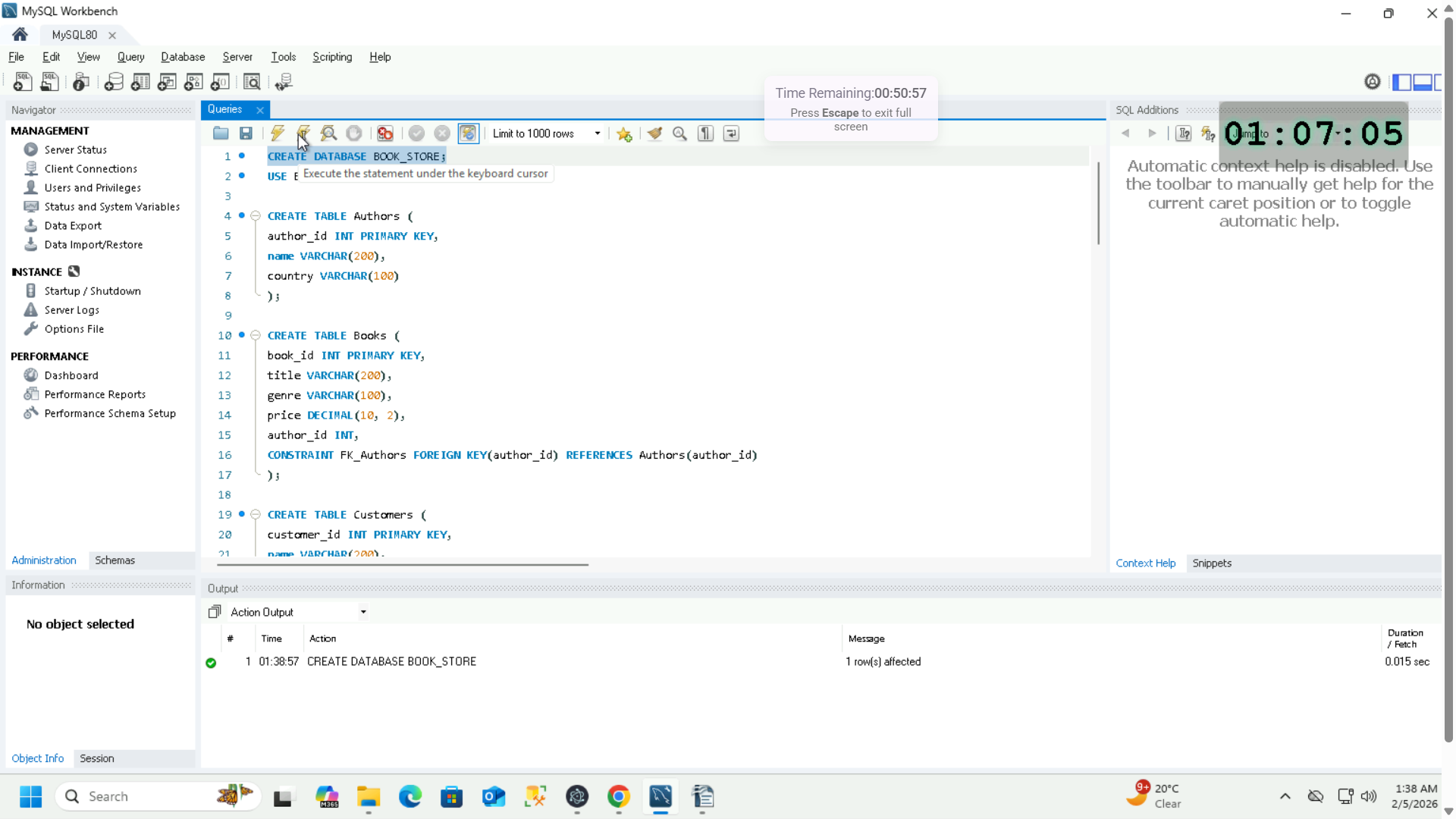
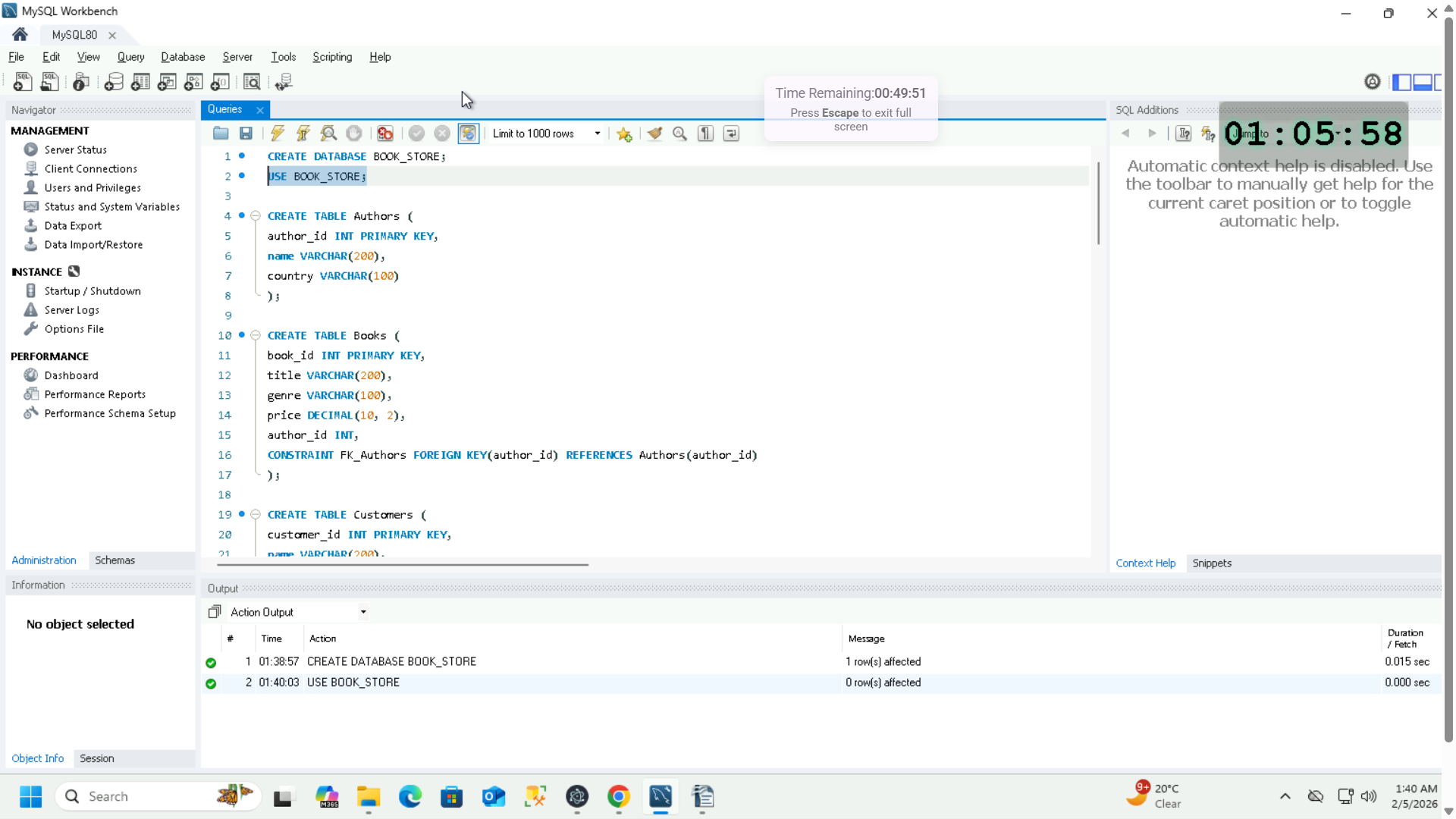
**Use Case 1:- (Create Queries)**

