CIS 552 Database Design
Final Project

Library Management Systems
Group 9

Submitted by

Sravani Kairam Konda - 02130417
Pragna Koneru – 02094308
Annie Gibian S – 02131167

Submitted to
Yukui Luo,Phd

Submitted to the Department of Computer Science.
University Of Massachusetts
North Dartmouth, Massachusetts

Problem statement:

The project involves the creation of a robust Library Management System (LMS) database with a focus on efficient book inventory management, user interactions, and administrative tasks. The system is designed to store comprehensive information about books, authors, genres, and users. It supports operations like adding, modifying, and deleting books and users, as well as recording transactions for checkouts and returns. The database aims to facilitate reporting and analytics, offering insights into popular books, genres, and overdue items. The design decisions and assumptions made during development will be documented, and SQL queries for essential operations and sample data insertion will be provided. The goal is to create a scalable, efficient, and user-friendly system for effective library management.

Introduction:

The Library Management System (LMS) database is a pivotal component in the realm of information management, dedicated to optimizing library operations. This structured schema aims to efficiently manage books, user interactions, and administrative tasks. Key features include dynamic book management, user registration, transaction tracking, author and genre organization, reporting tools, and performance optimization. Ensuring data integrity and categorization, the system is designed for scalability, regular maintenance, and seamless integration with a user-friendly interface. Ultimately, the LMS database strives to elevate the efficiency, accuracy, and user satisfaction in library operations, adapting to the evolving landscape of digital information.

Database Design:

The design of the Library Management System (LMS) database involves structuring and organizing data to efficiently handle various aspects of library operations. The schema is designed to provide flexibility, maintainability, and performance. Below is an explanation of the key components and design decisions:

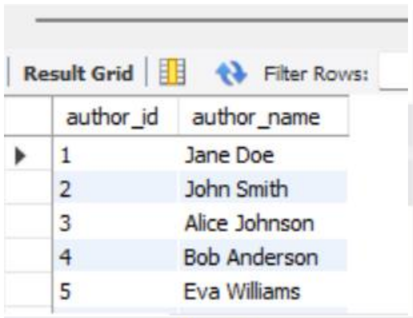
Entities and Tables:
Books Table: Central to the system, the books table stores information about each book, including book_id, title, author_id, genre_id, ISBN, publication_year, and available_copies.
Authors Table: The authors table holds information about authors, including author_id and author_name.
Genres Table: The genres table contains details about book genres, including genre_id and genre_name.
Users Table: The users table manages user information with fields such as user_id, user_name, email, and phone_number.
Transactions Table: The transactions table records book transactions, capturing transaction_id, user_id, book_id, checkout_date, and return_date.

Relationships:
The books table has foreign key relationships with both authors and genres tables, linking books to their respective authors and genres.

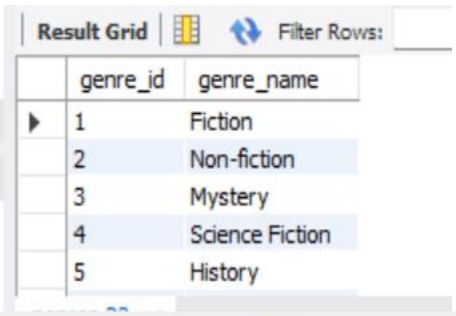
The transactions table establishes relationships with both the users and books tables, connecting transactions to specific users and books.

Normalization:

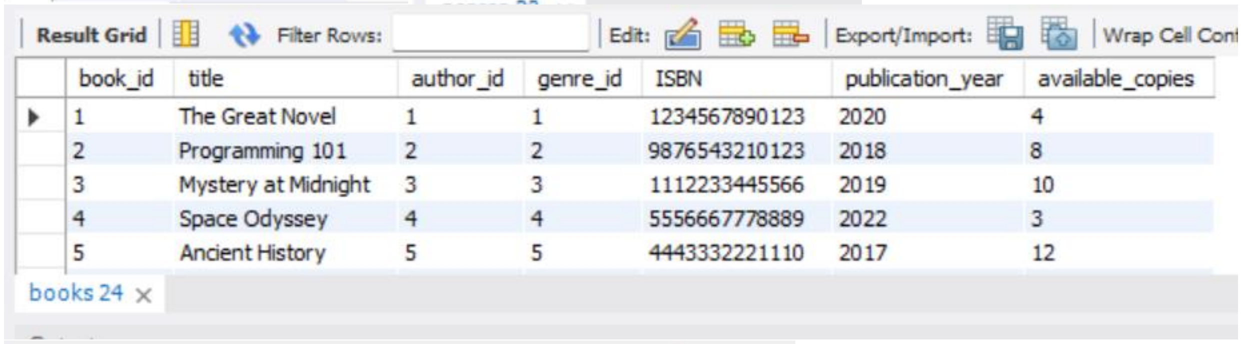
The schema follows normalization principles, minimizing redundancy and ensuring data integrity. Author and genre information is stored in separate tables, preventing duplication and facilitating updates.



author_id	author_name
1	Jane Doe
2	John Smith
3	Alice Johnson
4	Bob Anderson
5	Eva Williams

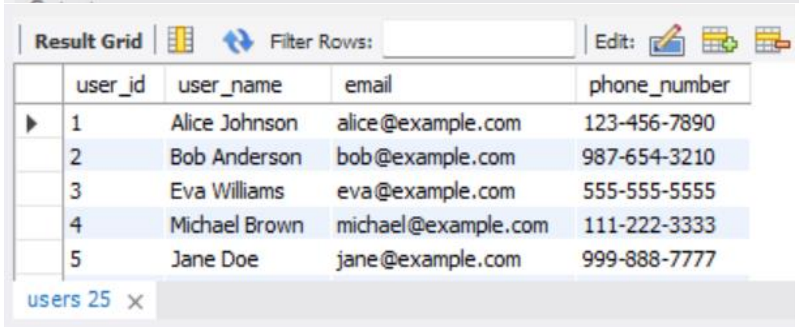


genre_id	genre_name
1	Fiction
2	Non-fiction
3	Mystery
4	Science Fiction
5	History



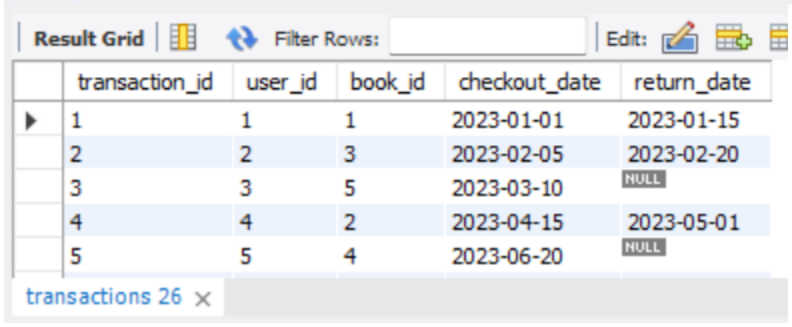
book_id	title	author_id	genre_id	ISBN	publication_year	available_copies
1	The Great Novel	1	1	1234567890123	2020	4
2	Programming 101	2	2	9876543210123	2018	8
3	Mystery at Midnight	3	3	1112233445566	2019	10
4	Space Odyssey	4	4	5556667778889	2022	3
5	Ancient History	5	5	4443332221110	2017	12

books 24



user_id	user_name	email	phone_number
1	Alice Johnson	alice@example.com	123-456-7890
2	Bob Anderson	bob@example.com	987-654-3210
3	Eva Williams	eva@example.com	555-555-5555
4	Michael Brown	michael@example.com	111-222-3333
5	Jane Doe	jane@example.com	999-888-7777

users 25



transaction_id	user_id	book_id	checkout_date	return_date
1	1	1	2023-01-01	2023-01-15
2	2	3	2023-02-05	2023-02-20
3	3	5	2023-03-10	NULL
4	4	2	2023-04-15	2023-05-01
5	5	4	2023-06-20	NULL

transactions 26

Data Integrity:

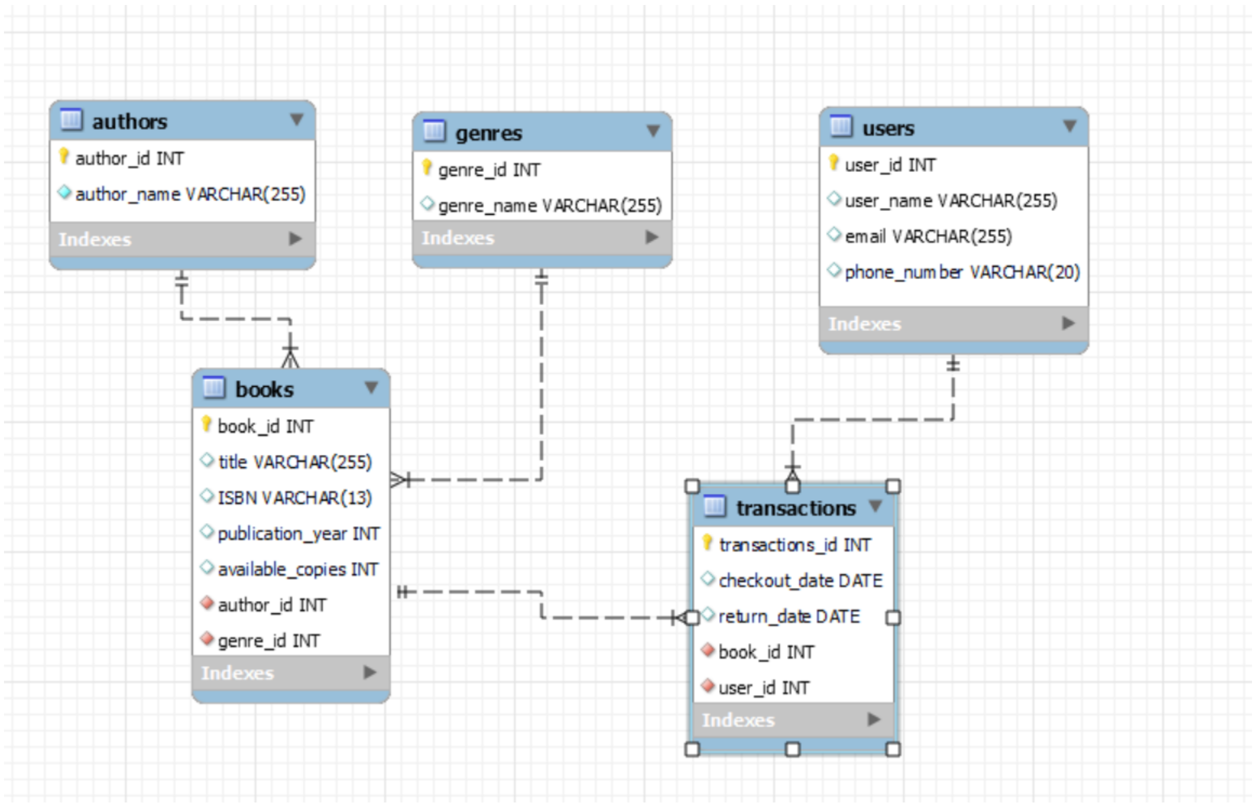
Data integrity in a database ensures the accuracy, consistency, and reliability of the stored data. The Library Management System (LMS) database schema is designed with various mechanisms to enforce data integrity. Here are some aspects of data integrity in the provided schema:

Primary Keys:
Primary keys uniquely identify each record in a table. In the LMS database schema:
book_id is the primary key in the books table.
author_id is the primary key in the authors table.
genre_id is the primary key in the genres table.
user_id is the primary key in the users table.
transaction_id is the primary key in the transactions table.
Foreign Keys:
Foreign keys establish relationships between tables, ensuring referential integrity. Examples include:
author_id in the books table is a foreign key referencing authors(author_id).
genre_id in the books table is a foreign key referencing genres(genre_id).
user_id in the transactions table is a foreign key referencing users(user_id).
book_id in the transactions table is a foreign key referencing books(book_id).
Indexes:
Indexes are applied to columns frequently used in search conditions and joins for optimal query performance.
Triggers:
Triggers are employed to automate certain actions, such as updating the number of available copies after a book is checked out.
Categorization:
The schema is organized into logical categories such as Book Information, User Information, Transactions, etc., providing clarity and ease of maintenance.
Constraints:
Primary and foreign key constraints are applied to enforce data integrity.
These mechanisms collectively contribute to data integrity by preventing inconsistencies, ensuring accurate relationships between tables, and enforcing constraints on the values stored in the database.

