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**.

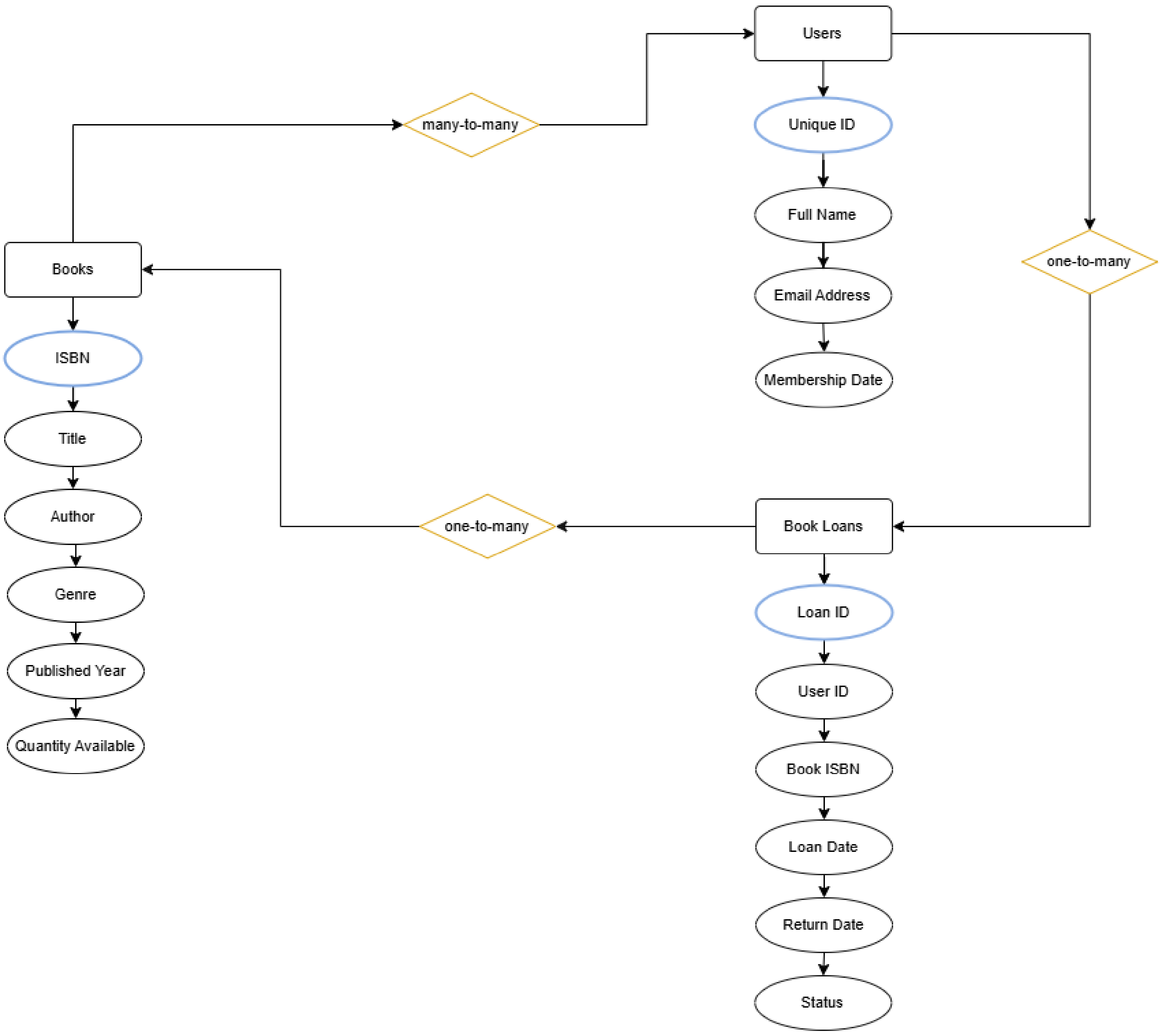
Case Study:

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



*Relationship with cardinalities:

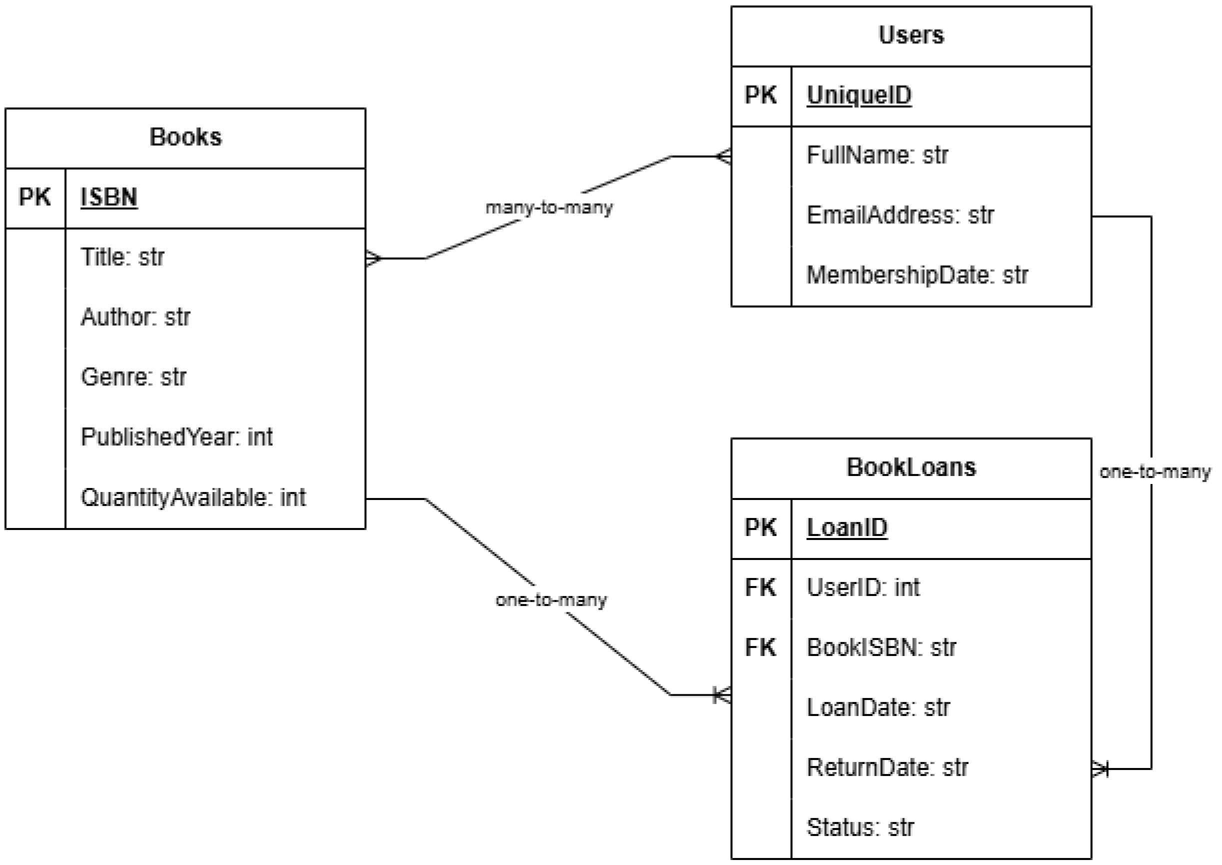
- A User can have many Book Loans, but each Book Loans is associated with only one User. (1-to-many)
- A User can borrow multiple Books, and a Book can be borrowed by multiple Users. (many-to-many)
- A Book can contain multiple Book Loans, but each Book Loans belongs to only one Book. (1-to-many)

*Primary Keys (PK)

- Blue

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).



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.

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

INSERT INTO Books (ISBN, Title, Author, Genre, PublishedYear, QuantityAvailable)
VALUES ('978-3-16-148410-0', 'How to Spot a Red Flag?', 'Valerie Jane', 'Romantic
Comedy', 2024, 5);

-- b. Add a new user to the system.

