**My SQL Task**

**Create the ecommerce database**

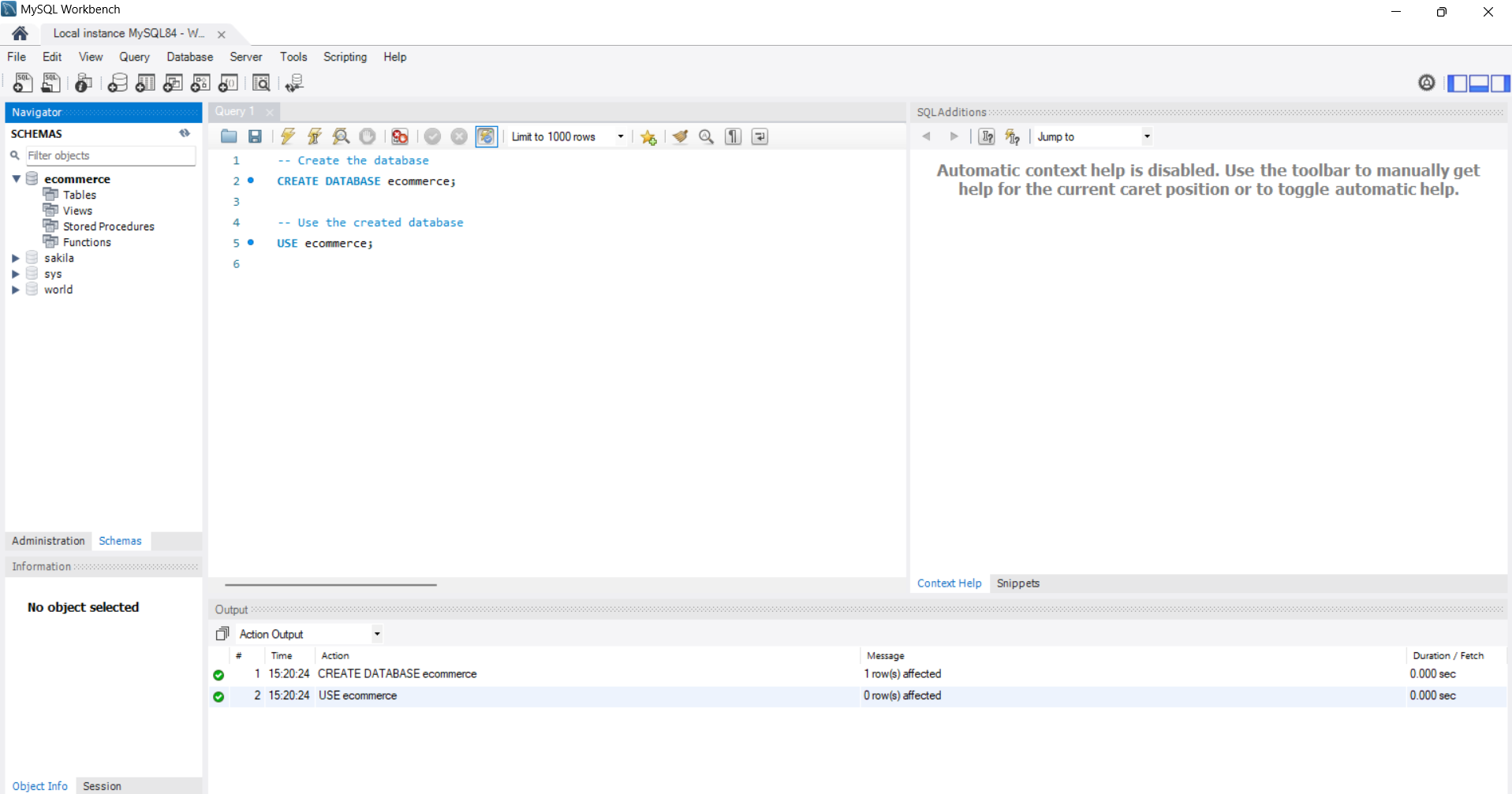
-- Create the database

CREATE DATABASE ecommerce;

-- Use the created database

USE ecommerce;

**Output:**



**Create the customers, orders, and products tables**

-- Create the 'customers' table

CREATE TABLE customers (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

address VARCHAR(255) NOT NULL

);

-- Create the 'orders' table

CREATE TABLE orders (

id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

order\_date DATE NOT NULL,

total\_amount DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (customer\_id) REFERENCES customers(id)

