Project Report

Big Data Management Analytics

# SQL - LIBRARY MANAGEMENT SYSTEM

Submitted to - Prof. Amarnath Mitra

**FORE School of Management, New Delhi**


**Submitted by -**

Shreeya Yashvi 055045

Kashish Srivastava 055046

# Table of Contents

## PROBLEM STATEMENT

The Library Management System (LMS) is a critical tool for managing library operations, including book tracking, member management, book issuance, reservations, fines, and reviews. However, managing such a system manually or with inefficient database structures can lead to data redundancy, inconsistency, and poor performance. The primary challenge is to design and implement a robust, scalable, and efficient database system that ensures data integrity, supports complex queries, and handles large volumes of data without compromising performance. Additionally, the system must enforce referential integrity through proper use of primary and foreign keys, and it must be normalized to avoid data anomalies.

## OBJECTIVES

1. To design and create a relational database for the Library Management System that supports all necessary functionalities, including book management, member management, branch management, transactions, fines, and reviews.
2. To develop SQL queries for efficient data retrieval, manipulation, and analysis, including SELECT, INSERT, UPDATE, and DELETE operations, as well as complex queries involving joins, subqueries, and aggregate functions.
3. To ensure the database adheres to normalization principles (1NF, 2NF) to eliminate data redundancy and anomalies, and to enforce data integrity through primary keys, foreign keys, and constraints.
4. To conduct stress tests on the database to evaluate its performance under various operations (INSERT, UPDATE, DELETE) and to ensure that cascading effects are properly handled, maintaining referential integrity between parent and child tables.
5. To provide insights into the database's performance, scalability, and integrity, and to offer recommendations for further improvements and optimizations.

## INTRODUCTION

This project involves the creation of a Library Management System using SQL. The system is designed to manage library operations such as book tracking, member management, book issuance, reservations, fines, and reviews. The database is structured to ensure data integrity, scalability, and efficient querying.

The database structure follows the relational database model using primary keys, foreign keys, constraints, and normalization techniques

The report covers:

- Database structure and schema
- Relationships between tables (Cardinality and Logical Constraints)
- SQL queries for data retrieval and manipulation
- Use of functions, logical operators, and arithmetic operations

## DATABASE OVERVIEW

The Library Management System database is designed to handle the following key functionalities:

- Book Management: Track books, authors, publishers, and categories.
- Member Management: Manage library members and their details.
- Branch Management: Handle multiple library branches and their staff.
- Transactions: Manage book issuance, returns, and reservations.
- Fines and Reviews: Track fines for overdue books and allow members to review books.

## DATABASE SCHEMA

- **Tables** - stores different types of data.
- **Columns** - different tables, specifying attributes and constraints.
- **Primary Keys (PK)** ensures each record is uniquely identifiable.
- **Foreign Keys (FK)** establishing relationships between tables.
- **Constraints** (e.g., NOT NULL, UNIQUE, DEFAULT) to maintain data integrity.

## DATABASE CREATION AND DESIGN

The database, `LibraryManagement`, has been created to store essential information about books, customers, branches, employees, book issuance, and returns.

Use database command (put that command/query as per shreeya's suggestion)

| Table Name | Description |
|---|---|
| Books | Stores book details (ISBN, title, category, rental price, author, publisher) |
| Customer | Holds customer details (ID, name, address, registration date) |
| Branch | Stores branch details (branch number, manager ID, address, contact) |
| Employee | Stores employee information (employee ID, name, position, salary, branch association) |
| Issue_Status | Records issued books (issue ID, issued customer, book name, issue date, ISBN) |
| Return_Status | Records returned books (return ID, returning customer, book name, return date, ISBN) |

Each table includes **primary keys** and **foreign keys** to establish relationships between different entities.

## TABLES AND THEIR RELATIONSHIPS

| Table Name | Columns | Primary Key | Foreign Keys (FKs) & Relationships |
|---|---|---|---|
| Authors | AuthorID, FirstName, LastName | AuthorID | - |
| Publishers | PublisherID, PublisherName, Address, Phone | PublisherID | - |

| | | | |
|---|---|---|---|
| Categories | CategoryID, CategoryName | CategoryID | - |
| Books | BookID, Title, ISBN, PublicationYear, PublisherID, CategoryID | BookID | FK: PublisherID → Publishers, CategoryID → Categories |
| BookAuthors | BookID, AuthorID | BookID, AuthorID | FK: BookID → Books, AuthorID → Authors |
| LibraryBranches | BranchID, BranchName, Address, Phone | BranchID | - |
| BookCopies | CopyID, BookID, BranchID, Status | CopyID | FK: BookID → Books, BranchID → LibraryBranches |
| Members | MemberID, FirstName, LastName, Email, Phone, Address, MembershipDate | MemberID | - |
| Issue | IssueID, CopyID, MemberID, LoanDate, DueDate, ReturnDate | IssueID | FK: CopyID → BookCopies, MemberID → Members |
| Reservations | ReservationID, CopyID, MemberID, ReservationDate, Status | ReservationID | FK: CopyID → BookCopies, MemberID → Members |
| Fines | FineID, MemberID, IssueID, Amount, DateIssued, DatePaid | FineID | FK: MemberID → Members, IssueID → Issue |
| Librarians | LibrarianID, FirstName, LastName, Email, Phone, BranchID | LibrarianID | FK: BranchID → LibraryBranches |
| BookReviews | ReviewID, BookID, MemberID, Rating, Comment, ReviewDate | ReviewID | FK: BookID → Books, MemberID → Members |

**RELATIONSHIP AND CARDINALITY**

| Tables Involved | Types | Cardinality | Logical Constraints |
|---|---|---|---|
| Books - Categories | **Many-to-One** (Many Books belong to one Category, but one Category can have multiple Books) | • 1 Category → Many Books<br>• 1 Book → 1 Category | A book must belong to a category (mandatory).<br><br>A category can exist without books (optional). |
| Books - BookCopies | **One-to-Many** (One Book can have multiple Copies, but each Copy belongs to one Book only) | • 1 Book → Many Copies<br>• 1 Copy → 1 Book | A book must have at least one copy (mandatory).<br><br>A copy cannot exist without a book (mandatory). |
| Books - Publishers | **Many-to-One** (Many Books can be published by one Publisher, but one Publisher can publish multiple Books) | • 1 Publisher → Many Books<br>• 1 Book → 1 Publisher | A book must have a publisher (mandatory).<br><br>A publisher can exist without books (optional). |
| Members - Reservations | **One-to-Many** (One Member can make multiple Reservations, but each Reservation belongs to one Member only) | • 1 Member → Many Reservations<br>• 1 Reservation → 1 Member | A reservation must be linked to a member (mandatory).<br><br>A member may not have any reservations (optional). |
| BookCopies - Reservations | **One-to-Many** (One Book Copy can be reserved multiple | • 1 Copy → Many Reservations<br>• 1 Reservation | A reservation must be linked to a book copy (mandatory). |

| | | | |
|---|---|---|---|
| | times, but each Reservation is for one Book Copy only) | ➔ 1 Copy | A book copy may not have any reservations (optional). |
| Reservations - Issue | **One-to-One** (A Reservation results in one Issue if the book is borrowed, otherwise, it remains unissued) | • 1 Reservation ➔ At most 1 Issue<br>• 1 Issue ➔ 1 Reservation | A reservation may or may not lead to an issue (optional).<br><br>A book cannot be issued without being reserved first (mandatory). |
| Issue - Fines | **One-to-One** (A Fine is issued for a specific Issue, and one Issue can result in at most one Fine) | • 1 Issue ➔ At most 1 Fine<br>• 1 Fine ➔ 1 Issue | A fine must be linked to a book issue (mandatory).<br><br>A book issue may not always result in a fine (optional). |
| Members - Issue | **One-to-Many** (One Member can borrow multiple books, but each Issue belongs to only one Member) | • 1 Member ➔ Many Issues<br>• 1 Issue ➔ 1 Member | A book issue must be linked to a member (mandatory).<br><br>A member may not have any issued books (optional). |
| LibraryBranch - BookCopies | **One-to-Many** (One Library Branch can have multiple Book Copies, but each Copy belongs to only one Branch) | • 1 Branch ➔ Many Copies<br>• 1 Copy ➔ 1 Branch | A book copy must belong to a branch (mandatory).<br><br>A branch may have no books yet (optional). |
| Librarians - LibraryBranch | **Many-to-One** (Many Librarians work in one Branch, but one Branch can have | • 1 Branch ➔ Many Librarians<br>• 1 Librarian ➔ 1 | A librarian must be assigned to a branch (mandatory). |

| | | | |
|---|---|---|---|
| | multiple Librarians) | Branch | A branch can exist without a librarian (optional). |
| Books - Authors | **Many-to-Many** (A Book can have multiple Authors, and an Author can write multiple Books) | • 1 Book → Many Authors<br>• 1 Author → Many Books | A book must have at least one author (mandatory).<br><br>An author may not have any books yet (optional). |