INSERT INTO Users (FullName, EmailAddress, MembershipDate)
VALUES ('Maurien Miclat', 'maumicla@gmail,com', CURRENT_DATE);
```

```
-- c. Record a book loan for a user.
INSERT INTO BookLoans (LoanID, UserID, BookISBN, LoanDate, ReturnDate, Status)
VALUES (' ZIAV83900' 11, '978-3-16-148410-0', CURRENT_DATE, DATE, 'borrowed');

-- d. Find all books borrowed by a specific user. --
SELECT b.Title, b.Author, bl. LoanDate, bl. Status
FROM BookLoans bl
JOIN Books  b ON bl.ISBN = b.ISBN
WHERE bl.UserID = 11;

-- e. List all overdue loans. --
SELECT bl.LoanID, u.FullName, b.Title, bl.LoanDate
FROM BookLoans bl
JOIN Users u ON bl.UserID = u.UserID
JOIN Books b ON bl.ISBN = b.ISBN
WHERE bl. Status = 'overdue';
```

Part 4: Data Integrity and Optimization

4. Explain how you would ensure:
    - The prevention of borrowing books when no copies are available. (15 pts)

      - The system checks book availability before allowing a loan by using a simple SELECT statement with EXISTS. If available, it immediately updates the quantity, preventing the borrowing of unavailable books and avoiding complex stored procedures.

```
-- Check availability first before inserting a book loan --
INSERT INTO BookLoans(UserID, BookISBN, LoanDate, Status)
SELECT 11, '978-3-16-148410-0', CURRENT_DATE, 'borrowed'
WHERE EXISTS (
    SELECT 11
    FROM Books
    WHERE BookISBN = '978-3-16-148410-0'
    AND QuantityAvailable > 0
);

-- Synchronously update the book quantity --
UPDATE Books
SET QuantityAvailable = QuantityAvailable - 1
WHERE ISBN = '978-3-16-148410-0'
AND QuantityAvailable > 0;
```

- Fast retrieval of overdue loans. (20 pts - with CODE and actual screenshot of performance)

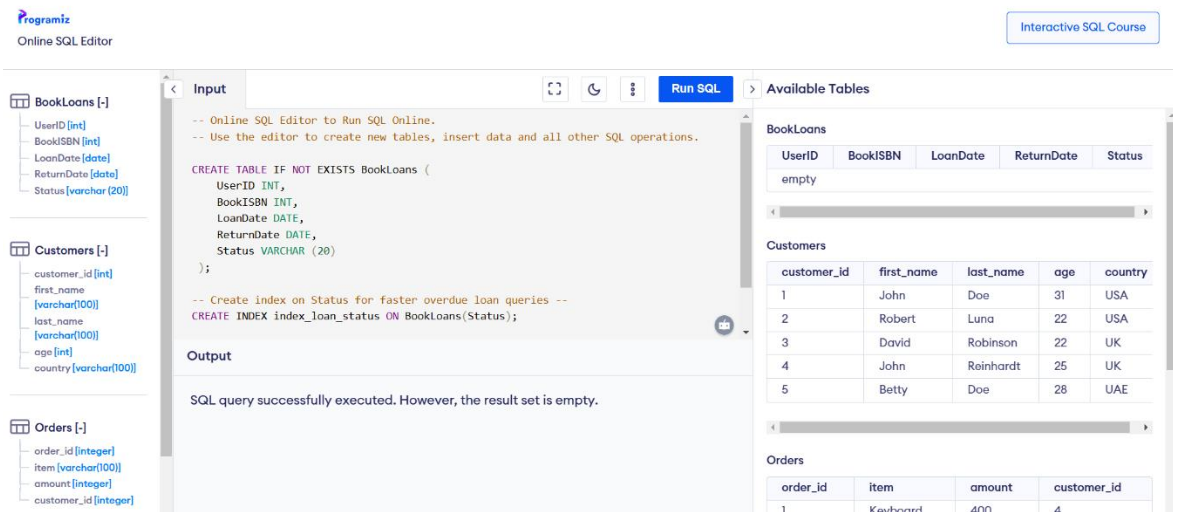
  - Indexing strategies improve query performance by creating a separate structure that helps the database engine quickly locate specific data rows. This allows the database to find relevant information without scanning the entire table, similar to using a book's index. This is especially helpful for complex queries with large datasets, as it greatly reduces data retrieval time.

```sql
CREATE TABLE IF NOT EXISTS BookLoans (
    UserID INT,
    BookISBN INT,
    LoanDate DATE,
    ReturnDate DATE,
    Status VARCHAR (20)
 );

INSERT INTO BookLoans (UserID, BookISBN, LoanDate, ReturnDate, 
Status)
VALUES (11, '978-3-16-148410-0', '2024/12/12', '2024/12/17', 
'borrowed'),
       (12, '924-1-10-064430-0', '2024/12/18', '2024/12/20', 
'returned'),
       (13, '991-0-21-238807-1', '2024/12/16', '2024/12/18', 
'overdue') ;


 -- Create index on Status for faster overdue loan queries --
CREATE INDEX index_loan_status ON BookLoans(Status);

-- Composite index for more complex queries--
CREATE INDEX index_loan_details ON BookLoans(Status, LoanDate, 
ReturnDate, UserID);
```



```sql
-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.

