**Library management system using mysql database**

CREATE TABLE Authors (

AuthorID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100) NOT NULL,

Nationality VARCHAR(50)

);

CREATE TABLE Categories (

CategoryID INT PRIMARY KEY AUTO\_INCREMENT,

CategoryName VARCHAR(100) NOT NULL

);

CREATE TABLE Publishers (

PublisherID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100) NOT NULL,

Address TEXT,

Phone VARCHAR(15)

);

CREATE TABLE Books (

BookID INT PRIMARY KEY AUTO\_INCREMENT,

Title VARCHAR(200) NOT NULL,

ISBN VARCHAR(20),

AuthorID INT,

PublisherID INT,

CategoryID INT,

YearPublished YEAR,

CopiesAvailable INT,

FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID),

FOREIGN KEY (PublisherID) REFERENCES Publishers(PublisherID),

FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)

);

CREATE TABLE Students (

StudentID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100) NOT NULL,

Email VARCHAR(100),

Phone VARCHAR(15),

Address TEXT,

EnrollmentYear YEAR

);

CREATE TABLE BorrowRecords (

LoanID INT PRIMARY KEY AUTO\_INCREMENT,

BookID INT,

StudentID INT,

BorrowDate DATE NOT NULL,

DueDate DATE NOT NULL,

ReturnDate DATE,

Status ENUM('Borrowed', 'Returned') DEFAULT 'Borrowed',

FOREIGN KEY (BookID) REFERENCES Books(BookID),

FOREIGN KEY (StudentID) REFERENCES Students(StudentID)

);

CREATE TABLE Reservations (

ReservationID INT PRIMARY KEY AUTO\_INCREMENT,

BookID INT,

StudentID INT,

ReservationDate DATE NOT NULL,

ExpirationDate DATE NOT NULL,

FOREIGN KEY (BookID) REFERENCES Books(BookID),

FOREIGN KEY (StudentID) REFERENCES Students(StudentID)

);

Papolate each table with raw data;

Authors table

INSERT INTO Authors (Name, Nationality) VALUES

('J.K. Rowling', 'British'),

('George R.R. Martin', 'American'),

('Agatha Christie', 'British'),

('J.R.R. Tolkien', 'British'),

('Stephen King', 'American');

Category table

INSERT INTO Categories (CategoryName) VALUES

('Fiction'),

('Science Fiction'),

('Fantasy'),

('Mystery'),

('Non-Fiction'),

('Historical Fiction');

Publieshers

INSERT INTO Publishers (Name, Address, Phone) VALUES

('Penguin Random House', '123 Main St, New York, NY', '123-456-7890'),

('HarperCollins', '456 Elm St, London, UK', '987-654-3210'),

('Macmillan Publishers', '789 Oak St, Sydney, AU', '456-789-1230'),

('Hachette Livre', '101 Maple Ave, Paris, FR', '321-654-9870');

Books

**INSERT INTO Books (Title, ISBN, AuthorID, PublisherID, CategoryID, YearPublished, CopiesAvailable) VALUES**

**('Harry Potter and the Philosopher\'s Stone', '9780747532743', 1, 1, 3, 1997, 5),**

**('A Game of Thrones', '9780553103540', 2, 2, 3, 1996, 3),**

**('Murder on the Orient Express', '9780062073501', 3, 2, 4, 1934, 4),**

**('The Hobbit', '9780345339683', 4, 3, 3, 1937, 2),**

**('The Shining', '9780307743657', 5, 1, 2, 1977, 3);**

**Students;**

**INSERT INTO Students (Name, Email, Phone, Address, EnrollmentYear) VALUES**

**('Alice Johnson', 'alice.johnson@example.com', '555-1234', '1 University Rd, Cityville', 2022),**

**('Bob Smith', 'bob.smith@example.com', '555-5678', '2 University Rd, Cityville', 2023),**

**('Catherine Lee', 'catherine.lee@example.com', '555-9101', '3 University Rd, Cityville', 2021),**

**('David Brown', 'david.brown@example.com', '555-1122', '4 University Rd, Cityville', 2020);**

**Reservations**

**INSERT INTO Reservations (BookID, StudentID, ReservationDate, ExpirationDate) VALUES**

**(5, 1, '2024-01-16', '2024-01-20'),**

**(3, 2, '2024-01-17', '2024-01-21');**

Here’s a comprehensive list of SQL queries ranging from simple to advanced, to test the library management system database:

**Simple Queries**

1. **List all Students**

sql

Copy code

SELECT \* FROM Students;

1. **List all Books by a Specific Author**

sql

Copy code

SELECT Title FROM Books WHERE AuthorID = 1;

1. **List all Categories**

sql

Copy code

SELECT \* FROM Categories;

1. **Find Students Enrolled in a Specific Year**

sql

Copy code

SELECT Name, Email FROM Students WHERE EnrollmentYear = 2023;

**Intermediate Queries**

1. **Count the Total Number of Books**

sql

Copy code

SELECT COUNT(\*) AS TotalBooks FROM Books;

1. **Find Books Borrowed by a Specific Student**

sql

Copy code

SELECT b.Title, br.BorrowDate, br.DueDate

FROM BorrowRecords br

JOIN Books b ON br.BookID = b.BookID

WHERE br.StudentID = 1;

1. **List Books by Category**

sql

Copy code

SELECT c.CategoryName, COUNT(b.BookID) AS TotalBooks

FROM Categories c

JOIN Books b ON c.CategoryID = b.CategoryID

GROUP BY c.CategoryName;

1. **Find Overdue Books**

sql

Copy code

SELECT b.Title, s.Name, br.DueDate

FROM BorrowRecords br

JOIN Books b ON br.BookID = b.BookID

JOIN Students s ON br.StudentID = s.StudentID

WHERE br.DueDate < CURDATE() AND br.Status = 'Borrowed';

1. **Check Reservation Status**

sql

Copy code

SELECT b.Title, s.Name, r.ReservationDate, r.ExpirationDate

FROM Reservations r

JOIN Books b ON r.BookID = b.BookID

JOIN Students s ON r.StudentID = s.StudentID

WHERE r.ExpirationDate > CURDATE();

**Advanced Queries**

1. **Find the Most Borrowed Book**

sql

Copy code

SELECT b.Title, COUNT(br.LoanID) AS TimesBorrowed

FROM BorrowRecords br

JOIN Books b ON br.BookID = b.BookID

GROUP BY b.Title

ORDER BY TimesBorrowed DESC

LIMIT 1;

1. **Find Students with the Most Borrowed Books**

sql

Copy code

SELECT s.Name, COUNT(br.LoanID) AS TotalBooksBorrowed

FROM BorrowRecords br

JOIN Students s ON br.StudentID = s.StudentID

GROUP BY s.Name

ORDER BY TotalBooksBorrowed DESC

LIMIT 1;

1. **Find Available Copies of a Book**

sql

Copy code

SELECT Title, CopiesAvailable

FROM Books

WHERE BookID = 1;

1. **List Authors and the Number of Books They Wrote**

sql

Copy code

SELECT a.Name, COUNT(b.BookID) AS TotalBooks

FROM Authors a

JOIN Books b ON a.AuthorID = b.AuthorID

GROUP BY a.Name

ORDER BY TotalBooks DESC;

1. **List Students Who Borrowed Books in a Specific Month**

sql

Copy code

SELECT s.Name, b.Title, br.BorrowDate

FROM BorrowRecords br

JOIN Students s ON br.StudentID = s.StudentID

JOIN Books b ON br.BookID = b.BookID

WHERE MONTH(br.BorrowDate) = 1 AND YEAR(br.BorrowDate) = 2024;

1. **List All Books Borrowed But Not Yet Returned**

sql

Copy code

SELECT b.Title, s.Name, br.BorrowDate, br.DueDate

FROM BorrowRecords br

JOIN Books b ON br.BookID = b.BookID

JOIN Students s ON br.StudentID = s.StudentID

WHERE br.Status = 'Borrowed';

**Complex Queries**

1. **Find Books That Are Both Borrowed and Reserved**

sql

Copy code

SELECT b.Title

FROM Books b

JOIN BorrowRecords br ON b.BookID = br.BookID

JOIN Reservations r ON b.BookID = r.BookID

WHERE br.Status = 'Borrowed';

1. **Find Top 3 Students with the Most Reservations**

sql

Copy code

SELECT s.Name, COUNT(r.ReservationID) AS TotalReservations

FROM Reservations r

JOIN Students s ON r.StudentID = s.StudentID

GROUP BY s.Name

ORDER BY TotalReservations DESC

LIMIT 3;

