**Title: Task 1 - Library Management System (using SQL)**

**Author: Snehal Yerkuntwar**

**1. Introduction**

- Design and develop a Library Management System using SQL. The project should involve three tables: Books, Members, BorrowingRecords. The system will manage book inventories, member details, and borrowing transactions.

**2. Features**

The project will include the following tasks:

Database Creation: a) Create Books table with columns BOOK\_ID, TITLE, AUTHOR, GENRE, YEAR\_PUBLISHED, AVAILABLE\_COPIES.

b) Create Members table with columns MEMBER\_ID, NAME, EMAIL, PHONE\_NO, ADDRESS, MEMBERSHIP\_DATE.

c) Create BorrowingRecords table with columns BORROW\_ID, MEMBER\_ID, BOOK\_ID, BORROW\_DATE, RETURN\_DATE. Set foreign key constraints linking MEMBER\_ID to Members and BOOK\_ID to Books.

**3. System Requirements**

- Software: MySQL with myCompiler is used for code and screen shots to validate the results.

**4. Task Description :**

Data Creation: Insert sample data into all three tables.

**A. Information Retrieval:**

a) Retrieve a list of books currently borrowed by a specific member.

b) Find members who have overdue books (borrowed more than 30 days ago, not returned). c) Retrieve books by genre along with the count of available copies.

d) Find the most borrowed book(s) overall.

e) Retrieve members who have borrowed books from at least three different genres.

**B. Reporting and Analytics:**

a) Calculate the total number of books borrowed per month.

b) Find the top three most active members based on the number of books borrowed.

c) Retrieve authors whose books have been borrowed at least 10 times.

d) Identify members who have never borrowed a book.

**Main Code: Task wise begins from here**

1. **Database creation (DDL)**

**-- Create Books table**

CREATE TABLE Books (

BOOK\_ID INT PRIMARY KEY,

TITLE VARCHAR(255) NOT NULL,

AUTHOR VARCHAR(255) NOT NULL,

GENRE VARCHAR(50),

YEAR\_PUBLISHED INT,

AVAILABLE\_COPIES INT NOT NULL

);

**-- Create Members table**

CREATE TABLE Members (

MEMBER\_ID INT PRIMARY KEY,

NAME VARCHAR(255) NOT NULL,

EMAIL VARCHAR(255) UNIQUE,

PHONE\_NO VARCHAR(20),

ADDRESS VARCHAR(255),

MEMBERSHIP\_DATE DATE NOT NULL

);

**-- Create BorrowingRecords table**

CREATE TABLE BorrowingRecords (

BORROW\_ID INT PRIMARY KEY,

MEMBER\_ID INT NOT NULL,

BOOK\_ID INT NOT NULL,

BORROW\_DATE DATE NOT NULL,

RETURN\_DATE DATE, -- Can be NULL if the book is not yet returned

FOREIGN KEY (MEMBER\_ID) REFERENCES Members(MEMBER\_ID),

FOREIGN KEY (BOOK\_ID) REFERENCES Books(BOOK\_ID)

);

**b. Insert sample data into all three tables** - **Data creation (DML)**

-- Insert sample data into Books table

INSERT INTO Books (BOOK\_ID, TITLE, AUTHOR, GENRE, YEAR\_PUBLISHED, AVAILABLE\_COPIES) VALUES

(1, 'The Lord of the Rings', 'J.R.R. Tolkien', 'Fantasy', 1954, 5),

(2, 'Pride and Prejudice', 'Jane Austen', 'Romance', 1813, 3),

(3, '1984', 'George Orwell', 'Dystopian', 1949, 2),

(4, 'The Hobbit', 'J.R.R. Tolkien', 'Fantasy', 1937, 4),

(5, 'To Kill a Mockingbird', 'Harper Lee', 'Fiction', 1960, 6),

(6, 'Dune', 'Frank Herbert', 'Science Fiction', 1965, 3),

(7, 'Foundation', 'Isaac Asimov', 'Science Fiction', 1951, 2),

(8, 'Sense and Sensibility', 'Jane Austen', 'Romance', 1811, 4),

(9, 'Animal Farm', 'George Orwell', 'Political Satire', 1945, 1);

-- Insert sample data into Members table

INSERT INTO Members (MEMBER\_ID, NAME, EMAIL, PHONE\_NO, ADDRESS, MEMBERSHIP\_DATE) VALUES

(101, 'Alice Smith', 'alice.smith@example.com', '123-456-7890', '123 Main St', '2023-01-15'),

(102, 'Bob Johnson', 'bob.johnson@example.com', '987-654-3210', '456 Oak Ave', '2022-03-20'),

(103, 'Charlie Brown', 'charlie.brown@example.com', '555-123-4567', '789 Pine Ln', '2023-06-01'),

(104, 'Diana Prince', 'diana.prince@example.com', '111-222-3333', '101 Themyscira', '2024-02-10'),

(105, 'Eve Adams', 'eve.adams@example.com', '444-555-6666', '202 Cedar Blvd', '2023-09-05');

-- Insert sample data into BorrowingRecords table

INSERT INTO BorrowingRecords (BORROW\_ID, MEMBER\_ID, BOOK\_ID, BORROW\_DATE, RETURN\_DATE) VALUES

(1, 101, 1, '2024-07-01', NULL), -- Alice borrowed Lord of the Rings, not returned

(2, 102, 3, '2024-06-20', '2024-07-15'), -- Bob borrowed 1984, returned

(3, 101, 2, '2024-07-10', NULL), -- Alice borrowed Pride and Prejudice, not returned

(4, 103, 4, '2024-07-05', '2024-07-25'), -- Charlie borrowed The Hobbit, returned

(5, 102, 5, '2024-06-10', NULL), -- Bob borrowed To Kill a Mockingbird, not returned

(6, 104, 6, '2024-07-12', '2024-07-30'), -- Diana borrowed Dune, returned

(7, 101, 7, '2024-07-03', '2024-07-28'), -- Alice borrowed Foundation, returned

(8, 103, 8, '2024-06-15', NULL), -- Charlie borrowed Sense and Sensibility, not returned

