1885 SMI UNIVERSITY

Group leader:

Hiba Zehra(BSE-23F-155)

Group members:

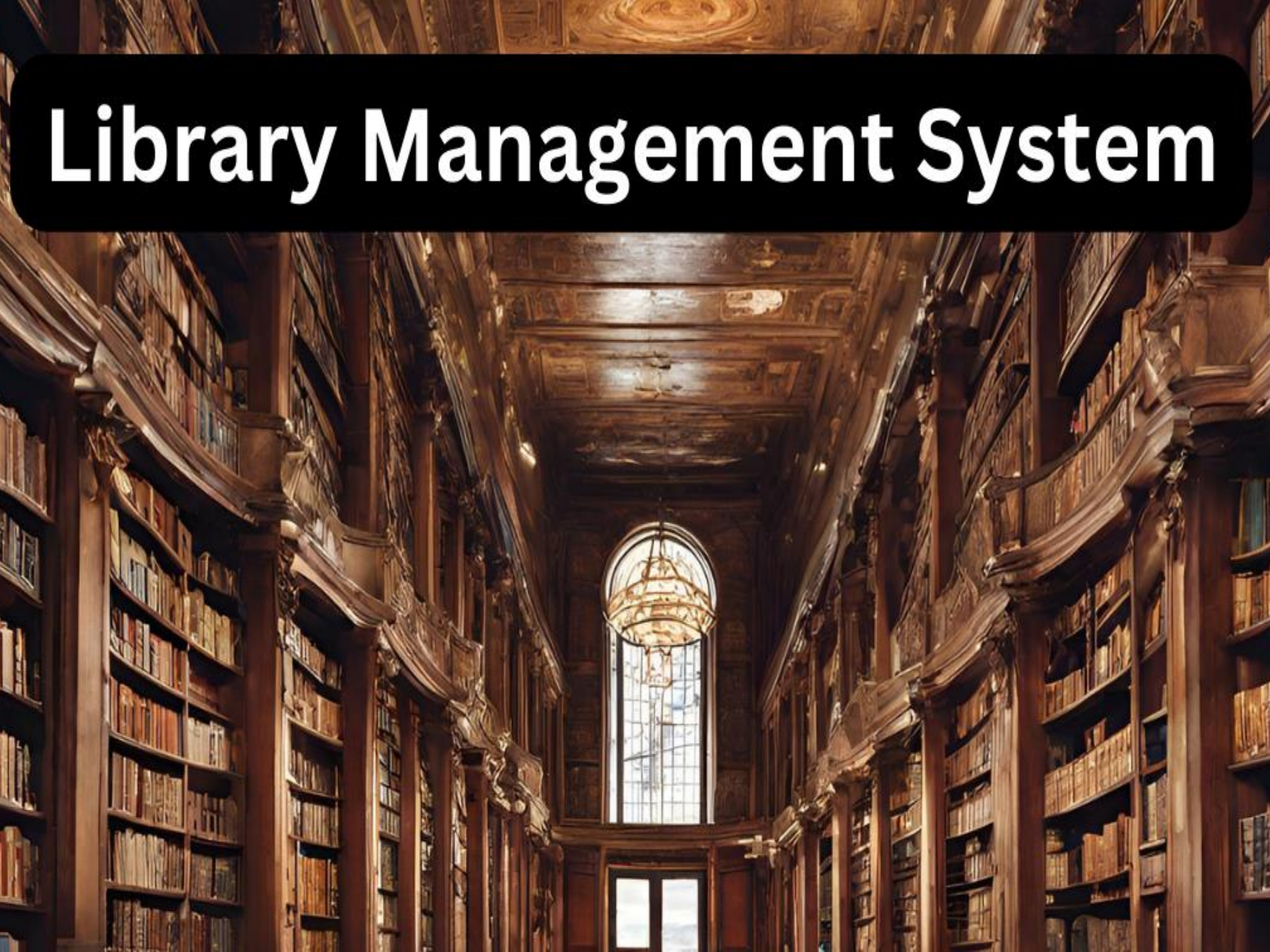
Zainab Anwer(BSE-23F-106)

Kumkum(BSE-23F-162)

Submitted To:

Miss Aqsa Umer

Library Management System



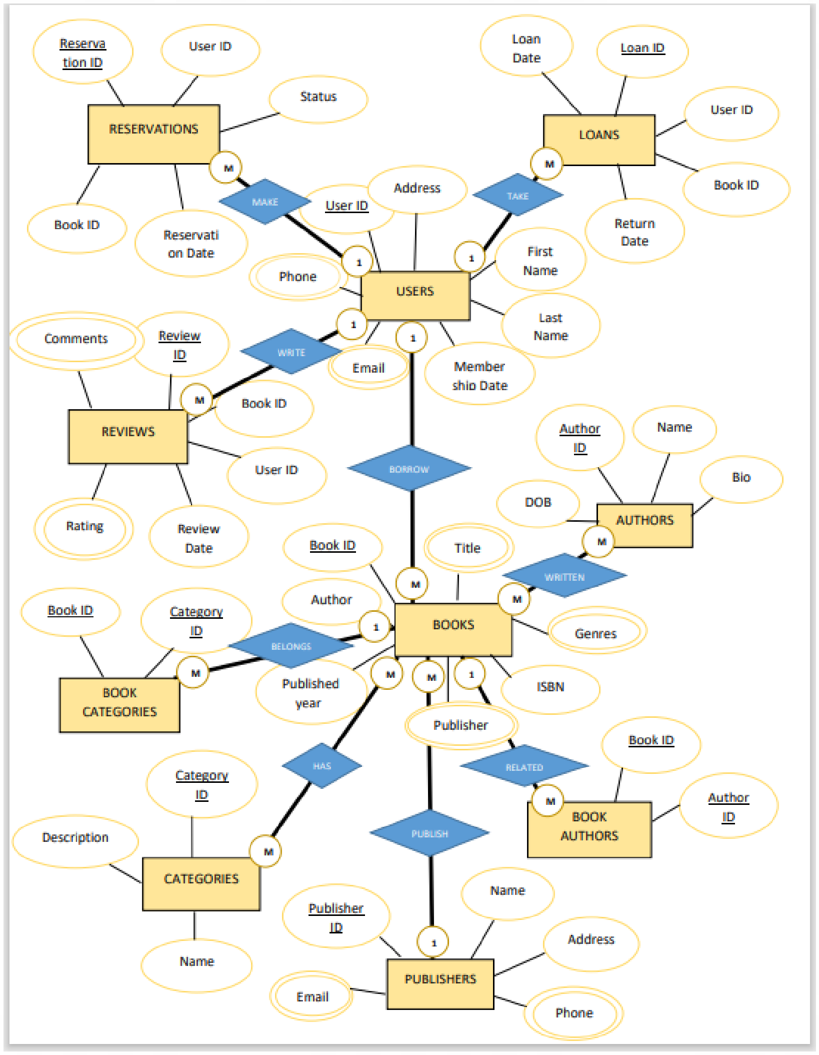
Introduction

A library management system, also known as an automated library system is software that has been develop to handle basic housekeeping function of a library.

Entity-Relationship Diagram (ERD)

- An Entity-Relationship Diagram (ERD) is data modeling technique.
- An ERD is an conceptual representational model of data.
- An Entity-Relationship Diagram (ERD) is a snapshot of data structures.
- An ERD shows entities tables in a database and relationships between tables within that database.

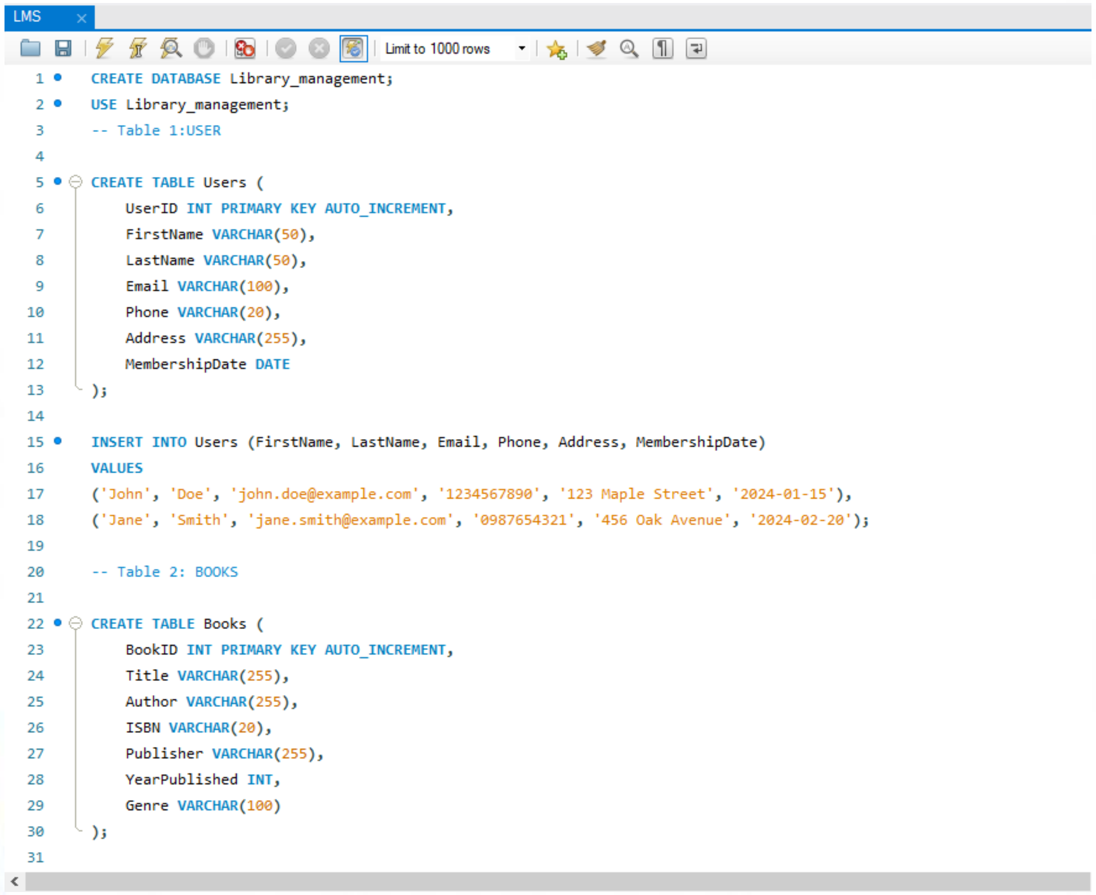
ER Model


Reservation ID
User ID
Status
RESERVATIONS
Book ID
Reservation Date
MAKE
Loan Date
Loan ID
User ID
LOANS
Book ID
Return Date
TAKE
User ID
Address
Phone
First Name
USERS
Last Name
Email
Membership Date
Comments
Review ID
WRITE
Book ID
REVIEWS
User ID
Rating
Review Date
BORROW
Author ID
Name
Bio
DOB
AUTHORS
WRITTEN
Book ID
Title
Author
BOOKS
Genres
ISBN
Published year
Publisher
Book ID
Category ID
BELONGS
BOOK CATEGORIES
HAS
RELATED
Book ID
Author ID
BOOK AUTHORS
Category ID
Description
CATEGORIES
Name
PUBLISH
Publisher ID
Name
Address
PUBLISHERS
Email
Phone
M
1