**SQL QUERIES FOR DATA RETRIEVAL AND MANIPULATION**

Following are the types of queries used for data retrieval and manipulation -

1. Data Retrieval Queries (SELECT Queries)

2. Data Manipulation Queries (INSERT, UPDATE, DELETE)

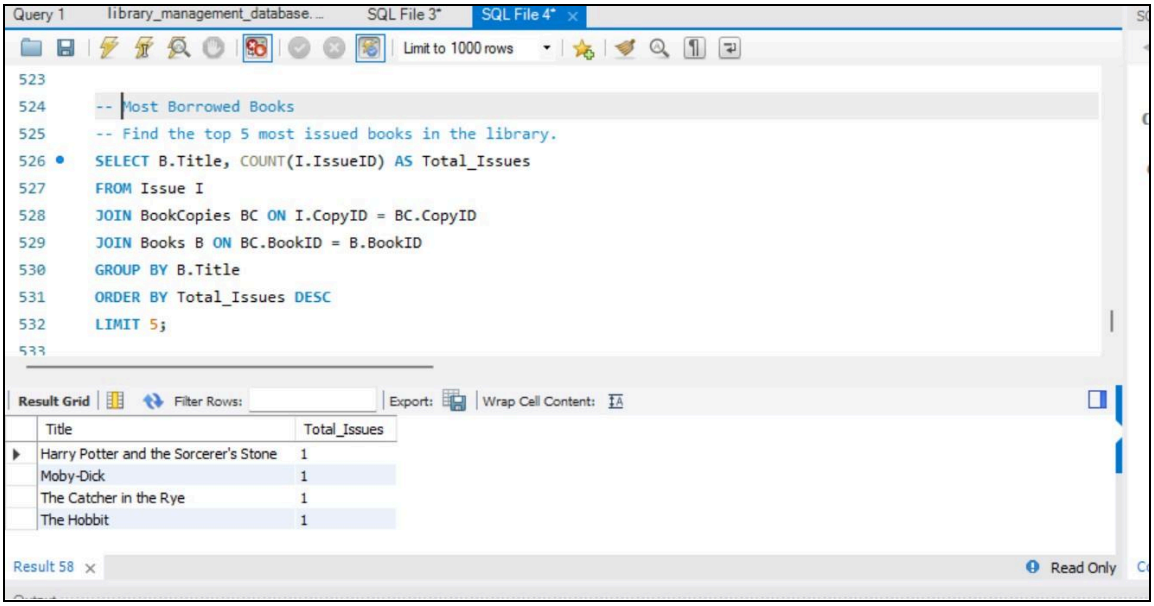
3. Aggregate and Analytical Queries

4. Joins and Nested Queries

## CODES AND OUTPUTS FROM THE QUERY RUN

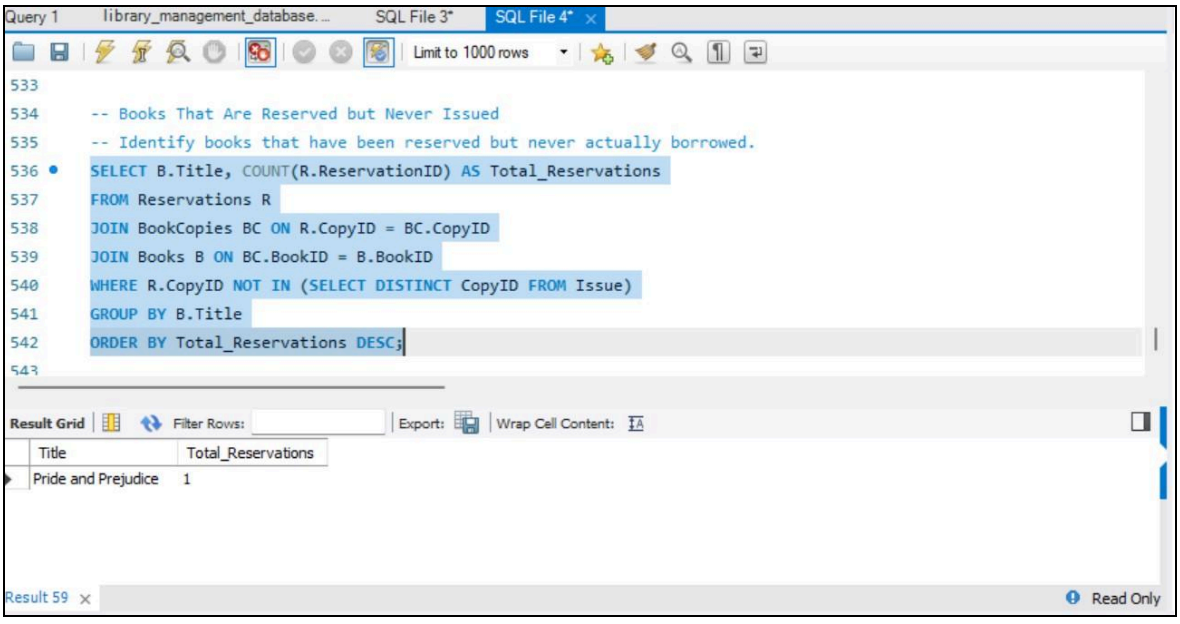
### a) Finding most issued books

- Sorts books by the number of issues in descending order and retrieves the top 5 most issued books.
- Result: Harry Potter, Moby-Dick, The catcher in the Rye, The Hobbit are the top most issued books.



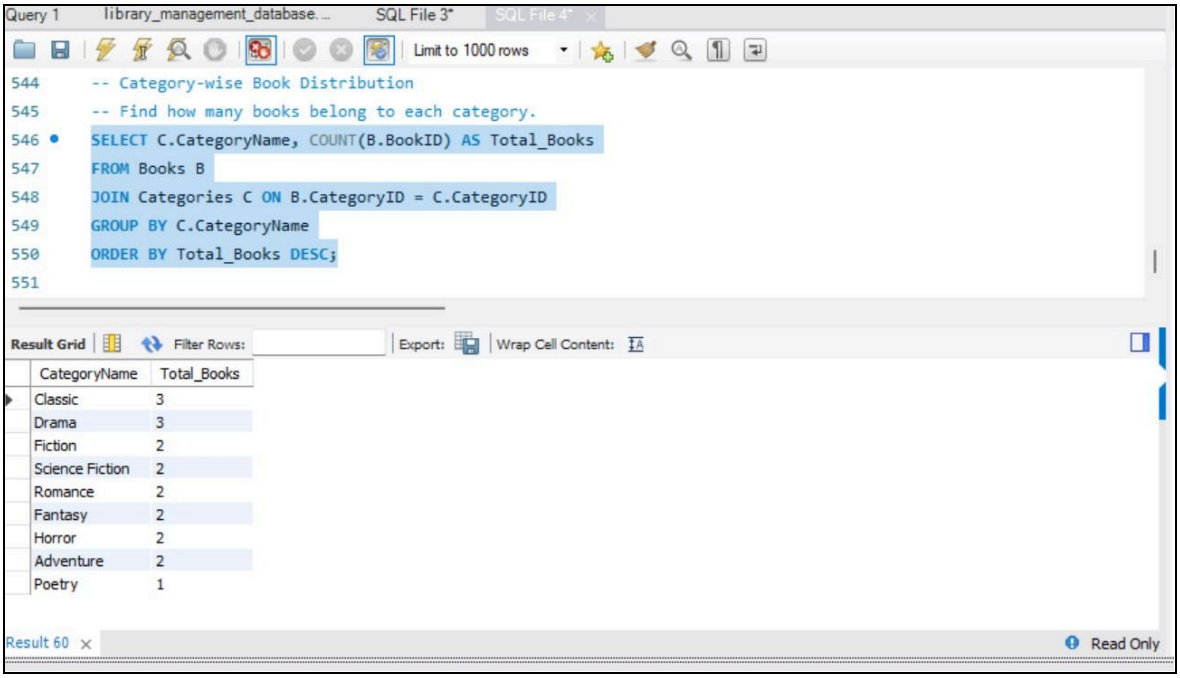

### b) Finding Books Reserved but Never Issued

- Finds books that were reserved but never actually issued.
- Result: Pride and Prejudice had one reservation but was never issued.



**c) Category-wise Book Distribution**

- Displays the number of books in each category.
- Categories with the most books: Classic (3), Drama (3), Fiction (2).




## 1NF

1NF test has been conducted on the Library Management Database to verify compliance with 1NF. The test includes:

- **Test of Atomicity/Multivariate**: Ensuring all columns contain atomic values.
- **Test of Unique Rows (Primary Key)**: Ensuring each table has a primary key.
- **Checking for Repeating Groups**: Ensuring no repeating groups exist in the tables.

**Methodology**

The following steps were taken to test for 1NF:

1. Identification of  Multi-Valued Attributes: Checked for columns that may contain multiple values (e.g., comma-separated values).
2. Checking for Primary Keys: Verified that each table has a primary key.

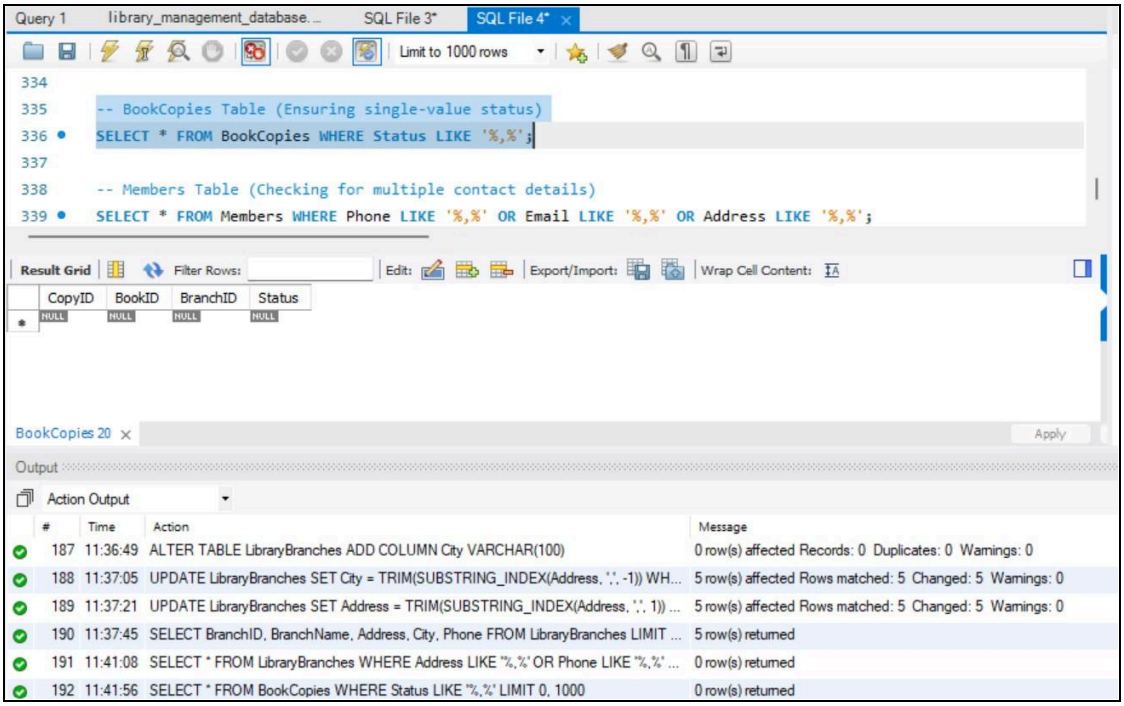
3. Checking for Repeating Groups: Ensured no repeating groups exist in the tables.
4. Run Queries: Executed SQL queries to detect multi-valued attributes, missing primary keys, and repeating groups.
5. Modifying Tables: If issues were found, modified the tables to ensure compliance with 1NF.
6. Verified Results: Re-run queries to confirm that the tables now comply with 1NF.

