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.

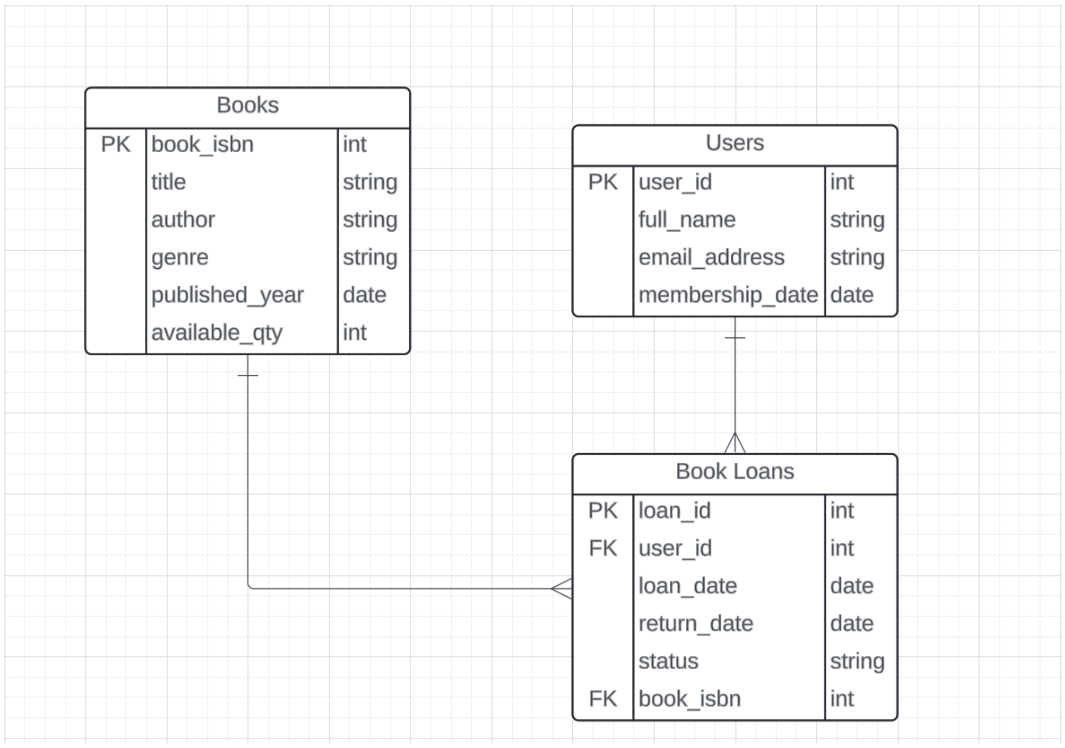
Case Study:

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 is 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
   ○ Attributes
   ○ Primary Keys
   ○ Relationships with cardinalities (e.g., one-to-many, many-to-many)



Books

PK	book_isbn	int
	title	string
	author	string
	genre	string
	published_year	date
	available_qty	int

Users

PK	user_id	int
	full_name	string
	email_address	string
	membership_date	date

Book Loans

PK	loan_id	int
FK	user_id	int
	loan_date	date
	return_date	date
	status	string
FK	book_isbn	int

