**library-management-system**

You are going to build a project based on Library Management System. It keeps track of all information about books in the library, their cost, status and total number of books available in the library.

Create a database named library and following TABLES in the database:   
  
1. Branch

create database library;

use library;

-- branch table

CREATE TABLE Branch (

Branch\_no CHAR(5) PRIMARY KEY,

Manager\_Id INT NOT NULL,

Branch\_address VARCHAR(30),

Contact\_no BIGINT

);  
2. Employee

CREATE TABLE employee (

Emp\_Id INT PRIMARY KEY,

Emp\_name VARCHAR(20),

Position VARCHAR(20),

Salary INT NOT NULL,

Branch\_no CHAR(5),

FOREIGN KEY (Branch\_no)

REFERENCES Branch (Branch\_no)

);   
3. Books

CREATE TABLE Books (

ISBN INT PRIMARY KEY,

Book\_title VARCHAR(650) NOT NULL,

Category VARCHAR(200),

Rental\_Price INT NOT NULL,

Status varchar(3),

Author VARCHAR(250),

Publisher VARCHAR(205)

);  
4. Customer

CREATE TABLE Customer (

Customer\_id INT PRIMARY KEY,

Customer\_name VARCHAR(20),

Customer\_address VARCHAR(30),

Reg\_date DATETIME DEFAULT CURRENT\_TIMESTAMP

);  
5. IssueStatus

CREATE TABLE IssueStatus (

Issue\_Id INT PRIMARY KEY,

Issued\_cust INT,

Issue\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

Isbn\_book INT,

FOREIGN KEY (Issued\_cust)

REFERENCES Customer (Customer\_id),

FOREIGN KEY (Isbn\_book)

REFERENCES Books (ISBN)

);  
6. ReturnStatus

CREATE TABLE ReturnStatus (

Return\_Id INT PRIMARY KEY,

Return\_cust INT,

Return\_book\_name VARCHAR(150),

Return\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

Isbn\_book2 INT,

FOREIGN KEY (Isbn\_book2)

REFERENCES Books (ISBN),

FOREIGN KEY (Return\_cust)

REFERENCES Customer (Customer\_id)

);

Created Some triggers and Views

-- Creating a view to get the no of books in the library ,available books , and books gone out

create view Books\_Count AS select(select count(\*) from Books) total\_no\_of\_books ,

(select count(\*) from Books where Status ='yes') no\_of\_books\_available\_now,

(select count(\*) from Books where Status ='No') no\_of\_books\_gone\_out;



-- creating trriger before inserting the data into issuestatus to not select avoid entering book not available by checking the books table

delimiter $$

create trigger before\_issue\_book\_to\_customer before insert on IssueStatus for each row

begin DECLARE book\_status varchar(3);

SELECT Status INTO book\_status FROM Books WHERE ISBN = NEW.Isbn\_book;

IF book\_status = "No" THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Cannot issue a book that is not available';

end if;

end$$

delimiter ;

-- creating trigger after insertion for changing update the book status form the book table as not available after that book is issuesed to customer

delimiter $$

create trigger after\_issue\_book\_to\_customer after insert on IssueStatus for each row

begin UPDATE Books

SET Status = "no"

WHERE ISBN = NEW.Isbn\_book;

end$$

delimiter ;

-- creating after triggers to change the book status from no to yes

delimiter $$

create trigger after\_return\_book\_to\_customer after insert on ReturnStatus for each row

begin UPDATE Books

SET Status = "Yes"

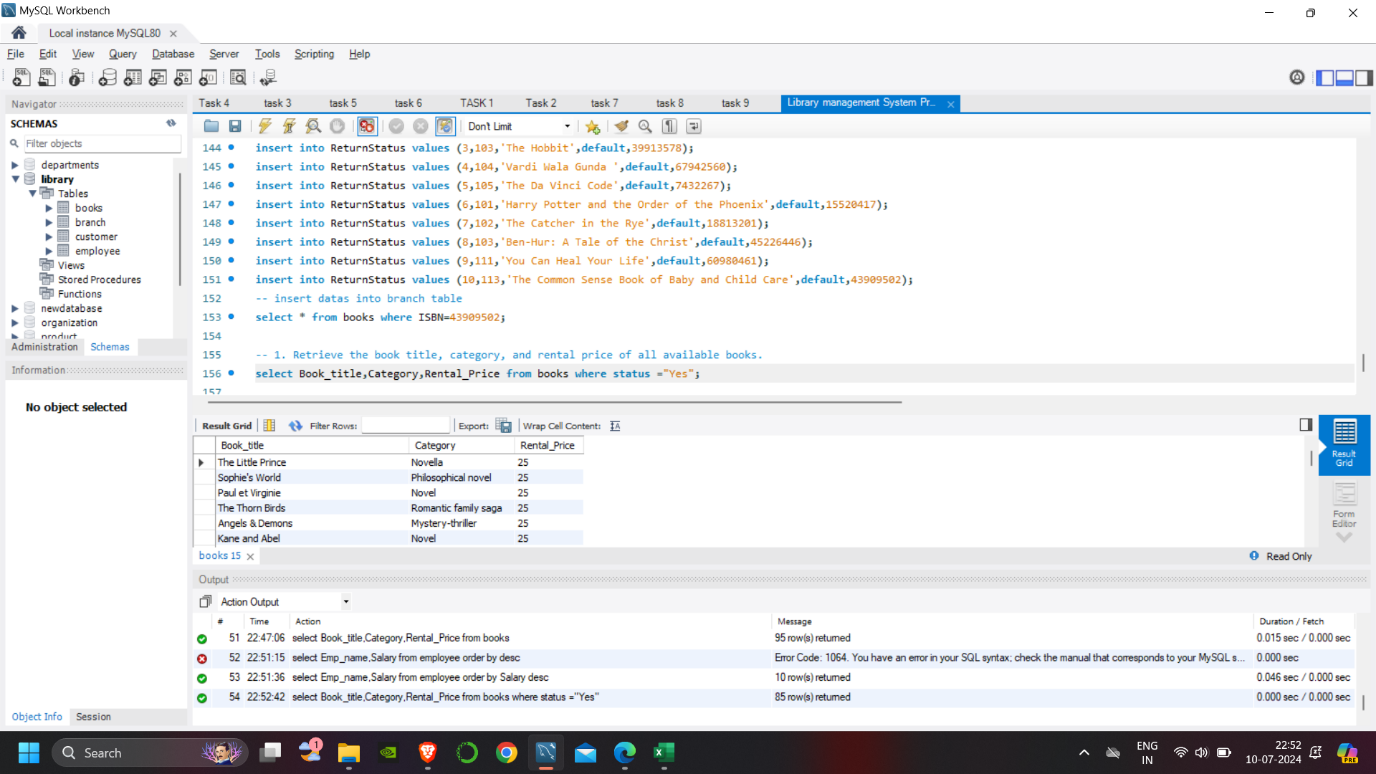
WHERE ISBN = NEW.Isbn\_book2;