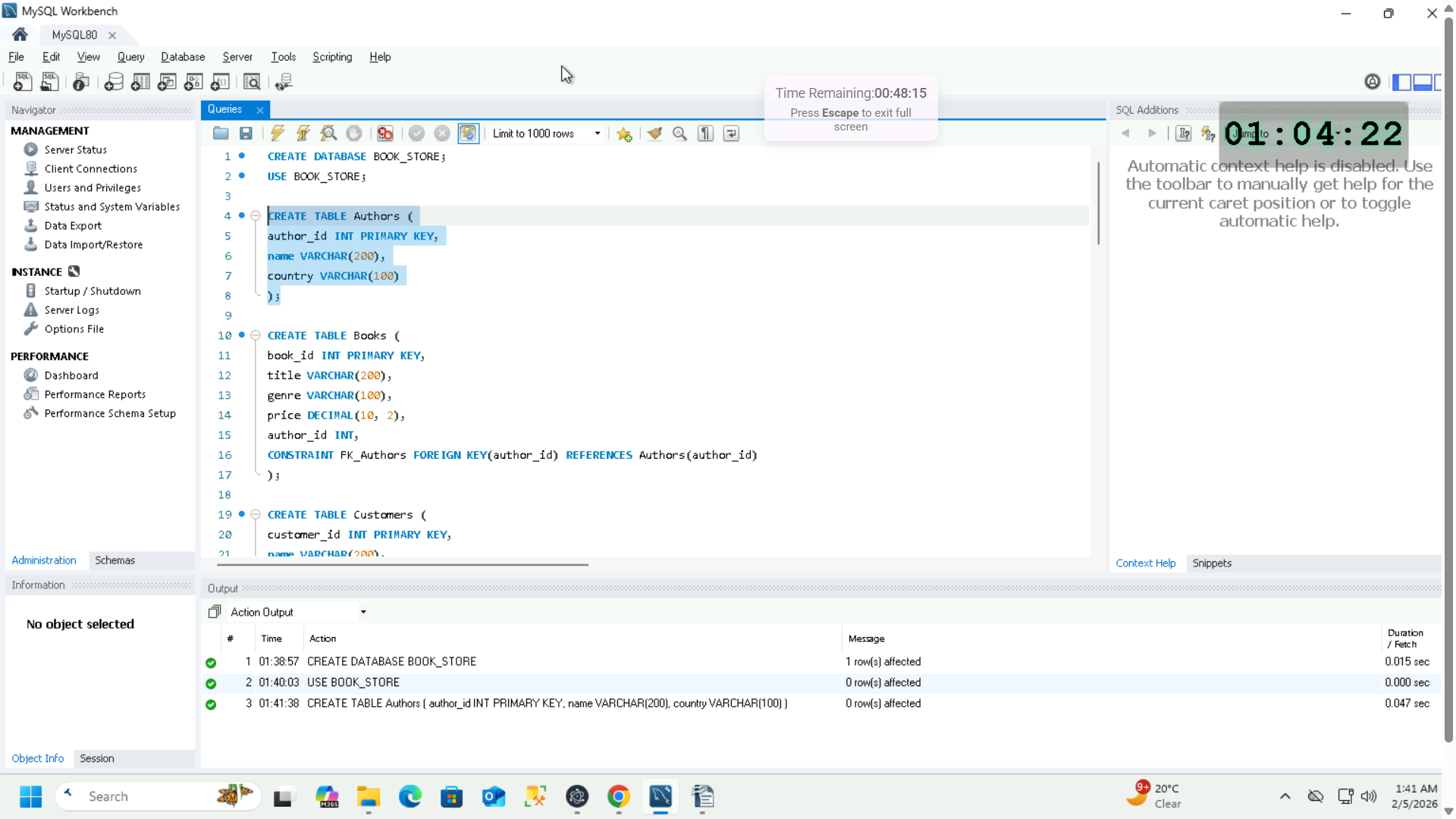
1. Creating database named Book\_Store  
   CREATE DATABASE BOOK\_STORE;



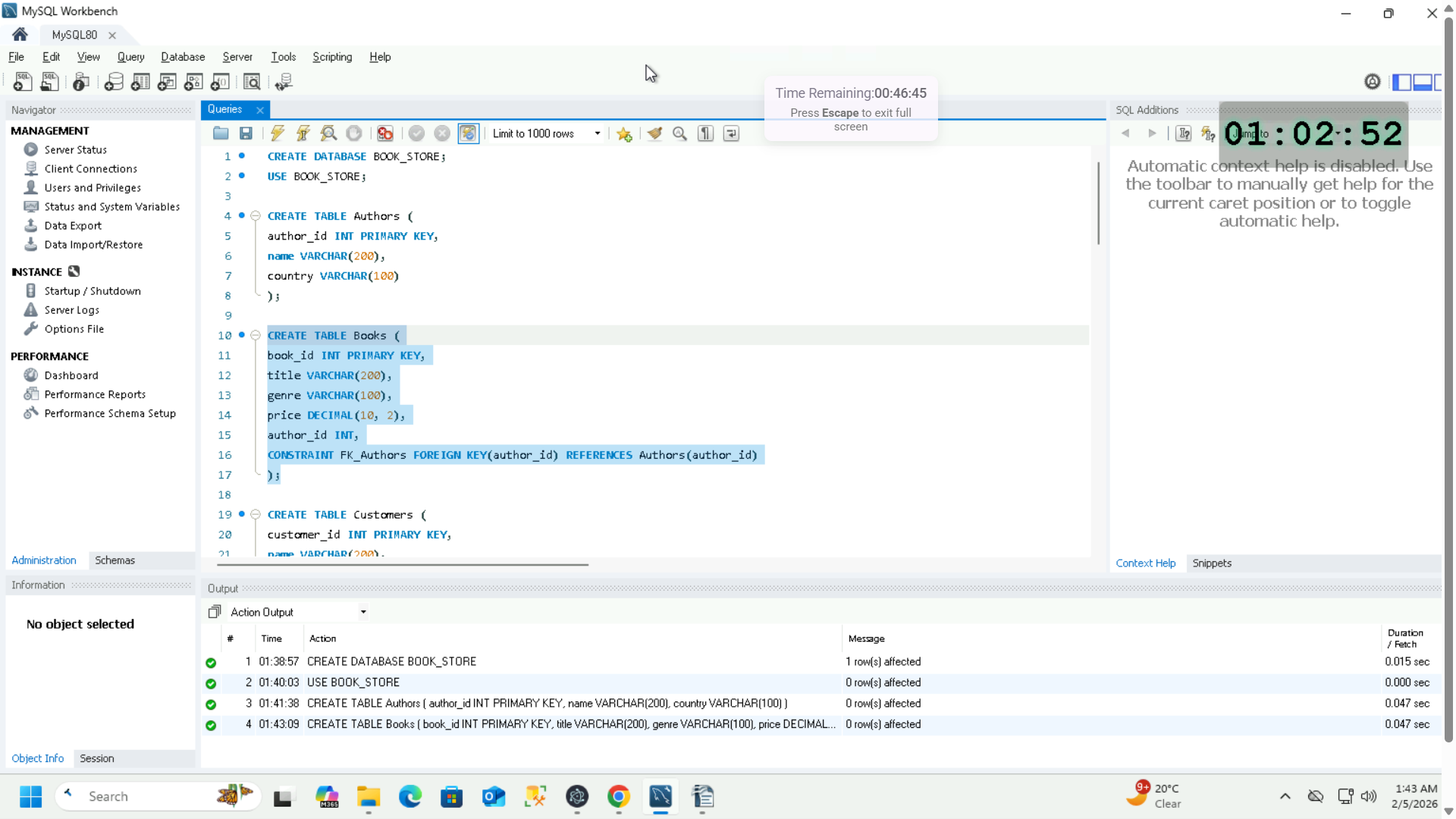
1. Now, use this created database  
   USE BOOK\_STORE;