(9, 105, 9, '2024-07-08', '2024-07-22'), -- Eve borrowed Animal Farm, returned

(10, 101, 4, '2024-07-20', NULL), -- Alice borrowed The Hobbit again, not returned

(11, 102, 1, '2024-08-01', NULL); -- Bob borrowed Lord of the Rings, not returned

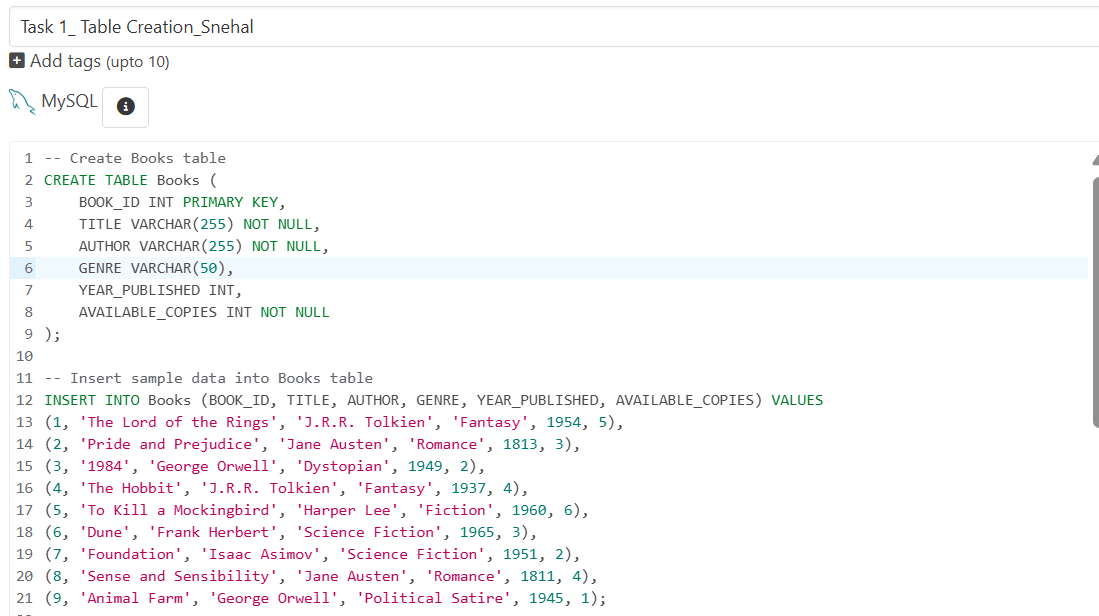
**-- Overdue books for testing (Adding Data for Overdue Books)**

INSERT INTO BorrowingRecords (BORROW\_ID, MEMBER\_ID, BOOK\_ID, BORROW\_DATE, RETURN\_DATE) VALUES

(12, 104, 2, '2024-06-01', NULL), -- Diana has an overdue book

**Step wise Database and Data Creation Screenshots from myCompiler**

1. Books Table and Sample data with Prompt results:

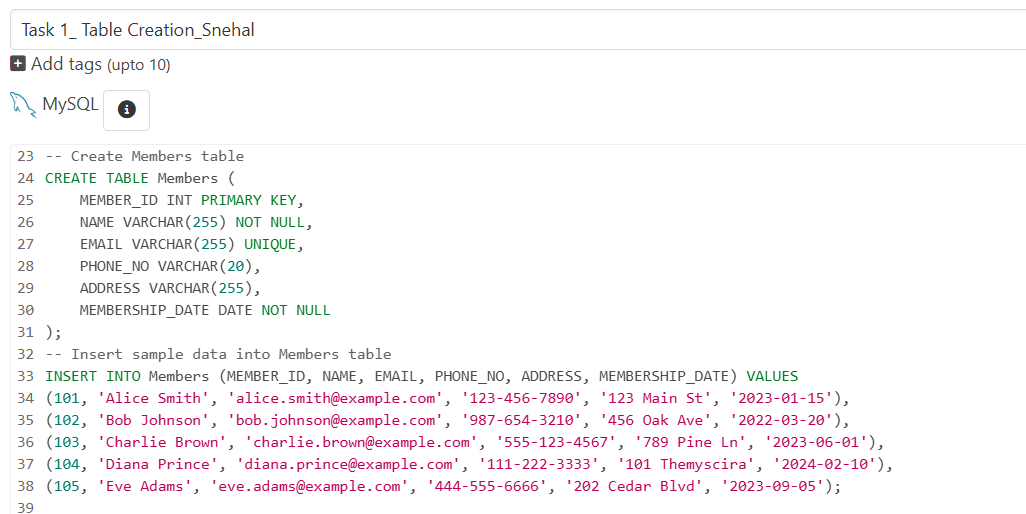


--- Sample data for Books Table is inserted

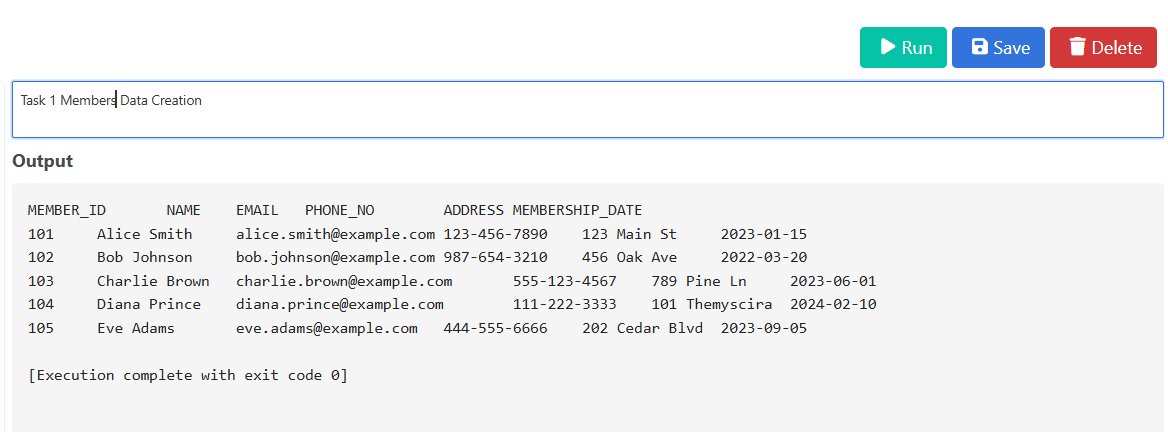
Command Prompt Screenshot for Books data with No Error



