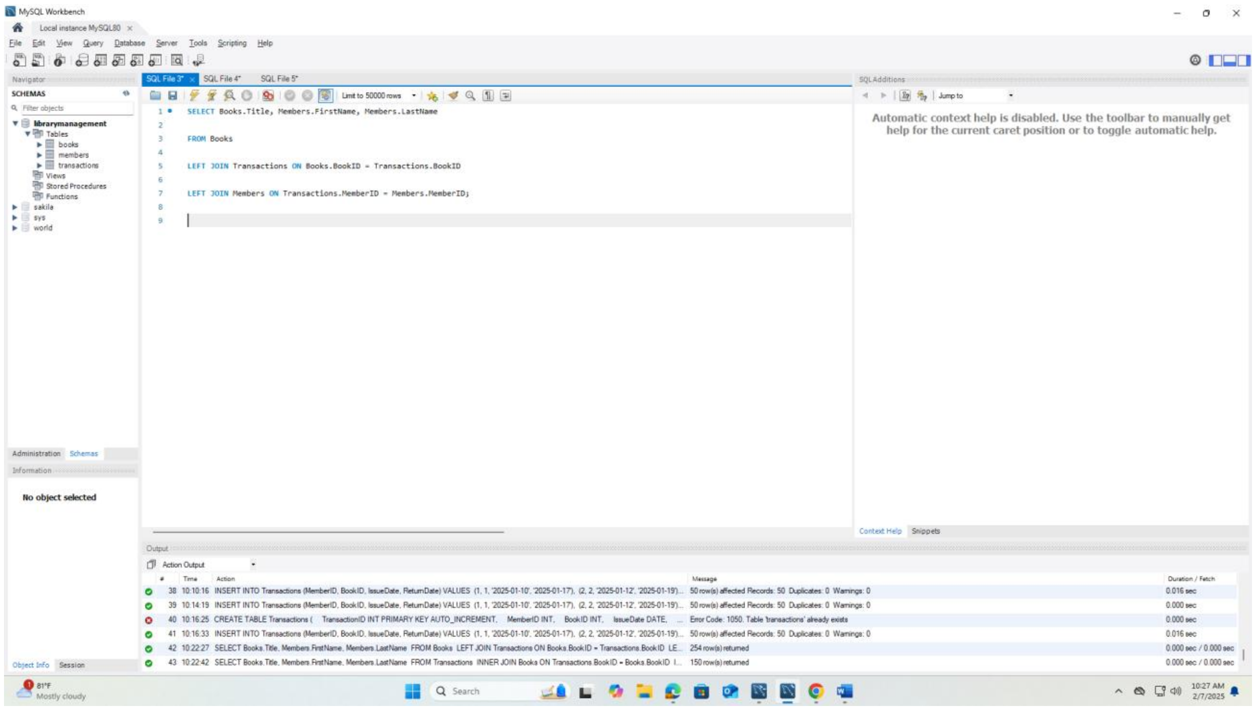
NAME: Daryl D. Silva

BSCpE – 2 B2

Laboratory Activity 1:

Laboratory Title: Installing MySQL and Setting Up the Database
Chapter No. and Topic: Chapter 1 - Relational Database Concepts
Discussions:
This activity will guide students through installing MySQL on their system and setting up a basic library management system database.

Result:
A running MySQL instance with a database called LibraryManagement.



```sql
SELECT Books.Title, Members.FirstName, Members.LastName

FROM Books

LEFT JOIN Transactions ON Books.BookID = Transactions.BookID

LEFT JOIN Members ON Transactions.MemberID = Members.MemberID;
```

```
#   Time      Action                                                                                                          Message                                                          Duration / Fetch
38  10:10:16  INSERT INTO Transactions (MemberID, BookID, IssueDate, ReturnDate) VALUES (1, 1, '2025-01-10', '2025-01-17'), (2, 2, '2025-01-12', '2025-01-19')...  50 row(s) affected Records: 50  Duplicates: 0  Warnings: 0  0.016 sec
39  10:14:19  INSERT INTO Transactions (MemberID, BookID, IssueDate, ReturnDate) VALUES (1, 1, '2025-01-10', '2025-01-17'), (2, 2, '2025-01-12', '2025-01-19')...  50 row(s) affected Records: 50  Duplicates: 0  Warnings: 0  0.000 sec
40  10:16:25  CREATE TABLE Transactions (    TransactionID INT PRIMARY KEY AUTO_INCREMENT,    MemberID INT,    BookID INT,    IssueDate DATE,   ...  Error Code: 1050. Table 'transactions' already exists  0.000 sec
41  10:16:33  INSERT INTO Transactions (MemberID, BookID, IssueDate, ReturnDate) VALUES (1, 1, '2025-01-10', '2025-01-17'), (2, 2, '2025-01-12', '2025-01-19')...  50 row(s) affected Records: 50  Duplicates: 0  Warnings: 0  0.016 sec
42  10:22:27  SELECT Books.Title, Members.FirstName, Members.LastName  FROM Books  LEFT JOIN Transactions ON Books.BookID = Transactions.BookID  LE...  254 row(s) returned  0.000 sec / 0.000 sec
43  10:22:42  SELECT Books.Title, Members.FirstName, Members.LastName  FROM Transactions  INNER JOIN Books ON Transactions.BookID = Books.BookID  I...  150 row(s) returned  0.000 sec / 0.000 sec
```