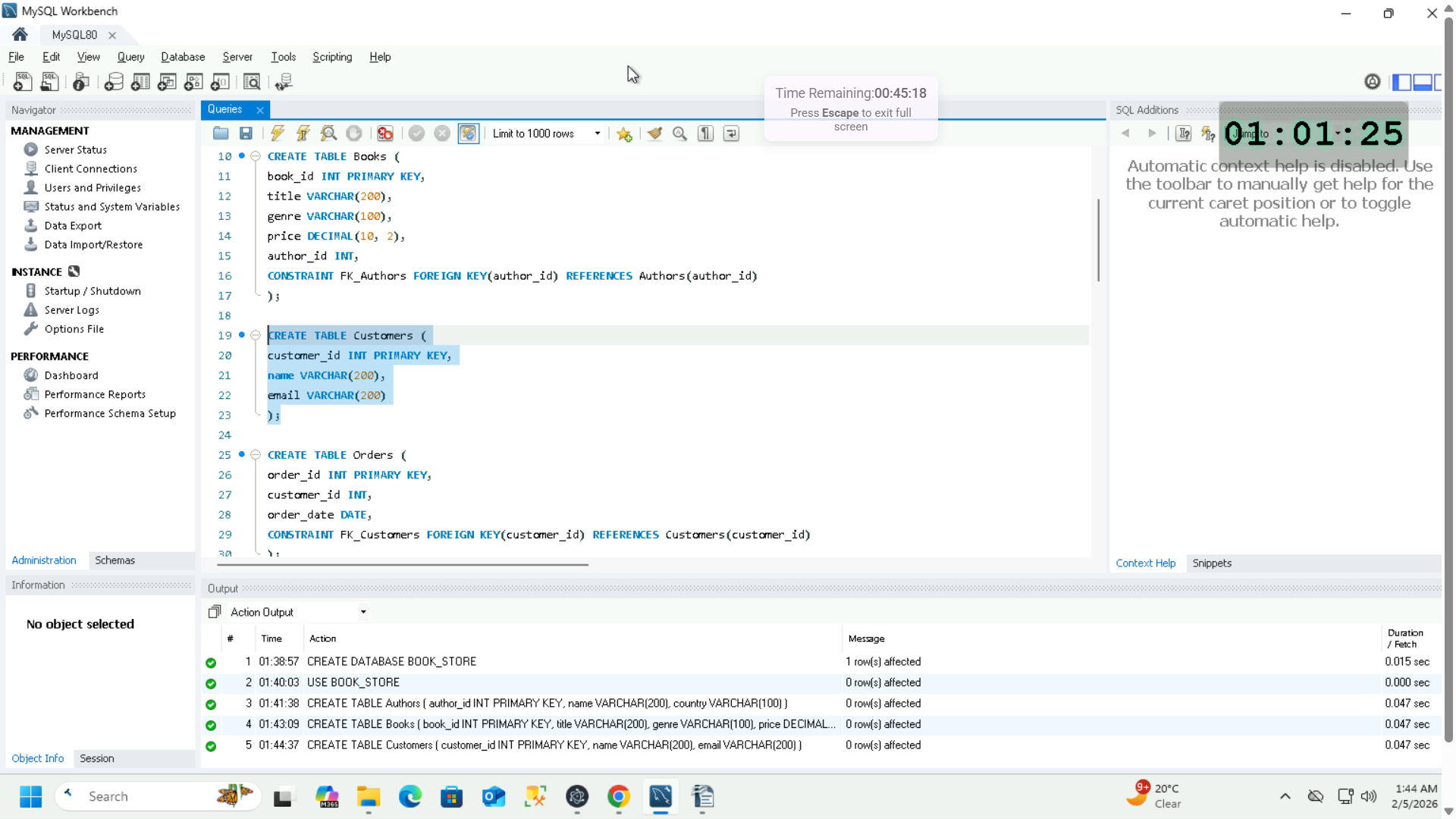
1. Create a table names Authors having primary key as author\_id  
   CREATE TABLE Authors (  
   author\_id INT PRIMARY KEY,  
   name VARCHAR(200),  
   country VARCHAR(100)  
   );



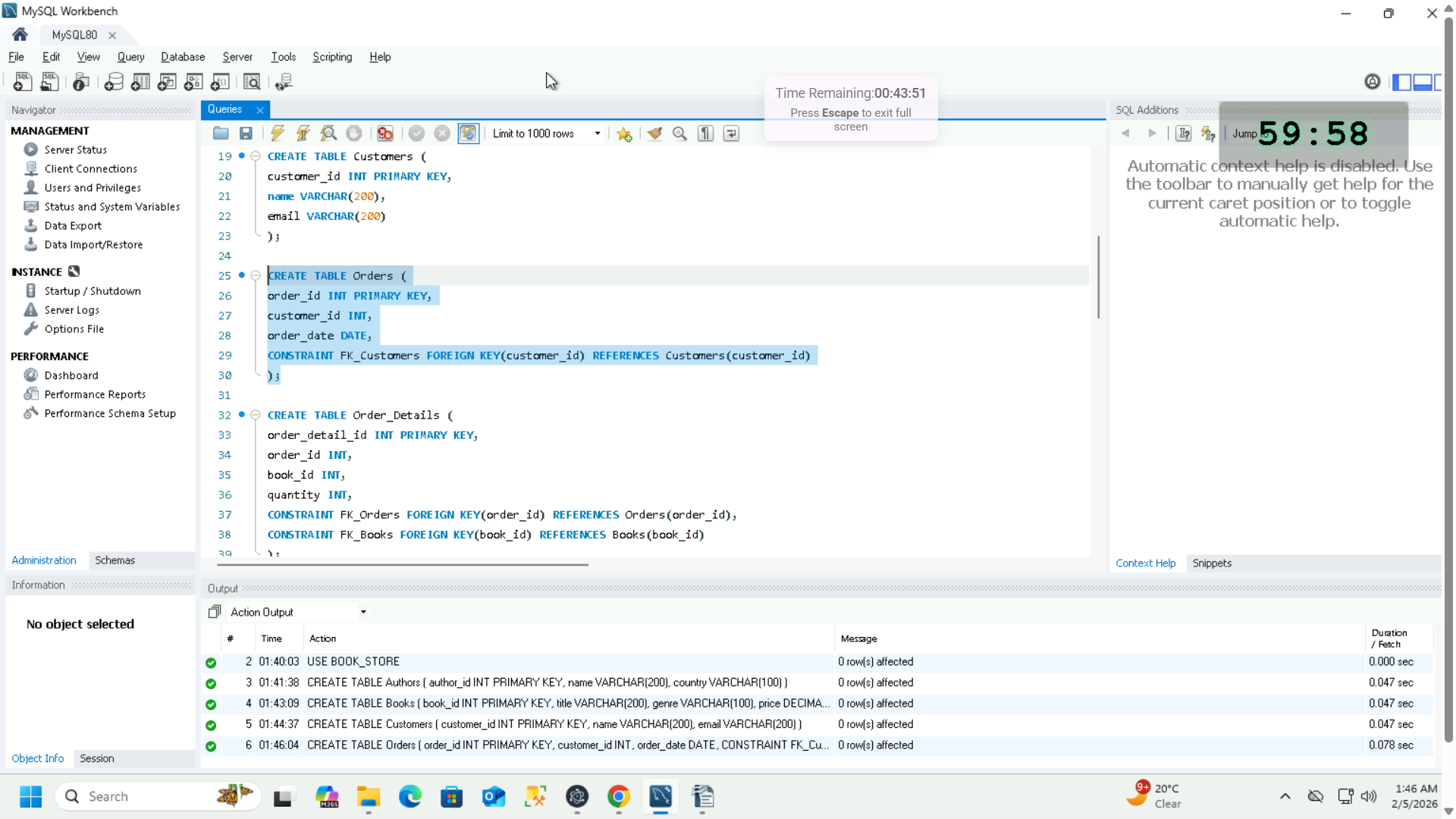
1. Create a table named Books having primary key as book\_id and foreign key as author\_id from the Authors table  
     