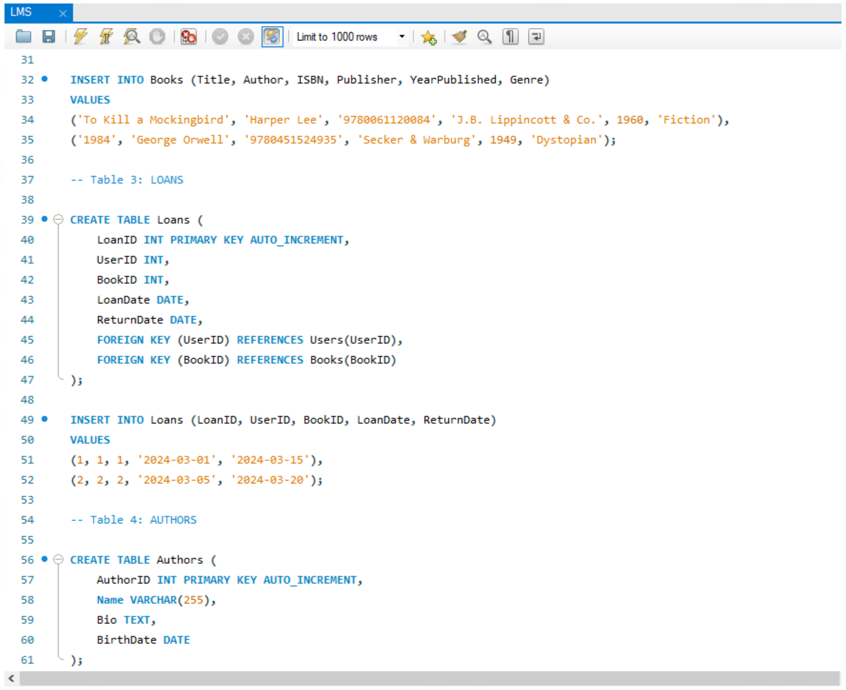
Contributions

Zainab Anwar:

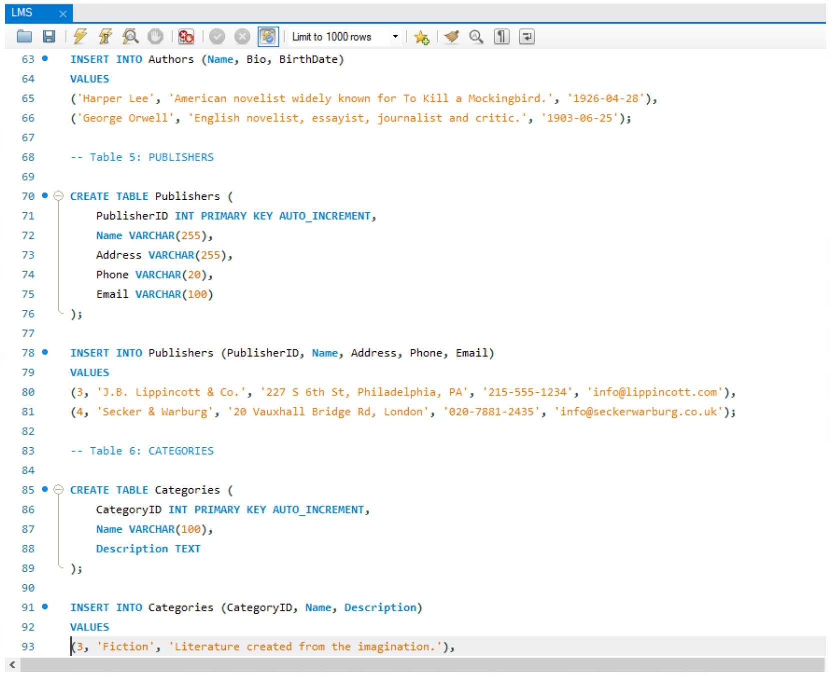
1. SQL Main File:



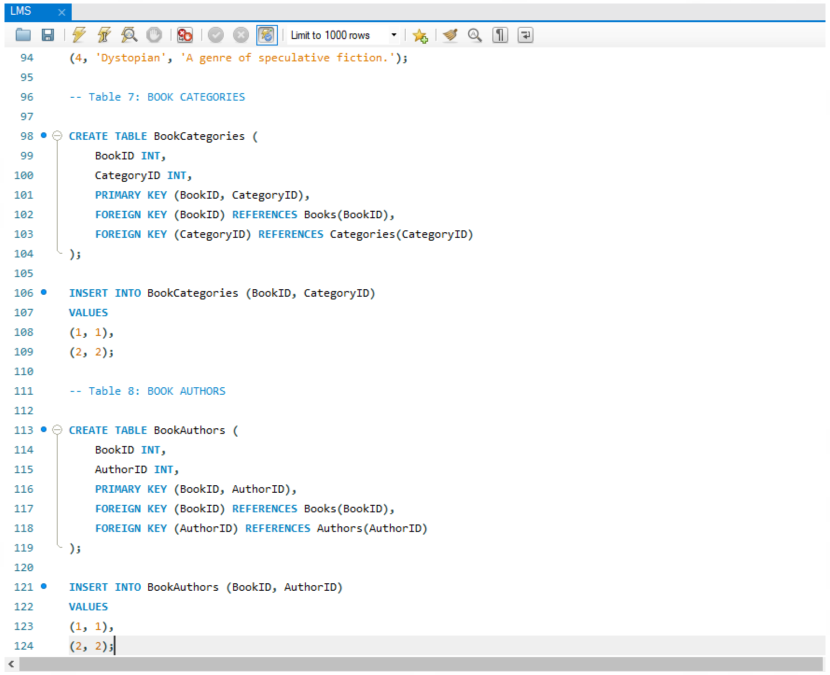
```sql
CREATE DATABASE Library_management;
USE Library_management;
-- Table 1:USER

CREATE TABLE Users (
    UserID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Email VARCHAR(100),
    Phone VARCHAR(20),
    Address VARCHAR(255),
    MembershipDate DATE
);

INSERT INTO Users (FirstName, LastName, Email, Phone, Address, MembershipDate)
VALUES
('John', 'Doe', 'john.doe@example.com', '1234567890', '123 Maple Street', '2024-01-15'),
('Jane', 'Smith', 'jane.smith@example.com', '0987654321', '456 Oak Avenue', '2024-02-20');

-- Table 2: BOOKS

CREATE TABLE Books (
    BookID INT PRIMARY KEY AUTO_INCREMENT,
    Title VARCHAR(255),
    Author VARCHAR(255),
    ISBN VARCHAR(20),
    Publisher VARCHAR(255),
    YearPublished INT,
    Genre VARCHAR(100)
);

```



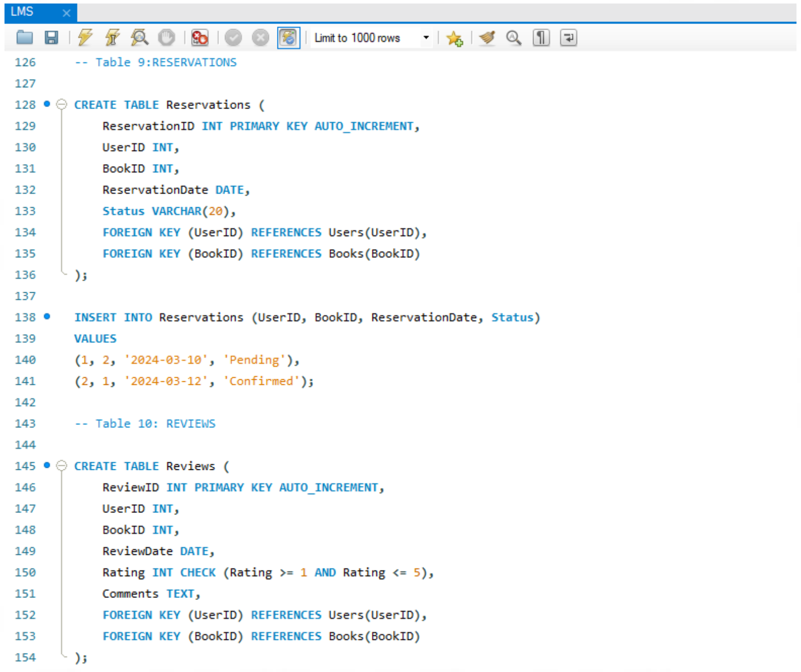
```sql

INSERT INTO Books (Title, Author, ISBN, Publisher, YearPublished, Genre)
VALUES
('To Kill a Mockingbird', 'Harper Lee', '9780061120084', 'J.B. Lippincott & Co.', 1960, 'Fiction'),
('1984', 'George Orwell', '9780451524935', 'Secker & Warburg', 1949, 'Dystopian');

-- Table 3: LOANS

CREATE TABLE Loans (
    LoanID INT PRIMARY KEY AUTO_INCREMENT,
    UserID INT,
    BookID INT,
    LoanDate DATE,
    ReturnDate DATE,
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID)
);

INSERT INTO Loans (LoanID, UserID, BookID, LoanDate, ReturnDate)
VALUES
(1, 1, 1, '2024-03-01', '2024-03-15'),
(2, 2, 2, '2024-03-05', '2024-03-20');

-- Table 4: AUTHORS

CREATE TABLE Authors (
    AuthorID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255),
    Bio TEXT,
    BirthDate DATE
);
```