## QUERIES FOR 1NF TEST

### 1. Condition 1 - Test of Atomicity/ Multivariate

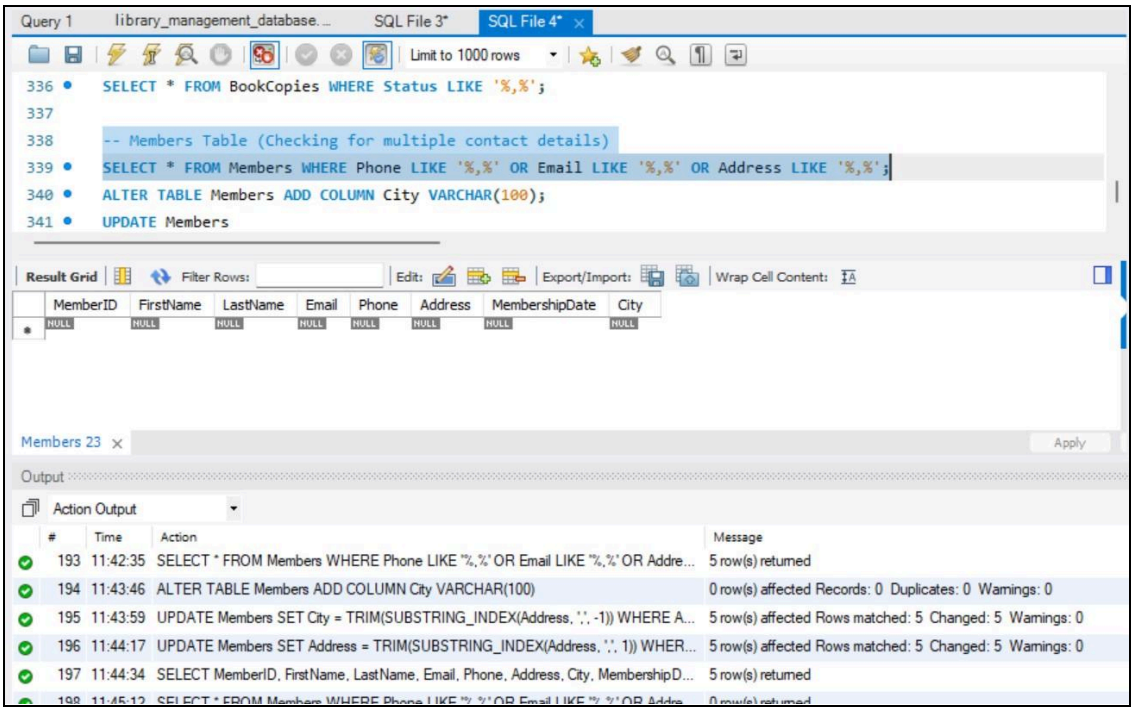
#### 1.1 BookCopies Table

- Query: Checked for multi-valued Status column.
- Result: No rows returned, indicating that the Status column contains atomic values.



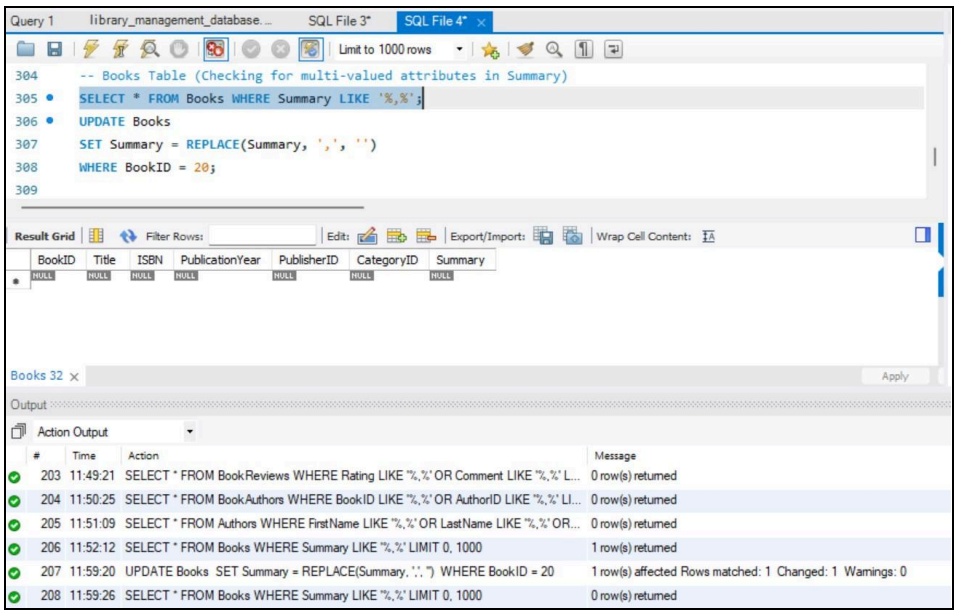
### 1.2 Members Table

- Query: Check for multi-valued Phone, Email, or Address columns.
- Result: No rows returned, confirming that these columns contain atomic values.



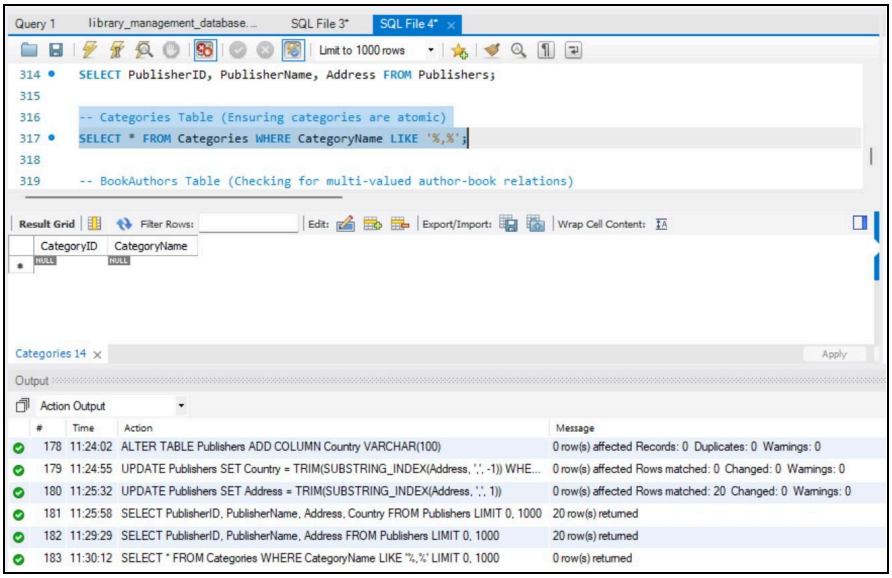

### 1.3 Books Table

- Query: Check for a multi-valued Summary column.
- Result: No rows returned, confirming that the Summary column contains atomic values.



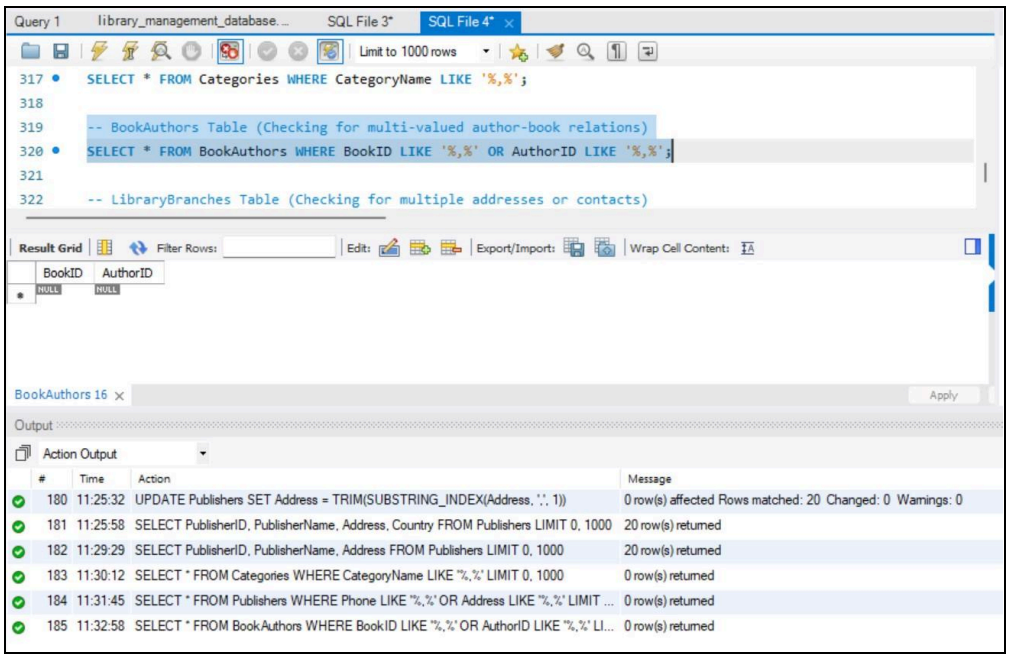
### 1.4 Categories Table

- Query: Check for multi-valued CategoryName columns.
- Result: No rows returned, indicating that the CategoryName column contains atomic values.



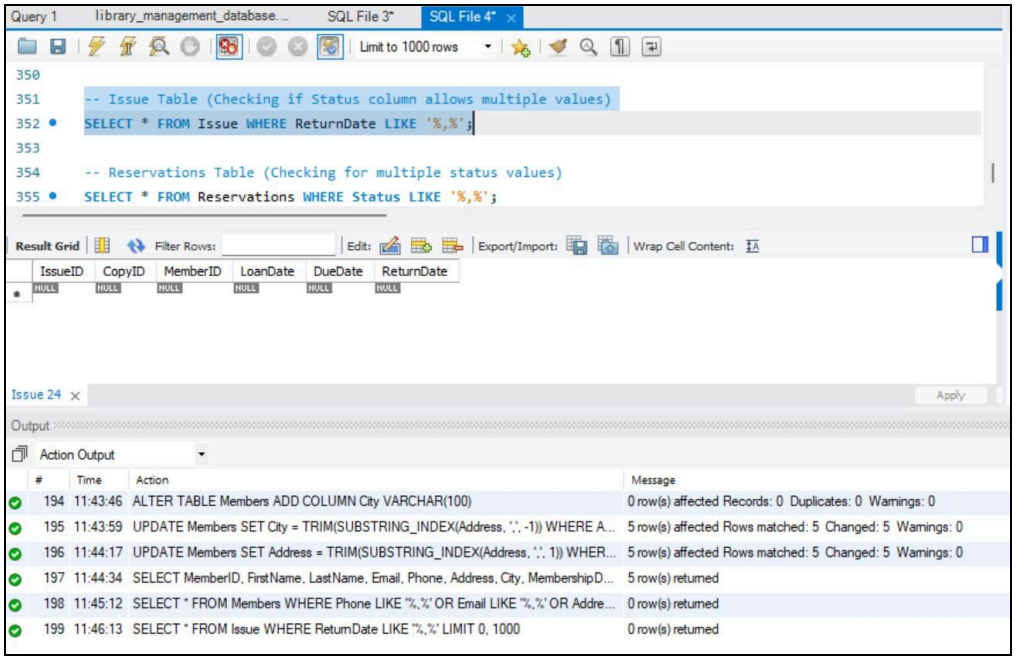

### 1.5 BookAuthors Table

- Query: Check for multi-valued BookID or AuthorID columns.
- Result: No rows returned, confirming that these columns contain atomic values.



