```sql
CREATE DATABASE Library;

USE Library;

-- Creating the Authors Table
CREATE TABLE Authors (
    AuthorID INT PRIMARY KEY IDENTITY(1,1),
    Name NVARCHAR(255) NOT NULL,
    BirthDate DATE,
    Nationality NVARCHAR(100)
);

SET IDENTITY_INSERT Authors ON;

-- Inserting the values into the Authors Table
INSERT INTO Authors (AuthorID, Name, BirthDate, Nationality)
VALUES
(1, 'F. Scott Fitzgerald', '1896-09-24', 'American'),
(2, 'Jane Austen', '1775-12-16', 'British'),
(3, 'George Orwell', '1903-06-25', 'British'),
(4, 'J.K. Rowling', '1965-07-31', 'British'),
(5, 'J.R.R. Tolkien', '1892-01-03', 'British'),
(6, 'Ernest Hemingway', '1899-07-21', 'American'),
(7, 'Agatha Christie', '1890-09-15', 'British'),
(8, 'Leo Tolstoy', '1828-09-09', 'Russian'),
(9, 'Mark Twain', '1835-11-30', 'American'),
(10, 'Charles Dickens', '1812-02-07', 'British'),
(11, 'Stephen King', '1947-09-21', 'American'),
(12, 'Isaac Asimov', '1920-01-02', 'American'),
(13, 'Harper Lee', '1926-04-28', 'American'),
(14, 'Emily Brontë', '1818-07-30', 'British'),
(15, 'Mary Shelley', '1797-08-30', 'British'),
(16, 'Oscar Wilde', '1854-10-16', 'Irish'),
(17, 'Gabriel Garcia Marquez', '1927-03-06', 'Colombian'),
(18, 'Arthur Conan Doyle', '1859-05-22', 'British'),
(19, 'Franz Kafka', '1883-07-03', 'Austrian'),
(20, 'Virginia Woolf', '1882-01-25', 'British');

SET IDENTITY_INSERT Authors OFF;

-- Showing all the records of the Authors Table
SELECT * from Authors;

-- Creating the Books Table
CREATE TABLE Books (
    BookID INT PRIMARY KEY IDENTITY(1,1),
    Title NVARCHAR(255) NOT NULL,
    AuthorID INT FOREIGN KEY REFERENCES Authors(AuthorID),
    ISBN NVARCHAR(20) UNIQUE NOT NULL,
    Availability BIT NOT NULL DEFAULT 1
);
```

```sql
SET IDENTITY_INSERT Books ON;

-- Inserting the values into the Books Table
INSERT INTO Books (BookID, Title, AuthorID, ISBN, Availability)
VALUES
(1, 'The Great Gatsby', 1, '9780743273565', 1),
(2, 'Pride and Prejudice', 2, '9780141439518', 1),
(3, '1984', 3, '9780451524935', 0),
(4, 'Harry Potter and the Sorcerer''s Stone', 4, '9780439554930',
1),
(5, 'The Hobbit', 5, '9780547928227', 1),
(6, 'The Old Man and the Sea', 6, '9780684801223', 0),
(7, 'Murder on the Orient Express', 7, '9780062693662', 1),
(8, 'War and Peace', 8, '9780199232765', 1),
(9, 'Adventures of Huckleberry Finn', 9, '9780486280615', 0),
(10, 'Oliver Twist', 10, '9780141439747', 1),
(11, 'The Shining', 11, '9780307743657', 1),
(12, 'Foundation', 12, '9780553293357', 1),
(13, 'To Kill a Mockingbird', 13, '9780061120084', 0),
(14, 'Wuthering Heights', 14, '9780141439556', 1),
(15, 'Frankenstein', 15, '9780486282114', 1),
(16, 'The Picture of Dorian Gray', 16, '9780141439570', 1),
(17, 'One Hundred Years of Solitude', 17, '9780060883287', 1),
(18, 'The Adventures of Sherlock Holmes', 18, '9781508475316', 0),
(19, 'The Metamorphosis', 19, '9780553213690', 1),
(20, 'Mrs Dalloway', 20, '9780156628709', 1);

SET IDENTITY_INSERT Books OFF;

-- Showing all the records of the Books Table
SELECT * from Books;

-- Creating the Patrons Table
CREATE TABLE Patrons (
    PatronID INT PRIMARY KEY IDENTITY(1,1),