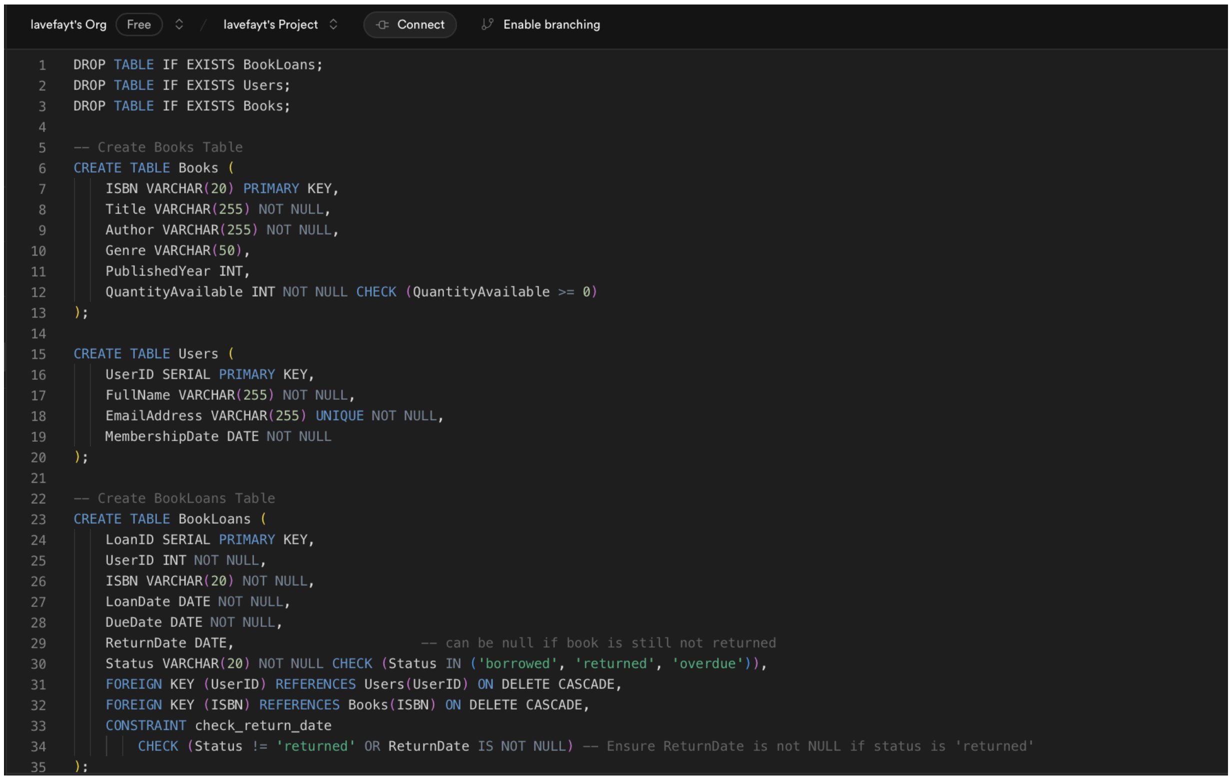
Books to BookLoans
- One to Many relationship. One book can have multiple loans but each loan references one specific loan

User to BookLoans
- One to Many relationship. One user can create multiple book loans, but each loan is linked to one specific user

Keys:
The PK of the entity Books is book_isbn because it uniquely identifies each book.
The PK of the entity Users is user_id because it uniquely identifies each user.
I added loan_id and assigned it as the PK because it uniquely identifies each loan record
Book Loans have two FKs which are user_id and book_isbn because user_id references the user id in the Users table and book_isbn references the book_isbn in the Books table

Part 2: Logical Design - 25pts

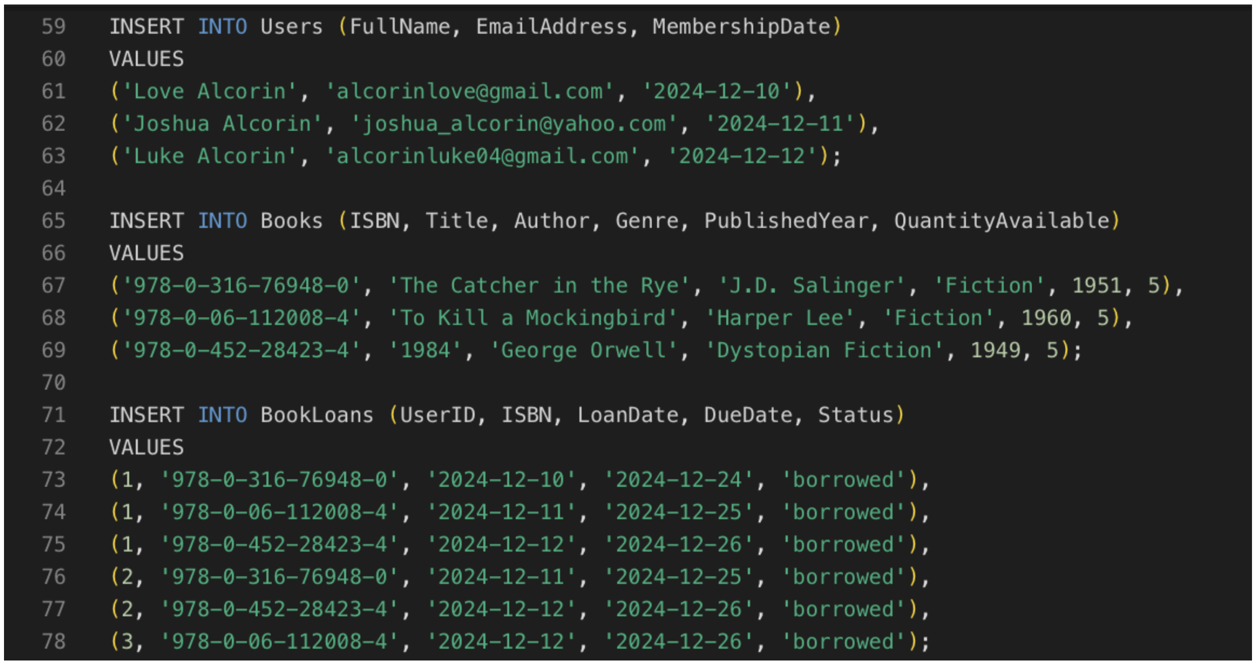
2. Translate the ER diagram into relational tables. Define:
   ○ Table schemas (list all attributes, data types, and constraints such as primary keys, foreign keys, and NOT NULL).