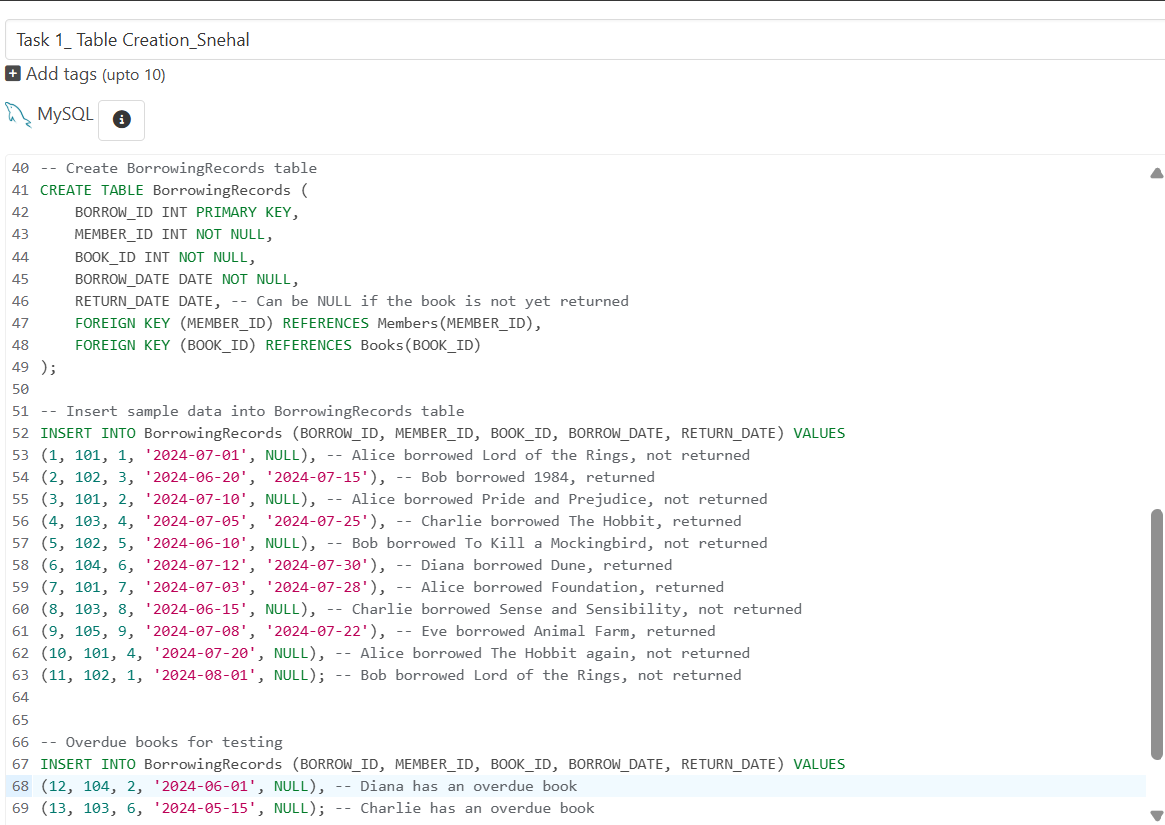
1. Member Table creation and sample data insertion