);

-- Create the 'products' table

CREATE TABLE products (

id INT AUTO\_INCREMENT PRIMARY KEY,

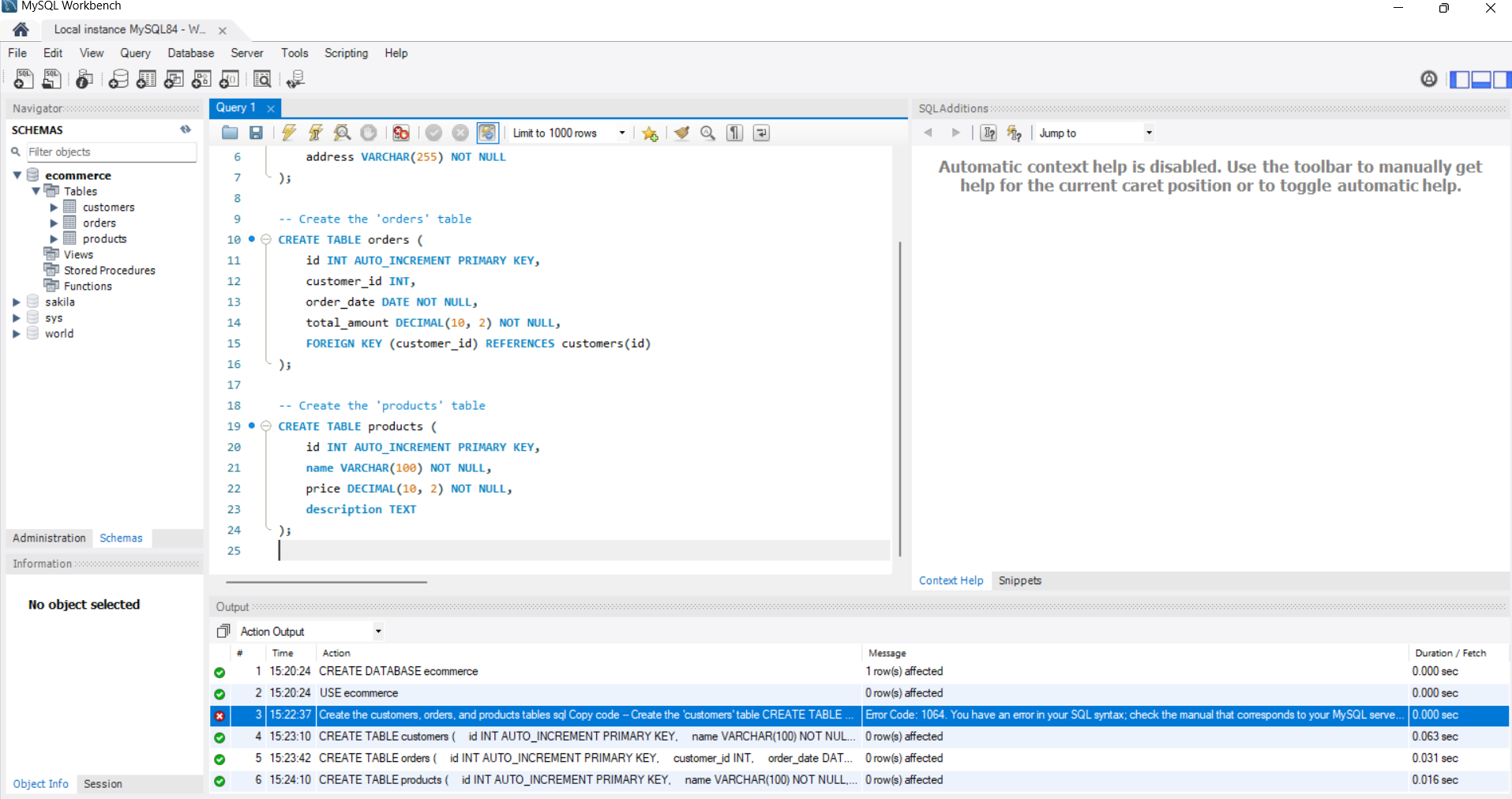
name VARCHAR(100) NOT NULL,

price DECIMAL(10, 2) NOT NULL,

description TEXT

);

