



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.

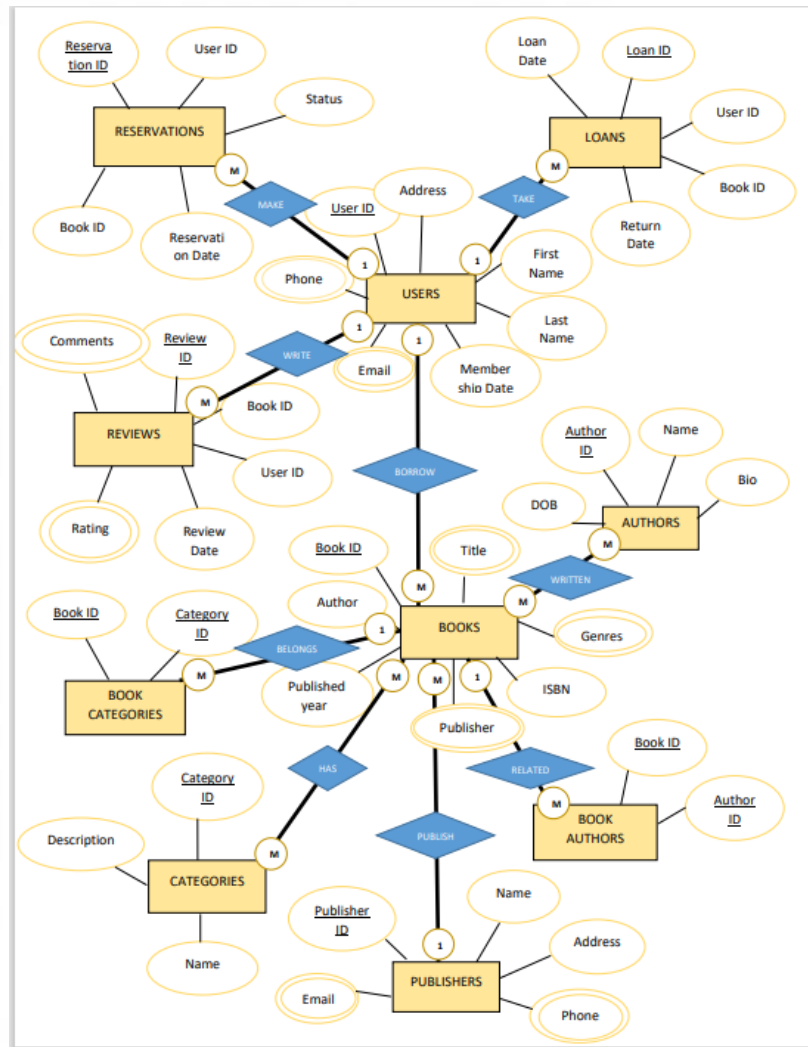


Entity-Relationship Diagram (ERD)

- **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.**



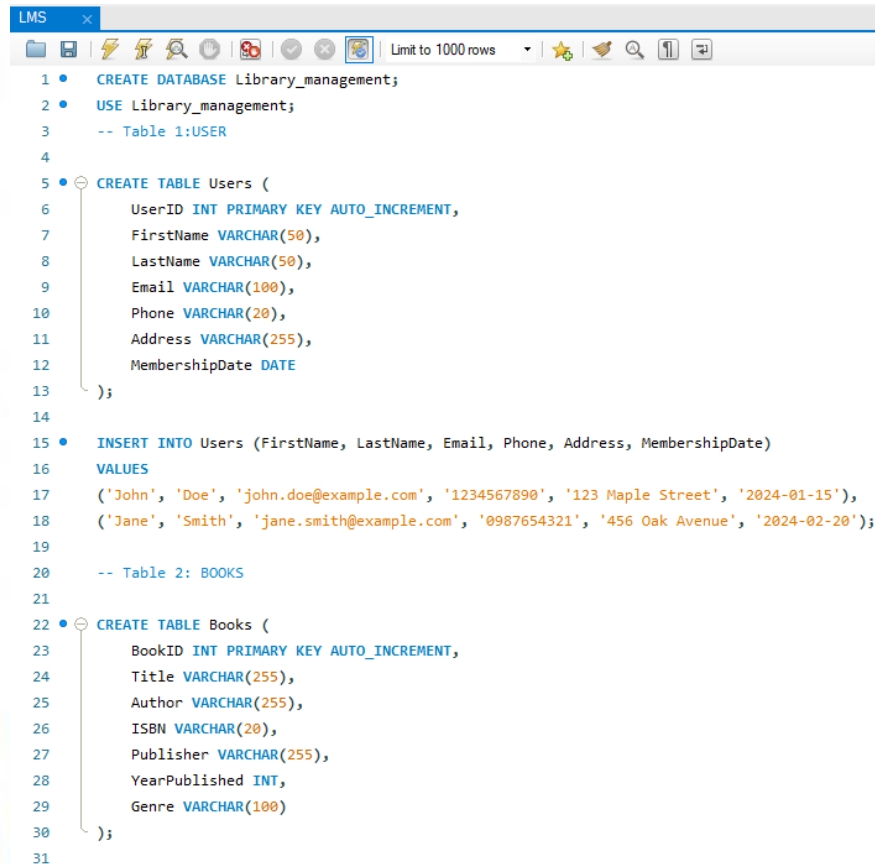
ER Model



Contributions

Zainab Anwar:

1. SQL Main File:



```
LMS
1 • CREATE DATABASE Library_management;
2 • USE Library_management;
3   -- Table 1:USER
4
5 • CREATE TABLE Users (
6     UserID INT PRIMARY KEY AUTO_INCREMENT,
7     FirstName VARCHAR(50),
8     LastName VARCHAR(50),
9     Email VARCHAR(100),
10    Phone VARCHAR(20),
11    Address VARCHAR(255),
12    MembershipDate DATE
13  );
14
15 • INSERT INTO Users (FirstName, LastName, Email, Phone, Address, MembershipDate)
16   VALUES
17   ('John', 'Doe', 'john.doe@example.com', '1234567890', '123 Maple Street', '2024-01-15'),
18   ('Jane', 'Smith', 'jane.smith@example.com', '0987654321', '456 Oak Avenue', '2024-02-20');
19
20   -- Table 2: BOOKS
21
22 • CREATE TABLE Books (
23     BookID INT PRIMARY KEY AUTO_INCREMENT,
24     Title VARCHAR(255),
25     Author VARCHAR(255),
26     ISBN VARCHAR(20),
27     Publisher VARCHAR(255),
28     YearPublished INT,
29     Genre VARCHAR(100)
30  );
31
```

```
LMS x
Limit to 1000 rows

31
32 • INSERT INTO Books (Title, Author, ISBN, Publisher, YearPublished, Genre)
33 VALUES
34 ('To Kill a Mockingbird', 'Harper Lee', '9780061120084', 'J.B. Lippincott & Co.', 1960, 'Fiction'),
35 ('1984', 'George Orwell', '9780451524935', 'Secker & Warburg', 1949, 'Dystopian');
36
37 -- Table 3: LOANS
38
39 • CREATE TABLE Loans (
40     LoanID INT PRIMARY KEY AUTO_INCREMENT,
41     UserID INT,
42     BookID INT,
43     LoanDate DATE,
44     ReturnDate DATE,
45     FOREIGN KEY (UserID) REFERENCES Users(UserID),
46     FOREIGN KEY (BookID) REFERENCES Books(BookID)
47 );
48
49 • INSERT INTO Loans (LoanID, UserID, BookID, LoanDate, ReturnDate)
50 VALUES
51 (1, 1, 1, '2024-03-01', '2024-03-15'),
52 (2, 2, 2, '2024-03-05', '2024-03-20');
53
54 -- Table 4: AUTHORS
55
56 • CREATE TABLE Authors (
57     AuthorID INT PRIMARY KEY AUTO_INCREMENT,
58     Name VARCHAR(255),
59     Bio TEXT,
60     BirthDate DATE
61 );
```

```
LMS x
Limit to 1000 rows

63 • INSERT INTO Authors (Name, Bio, BirthDate)
64 VALUES
65 ('Harper Lee', 'American novelist widely known for To Kill a Mockingbird.', '1926-04-28'),
66 ('George Orwell', 'English novelist, essayist, journalist and critic.', '1903-06-25');
67
68 -- Table 5: PUBLISHERS
69
70 • CREATE TABLE Publishers (
71     PublisherID INT PRIMARY KEY AUTO_INCREMENT,
72     Name VARCHAR(255),
73     Address VARCHAR(255),
74     Phone VARCHAR(20),
75     Email VARCHAR(100)
76 );
77
78 • INSERT INTO Publishers (PublisherID, Name, Address, Phone, Email)
79 VALUES
80 (3, 'J.B. Lippincott & Co.', '227 S 6th St, Philadelphia, PA', '215-555-1234', 'info@lippincott.com'),
81 (4, 'Secker & Warburg', '20 Vauxhall Bridge Rd, London', '020-7881-2435', 'info@seckerwarburg.co.uk');
82
83 -- Table 6: CATEGORIES
84
85 • CREATE TABLE Categories (
86     CategoryID INT PRIMARY KEY AUTO_INCREMENT,
87     Name VARCHAR(100),
88     Description TEXT
89 );
90
91 • INSERT INTO Categories (CategoryID, Name, Description)
92 VALUES
93 (3, 'Fiction', 'Literature created from the imagination.'),
```

```
LMS x
Limit to 1000 rows
94 (4, 'Dystopian', 'A genre of speculative fiction.');
```

-- Table 7: BOOK CATEGORIES

```
97
98 • CREATE TABLE BookCategories (
99     BookID INT,
100     CategoryID INT,
101     PRIMARY KEY (BookID, CategoryID),
102     FOREIGN KEY (BookID) REFERENCES Books(BookID),
103     FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
104 );
105
106 • INSERT INTO BookCategories (BookID, CategoryID)
107 VALUES
108 (1, 1),
109 (2, 2);
110
111 -- Table 8: BOOK AUTHORS
112
113 • CREATE TABLE BookAuthors (
114     BookID INT,
115     AuthorID INT,
116     PRIMARY KEY (BookID, AuthorID),
117     FOREIGN KEY (BookID) REFERENCES Books(BookID),
118     FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
119 );
120
121 • INSERT INTO BookAuthors (BookID, AuthorID)
122 VALUES
123 (1, 1),
124 (2, 2);
```

```
LMS x
Limit to 1000 rows
126 -- Table 9: RESERVATIONS
127
128 • CREATE TABLE Reservations (
129     ReservationID INT PRIMARY KEY AUTO_INCREMENT,
130     UserID INT,
131     BookID INT,
132     ReservationDate DATE,
133     Status VARCHAR(20),
134     FOREIGN KEY (UserID) REFERENCES Users(UserID),
135     FOREIGN KEY (BookID) REFERENCES Books(BookID)
136 );
137
138 • INSERT INTO Reservations (UserID, BookID, ReservationDate, Status)
139 VALUES
140 (1, 2, '2024-03-10', 'Pending'),
141 (2, 1, '2024-03-12', 'Confirmed');
142
143 -- Table 10: REVIEWS
144
145 • CREATE TABLE Reviews (
146     ReviewID INT PRIMARY KEY AUTO_INCREMENT,
147     UserID INT,
148     BookID INT,
149     ReviewDate DATE,
150     Rating INT CHECK (Rating >= 1 AND Rating <= 5),
151     Comments TEXT,
152     FOREIGN KEY (UserID) REFERENCES Users(UserID),
153     FOREIGN KEY (BookID) REFERENCES Books(BookID)
154 );
```