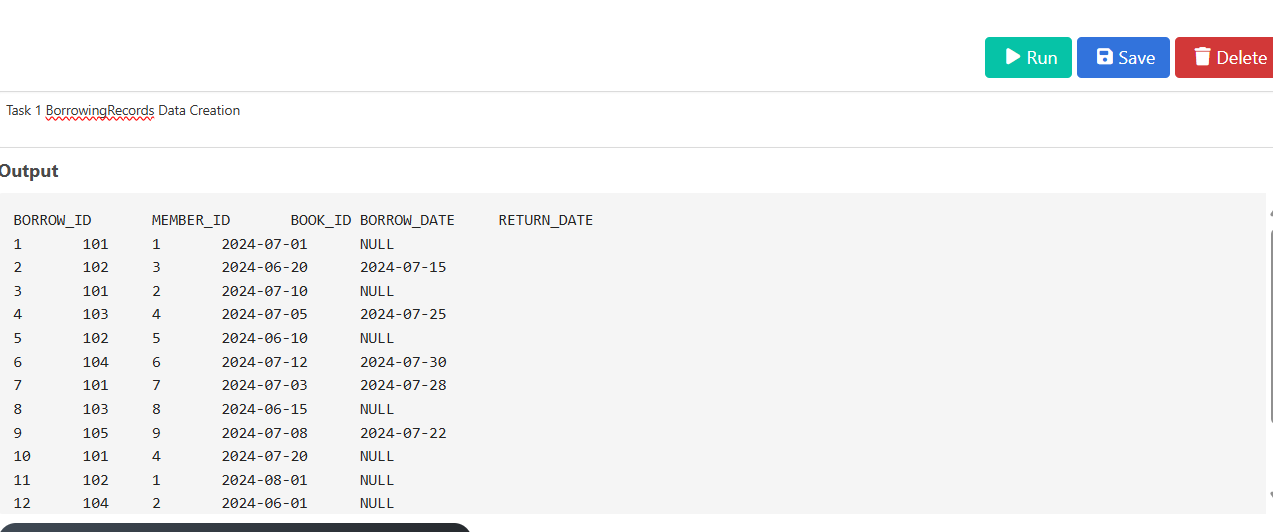
Result screenshot for Members

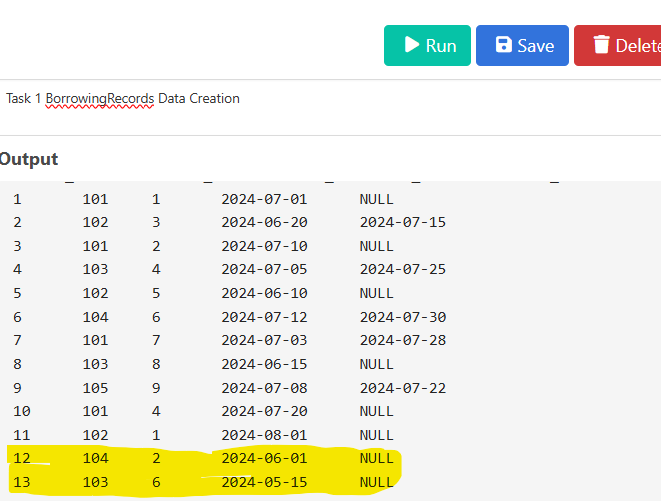


1. BorrowingRecords Table and Sample data insertion with Overdue books for Testing



Screenshot for BorrowingRecords with Overdue Books for Testing Prompt when Run





--- Here we see that MemberID 104, 103 have Overdue book data has been inserted

1. **Information Retrieval:**
2. **Retrieve a list of books currently borrowed by a specific member (e.g., Member ID 101)**

SELECT

M.NAME AS MemberName,

B.TITLE,

B.AUTHOR,

BR.BORROW\_DATE

FROM

Members M

JOIN

BorrowingRecords BR ON M.MEMBER\_ID = BR.MEMBER\_ID

JOIN

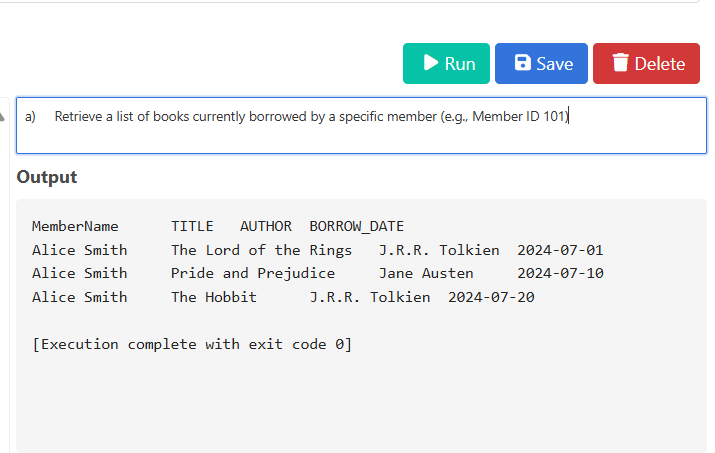
Books B ON BR.BOOK\_ID = B.BOOK\_ID

WHERE

M.MEMBER\_ID = 101 AND BR.RETURN\_DATE IS NULL;

---- This query identifies List of books currently borrowed by MemberID 101 named Alice. We Use JOIN to combine attributes from all three tables created to get above results.





1. **Find members who have overdue books (borrowed more than 30 days ago, not returned)**

SELECT m.NAME, b.TITLE, br.BORROW\_DATE

FROM Members m

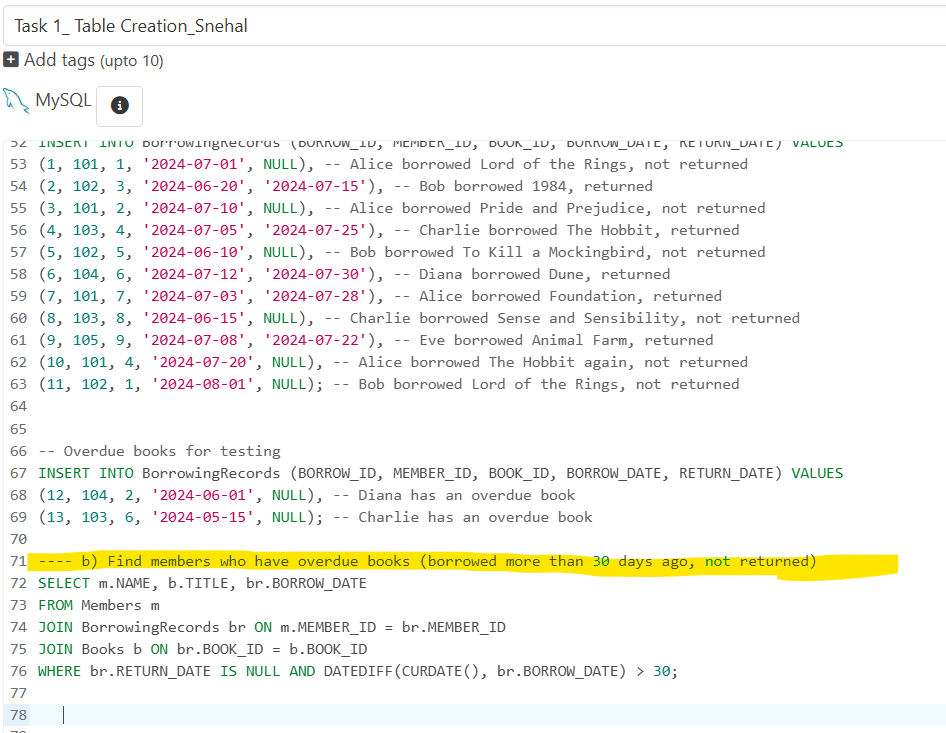
JOIN BorrowingRecords br ON m.MEMBER\_ID = br.MEMBER\_ID