   CREATE TABLE Books (  
   book\_id INT PRIMARY KEY,  
   title VARCHAR(200),  
   genre VARCHAR(100),  
   price DECIMAL(10,2),  
   author\_id INT,  
   CONSTRAINT FK\_Authors FOREIGN KEY(author\_id) REFERENCES Authors(author\_ID)  
   );



1. Create a table named Customers having customer\_id as Primary key  
   CREATE TABLE Customers (  
   customer\_id INT PRIMARY KEY,  
   name VARCHAR(200),  
   email VARCHAR(200)  
   );



1. Create a table named Orders having order\_id as Primary key and customer\_id as Foreign Key from the Customers table  
   CREATE TABLE Orders (  
   order\_id INT PRIMARY KEY,  
   customer\_id INT,  
   order\_date DATE,   
   CONSTRAINT FK\_Customers FOREIGN KEY(customer\_id) REFERENCES Customers(customer\_id)  
   );



1. Create a table named Order\_Details having order\_detail\_id as Primary key and order\_id as Foreign Key from Orders table and book\_id as Foreign Key from Books table  
   CREATE TABLE Order\_Details (

order\_detail\_id INT PRIMARY KEY,

order\_id INT,

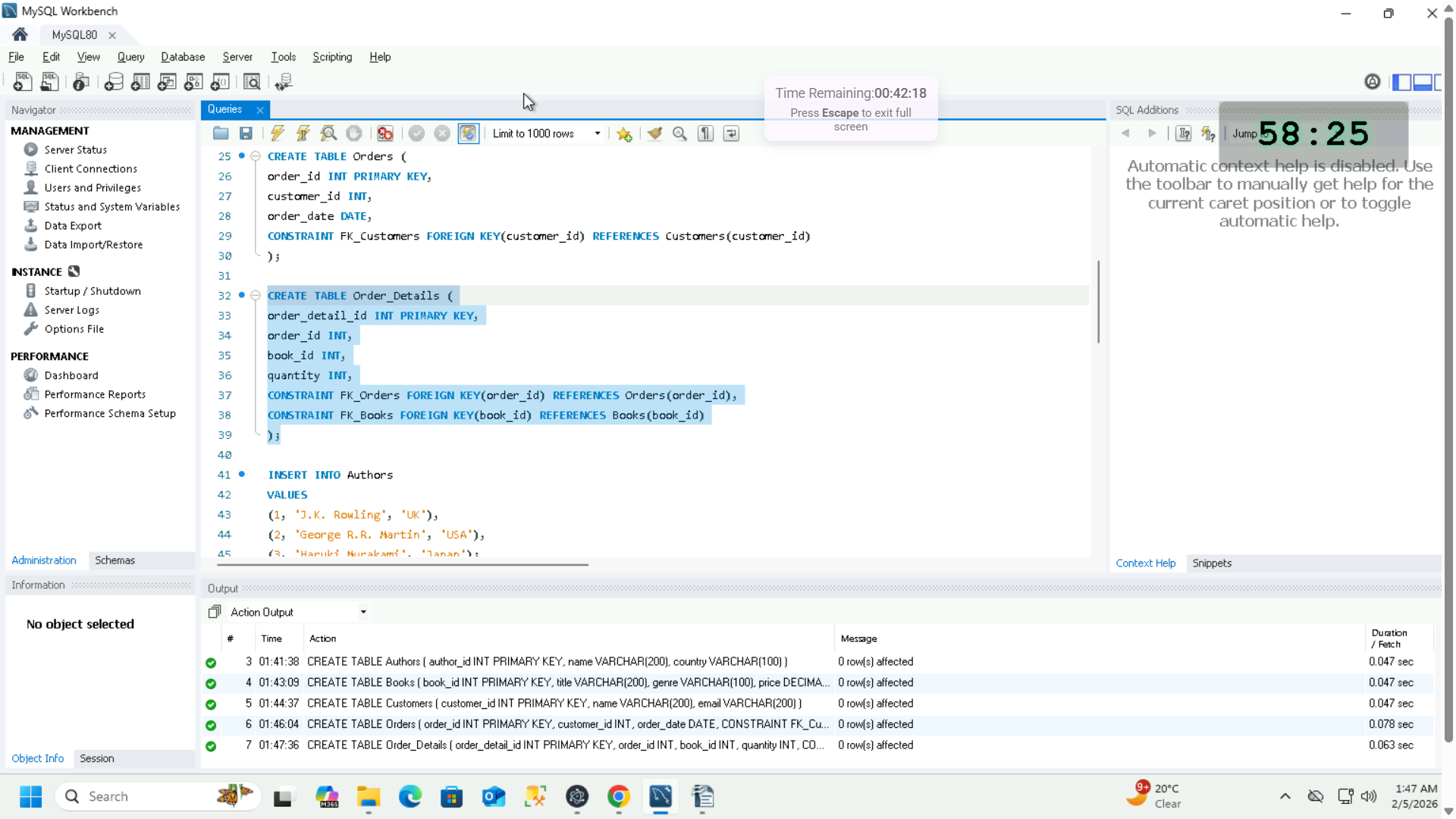
book\_id INT

Quantity INT,

CONSTRAINT FK\_Orders FOREIGN KEY(order\_id) REFERENCES Orders(order\_id),

CONSTRAINT FK\_Books FOREIGN KEY(book\_id) REFERENCES Books(book\_id)

);



**Use Case 2:- (Insert Queries)**

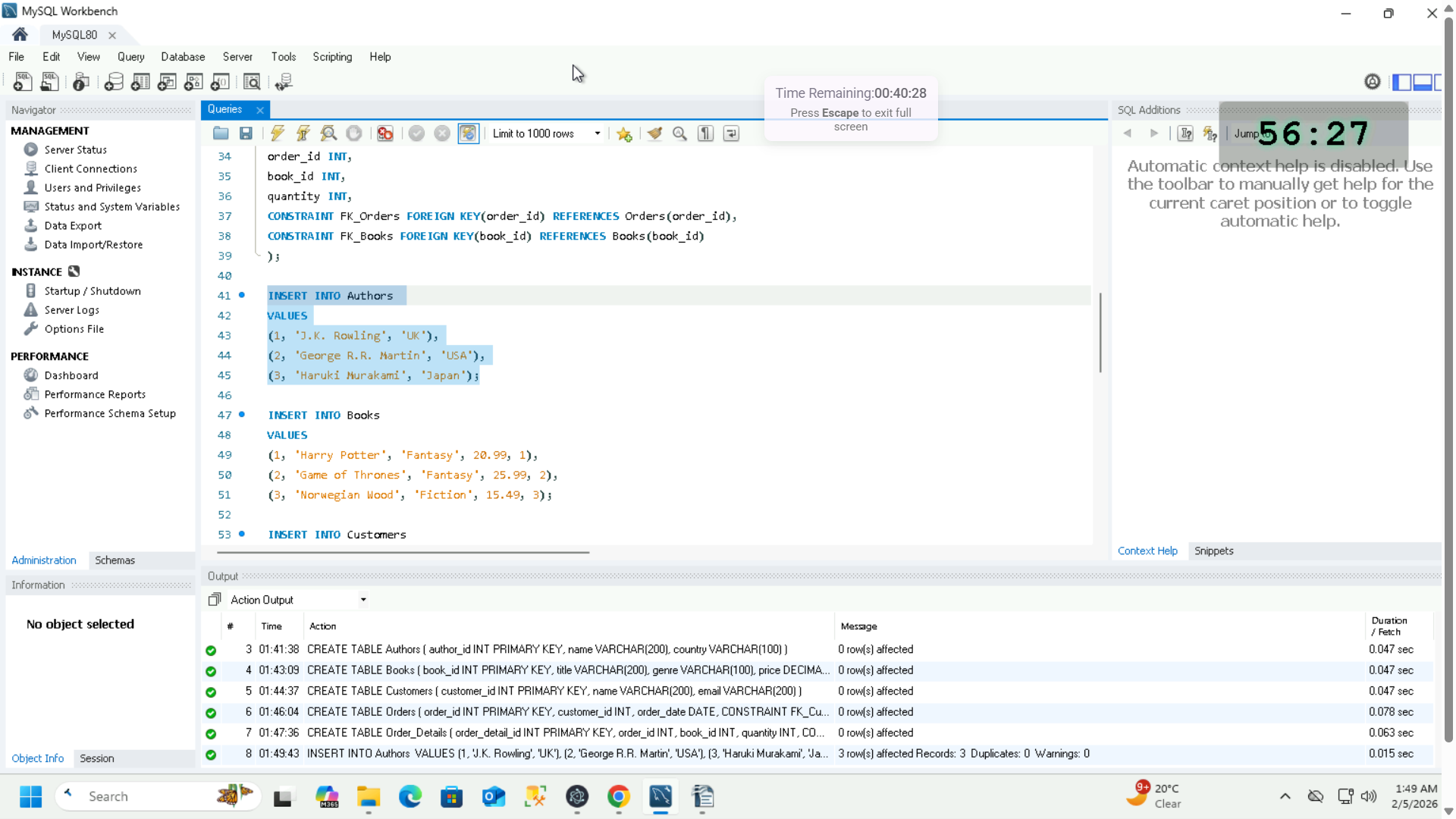
1. Insert Values into Authors table  
   INSERT INTO Authors