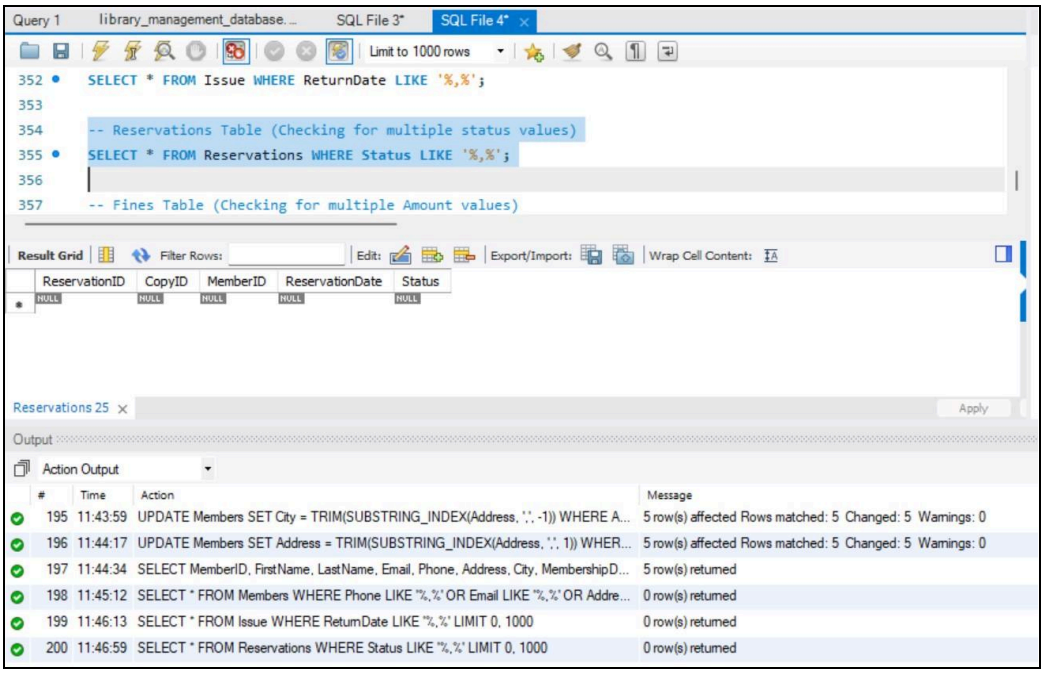
### 1.6 Issue Table

- Query: Check for multi-valued ReturnDate columns.
- Result: No rows returned, indicating that the ReturnDate column contains atomic values.



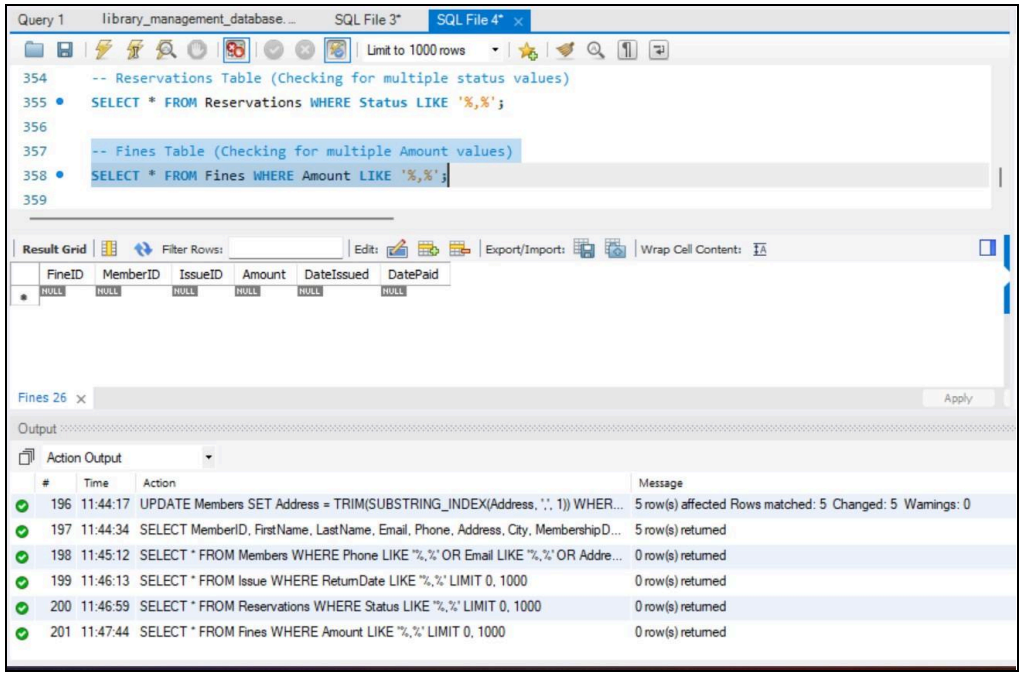

### 1.7 Reservations Table

- Query: Check for multi-valued Status columns.
- Result: No rows returned, confirming that the Status column contains atomic values.



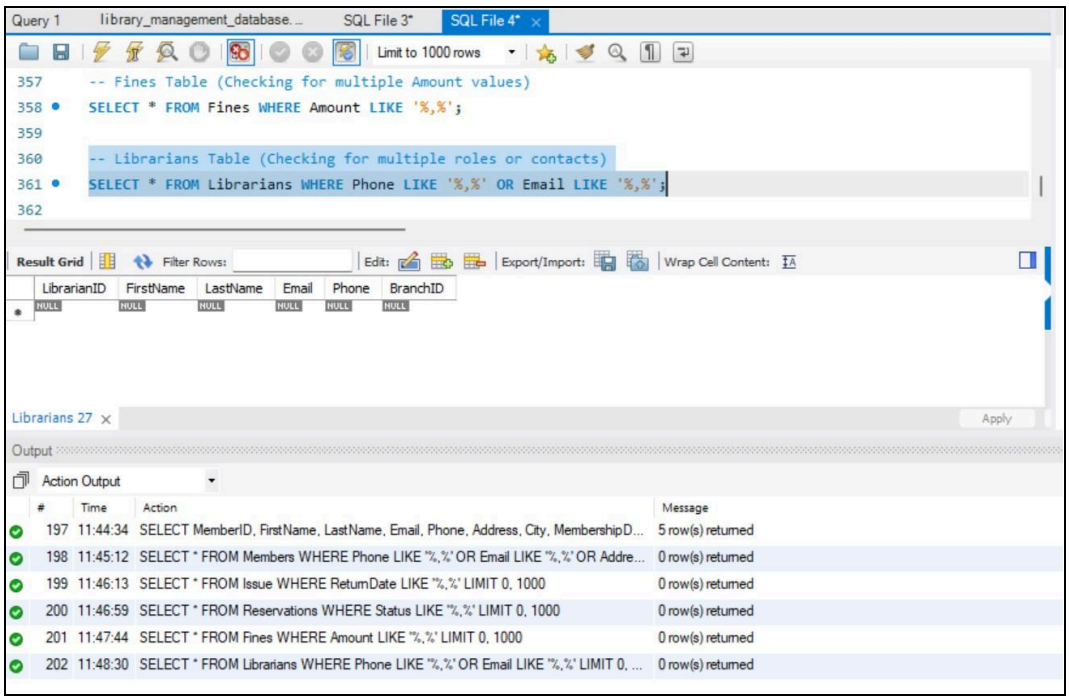
### 1.8 Fines Table

- Query: Check for multi-valued Amount columns.
- Result: No rows returned, indicating that the Amount column contains atomic values.



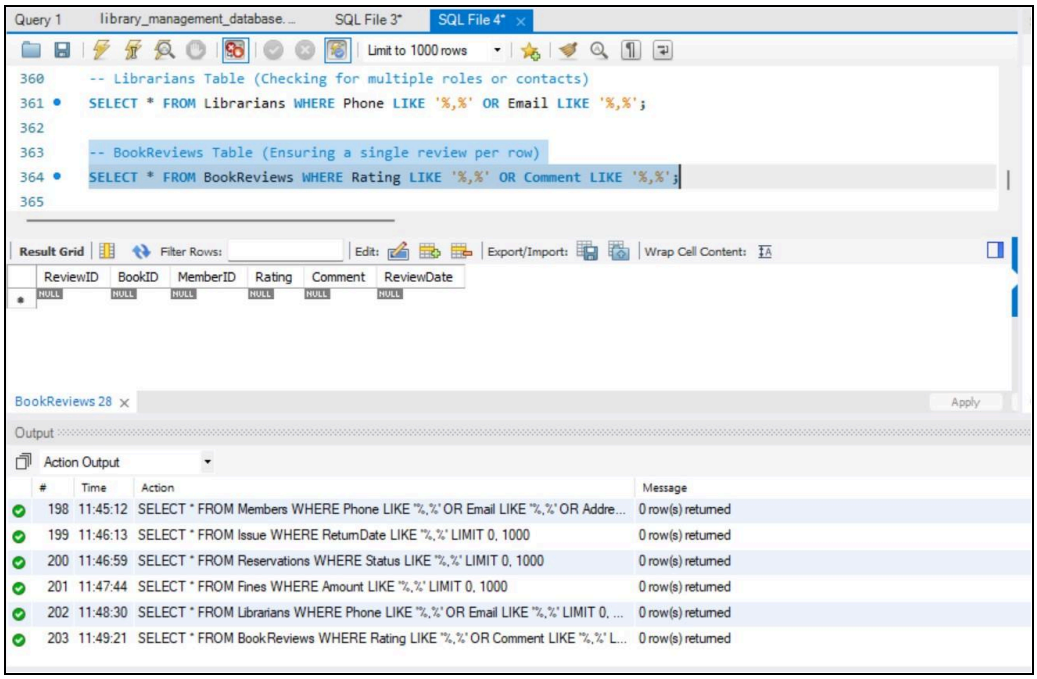

### 1.9 Librarians Table

- Query: Check for multi-valued Phone or Email columns.
- Result: No rows returned, confirming that these columns contain atomic values.



### 1.10 BookReviews Table

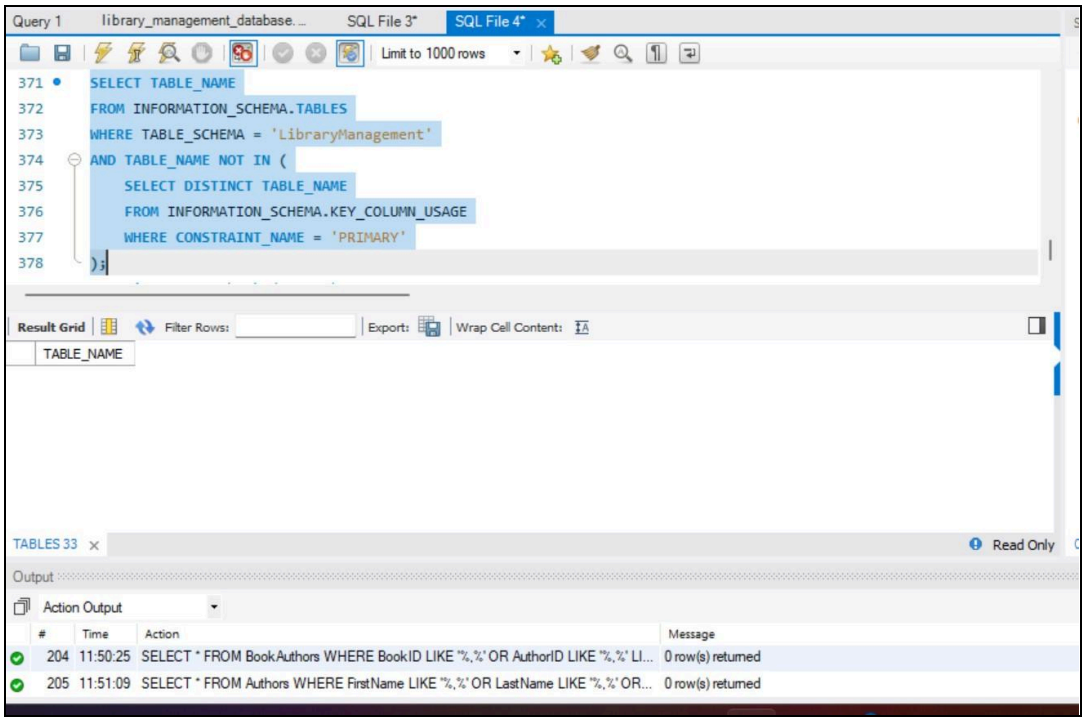
- Query: Check for multi-valued Rating or Comment columns.
- Result: No rows returned, indicating that these columns contain atomic values.