UML Diagrams:

Unified Modeling Language (UML) diagrams serve as visual representations for illustrating the interconnections within a database's various schemas. These graphical depictions showcase the relationships among different schemas, providing insights into their connections.
The database encompasses multiple schemas, each meticulously depicted through MySQL Workbench. The UML diagram functions as a logical model, offering a comprehensive understanding of how tables are intricately linked in the database. Key components in this schema include tables housing primary keys like author_id,book_id,genre_id etc. The linkage between tables is established through foreign

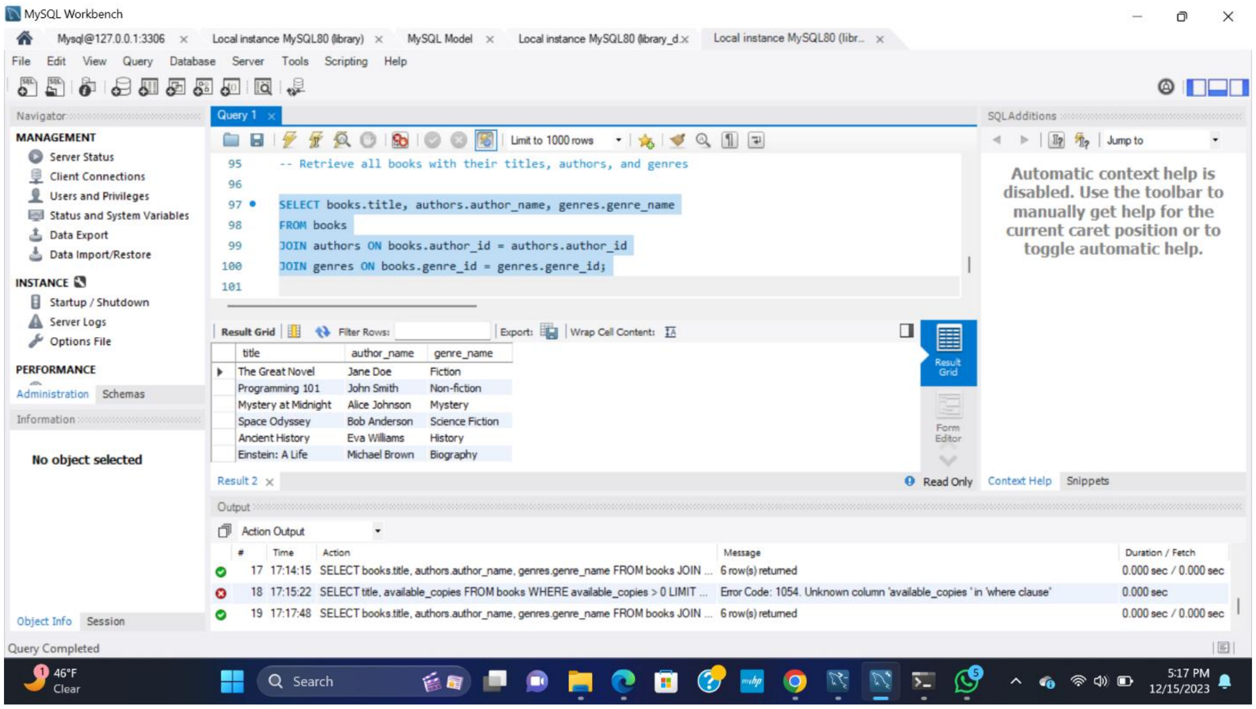
keys, which correspond to the primary keys in other tables, creating a cohesive structure that defines the database's relational architecture.



Data Collection & SQL Development:

Analyzing databases:

- The below query retrieves a result set that includes the title of each book, the name of its author, and the genre to which it belongs. The result provides a comprehensive view of the books in the database, presenting information about their titles, authors, and genres in a single output.

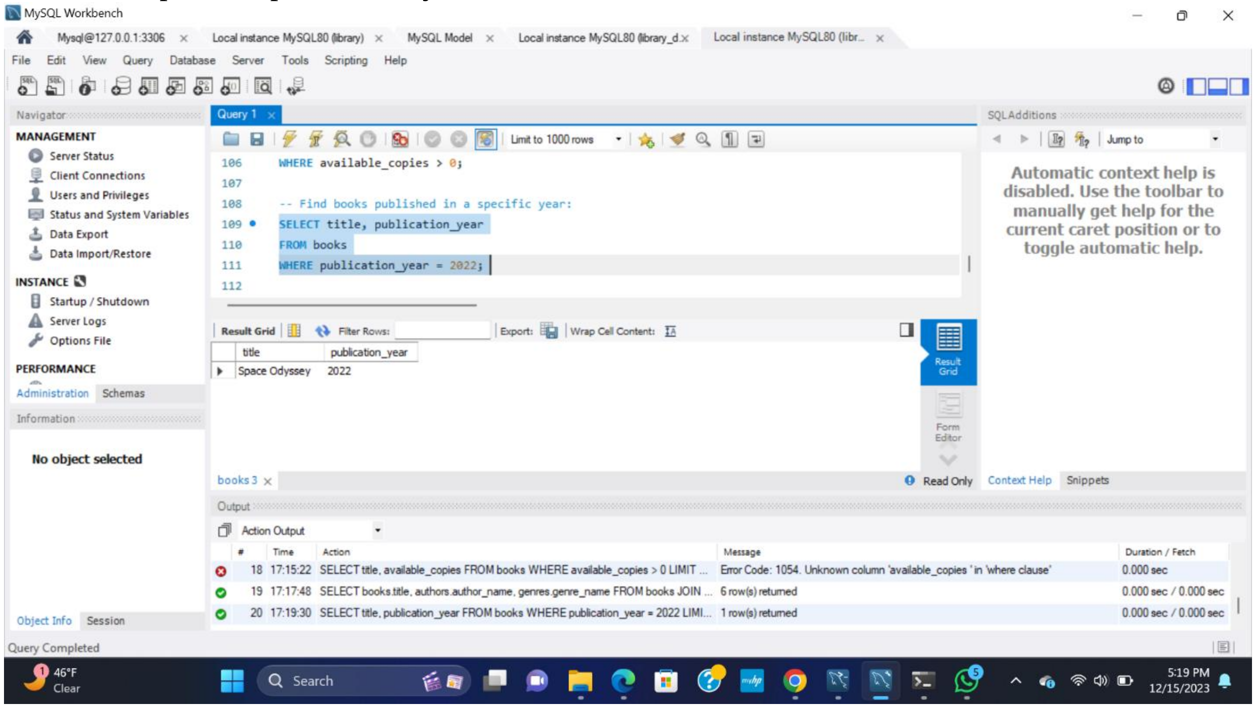


```
-- Retrieve all books with their titles, authors, and genres

SELECT books.title, authors.author_name, genres.genre_name
FROM books
JOIN authors ON books.author_id = authors.author_id
JOIN genres ON books.genre_id = genres.genre_id;
```

title	author_name	genre_name
The Great Novel	Jane Doe	Fiction
Programming 101	John Smith	Non-fiction
Mystery at Midnight	Alice Johnson	Mystery
Space Odyssey	Bob Anderson	Science Fiction
Ancient History	Eva Williams	History
Einstein: A Life	Michael Brown	Biography

```
#   Time      Action                                                                       Message                                                                  Duration / Fetch
17  17:14:15  SELECT books.title, authors.author_name, genres.genre_name FROM books JOIN ...  6 row(s) returned                                                        0.000 sec / 0.000 sec
18  17:15:22  SELECT title, available_copies FROM books WHERE available_copies > 0 LIMIT ...  Error Code: 1054. Unknown column 'available_copies ' in 'where clause'   0.000 sec
19  17:17:48  SELECT books.title, authors.author_name, genres.genre_name FROM books JOIN ...  6 row(s) returned                                                        0.000 sec / 0.000 sec
```

- The query retrieves a result set containing the titles and publication years of books that were published specifically in the year 2022. The result provides a list of books that meet the specified publication year criterion.

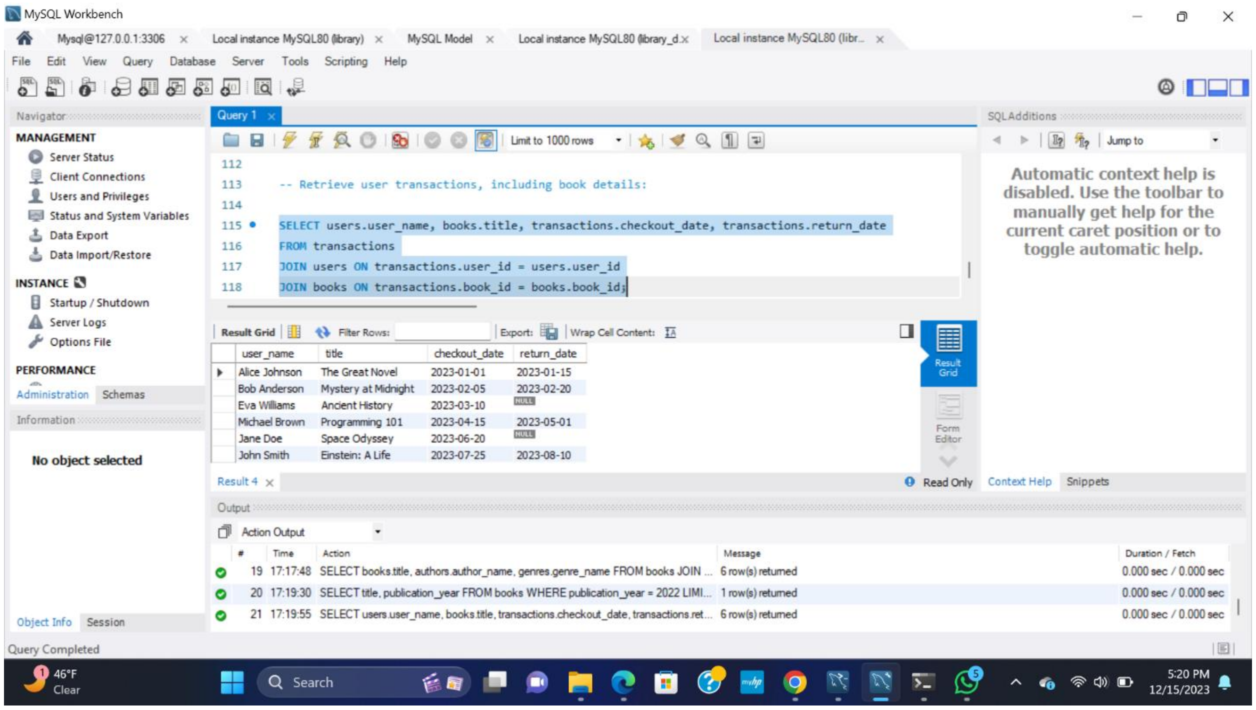


```
WHERE available_copies > 0;

-- Find books published in a specific year:
SELECT title, publication_year
FROM books
WHERE publication_year = 2022;
```

title	publication_year
Space Odyssey	2022

```
#   Time      Action                                                                       Message                                                                  Duration / Fetch
18  17:15:22  SELECT title, available_copies FROM books WHERE available_copies > 0 LIMIT ...  Error Code: 1054. Unknown column 'available_copies ' in 'where clause'   0.000 sec
19  17:17:48  SELECT books.title, authors.author_name, genres.genre_name FROM books JOIN ...  6 row(s) returned                                                        0.000 sec / 0.000 sec
20  17:19:30  SELECT title, publication_year FROM books WHERE publication_year = 2022 LIMI...  1 row(s) returned                                                        0.000 sec / 0.000 sec
```

- The query retrieves a result set that includes the ‘user name’, book title, checkout date, and return date for each transaction recorded in the database. It provides a comprehensive view of user interactions with the library, displaying details about the borrowed books and the associated transaction dates.