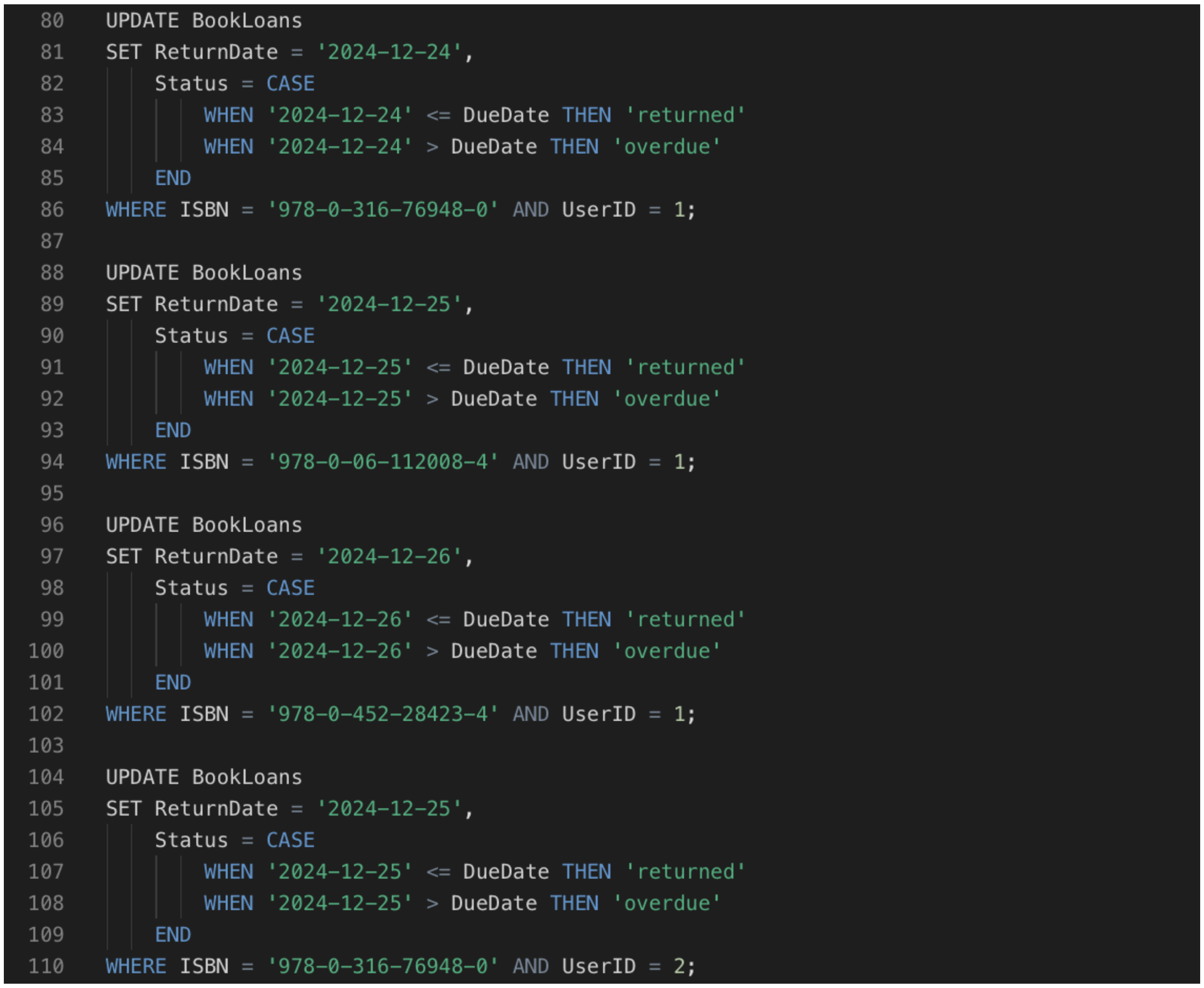
```sql
DROP TABLE IF EXISTS BookLoans;
DROP TABLE IF EXISTS Users;
DROP TABLE IF EXISTS Books;

-- Create Books Table
CREATE TABLE Books (
    ISBN VARCHAR(20) PRIMARY KEY,
    Title VARCHAR(255) NOT NULL,
    Author VARCHAR(255) NOT NULL,
    Genre VARCHAR(50),
    PublishedYear INT,
    QuantityAvailable INT NOT NULL CHECK (QuantityAvailable >= 0)
);

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

-- Create BookLoans Table
CREATE TABLE BookLoans (
    LoanID SERIAL PRIMARY KEY,
    UserID INT NOT NULL,
    ISBN VARCHAR(20) NOT NULL,
    LoanDate DATE NOT NULL,
    DueDate DATE NOT NULL,
    ReturnDate DATE,                           -- can be null if book is still not returned
    Status VARCHAR(20) NOT NULL CHECK (Status IN ('borrowed', 'returned', 'overdue')),
    FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE,
    FOREIGN KEY (ISBN) REFERENCES Books(ISBN) ON DELETE CASCADE,
    CONSTRAINT check_return_date
        CHECK (Status != 'returned' OR ReturnDate IS NOT NULL) -- Ensure ReturnDate is not NULL if status is 'returned'
);
```