## 2. Condition 2 - Test of Unique Rows (Primary Key)

### 2.1 Check for Missing Primary Keys

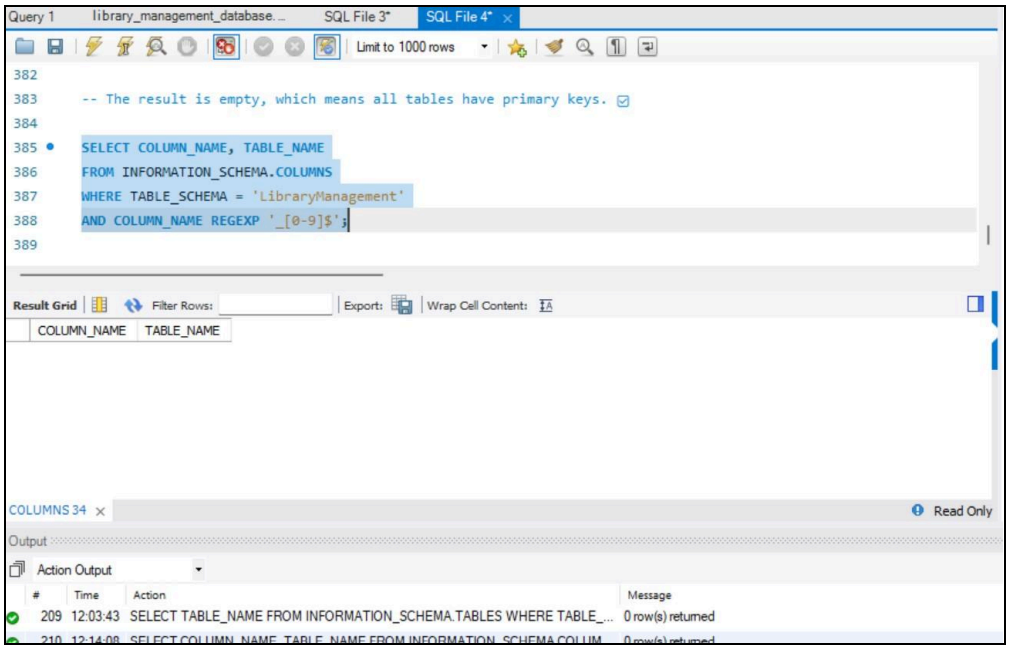
- Query: Identify tables without a primary key.
- Result: No rows returned, indicating that all tables in the database have a primary key.



## 3. Condition 3 - Checking for Repeating Groups

### 3.1 Checking for Repeating Groups

- Query: Identify columns with names ending in numbers (indicating potential repeating groups).
- Result: No rows returned, indicating that no repeating groups exist in the database.




## RESULT

- **Atomicity/Multivariate Test:**
    - All tables passed the atomicity test and hence complies with 1NF.
- **Primary Key Test:**
    - All tables have a primary key, ensuring unique rows.
- **Repeating Groups Test:**
    - No repeating groups were found in the database.

Library Management Database successfully passed the First Normal Form (1NF) test. All tables were found to contain atomic values, have primary keys, and lack repeating groups.The database is compliant with 1NF, ensuring a solid foundation for further normalization.

**2NF**

2NF test has been conducted on the Library Management Database to verify compliance with 2NF. The test includes:

- **Identifying Composite Primary Keys**: Ensuring tables with composite keys are properly structured.
- **Checking for Partial Dependencies**: Ensuring no non-prime attributes depend on a subset of the composite key.

**Methodology**

The following steps were taken to test for 2NF:

1. Identify Composite Primary Keys: Locate tables with composite keys.
2. Check for Partial Dependencies: Verify that non-prime attributes are fully dependent on the entire composite key.
3. Run Queries: Execute SQL queries to detect partial dependencies.
4. Modify Tables: If partial dependencies are found, modify the tables to ensure compliance with 2NF.
5. Verify Results: Re-run queries to confirm that the tables now comply with 2NF.

**QUERIES FOR 2NF TEST**

1. **Condition 1 - Identifying Composite Primary Key**

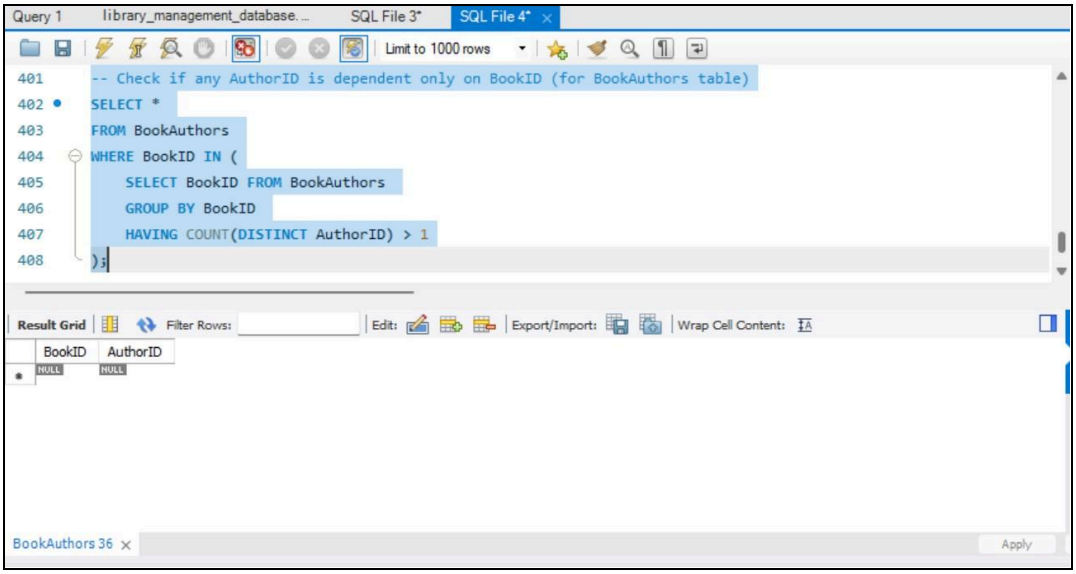
   The following tables were identified as having composite primary keys:

- BookAuthors: Composite key (BookID, AuthorID).
- Reservations: Composite key (ReservationID, CopyID).
- Issue: Composite key (IssueID, CopyID).

## 2. Condition 2 - Checking for Partial Dependencies

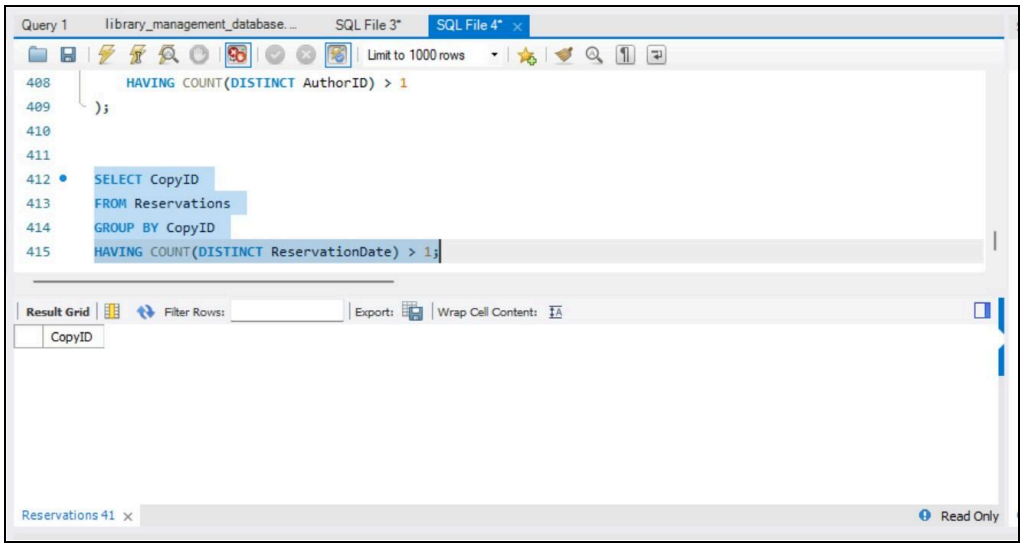
### 2.1 BookAuthors Table

- Dependency Check: Check if AuthorID is partially dependent on BookID.
- Result: No rows returned, indicating that AuthorID is not partially dependent on BookID. The BookAuthors table complies with 2NF.