end$$

delimiter ;

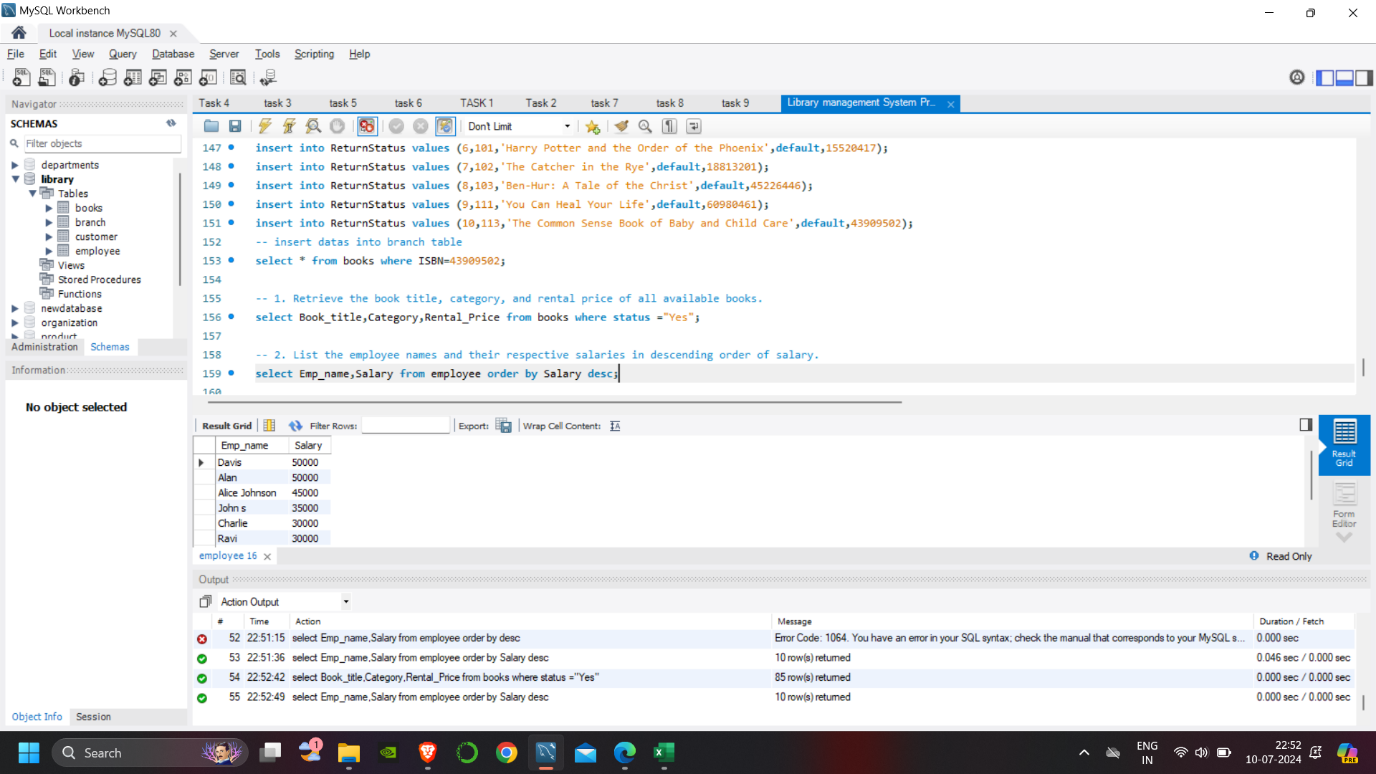
-- 1. Retrieve the book title, category, and rental price of all available books.

select Book\_title,Category,Rental\_Price from books where status ="yes";



-- 2. List the employee names and their respective salaries in descending order of salary.

select Emp\_name,Salary from employee order by Salary desc;



-- 3. Retrieve the book titles and the corresponding customers who have issued those books.