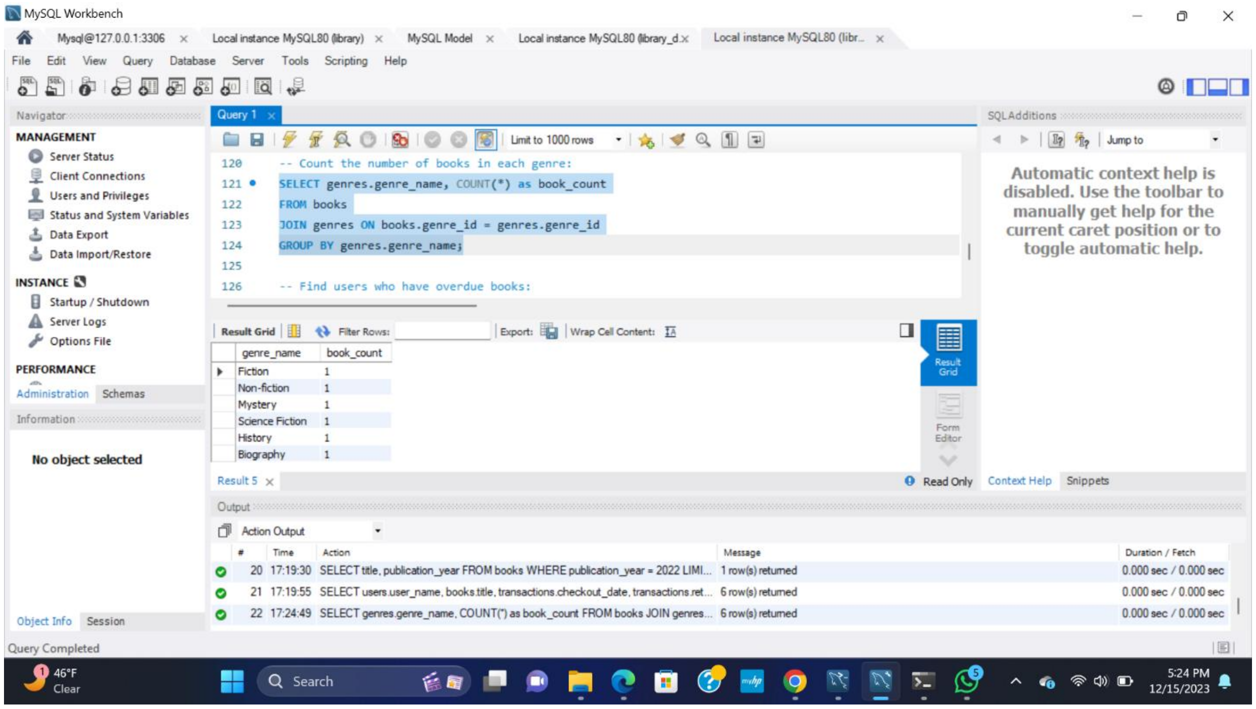


```sql
-- Retrieve user transactions, including book details:

SELECT users.user_name, books.title, transactions.checkout_date, transactions.return_date
FROM transactions
JOIN users ON transactions.user_id = users.user_id
JOIN books ON transactions.book_id = books.book_id;
```

user_name	title	checkout_date	return_date
Alice Johnson	The Great Novel	2023-01-01	2023-01-15
Bob Anderson	Mystery at Midnight	2023-02-05	2023-02-20
Eva Williams	Ancient History	2023-03-10	NULL
Michael Brown	Programming 101	2023-04-15	2023-05-01
Jane Doe	Space Odyssey	2023-06-20	NULL
John Smith	Einstein: A Life	2023-07-25	2023-08-10

```
#   Time      Action                                                                        Message           Duration / Fetch
19  17:17:48  SELECT books.title, authors.author_name, genres.genre_name FROM books JOIN ...  6 row(s) returned  0.000 sec / 0.000 sec
20  17:19:30  SELECT title, publication_year FROM books WHERE publication_year = 2022 LIMI...  1 row(s) returned  0.000 sec / 0.000 sec
21  17:19:55  SELECT users.user_name, books.title, transactions.checkout_date, transactions.ret...  6 row(s) returned  0.000 sec / 0.000 sec
```

- The query retrieves a result set that includes the genre names and the count of books in each genre. It provides a summary view of the library's book distribution across different genres, allowing users to see how many books belong to each genre category.

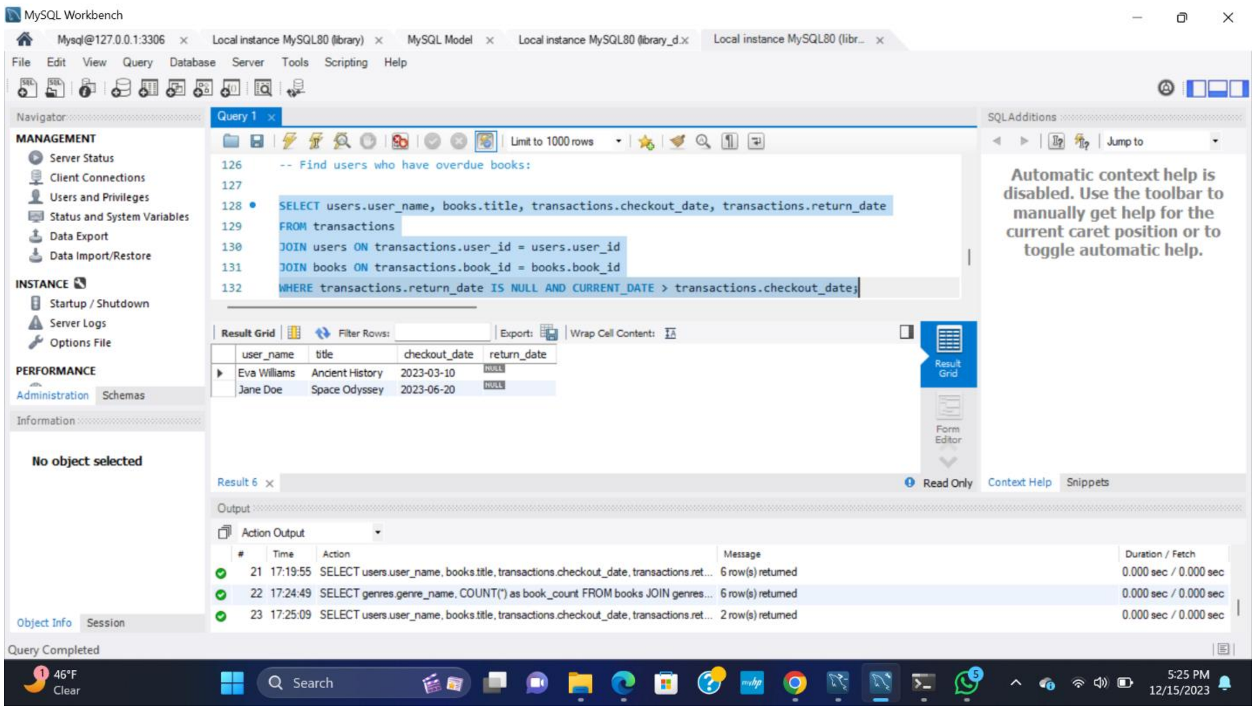


```sql
-- Count the number of books in each genre:
SELECT genres.genre_name, COUNT(*) as book_count
FROM books
JOIN genres ON books.genre_id = genres.genre_id
GROUP BY genres.genre_name;

-- Find users who have overdue books:
```

genre_name	book_count
Fiction	1
Non-fiction	1
Mystery	1
Science Fiction	1
History	1
Biography	1

```
#   Time      Action                                                                        Message           Duration / Fetch
20  17:19:30  SELECT title, publication_year FROM books WHERE publication_year = 2022 LIMI...  1 row(s) returned  0.000 sec / 0.000 sec
21  17:19:55  SELECT users.user_name, books.title, transactions.checkout_date, transactions.ret...  6 row(s) returned  0.000 sec / 0.000 sec
22  17:24:49  SELECT genres.genre_name, COUNT(*) as book_count FROM books JOIN genres...  6 row(s) returned  0.000 sec / 0.000 sec
```

- The query retrieves a result set that includes the ‘user name’, book title, checkout date, and return date for transactions where the book has not been returned, and the current date is past the expected return date. This provides a list of users with overdue books, helping the library identify and manage outstanding book returns.

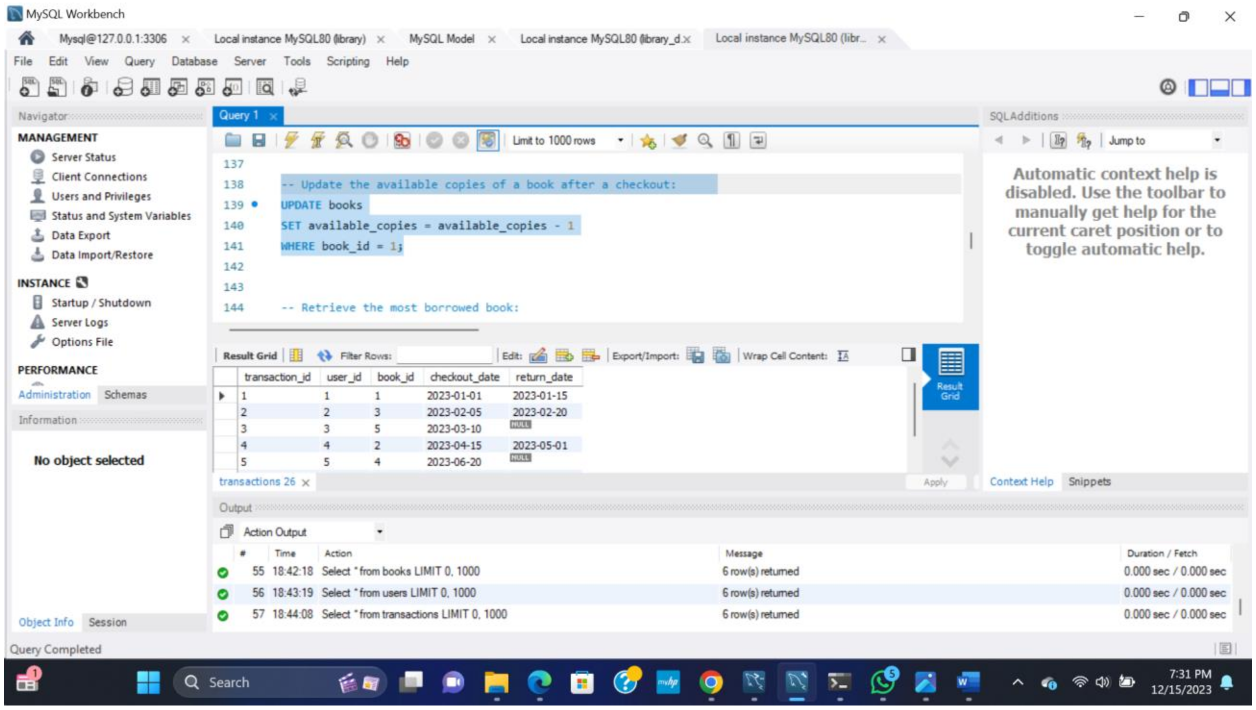


```sql
-- Find users who have overdue books:

SELECT users.user_name, books.title, transactions.checkout_date, transactions.return_date
FROM transactions
JOIN users ON transactions.user_id = users.user_id
JOIN books ON transactions.book_id = books.book_id
WHERE transactions.return_date IS NULL AND CURRENT_DATE > transactions.checkout_date;
```

```
user_name     title            checkout_date   return_date
Eva Williams  Ancient History  2023-03-10      NULL
Jane Doe      Space Odyssey    2023-06-20      NULL
```

```
#   Time      Action                                                                          Message              Duration / Fetch
21  17:19:55  SELECT users.user_name, books.title, transactions.checkout_date, transactions.ret...  6 row(s) returned  0.000 sec / 0.000 sec
22  17:24:49  SELECT genres.genre_name, COUNT(*) as book_count FROM books JOIN genres...           6 row(s) returned  0.000 sec / 0.000 sec
23  17:25:09  SELECT users.user_name, books.title, transactions.checkout_date, transactions.ret...  2 row(s) returned  0.000 sec / 0.000 sec
```

- The query updates the ‘available_copies’ column for a specific book ‘(book_id = 1)’ in the books table after a checkout. It reduces the number of available copies by 1, reflecting that a copy of the book has been checked out. This type of query is commonly used in a Library Management System to keep track of the availability of books after transactions such as checkouts or returns.