    Name NVARCHAR(255),
    ContactInfo NVARCHAR(255)
);

SET IDENTITY_INSERT Patrons ON;

-- Inserting the values into the Patrons Table
INSERT INTO Patrons (PatronID, Name, ContactInfo) VALUES
(1, 'Alice Johnson', 'alice.j@gmail.com'),
(2, 'Bob Smith', 'bob.smith@yahoo.com'),
(3, 'Carol Davis', 'carol.davis@hotmail.com'),
(4, 'David Brown', 'david.brown@gmail.com'),
(5, 'Eve Thompson', 'eve.t@gmail.com'),
(6, 'Frank Harris', 'frank.h@hotmail.com'),
(7, 'Grace Lee', 'grace.lee@yahoo.com'),
(8, 'Hank Moore', 'hank.moore@gmail.com'),
(9, 'Ivy Clark', 'ivy.clark@gmail.com'),
```

```
(10, 'Jack Lewis', 'jack.l@hotmail.com'),
(11, 'Kathy Young', 'kathy.young@yahoo.com'),
(12, 'Leo Allen', 'leo.a@gmail.com'),
(13, 'Mia Hall', 'mia.h@gmail.com'),
(14, 'Noah Scott', 'noah.s@hotmail.com'),
(15, 'Olivia Adams', 'olivia.adams@yahoo.com'),
(16, 'Paul Baker', 'paul.b@gmail.com'),
(17, 'Quinn Campbell', 'quinn.c@hotmail.com'),
(18, 'Ruby Mitchell', 'ruby.m@gmail.com'),
(19, 'Sam Carter', 'sam.c@gmail.com'),
(20, 'Tina Ross', 'tina.ross@yahoo.com');

SET IDENTITY_INSERT Patrons OFF;

-- Showing all the records of the Patrons Table
SELECT * from Patrons;

-- Creating the BorrowHistory Table
CREATE TABLE BorrowHistory (
    BorrowID INT PRIMARY KEY IDENTITY(1,1),
    BookID INT FOREIGN KEY REFERENCES Books(BookID),
    PatronID INT FOREIGN KEY REFERENCES Patrons(PatronID),
    BorrowDate DATETIME DEFAULT GETDATE(),
    ReturnDate DATETIME NULL
);

SET IDENTITY_INSERT BorrowHistory ON;

-- Inserting the values into the BorrowHistory Table
INSERT INTO BorrowHistory (BorrowID, BookID, PatronID, BorrowDate,
ReturnDate) VALUES
(1, 1, 1, '2024-01-15', '2024-01-25'),
(2, 2, 2, '2024-01-20', '2024-01-28'),
(3, 3, 3, '2024-01-25', NULL),
(4, 4, 4, '2024-01-28', '2024-02-05'),
(5, 5, 5, '2024-02-01', '2024-02-10'),
(6, 6, 6, '2024-02-10', NULL),
(7, 7, 7, '2024-02-15', '2024-02-25'),
(8, 8, 8, '2024-02-18', '2024-02-28'),
(9, 9, 9, '2024-03-01', NULL),
(10, 10, 10, '2024-03-05', '2024-03-15'),
(11, 11, 11, '2024-03-08', '2024-03-20'),
(12, 12, 12, '2024-03-10', '2024-03-20'),
(13, 13, 13, '2024-03-15', NULL),
(14, 14, 14, '2024-03-20', '2024-03-30'),
(15, 15, 15, '2024-03-22', '2024-04-01'),
(16, 16, 16, '2024-03-25', '2024-04-05'),
(17, 17, 17, '2024-04-01', NULL),
(18, 18, 18, '2024-04-05', '2024-04-15'),
(19, 19, 19, '2024-04-10', '2024-04-20'),
(20, 20, 20, '2024-04-12', NULL);
```

```sql
SET IDENTITY_INSERT BorrowHistory OFF;

-- Showing all the records of the BorrowHistory Table
SELECT * from BorrowHistory;

-- Adding the new book record into the data
INSERT INTO Books (Title, AuthorID, ISBN, Availability)
VALUES ('The Silmarillion', 5, '9780618391110', 1);

-- Updating the book details in the data
UPDATE Books
SET Title = 'The Chronicles of Middle-earth',
    AuthorID = 5,
    ISBN = '9780987654321'
WHERE BookID = 10;

SELECT * FROM Books;

-- Marking the book as borrowed
UPDATE Books
SET Availability = 0
WHERE BookID = 3;

-- Marking the book as returned
UPDATE Books
SET Availability = 1
WHERE BookID = 3;

SELECT * FROM Books;

-- Searching the Books by Title, Author, or ISBN
