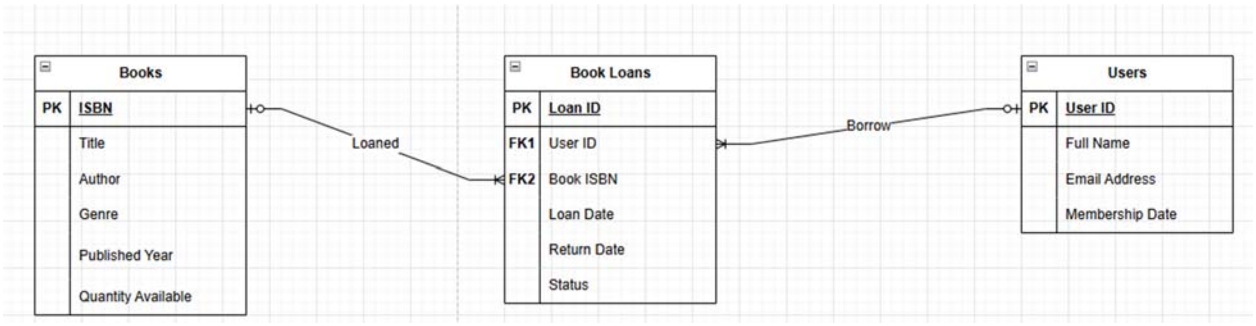
Part 1 - Conceptual Design

Design for the Entity Relationship (ER) Diagram with their primary key, attributes, relationships and cardinalities



Part 2 – Logical Design

Translating the Entity Relationship (ER) Diagram into relational tables or table schema

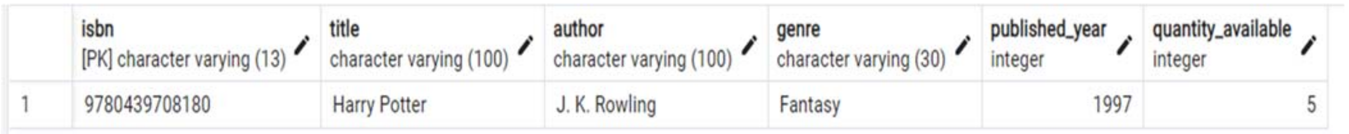
```
CREATE TABLE Books (
    ISBN VARCHAR(13) PRIMARY KEY,
    Title VARCHAR(100) NOT NULL,
    Author VARCHAR(100) NOT NULL,
    Genre VARCHAR(30),
    Published_Year INT,
    Quantity_Available INT NOT NULL CHECK (Quantity_Available >= 0)
);


CREATE TABLE Users (
    User_ID SERIAL PRIMARY KEY,
    Full_Name VARCHAR(200) NOT NULL,
    Email_Address VARCHAR(200) UNIQUE NOT NULL,
    Membership_Date DATE NOT NULL
);


CREATE TABLE Book_Loans (
    Loan_ID SERIAL PRIMARY KEY,
    User_ID INT NOT NULL,
    Book_ISBN VARCHAR(13) NOT NULL,
    Loan_Date DATE NOT NULL,
    Return_Date DATE,
    Status loan_status NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES Users(User_ID),
    FOREIGN KEY (Book_ISBN) REFERENCES Books(ISBN),
    CHECK (Return_Date IS NULL OR Return_Date >= Loan_Date)
);

CREATE TYPE loan_status AS ENUM ('borrowed', 'returned', 'overdue');
```

Part 3 – SQL Queries

SQL Queries for the following scenarios

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

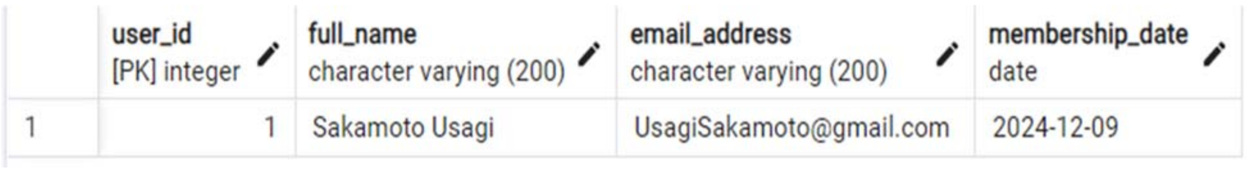
```sql
INSERT INTO Books (Title, Author, ISBN, Genre, Published_Year, Quantity_Available)
VALUES ('Harry Potter', 'J. K. Rowling', '9780439708180', 'Fantasy', 1997, 5);
```



	isbn [PK] character varying (13)	title character varying (100)	author character varying (100)	genre character varying (30)	published_year integer	quantity_available integer
1	9780439708180	Harry Potter	J. K. Rowling	Fantasy	1997	5