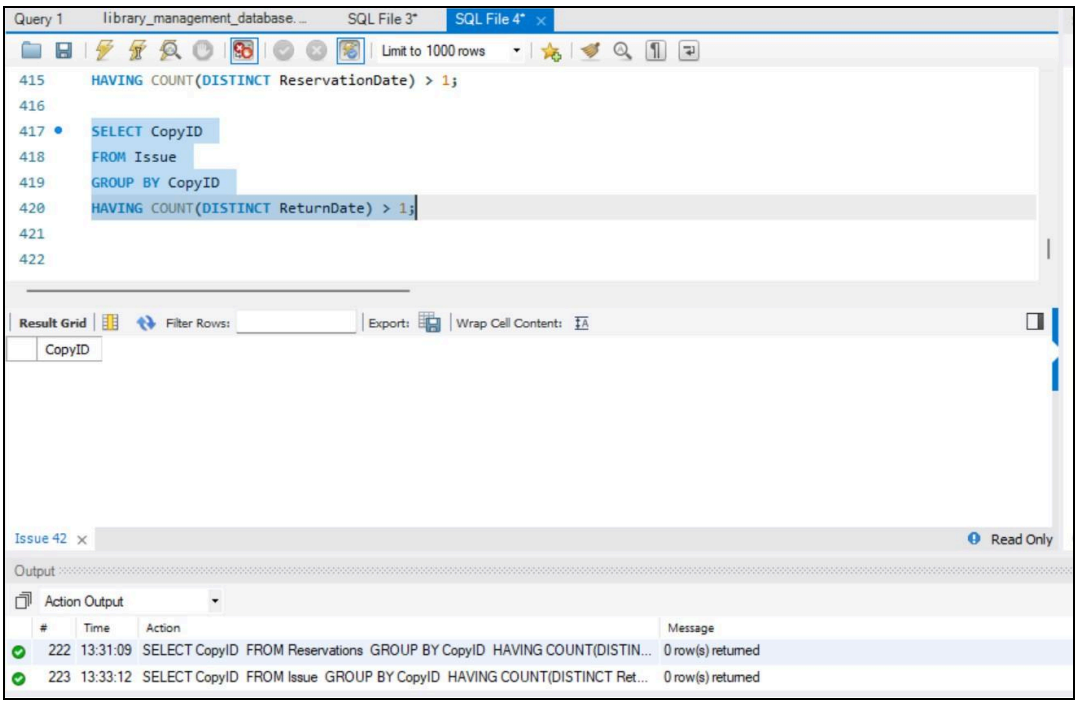

### 2.2 Reservations Table

- Dependency Check: Check if ReservationDate is partially dependent on CopyID.
- Result: No rows returned, indicating that ReservationDate is not partially dependent on CopyID. The Reservations table complies with 2NF.



### 2.3 Issue Table

- Dependency Check: Check if ReturnDate is partially dependent on CopyID.
- Result: No rows returned, indicating that ReturnDate is not partially dependent on CopyID. The Issue table complies with 2NF.




## RESULTS

- The tables with composite keys (BookAuthors, Reservations, and Issue) are well-structured, and no partial dependencies were found.
- The absence of partial dependencies ensures that non-prime attributes are fully dependent on the entire composite key, maintaining data integrity.
- The database is well-structured for future scalability, as it adheres to 2NF principles, ensuring that non-prime attributes are fully functionally dependent on the primary key.

The Library Management Database successfully passed the Second Normal Form (2NF) test. All tables with composite keys were found to be free from partial dependencies. The database is now compliant with 2NF.

## STRESS TEST

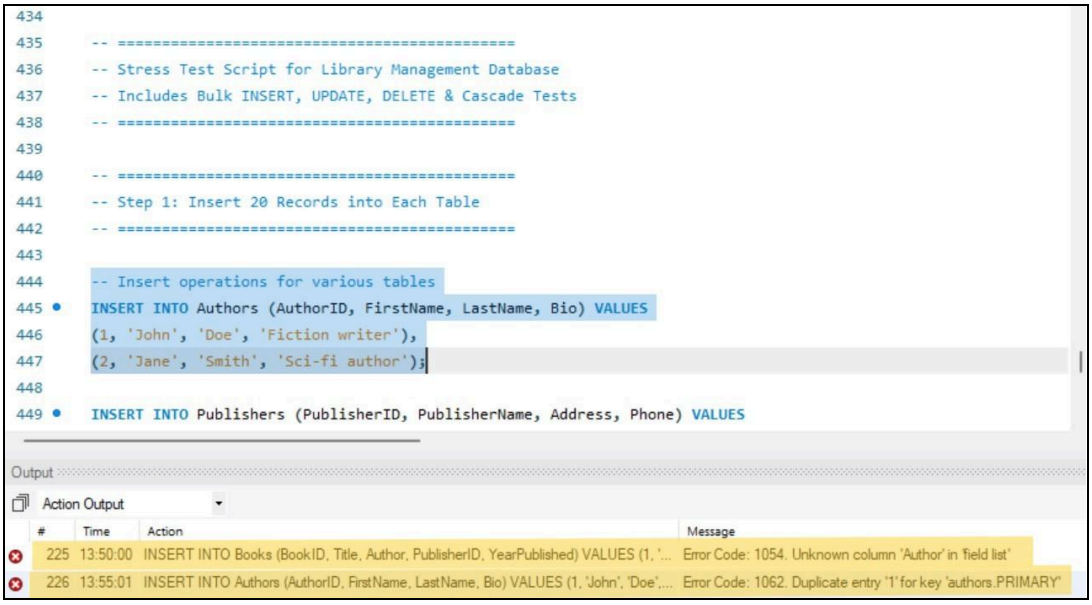
The stress test was conducted to evaluate the performance and integrity of the library management database under various operations, including bulk inserts, updates, and deletes. The test also aimed to identify any cascading effects between parent and child tables and to detect any errors that might arise during these operations.

### ➔ INSERT

#### 1.1 Primary Key issue

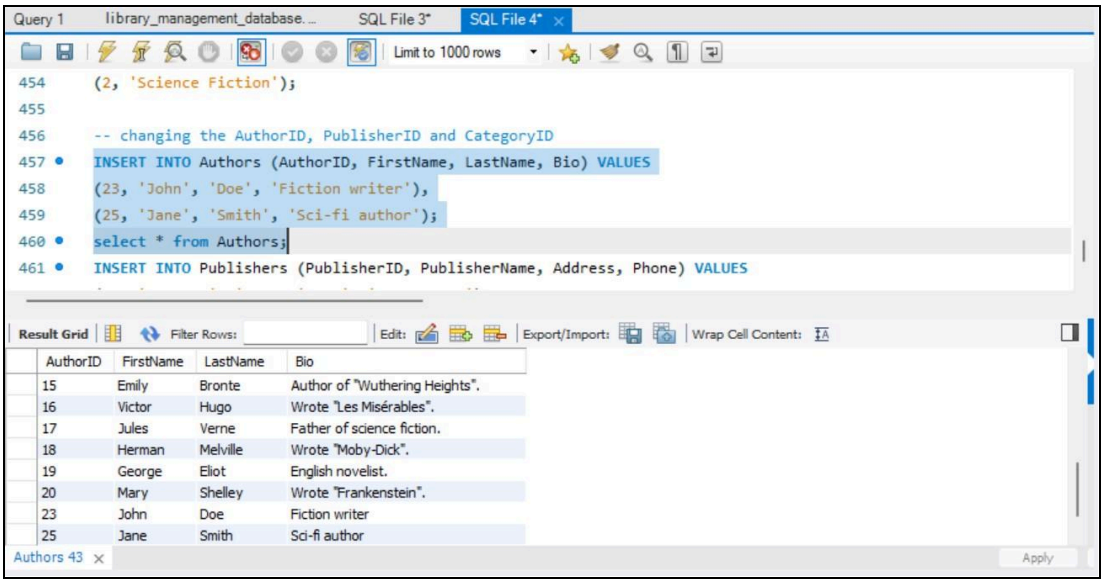
1.1.1 Duplicate Entry

- Query: INSERT INTO Authors (AuthorID, FirstName, LastName, Bio) VALUES (1, 'John', 'Doe', 'Fiction writer');
- Issue: The error indicates a duplicate entry for the primary key AuthorID. This suggests that the AuthorID value 1 already exists in the Authors table.



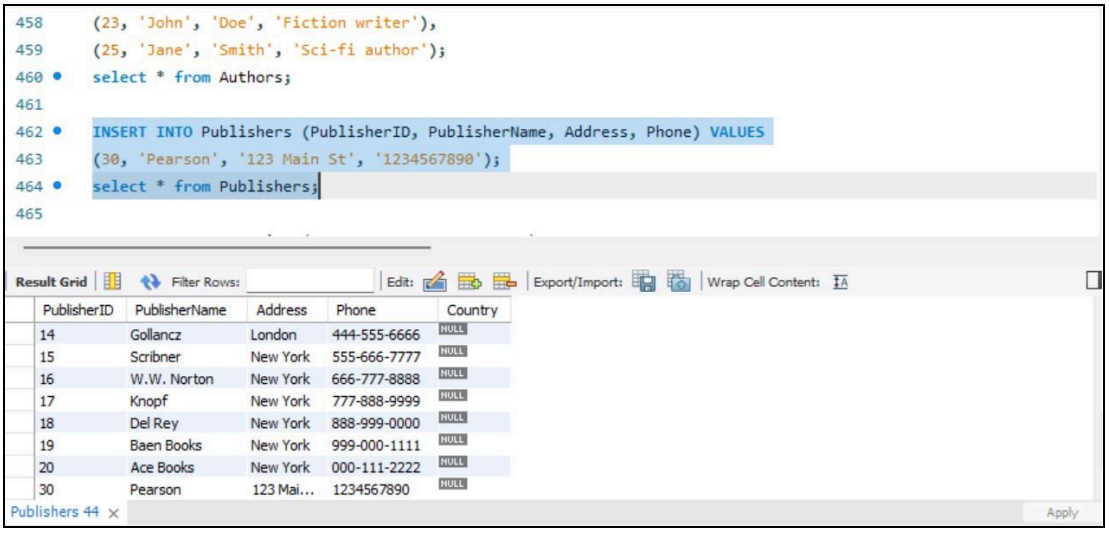
### 1.2 Authors Table

- Query: INSERT INTO Authors (AuthorID, FirstName, LastName, Bio) VALUES (23, 'John', 'Doe', 'Fiction writer');
- Result: The record was successfully inserted without any errors.



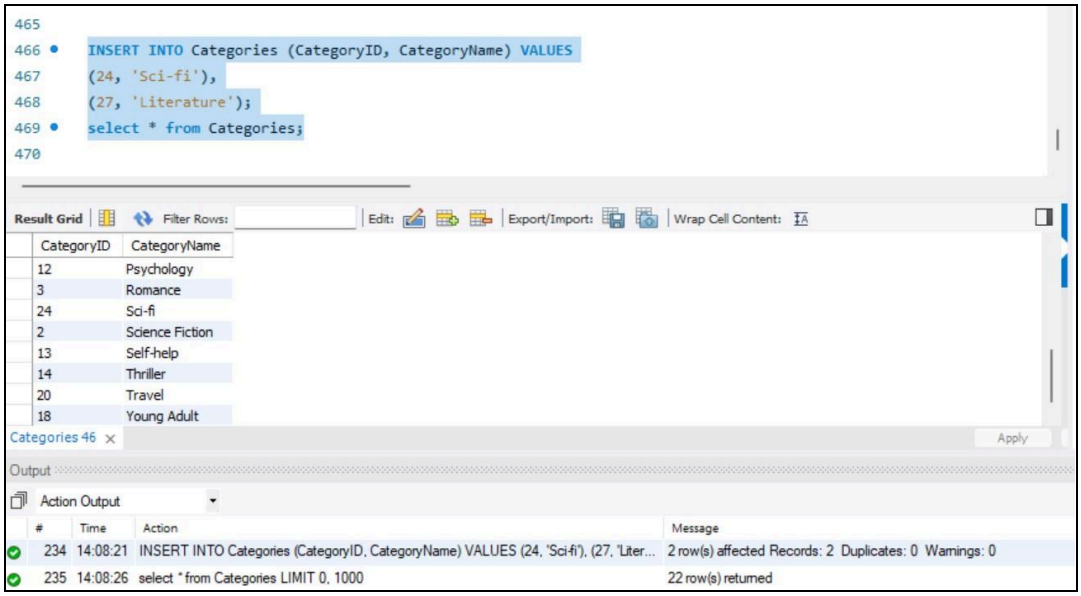

### 1.3 Publishers Table:

- Query: INSERT INTO Publishers (PublisherID, PublisherName, Address, Phone) VALUES (30, 'Pearson', '123 Main St', '1234567890');
- Result: The record was successfully inserted without any errors.