SELECT

B.Book\_title, I.Issued\_cust, C.Customer\_name

FROM

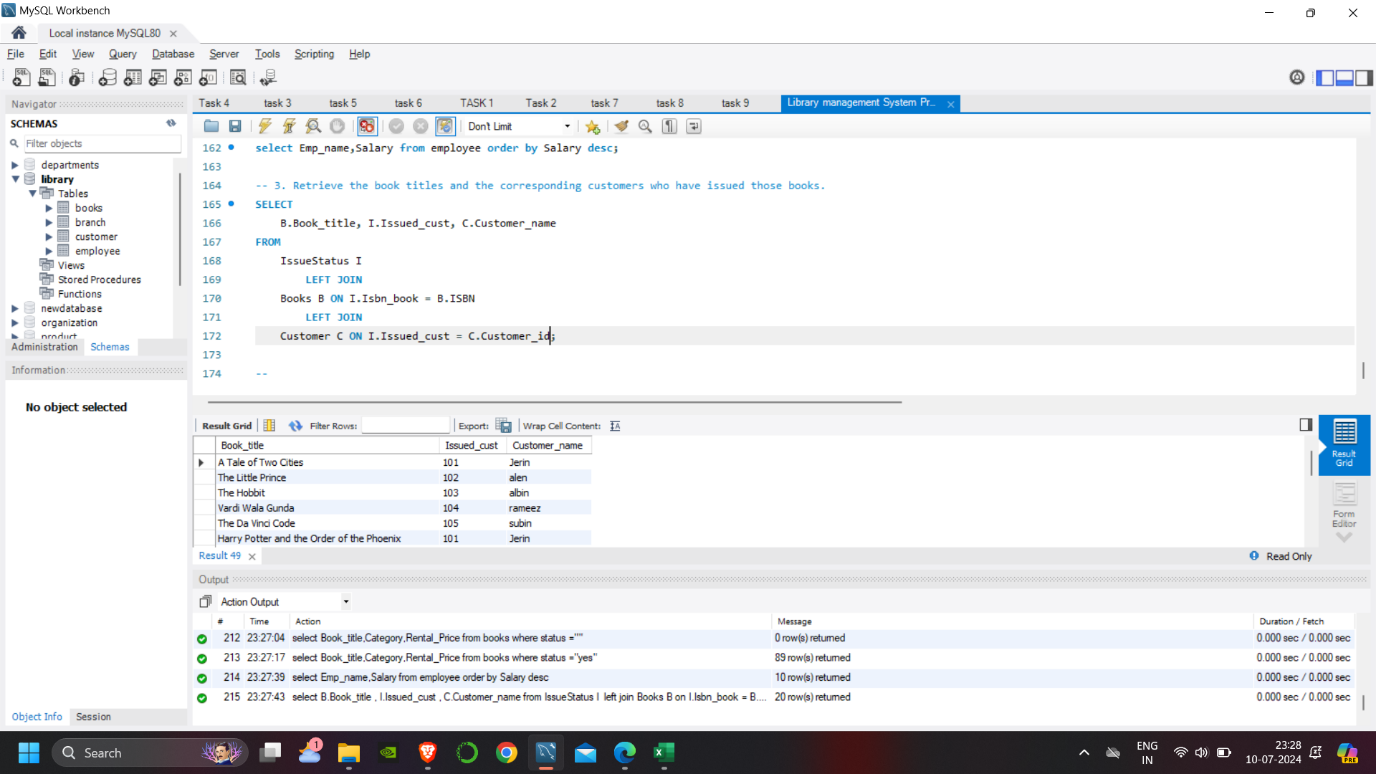
IssueStatus I

LEFT JOIN

Books B ON I.Isbn\_book = B.ISBN

LEFT JOIN

Customer C ON I.Issued\_cust = C.Customer\_id;



-- 4. Display the total count of books in each category.

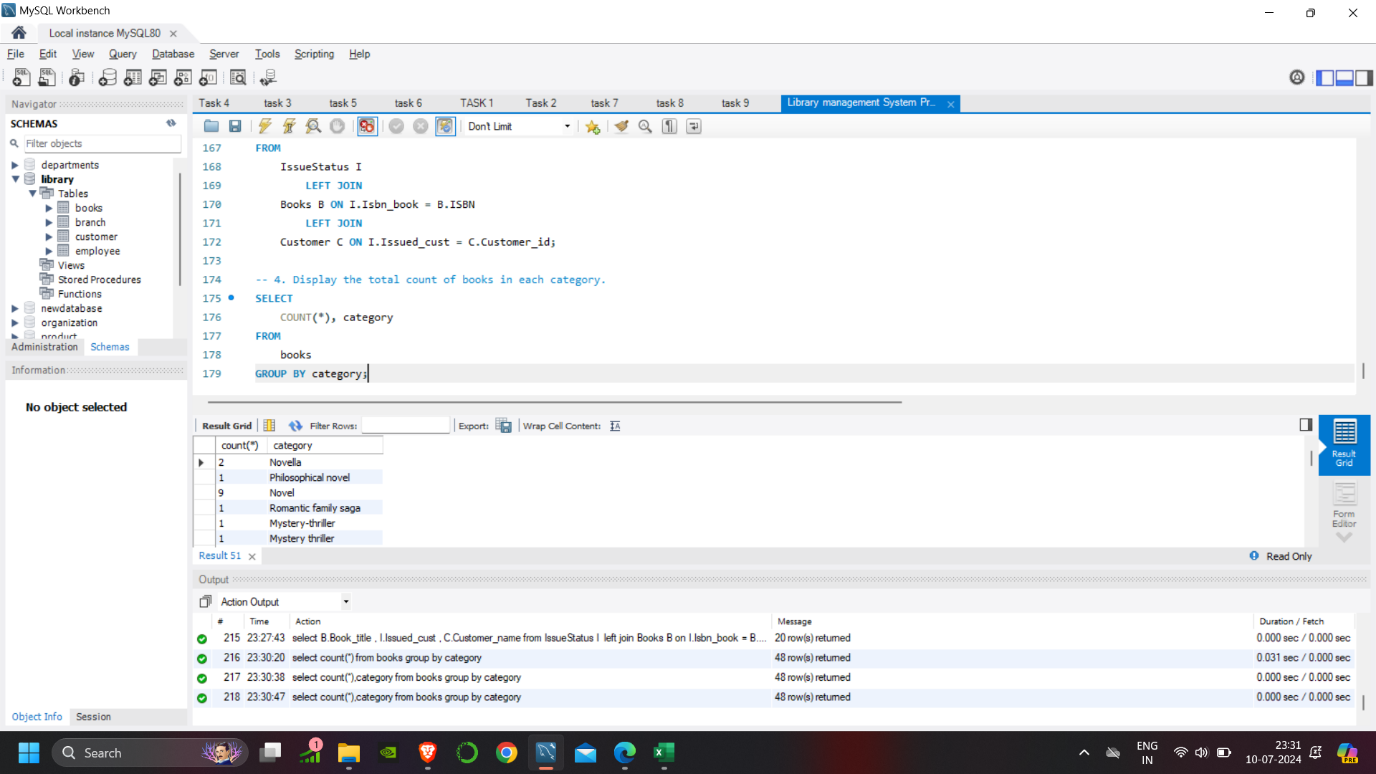
SELECT

COUNT(\*), category

FROM

books

GROUP BY category;



-- 5. Retrieve the employee names and their positions for the employees whose salaries are above Rs.50,000.