Additional Questions/Discussions:

1. **Why is MySQL popular for DBMS?**

MySQL is widely used because it is an open-source relational database management system (RDBMS) that is both free and highly efficient. It offers strong data security, fast query performance, and the ability to handle large volumes of data. Its scalability ensures it can grow with business needs, and its ease of use makes it accessible to developers. Additionally, MySQL supports a wide variety of programming languages, making it versatile for different applications.

2. **What are the advantages of using MySQL for a library management system?**

Using MySQL in a library management system simplifies data organization by automatically storing and retrieving records of books, members, and transactions. This reduces the time spent on manual tasks, such as record-keeping and searching for books, improving overall operational efficiency. The system's ease of use means that librarians can quickly adopt and manage it, while students benefit from quick access to library resources. Furthermore, MySQL ensures data integrity and security, making it a reliable choice for managing sensitive information.

Conclusions:

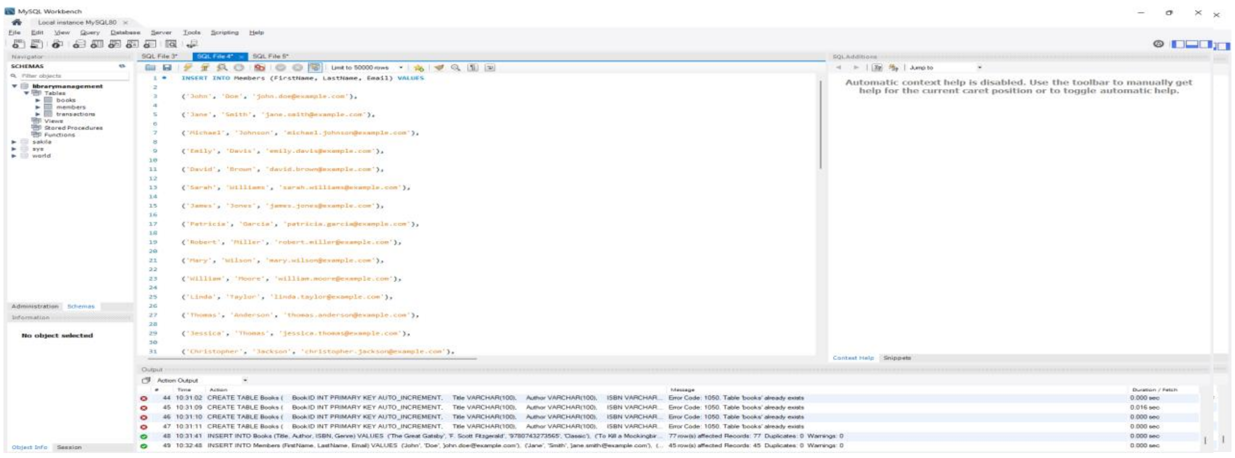
Laboratory Activity 1 focused on installing MySQL and setting up a database, providing a foundational understanding of relational database concepts. By completing this activity, students gained hands-on experience with MySQL installation and configuration, reinforcing key principles of database management. This exercise highlighted the importance of setting up a functional database environment to work with relational data effectively.

Laboratory Activity 2:

Laboratory Title: Creating Tables and Establishing Primary Keys
Chapter No. and Topic: Chapter 1 - Relational Database Concepts
Discussions:
This activity focuses on creating the main tables for the Library Management System, with primary keys for each table.