**Output:**



**Insert Sample Data into the Tables**

-- Insert sample data into the 'customers' table

INSERT INTO customers (name, email, address) VALUES

('Alice Johnson', 'alice@example.com', '123 Maple St'),

('Bob Smith', 'bob@example.com', '456 Oak Ave'),

('Charlie Brown', 'charlie@example.com', '789 Pine Rd');

-- Insert sample data into the 'orders' table

INSERT INTO orders (customer\_id, order\_date, total\_amount) VALUES

(1, '2024-10-15', 120.50),

(2, '2024-10-16', 200.00),

(3, '2024-10-10', 75.30);

-- Insert sample data into the 'products' table

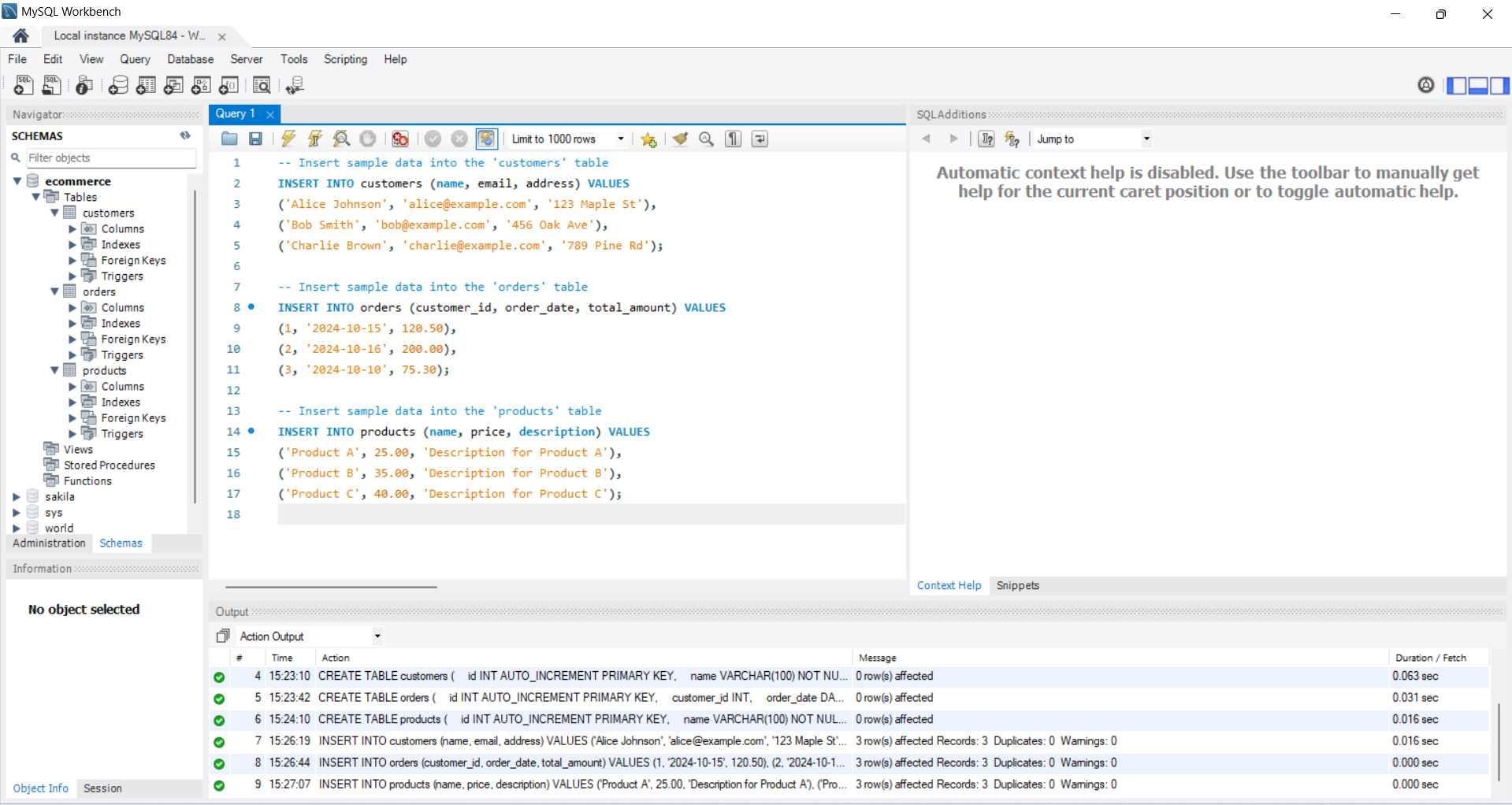
INSERT INTO products (name, price, description) VALUES