SELECT

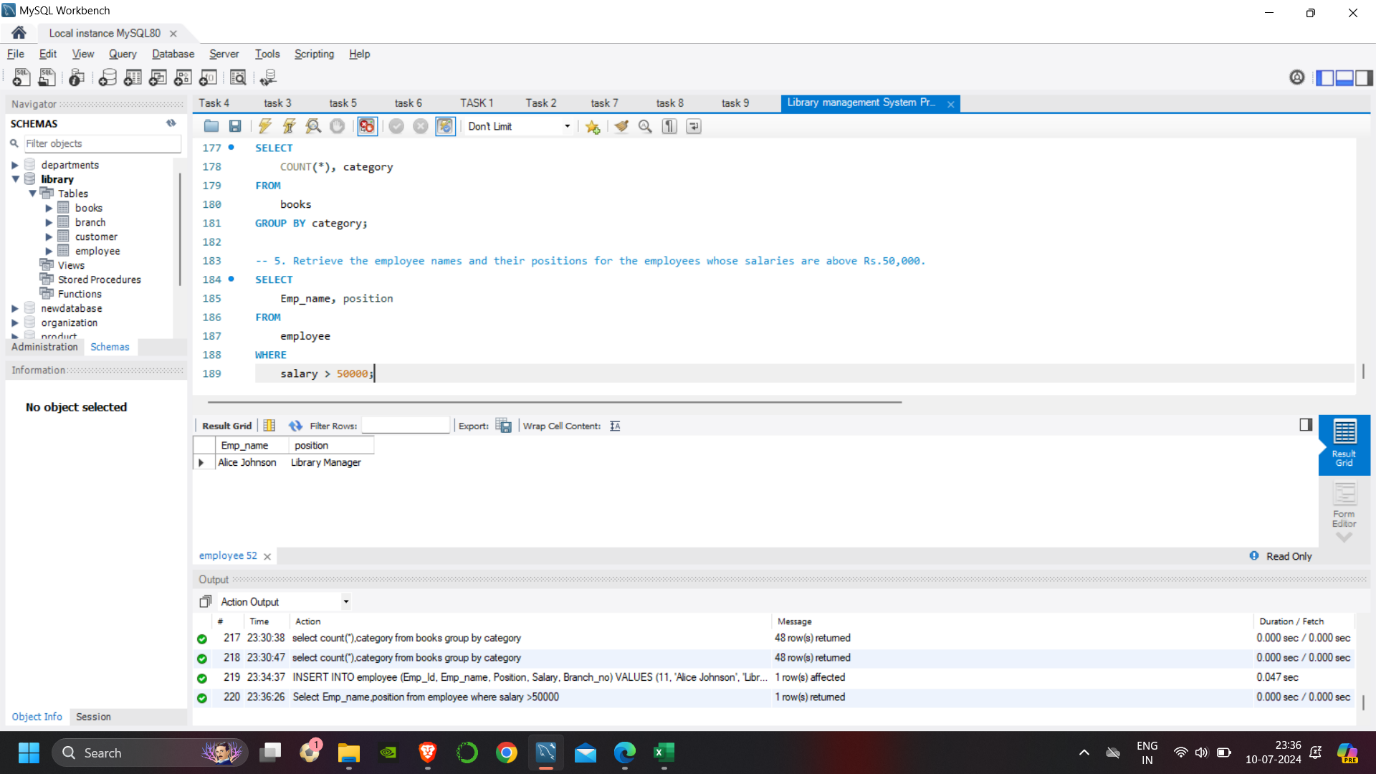
Emp\_name, position

FROM

employee

WHERE

salary > 50000;



-- 6. List the customer names who registered before 2022-01-01 and have not issued any books yet.

SELECT

customer\_name

FROM

customer

WHERE

customer\_id NOT IN (SELECT

issued\_cust

FROM

IssueStatus)

AND Reg\_date < '2022-01-01';