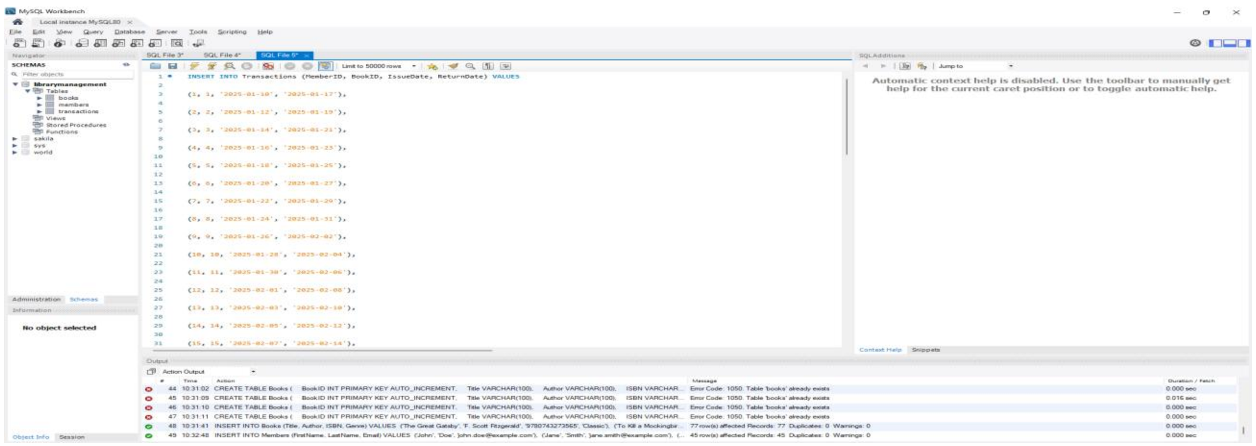
Activity Description:
Create tables such as Books, Members, and Transactions for the library system.

Result:
Three tables (Books, Members, and Transactions) are create d.



```sql
INSERT INTO Members (FirstName, LastName, Email) VALUES

('John', 'Doe', 'john.doe@example.com'),

('Jane', 'Smith', 'jane.smith@example.com'),

('Michael', 'Johnson', 'michael.johnson@example.com'),

('Emily', 'Davis', 'emily.davis@example.com'),

('David', 'Brown', 'david.brown@example.com'),

('Sarah', 'Williams', 'sarah.williams@example.com'),

('James', 'Jones', 'james.jones@example.com'),

('Patricia', 'Garcia', 'patricia.garcia@example.com'),

('Robert', 'Miller', 'robert.miller@example.com'),

('Mary', 'Wilson', 'mary.wilson@example.com'),

('William', 'Moore', 'william.moore@example.com'),

('Linda', 'Taylor', 'linda.taylor@example.com'),

('Thomas', 'Anderson', 'thomas.anderson@example.com'),

('Jessica', 'Thomas', 'jessica.thomas@example.com'),

('Christopher', 'Jackson', 'christopher.jackson@example.com'),
```

```
#   Time      Action                                                                                                                                    Message                                                          Duration / Fetch
44  10:31:02  CREATE TABLE Books (   BookID INT PRIMARY KEY AUTO_INCREMENT,   Title VARCHAR(100),   Author VARCHAR(100),   ISBN VARCHAR...              Error Code: 1050. Table 'books' already exists                   0.000 sec
45  10:31:09  CREATE TABLE Books (   BookID INT PRIMARY KEY AUTO_INCREMENT,   Title VARCHAR(100),   Author VARCHAR(100),   ISBN VARCHAR...              Error Code: 1050. Table 'books' already exists                   0.016 sec
46  10:31:10  CREATE TABLE Books (   BookID INT PRIMARY KEY AUTO_INCREMENT,   Title VARCHAR(100),   Author VARCHAR(100),   ISBN VARCHAR...              Error Code: 1050. Table 'books' already exists                   0.000 sec
47  10:31:11  CREATE TABLE Books (   BookID INT PRIMARY KEY AUTO_INCREMENT,   Title VARCHAR(100),   Author VARCHAR(100),   ISBN VARCHAR...              Error Code: 1050. Table 'books' already exists                   0.000 sec
48  10:31:41  INSERT INTO Books (Title, Author, ISBN, Genre) VALUES ('The Great Gatsby', 'F. Scott Fitzgerald', '9780743273565', 'Classic'), ('To Kill a Mockingbir...   77 row(s) affected Records: 77  Duplicates: 0  Warnings: 0   0.000 sec