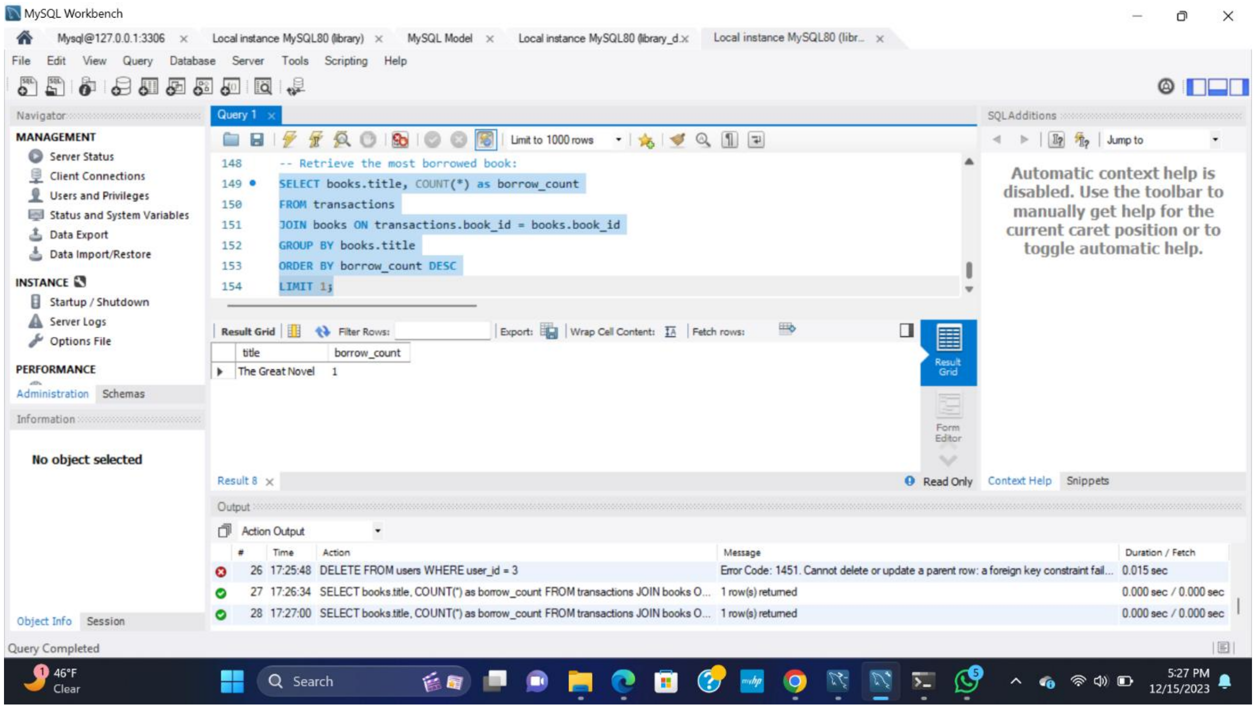


```sql

-- Update the available copies of a book after a checkout:
UPDATE books
SET available_copies = available_copies - 1
WHERE book_id = 1;


-- Retrieve the most borrowed book:
```

transaction_id	user_id	book_id	checkout_date	return_date
1	1	1	2023-01-01	2023-01-15
2	2	3	2023-02-05	2023-02-20
3	3	5	2023-03-10	NULL
4	4	2	2023-04-15	2023-05-01
5	5	4	2023-06-20	NULL

```
#   Time      Action                                      Message            Duration / Fetch
55  18:42:18  Select * from books LIMIT 0, 1000           6 row(s) returned  0.000 sec / 0.000 sec
56  18:43:19  Select * from users LIMIT 0, 1000           6 row(s) returned  0.000 sec / 0.000 sec
57  18:44:08  Select * from transactions LIMIT 0, 1000    6 row(s) returned  0.000 sec / 0.000 sec
```

- The query retrieves a result set that includes the title of the most borrowed book and the count of transactions for that book. The result provides information about the book that has been borrowed the most based on the transaction data in the transactions table.



```sql
-- Retrieve the most borrowed book:
SELECT books.title, COUNT(*) as borrow_count
FROM transactions
JOIN books ON transactions.book_id = books.book_id
GROUP BY books.title
ORDER BY borrow_count DESC
LIMIT 1;
```

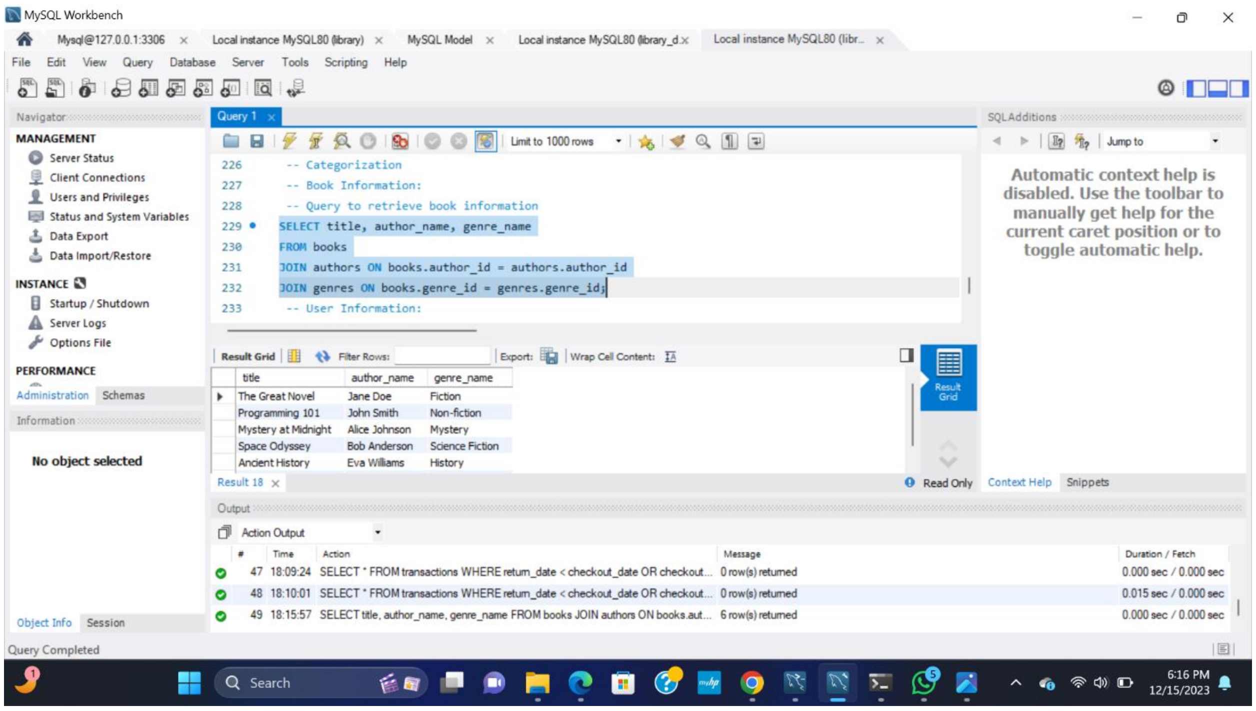
title	borrow_count
The Great Novel	1

```
#   Time      Action                                                                    Message                                                                              Duration / Fetch
26  17:25:48  DELETE FROM users WHERE user_id = 3                                       Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fail...  0.015 sec
27  17:26:34  SELECT books.title, COUNT(*) as borrow_count FROM transactions JOIN books O...  1 row(s) returned                                                                    0.000 sec / 0.000 sec
28  17:27:00  SELECT books.title, COUNT(*) as borrow_count FROM transactions JOIN books O...  1 row(s) returned                                                                    0.000 sec / 0.000 sec
```

Categorization:

In SQL, categorization refers to the process of organizing and grouping data based on specific criteria. This categorization, often paired with aggregate functions, divides the result set into distinct groups, allowing for calculations and summaries for each category. It is a powerful tool in SQL for analyzing and summarizing large datasets, commonly used in reporting and analytics to reveal patterns and trends within the data.

- The SQL query provided retrieves essential book information, categorizing details such as title, author name, and genre. This categorization aligns with the relational structure of a database, utilizing JOIN operations to seamlessly integrate data from the books, authors, and genres tables. By connecting book details with information about their authors and genres, the query ensures a comprehensive view of book-related data while maintaining data integrity. This approach acknowledges the importance of organized categorization within a relational database system, facilitating efficient retrieval of key book attributes in a cohesive manner.

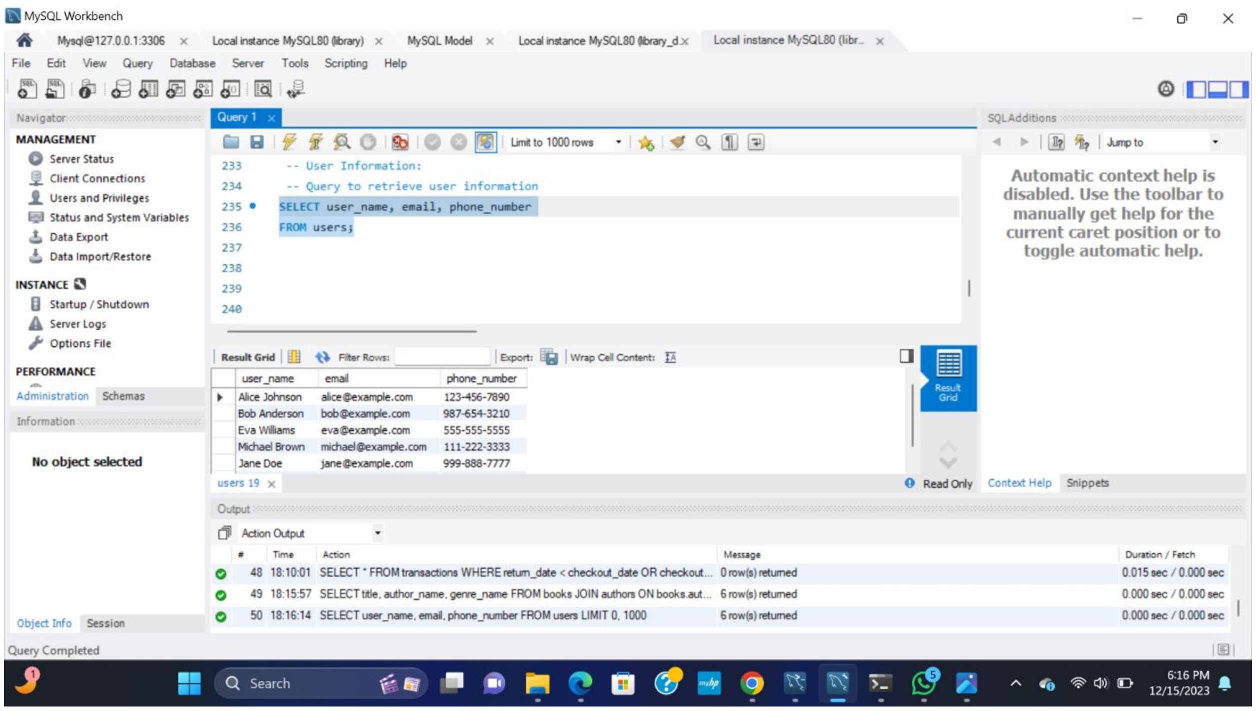


```sql
-- Categorization
-- Book Information:
-- Query to retrieve book information
SELECT title, author_name, genre_name
FROM books
JOIN authors ON books.author_id = authors.author_id
JOIN genres ON books.genre_id = genres.genre_id;
-- User Information:
```

title	author_name	genre_name
The Great Novel	Jane Doe	Fiction
Programming 101	John Smith	Non-fiction
Mystery at Midnight	Alice Johnson	Mystery
Space Odyssey	Bob Anderson	Science Fiction
Ancient History	Eva Williams	History

```
#   Time      Action                                                                   Message            Duration / Fetch
47  18:09:24  SELECT * FROM transactions WHERE return_date < checkout_date OR checkout...  0 row(s) returned  0.000 sec / 0.000 sec
48  18:10:01  SELECT * FROM transactions WHERE return_date < checkout_date OR checkout...  0 row(s) returned  0.015 sec / 0.000 sec
49  18:15:57  SELECT title, author_name, genre_name FROM books JOIN authors ON books.aut...  6 row(s) returned  0.000 sec / 0.000 sec
```

- The SQL query provided is focused on retrieving user information and aligns with a categorization structure centered around the "User Information" category. In this context, the category emphasizes essential details about users, including their 'user_name', 'email', and 'phone_number'. The query simplifies the user information retrieval process by selecting specific columns from the 'users' table, reflecting an organized approach to managing and presenting user-related data.