JOIN Books b ON br.BOOK\_ID = b.BOOK\_ID

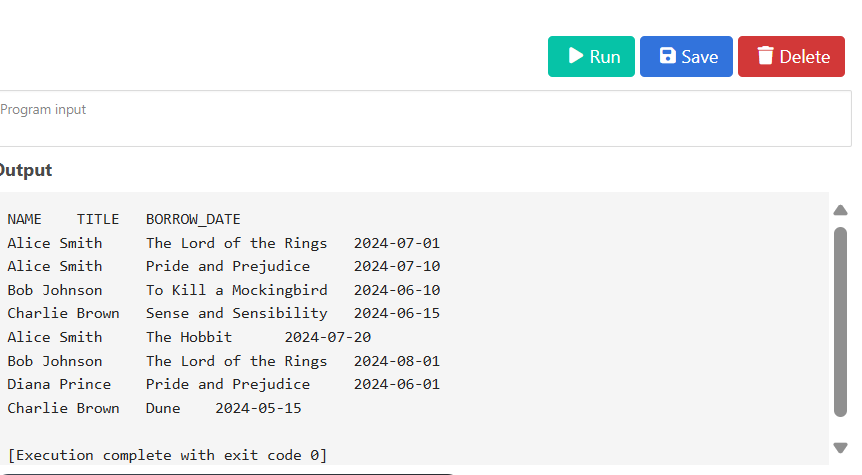
WHERE br.RETURN\_DATE IS NULL AND DATEDIFF(CURDATE(), br.BORROW\_DATE) > 30;

---- This query identifies members who have books that are currently overdue, where overdue is defined as being borrowed more than 30 days ago and not yet returned.

---- Query execution Screenshot and output prompt as below



Output Screenshot



--- Using DATEIFF function to get difference in Borrowing date and Return Date with Current() date that retrieves data on Overdue books with Member Name and Books Title.

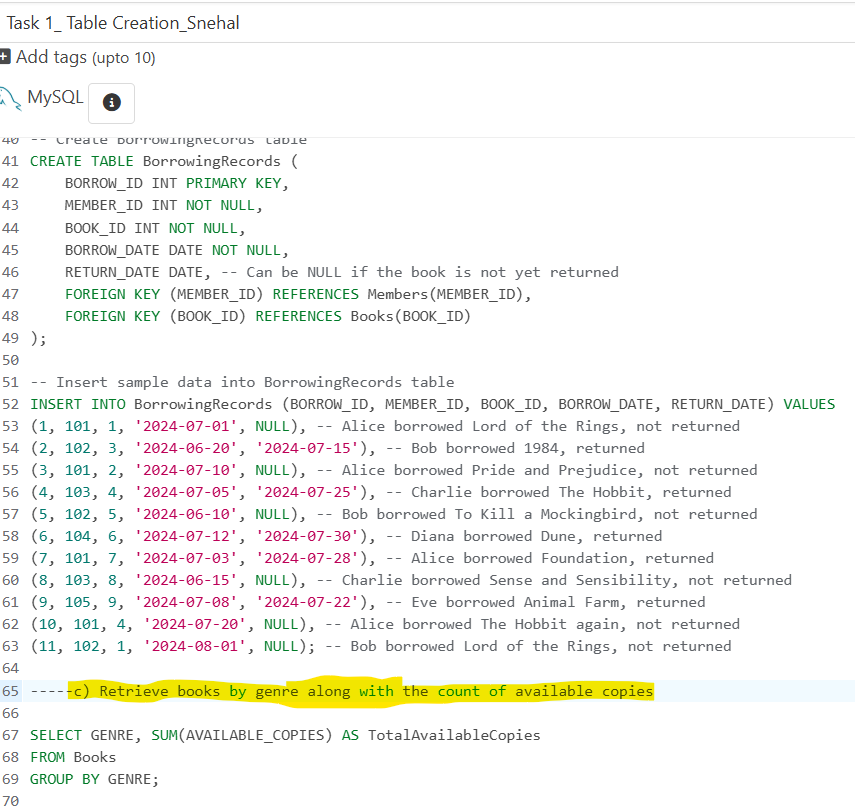
1. **Retrieve books by genre along with the count of available copies**

SELECT GENRE, SUM(AVAILABLE\_COPIES) AS TotalAvailableCopies

FROM Books

GROUP BY GENRE;

**-----** This query will group all the books together based on their genre and return the sum of the available copies for each genre.

****

****

--- The query retrieves Genre and Total Available Copies. Used SUM() function to achieve this output.

Count() could also be included but this has delivery as accurate result as asked.