49  10:32:48  INSERT INTO Members (FirstName, LastName, Email) VALUES ('John', 'Doe', 'john.doe@example.com'), ('Jane', 'Smith', 'jane.smith@example.com'), (...   45 row(s) affected Records: 45  Duplicates: 0  Warnings: 0   0.000 sec
```



```sql
INSERT INTO Transactions (MemberID, BookID, IssueDate, ReturnDate) VALUES

(1, 1, '2025-01-10', '2025-01-17'),

(2, 2, '2025-01-12', '2025-01-19'),

(3, 3, '2025-01-14', '2025-01-21'),

(4, 4, '2025-01-16', '2025-01-23'),

(5, 5, '2025-01-18', '2025-01-25'),

(6, 6, '2025-01-20', '2025-01-27'),

(7, 7, '2025-01-22', '2025-01-29'),

(8, 8, '2025-01-24', '2025-01-31'),

(9, 9, '2025-01-26', '2025-02-02'),

(10, 10, '2025-01-28', '2025-02-04'),

(11, 11, '2025-01-30', '2025-02-06'),

(12, 12, '2025-02-01', '2025-02-08'),

(13, 13, '2025-02-03', '2025-02-10'),

(14, 14, '2025-02-05', '2025-02-12'),

(15, 15, '2025-02-07', '2025-02-14'),
```

```
#   Time      Action                                                                                                                                    Message                                                          Duration / Fetch
44  10:31:02  CREATE TABLE Books (   BookID INT PRIMARY KEY AUTO_INCREMENT,   Title VARCHAR(100),   Author VARCHAR(100),   ISBN VARCHAR...              Error Code: 1050. Table 'books' already exists                   0.000 sec
45  10:31:09  CREATE TABLE Books (   BookID INT PRIMARY KEY AUTO_INCREMENT,   Title VARCHAR(100),   Author VARCHAR(100),   ISBN VARCHAR...              Error Code: 1050. Table 'books' already exists                   0.016 sec
46  10:31:10  CREATE TABLE Books (   BookID INT PRIMARY KEY AUTO_INCREMENT,   Title VARCHAR(100),   Author VARCHAR(100),   ISBN VARCHAR...              Error Code: 1050. Table 'books' already exists                   0.000 sec
47  10:31:11  CREATE TABLE Books (   BookID INT PRIMARY KEY AUTO_INCREMENT,   Title VARCHAR(100),   Author VARCHAR(100),   ISBN VARCHAR...              Error Code: 1050. Table 'books' already exists                   0.000 sec
48  10:31:41  INSERT INTO Books (Title, Author, ISBN, Genre) VALUES ('The Great Gatsby', 'F. Scott Fitzgerald', '9780743273565', 'Classic'), ('To Kill a Mockingbir...   77 row(s) affected Records: 77  Duplicates: 0  Warnings: 0   0.000 sec
49  10:32:48  INSERT INTO Members (FirstName, LastName, Email) VALUES ('John', 'Doe', 'john.doe@example.com'), ('Jane', 'Smith', 'jane.smith@example.com'), (...   45 row(s) affected Records: 45  Duplicates: 0  Warnings: 0   0.000 sec
```