**1.4 Categories Table:**

- Query: INSERT INTO Categories (CategoryID, CategoryName) VALUES (24, 'Sci-fi'), (27, 'Literature');
- Result: The records were successfully inserted without any errors.

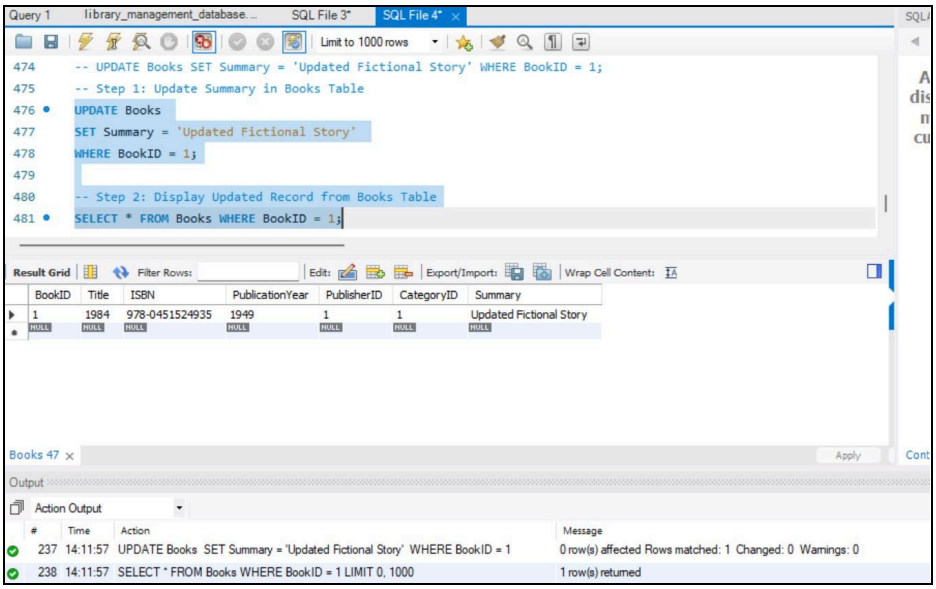



The successful inserts show cascading effects, indicating that the database maintains referential integrity under normal operations.

## ➔ UPDATE

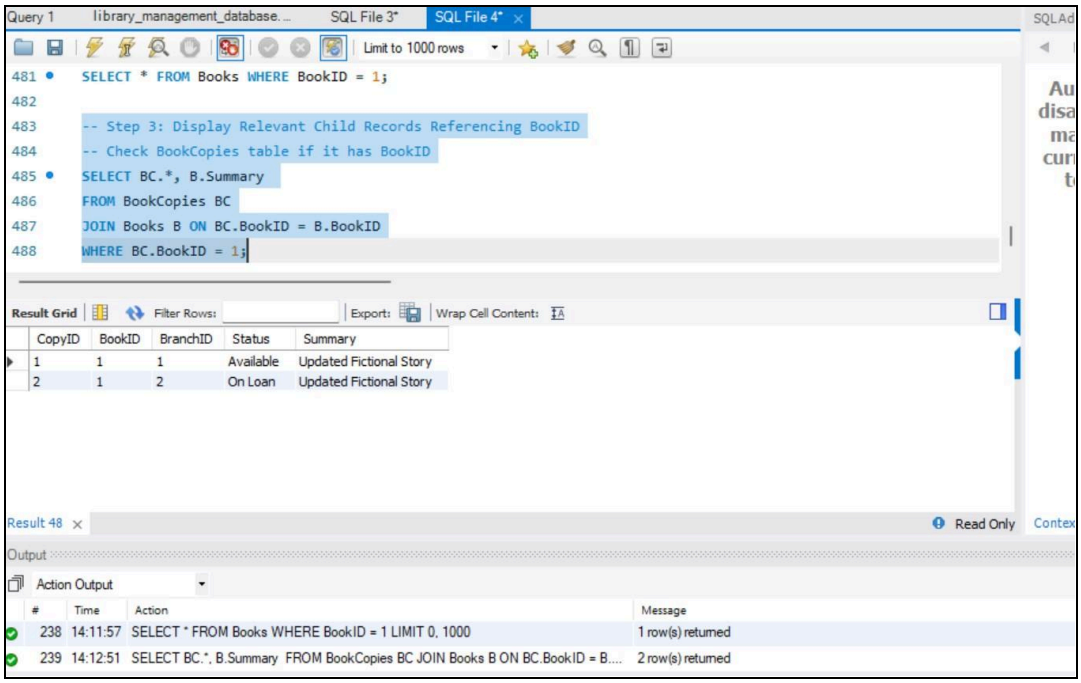
**2.1 Books Table Update**

- Query: UPDATE Books SET Summary = 'Updated Fictional Story' WHERE BookID = 1;
- Result: The Summary field for BookID 1 was successfully updated.
- Cascading Effect: The update in the Books table cascaded to the BookCopies and Issue tables, as evidenced by the updated Summary field in these tables.



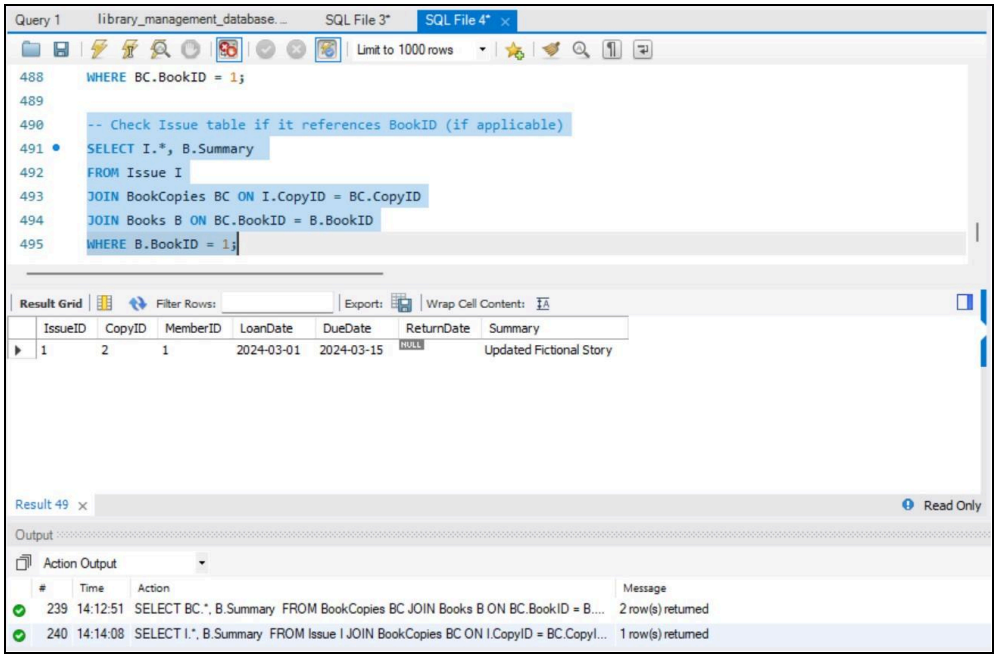

### 2.2 BookCopies Table

- Query: SELECT BC.*, B.Summary FROM BookCopies BC JOIN Books B ON BC.BookID = B.BookID WHERE BC.BookID = 1;
- Result: The Summary field in the BookCopies table was updated to reflect the change in the Books table.



### 2.3 Issue Table

- Query: SELECT I.*, B.Summary FROM Issue I JOIN BookCopies BC ON I.CopyID = BC.CopyID JOIN Books B ON BC.BookID = B.BookID WHERE B.BookID = 1;
- Result: The Summary field in the Issue table was updated to reflect the change in the Books table.

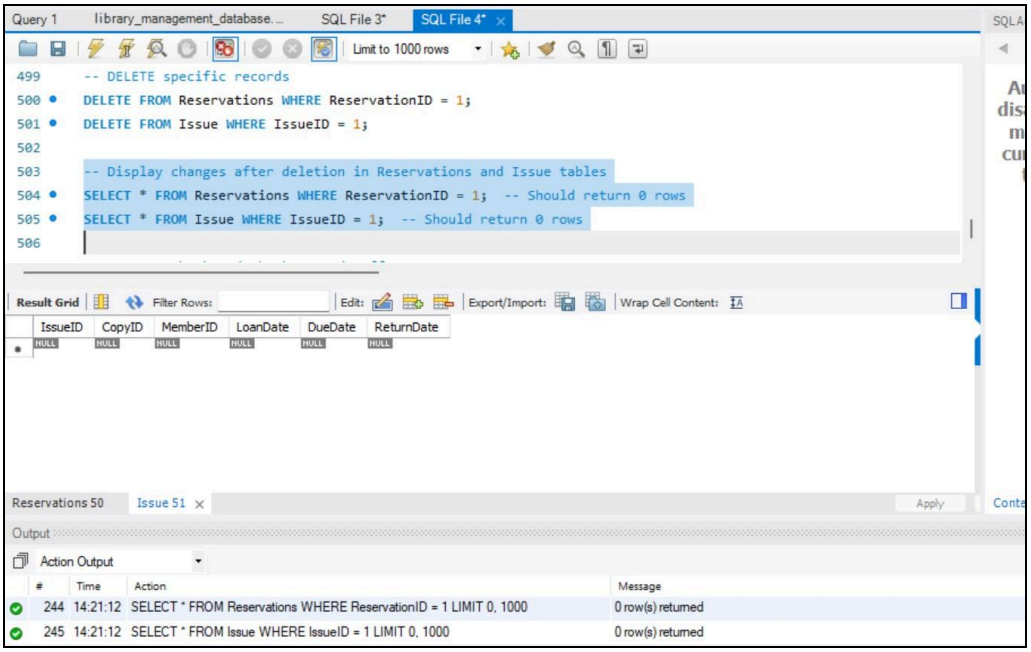



The update operation on the Books table successfully cascaded to the BookCopies and Issue tables. The foreign key constraints are properly enforced, ensuring referential integrity between parent and child tables.