1. **Find the most borrowed book(s) overall**

SELECT

B.TITLE,

B.AUTHOR,

COUNT(BR.BOOK\_ID) AS BorrowCount

FROM

Books B

JOIN

BorrowingRecords BR ON B.BOOK\_ID = BR.BOOK\_ID

GROUP BY

B.BOOK\_ID, B.TITLE, B.AUTHOR

ORDER BY

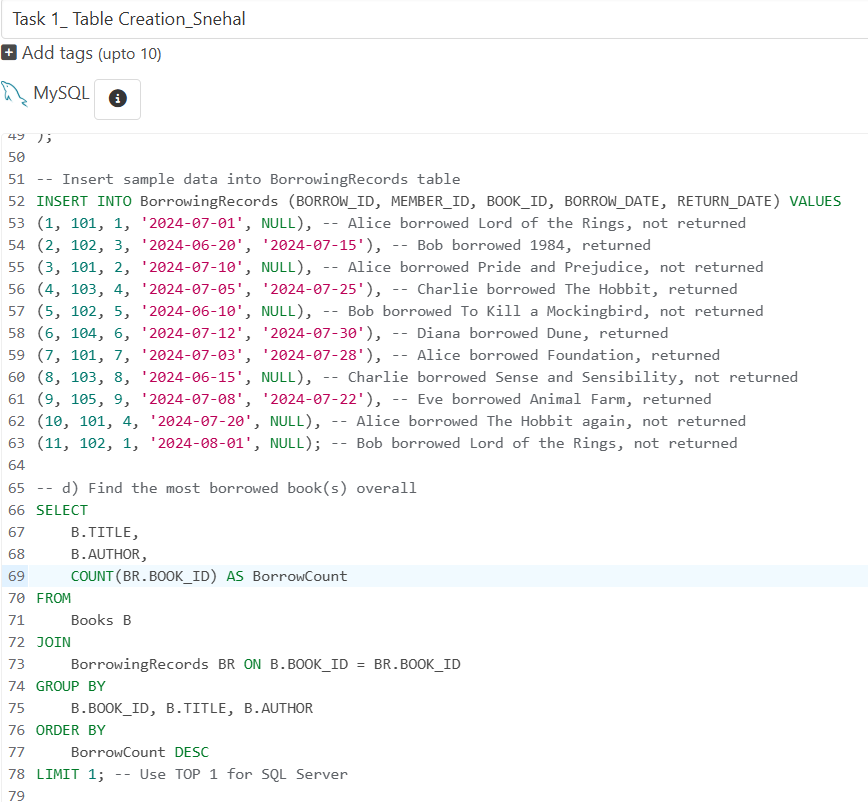
BorrowCount DESC

LIMIT 1; -- Use TOP 1 for SQL Server

----- This query identifies the book or books that have been borrowed the most times across all borrowing records.

--- Using JOIN on Book ID from Books and BorrowingRecords table to get required output along with LIMIT to retrieve Top 1 Borrowed book

--- In Output we get Book Title, Author and Borrow Count

****

****

1. **Retrieve members who have borrowed books from at least three different genres**

SELECT m.NAME

FROM Members m

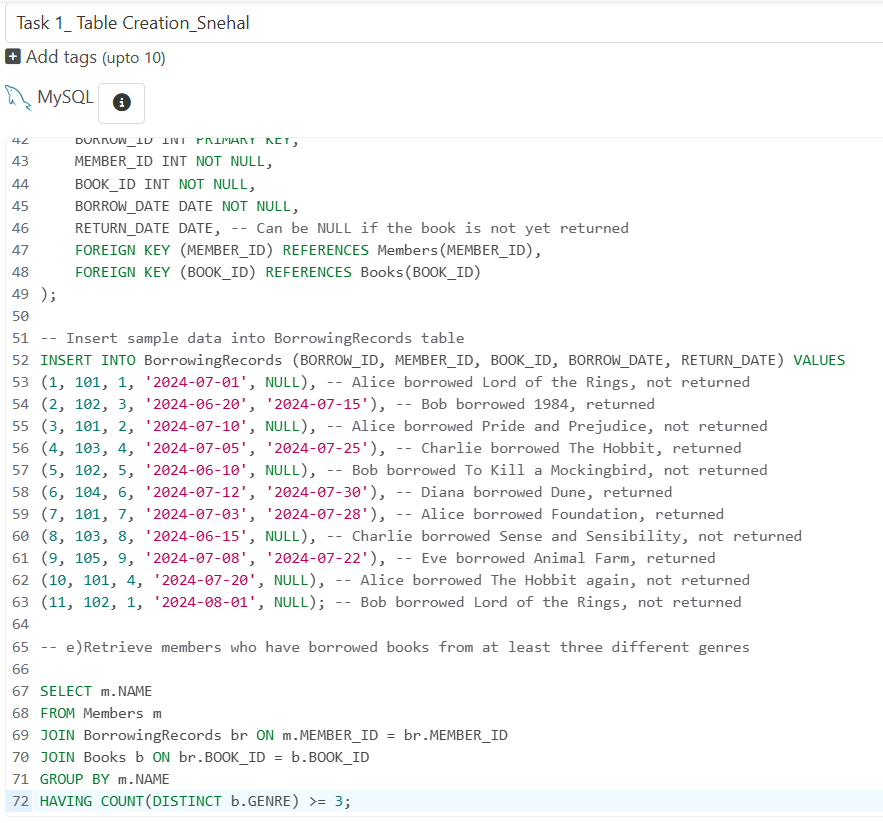
JOIN BorrowingRecords br ON m.MEMBER\_ID = br.MEMBER\_ID

JOIN Books b ON br.BOOK\_ID = b.BOOK\_ID

GROUP BY m.NAME

HAVING COUNT(DISTINCT b.GENRE) >= 3;

---- This query retrieves members who have borrowed books belonging to at least three distinct genres by joining the tables, grouping by member name, and then filtering with a HAVING clause to count the distinct genres associated with each member's borrowings.

****



---- In the output Names of Members is retrieved, as per sample data Alice Smith and Bob Johnson has borrowed books from atleast three different genre

1. **Reporting and analytics:**
2. **Calculate the total number of books borrowed per month**

