
SET status_of_book = 'overdue'
WHERE loan_date + INTERVAL '3 days' < return_date
  AND status_of_book = 'borrowed';

▷ Run | New Tab
UPDATE "BOOK_LOAN"
SET status_of_book = 'returned'
WHERE loan_date + INTERVAL '3 days' > return_date
    AND status_of_book = 'borrowed';
```

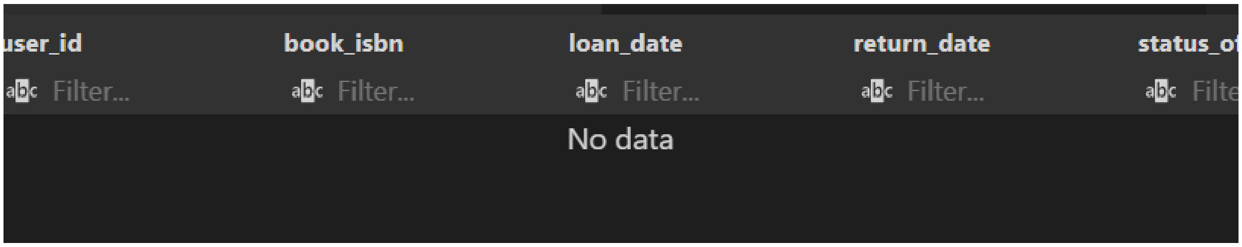
Part 4: Data Integrity and Optimization

4. Explain how you would ensure:

○ The prevention of borrowing books when no copies are available. (15 pts) ○ Fast retrieval of overdue loans. (20 pts - with CODE and actual screenshot of performance)

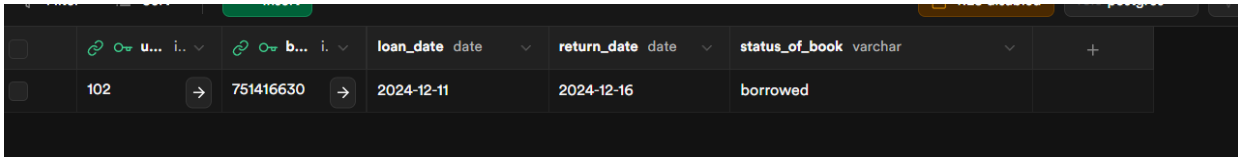