1. **List All Books Published After a Certain Year**

sql

Copy code

SELECT Title, YearPublished

FROM Books

WHERE YearPublished > 2000;

1. **Find the Average Borrow Duration for Books**

sql

Copy code

SELECT AVG(DATEDIFF(ReturnDate, BorrowDate)) AS AverageBorrowDuration

FROM BorrowRecords

WHERE ReturnDate IS NOT NULL;

1. **Find Books Borrowed by Students in Their First Year**

sql

Copy code

SELECT b.Title, s.Name, s.EnrollmentYear, br.BorrowDate

FROM BorrowRecords br

JOIN Books b ON br.BookID = b.BookID

JOIN Students s ON br.StudentID = s.StudentID

WHERE YEAR(br.BorrowDate) = s.EnrollmentYear;

**Dynamic and Parameterized Queries**

1. **Search Books by Title (Partial Match)**

sql

Copy code

SELECT \* FROM Books WHERE Title LIKE '%Harry%';

1. **Get Book Details by ISBN**

sql

Copy code

SELECT \* FROM Books WHERE ISBN = '9780747532743';

1. **Generate Fine for Overdue Books**

sql

Copy code

SELECT s.Name, b.Title, DATEDIFF(CURDATE(), br.DueDate) \* 2 AS FineAmount

FROM BorrowRecords br

JOIN Books b ON br.BookID = b.BookID

JOIN Students s ON br.StudentID = s.StudentID

WHERE br.DueDate < CURDATE() AND br.Status = 'Borrowed';

Let me know if you need more specialized queries or assistance running these in a particular database system!

**Some more queries to test the database from beginner to expert level**

**Beginner Level**

1. **List all Librarians**

sql

Copy code

SELECT \* FROM Librarians;

1. **Find All Books Published by a Specific Publisher**

sql

Copy code

SELECT Title FROM Books WHERE PublisherID = 2;

1. **List Students with a Specific Name**

sql

Copy code

SELECT \* FROM Students WHERE Name = 'Alice Johnson';

1. **Retrieve Books Available in a Specific Category**

sql

Copy code

SELECT Title FROM Books WHERE CategoryID = 3;

1. **Find Total Copies of a Specific Book**

sql

Copy code

SELECT CopiesAvailable FROM Books WHERE Title = 'The Hobbit';

**Intermediate Level**

1. **List All Borrowed Books and Their Due Dates**

sql

Copy code

SELECT b.Title, br.DueDate

FROM BorrowRecords br

JOIN Books b ON br.BookID = b.BookID

WHERE br.Status = 'Borrowed';

1. **Find Total Number of Books Reserved by Each Student**

sql

Copy code

SELECT s.Name, COUNT(r.ReservationID) AS TotalReservations

FROM Reservations r

JOIN Students s ON r.StudentID = s.StudentID

GROUP BY s.Name;

1. **Retrieve the Names of Students Who Reserved Specific Books**

sql

Copy code

SELECT DISTINCT s.Name

FROM Reservations r

JOIN Students s ON r.StudentID = s.StudentID

WHERE r.BookID IN (1, 2);

1. **List All Borrowers Who Have Not Returned a Book**

sql

Copy code

SELECT s.Name, b.Title

**FROM BorrowRecords br**

**JOIN Students s ON br.StudentID = s.StudentID**

**JOIN Books b ON br.BookID = b.BookID**

**WHERE br.Status = 'Borrowed';**

1. **Find Students Who Borrowed Books in the Last Month**

*sql*

*Copy code*

*SELECT s.Name, b.Title, br.BorrowDate*

*FROM BorrowRecords br*

*JOIN Students s ON br.StudentID = s.StudentID*

*JOIN Books b ON br.BookID = b.BookID*

*WHERE br.BorrowDate >= DATE\_SUB(CURDATE(), INTERVAL 1 MONTH);*

**Advanced Level**

1. **Calculate the Fine for Each Overdue Book**

sql

Copy code

SELECT br.LoanID, s.Name, b.Title,

DATEDIFF(CURDATE(), br.DueDate) \* 2 AS FineAmount

FROM BorrowRecords br

JOIN Students s ON br.StudentID = s.StudentID

JOIN Books b ON br.BookID = b.BookID

WHERE br.DueDate < CURDATE() AND br.Status = 'Borrowed';