SELECT b.BookID, b.Title, a.Name AS Author, b.ISBN, b.Availability
FROM Books b
JOIN Authors a ON b.AuthorID = a.AuthorID
WHERE b.Title LIKE '%Harry%'
   OR a.Name LIKE '%Rowling%'
   OR b.ISBN = '9780439554930';

-- Tracking the Borrowing and Returning History of the Books
SELECT bh.BorrowID, b.Title, p.Name AS Patron, bh.BorrowDate,
bh.ReturnDate
FROM BorrowHistory bh
JOIN Books b ON bh.BookID = b.BookID
JOIN Patrons p ON bh.PatronID = p.PatronID
ORDER BY bh.BorrowDate DESC;


-- Using Window Function and CTE:- Below Example shows the Count
number of books borrowed by each patron, showing the most recent
borrow first
WITH BorrowCounts AS ( SELECT
        p.PatronID, p.Name,
```

```
        COUNT(bh.BorrowID) OVER (PARTITION BY p.PatronID) AS
TotalBorrows,
        ROW_NUMBER() OVER (PARTITION BY p.PatronID ORDER BY
bh.BorrowDate DESC) AS RecentBorrowRank,
        bh.BorrowDate, b.Title
    FROM BorrowHistory bh
    JOIN Patrons p ON bh.PatronID = p.PatronID
    JOIN Books b ON bh.BookID = b.BookID
)
SELECT PatronID, Name, Title, BorrowDate, RecentBorrowRank,
TotalBorrows
FROM BorrowCounts WHERE RecentBorrowRank = 1;


-- Identifying the Top Borrowers Using Subquery
SELECT TOP 5 p.Name, COUNT(bh.BorrowID) AS BorrowCount
FROM BorrowHistory bh
JOIN Patrons p ON bh.PatronID = p.PatronID
GROUP BY p.Name
ORDER BY BorrowCount DESC;

-- Creating a backup of the Books table using the SELECT INTO
clause
SELECT * INTO BooksBackup FROM Books;

-- Showing the backup of all the records for Books Table
SELECT * FROM BooksBackup;


-- Implementing the MERGE command to streamline borrow and return
operations while maintaining data integrity
-- A) Borrow Operation :-
DECLARE @BookID INT = 3,
        @PatronID INT = 1,
        @BorrowDate DATE = GETDATE();

MERGE BorrowHistory AS target
USING (
    SELECT @BookID AS BookID, @PatronID AS PatronID
) AS source
ON target.BookID = source.BookID
   AND target.ReturnDate IS NULL  -- Book is currently borrowed

WHEN NOT MATCHED BY TARGET
    AND EXISTS (SELECT 1 FROM Books WHERE BookID = @BookID AND
Availability = 1)
THEN
    INSERT (BookID, PatronID, BorrowDate, ReturnDate)
    VALUES (@BookID, @PatronID, @BorrowDate, NULL);

-- Updating the availability after successful insert
IF @@ROWCOUNT > 0
```

```
BEGIN
    UPDATE Books
    SET Availability = 0
    WHERE BookID = @BookID;

    PRINT 'Book borrowed successfully.';
END
ELSE
BEGIN
    PRINT 'Book is currently borrowed and not available.';
END




-- B) Return Operation :-
DECLARE @BookID INT = 3,
        @ReturnDate DATE = GETDATE();

-- Update BorrowHistory with MERGE
MERGE BorrowHistory AS target
USING (SELECT @BookID AS BookID) AS source
ON target.BookID = source.BookID AND target.ReturnDate IS NULL

WHEN MATCHED THEN
    UPDATE SET ReturnDate = @ReturnDate;

-- If the MERGE matched and updated a record, set book
availability to 1
IF @@ROWCOUNT > 0
BEGIN
    UPDATE Books
    SET Availability = 1
    WHERE BookID = @BookID;

    PRINT 'Book return processed successfully.';
END
ELSE
BEGIN
    PRINT 'No active borrow record found for this book.';
END
```