CREATE TABLE IF NOT EXISTS BookLoans (
    UserID INT,
    BookISBN INT,
    LoanDate DATE,
    ReturnDate DATE,
    Status VARCHAR (20)
 );

-- Create index on Status for faster overdue loan queries --
CREATE INDEX index_loan_status ON BookLoans(Status);
```

SQL query successfully executed. However, the result set is empty.

BookLoans

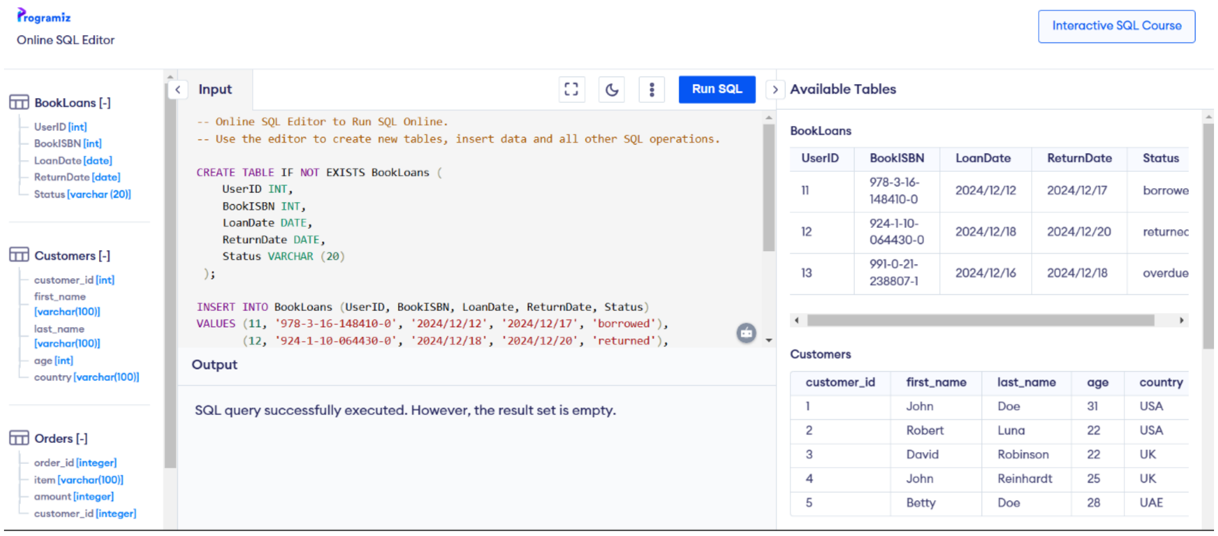
UserID	BookISBN	LoanDate	ReturnDate	Status
empty				

Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

Orders

order_id	item	amount	customer_id
1	Keyboard	400	4



```sql
-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.

CREATE TABLE IF NOT EXISTS BookLoans (
    UserID INT,
    BookISBN INT,
    LoanDate DATE,
    ReturnDate DATE,
    Status VARCHAR (20)
 );

INSERT INTO BookLoans (UserID, BookISBN, LoanDate, ReturnDate, Status)
VALUES (11, '978-3-16-148410-0', '2024/12/12', '2024/12/17', 'borrowed'),
       (12, '924-1-10-064430-0', '2024/12/18', '2024/12/20', 'returned'),
```

```
SQL query successfully executed. However, the result set is empty.
```

BookLoans

UserID	BookISBN	LoanDate	ReturnDate	Status
11	978-3-16-148410-0	2024/12/12	2024/12/17	borrowe
12	924-1-10-064430-0	2024/12/18	2024/12/20	returnec
13	991-0-21-238807-1	2024/12/16	2024/12/18	overdue

Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

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.

   - The main challenges include storage, performance, redundancy, optimization of data and queries, and managing loan transactions. Solutions involve utilizing cloud-based services like Google Cloud and AWS, distributing large databases across multiple machines, and replicating data in various geographic locations. It's essential to regularly monitor query performance and manage transactions effectively.

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.