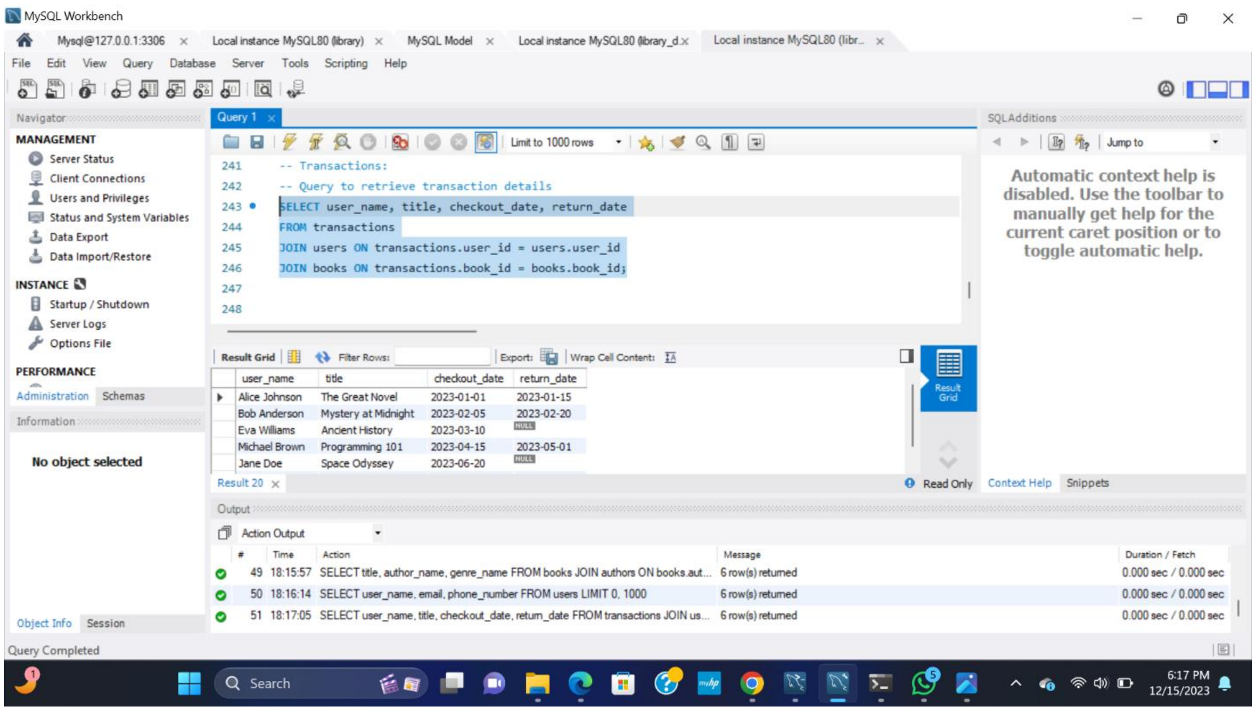


```sql
-- User Information:
-- Query to retrieve user information
SELECT user_name, email, phone_number
FROM users;
```

user_name	email	phone_number
Alice Johnson	alice@example.com	123-456-7890
Bob Anderson	bob@example.com	987-654-3210
Eva Williams	eva@example.com	555-555-5555
Michael Brown	michael@example.com	111-222-3333
Jane Doe	jane@example.com	999-888-7777

```
#   Time      Action                                                                  Message            Duration / Fetch
48  18:10:01  SELECT * FROM transactions WHERE return_date < checkout_date OR checkout...  0 row(s) returned  0.015 sec / 0.000 sec
49  18:15:57  SELECT title, author_name, genre_name FROM books JOIN authors ON books.aut... 6 row(s) returned  0.000 sec / 0.000 sec
50  18:16:14  SELECT user_name, email, phone_number FROM users LIMIT 0, 1000             6 row(s) returned  0.000 sec / 0.000 sec
```

- This query retrieves details about book transactions, including the user name, book title, checkout date, and return date.It involves a combination of JOIN operations to link information from multiple tables:
  transactions: Contains information about the transactions, including user_id and book_id.
  users: Contains information about users, including user_name.
  books: Contains information about books, including title.
  The JOIN conditions (transactions.user_id = users.user_id and transactions.book_id = books.book_id) specify how the data from different tables should be combined based on common columns



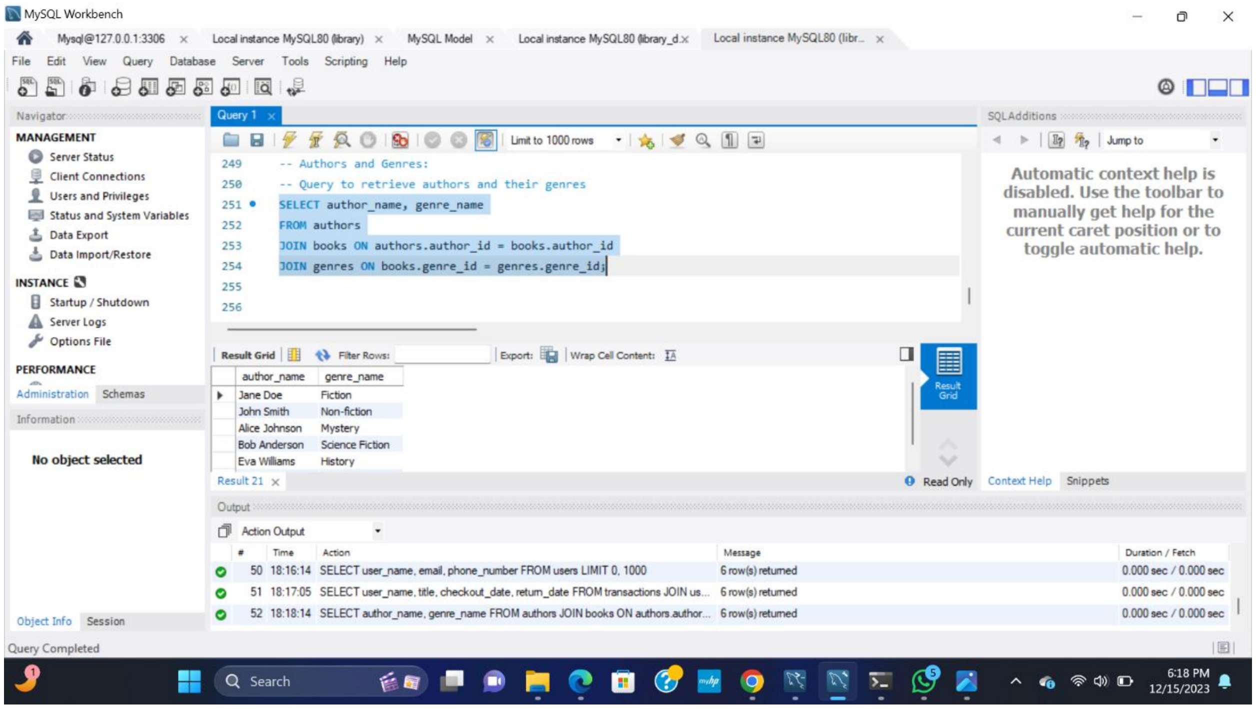
```
-- Transactions:
-- Query to retrieve transaction details
SELECT user_name, title, checkout_date, return_date
FROM transactions
JOIN users ON transactions.user_id = users.user_id
JOIN books ON transactions.book_id = books.book_id;
```

user_name	title	checkout_date	return_date
Alice Johnson	The Great Novel	2023-01-01	2023-01-15
Bob Anderson	Mystery at Midnight	2023-02-05	2023-02-20
Eva Williams	Ancient History	2023-03-10	NULL
Michael Brown	Programming 101	2023-04-15	2023-05-01
Jane Doe	Space Odyssey	2023-06-20	NULL

```
#   Time      Action                                                                      Message            Duration / Fetch
49  18:15:57  SELECT title, author_name, genre_name FROM books JOIN authors ON books.aut...  6 row(s) returned  0.000 sec / 0.000 sec
50  18:16:14  SELECT user_name, email, phone_number FROM users LIMIT 0, 1000               6 row(s) returned  0.000 sec / 0.000 sec
51  18:17:05  SELECT user_name, title, checkout_date, return_date FROM transactions JOIN us...  6 row(s) returned  0.000 sec / 0.000 sec
```

MySQL Workbench
Query 1
Limit to 1000 rows
Result Grid
Result 20
Action Output
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
No object selected
Query Completed
6:17 PM
12/15/2023

- This query retrieves information about authors and the genres associated with their books. It involves two JOIN operations:
  authors: Contains information about authors, including author_id and author_name.
  books: Contains information about books, including author_id (linking to authors) and genre_id.
  genres: Contains information about book genres, including genre_id and genre_name.
  The JOIN conditions (authors.author_id = books.author_id and books.genre_id = genres.genre_id) specify how the data from different tables should be combined based on common columns.



```
-- Authors and Genres:
-- Query to retrieve authors and their genres
SELECT author_name, genre_name
FROM authors
JOIN books ON authors.author_id = books.author_id
JOIN genres ON books.genre_id = genres.genre_id;
```

author_name	genre_name
Jane Doe	Fiction
John Smith	Non-fiction
Alice Johnson	Mystery
Bob Anderson	Science Fiction
Eva Williams	History

```
#   Time      Action                                                                  Message            Duration / Fetch
50  18:16:14  SELECT user_name, email, phone_number FROM users LIMIT 0, 1000          6 row(s) returned  0.000 sec / 0.000 sec
51  18:17:05  SELECT user_name, title, checkout_date, return_date FROM transactions JOIN us...  6 row(s) returned  0.000 sec / 0.000 sec
52  18:18:14  SELECT author_name, genre_name FROM authors JOIN books ON authors.author...  6 row(s) returned  0.000 sec / 0.000 sec
```

Query Completed

Triggers:

In a library management system, triggers can be used to automatically perform actions in response to certain events on the tables. Below are examples of triggers for the given database schema. We need to adapt them based on your specific requirements.

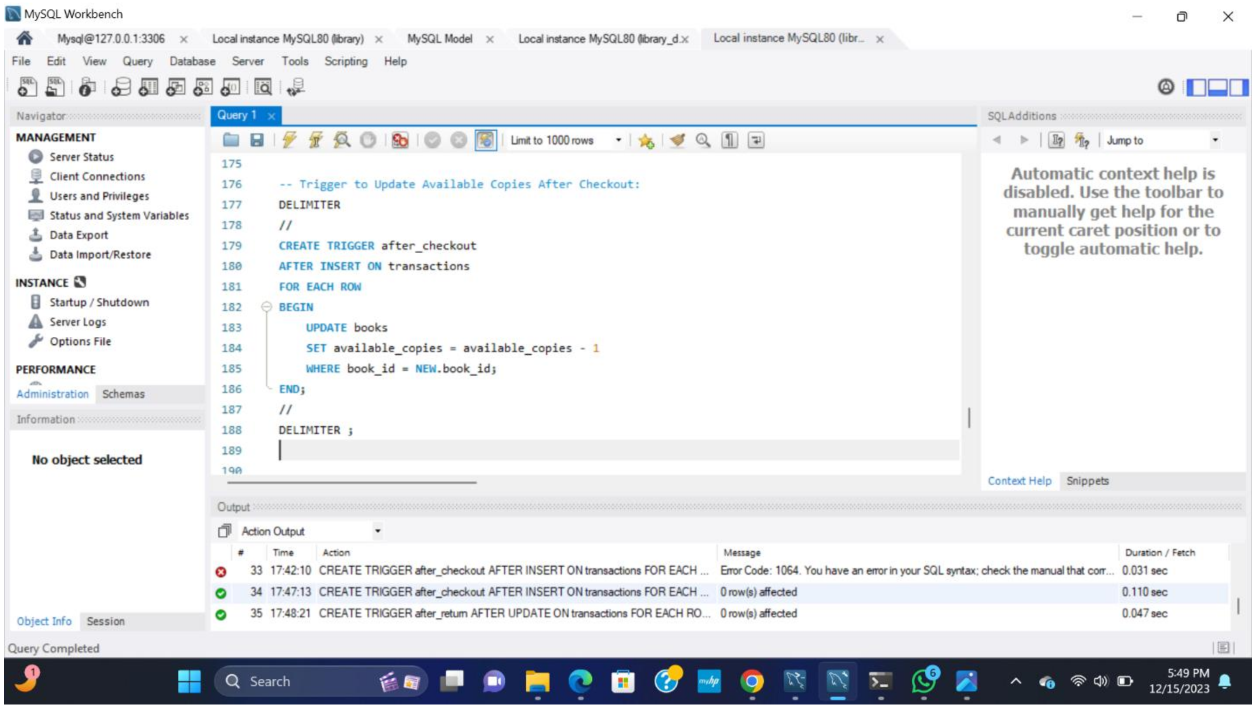
Trigger to Update Available Copies After Checkout:

This trigger updates the available_copies column in the books table when a book is checked out.

This trigger is set to execute AFTER INSERT on the Transactions table, meaning it will be triggered after a new row is inserted into the Transactions table.

The FOR EACH ROW clause specifies that the trigger should be executed once for each row affected by the trigger event.

The trigger checks if the CheckoutDate is not null (i.e., a book is being checked out) and then updates the Quantity in the Books table, reducing it by 1 for the specific book that is being checked out.