Part 3: SQL Queries

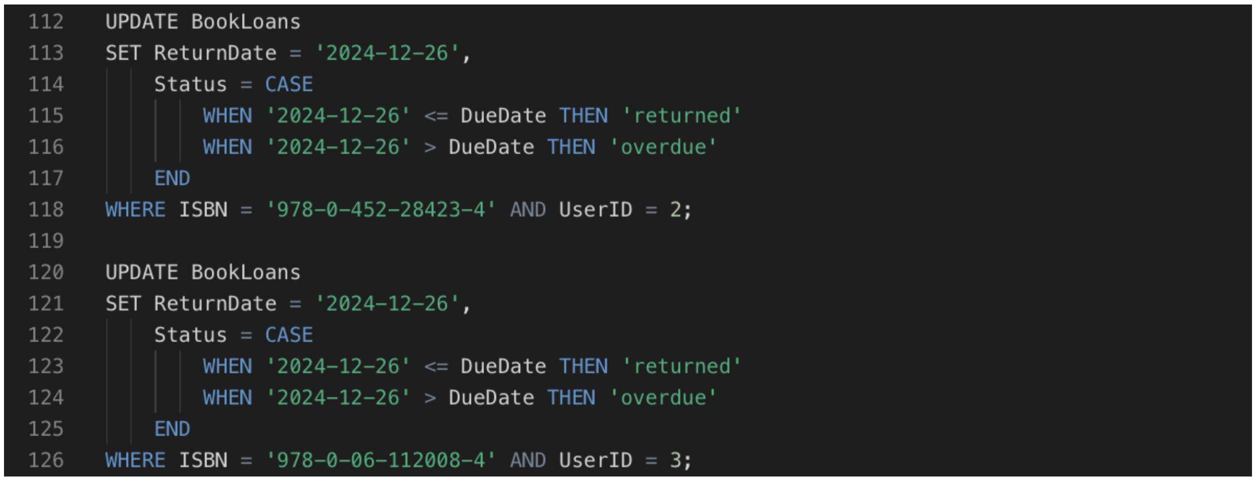
3. Write SQL queries for the following scenarios (15pts each):
   ○ a. Insert a new book into the library with a quantity of 5.
   ○ b. Add a new user to the system.
   ○ c. Record a book loan for a user.
   ○ d. Find all books borrowed by a specific user.
   ○ e. List all overdue loans.



```sql
INSERT INTO Users (FullName, EmailAddress, MembershipDate)
VALUES
('Love Alcorin', 'alcorinlove@gmail.com', '2024-12-10'),
('Joshua Alcorin', 'joshua_alcorin@yahoo.com', '2024-12-11'),
('Luke Alcorin', 'alcorinluke04@gmail.com', '2024-12-12');

INSERT INTO Books (ISBN, Title, Author, Genre, PublishedYear, QuantityAvailable)
VALUES
('978-0-316-76948-0', 'The Catcher in the Rye', 'J.D. Salinger', 'Fiction', 1951, 5),
('978-0-06-112008-4', 'To Kill a Mockingbird', 'Harper Lee', 'Fiction', 1960, 5),
('978-0-452-28423-4', '1984', 'George Orwell', 'Dystopian Fiction', 1949, 5);

INSERT INTO BookLoans (UserID, ISBN, LoanDate, DueDate, Status)
VALUES
(1, '978-0-316-76948-0', '2024-12-10', '2024-12-24', 'borrowed'),
(1, '978-0-06-112008-4', '2024-12-11', '2024-12-25', 'borrowed'),
(1, '978-0-452-28423-4', '2024-12-12', '2024-12-26', 'borrowed'),
(2, '978-0-316-76948-0', '2024-12-11', '2024-12-25', 'borrowed'),
(2, '978-0-452-28423-4', '2024-12-12', '2024-12-26', 'borrowed'),
(3, '978-0-06-112008-4', '2024-12-12', '2024-12-26', 'borrowed');
```