Additional Questions/Discussions:

1. What is the importance of primary keys in a relational database?

   The primary key in a relational database is essential because it uniquely identifies each record in a table. This ensures that no two rows have the same identifier, preventing duplication and maintaining data integrity. Primary keys also facilitate efficient data retrieval and enable the establishment of relationships between different tables, which is crucial for organizing and managing complex data structures.

2. How do foreign keys maintain referential integrity?

Foreign keys maintain referential integrity by ensuring that relationships between tables remain consistent. They link a record in one table (the foreign key) to a corresponding record in another table (the primary key). This prevents actions that would create "orphaned" records, such as deleting or updating a primary key record that has associated foreign key references. As a result, foreign keys help ensure that the data remains accurate and consistent across related tables.

Conclusions:

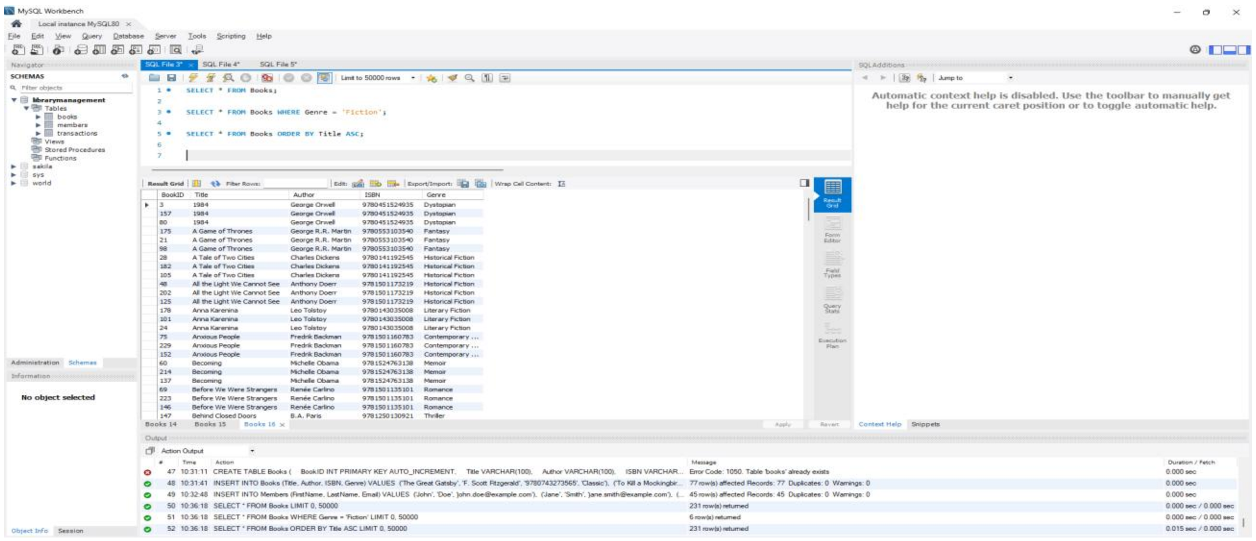
Laboratory Activity 2 focused on creating tables and establishing primary keys, reinforcing key concepts in relational database management. By completing this activity, students learned how to structure data effectively using tables and assign primary keys to ensure each record is uniquely identifiable. This hands-on experience emphasized the importance of primary keys in maintaining data integrity and laid the foundation for building well-organized and efficient relational databases..

Laboratory Activity 3:

Laboratory Title: Structured Query Language (SQL) - Basic Queries
Chapter No. and Topic: Chapter 2 - Structured Query Language (SQL)
Discussions:
This activity covers the basics of querying data from a table using SQL.

Activity Description:
Learn how to retrieve data using SELECT, filter with WHERE clauses, and sort results using ORDER BY.

Result:
Basic queries to retrieve and filter data from the Books table.



```sql
SELECT * FROM Books;

SELECT * FROM Books WHERE Genre = 'Fiction';

SELECT * FROM Books ORDER BY Title ASC;
```

BookID	Title	Author	ISBN	Genre
3	1984	George Orwell	9780451524935	Dystopian
157	1984	George Orwell	9780451524935	Dystopian
80	1984	George Orwell	9780451524935	Dystopian
175	A Game of Thrones	George R.R. Martin	9780553103540	Fantasy
21	A Game of Thrones	George R.R. Martin	9780553103540	Fantasy
98	A Game of Thrones	George R.R. Martin	9780553103540	Fantasy
28	A Tale of Two Cities	Charles Dickens	9780141192545	Historical Fiction
182	A Tale of Two Cities	Charles Dickens	9780141192545	Historical Fiction
105	A Tale of Two Cities	Charles Dickens	9780141192545	Historical Fiction
48	All the Light We Cannot See	Anthony Doerr	9781501173219	Historical Fiction
202	All the Light We Cannot See	Anthony Doerr	9781501173219	Historical Fiction
125	All the Light We Cannot See	Anthony Doerr	9781501173219	Historical Fiction
178	Anna Karenina	Leo Tolstoy	9780143035008	Literary Fiction
101	Anna Karenina	Leo Tolstoy	9780143035008	Literary Fiction
24	Anna Karenina	Leo Tolstoy	9780143035008	Literary Fiction
75	Anxious People	Fredrik Backman	9781501160783	Contemporary ...
229	Anxious People	Fredrik Backman	9781501160783	Contemporary ...
152	Anxious People	Fredrik Backman	9781501160783	Contemporary ...
60	Becoming	Michelle Obama	9781524763138	Memoir
214	Becoming	Michelle Obama	9781524763138	Memoir
137	Becoming	Michelle Obama	9781524763138	Memoir
69	Before We Were Strangers	Renée Carlino	9781501135101	Romance
223	Before We Were Strangers	Renée Carlino	9781501135101	Romance
146	Before We Were Strangers	Renée Carlino	9781501135101	Romance
147	Behind Closed Doors	B.A. Paris	9781250130921	Thriller

```
#   Time      Action                                                                                                   Message                                                      Duration / Fetch
47  10:31:11  CREATE TABLE Books (    BookID INT PRIMARY KEY AUTO_INCREMENT,    Title VARCHAR(100),    Author VARCHAR(100),    ISBN VARCHAR...   Error Code: 1050. Table 'books' already exists   0.000 sec
48  10:31:41  INSERT INTO Books (Title, Author, ISBN, Genre) VALUES ('The Great Gatsby', 'F. Scott Fitzgerald', '9780743273565', 'Classic'), ('To Kill a Mockingbir...   77 row(s) affected Records: 77  Duplicates: 0  Warnings: 0   0.000 sec
49  10:32:48  INSERT INTO Members (FirstName, LastName, Email) VALUES ('John', 'Doe', 'john.doe@example.com'), ('Jane', 'Smith', 'jane.smith@example.com'), (...   45 row(s) affected Records: 45  Duplicates: 0  Warnings: 0   0.000 sec
50  10:36:18  SELECT * FROM Books LIMIT 0, 50000                                                                       231 row(s) returned                                          0.000 sec / 0.000 sec
51  10:36:18  SELECT * FROM Books WHERE Genre = 'Fiction' LIMIT 0, 50000                                               6 row(s) returned                                            0.000 sec / 0.000 sec
52  10:36:18  SELECT * FROM Books ORDER BY Title ASC LIMIT 0, 50000                                                    231 row(s) returned                                          0.015 sec / 0.000 sec
```