155

```
156 • INSERT INTO Reviews (UserID, BookID, ReviewDate, Rating, Comments)
157 VALUES
158 (1, 1, '2024-03-20', 5, 'An excellent read.'),
159 (2, 2, '2024-03-22', 4, 'Thought-provoking and well-written.');
```

2. Queries:

Table 1:

[illegible]



Sql1 Sql7 **Sql8** database project

Limit to 1000 rows

```
12 • INSERT INTO Reviews (UserID, BookID, ReviewDate, Rating, Comments)
13     VALUES
14     (1, 1, '2024-03-20', 5, 'An excellent read.'),
15     (2, 2, '2024-03-22', 4, 'Thought-provoking and well-written.');
```

```
17 • SELECT * FROM Reviews
18     WHERE BookID = 1;
```

```
20 • SELECT BookID, AVG(Rating) AS AverageRating
21     FROM Reviews
22     WHERE BookID = 1
23     GROUP BY BookID;
```

```
25 • SELECT * FROM Reviews
26     WHERE BookID = 1
27     ORDER BY ReviewDate DESC
```

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

	BookID	AverageRating
▶	1	5.0000

Result 2 x

sql1 Sql7 **Sql8** database project

Limit to 1000 rows

```
21 FROM Reviews
22 WHERE BookID = 1
23 GROUP BY BookID;
```

```
25 • SELECT * FROM Reviews
26     WHERE BookID = 1
27     ORDER BY ReviewDate DESC
28     LIMIT 1;
```

```
30 • SELECT * FROM Reviews
31     WHERE UserID = 1;
```

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

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

	BookID	ReviewCount	AverageRating
▶	1	1	5.0000
	2	1	4.0000

Result 5 x

Table 2:

sql1 | Sql7 | Sql8 | database project

Limit to 1000 rows

```
49 • SELECT
50   u.UserID,
51   u.FirstName,
52   u.LastName,
53   (SELECT r.ReservationID FROM Reservations r WHERE r.UserID = u.UserID ORDER BY r.ReservationDate DESC LIMIT 1) AS ReservationID,
54   (SELECT r.ReservationDate FROM Reservations r WHERE r.UserID = u.UserID ORDER BY r.ReservationDate DESC LIMIT 1) AS ReservationDate,
55   (SELECT r.Status FROM Reservations r WHERE r.UserID = u.UserID ORDER BY r.ReservationDate DESC LIMIT 1) AS Status
56 FROM
57   Users u;
58
59 • SELECT
60   u.UserID,
61   u.FirstName,
62   u.LastName,
63   COALESCE((SELECT COUNT(r.ReservationID) FROM Reservations r WHERE r.UserID = u.UserID AND r.ReservationDate >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH),
64 FROM
```

Result Grid

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

Result 4 x

Activate Windows

Read Only

sql1 | Sql7 | Sql8 | database project

Limit to 1000 rows

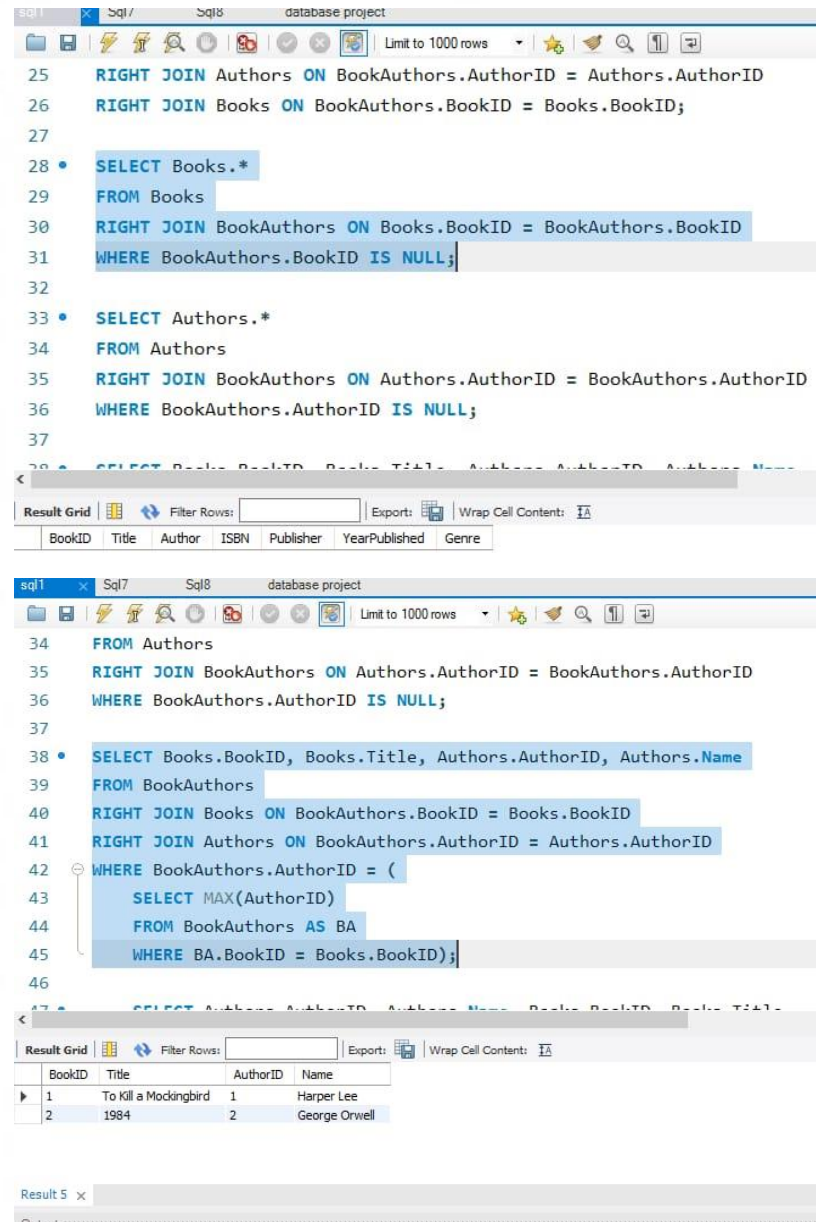
```
13 • INSERT INTO Reservations (UserID, BookID, ReservationDate, Status)
14   VALUES
15   (1, 2, '2024-03-10', 'Pending'),
16   (2, 1, '2024-03-12', 'Confirmed');
17 • SELECT
18   r.ReservationID,
19   r.BookID,
20   r.ReservationDate,
21   r.Status,
22   u.UserID,
23   u.FirstName,
24   u.LastName
25 FROM
26   (SELECT * FROM Reservations) r
27   LEFT JOIN (SELECT * FROM Users) u ON r.UserID = u.UserID;
28
```