B. Add a new user to the system.

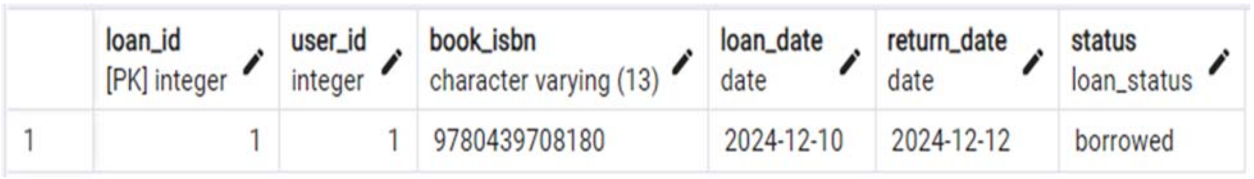
```sql
INSERT INTO Users (Full_Name, Email_Address, Membership_Date)
VALUES ('Sakamoto Usagi', 'UsagiSakamoto@gmail.com', '2024-12-9');
```



	user_id [PK] integer	full_name character varying (200)	email_address character varying (200)	membership_date date
1	1	Sakamoto Usagi	UsagiSakamoto@gmail.com	2024-12-09

C. Record a book loan for a user.

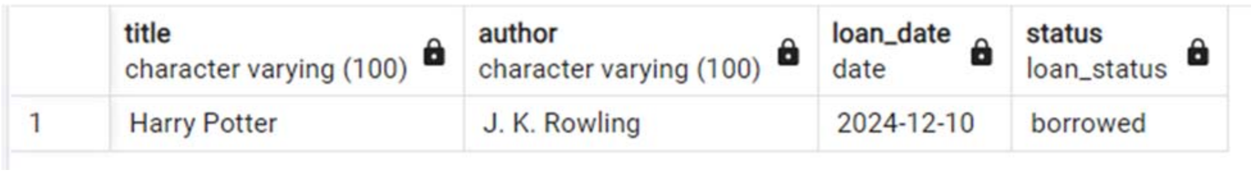
```sql
INSERT INTO Book_Loans (User_ID, Book_ISBN, Loan_Date, Return_Date, Status)
VALUES (1, '9780439708180', '2024-12-10', '2024-12-12', 'borrowed');
```



	loan_id [PK] integer	user_id integer	book_isbn character varying (13)	loan_date date	return_date date	status loan_status
1	1	1	9780439708180	2024-12-10	2024-12-12	borrowed

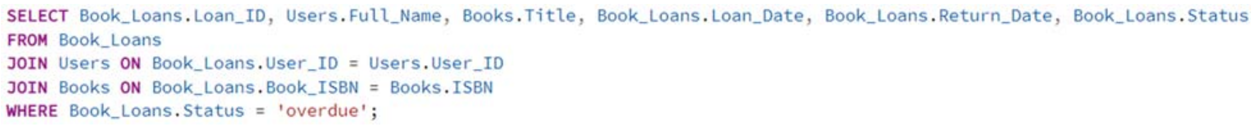
D. Find all books borrowed by a specific user.

```sql
SELECT Books.Title, Books.Author, Book_Loans.Loan_Date, Book_Loans.Status
FROM Books
JOIN Book_Loans ON Books.ISBN = Book_Loans.Book_ISBN
WHERE Book_Loans.User_ID = 1;
```



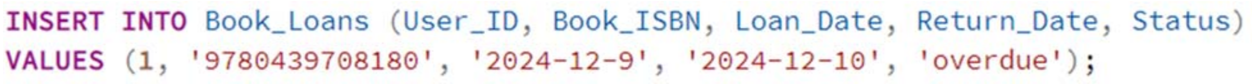
	title character varying (100)	author character varying (100)	loan_date date	status loan_status
1	Harry Potter	J. K. Rowling	2024-12-10	borrowed

E. List all overdue loans.

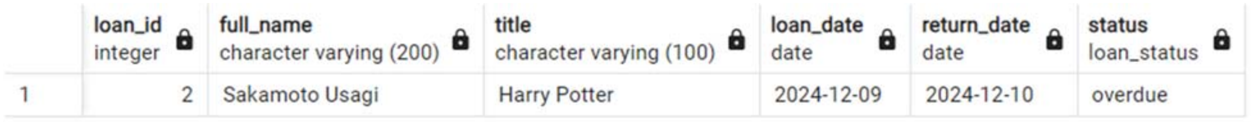


```sql
SELECT Book_Loans.Loan_ID, Users.Full_Name, Books.Title, Book_Loans.Loan_Date, Book_Loans.Return_Date, Book_Loans.Status
FROM Book_Loans
JOIN Users ON Book_Loans.User_ID = Users.User_ID
JOIN Books ON Book_Loans.Book_ISBN = Books.ISBN
WHERE Book_Loans.Status = 'overdue';
```