```
▷ Run | New Tab | JSON
SELECT * FROM  "BOOK_LOAN" WHERE status_of_book = 'overdue'
```



user_id	book_isbn	loan_date	return_date	status_of
Filter...	Filter...	Filter...	Filter...	Filte

No data



u... i..	b... i.	loan_date date	return_date date	status_of_book varchar
102	751416630	2024-12-11	2024-12-16	borrowed

I would ensure the borrowing of books when no copies are available by checking the quantity of the book available and if it’s not available, then I won’t let the user borrow the book. If the book quantity is greater than zero, then I would let the user borrow it.

For a fast retrieval of overdue loans, I would just simply check the status of the book in the table Book loan and return all the books that have the status of overdue

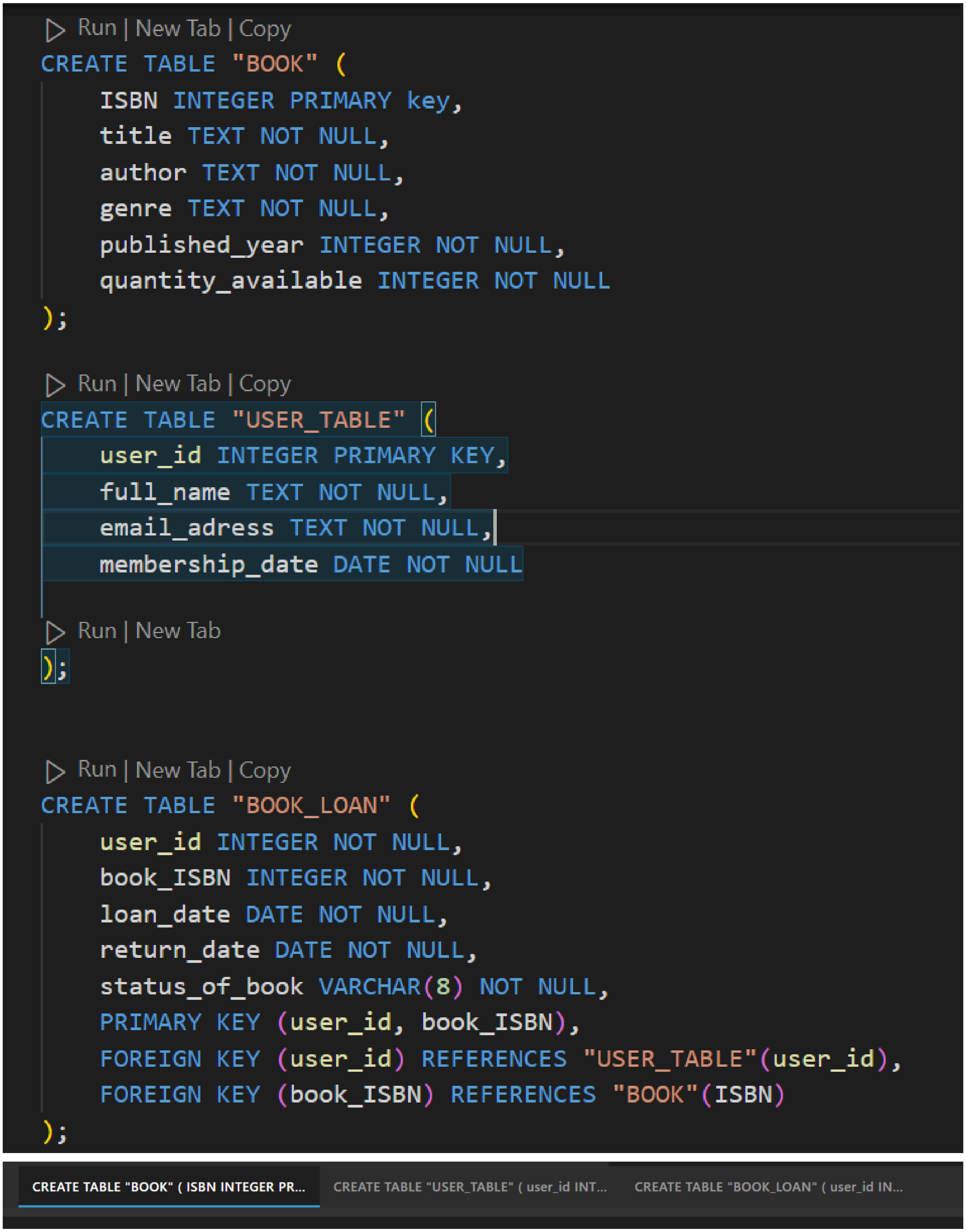
Part 5: Reflection (25 pts)

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

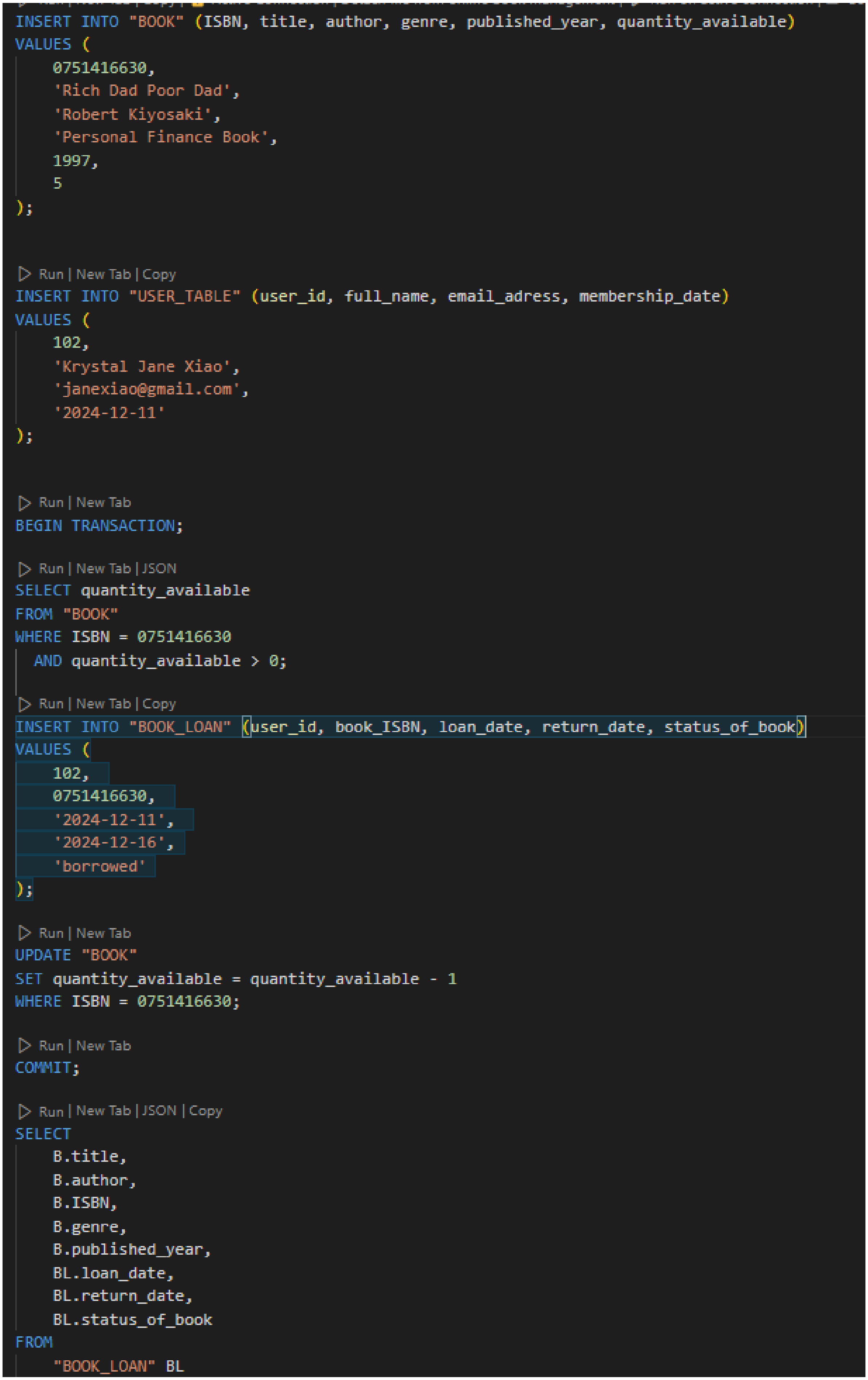
    1. Slow performance - a solution for this is to create more manageable segments of a large table in order to improve and make the query performance faster which will make a better user experience
    2. Scalability - In order to solve this issue, we need to add more servers in the database in order to accommodate all the data and expand the system. It is also advisable to add more nodes and apply the concept of caching, which is a temporary storage layer.