1. **Find All Students Who Borrowed Books by a Specific Author**

sql

Copy code

SELECT DISTINCT s.Name

FROM BorrowRecords br

JOIN Books b ON br.BookID = b.BookID

JOIN Students s ON br.StudentID = s.StudentID

WHERE b.AuthorID = 1;

1. **Find All Books Borrowed More Than 3 Times**

sql

Copy code

SELECT b.Title, COUNT(br.LoanID) AS TimesBorrowed

FROM BorrowRecords br

JOIN Books b ON br.BookID = b.BookID

GROUP BY b.Title

HAVING TimesBorrowed > 3;

1. **List Students Who Reserved Books But Never Borrowed**

sql

Copy code

SELECT DISTINCT s.Name

FROM Reservations r

JOIN Students s ON r.StudentID = s.StudentID

WHERE s.StudentID NOT IN (

SELECT DISTINCT StudentID FROM BorrowRecords

);

1. **Find Books That Are Currently Reserved but Not Borrowed**

sql

Copy code

SELECT b.Title

FROM Books b

JOIN Reservations r ON b.BookID = r.BookID

WHERE b.BookID NOT IN (

SELECT BookID FROM BorrowRecords WHERE Status = 'Borrowed'

);

1. **Retrieve the Top 3 Most Popular Categories**

sql

Copy code

SELECT c.CategoryName, COUNT(br.BookID) AS TotalBorrows

FROM BorrowRecords br

JOIN Books b ON br.BookID = b.BookID

JOIN Categories c ON b.CategoryID = c.CategoryID

GROUP BY c.CategoryName

ORDER BY TotalBorrows DESC

LIMIT 3;

1. **List Students Who Borrowed and Reserved the Same Book**

sql

Copy code

SELECT DISTINCT s.Name, b.Title

FROM BorrowRecords br

JOIN Reservations r ON br.BookID = r.BookID AND br.StudentID = r.StudentID

JOIN Books b ON br.BookID = b.BookID

JOIN Students s ON br.StudentID = s.StudentID;

1. **List Books Borrowed by Students in the Past Year**

sql

Copy code

SELECT b.Title, s.Name, br.BorrowDate

FROM BorrowRecords br

JOIN Books b ON br.BookID = b.BookID

JOIN Students s ON br.StudentID = s.StudentID

WHERE br.BorrowDate >= DATE\_SUB(CURDATE(), INTERVAL 1 YEAR);

1. **Calculate the Average Borrow Duration for Books by Category**

sql

Copy code

SELECT c.CategoryName, AVG(DATEDIFF(br.ReturnDate, br.BorrowDate)) AS AvgBorrowDuration

FROM BorrowRecords br

JOIN Books b ON br.BookID = b.BookID

JOIN Categories c ON b.CategoryID = c.CategoryID

WHERE br.ReturnDate IS NOT NULL

GROUP BY c.CategoryName;

1. **Find the Author Whose Books Are Borrowed the Most**

sql

Copy code

SELECT a.Name, COUNT(br.BookID) AS TotalBorrows

FROM BorrowRecords br

JOIN Books b ON br.BookID = b.BookID

JOIN Authors a ON b.AuthorID = a.AuthorID

GROUP BY a.Name

ORDER BY TotalBorrows DESC

LIMIT 1;

These queries test different aspects of the library management system, including joins, aggregations, filtering, grouping, and subqueries. If you'd like more explanation on any query or need help testing them in a specific SQL environment, let me know!

4o

Here are queries ranging from simple to advanced that you can use to check the reviews table in the library\_management\_system database.

### 1. ****Simple Queries****

#### a. ****Select all data from the**** reviews ****table****

sql

Copy code

SELECT \* FROM reviews;

* This will retrieve all columns and rows from the reviews table.

#### b. ****Select specific columns (e.g.,**** review****,**** student\_id****, and**** book\_isbn****)****

sql

Copy code

SELECT review, student\_id, book\_isbn FROM reviews;

* This retrieves only the review, student\_id, and book\_isbn columns from the reviews table.

#### c. ****Count the number of reviews in the table****

sql

Copy code

SELECT COUNT(\*) AS total\_reviews FROM reviews;

* This returns the total number of reviews in the reviews table.