Result Grid

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

Result 1 x

Table 3:



```
25 RIGHT JOIN Authors ON BookAuthors.AuthorID = Authors.AuthorID
26 RIGHT JOIN Books ON BookAuthors.BookID = Books.BookID;
27
28 • SELECT Books.*
29 FROM Books
30 RIGHT JOIN BookAuthors ON Books.BookID = BookAuthors.BookID
31 WHERE BookAuthors.BookID IS NULL;
32
33 • SELECT Authors.*
34 FROM Authors
35 RIGHT JOIN BookAuthors ON Authors.AuthorID = BookAuthors.AuthorID
36 WHERE BookAuthors.AuthorID IS NULL;
37
38 • SELECT Books.BookID, Books.Title, Authors.AuthorID, Authors.Name
39 FROM BookAuthors
40 RIGHT JOIN Books ON BookAuthors.BookID = Books.BookID
41 RIGHT JOIN Authors ON BookAuthors.AuthorID = Authors.AuthorID
42 WHERE BookAuthors.AuthorID = (
43     SELECT MAX(AuthorID)
44     FROM BookAuthors AS BA
45     WHERE BA.BookID = Books.BookID);
46
47 • SELECT Authors.AuthorID, Authors.Name, Books.BookID, Books.Title
```

Result Grid

BookID	Title	Author	ISBN	Publisher	YearPublished	Genre
1	To Kill a Mockingbird	1				
2	1984	2				

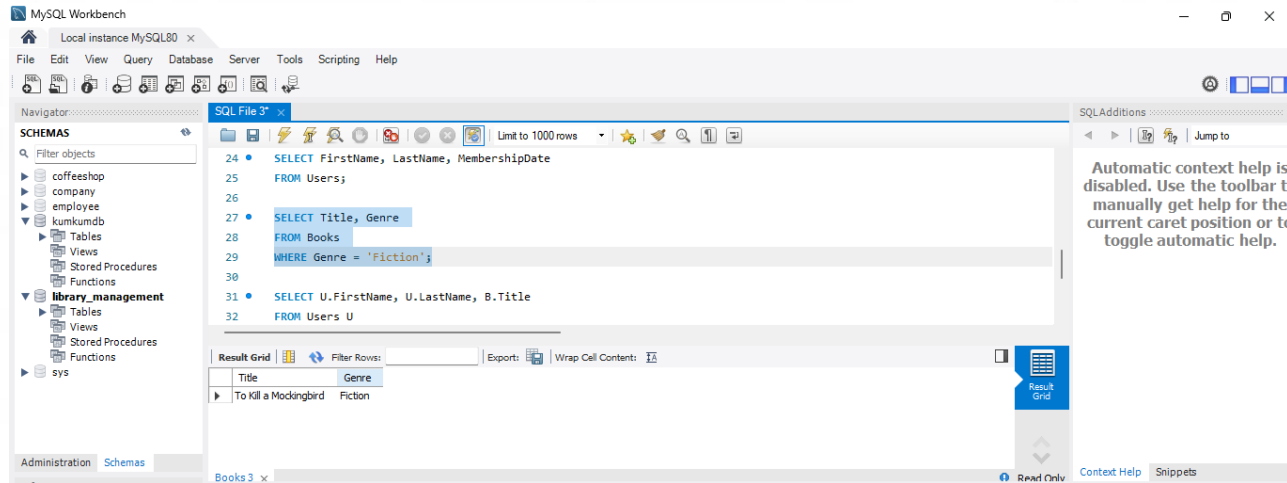
Result 5 x

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

Kumkum:

1. Queries:

Table 4:



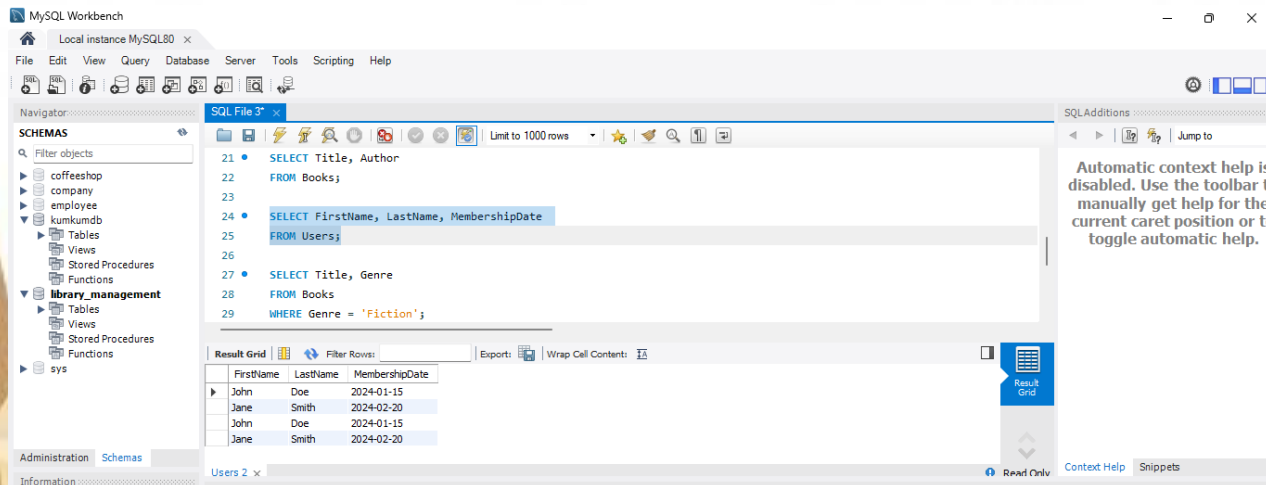
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'kumkumb' selected. The main editor window contains the following SQL query:

```
24 SELECT FirstName, LastName, MembershipDate
25 FROM Users;
26
27 SELECT Title, Genre
28 FROM Books
29 WHERE Genre = 'Fiction';
30
31 SELECT U.FirstName, U.LastName, B.Title
32 FROM Users U
```

The 'Result Grid' at the bottom shows the results for the second query:

Title	Genre
To Kill a Mockingbird	Fiction

The right sidebar contains a text box with the message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'kumkumb' selected. The main editor window contains the following SQL query:

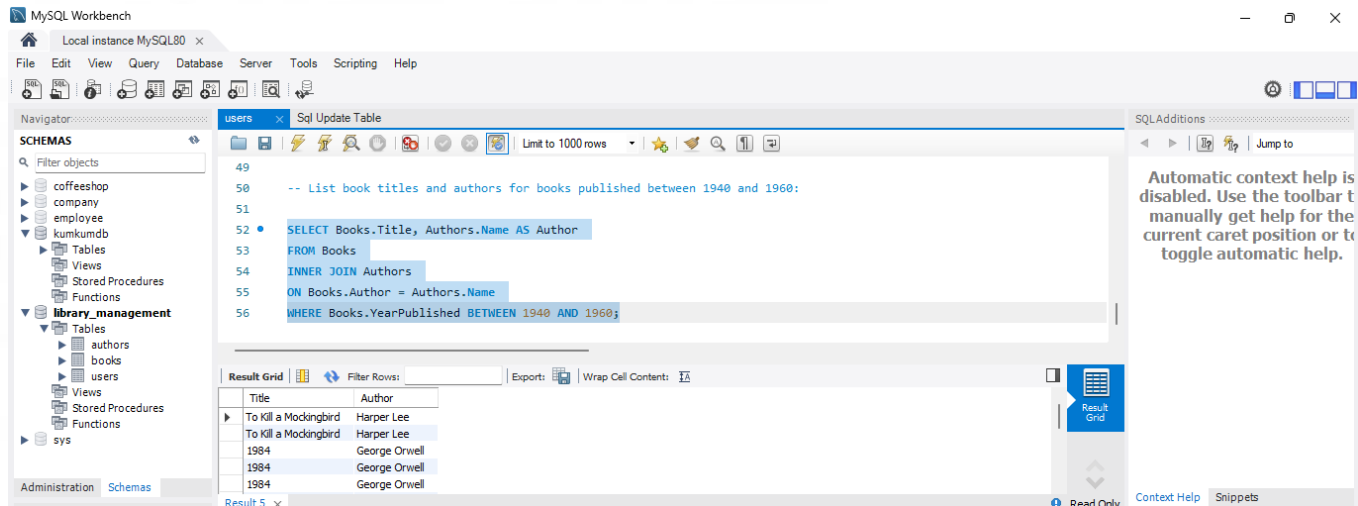
```
21 SELECT Title, Author
22 FROM Books;
23
24 SELECT FirstName, LastName, MembershipDate
25 FROM Users;
26
27 SELECT Title, Genre
28 FROM Books
29 WHERE Genre = 'Fiction';
```

The 'Result Grid' at the bottom shows the results for the second query:

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

The right sidebar contains a text box with the message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Table 5:



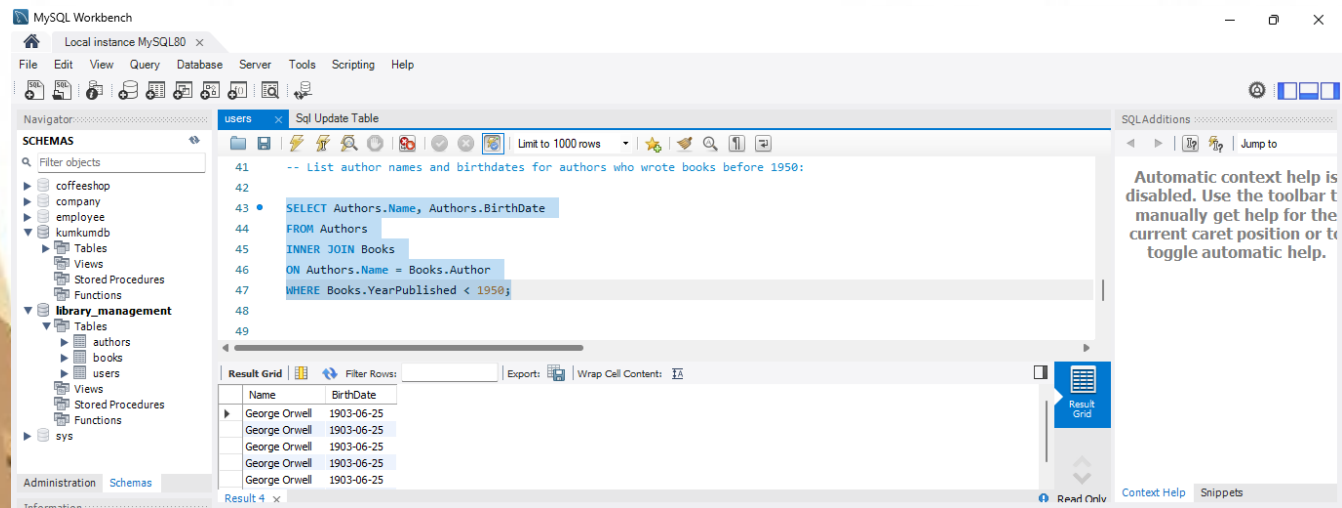
MySQL Workbench interface showing a query in the SQL Update Table editor. The query is:

```
-- 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;
```

The result grid displays the following data:

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

SQLAdditions sidebar text: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.



MySQL Workbench interface showing a query in the SQL Update Table editor. The query is:

```
-- 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;
```

The result grid displays the following data:

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
George Orwell	1903-06-25

SQLAdditions sidebar text: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

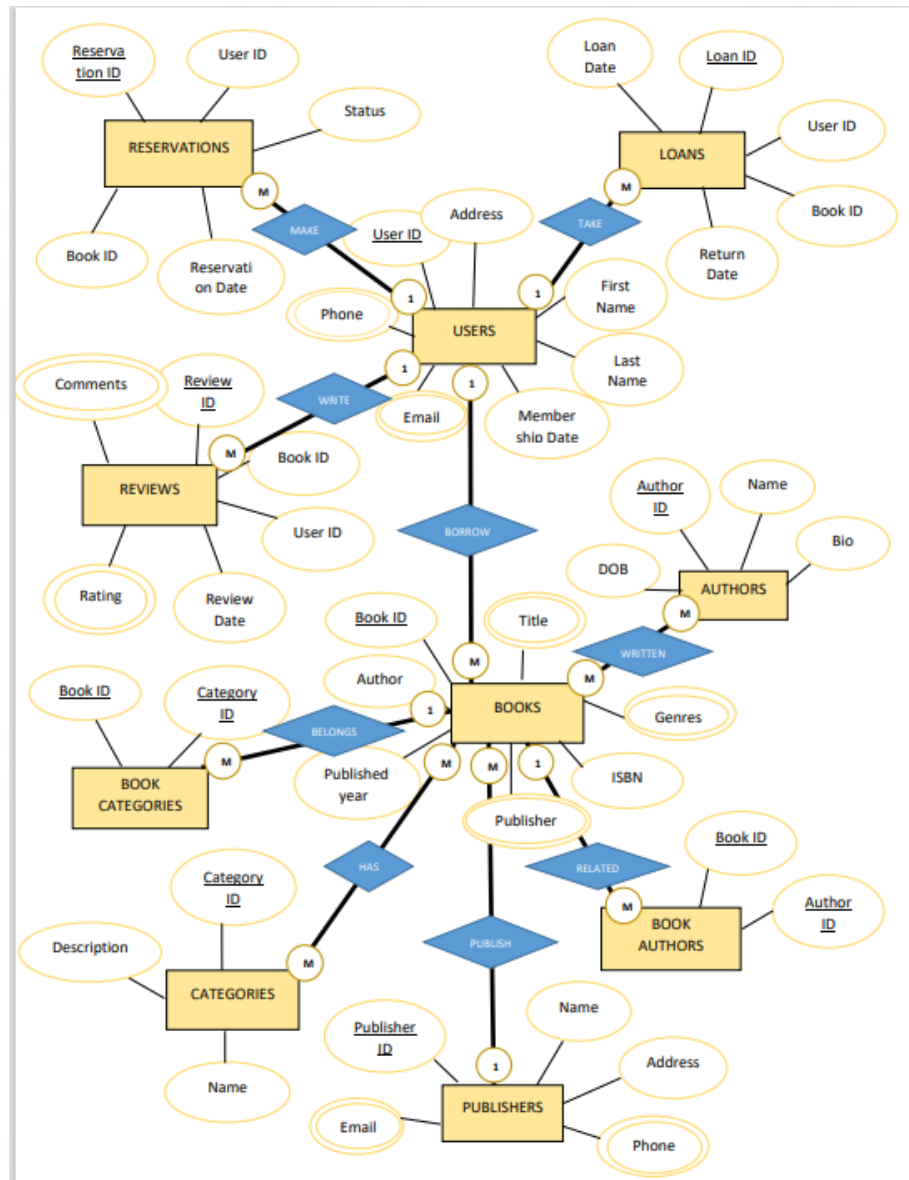
Table 6:



```
6 CREATE TABLE Books (  
7     BookID INT PRIMARY KEY AUTO_INCREMENT,  
8     Title VARCHAR(255),  
9     Author VARCHAR(255),  
10    ISBN VARCHAR(20),  
11    Publisher VARCHAR(255),  
12    YearPublished INT,  
13    Genre VARCHAR(100)  
14 );  
15  
16 INSERT INTO Books (Title, Author, ISBN, Publisher, YearPublished, Genre)  
17 VALUES  
18 ('To Kill a Mockingbird', 'Harper Lee', '9780061120084', 'J.B. Lippincott & Co.', 1960, 'Fiction'),  
19 ('1984', 'George Orwell', '9780451524935', 'Secker & Warburg', 1949, 'Dystopian');  
20  
21 -- Update book title:  
22  
23 UPDATE Books SET Title = 'To Kill a Mockingbird (50th Anniversary Edition)' WHERE BookID = 1;  
24  
25  
26 -- Update author name:  
27  
28 UPDATE Books SET Author = 'Harper Lee ( deceased )' WHERE BookID = 1;  
29  
30  
31 -- Update publisher information:  
32  
33 UPDATE Books SET Publisher = 'Penguin Books' WHERE BookID = 2;  
34  
35  
36 -- Update year published:  
37  
38 UPDATE Books SET YearPublished = 1950 WHERE BookID = 2;  
39  
40  
41 -- Update genre:  
42  
43 UPDATE Books SET Genre = 'Classic Fiction' WHERE BookID = 1;
```