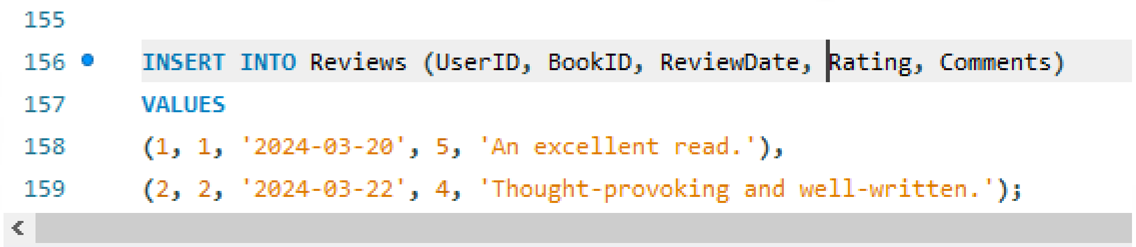
```sql
INSERT INTO Authors (Name, Bio, BirthDate)
VALUES
('Harper Lee', 'American novelist widely known for To Kill a Mockingbird.', '1926-04-28'),
('George Orwell', 'English novelist, essayist, journalist and critic.', '1903-06-25');

-- Table 5: PUBLISHERS

CREATE TABLE Publishers (
    PublisherID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255),
    Address VARCHAR(255),
    Phone VARCHAR(20),
    Email VARCHAR(100)
);

INSERT INTO Publishers (PublisherID, Name, Address, Phone, Email)
VALUES
(3, 'J.B. Lippincott & Co.', '227 S 6th St, Philadelphia, PA', '215-555-1234', 'info@lippincott.com'),
(4, 'Secker & Warburg', '20 Vauxhall Bridge Rd, London', '020-7881-2435', 'info@seckerwarburg.co.uk');

-- Table 6: CATEGORIES

CREATE TABLE Categories (
    CategoryID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100),
    Description TEXT
);

INSERT INTO Categories (CategoryID, Name, Description)
VALUES
(3, 'Fiction', 'Literature created from the imagination.'),
```



```sql
(4, 'Dystopian', 'A genre of speculative fiction.');

-- Table 7: BOOK CATEGORIES

CREATE TABLE BookCategories (
    BookID INT,
    CategoryID INT,
    PRIMARY KEY (BookID, CategoryID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);

INSERT INTO BookCategories (BookID, CategoryID)
VALUES
(1, 1),
(2, 2);

-- Table 8: BOOK AUTHORS

CREATE TABLE BookAuthors (
    BookID INT,
    AuthorID INT,
    PRIMARY KEY (BookID, AuthorID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
);

INSERT INTO BookAuthors (BookID, AuthorID)
VALUES
(1, 1),
(2, 2);
```



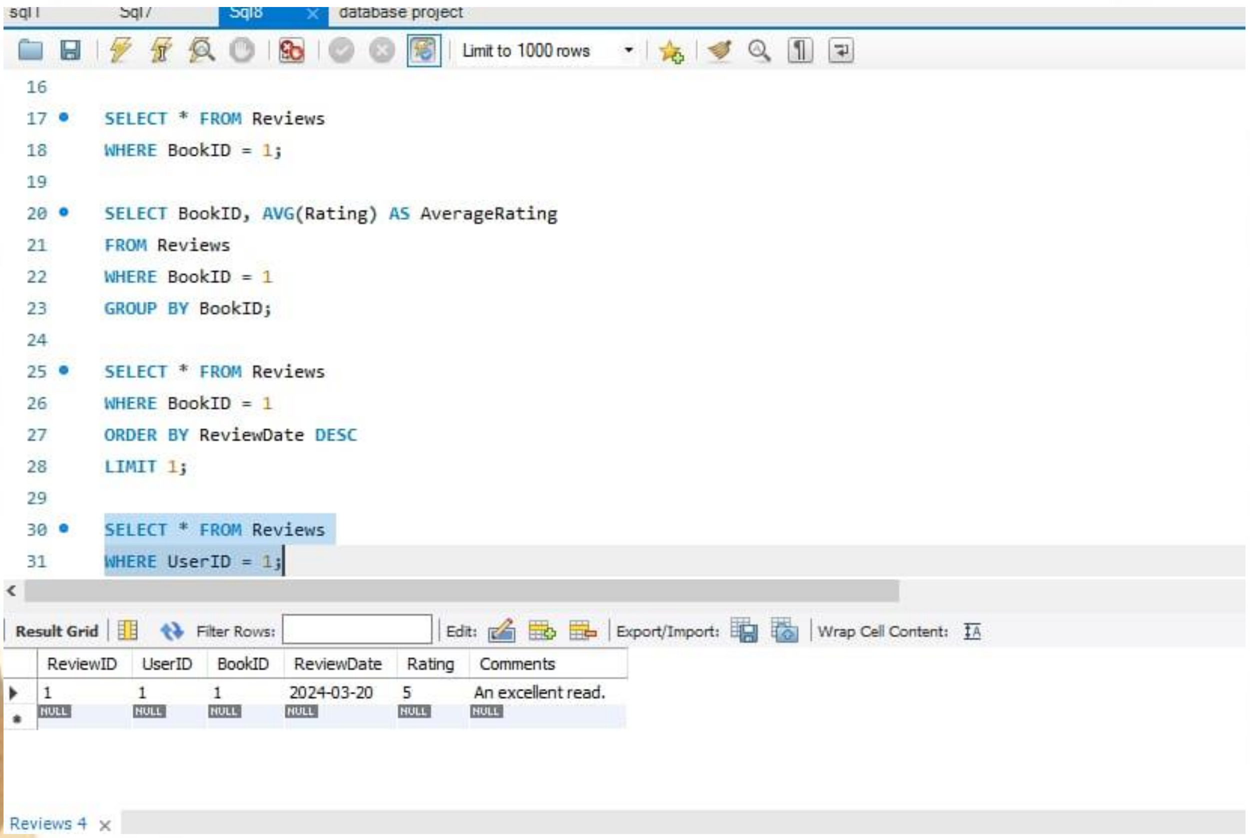
```sql
-- Table 9:RESERVATIONS

CREATE TABLE Reservations (
    ReservationID INT PRIMARY KEY AUTO_INCREMENT,
    UserID INT,
    BookID INT,
    ReservationDate DATE,
    Status VARCHAR(20),
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID)
);

INSERT INTO Reservations (UserID, BookID, ReservationDate, Status)
VALUES
(1, 2, '2024-03-10', 'Pending'),
(2, 1, '2024-03-12', 'Confirmed');

-- Table 10: REVIEWS

CREATE TABLE Reviews (
    ReviewID INT PRIMARY KEY AUTO_INCREMENT,
    UserID INT,
    BookID INT,
    ReviewDate DATE,
    Rating INT CHECK (Rating >= 1 AND Rating <= 5),
    Comments TEXT,
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID)
);
```



```sql
INSERT INTO Reviews (UserID, BookID, ReviewDate, Rating, Comments)
VALUES
(1, 1, '2024-03-20', 5, 'An excellent read.'),
(2, 2, '2024-03-22', 4, 'Thought-provoking and well-written.');
```

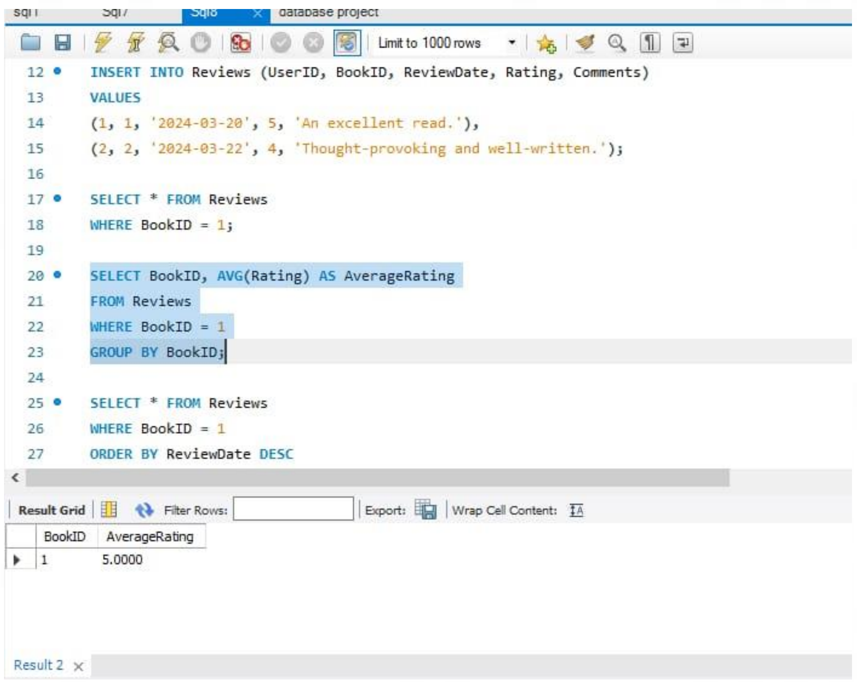
2. Queries:

Table 1:



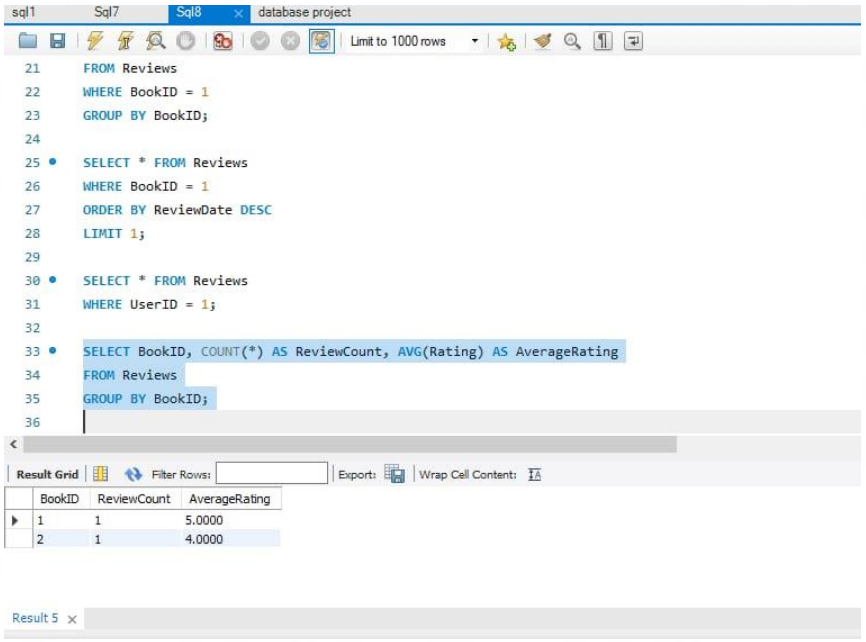
```sql
SELECT * FROM Reviews
WHERE BookID = 1;

SELECT BookID, AVG(Rating) AS AverageRating
FROM Reviews
WHERE BookID = 1
GROUP BY BookID;

SELECT * FROM Reviews
WHERE BookID = 1
ORDER BY ReviewDate DESC
LIMIT 1;

SELECT * FROM Reviews
WHERE UserID = 1;
```

```
ReviewID  UserID  BookID  ReviewDate  Rating  Comments
1         1       1       2024-03-20  5       An excellent read.
NULL      NULL    NULL    NULL        NULL    NULL
```



```
INSERT INTO Reviews (UserID, BookID, ReviewDate, Rating, Comments)
VALUES
(1, 1, '2024-03-20', 5, 'An excellent read.'),
(2, 2, '2024-03-22', 4, 'Thought-provoking and well-written.');

SELECT * FROM Reviews
WHERE BookID = 1;

SELECT BookID, AVG(Rating) AS AverageRating
FROM Reviews
WHERE BookID = 1
GROUP BY BookID;

SELECT * FROM Reviews
WHERE BookID = 1
ORDER BY ReviewDate DESC
```

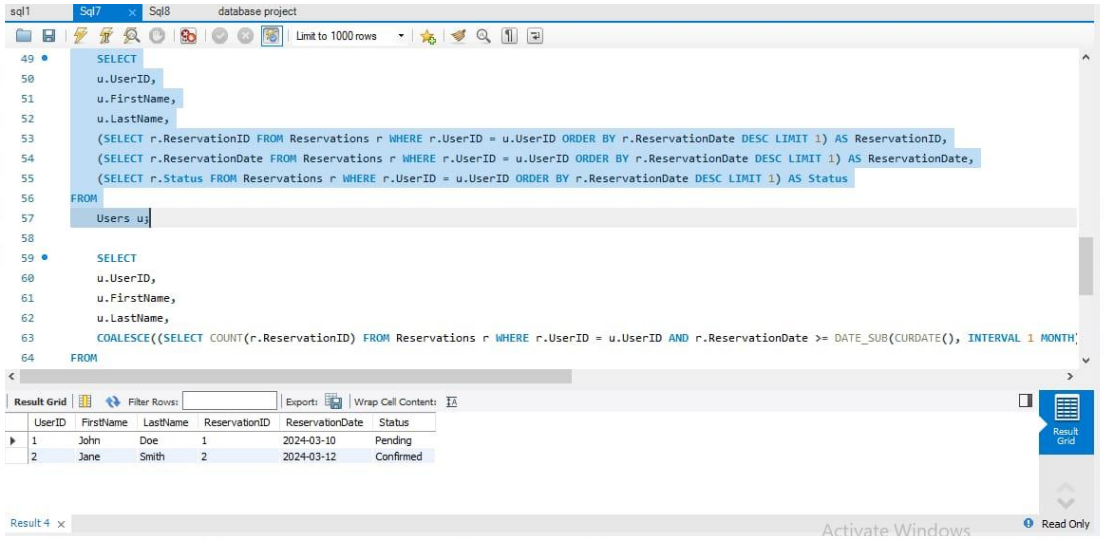
```
BookID  AverageRating
1       5.0000
```



```
FROM Reviews
WHERE BookID = 1
GROUP BY BookID;

SELECT * FROM Reviews
WHERE BookID = 1
ORDER BY ReviewDate DESC
LIMIT 1;

SELECT * FROM Reviews
WHERE UserID = 1;

SELECT BookID, COUNT(*) AS ReviewCount, AVG(Rating) AS AverageRating
FROM Reviews
GROUP BY BookID;
```

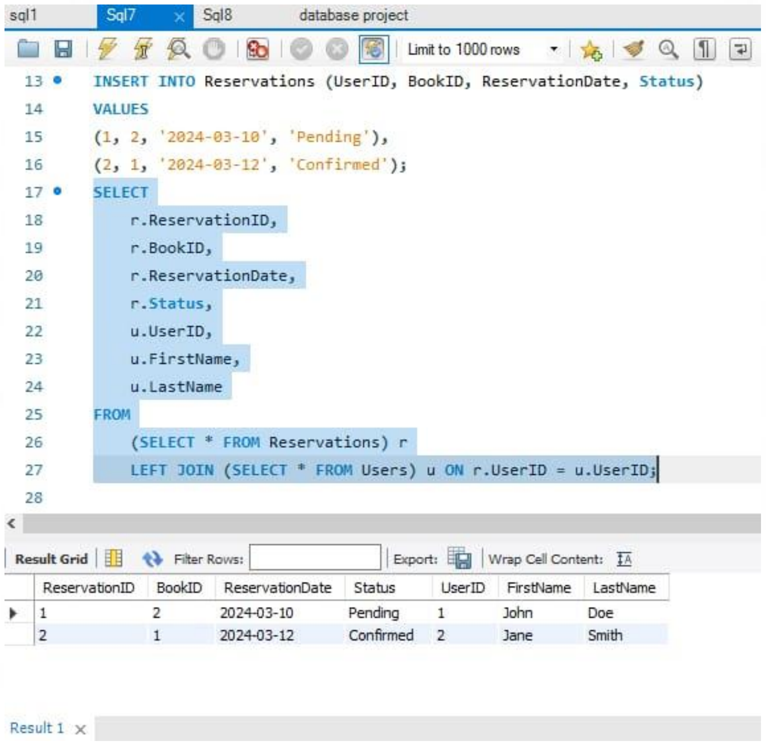
```
BookID  ReviewCount  AverageRating
1       1            5.0000
2       1            4.0000
```

Table 2:



```sql
    SELECT
    u.UserID,
    u.FirstName,
    u.LastName,
    (SELECT r.ReservationID FROM Reservations r WHERE r.UserID = u.UserID ORDER BY r.ReservationDate DESC LIMIT 1) AS ReservationID,
    (SELECT r.ReservationDate FROM Reservations r WHERE r.UserID = u.UserID ORDER BY r.ReservationDate DESC LIMIT 1) AS ReservationDate,
    (SELECT r.Status FROM Reservations r WHERE r.UserID = u.UserID ORDER BY r.ReservationDate DESC LIMIT 1) AS Status
FROM
    Users u;

    SELECT
    u.UserID,
    u.FirstName,
    u.LastName,
    COALESCE((SELECT COUNT(r.ReservationID) FROM Reservations r WHERE r.UserID = u.UserID AND r.ReservationDate >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
FROM
```

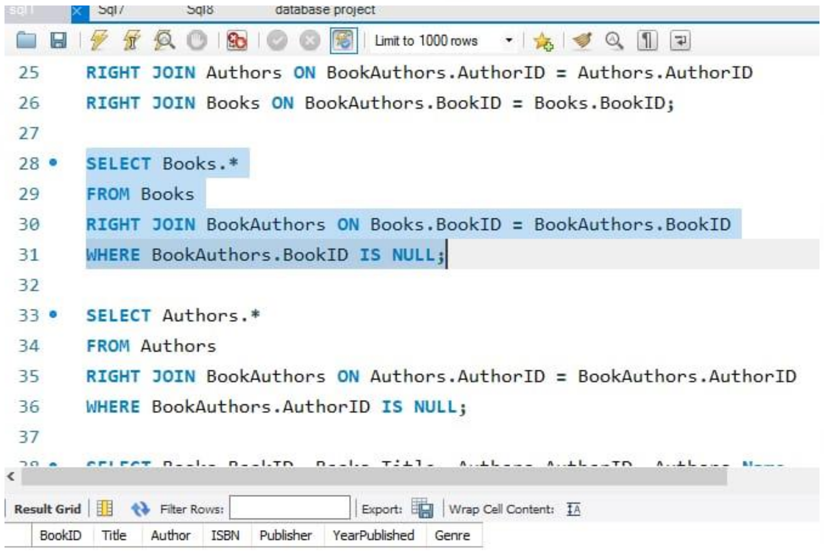
UserID	FirstName	LastName	ReservationID	ReservationDate	Status
1	John	Doe	1	2024-03-10	Pending
2	Jane	Smith	2	2024-03-12	Confirmed



```sql
INSERT INTO Reservations (UserID, BookID, ReservationDate, Status)
VALUES
(1, 2, '2024-03-10', 'Pending'),
(2, 1, '2024-03-12', 'Confirmed');
SELECT
    r.ReservationID,
    r.BookID,
    r.ReservationDate,
    r.Status,
    u.UserID,
    u.FirstName,
    u.LastName
FROM
    (SELECT * FROM Reservations) r
    LEFT JOIN (SELECT * FROM Users) u ON r.UserID = u.UserID;
```

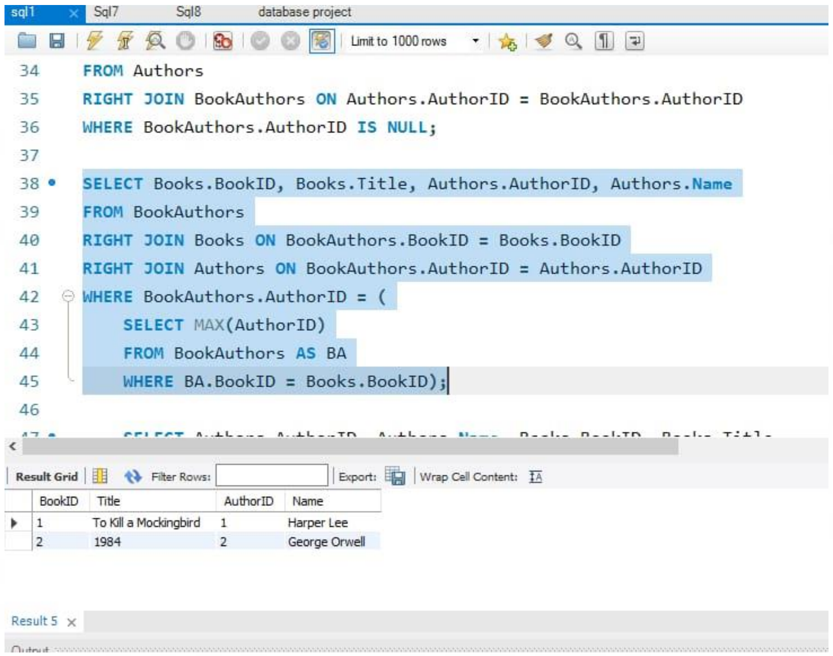
ReservationID	BookID	ReservationDate	Status	UserID	FirstName	LastName
1	2	2024-03-10	Pending	1	John	Doe
2	1	2024-03-12	Confirmed	2	Jane	Smith