VALUES

(1, ‘J.K. Rowling’, ‘UK’),

(2, ‘George R.R. Martin’, ‘USA’),

(3, ‘Haruki Murakami’, ‘Japan’);



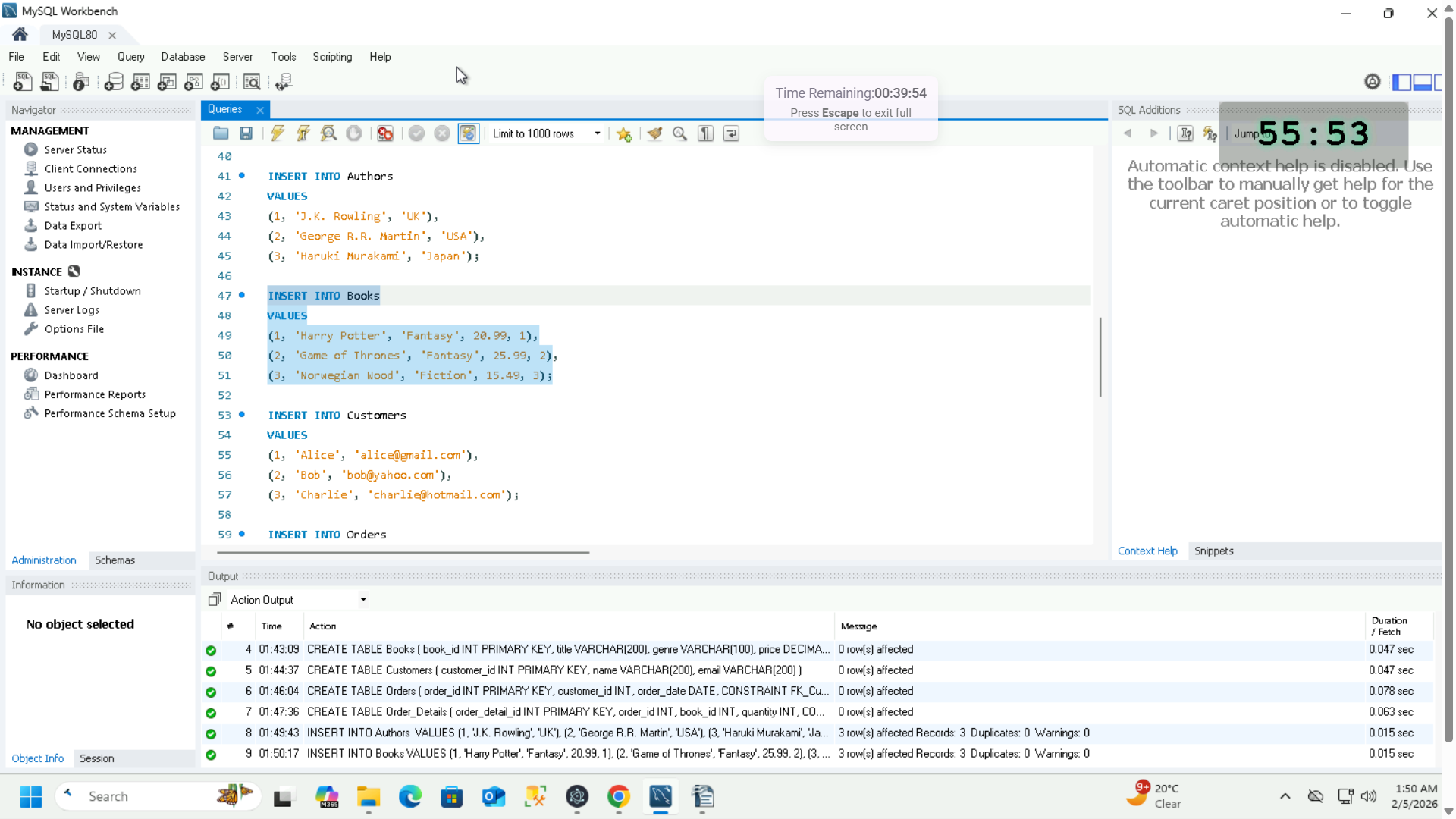
1. Insert Values into Books table  
   INSERT INTO Books

VALUES

(1, ‘Harry Potter’, ‘Fantasy’, 20.99, 1),

(2, ‘Game of Thrones’, ‘Fantasy’, 25.99, 2),

(3, ‘Norwegian Wood’, ‘Fiction’, 15.49, 3);



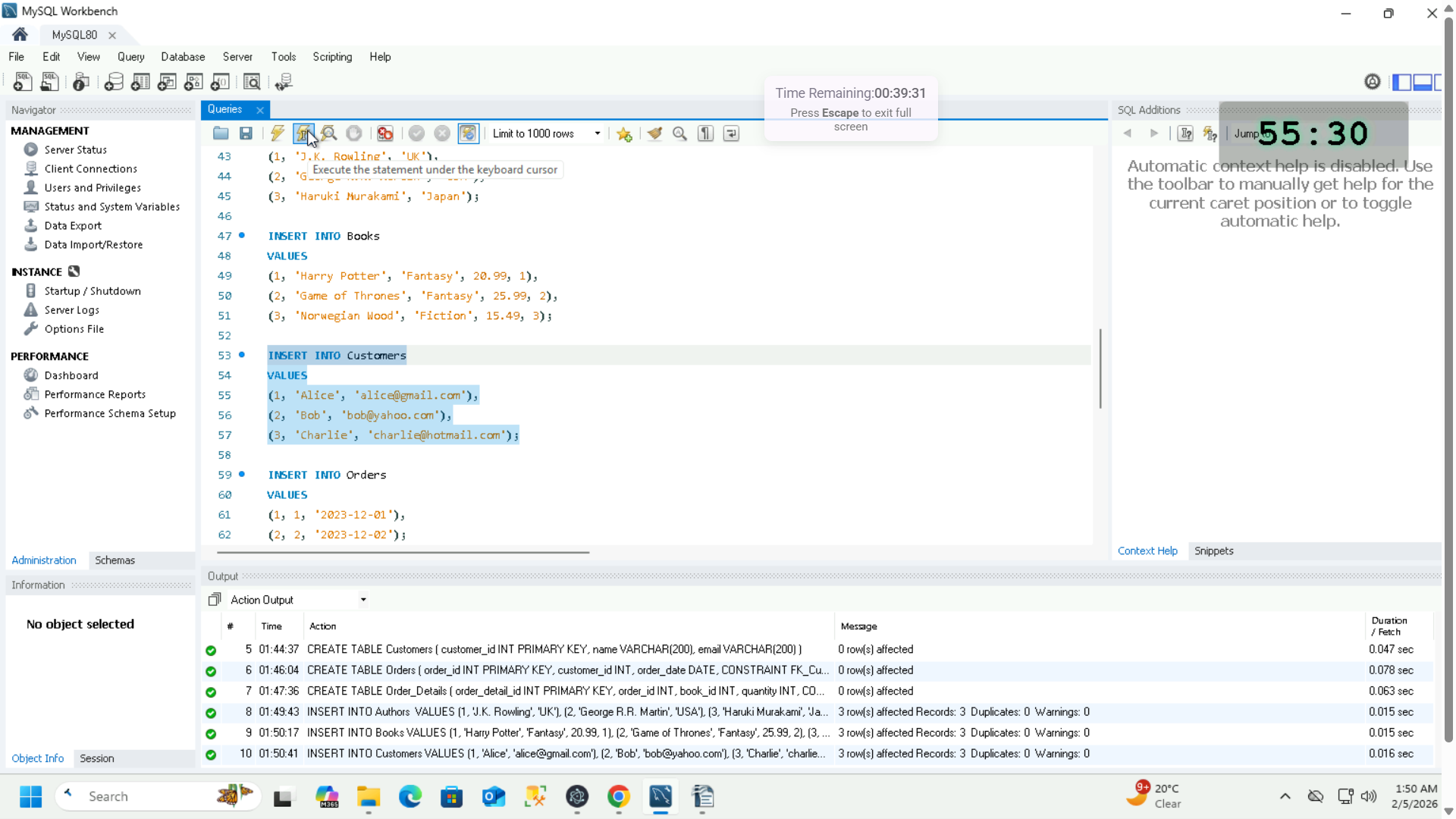
1. Insert Values into Customers table  
   INSERT INTO Customers