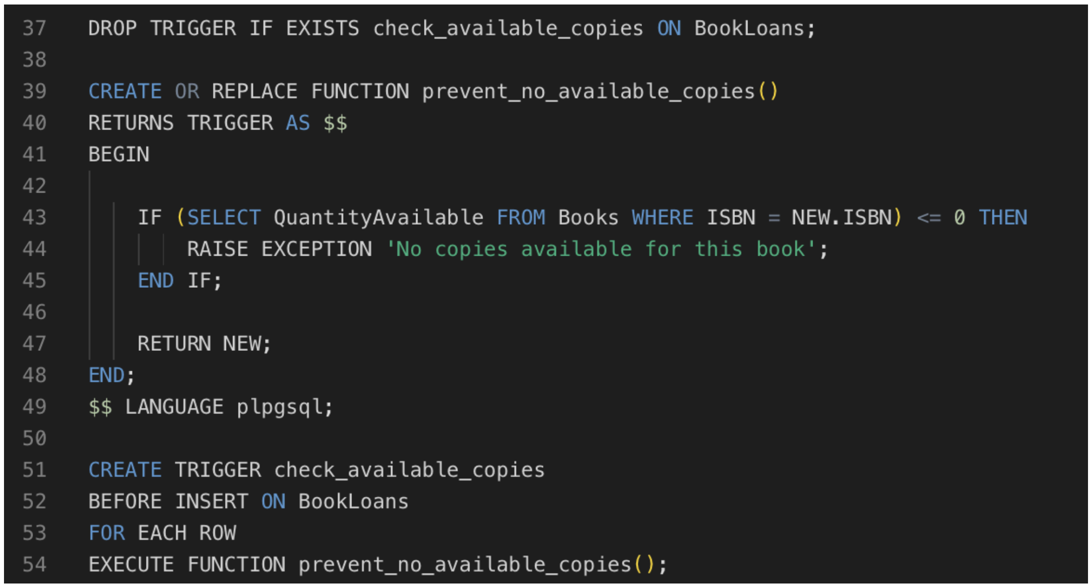
```sql
UPDATE BookLoans
SET ReturnDate = '2024-12-24',
    Status = CASE
        WHEN '2024-12-24' <= DueDate THEN 'returned'
        WHEN '2024-12-24' > DueDate THEN 'overdue'
    END
WHERE ISBN = '978-0-316-76948-0' AND UserID = 1;

UPDATE BookLoans
SET ReturnDate = '2024-12-25',
    Status = CASE
        WHEN '2024-12-25' <= DueDate THEN 'returned'
        WHEN '2024-12-25' > DueDate THEN 'overdue'
    END
WHERE ISBN = '978-0-06-112008-4' AND UserID = 1;

UPDATE BookLoans
SET ReturnDate = '2024-12-26',
    Status = CASE
        WHEN '2024-12-26' <= DueDate THEN 'returned'
        WHEN '2024-12-26' > DueDate THEN 'overdue'
    END
WHERE ISBN = '978-0-452-28423-4' AND UserID = 1;

UPDATE BookLoans
SET ReturnDate = '2024-12-25',
    Status = CASE
        WHEN '2024-12-25' <= DueDate THEN 'returned'
        WHEN '2024-12-25' > DueDate THEN 'overdue'
    END
WHERE ISBN = '978-0-316-76948-0' AND UserID = 2;
```



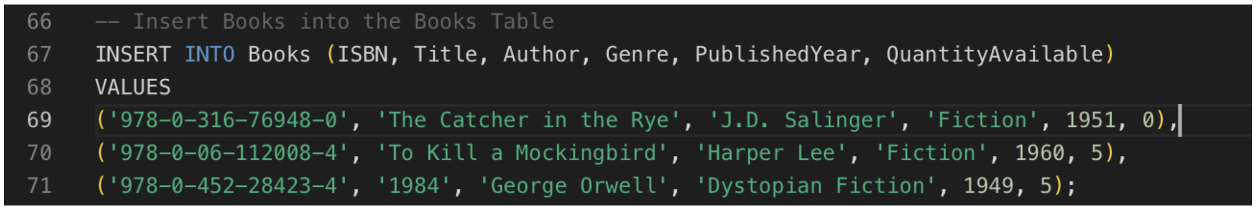
```sql
UPDATE BookLoans
SET ReturnDate = '2024-12-26',
    Status = CASE
        WHEN '2024-12-26' <= DueDate THEN 'returned'
        WHEN '2024-12-26' > DueDate THEN 'overdue'
    END
WHERE ISBN = '978-0-452-28423-4' AND UserID = 2;

UPDATE BookLoans
SET ReturnDate = '2024-12-26',
    Status = CASE
        WHEN '2024-12-26' <= DueDate THEN 'returned'
        WHEN '2024-12-26' > DueDate THEN 'overdue'
    END
WHERE ISBN = '978-0-06-112008-4' AND UserID = 3;
```