-- another way

SELECT

c.customer\_name

FROM

customer c

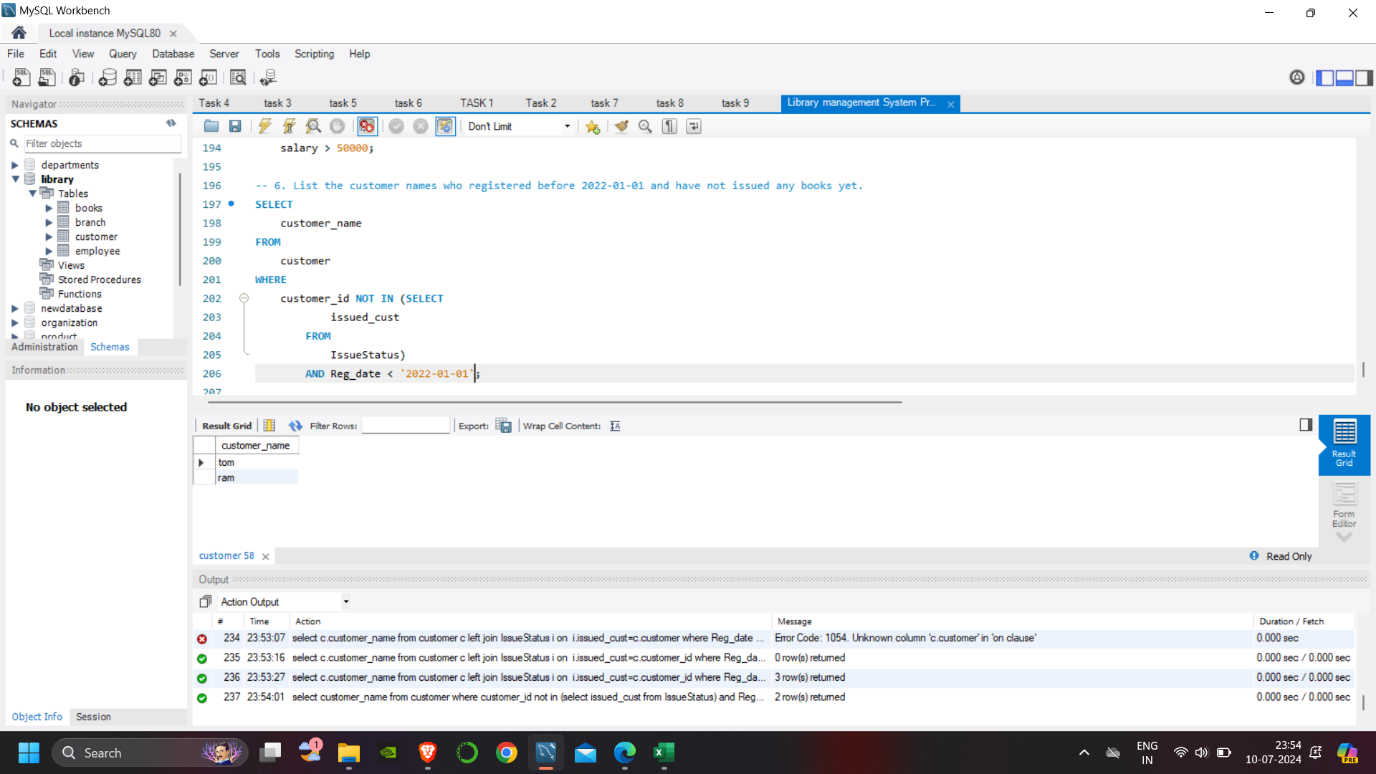
LEFT JOIN

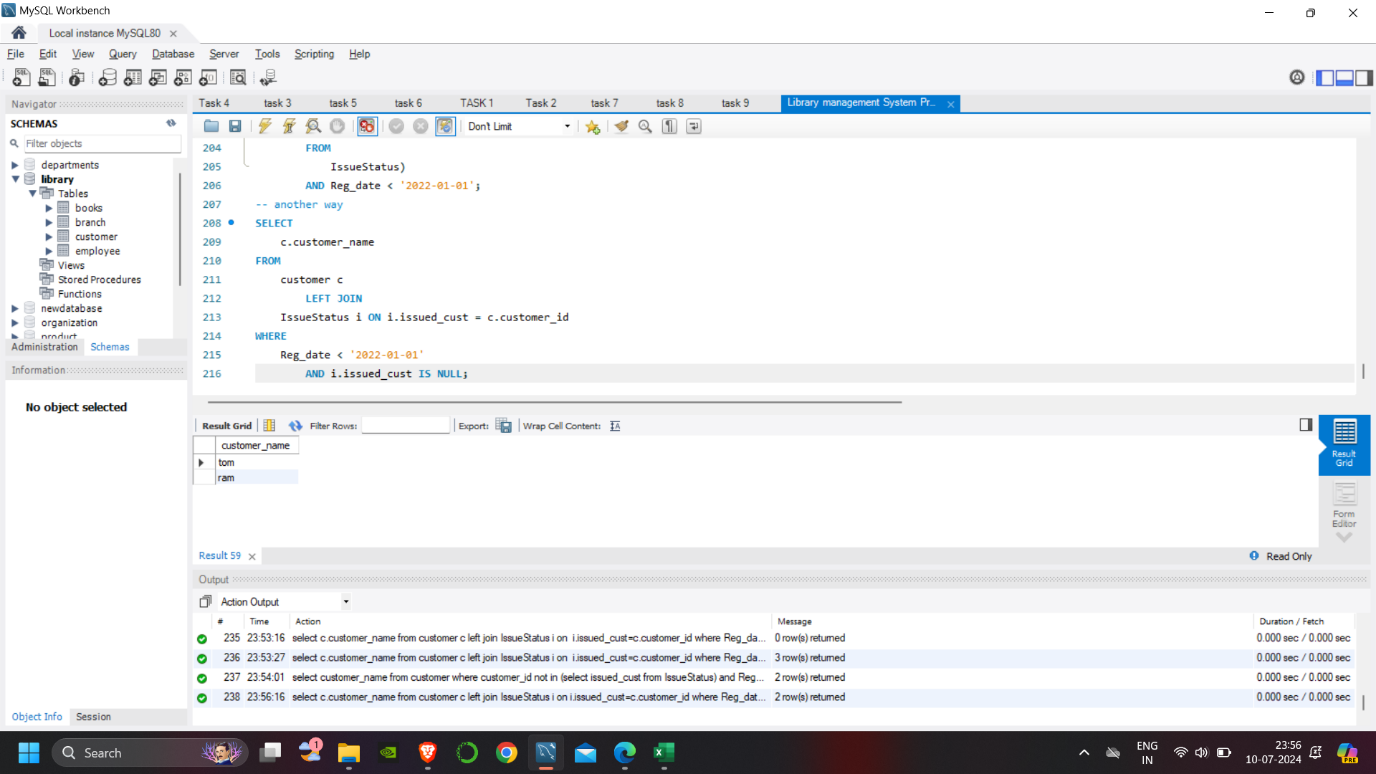
IssueStatus i ON i.issued\_cust = c.customer\_id

WHERE

Reg\_date < '2022-01-01'

AND i.issued\_cust IS NULL;



-- 7. Display the branch numbers and the total count of employees in each branch.