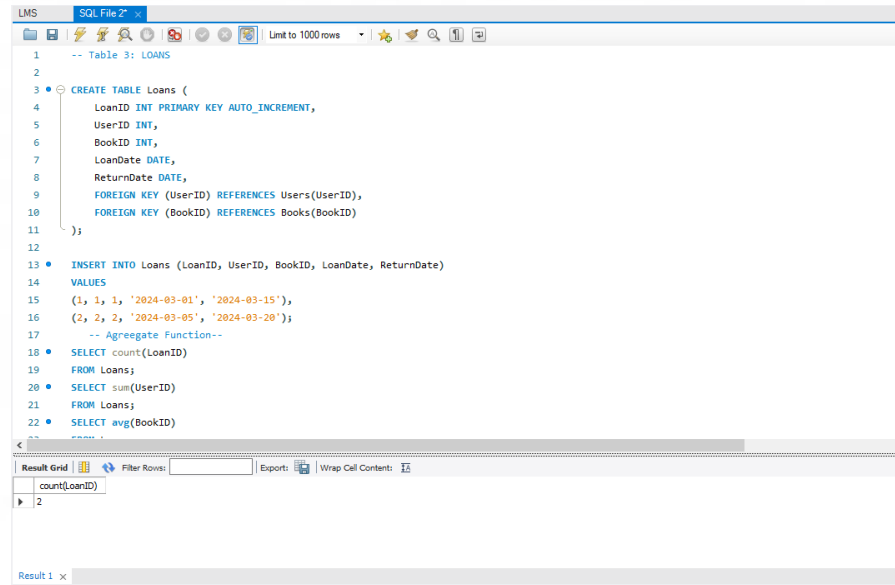
Hiba Zehra:

1. ER Model:



2. Queries:

Table 7:



The screenshot shows a SQL editor window titled "SQL File 2" with a toolbar and a "Limit to 1000 rows" dropdown. The SQL code is as follows:

```
-- 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)
);

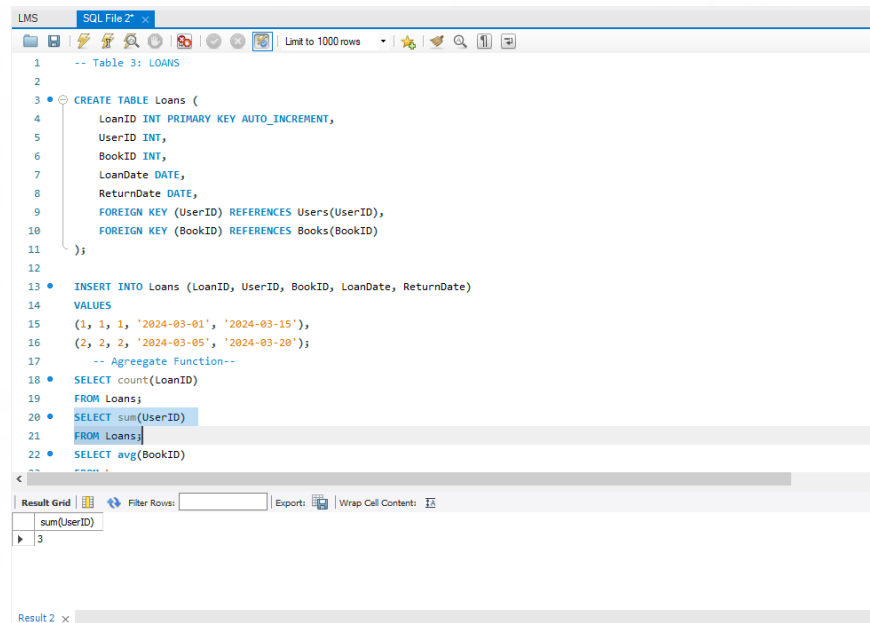
INSERT INTO Loans (LoanID, UserID, BookID, LoanDate, ReturnDate)
VALUES
(1, 1, 1, '2024-03-01', '2024-03-15'),
(2, 2, 2, '2024-03-05', '2024-03-20');

-- Aggregate Function--
SELECT count(LoanID)
FROM Loans;
SELECT sum(UserID)
FROM Loans;
SELECT avg(BookID)
```

The "Result Grid" at the bottom shows the results of the first query:

count(LoanID)
2

Below the result grid, it says "Result 1 x".