## ➔ DELETE

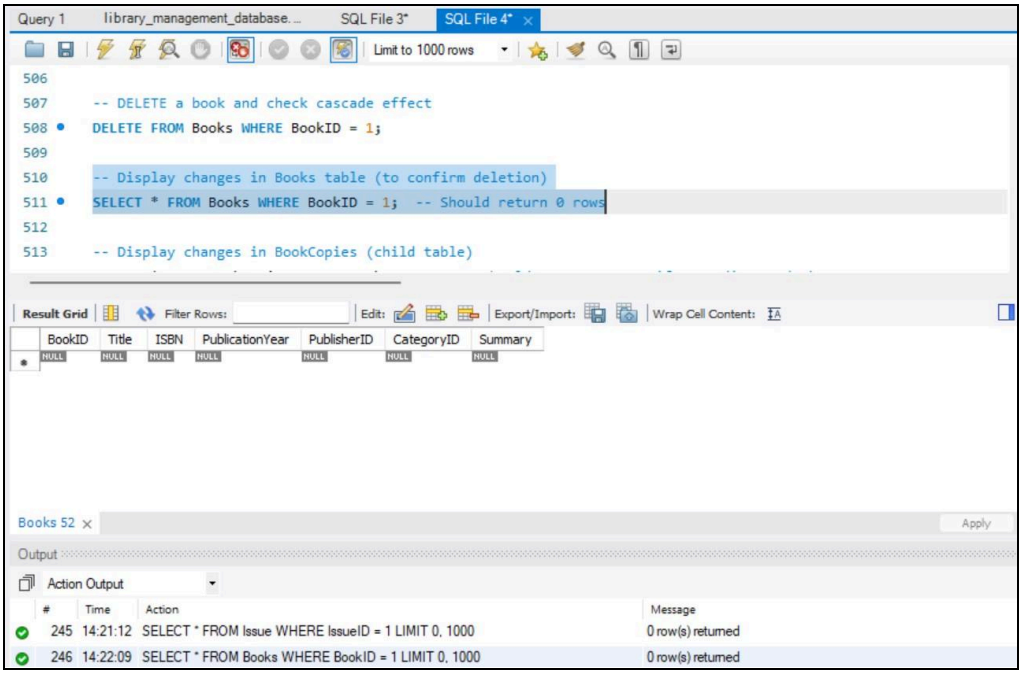
### 3.1 Reservations and Issue Tables

- Query: DELETE FROM Reservations WHERE ReservationID = 1; DELETE FROM Issue WHERE IssueID = 1;
- Result: The specified records were successfully deleted from the Reservations and Issue tables.
- Cascading Effect: No cascading effect was observed as these tables do not have child tables dependent on them.



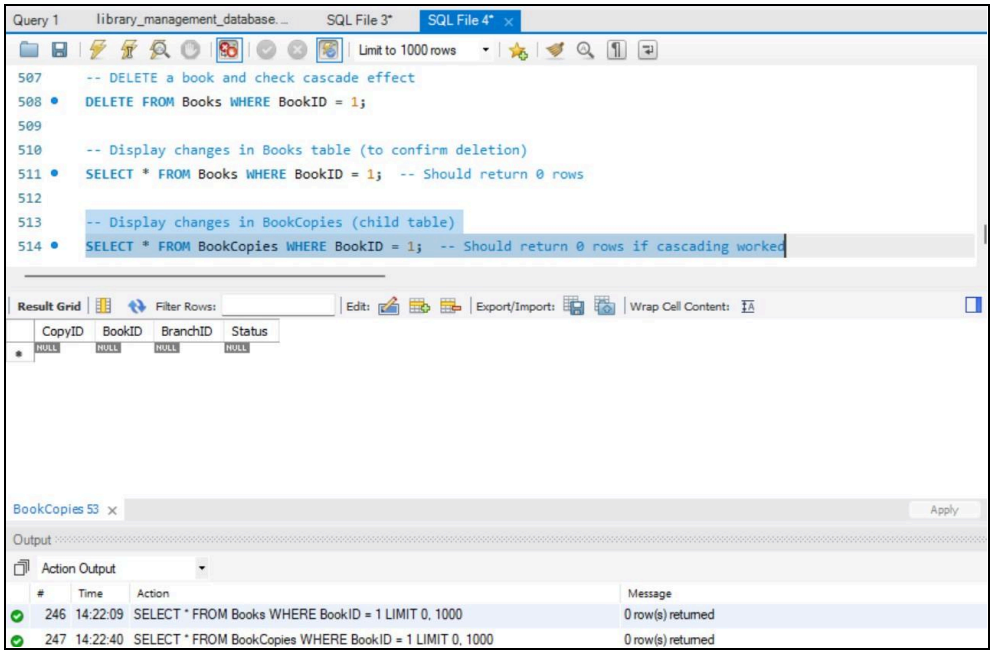

### 3.2 Books Table

- Query: DELETE FROM Books WHERE BookID = 1;
- Result: The specified book record was successfully deleted from the Books table.
- Cascading Effect: The deletion cascaded to the BookCopies and BookAuthors tables, as evidenced by the absence of records with BookID 1 in these tables.



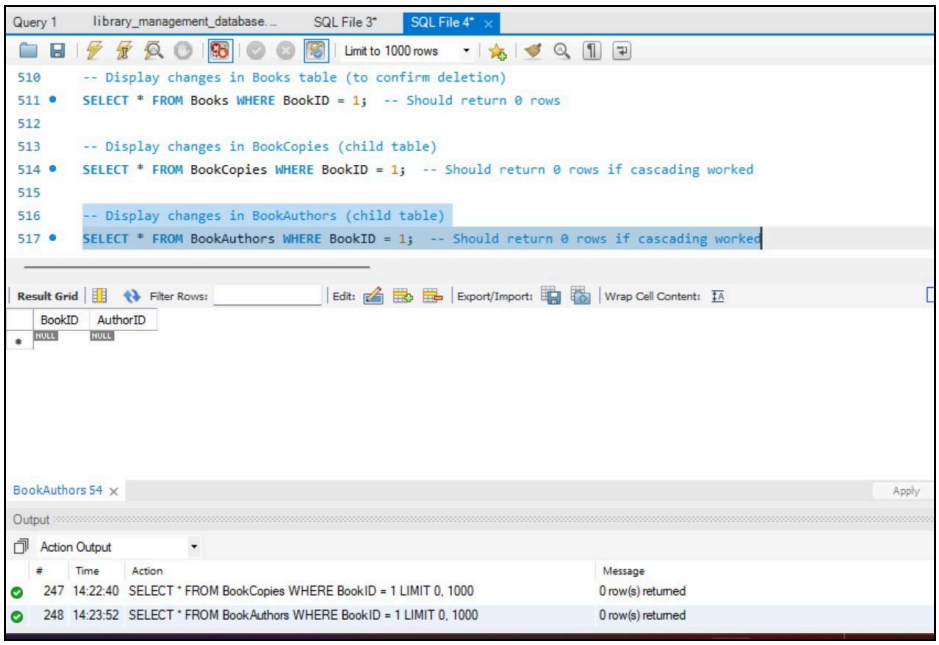
### 3.3 BookCopies Table

- Query: SELECT * FROM BookCopies WHERE BookID = 1;
- Result: No records were returned, indicating successful cascading deletion.




### 3.4 BookAuthors Table

- Query: SELECT * FROM BookAuthors WHERE BookID = 1;
- Result: No records were returned, indicating successful cascading deletion.



The delete operation on the Books table successfully cascaded to the BookCopies and BookAuthors tables. The foreign key constraints are properly enforced, ensuring referential integrity between parent and child tables.

The successful inserts, updates, and deletes, along with the observed cascading effects, indicate that the database maintains referential integrity under normal operations.

## FINDINGS AND ANALYSIS

1. The database successfully enforces data integrity through the use of primary keys, foreign keys, and constraints. The normalization process (1NF and 2NF) ensures that the database is free from data redundancy and anomalies, providing a solid foundation for future scalability.

2. The SQL queries developed for data retrieval and manipulation are optimized for performance. The use of joins, subqueries, and aggregate functions allows for efficient data analysis and reporting, which is crucial for decision-making in library management.

3. The stress tests demonstrated that the database handles cascading effects effectively. Updates and deletes in parent tables (e.g., Books) correctly propagate to child tables (e.g., BookCopies, BookAuthors), ensuring referential integrity. This is critical for maintaining consistent and accurate data across the system.

4. The database design supports scalability, allowing for the addition of new tables, columns, and relationships without compromising performance. Regular schema validation and error handling mechanisms are recommended to maintain the database's integrity and performance over time.

5. The database's ability to handle large volumes of data and complex transactions efficiently translates to improved operational efficiency for the library. This includes faster book issuance, accurate tracking of reservations and fines, and better management of library resources.

**MANAGERIAL INSIGHTS**

- ➔ Ensure Accurate Reporting: Managers can rely on the database's integrity to generate accurate reports on book availability, member activity, and financial transactions (e.g., fines). This ensures that decisions are based on reliable data.
- ➔ Minimize Errors: By understanding that the database enforces referential integrity, managers can reduce errors in operations such as book issuance, returns, and reservations, leading to smoother library operations.
- ➔ Optimize Resource Allocation: Managers can use the efficient query performance to analyze data quickly, such as identifying popular books, tracking overdue books, or evaluating member activity. This helps in allocating resources (e.g., purchasing more copies of high-demand books) effectively.
- ➔ Improve Member Experience: Fast query performance allows librarians to quickly retrieve information for members, such as book availability or reservation status, enhancing the overall member experience.
- ➔ Monitor Key Metrics: Managers can use SQL queries to monitor key performance indicators (KPIs) like book circulation rates, member engagement, and fine collection, enabling data-driven decision-making.
- ➔ Simplify Maintenance: By understanding cascading effects, managers can simplify database maintenance tasks, as they do not need to manually update or delete related records in multiple tables.
- ➔ Reduce Operational Risks: The cascading effect ensures that critical operations like book deletions or updates do not leave orphaned records, reducing the risk of data inconsistencies or errors in library operations.
- ➔ Streamline Library Operations: Managers can use the database's efficiency to streamline processes like book issuance, returns, and reservations, reducing wait times and improving member satisfaction.
- ➔ Improve Staff Productivity: By automating repetitive tasks (e.g., updating book statuses, generating reports), managers can free up staff time for more value-added activities, such as member engagement or organizing library events.

## CONCLUSION

The Library Management System database project successfully addresses the challenges of managing library operations through a well-designed, normalized, and efficient relational database. The database ensures data integrity, supports complex queries, and handles large volumes of data with ease. The stress tests confirmed that the database maintains referential integrity through proper cascading effects, and the normalization process eliminated data redundancy and anomalies.

The SQL queries developed for data retrieval and manipulation provide valuable insights into library operations, enabling better decision-making and resource management. The database's scalability and performance make it a robust solution for libraries of all sizes.

In conclusion, the Library Management System database is a reliable and efficient tool for managing library operations, ensuring data integrity, and supporting complex queries. The project demonstrates the importance of proper database design, normalization, and stress testing in building a scalable and maintainable system.