    3. Problems with data integrity- by implementing database replication techniques, data can make large-scale systems scalable, fault-tolerant, and highly available. ACID Transactions can also be a good solution for this problem because despite during failures, it can ensure data integrity and consistency.

Deliverables:

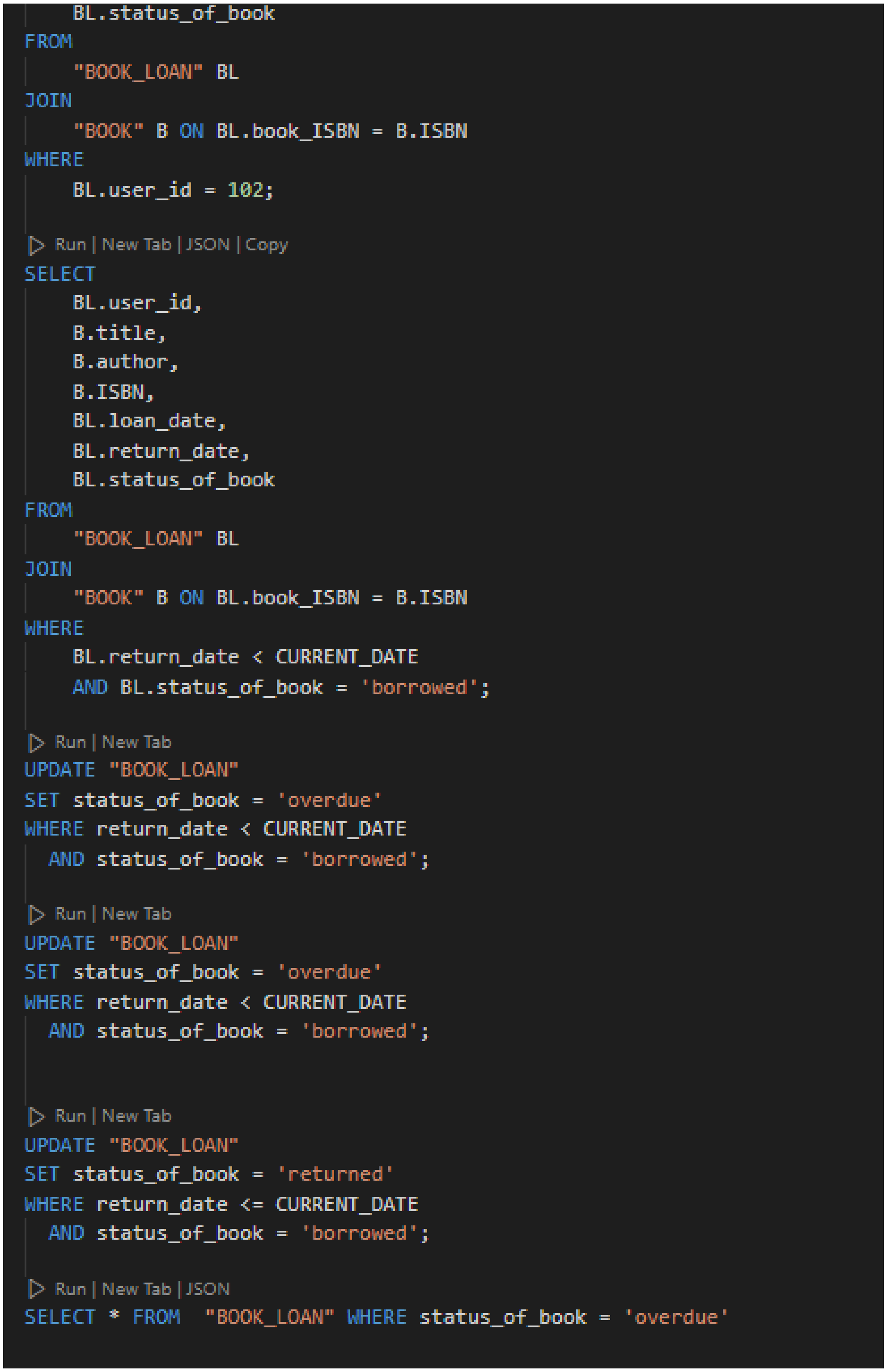
- ER Diagram (hand-drawn or created using software).
- SQL table definitions and queries.
- Written responses to conceptual and reflection questions.
- Any assumptions you made.


▷ Run | New Tab | Copy
▷ Run | New Tab | Copy
▷ Run | New Tab
▷ Run | New Tab | Copy
CREATE TABLE "BOOK" (ISBN INTEGER PR...
CREATE TABLE "USER_TABLE" (user_id INT...
CREATE TABLE "BOOK_LOAN" (user_id IN...