SELECT

b.Branch\_no, COUNT(e.Emp\_Id)

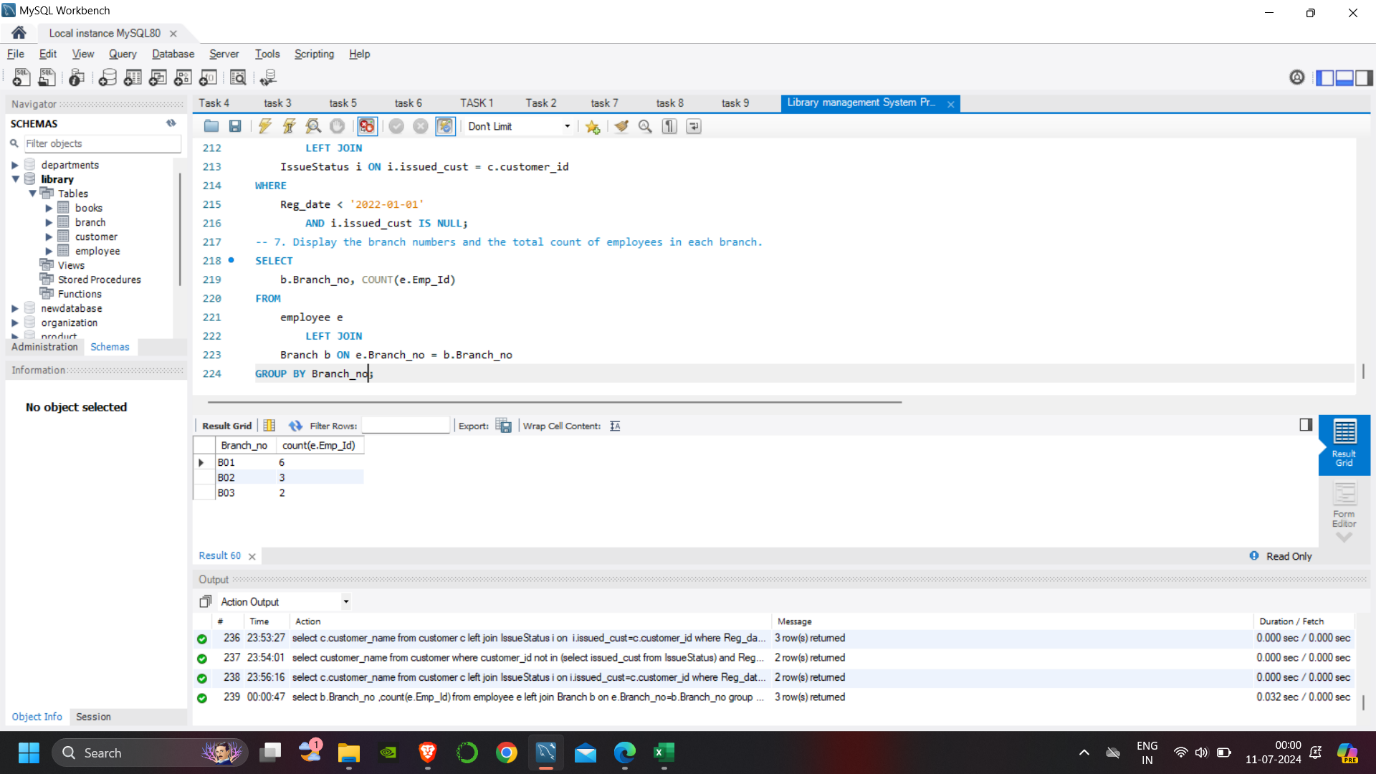
FROM

employee e

LEFT JOIN

Branch b ON e.Branch\_no = b.Branch\_no

GROUP BY Branch\_no;



-- 8. Display the names of customers who have issued books in the month of June 2023.

SELECT

c.customer\_name

FROM

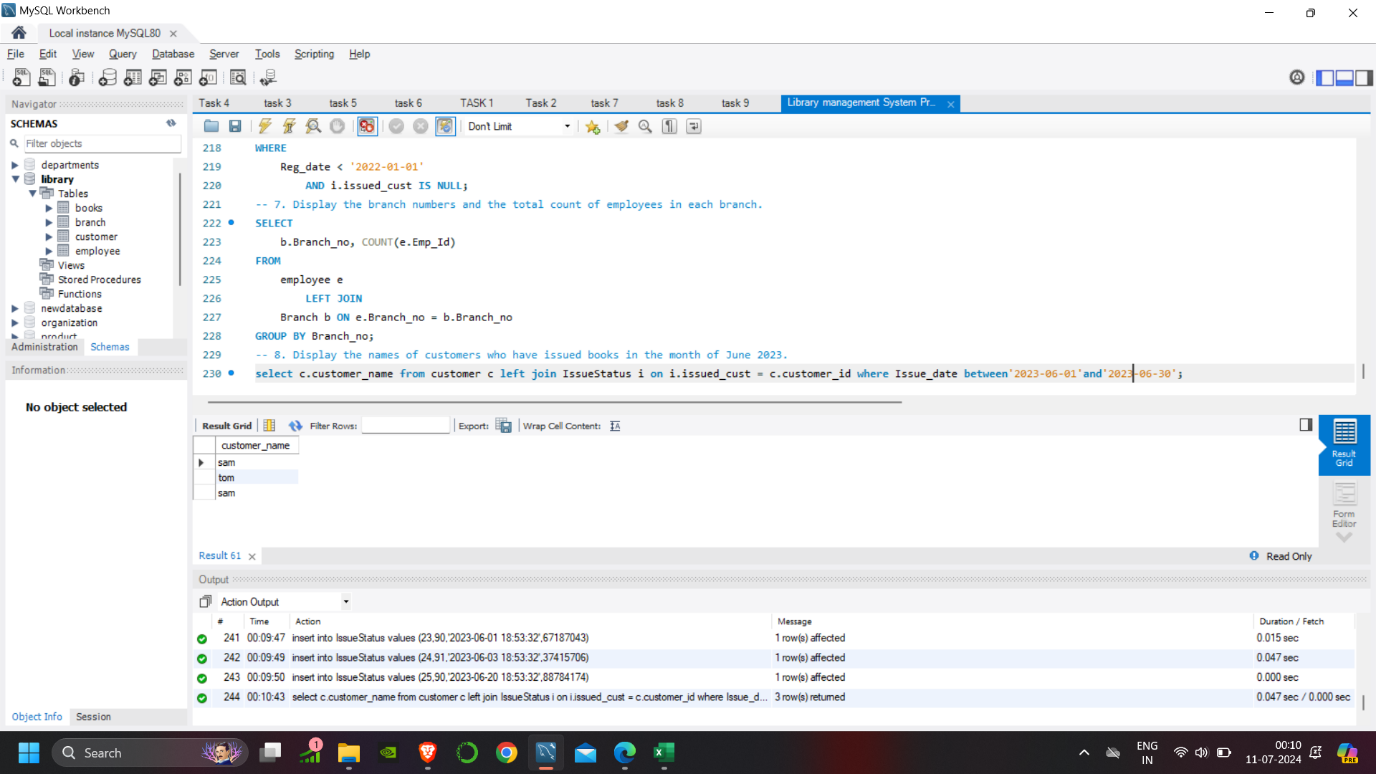
customer c

LEFT JOIN

IssueStatus i ON i.issued\_cust = c.customer\_id

WHERE

Issue\_date BETWEEN '2023-06-01' AND '2023-06-30';



-- 9. Retrieve book\_title from book table containing history.