Table 3:



```
RIGHT JOIN Authors ON BookAuthors.AuthorID = Authors.AuthorID
RIGHT JOIN Books ON BookAuthors.BookID = Books.BookID;

SELECT Books.*
FROM Books
RIGHT JOIN BookAuthors ON Books.BookID = BookAuthors.BookID
WHERE BookAuthors.BookID IS NULL;

SELECT Authors.*
FROM Authors
RIGHT JOIN BookAuthors ON Authors.AuthorID = BookAuthors.AuthorID
WHERE BookAuthors.AuthorID IS NULL;
```

BookID	Title	Author	ISBN	Publisher	YearPublished	Genre



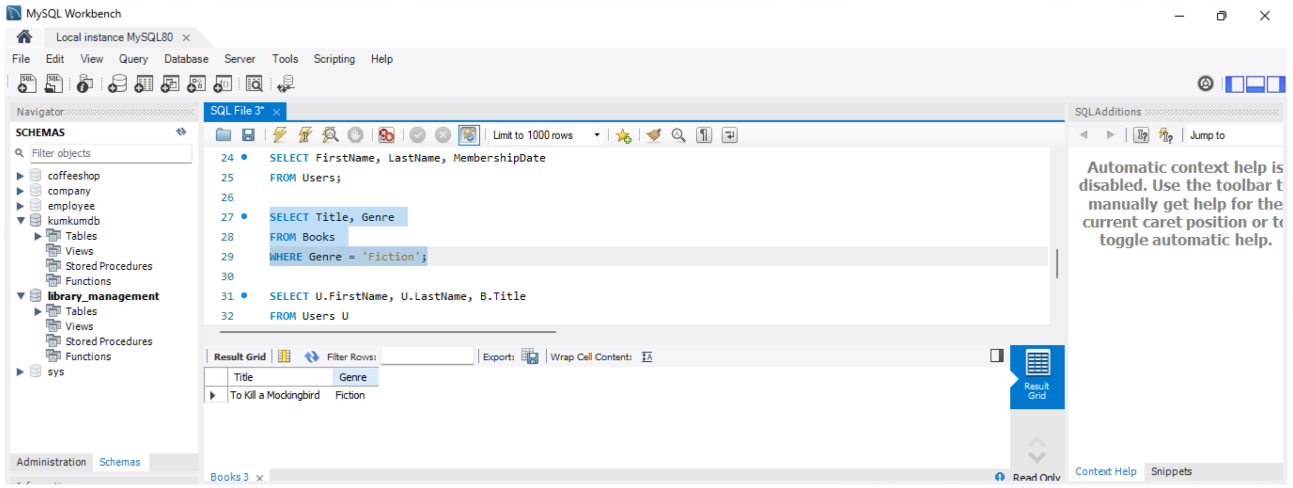
```
FROM Authors
RIGHT JOIN BookAuthors ON Authors.AuthorID = BookAuthors.AuthorID
WHERE BookAuthors.AuthorID IS NULL;

SELECT Books.BookID, Books.Title, Authors.AuthorID, Authors.Name
FROM BookAuthors
RIGHT JOIN Books ON BookAuthors.BookID = Books.BookID
RIGHT JOIN Authors ON BookAuthors.AuthorID = Authors.AuthorID
WHERE BookAuthors.AuthorID = (
    SELECT MAX(AuthorID)
    FROM BookAuthors AS BA
    WHERE BA.BookID = Books.BookID);
```

BookID	Title	AuthorID	Name
1	To Kill a Mockingbird	1	Harper Lee
2	1984	2	George Orwell

Kumkum:

1. Queries:

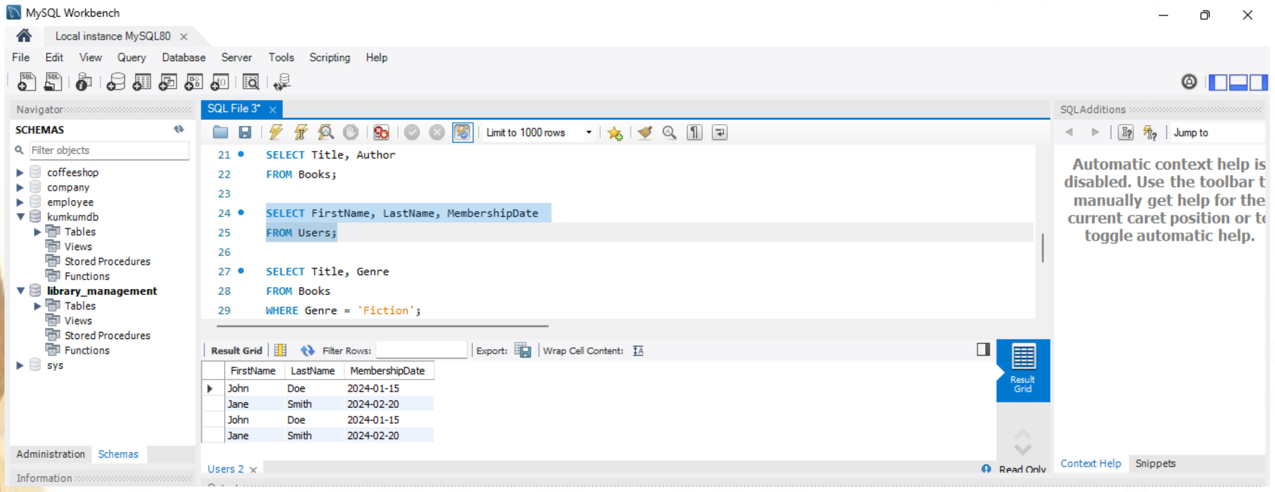
Table 4:



```
SELECT FirstName, LastName, MembershipDate
FROM Users;

SELECT Title, Genre
FROM Books
WHERE Genre = 'Fiction';

SELECT U.FirstName, U.LastName, B.Title
FROM Users U
```

Title	Genre
To Kill a Mockingbird	Fiction

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or to toggle automatic help.

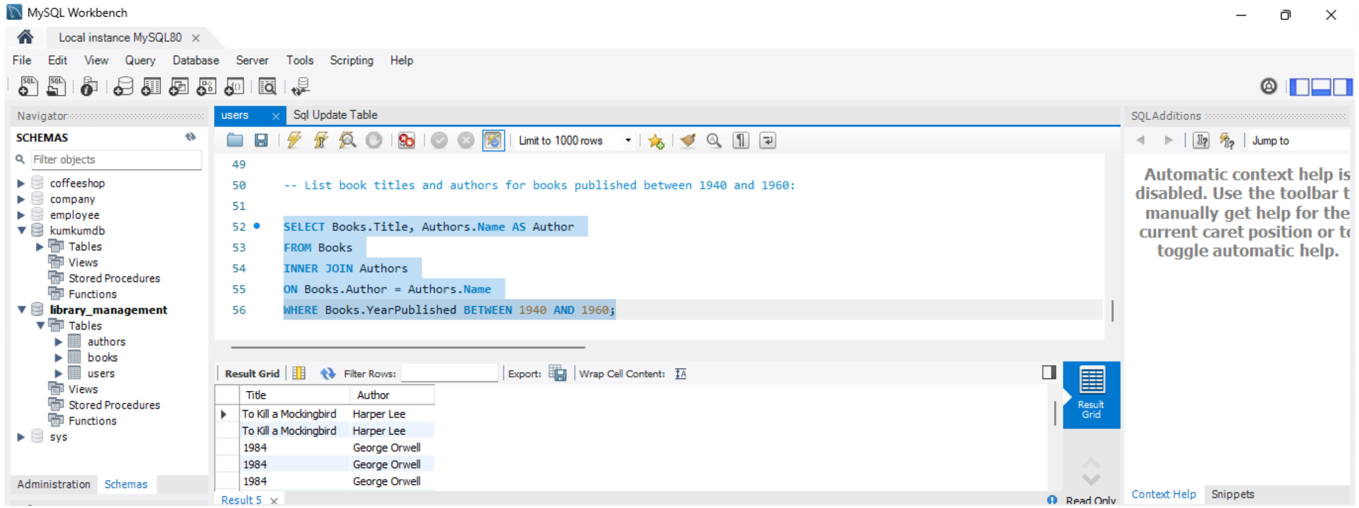


```
SELECT Title, Author
FROM Books;

SELECT FirstName, LastName, MembershipDate
FROM Users;

SELECT Title, Genre
FROM Books
WHERE Genre = 'Fiction';
```

FirstName	LastName	MembershipDate
John	Doe	2024-01-15
Jane	Smith	2024-02-20
John	Doe	2024-01-15
Jane	Smith	2024-02-20

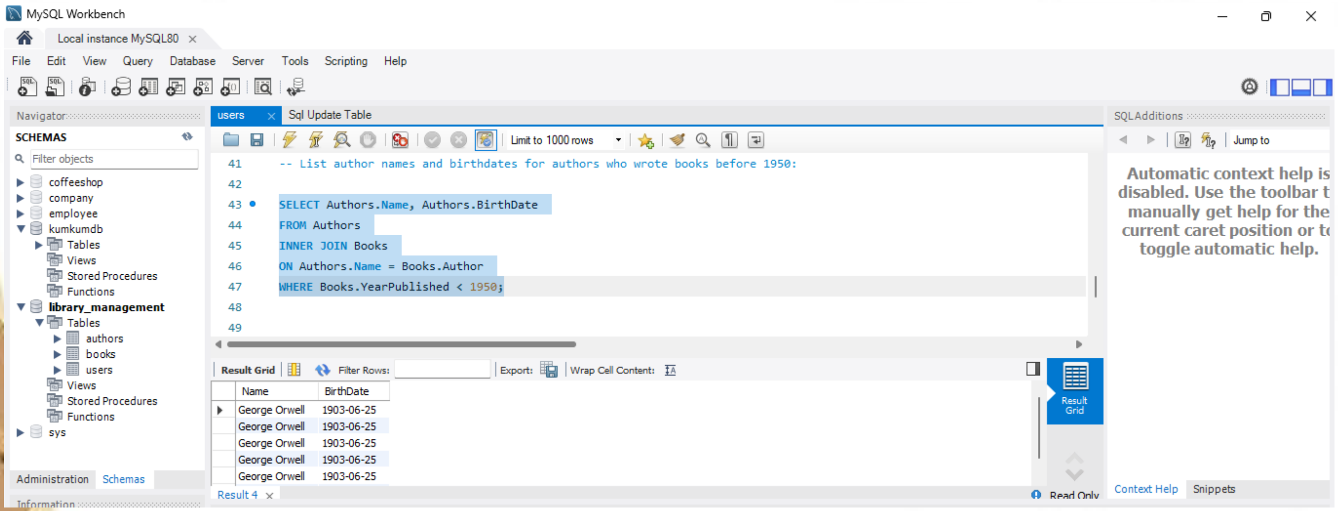
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or to toggle automatic help.