Additional Questions/Discussions:

The WHERE clause improves SQL queries by filtering records based on specified conditions, allowing users to retrieve only relevant data. This enhances query efficiency and precision by excluding unnecessary information. The ORDER BY clause further improves functionality by sorting the query results in ascending or descending order, helping users to view data in a structured and organized manner. Together, they make SQL queries more powerful and tailored to specific needs.

Conclusions:

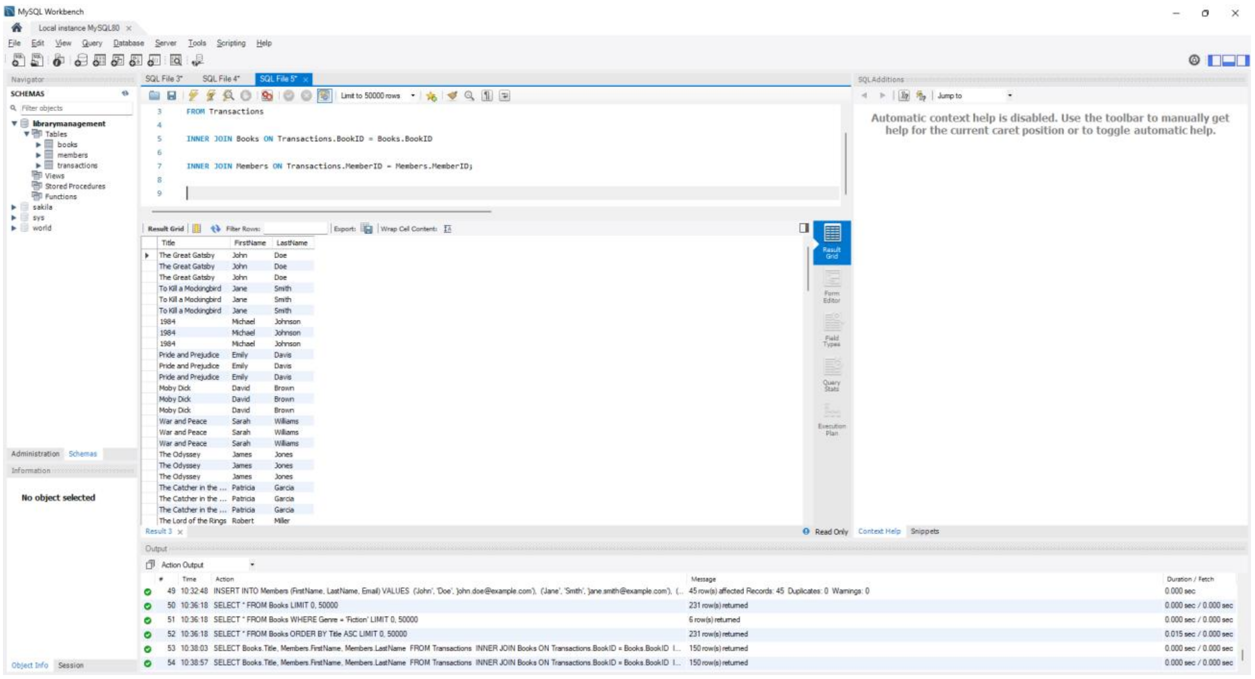
Laboratory Activity 3 focused on using Structured Query Language (SQL) to perform basic queries, providing a solid foundation for interacting with relational databases. Students gained practical experience with SQL commands such as SELECT, WHERE, and ORDER BY, which are essential for filtering, sorting, and retrieving data. This activity helped reinforce the concepts of data manipulation and query optimization, essential skills for efficiently managing and analyzing data in real-world database systems.

Laboratory Activity 4:

Laboratory Title: SQL - JOIN Operation
Chapter No. and Topic: Chapter 2 - Structured Query Language (SQL)
Discussions:
This activity introduces students to SQL JOIN operations for combining data from multiple tables.