VALUES

(1, ‘Alice’, ‘alice@gmail.com’),

(2, ‘Bob’, ‘bob@yahoo.com’),

(3, ‘Charlie’, ‘charlie@hotmail.com’);

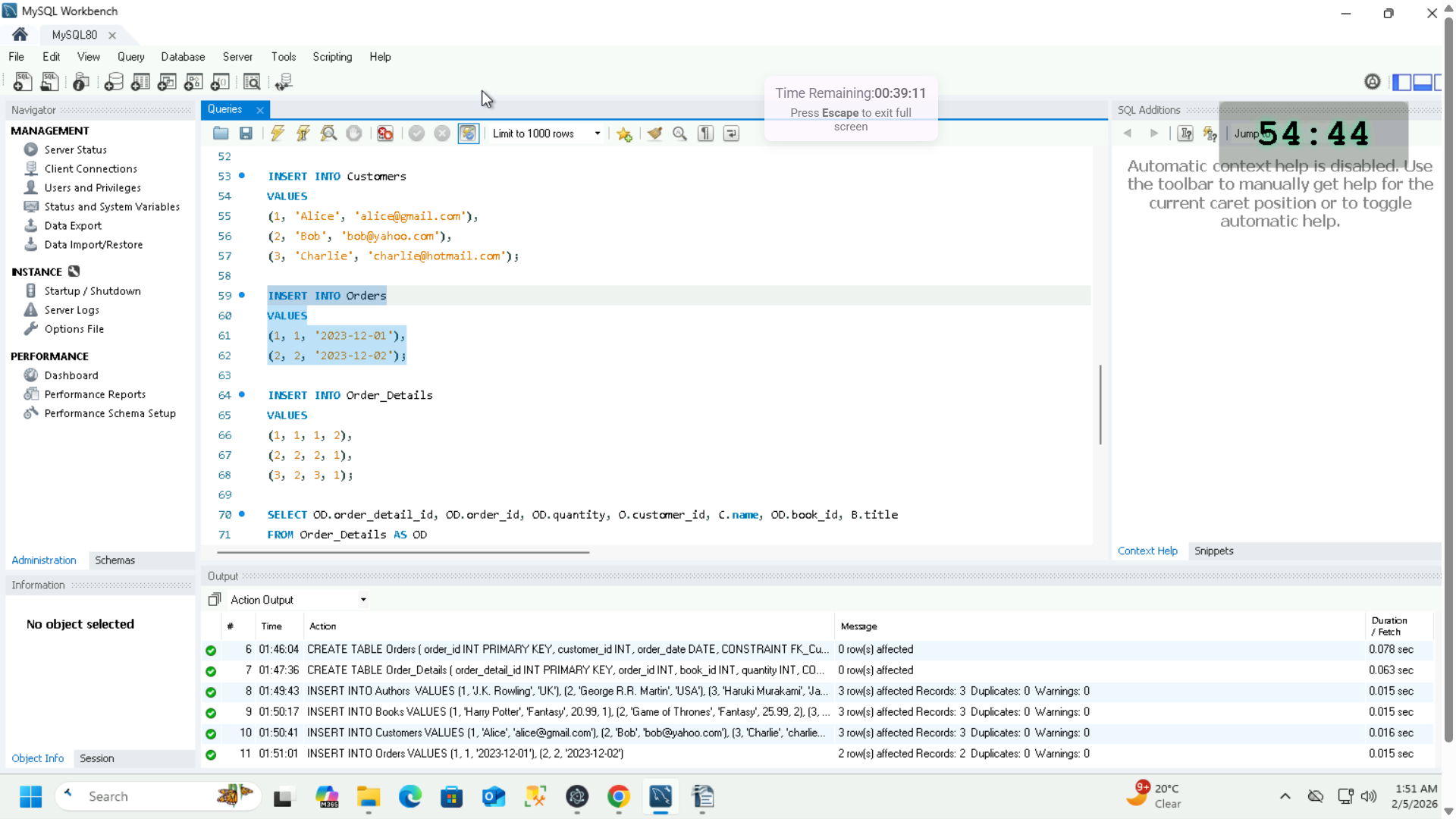


1. Insert Values into Orders table  
   INSERT INTO Orders

VALUES

(1, 1, ‘2023-12-01’),

(2, 2, ‘2023-12-02’);



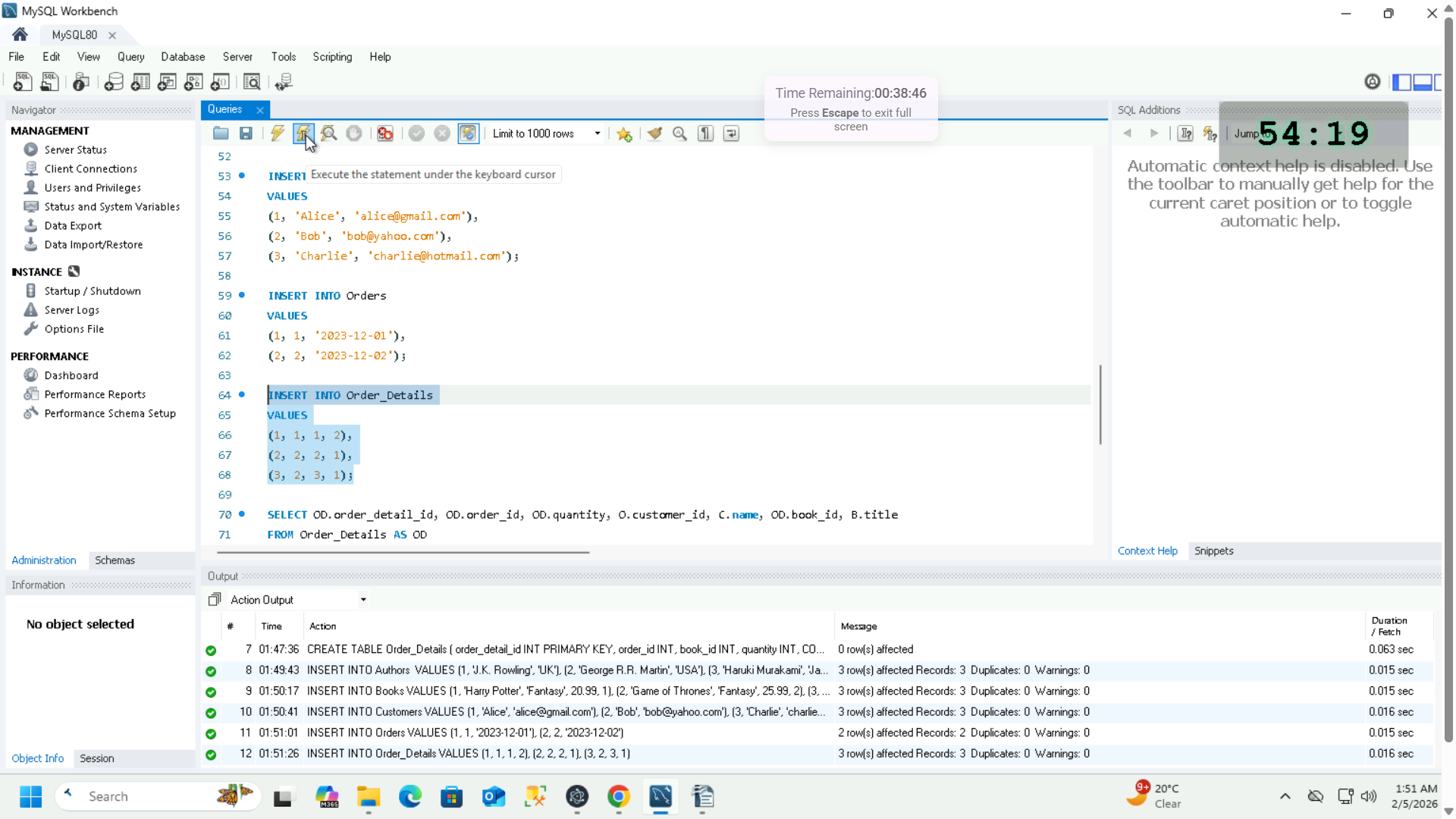
1. Insert Values into Order\_Details table  
   INSERT INTO Order\_Details