Table 5:



```sql
-- List book titles and authors for books published between 1940 and 1960:

SELECT Books.Title, Authors.Name AS Author
FROM Books
INNER JOIN Authors
ON Books.Author = Authors.Name
WHERE Books.YearPublished BETWEEN 1940 AND 1960;
```

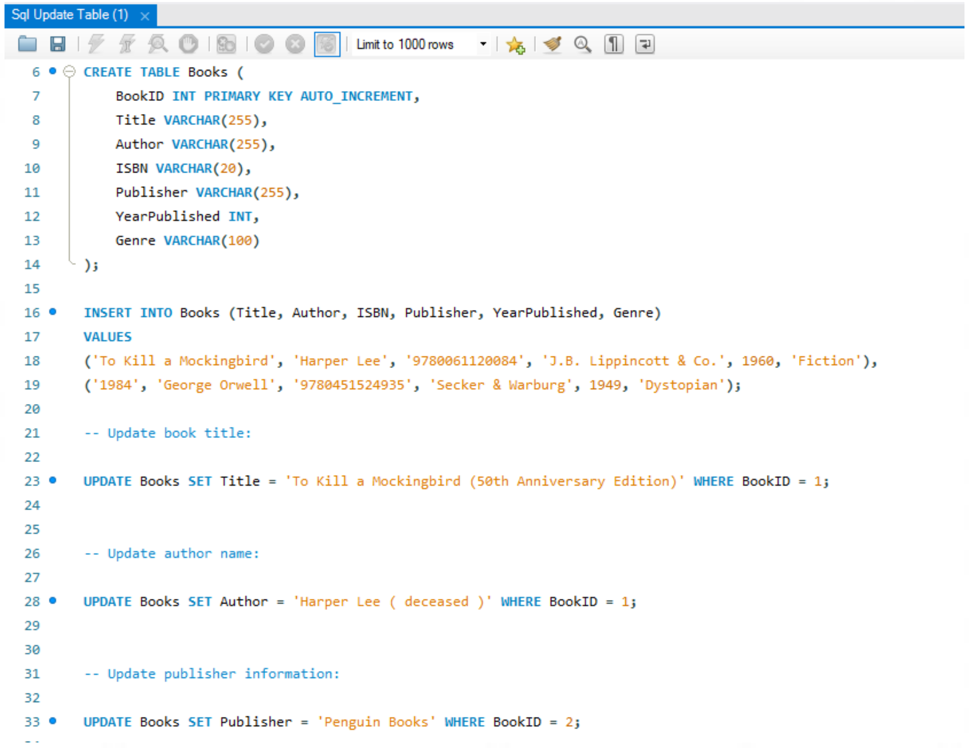
Title	Author
To Kill a Mockingbird	Harper Lee
To Kill a Mockingbird	Harper Lee
1984	George Orwell
1984	George Orwell
1984	George Orwell



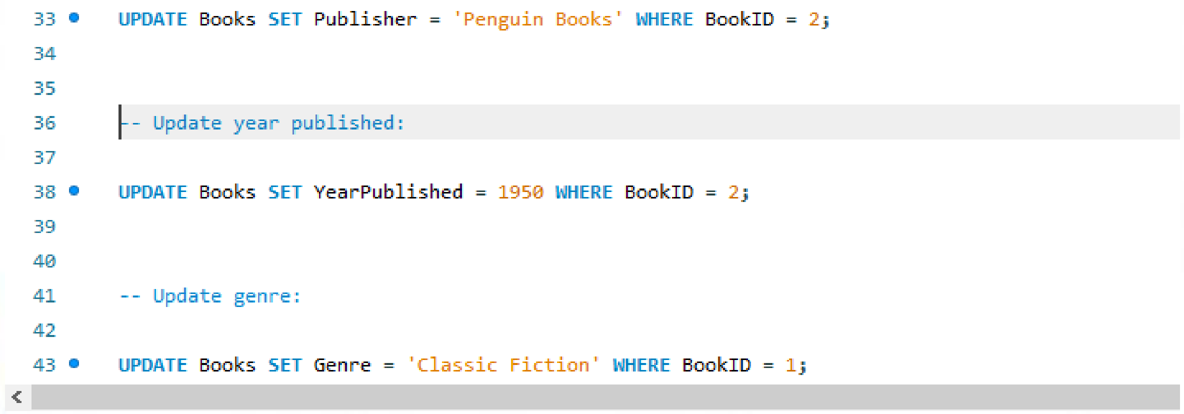
```sql
-- List author names and birthdates for authors who wrote books before 1950:

SELECT Authors.Name, Authors.BirthDate
FROM Authors
INNER JOIN Books
ON Authors.Name = Books.Author
WHERE Books.YearPublished < 1950;
```

Name	BirthDate
George Orwell	1903-06-25
George Orwell	1903-06-25
George Orwell	1903-06-25
George Orwell	1903-06-25
George Orwell	1903-06-25

Table 6:



```sql
CREATE TABLE Books (
    BookID INT PRIMARY KEY AUTO_INCREMENT,
    Title VARCHAR(255),
    Author VARCHAR(255),
    ISBN VARCHAR(20),
    Publisher VARCHAR(255),
    YearPublished INT,
    Genre VARCHAR(100)
);

INSERT INTO Books (Title, Author, ISBN, Publisher, YearPublished, Genre)
VALUES
('To Kill a Mockingbird', 'Harper Lee', '9780061120084', 'J.B. Lippincott & Co.', 1960, 'Fiction'),
('1984', 'George Orwell', '9780451524935', 'Secker & Warburg', 1949, 'Dystopian');

-- Update book title:

UPDATE Books SET Title = 'To Kill a Mockingbird (50th Anniversary Edition)' WHERE BookID = 1;


-- Update author name:

UPDATE Books SET Author = 'Harper Lee ( deceased )' WHERE BookID = 1;


-- Update publisher information:

UPDATE Books SET Publisher = 'Penguin Books' WHERE BookID = 2;
```

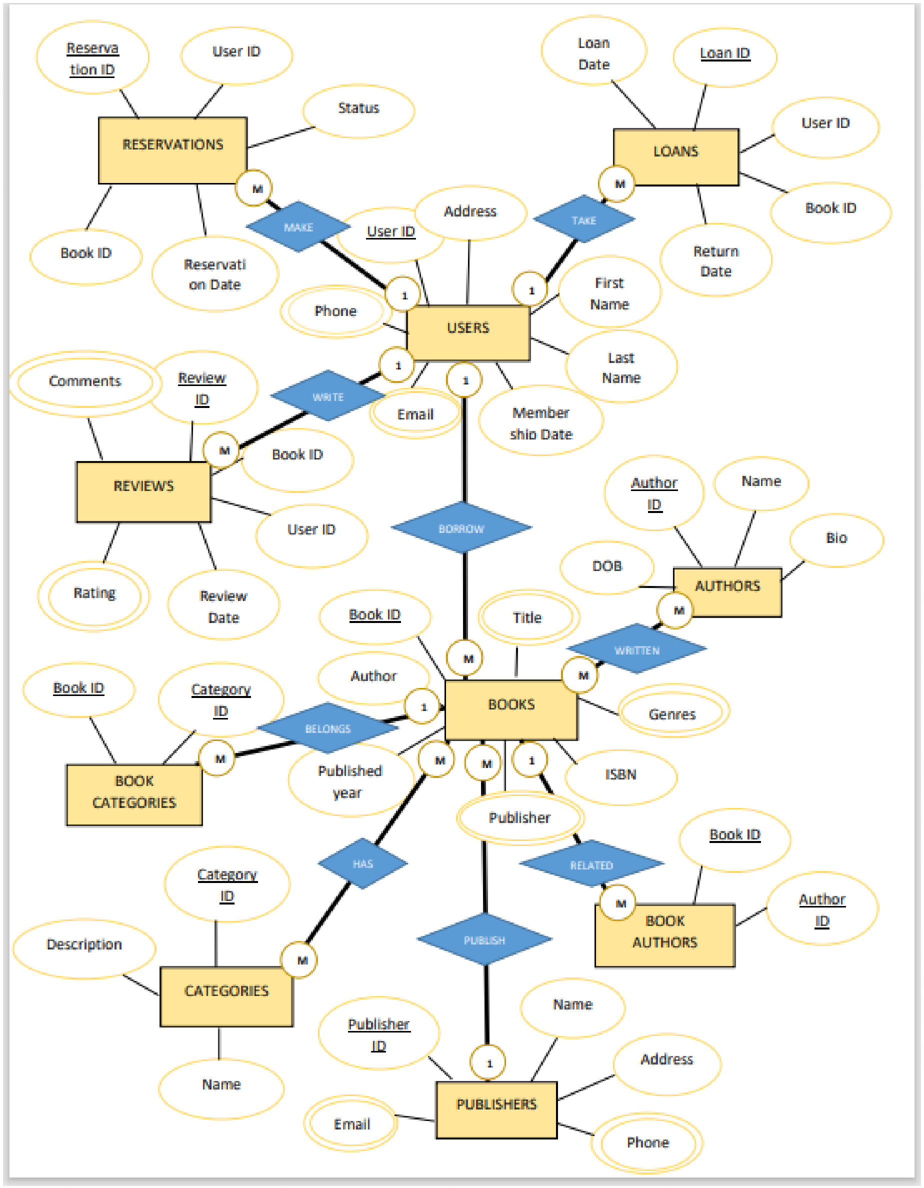


UPDATE Books SET Publisher = 'Penguin Books' WHERE BookID = 2;

```sql


-- Update year published:

UPDATE Books SET YearPublished = 1950 WHERE BookID = 2;


-- Update genre:

UPDATE Books SET Genre = 'Classic Fiction' WHERE BookID = 1;
```