Inserted Value with an “Overdue” status

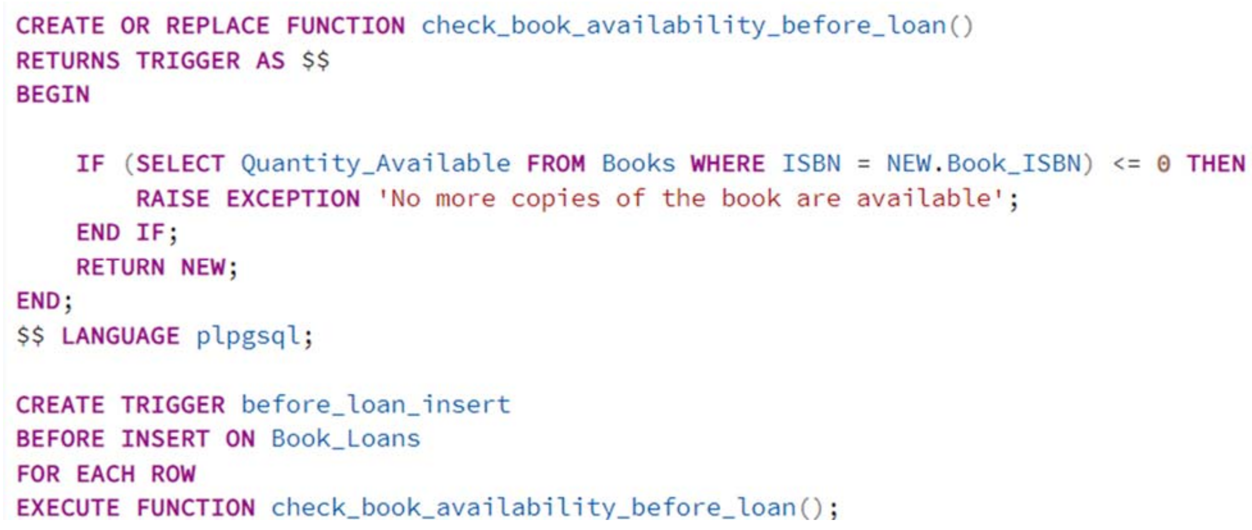


```sql
INSERT INTO Book_Loans (User_ID, Book_ISBN, Loan_Date, Return_Date, Status)
VALUES (1, '9780439708180', '2024-12-9', '2024-12-10', 'overdue');
```



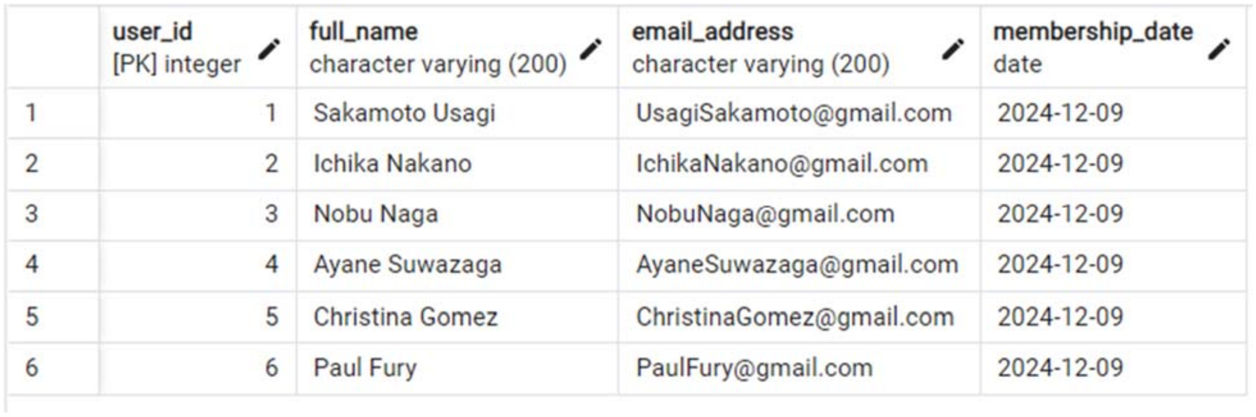
	loan_id integer	full_name character varying (200)	title character varying (100)	loan_date date	return_date date	status loan_status
1	2	Sakamoto Usagi	Harry Potter	2024-12-09	2024-12-10	overdue