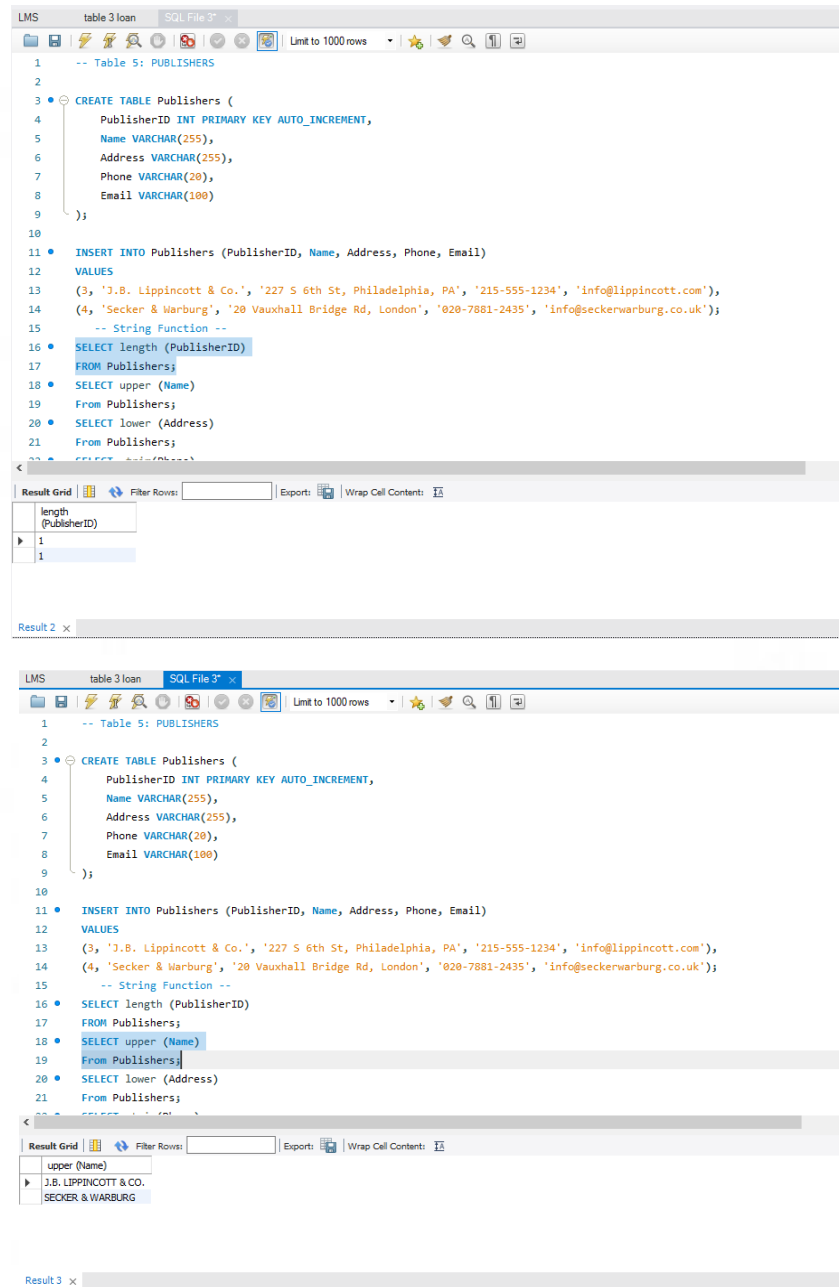


The screenshot shows the same SQL editor window with the same SQL code as above. The "Result Grid" now shows the results of the second query:

sum(UserID)
3

Below the result grid, it says "Result 2 x".

Table 8:



The screenshot displays two SQL queries and their results in a SQL IDE. The first query creates a table named Publishers and inserts data. The second query selects the length of the PublisherID field.

Query 1:

```
-- 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;
```

Result 1:

length (PublisherID)
1

Query 2:

```
-- 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 upper (Name)
FROM Publishers;
```

Result 2:

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

Table 9:

```
LMS SQL File4* x
Limit to 1000 rows

2
3 CREATE TABLE BookCategories (
4     BookID INT,
5     CategoryID INT,
6     PRIMARY KEY (BookID, CategoryID),
7     FOREIGN KEY (BookID) REFERENCES Books(BookID),
8     FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
9 );
10
11 INSERT INTO BookCategories (BookID, CategoryID)
12 VALUES
13 (1, 1),
14 (2, 2);
15 -- Sub Queries --
16 -- INNER QUERY --
17 SELECT BookID
18 FROM BookCategories
19 WHERE BookID IN (select BookID from BookCategories);
20 -- COMPARISON QUERY --
21 SELECT avg (CategoryID) FROM BookCategories;
22 SELECT BookCategories
```

Result Grid

BookID
1
2

okCategories 1 x

```
LMS SQL File4* x
Limit to 1000 rows

5     CategoryID INT,
6     PRIMARY KEY (BookID, CategoryID),
7     FOREIGN KEY (BookID) REFERENCES Books(BookID),
8     FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
9 );
10
11 INSERT INTO BookCategories (BookID, CategoryID)
12 VALUES
13 (1, 1),
14 (2, 2);
15 -- Sub Queries --
16 -- INNER QUERY --
17 SELECT BookID
18 FROM BookCategories
19 WHERE BookID IN (select BookID from BookCategories);
20 -- COMPARISON QUERY --
21 SELECT avg (CategoryID) FROM BookCategories;
22 SELECT BookCategories
23 FROM CategoryID > (select avg (CategoryID) from BookCategories);
24
```

Result Grid

avg (CategoryID)
1.5000

Result 2 x

Table 10:

LMS table 6 categories

```
-- 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

CategoryID
3
4

Categories 8 x

LMS table 6 categories

```
-- 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

Name
Dystopian

Categories 8 Categories 9 x

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