VALUES

(1, 1, 1, 2),

(2, 2, 2, 1),

(3, 2, 3, 1);



**Use Case 3:- (Joins)**

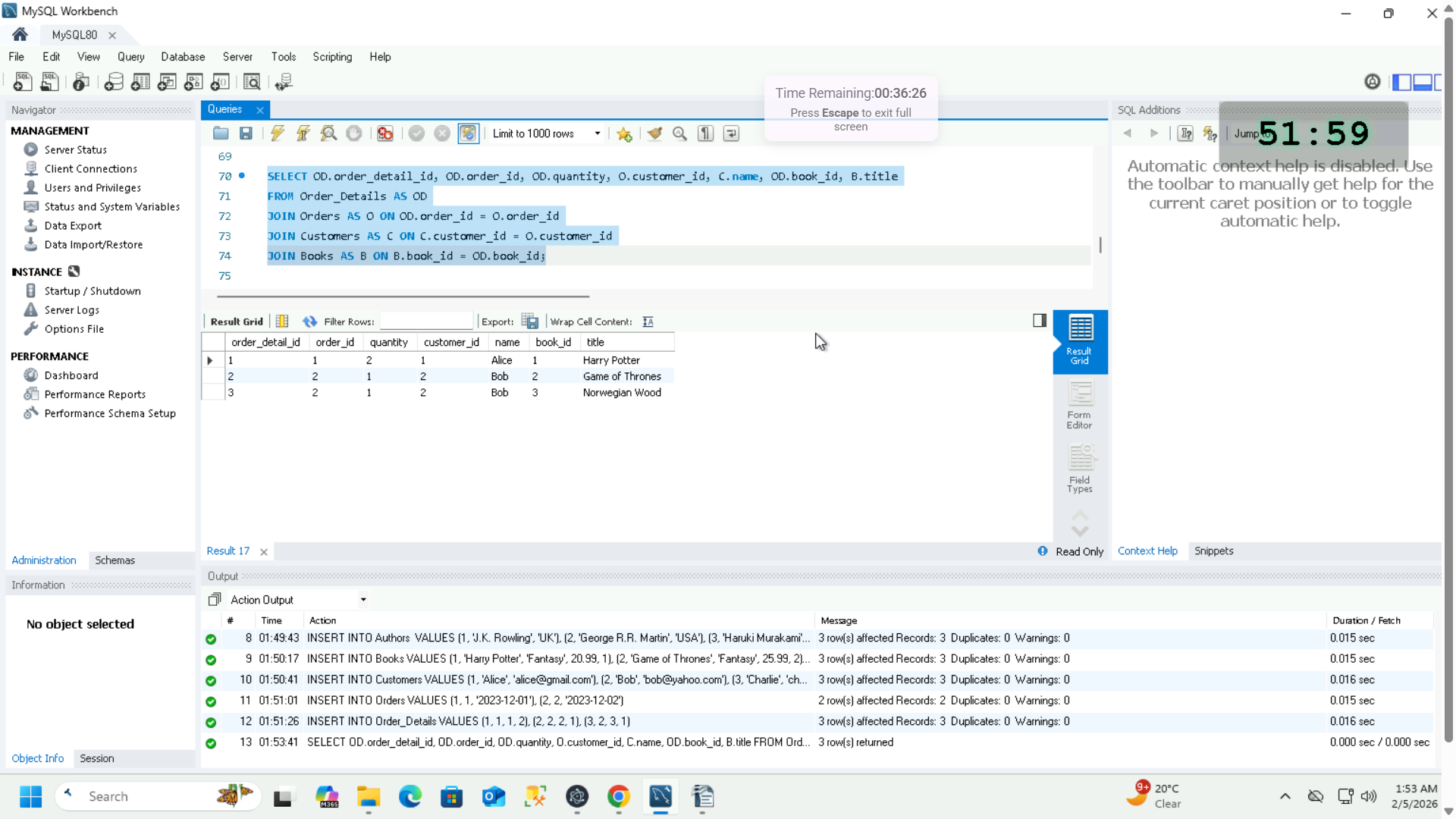
1. Joining Order\_details, Orders, Books & Customers tables using their Foreign Keys to display all orders along with the names of customers and the title of books they ordered  
   SELECT OD.order\_detail\_id, OD.order\_id, OD.quantity, O.customer\_id, C.name, OD.book\_id, B.title

FROM Order\_Details AS OD

JOIN Orders AS O ON OD.order\_id = O.order\_id

JOIN Customers AS C ON C.customer\_id = O.customer\_ID

JOIN Books AS B ON B.book\_id = OD.book\_id;