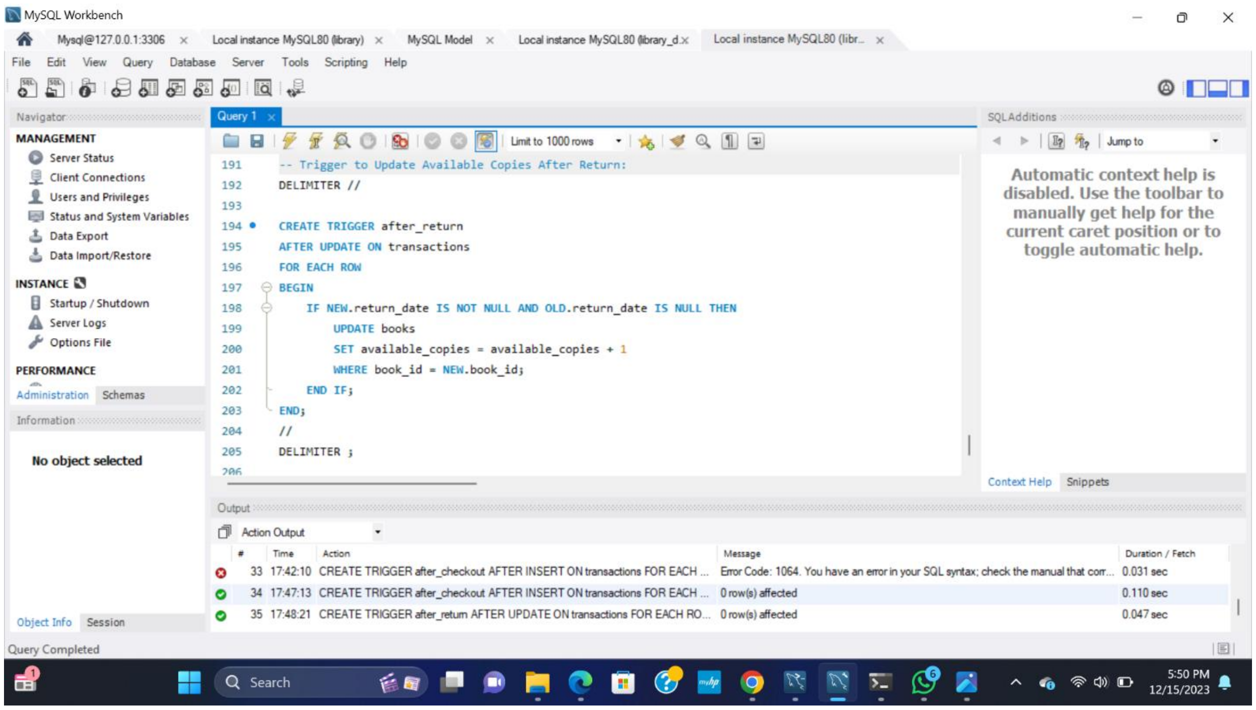
```sql
-- Trigger to Update Available Copies After Checkout:
DELIMITER
//
CREATE TRIGGER after_checkout
AFTER INSERT ON transactions
FOR EACH ROW
BEGIN
    UPDATE books
    SET available_copies = available_copies - 1
    WHERE book_id = NEW.book_id;
END;
//
DELIMITER ;
```

```
#   Time      Action                                                          Message                                                                      Duration / Fetch
33  17:42:10  CREATE TRIGGER after_checkout AFTER INSERT ON transactions FOR EACH ...  Error Code: 1064. You have an error in your SQL syntax; check the manual that corr...  0.031 sec
34  17:47:13  CREATE TRIGGER after_checkout AFTER INSERT ON transactions FOR EACH ...  0 row(s) affected                                                            0.110 sec
35  17:48:21  CREATE TRIGGER after_return AFTER UPDATE ON transactions FOR EACH RO...   0 row(s) affected                                                            0.047 sec
```

Query Completed

Trigger to Update Available Copies After Return:
This trigger updates the available_copies column in the books table when a book is returned.
This trigger is set to execute AFTER UPDATE on the Transactions table, meaning it will be triggered after a row is updated in the Transactions table.
Similar to the checkout trigger, it checks if the ReturnDate is not null (i.e., a book is being returned) and then updates the Quantity in the Books table, increasing it by 1 for the specific book that is being returned.

These triggers assume that the transactions table has columns book_id, checkout_date, and return_date.



```sql
-- Trigger to Update Available Copies After Return:
DELIMITER //

CREATE TRIGGER after_return
AFTER UPDATE ON transactions
FOR EACH ROW
BEGIN
    IF NEW.return_date IS NOT NULL AND OLD.return_date IS NULL THEN
        UPDATE books
        SET available_copies = available_copies + 1
        WHERE book_id = NEW.book_id;
    END IF;
END;
//
DELIMITER ;
```

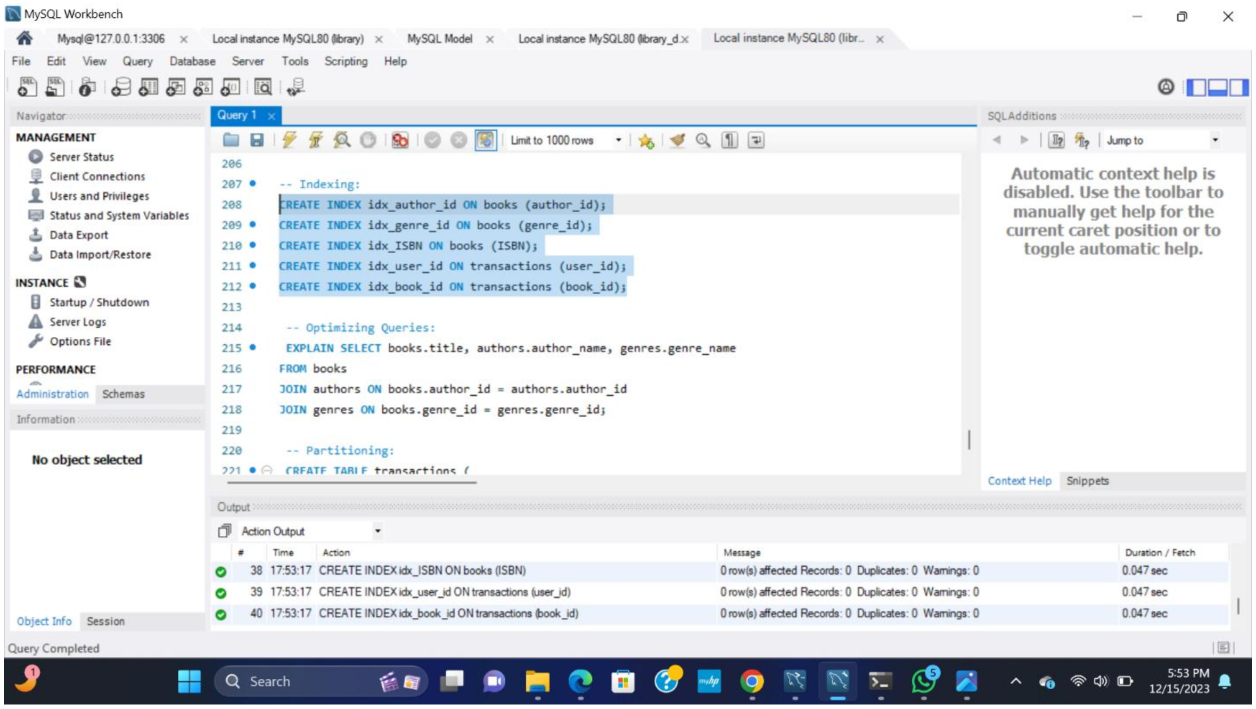
```
#   Time      Action                                                               Message                                                                      Duration / Fetch
33  17:42:10  CREATE TRIGGER after_checkout AFTER INSERT ON transactions FOR EACH ...  Error Code: 1064. You have an error in your SQL syntax; check the manual that corr...  0.031 sec
34  17:47:13  CREATE TRIGGER after_checkout AFTER INSERT ON transactions FOR EACH ...  0 row(s) affected                                                            0.110 sec
35  17:48:21  CREATE TRIGGER after_return AFTER UPDATE ON transactions FOR EACH RO...    0 row(s) affected                                                            0.047 sec
```

Performance Tuning:

Performance tuning for a database involves optimizing queries, indexing, and overall database design to enhance efficiency. Here are some general tips for performance tuning in the context of the library management system schema:

Indexing:

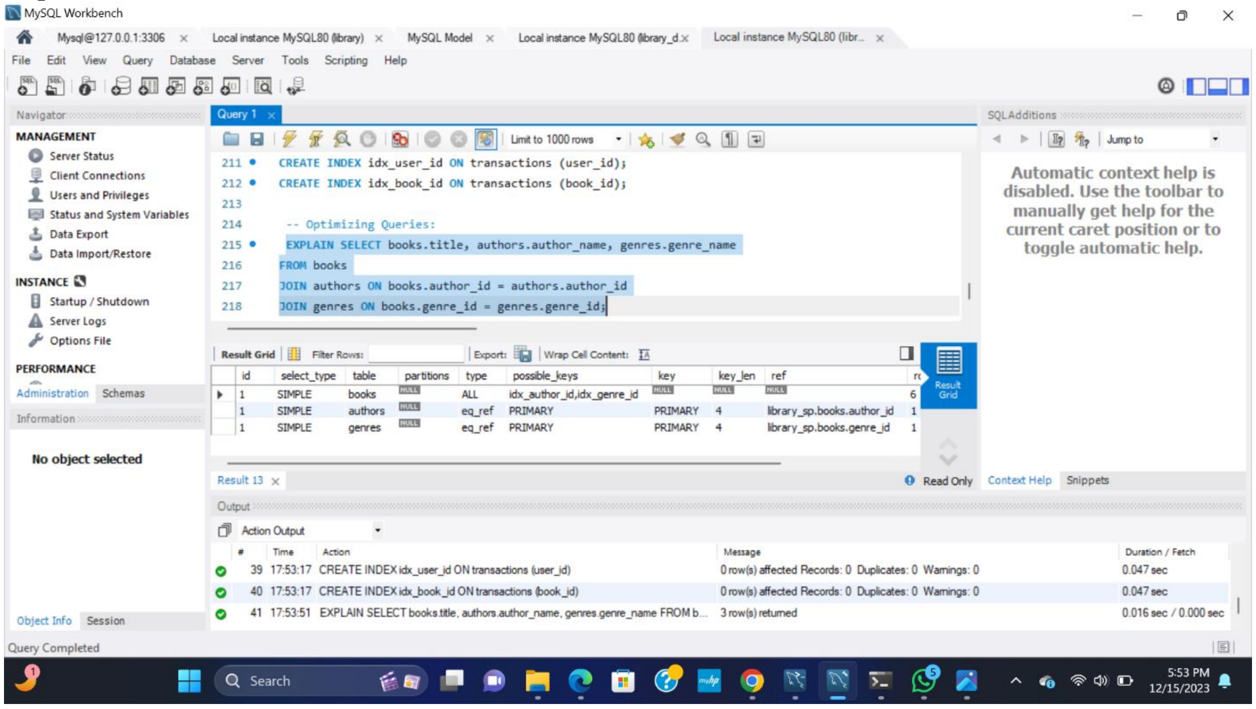
Ensure that columns used in search conditions, joins, and order by clauses are properly indexed. In the given schema, consider indexing columns like author_id, genre_id, ISBN, user_id, and book_id based on the queries you frequently run.



```
-- Indexing:
CREATE INDEX idx_author_id ON books (author_id);
CREATE INDEX idx_genre_id ON books (genre_id);
CREATE INDEX idx_ISBN ON books (ISBN);
CREATE INDEX idx_user_id ON transactions (user_id);
CREATE INDEX idx_book_id ON transactions (book_id);

 -- Optimizing Queries:
 EXPLAIN SELECT books.title, authors.author_name, genres.genre_name
FROM books
JOIN authors ON books.author_id = authors.author_id
JOIN genres ON books.genre_id = genres.genre_id;

 -- Partitioning:
 CREATE TABLE transactions (
```

```
#   Time      Action                                              Message                                                      Duration / Fetch
38  17:53:17  CREATE INDEX idx_ISBN ON books (ISBN)               0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0    0.047 sec
39  17:53:17  CREATE INDEX idx_user_id ON transactions (user_id)  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0    0.047 sec
40  17:53:17  CREATE INDEX idx_book_id ON transactions (book_id)  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0    0.047 sec
```

Optimizing Queries:
Review and optimize complex queries to ensure they are written efficiently. Consider using EXPLAIN to analyze query execution plans.
This helps you understand how the database is executing the query and identify areas for improvement.