Part 4: Data Integrity and Optimization

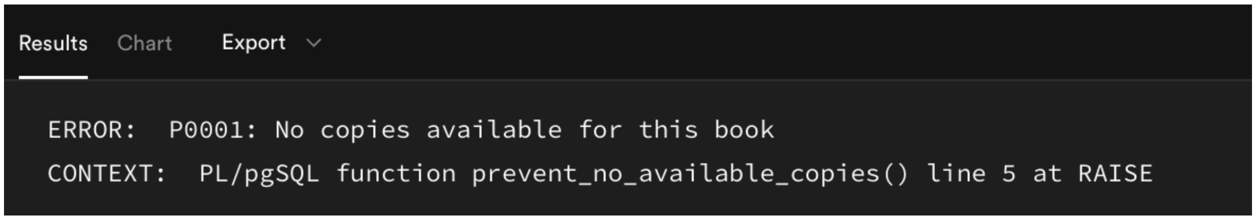
4. Explain how you would ensure:
   ○ The prevention of borrowing books when no copies are available. (15 pts)
   ○ Fast retrieval of overdue loans. (20 pts - with CODE and actual screenshot of performance)



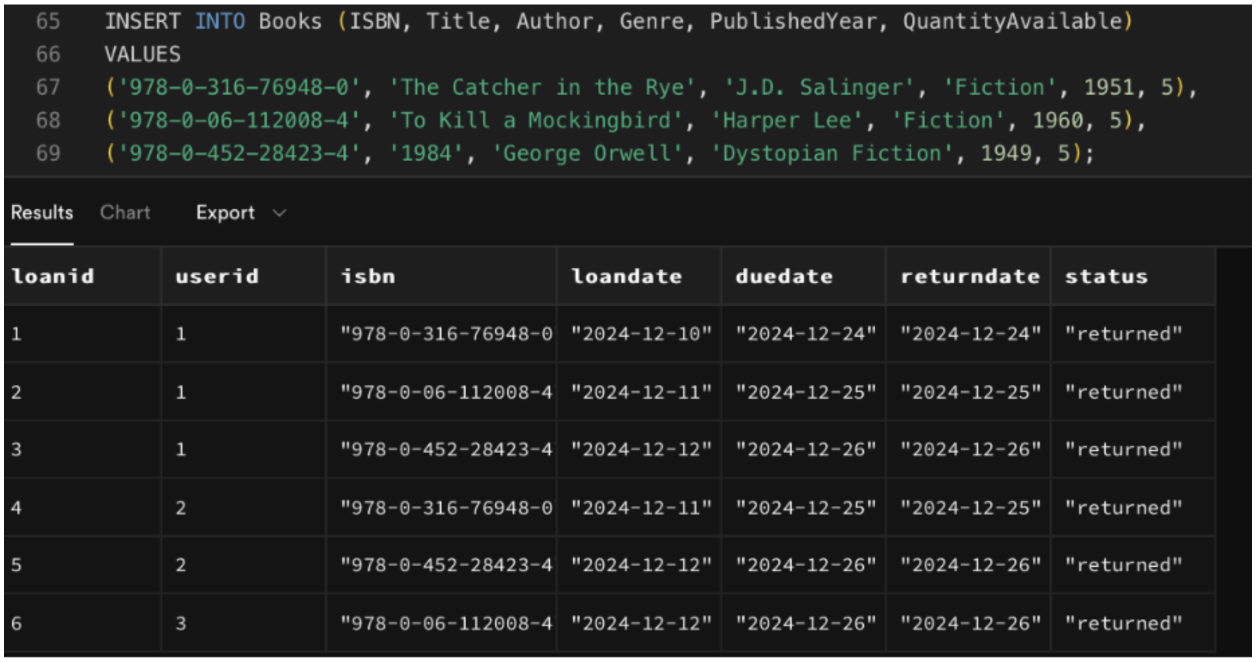
```sql
DROP TRIGGER IF EXISTS check_available_copies ON BookLoans;

CREATE OR REPLACE FUNCTION prevent_no_available_copies()
RETURNS TRIGGER AS $$
BEGIN

    IF (SELECT QuantityAvailable 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 check_available_copies
BEFORE INSERT ON BookLoans
FOR EACH ROW
EXECUTE FUNCTION prevent_no_available_copies();
```