```
CREATE TABLE "BOOK" (
    ISBN INTEGER PRIMARY key,
    title TEXT NOT NULL,
    author TEXT NOT NULL,
    genre TEXT NOT NULL,
    published_year INTEGER NOT NULL,
    quantity_available INTEGER NOT NULL
);

CREATE TABLE "USER_TABLE" (
    user_id INTEGER PRIMARY KEY,
    full_name TEXT NOT NULL,
    email_adress TEXT NOT NULL,
    membership_date DATE NOT NULL

);


CREATE TABLE "BOOK_LOAN" (
    user_id INTEGER NOT NULL,
    book_ISBN INTEGER NOT NULL,
    loan_date DATE NOT NULL,
    return_date DATE NOT NULL,
    status_of_book VARCHAR(8) NOT NULL,
    PRIMARY KEY (user_id, book_ISBN),
    FOREIGN KEY (user_id) REFERENCES "USER_TABLE"(user_id),
    FOREIGN KEY (book_ISBN) REFERENCES "BOOK"(ISBN)
);
```



```sql
INSERT INTO "BOOK" (ISBN, title, author, genre, published_year, quantity_available)
VALUES (
    0751416630,
    'Rich Dad Poor Dad',
    'Robert Kiyosaki',
    'Personal Finance Book',
    1997,
    5
);


INSERT INTO "USER_TABLE" (user_id, full_name, email_adress, membership_date)
VALUES (
    102,
    'Krystal Jane Xiao',
    'janexiao@gmail.com',
    '2024-12-11'
);


BEGIN TRANSACTION;

SELECT quantity_available
FROM "BOOK"
WHERE ISBN = 0751416630
  AND quantity_available > 0;

INSERT INTO "BOOK_LOAN" (user_id, book_ISBN, loan_date, return_date, status_of_book)
VALUES (
    102,
    0751416630,
    '2024-12-11',
    '2024-12-16',
    'borrowed'
);

UPDATE "BOOK"
SET quantity_available = quantity_available - 1
WHERE ISBN = 0751416630;

COMMIT;

SELECT
    B.title,
    B.author,
    B.ISBN,
    B.genre,
    B.published_year,
    BL.loan_date,
    BL.return_date,
    BL.status_of_book
FROM
    "BOOK_LOAN" BL
```



```
    BL.status_of_book
FROM
    "BOOK_LOAN" BL
JOIN
    "BOOK" B ON BL.book_ISBN = B.ISBN
WHERE
    BL.user_id = 102;

▷ Run | New Tab | JSON | Copy
SELECT
    BL.user_id,
    B.title,
    B.author,
    B.ISBN,
    BL.loan_date,
    BL.return_date,
    BL.status_of_book
FROM
    "BOOK_LOAN" BL
JOIN
    "BOOK" B ON BL.book_ISBN = B.ISBN
WHERE
    BL.return_date < CURRENT_DATE
    AND BL.status_of_book = 'borrowed';

▷ Run | New Tab
UPDATE "BOOK_LOAN"
SET status_of_book = 'overdue'
WHERE return_date < CURRENT_DATE
  AND status_of_book = 'borrowed';

▷ Run | New Tab
UPDATE "BOOK_LOAN"
SET status_of_book = 'overdue'
WHERE return_date < CURRENT_DATE
  AND status_of_book = 'borrowed';


▷ Run | New Tab
UPDATE "BOOK_LOAN"
SET status_of_book = 'returned'
WHERE return_date <= CURRENT_DATE
  AND status_of_book = 'borrowed';

▷ Run | New Tab | JSON
SELECT * FROM  "BOOK_LOAN" WHERE status_of_book = 'overdue'
```