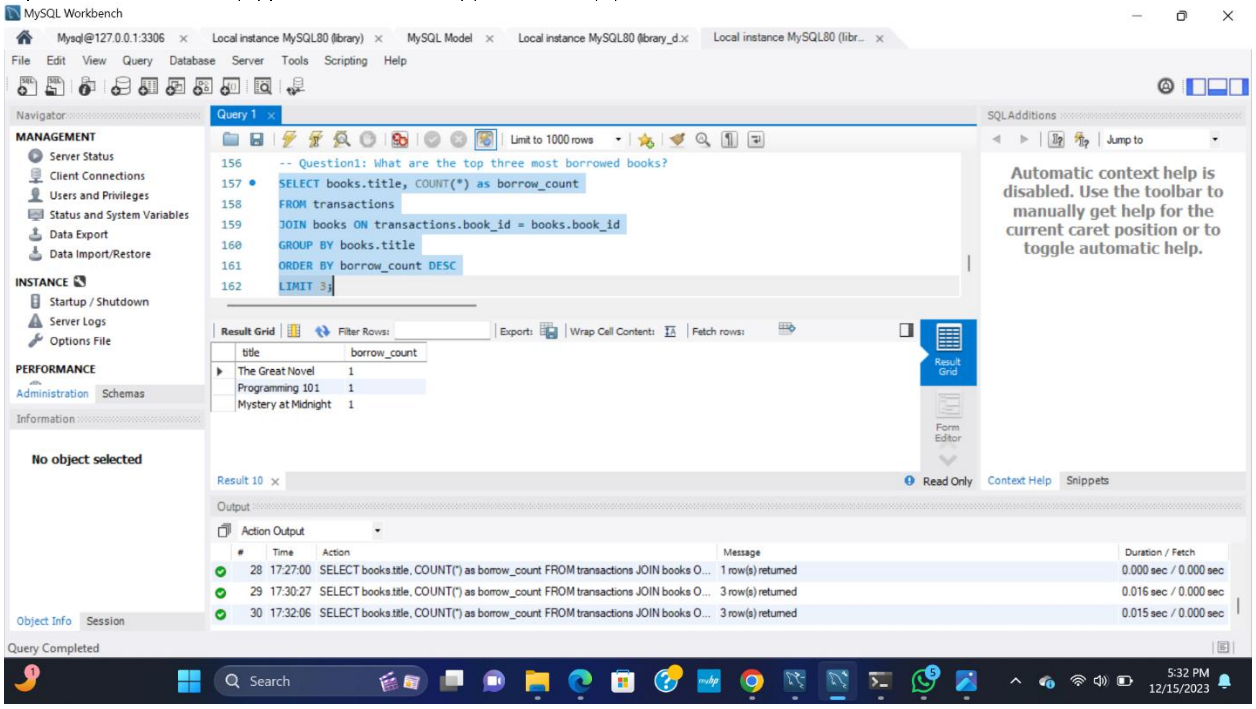
```
CREATE INDEX idx_user_id ON transactions (user_id);
CREATE INDEX idx_book_id ON transactions (book_id);

 -- Optimizing Queries:
 EXPLAIN SELECT books.title, authors.author_name, genres.genre_name
FROM books
JOIN authors ON books.author_id = authors.author_id
JOIN genres ON books.genre_id = genres.genre_id;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	r
1	SIMPLE	books	NULL	ALL	idx_author_id,idx_genre_id	NULL	NULL	NULL	6
1	SIMPLE	authors	NULL	eq_ref	PRIMARY	PRIMARY	4	library_sp.books.author_id	1
1	SIMPLE	genres	NULL	eq_ref	PRIMARY	PRIMARY	4	library_sp.books.genre_id	1

```
#   Time      Action                                                                  Message                                                      Duration / Fetch
39  17:53:17  CREATE INDEX idx_user_id ON transactions (user_id)                      0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0    0.047 sec
40  17:53:17  CREATE INDEX idx_book_id ON transactions (book_id)                      0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0    0.047 sec
41  17:53:51  EXPLAIN SELECT books.title, authors.author_name, genres.genre_name FROM b...  3 row(s) returned                                       0.016 sec / 0.000 sec
```

Questions and Solutions:

1. What are the top three most borrowed books?
TRC Query:
{⟨b.title, COUNT(*)⟩ | Transaction(t) ∧ Book(b) ∧ t.book_id = b.book_id }

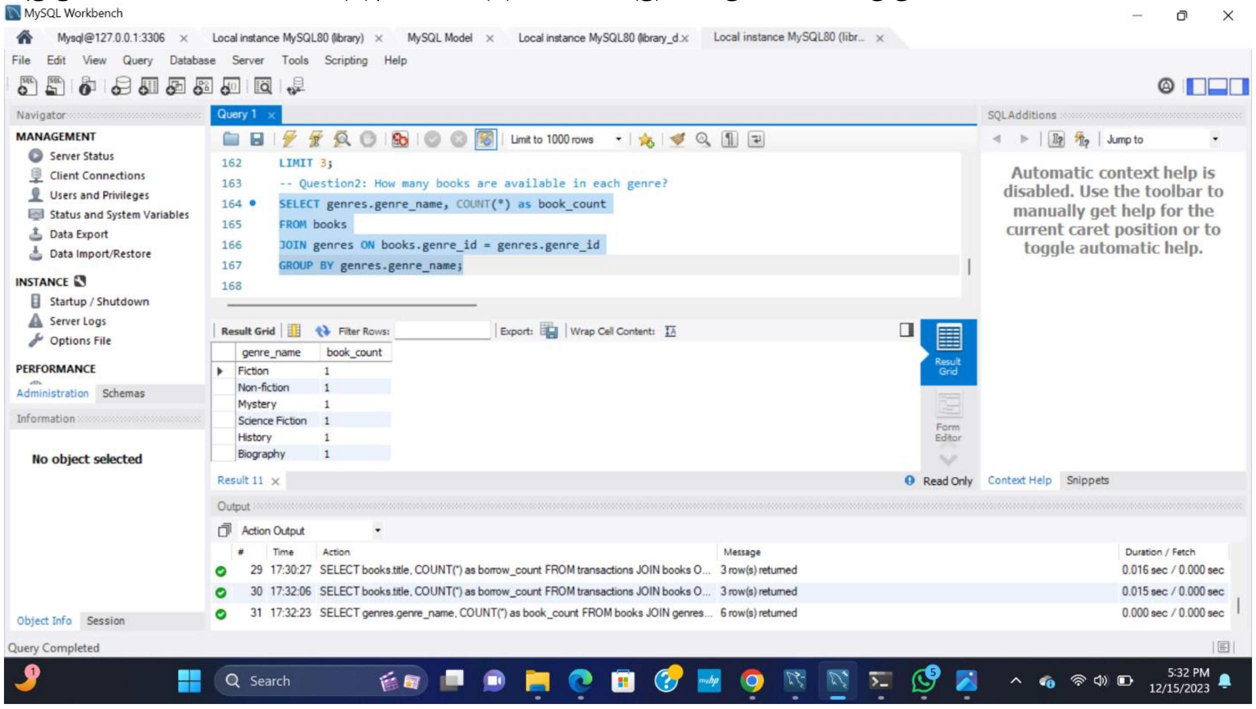


```
-- Question1: What are the top three most borrowed books?
SELECT books.title, COUNT(*) as borrow_count
FROM transactions
JOIN books ON transactions.book_id = books.book_id
GROUP BY books.title
ORDER BY borrow_count DESC
LIMIT 3;
```

title	borrow_count
The Great Novel	1
Programming 101	1
Mystery at Midnight	1

```
#   Time      Action                                                                        Message             Duration / Fetch
28  17:27:00  SELECT books.title, COUNT(*) as borrow_count FROM transactions JOIN books O...  1 row(s) returned   0.000 sec / 0.000 sec
29  17:30:27  SELECT books.title, COUNT(*) as borrow_count FROM transactions JOIN books O...  3 row(s) returned   0.016 sec / 0.000 sec
30  17:32:06  SELECT books.title, COUNT(*) as borrow_count FROM transactions JOIN books O...  3 row(s) returned   0.015 sec / 0.000 sec
```

2.How many books are available in each genre?
TRC Query:
{⟨g.genre_name, COUNT(*)⟩ | Book(b) ∧ Genre(g) ∧ b.genre_id = g.genre_id }

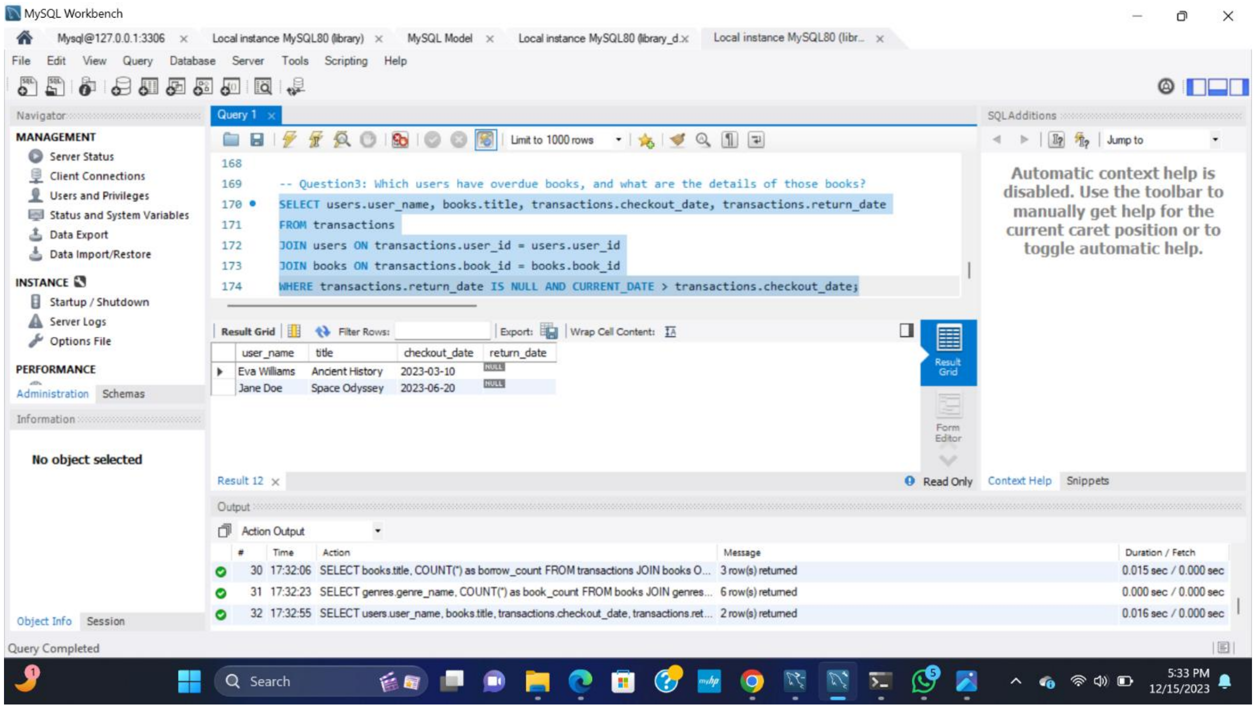


```
LIMIT 3;
-- Question2: How many books are available in each genre?
SELECT genres.genre_name, COUNT(*) as book_count
FROM books
JOIN genres ON books.genre_id = genres.genre_id
GROUP BY genres.genre_name;
```

genre_name	book_count
Fiction	1
Non-fiction	1
Mystery	1
Science Fiction	1
History	1
Biography	1

```
#   Time      Action                                                                        Message             Duration / Fetch
29  17:30:27  SELECT books.title, COUNT(*) as borrow_count FROM transactions JOIN books O...  3 row(s) returned   0.016 sec / 0.000 sec
30  17:32:06  SELECT books.title, COUNT(*) as borrow_count FROM transactions JOIN books O...  3 row(s) returned   0.015 sec / 0.000 sec
31  17:32:23  SELECT genres.genre_name, COUNT(*) as book_count FROM books JOIN genres...     6 row(s) returned   0.000 sec / 0.000 sec
```

3.Which users have overdue books, and what are the details of those books?
TRC Query:
{u.user_name, b.title, t.checkout_date, t.return_date) | Transaction(t) ^
User (u) ^ Book (b)
∧ t.user_id = u.user_id ∧ t.book id = b.book id ∧ t.return_date =
NULL ∧ CURRENT_DATE > t.checkout_date }



```
-- Question3: Which users have overdue books, and what are the details of those books?
SELECT users.user_name, books.title, transactions.checkout_date, transactions.return_date
FROM transactions
JOIN users ON transactions.user_id = users.user_id
JOIN books ON transactions.book_id = books.book_id
WHERE transactions.return_date IS NULL AND CURRENT_DATE > transactions.checkout_date;
```

user_name	title	checkout_date	return_date
Eva Williams	Ancient History	2023-03-10	NULL
Jane Doe	Space Odyssey	2023-06-20	NULL