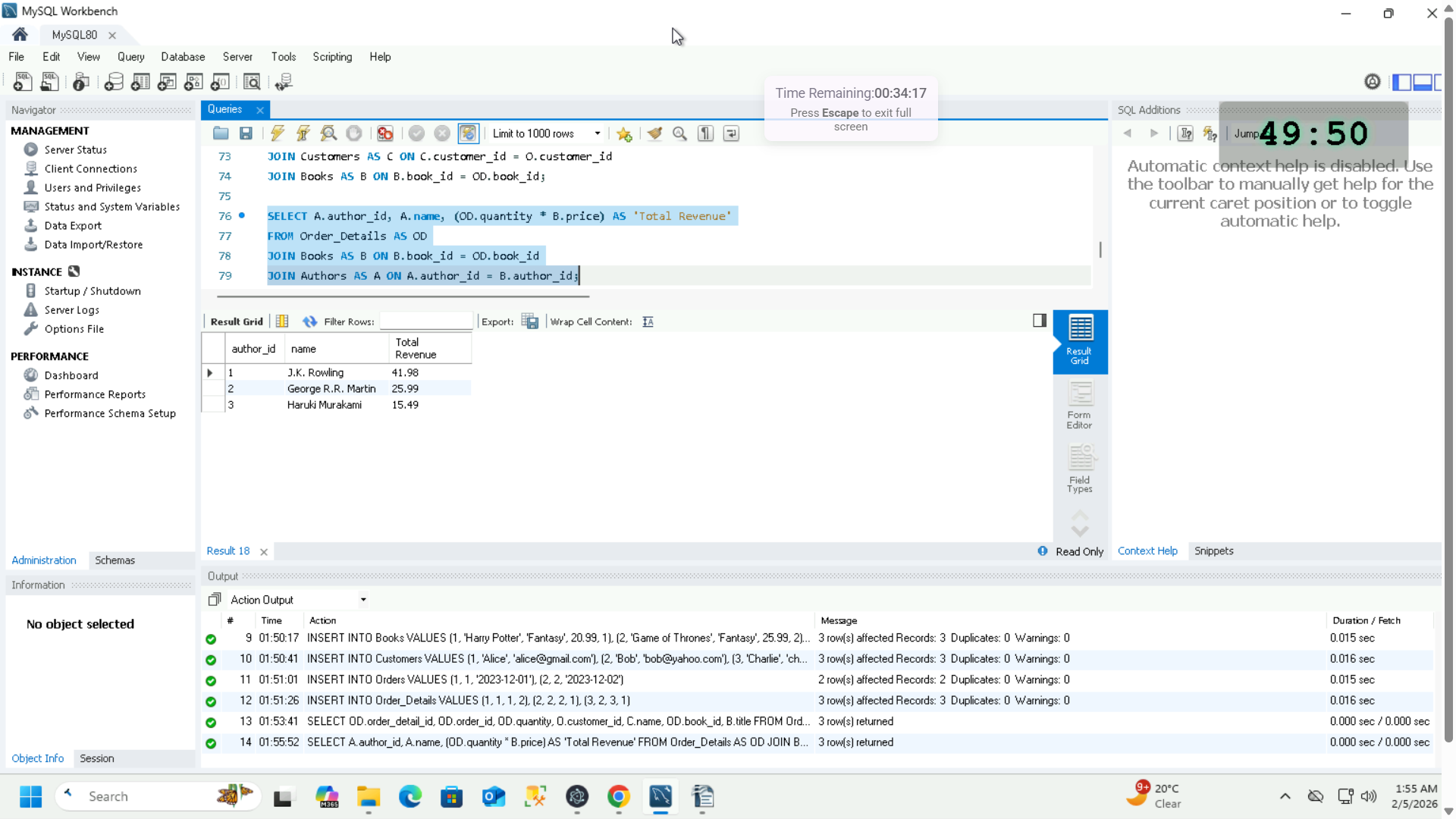


1. Joining Order\_details, Books & Authors tables using their Foreign keys to display the list of authors and their total revenue generated from their books, which is book price multiplied by the quantity of books sold displayed under Total Revenue column  
   SELECT A.author\_id, A.name, (OD.quantity \* B.price) AS ‘Total Revenue’

FROM Order\_Details AS OD

JOIN Books AS B ON B.book\_id = OD.book\_id

JOIN Authors AS A ON A.author\_id = B.author\_id;



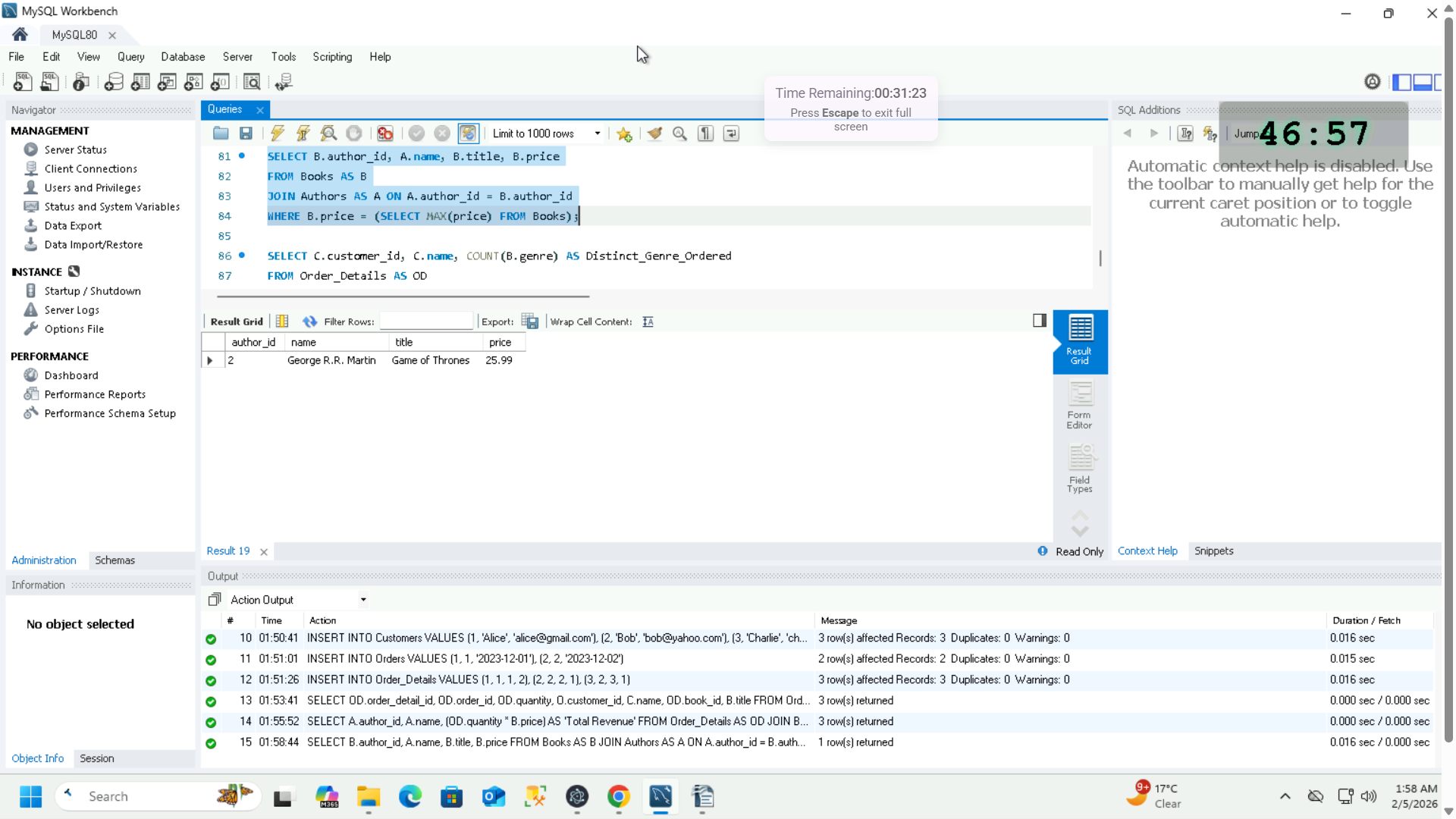
**Use Case 4:- (Sub Queries)**

1. Joining Books & Authors tables using foreign key and inside where condition there is a subquery which outputs the maximum price among all book prices. MAX is an aggregate function.  
   SELECT B.author\_id, A.name, B.title, B.price

FROM Books AS B

JOIN Authors AS A ON A.author\_id = B.author\_id

WHERE B.price = (SELECT MAX(price) FROM Books);



1. Joining Order\_details, Books, Orders & Customers tables using their foreign keys to List customers who have ordered more than one type of book genre. This is done using the COUNT() aggregate function which operated on book genre and counts its occurrences. Next, the Group By clause groups similar Customer ids & names based on the condition that the occurrences of book genre for a specific customer Is more than 1  
     
   SELECT C.customer\_id, C.name, COUNT(B.genre) AS Distinct\_Genre\_Ordered  
   FROM Order\_Details AS OD  
   JOIN Orders AS O ON O.order\_id = OD.order\_id  
   JOIN Books AS B ON OD.book\_id = B.book\_id  
   JOIN Customers AS C ON C.customer\_id = O.customer\_id  
   GROUP BY C.customer\_id, C.name  
   HAVING Distinct\_Genre\_Ordered > 1;