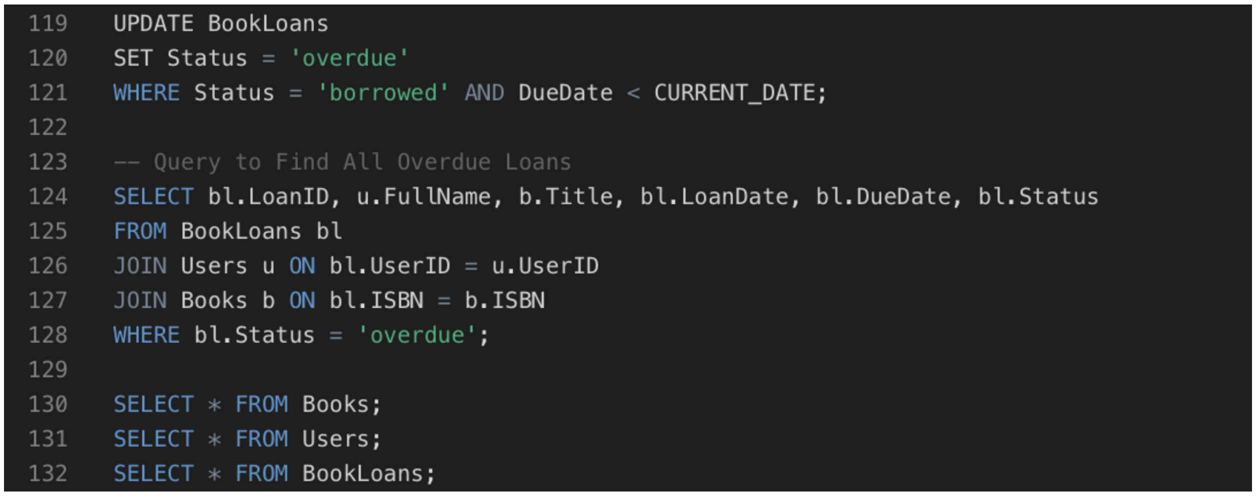
```sql
-- Insert Books into the Books Table
INSERT INTO Books (ISBN, Title, Author, Genre, PublishedYear, QuantityAvailable)
VALUES
('978-0-316-76948-0', 'The Catcher in the Rye', 'J.D. Salinger', 'Fiction', 1951, 0),
('978-0-06-112008-4', 'To Kill a Mockingbird', 'Harper Lee', 'Fiction', 1960, 5),
('978-0-452-28423-4', '1984', 'George Orwell', 'Dystopian Fiction', 1949, 5);
```



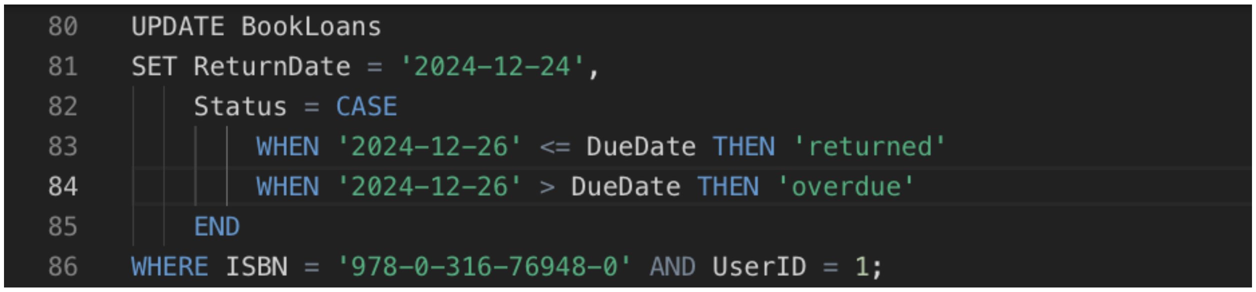
```
ERROR:  P0001: No copies available for this book
CONTEXT:  PL/pgSQL function prevent_no_available_copies() line 5 at RAISE
```