('Product A', 25.00, 'Description for Product A'),

('Product B', 35.00, 'Description for Product B'),

('Product C', 40.00, 'Description for Product C');

**Output:**



**Queries**

**Retrieve all customers who have placed an order in the last 30 days**

-- Retrieve customers who placed an order in the last 30 days

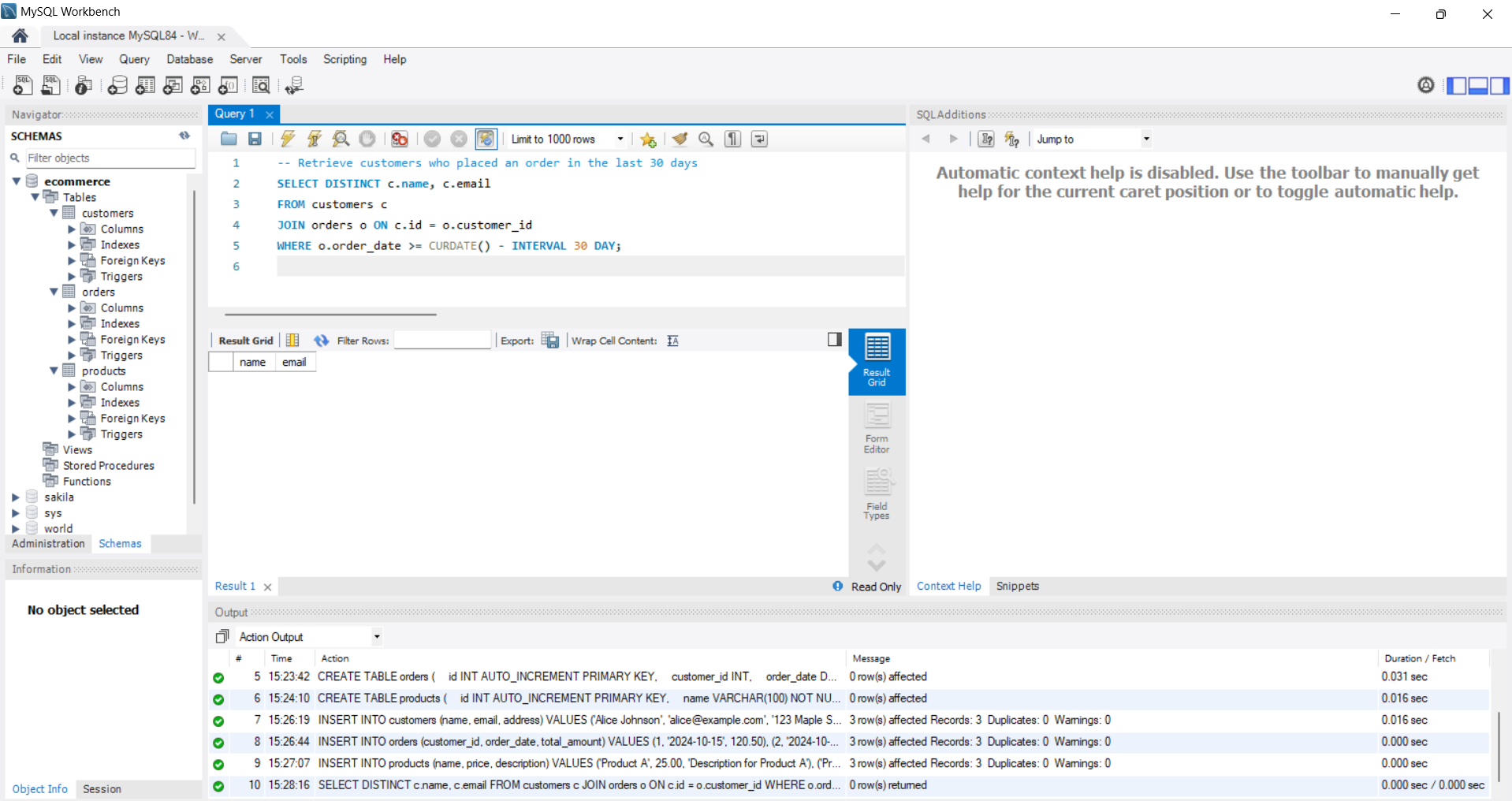
SELECT DISTINCT c.name, c.email

FROM customers c

JOIN orders o ON c.id = o.customer\_id

WHERE o.order\_date >= CURDATE() - INTERVAL 30 DAY;