SELECT YEAR(BORROW\_DATE) AS BorrowYear,

MONTH(BORROW\_DATE) AS BorrowMonth,

COUNT(BORROW\_ID) AS TotalBorrowedBooks

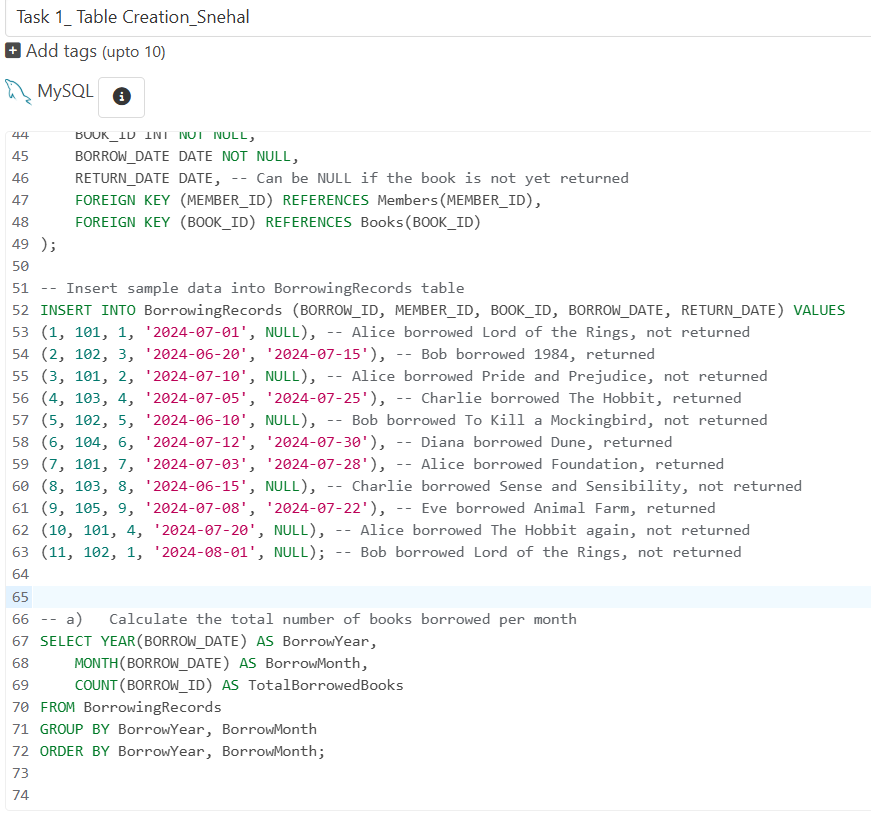
FROM BorrowingRecords

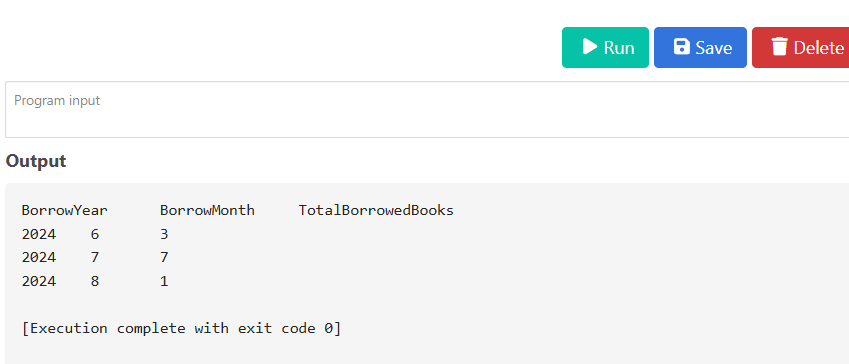
GROUP BY BorrowYear, BorrowMonth

ORDER BY BorrowYear, BorrowMonth;

---- This query calculates the total number of books borrowed each month, extracting the year and month from the borrow date and then grouping the data to aggregate the counts.

--- By using COUNT and GROUP BY we get the required output with Borrow Year, Month and count of books





1. **Find the top three most active members based on the number of books borrowed**

SELECT

M.NAME AS MemberName,

COUNT(BR.BOOK\_ID) AS BooksBorrowed

FROM

Members M

JOIN

BorrowingRecords BR ON M.MEMBER\_ID = BR.MEMBER\_ID

GROUP BY

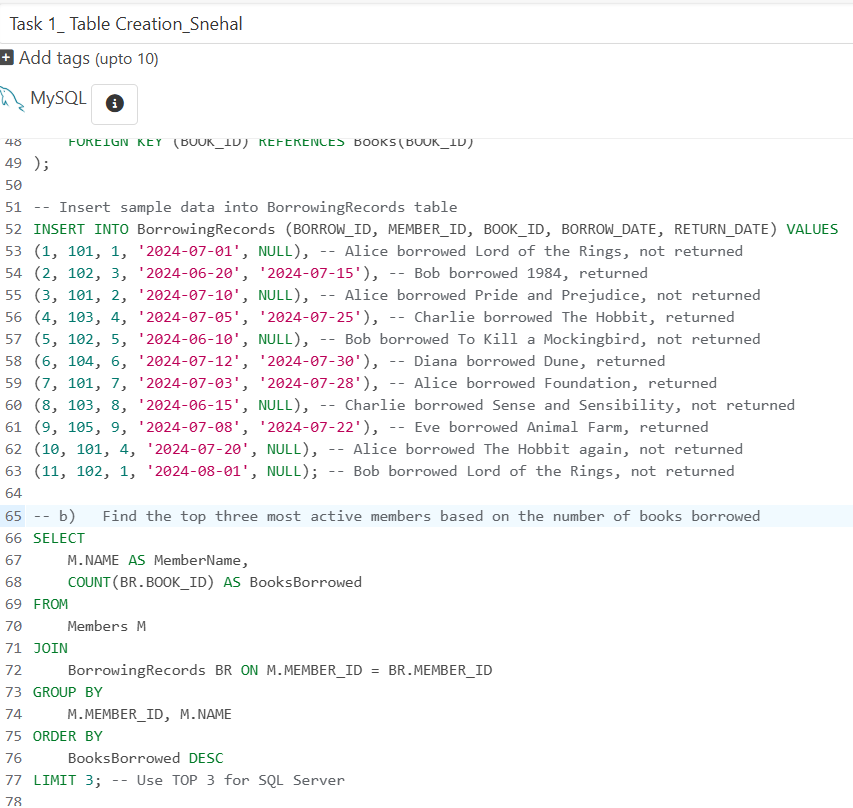
M.MEMBER\_ID, M.NAME

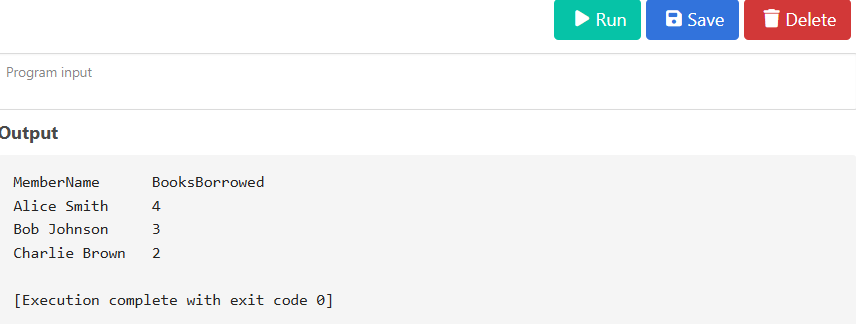
ORDER BY

BooksBorrowed DESC

LIMIT 3; -- Use TOP 3 for SQL Server

---- This query identifies the top three members who have borrowed the most books by joining the Members and BorrowingRecords tables, grouping the results by member name, and ordering them in descending order based on the count of books borrowed