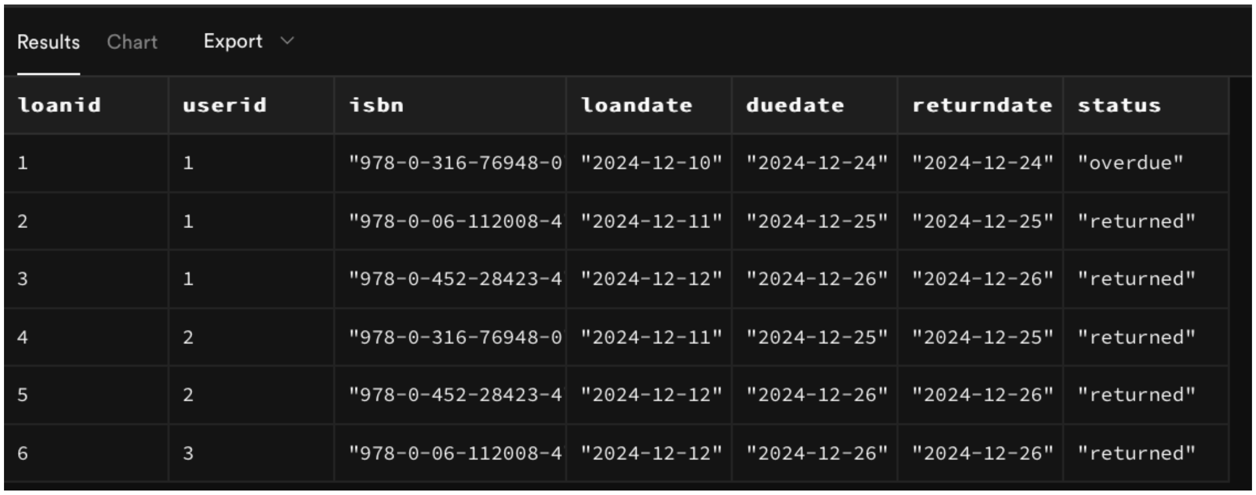

```sql
INSERT INTO Books (ISBN, Title, Author, Genre, PublishedYear, QuantityAvailable)
VALUES
('978-0-316-76948-0', 'The Catcher in the Rye', 'J.D. Salinger', 'Fiction', 1951, 5),
('978-0-06-112008-4', 'To Kill a Mockingbird', 'Harper Lee', 'Fiction', 1960, 5),
('978-0-452-28423-4', '1984', 'George Orwell', 'Dystopian Fiction', 1949, 5);
```

loanid	userid	isbn	loandate	duedate	returndate	status
1	1	"978-0-316-76948-0	"2024-12-10"	"2024-12-24"	"2024-12-24"	"returned"
2	1	"978-0-06-112008-4	"2024-12-11"	"2024-12-25"	"2024-12-25"	"returned"
3	1	"978-0-452-28423-4	"2024-12-12"	"2024-12-26"	"2024-12-26"	"returned"
4	2	"978-0-316-76948-0	"2024-12-11"	"2024-12-25"	"2024-12-25"	"returned"
5	2	"978-0-452-28423-4	"2024-12-12"	"2024-12-26"	"2024-12-26"	"returned"
6	3	"978-0-06-112008-4	"2024-12-12"	"2024-12-26"	"2024-12-26"	"returned"


```sql
UPDATE BookLoans
SET Status = 'overdue'
WHERE Status = 'borrowed' AND DueDate < CURRENT_DATE;

-- Query to Find All Overdue Loans
SELECT bl.LoanID, u.FullName, b.Title, bl.LoanDate, bl.DueDate, bl.Status
FROM BookLoans bl
JOIN Users u ON bl.UserID = u.UserID
JOIN Books b ON bl.ISBN = b.ISBN
WHERE bl.Status = 'overdue';

SELECT * FROM Books;
SELECT * FROM Users;
SELECT * FROM BookLoans;
```

Example of Overdue Loans:


```sql
UPDATE BookLoans
SET ReturnDate = '2024-12-24',
    Status = CASE
        WHEN '2024-12-26' <= DueDate THEN 'returned'
        WHEN '2024-12-26' > DueDate THEN 'overdue'
    END
WHERE ISBN = '978-0-316-76948-0' AND UserID = 1;
```



loanid	userid	isbn	loandate	duedate	returndate	status
1	1	"978-0-316-76948-0	"2024-12-10"	"2024-12-24"	"2024-12-24"	"overdue"
2	1	"978-0-06-112008-4	"2024-12-11"	"2024-12-25"	"2024-12-25"	"returned"
3	1	"978-0-452-28423-4	"2024-12-12"	"2024-12-26"	"2024-12-26"	"returned"
4	2	"978-0-316-76948-0	"2024-12-11"	"2024-12-25"	"2024-12-25"	"returned"
5	2	"978-0-452-28423-4	"2024-12-12"	"2024-12-26"	"2024-12-26"	"returned"
6	3	"978-0-06-112008-4	"2024-12-12"	"2024-12-26"	"2024-12-26"	"returned"

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.
    - Storage Constraints. Archiving old loan records can be a solution.
    - Slow Query. Indexes can be used to allow the database to find rows much faster without scanning the entire table
    - Security and Data Privacy. Encrypt sensitive data such as email addresses and implement role-based access controls

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.