Activity Description:
Learn how to use INNER JOIN, LEFT JOIN, and RIGHT JOIN to combine tables.

Result:
JOIN operations linking tables to retrieve combined data.



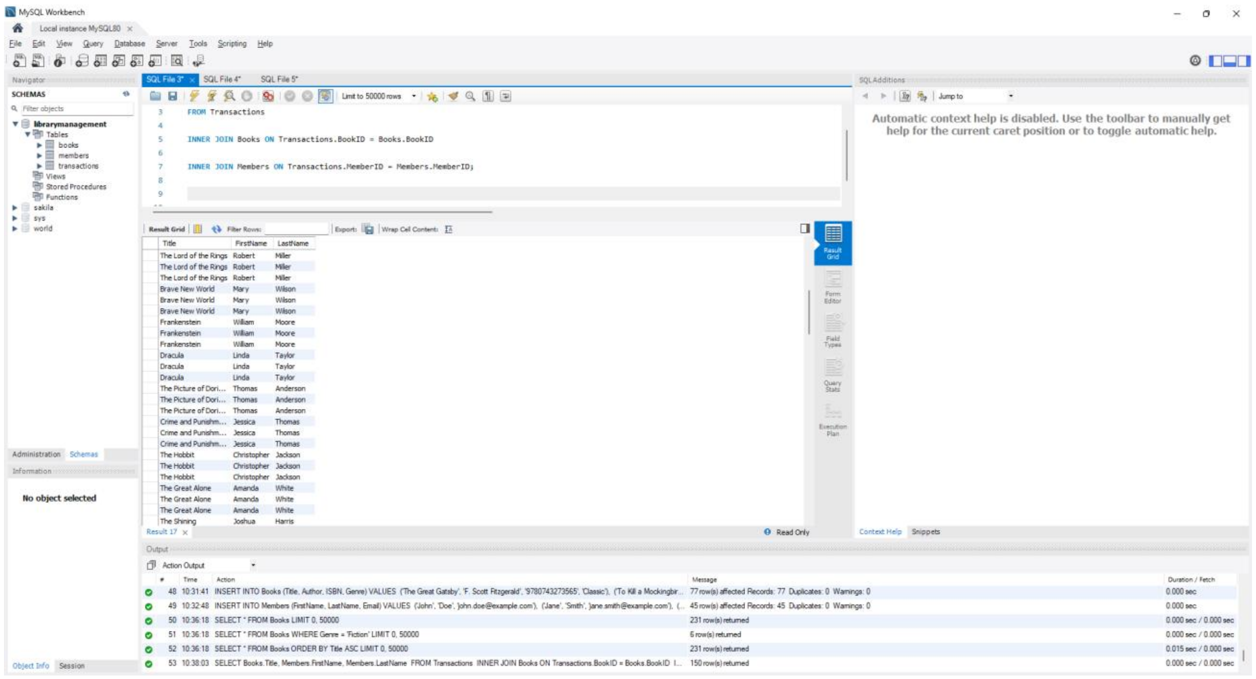
```sql
FROM Transactions

INNER JOIN Books ON Transactions.BookID = Books.BookID

INNER JOIN Members ON Transactions.MemberID = Members.MemberID;
```

Title	FirstName	LastName
The Great Gatsby	John	Doe
The Great Gatsby	John	Doe
The Great Gatsby	John	Doe
To Kill a Mockingbird	Jane	Smith
To Kill a Mockingbird	Jane	Smith
To Kill a Mockingbird	Jane	Smith
1984	Michael	Johnson
1984	Michael	Johnson
1984	Michael	Johnson
Pride and Prejudice	Emily	Davis
Pride and Prejudice	Emily	Davis
Pride and Prejudice	Emily	Davis
Moby Dick	David	Brown
Moby Dick	David	Brown
Moby Dick	David	Brown
War and Peace	Sarah	Williams
War and Peace	Sarah	Williams
War and Peace	Sarah	Williams
The Odyssey	James	Jones
The Odyssey	James	Jones
The Odyssey	James	Jones
The Catcher in the ...	Patricia	Garcia
The Catcher in the ...	Patricia	Garcia
The Catcher in the ...	Patricia	Garcia
The Lord of the Rings	Robert	Miller