1. **Retrieve authors whose books have been borrowed at least 10 times**

SELECT

B.AUTHOR,

COUNT(BR.BOOK\_ID) AS TotalBorrows

FROM

Books B

JOIN

BorrowingRecords BR ON B.BOOK\_ID = BR.BOOK\_ID

GROUP BY

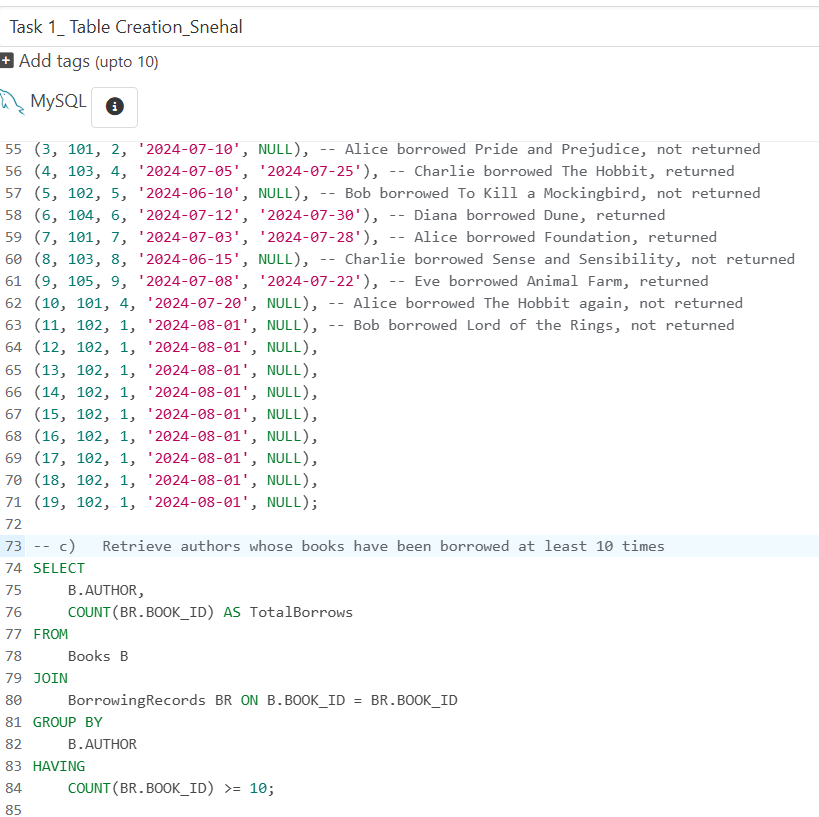
B.AUTHOR

HAVING

COUNT(BR.BOOK\_ID) >= 10;

---- This query identifies authors whose books have been borrowed a significant number of times (at least 10) by joining the Books and BorrowingRecords tables, grouping the results by author, and then using the HAVING clause to filter the authors based on the total count of borrowed books.

--- Data Manipulation in BorrowingRecords for MemberID 102 to get required output..





1. **Identify members who have never borrowed a book**

SELECT m.NAME, m.MEMBER\_ID

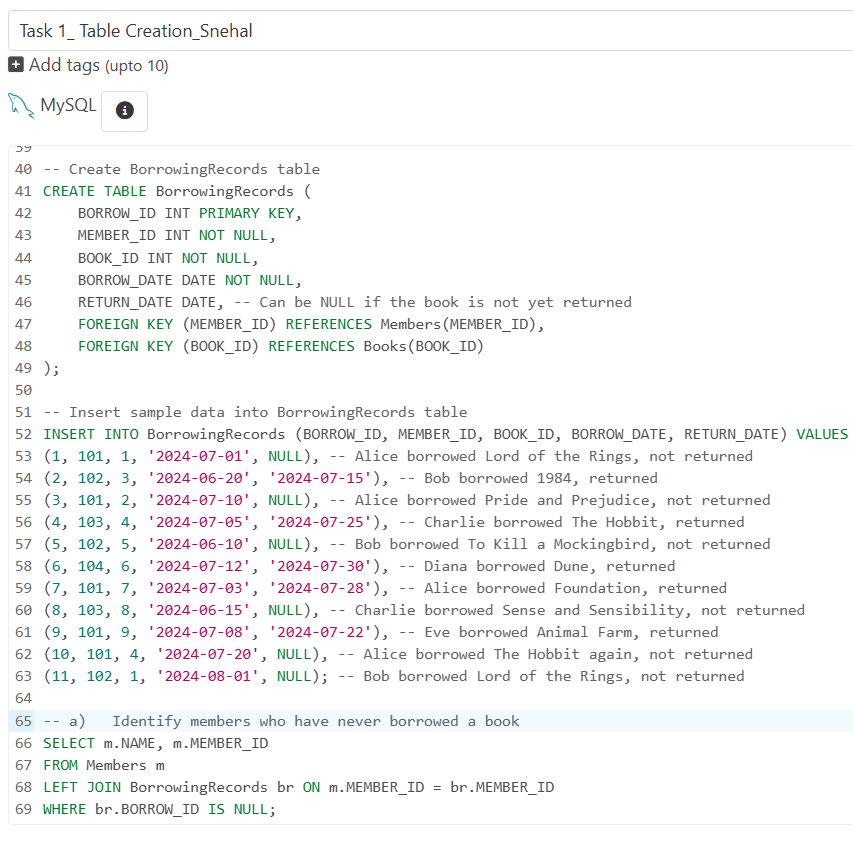
FROM Members m

LEFT JOIN BorrowingRecords br ON m.MEMBER\_ID = br.MEMBER\_ID

WHERE br.BORROW\_ID IS NULL;

--- Data manipulation in BorrowingRecords table for MemberID – 105 to get the required Output

--- This query identifies members who have not borrowed any books by performing a LEFT JOIN between the Members and BorrowingRecords tables, and then selecting those records where there is no matching entry in the BorrowingRecords table (indicating they haven't borrowed any books).





**TASK 1 ENDS HERE**