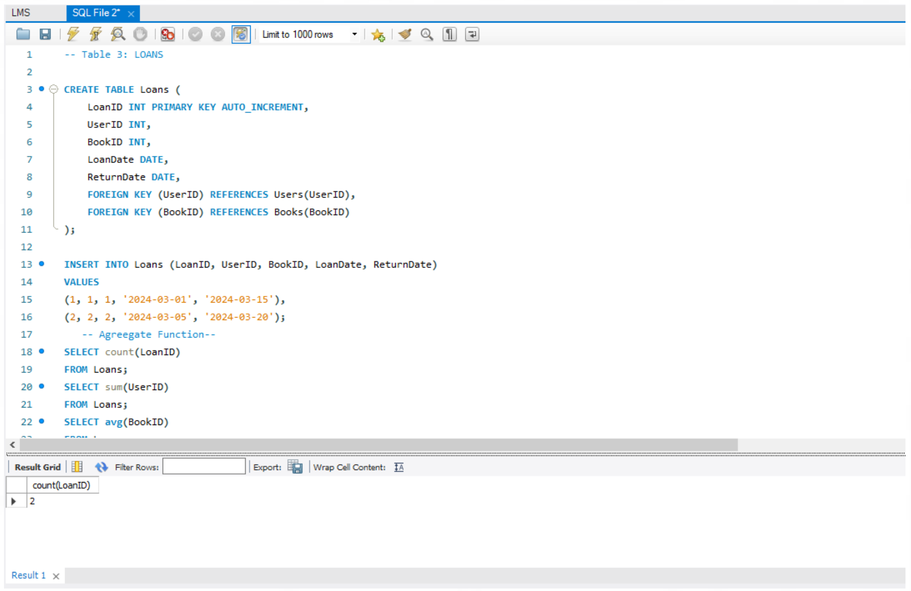
Hiba Zehra:

1. ER Model:



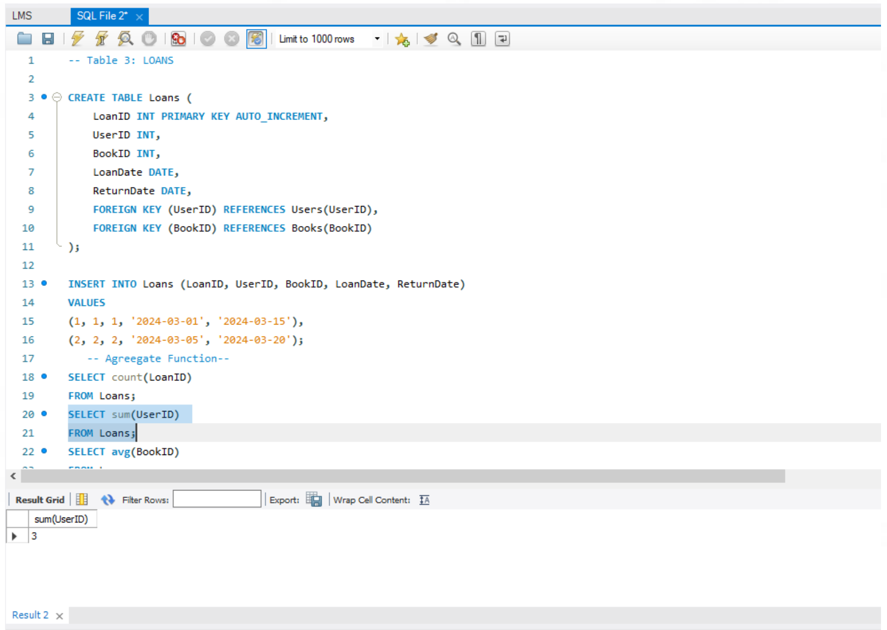
2. Queries:

Table 7:



```
-- Table 3: LOANS

CREATE TABLE Loans (
    LoanID INT PRIMARY KEY AUTO_INCREMENT,
    UserID INT,
    BookID INT,
    LoanDate DATE,
    ReturnDate DATE,
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID)
);

INSERT INTO Loans (LoanID, UserID, BookID, LoanDate, ReturnDate)
VALUES
(1, 1, 1, '2024-03-01', '2024-03-15'),
(2, 2, 2, '2024-03-05', '2024-03-20');
   -- Agreegate Function--
SELECT count(LoanID)
FROM Loans;
SELECT sum(UserID)
FROM Loans;
SELECT avg(BookID)
```

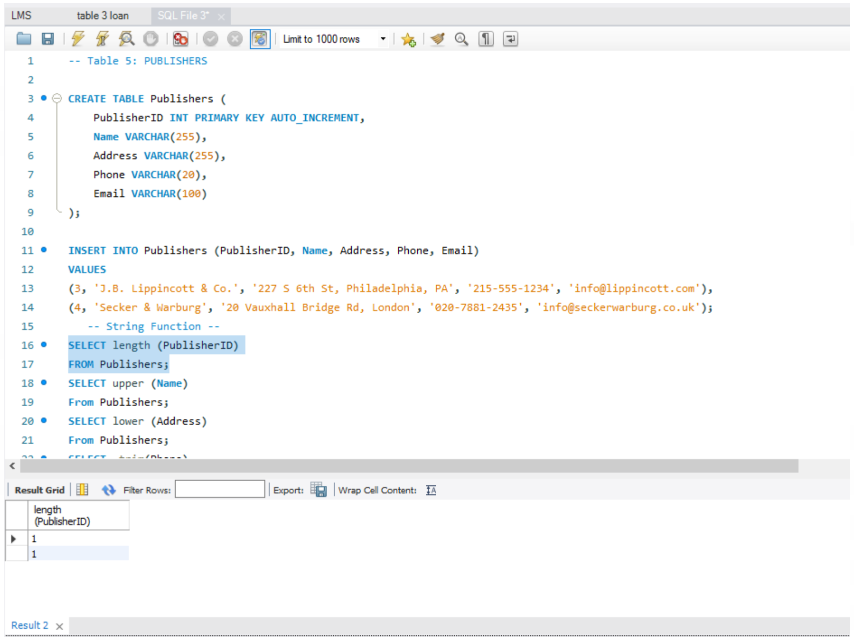
```
count(LoanID)
2
```



```
-- Table 3: LOANS

CREATE TABLE Loans (
    LoanID INT PRIMARY KEY AUTO_INCREMENT,
    UserID INT,
    BookID INT,
    LoanDate DATE,
    ReturnDate DATE,
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID)
);

INSERT INTO Loans (LoanID, UserID, BookID, LoanDate, ReturnDate)
VALUES
(1, 1, 1, '2024-03-01', '2024-03-15'),
(2, 2, 2, '2024-03-05', '2024-03-20');
   -- Agreegate Function--
SELECT count(LoanID)
FROM Loans;
SELECT sum(UserID)
FROM Loans;
SELECT avg(BookID)
```

```
sum(UserID)
3
```

Table 8:



LMS | table 3 loan | SQL File 3*

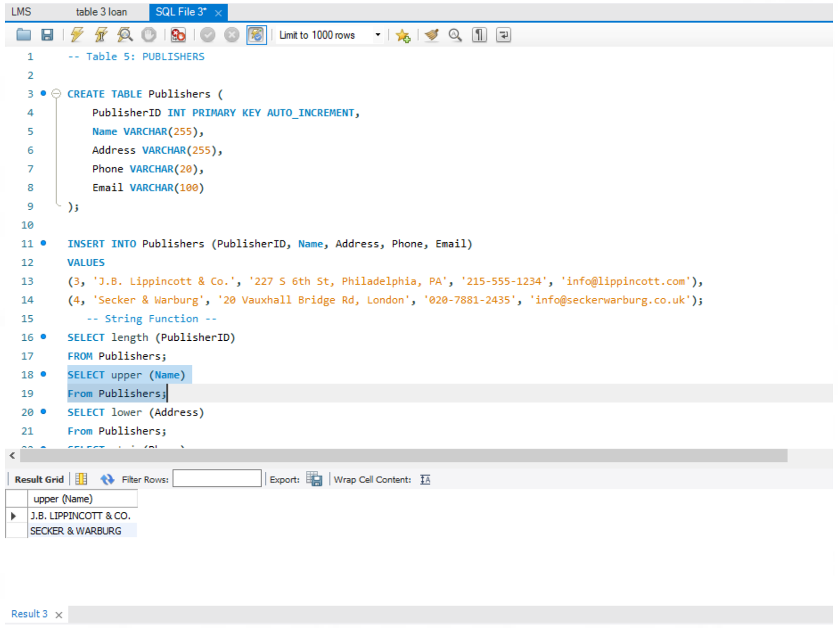
Limit to 1000 rows

```sql
-- Table 5: PUBLISHERS

CREATE TABLE Publishers (
    PublisherID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255),
    Address VARCHAR(255),
    Phone VARCHAR(20),
    Email VARCHAR(100)
);

INSERT INTO Publishers (PublisherID, Name, Address, Phone, Email)
VALUES
(3, 'J.B. Lippincott & Co.', '227 S 6th St, Philadelphia, PA', '215-555-1234', 'info@lippincott.com'),
(4, 'Secker & Warburg', '20 Vauxhall Bridge Rd, London', '020-7881-2435', 'info@seckerwarburg.co.uk');
   -- String Function --
SELECT length (PublisherID)
FROM Publishers;
SELECT upper (Name)
From Publishers;
SELECT lower (Address)
From Publishers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

length (PublisherID)
1
1

Result 2



LMS | table 3 loan | SQL File 3*

Limit to 1000 rows

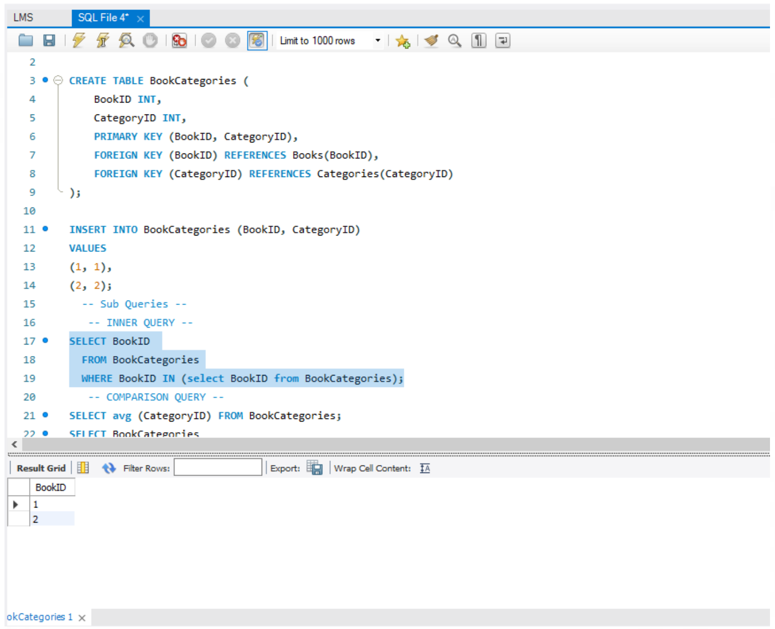
```sql
-- Table 5: PUBLISHERS