SELECT

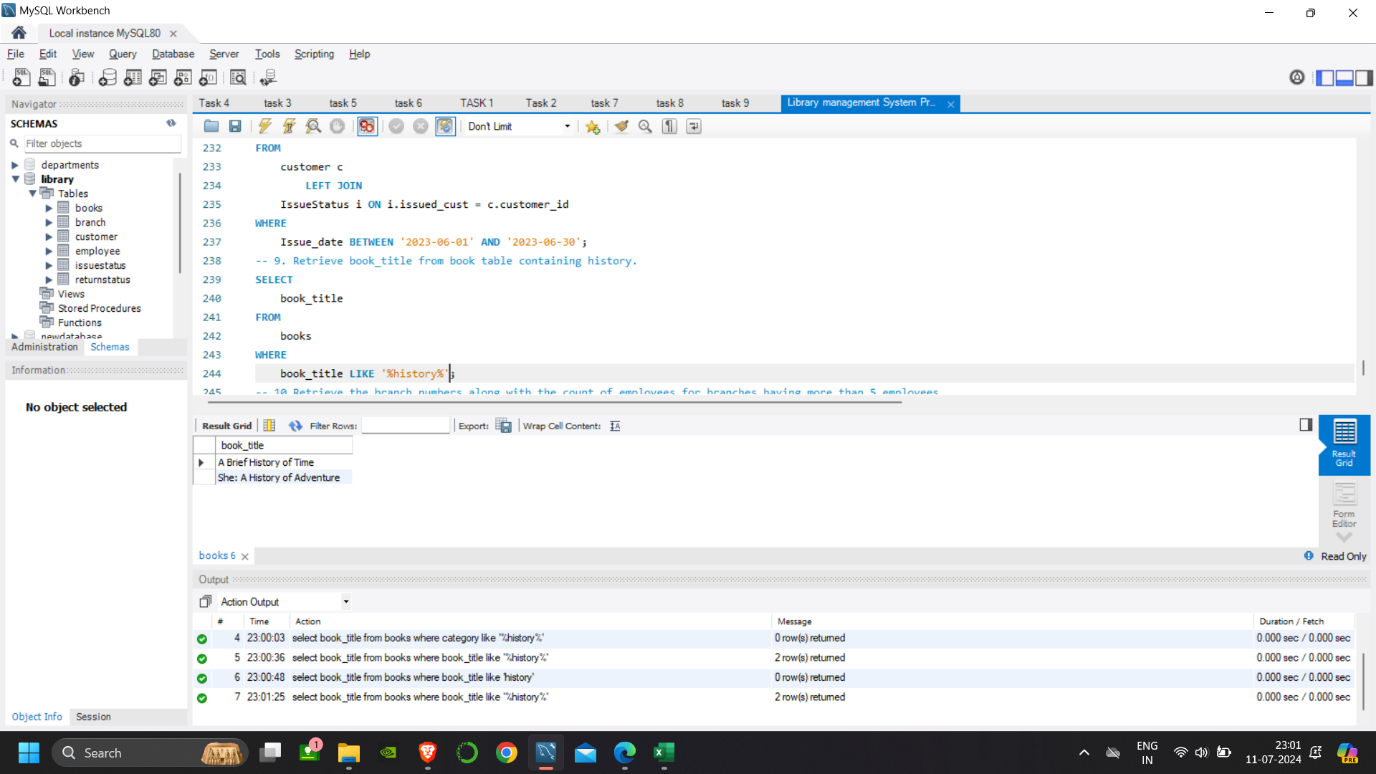
book\_title

FROM

books

WHERE

book\_title LIKE '%history%';



-- 10.Retrieve the branch numbers along with the count of employees for branches having more than 5 employees