### 2. ****Intermediate Queries****

#### a. ****Get reviews by a specific student (e.g.,**** student\_id = 1****)****

sql

Copy code

SELECT \* FROM reviews WHERE student\_id = 1;

* This returns all reviews from the student with student\_id = 1.

#### b. ****Get reviews for a specific book (e.g.,**** book\_isbn = '978-3-16-148410-0'****)****

sql

Copy code

SELECT \* FROM reviews WHERE book\_isbn = '978-3-16-148410-0';

* This returns all reviews for the book with book\_isbn = '978-3-16-148410-0'.

#### c. ****Get reviews after a specific date (e.g., after '2024-01-01')****

sql

Copy code

SELECT \* FROM reviews WHERE review\_date > '2024-01-01';

* This retrieves all reviews submitted after January 1, 2024.

#### d. ****Get reviews and student names (assuming you have a**** students ****table with**** StudentID ****and**** StudentName****)****

sql

Copy code

SELECT r.review, s.StudentName

FROM reviews r

JOIN students s ON r.student\_id = s.StudentID;

* This joins the reviews table with the students table to get the student name along with the review.

### 3. ****Advanced Queries****

#### a. ****Get the most recent review for each book****

sql

Copy code

SELECT book\_isbn, review, review\_date

FROM reviews

WHERE review\_date IN (

SELECT MAX(review\_date)

FROM reviews

GROUP BY book\_isbn

);

* This returns the most recent review for each book by comparing the maximum review date for each book\_isbn.

#### b. ****Get the average number of reviews per student****

sql

Copy code

SELECT student\_id, COUNT(\*) AS num\_reviews, AVG(review\_date) AS avg\_review\_date

FROM reviews

GROUP BY student\_id;

* This groups the reviews by student\_id, counts the number of reviews each student has made, and calculates the average review date.

#### c. ****Get reviews for books with the highest number of reviews****

sql

Copy code

SELECT book\_isbn, COUNT(\*) AS review\_count

FROM reviews

GROUP BY book\_isbn

ORDER BY review\_count DESC

LIMIT 1;

* This retrieves the book\_isbn with the most reviews and the count of those reviews.

#### d. ****Find the most active students (students with the highest number of reviews)****

sql

Copy code

SELECT student\_id, COUNT(\*) AS review\_count

FROM reviews

GROUP BY student\_id

ORDER BY review\_count DESC

LIMIT 5;

* This retrieves the top 5 students with the highest number of reviews.

#### e. ****Find the average rating for each book (assuming a**** rating ****column exists in the**** reviews ****table)****

sql

Copy code

SELECT book\_isbn, AVG(rating) AS avg\_rating

FROM reviews

GROUP BY book\_isbn

ORDER BY avg\_rating DESC;

* This calculates the average rating for each book (if there’s a rating column) and sorts the books by average rating in descending order.

#### f. ****Get the reviews that contain a specific keyword (e.g., 'great')****

sql

Copy code

SELECT review, student\_id, book\_isbn

FROM reviews

WHERE review LIKE '%great%';

* This searches for reviews that contain the word "great" and returns the review, student ID, and book ISBN.

### Key Points:

* **Simple queries** are typically used for basic data retrieval.
* **Intermediate queries** involve filtering, joining tables, or aggregating data.
* **Advanced queries** use complex operations like subqueries, grouping, ordering, and functions to analyze or summarize data.

Make sure to adapt the column names, table names, and conditions based on your actual database schema.

4o mini

##### You said:

also give me queries to the name of book and student and relvent informant about student and books form reviwe table

###### ChatGPT said:

ChatGPT

To retrieve the names of books and students along with relevant information from the reviews table, you can use **JOIN** queries to combine data from the reviews, students, and books tables. Below are some example queries that fetch such information.

### 1. ****Retrieve Book Name, Student Name, and Review Content****

Assuming the following table structures:

* reviews: review, student\_id, book\_isbn, review\_date
* students: StudentID, StudentName, etc.
* books: BookID, BookTitle, ISBN, etc.