CREATE TABLE Publishers (
    PublisherID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255),
    Address VARCHAR(255),
    Phone VARCHAR(20),
    Email VARCHAR(100)
);

INSERT INTO Publishers (PublisherID, Name, Address, Phone, Email)
VALUES
(3, 'J.B. Lippincott & Co.', '227 S 6th St, Philadelphia, PA', '215-555-1234', 'info@lippincott.com'),
(4, 'Secker & Warburg', '20 Vauxhall Bridge Rd, London', '020-7881-2435', 'info@seckerwarburg.co.uk');
   -- String Function --
SELECT length (PublisherID)
FROM Publishers;
SELECT upper (Name)
From Publishers;
SELECT lower (Address)
From Publishers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

upper (Name)
J.B. LIPPINCOTT & CO.
SECKER & WARBURG

Result 3

Table 9:



LMS | SQL File 4*

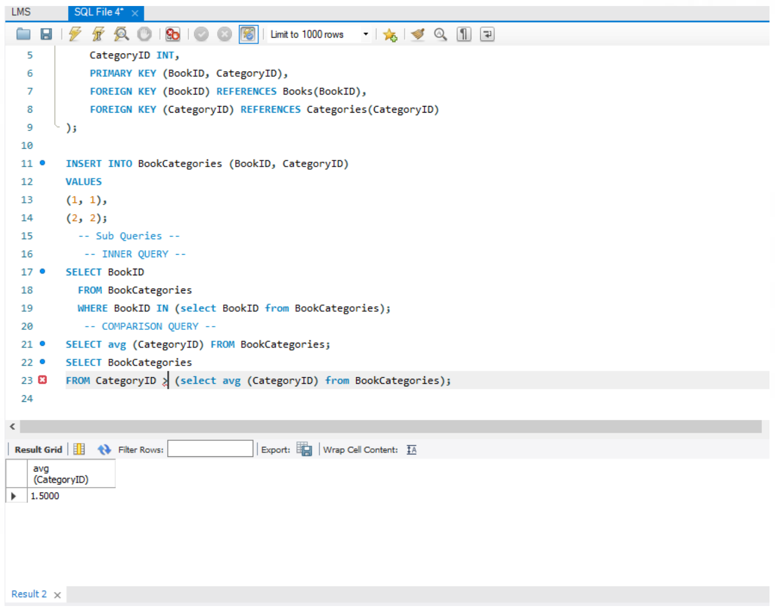
Limit to 1000 rows

```sql

CREATE TABLE BookCategories (
    BookID INT,
    CategoryID INT,
    PRIMARY KEY (BookID, CategoryID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);

INSERT INTO BookCategories (BookID, CategoryID)
VALUES
(1, 1),
(2, 2);
  -- Sub Queries --
   -- INNER QUERY --
SELECT BookID
  FROM BookCategories
  WHERE BookID IN (select BookID from BookCategories);
   -- COMPARISON QUERY --
SELECT avg (CategoryID) FROM BookCategories;
SELECT BookCategories
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
BookID
1
2
```

okCategories 1



LMS | SQL File 4*

Limit to 1000 rows

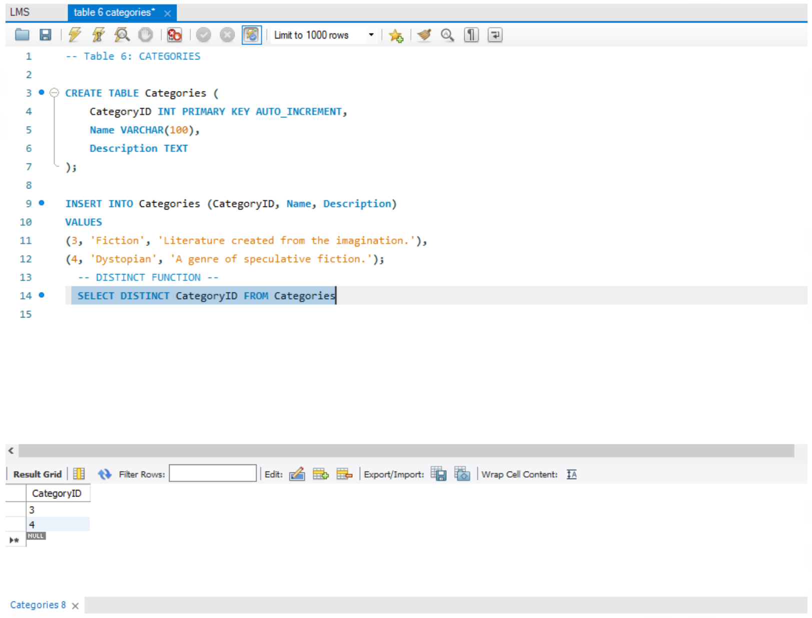
```sql
    CategoryID INT,
    PRIMARY KEY (BookID, CategoryID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);

INSERT INTO BookCategories (BookID, CategoryID)
VALUES
(1, 1),
(2, 2);
  -- Sub Queries --
   -- INNER QUERY --
SELECT BookID
  FROM BookCategories
  WHERE BookID IN (select BookID from BookCategories);
   -- COMPARISON QUERY --
SELECT avg (CategoryID) FROM BookCategories;
SELECT BookCategories
FROM CategoryID > (select avg (CategoryID) from BookCategories);

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
avg
(CategoryID)
1.5000
```

Result 2

Table 10:



LMS | table 6 categories*

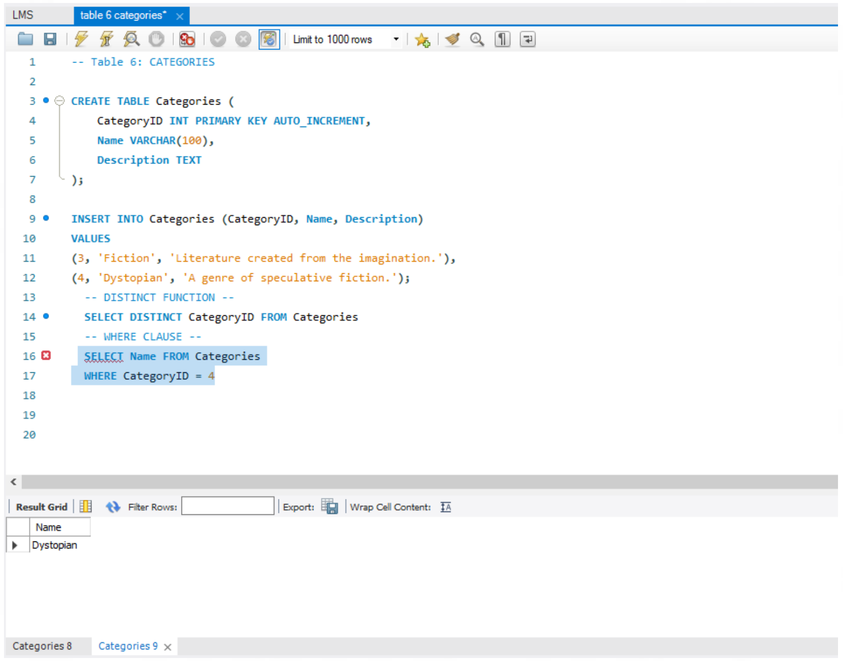
Limit to 1000 rows

```sql
-- Table 6: CATEGORIES

CREATE TABLE Categories (
    CategoryID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100),
    Description TEXT
);

INSERT INTO Categories (CategoryID, Name, Description)
VALUES
(3, 'Fiction', 'Literature created from the imagination.'),
(4, 'Dystopian', 'A genre of speculative fiction.');
  -- DISTINCT FUNCTION --
  SELECT DISTINCT CategoryID FROM Categories
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

```
CategoryID
3
4
NULL
```

Categories 8



LMS | table 6 categories*

Limit to 1000 rows

```sql
-- Table 6: CATEGORIES

CREATE TABLE Categories (
    CategoryID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100),
    Description TEXT
);

INSERT INTO Categories (CategoryID, Name, Description)
VALUES
(3, 'Fiction', 'Literature created from the imagination.'),
(4, 'Dystopian', 'A genre of speculative fiction.');
  -- DISTINCT FUNCTION --
  SELECT DISTINCT CategoryID FROM Categories
  -- WHERE CLAUSE --
  SELECT Name FROM Categories
  WHERE CategoryID = 4
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
Name
Dystopian
```

Categories 8 | Categories 9

Purpose

- To make the existing system more efficient.
- To provide a user friendly environment.
- Make functionalities of library faster.
- Provide a system where the library staff can catch defaulters and not let them escape.

Advantages

1. It keeps the record.
2. Improves method of handling books.
3. Reduction of errors.

Application

1. **Highly Secure, Scalable and Reliable.**

2. **Can be used in any library.**

Conclusion

- Library Management System is valuable tool for educational institutions, enabling efficient management of library resources and operations.

Thank You