SELECT

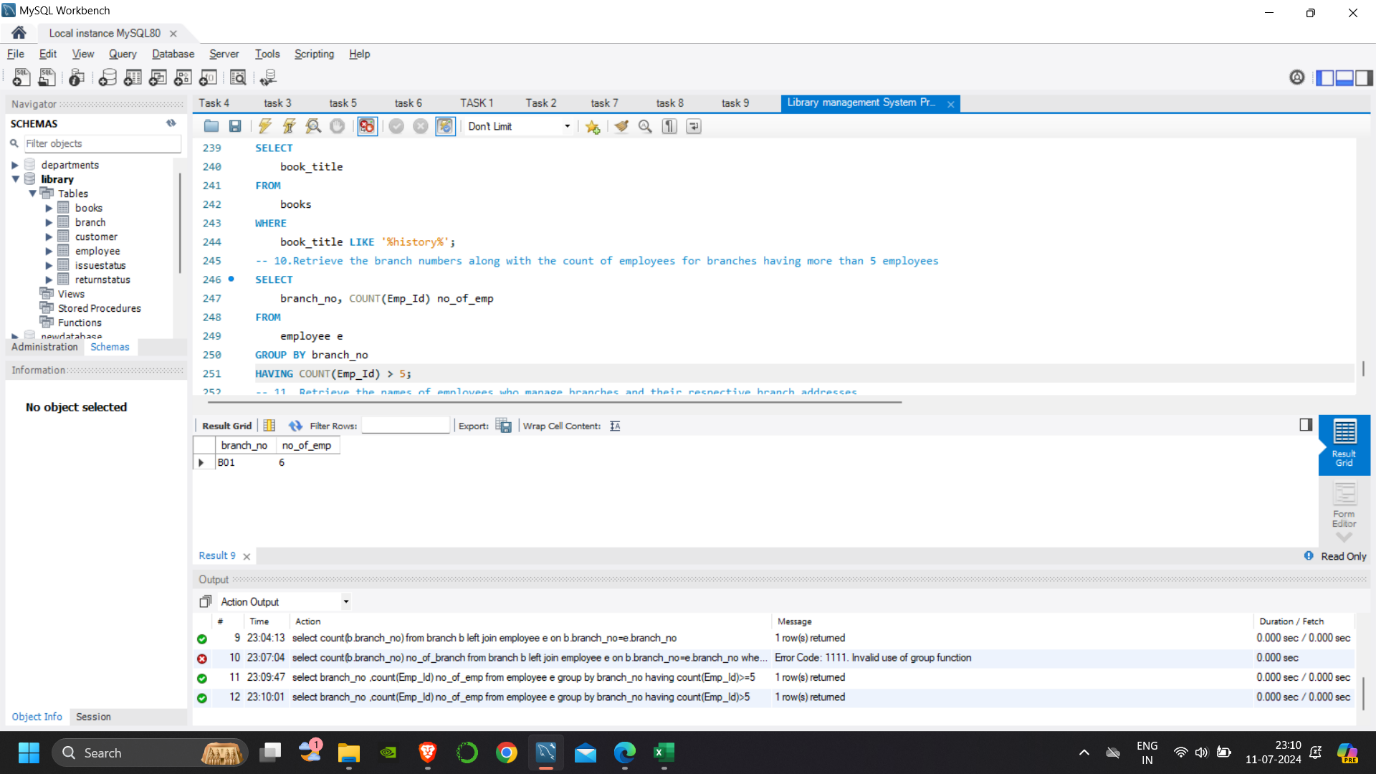
branch\_no, COUNT(Emp\_Id) no\_of\_emp

FROM

employee e

GROUP BY branch\_no

HAVING COUNT(Emp\_Id) > 5;



-- 11. Retrieve the names of employees who manage branches and their respective branch addresses.

SELECT

e.Emp\_name, b.Branch\_address, e.position

FROM

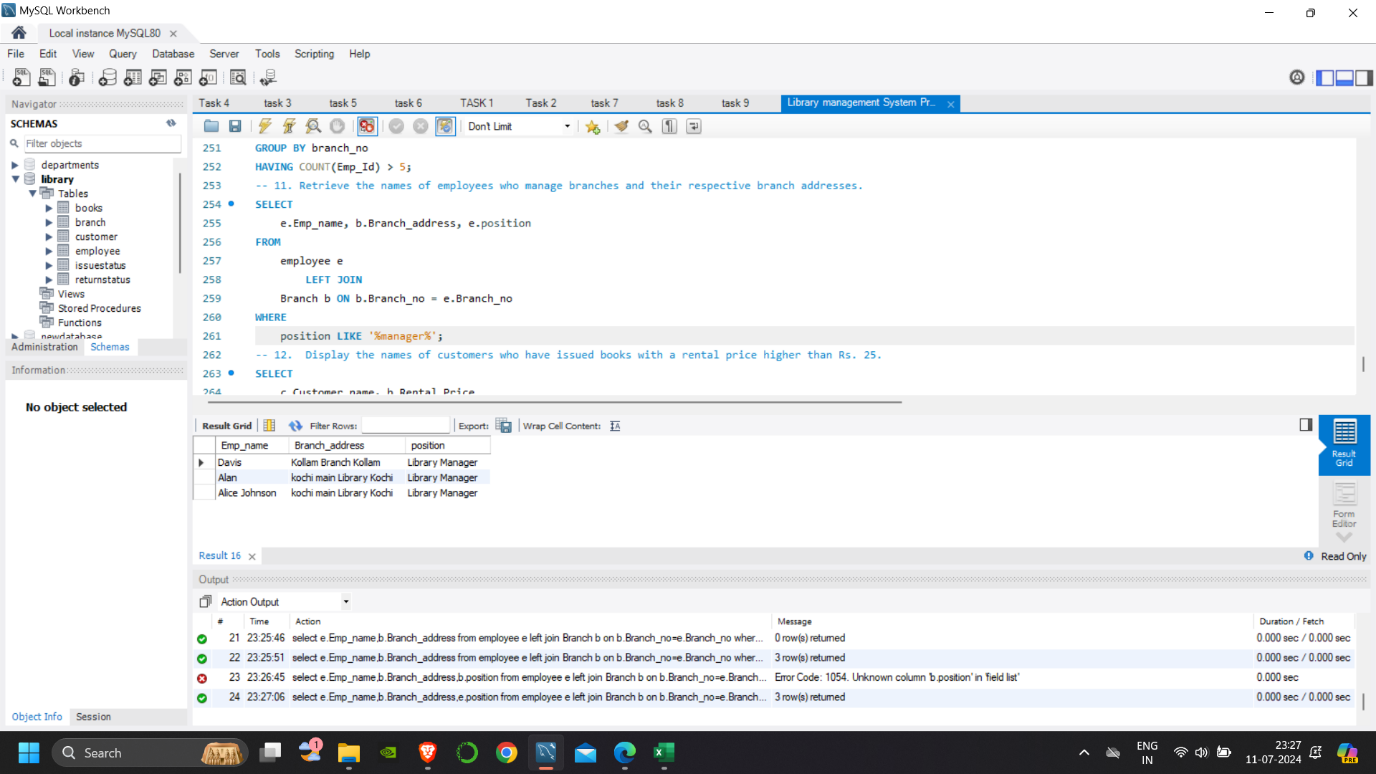
employee e

LEFT JOIN

Branch b ON b.Branch\_no = e.Branch\_no

WHERE

position LIKE '%manager%';



-- 12. Display the names of customers who have issued books with a rental price higher than Rs. 25.

SELECT

c.Customer\_name, b.Rental\_Price

FROM

IssueStatus i

LEFT JOIN

customer c ON c.Customer\_id = i.Issued\_cust

LEFT JOIN

books b ON b.ISBN = i.Isbn\_book

WHERE

Rental\_Price > 25;