Result 3

Output
Action Output

#	Time	Action	Message	Duration / Fetch
49	10:32:48	INSERT INTO Members (FirstName, LastName, Email) VALUES ('John', 'Doe', 'john.doe@example.com'), ('Jane', 'Smith', 'jane.smith@example.com'), (...	45 row(s) affected Records: 45 Duplicates: 0 Warnings: 0	0.000 sec
50	10:36:18	SELECT * FROM Books LIMIT 0, 50000	231 row(s) returned	0.000 sec / 0.000 sec
51	10:36:18	SELECT * FROM Books WHERE Genre = 'Fiction' LIMIT 0, 50000	6 row(s) returned	0.000 sec / 0.000 sec
52	10:36:18	SELECT * FROM Books ORDER BY Title ASC LIMIT 0, 50000	231 row(s) returned	0.015 sec / 0.000 sec
53	10:38:03	SELECT Books.Title, Members.FirstName, Members.LastName FROM Transactions INNER JOIN Books ON Transactions.BookID = Books.BookID I...	150 row(s) returned	0.000 sec / 0.000 sec
54	10:38:57	SELECT Books.Title, Members.FirstName, Members.LastName FROM Transactions INNER JOIN Books ON Transactions.BookID = Books.BookID I...	150 row(s) returned	0.000 sec / 0.000 sec



```sql
FROM Transactions

INNER JOIN Books ON Transactions.BookID = Books.BookID

INNER JOIN Members ON Transactions.MemberID = Members.MemberID;
```

Title	FirstName	LastName
The Lord of the Rings	Robert	Miller
The Lord of the Rings	Robert	Miller
The Lord of the Rings	Robert	Miller
Brave New World	Mary	Wilson
Brave New World	Mary	Wilson
Brave New World	Mary	Wilson
Frankenstein	William	Moore
Frankenstein	William	Moore
Frankenstein	William	Moore
Dracula	Linda	Taylor
Dracula	Linda	Taylor
Dracula	Linda	Taylor
The Picture of Dori...	Thomas	Anderson
The Picture of Dori...	Thomas	Anderson
The Picture of Dori...	Thomas	Anderson
Crime and Punishm...	Jessica	Thomas
Crime and Punishm...	Jessica	Thomas
Crime and Punishm...	Jessica	Thomas
The Hobbit	Christopher	Jackson
The Hobbit	Christopher	Jackson
The Hobbit	Christopher	Jackson
The Great Alone	Amanda	White
The Great Alone	Amanda	White
The Great Alone	Amanda	White
The Shining	Joshua	Harris

```
#   Time      Action                                                                                                                                  Message                                                            Duration / Fetch
48  10:31:41  INSERT INTO Books (Title, Author, ISBN, Genre) VALUES ('The Great Gatsby', 'F. Scott Fitzgerald', '9780743273565', 'Classic'), ('To Kill a Mockingbir...  77 row(s) affected Records: 77 Duplicates: 0 Warnings: 0   0.000 sec
49  10:32:48  INSERT INTO Members (FirstName, LastName, Email) VALUES ('John', 'Doe', 'john.doe@example.com'), ('Jane', 'Smith', 'jane.smith@example.com'), (...  45 row(s) affected Records: 45 Duplicates: 0 Warnings: 0   0.000 sec
50  10:36:18  SELECT * FROM Books LIMIT 0, 50000                                                                                                      231 row(s) returned                                                0.000 sec / 0.000 sec
51  10:36:18  SELECT * FROM Books WHERE Genre = 'Fiction' LIMIT 0, 50000                                                                              6 row(s) returned                                                  0.000 sec / 0.000 sec
52  10:36:18  SELECT * FROM Books ORDER BY Title ASC LIMIT 0, 50000                                                                                   231 row(s) returned                                                0.015 sec / 0.000 sec
53  10:38:03  SELECT Books.Title, Members.FirstName, Members.LastName FROM Transactions INNER JOIN Books ON Transactions.BookID = Books.BookID I...    150 row(s) returned                                                0.000 sec / 0.000 sec
```

Additional Questions/Discussions:

- How does the LEFT JOIN differ from the INNER JOIN?

The INNER JOIN retrieves only the records that have matching values in both tables being joined. If there’s no match, the row is excluded from the result. In contrast, the LEFT JOIN (or LEFT OUTER JOIN) returns all records from the left table and the matching records from the right table. If there’s no match, the result will include NULL values for columns from the right table, ensuring that no data from the left table is omitted.

Conclusions:

Laboratory Activity 4 focused on SQL JOIN operations, providing hands-on experience with combining data from multiple tables. Students learned how to use different types of JOINs, including INNER JOIN and LEFT JOIN, to retrieve related data based on matching keys. This activity reinforced the importance of relational database concepts and demonstrated how to effectively extract and manipulate data across multiple tables, a crucial skill for efficient data analysis and reporting in real-world scenarios.