#### Query to Get Book Title, Student Name, and Review

sql

Copy code

SELECT b.BookTitle, s.StudentName, r.review, r.review\_date

FROM reviews r

JOIN students s ON r.student\_id = s.StudentID

JOIN books b ON r.book\_isbn = b.ISBN;

**Explanation:**

* This query joins the reviews table with the students and books tables using the student\_id and book\_isbn as foreign keys.
* It returns the **book title**, **student name**, **review**, and **review date**.

### 2. ****Retrieve Student Name, Book Name, and Review Content for a Specific Book****

If you want to see reviews for a specific book (e.g., ISBN = '978-3-16-148410-0'), you can filter the results:

sql

Copy code

SELECT s.StudentName, b.BookTitle, r.review, r.review\_date

FROM reviews r

JOIN students s ON r.student\_id = s.StudentID

JOIN books b ON r.book\_isbn = b.ISBN

WHERE r.book\_isbn = '978-3-16-148410-0';

**Explanation:**

* This query filters the results by the book\_isbn to show reviews specifically for a given book (in this case, 978-3-16-148410-0).
* It retrieves the **student name**, **book title**, **review**, and **review date**.

### 3. ****Retrieve All Books and Corresponding Reviews with Student Information****

If you want to list all books and their corresponding reviews, including the student who made the review:

sql

Copy code

SELECT b.BookTitle, s.StudentName, r.review, r.review\_date

FROM reviews r

JOIN students s ON r.student\_id = s.StudentID

JOIN books b ON r.book\_isbn = b.ISBN

ORDER BY b.BookTitle, r.review\_date;

**Explanation:**

* This query lists all reviews for all books.
* It orders the results by **book title** and **review date**.

### 4. ****Get the Most Recent Review for Each Book with Student and Book Information****

To retrieve the most recent review for each book, you can use a subquery:

sql

Copy code

SELECT b.BookTitle, s.StudentName, r.review, r.review\_date

FROM reviews r

JOIN students s ON r.student\_id = s.StudentID

JOIN books b ON r.book\_isbn = b.ISBN

WHERE r.review\_date IN (

SELECT MAX(review\_date)

FROM reviews

WHERE book\_isbn = r.book\_isbn

GROUP BY book\_isbn

);

**Explanation:**

* This query retrieves the most recent review for each book by selecting the **maximum review date** for each book\_isbn.
* It shows the **book title**, **student name**, **review**, and **review date** for the latest review of each book.

### 5. ****Get Reviews for a Specific Student and Their Reviewed Books****

If you want to list all reviews made by a specific student (e.g., student\_id = 1), and the books they reviewed:

sql

Copy code

SELECT b.BookTitle, s.StudentName, r.review, r.review\_date

FROM reviews r

JOIN students s ON r.student\_id = s.StudentID

JOIN books b ON r.book\_isbn = b.ISBN

WHERE r.student\_id = 1;

**Explanation:**

* This query retrieves all reviews written by the student with student\_id = 1.
* It shows the **book title**, **student name**, **review**, and **review date** for all of that student's reviews.

### 6. ****Get the Number of Reviews Each Book Has with Student Information****

If you want to count the number of reviews each book has and get student information for each review:

sql

Copy code

SELECT b.BookTitle, COUNT(r.id) AS review\_count, GROUP\_CONCAT(s.StudentName ORDER BY s.StudentName) AS reviewers

FROM reviews r

JOIN students s ON r.student\_id = s.StudentID

JOIN books b ON r.book\_isbn = b.ISBN

GROUP BY b.BookTitle

ORDER BY review\_count DESC;

**Explanation:**

* This query counts the number of reviews for each book using COUNT(r.id).
* It also uses GROUP\_CONCAT to list all the student names who reviewed each book, sorted by their names.
* The results are ordered by the number of reviews in descending order.

### 7. ****Get Reviews with Book and Student Information for a Date Range****

If you want to get reviews within a specific date range, say between 2024-01-01 and 2024-01-10:

sql

Copy code

SELECT b.BookTitle, s.StudentName, r.review, r.review\_date

FROM reviews r

JOIN students s ON r.student\_id = s.StudentID

JOIN books b ON r.book\_isbn = b.ISBN

WHERE r.review\_date BETWEEN '2024-01-01' AND '2024-01-10'

ORDER BY r.review\_date;

**Explanation:**

* This query filters reviews within the given date range (2024-01-01 to 2024-01-10).
* It retrieves the **book title**, **student name**, **review**, and **review date** for reviews submitted during this period.