Part 4: Data Integrity and Optimization



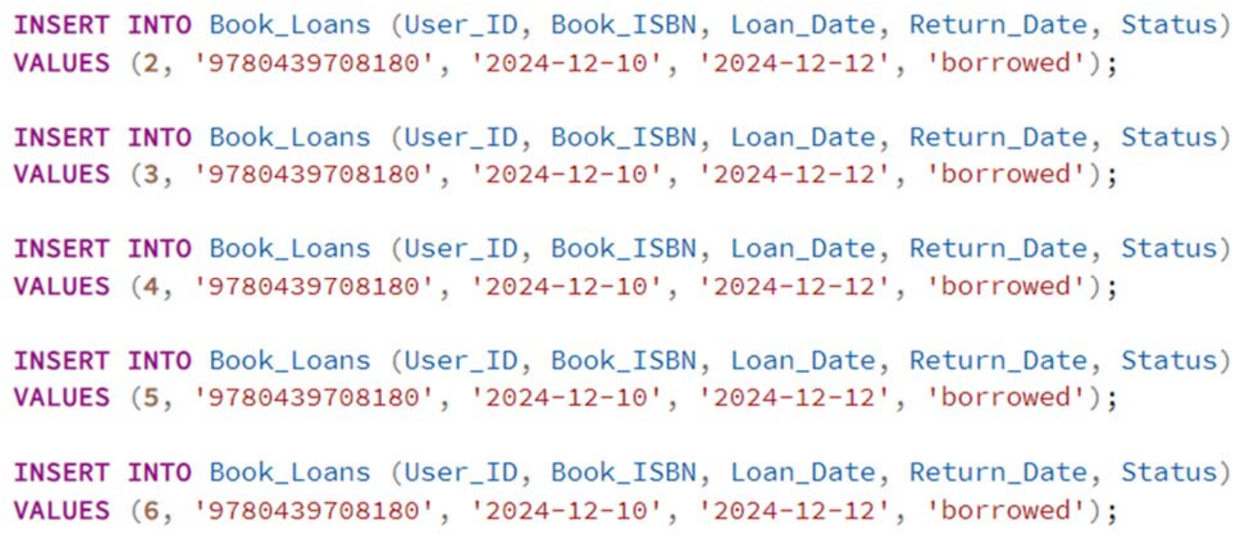
```sql
CREATE OR REPLACE FUNCTION check_book_availability_before_loan()
RETURNS TRIGGER AS $$
BEGIN

    IF (SELECT Quantity_Available FROM Books WHERE ISBN = NEW.Book_ISBN) <= 0 THEN
        RAISE EXCEPTION 'No more copies of the book are available';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_loan_insert
BEFORE INSERT ON Book_Loans
FOR EACH ROW
EXECUTE FUNCTION check_book_availability_before_loan();
```

New Members



	user_id [PK] integer	full_name character varying (200)	email_address character varying (200)	membership_date date
1	1	Sakamoto Usagi	UsagiSakamoto@gmail.com	2024-12-09
2	2	Ichika Nakano	IchikaNakano@gmail.com	2024-12-09
3	3	Nobu Naga	NobuNaga@gmail.com	2024-12-09
4	4	Ayane Suwazaga	AyaneSuwazaga@gmail.com	2024-12-09
5	5	Christina Gomez	ChristinaGomez@gmail.com	2024-12-09
6	6	Paul Fury	PaulFury@gmail.com	2024-12-09

5 Books of Harry Potter currently there is only 4 available copies



```
INSERT INTO Book_Loans (User_ID, Book_ISBN, Loan_Date, Return_Date, Status)
VALUES (2, '9780439708180', '2024-12-10', '2024-12-12', 'borrowed');

INSERT INTO Book_Loans (User_ID, Book_ISBN, Loan_Date, Return_Date, Status)
VALUES (3, '9780439708180', '2024-12-10', '2024-12-12', 'borrowed');

INSERT INTO Book_Loans (User_ID, Book_ISBN, Loan_Date, Return_Date, Status)
VALUES (4, '9780439708180', '2024-12-10', '2024-12-12', 'borrowed');

INSERT INTO Book_Loans (User_ID, Book_ISBN, Loan_Date, Return_Date, Status)
VALUES (5, '9780439708180', '2024-12-10', '2024-12-12', 'borrowed');

INSERT INTO Book_Loans (User_ID, Book_ISBN, Loan_Date, Return_Date, Status)
VALUES (6, '9780439708180', '2024-12-10', '2024-12-12', 'borrowed');
```

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 challenges I think that would arise if the scaling of the database to handle millions of users or data, I think data integrity and consistency might be one of them because of how millions of data or information are being poured in the database, it needs to be managed or maintained very often to ensure data consistency and prevent invalid data to be put inside the database.