```
#   Time      Action                                                                          Message            Duration / Fetch
30  17:32:06  SELECT books.title, COUNT(*) as borrow_count FROM transactions JOIN books O...   3 row(s) returned  0.015 sec / 0.000 sec
31  17:32:23  SELECT genres.genre_name, COUNT(*) as book_count FROM books JOIN genres...      6 row(s) returned  0.000 sec / 0.000 sec
32  17:32:55  SELECT users.user_name, books.title, transactions.checkout_date, transactions.ret...  2 row(s) returned  0.016 sec / 0.000 sec
```

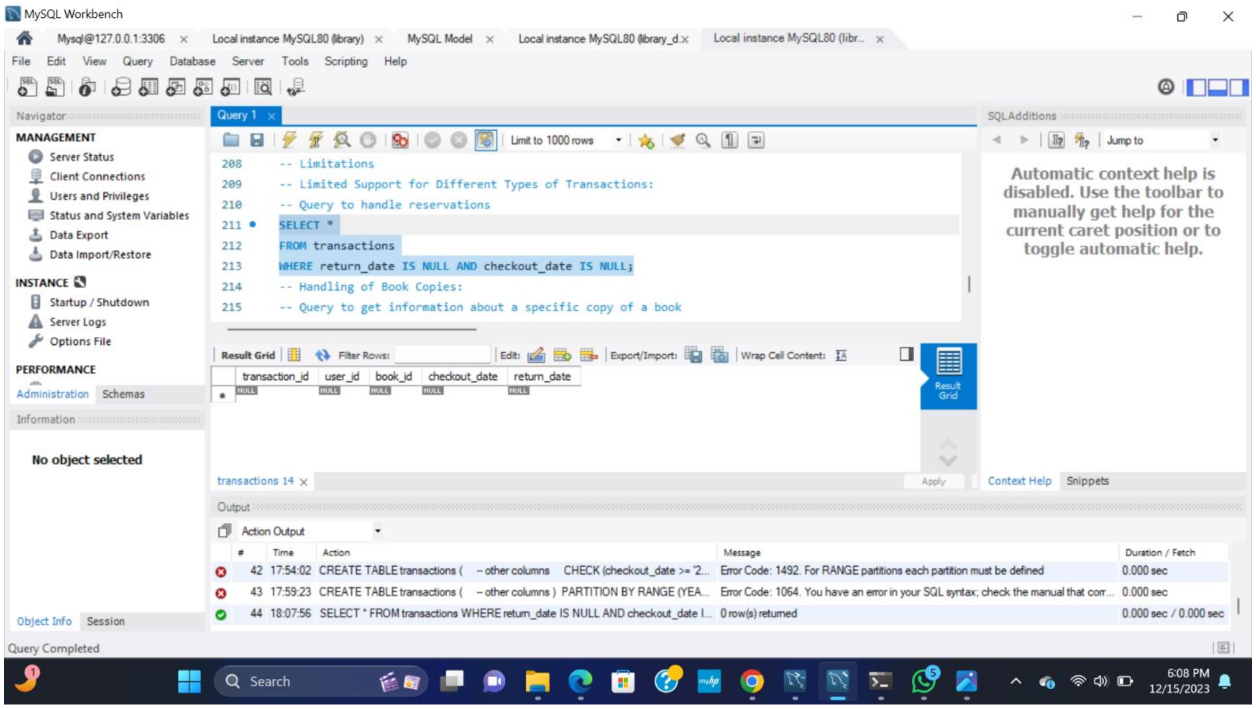
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Query Completed

Limitations:

While the provided database schema for a library management system is functional, it may have some limitations based on specific requirements and use cases. Here are a few potential limitations and examples of SQL queries that could highlight them:

Limited Support for Different Types of Transactions:
The current schema may not fully support complex transaction scenarios, such as multiple copies of the same book or handling different types of transactions (e.g., reservations, renewals)



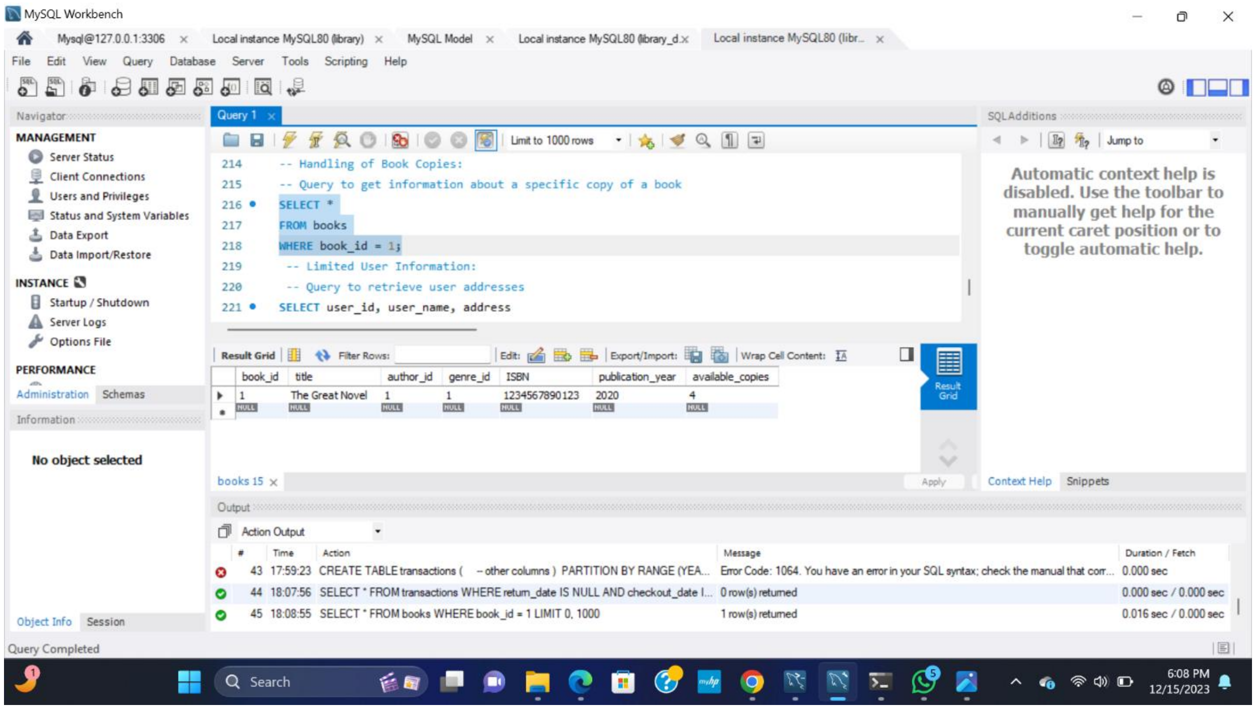
```sql
-- Limitations
-- Limited Support for Different Types of Transactions:
-- Query to handle reservations
SELECT *
FROM transactions
WHERE return_date IS NULL AND checkout_date IS NULL;
-- Handling of Book Copies:
-- Query to get information about a specific copy of a book
```

transaction_id	user_id	book_id	checkout_date	return_date
NULL	NULL	NULL	NULL	NULL

```
#   Time      Action                                                                  Message                                                                    Duration / Fetch
42  17:54:02  CREATE TABLE transactions (   -- other columns   CHECK (checkout_date >= '2...  Error Code: 1492. For RANGE partitions each partition must be defined        0.000 sec
43  17:59:23  CREATE TABLE transactions (   -- other columns ) PARTITION BY RANGE (YEA...     Error Code: 1064. You have an error in your SQL syntax; check the manual that corr...  0.000 sec
44  18:07:56  SELECT * FROM transactions WHERE return_date IS NULL AND checkout_date I...  0 row(s) returned                                                          0.000 sec / 0.000 sec
```

Handling of Book Copies:

If you want to support multiple copies of the same book, the current schema may not differentiate between individual copies. It treats all copies of a book as a single entity.



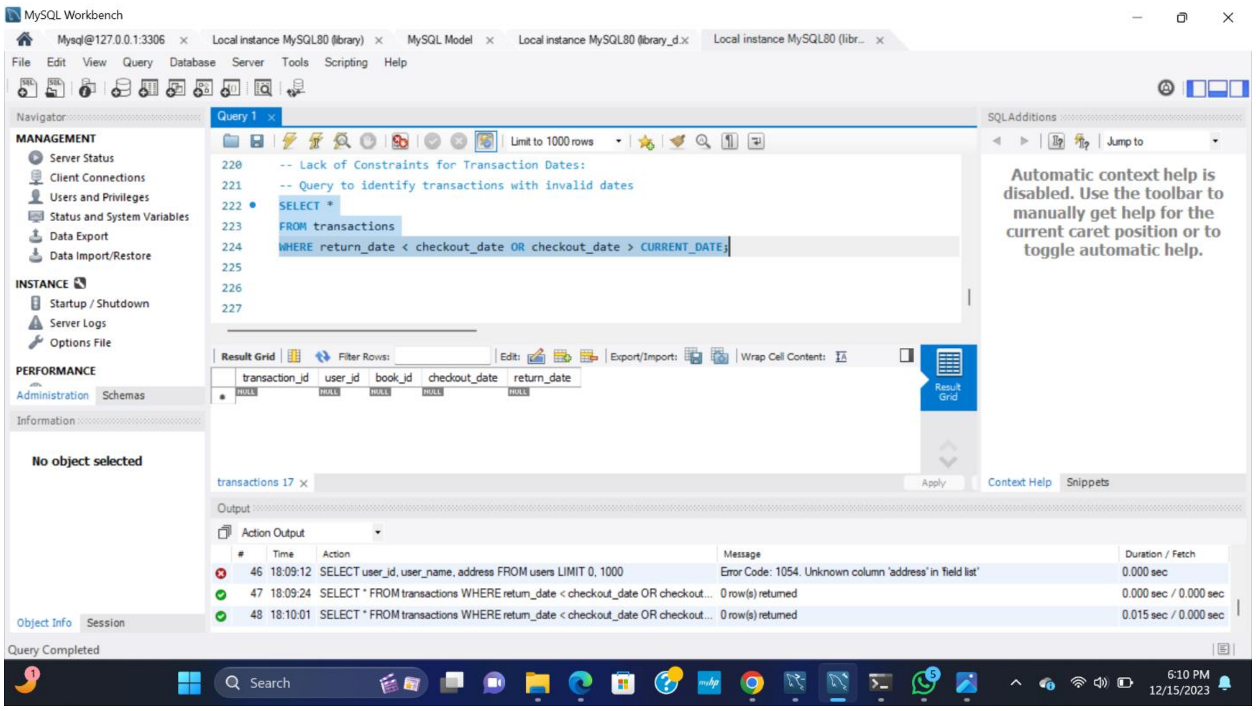
```sql
-- Handling of Book Copies:
-- Query to get information about a specific copy of a book
SELECT *
FROM books
WHERE book_id = 1;
 -- Limited User Information:
 -- Query to retrieve user addresses
SELECT user_id, user_name, address
```

book_id	title	author_id	genre_id	ISBN	publication_year	available_copies
1	The Great Novel	1	1	1234567890123	2020	4
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
#   Time      Action                                                                  Message                                                                    Duration / Fetch
43  17:59:23  CREATE TABLE transactions (   -- other columns ) PARTITION BY RANGE (YEA...     Error Code: 1064. You have an error in your SQL syntax; check the manual that corr...  0.000 sec
44  18:07:56  SELECT * FROM transactions WHERE return_date IS NULL AND checkout_date I...  0 row(s) returned                                                          0.000 sec / 0.000 sec
45  18:08:55  SELECT * FROM books WHERE book_id = 1 LIMIT 0, 1000                          1 row(s) returned                                                          0.016 sec / 0.000 sec
```

Lack of Constraints for Transaction Dates:

The current schema does not enforce constraints on transaction dates, which could lead to inconsistencies in the data.



```
-- Lack of Constraints for Transaction Dates:
-- Query to identify transactions with invalid dates
SELECT *
FROM transactions
WHERE return_date < checkout_date OR checkout_date > CURRENT_DATE;
```

transaction_id	user_id	book_id	checkout_date	return_date
NULL	NULL	NULL	NULL	NULL

```
#   Time      Action                                                               Message                                                   Duration / Fetch
46  18:09:12  SELECT user_id, user_name, address FROM users LIMIT 0, 1000          Error Code: 1054. Unknown column 'address' in 'field list'  0.000 sec
47  18:09:24  SELECT * FROM transactions WHERE return_date < checkout_date OR checkout...  0 row(s) returned                              0.000 sec / 0.000 sec
48  18:10:01  SELECT * FROM transactions WHERE return_date < checkout_date OR checkout...  0 row(s) returned                              0.015 sec / 0.000 sec
```

In summary, the limitations in the provided SQL queries include a potential oversimplification of transaction types, insufficient tracking of individual book copies, and a lack of comprehensive constraints and logic for handling transaction dates. Addressing these limitations would contribute to a more robust and adaptable database system capable of handling a wider range of scenarios in a library management context.

Conclusion:

In summary, the Library Management System (LMS) project provides an effective solution for library organization and management. The structured database, powered by SQL queries, ensures accurate data collection, reporting, and performance optimization. Key principles of data integrity, maintenance, and scalability are prioritized, enhancing the system's reliability and adaptability. With a user-friendly design, the LMS project serves as a dynamic foundation for modern libraries, promising efficient information management and responsive user service, setting the stage for ongoing enhancements.

Github:

The complete project details can be found in my github account by following this link,

Library-Management-System