**Output:**



**Get the total amount of all orders placed by each customer**

-- Get total amount of orders placed by each customer

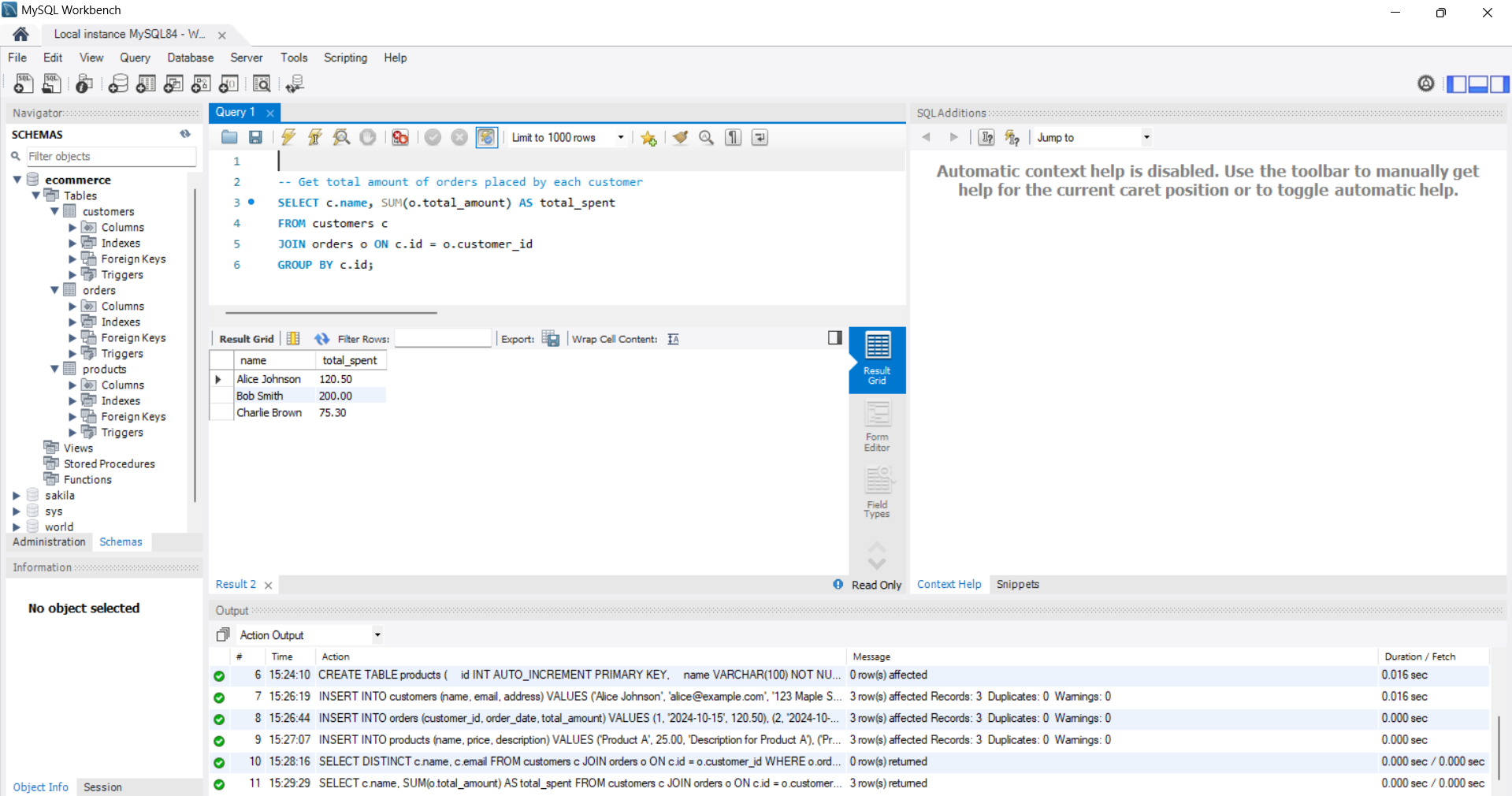
SELECT c.name, SUM(o.total\_amount) AS total\_spent

FROM customers c

JOIN orders o ON c.id = o.customer\_id

GROUP BY c.id;

**Output:**



**Update the price of Product C to 45.00**

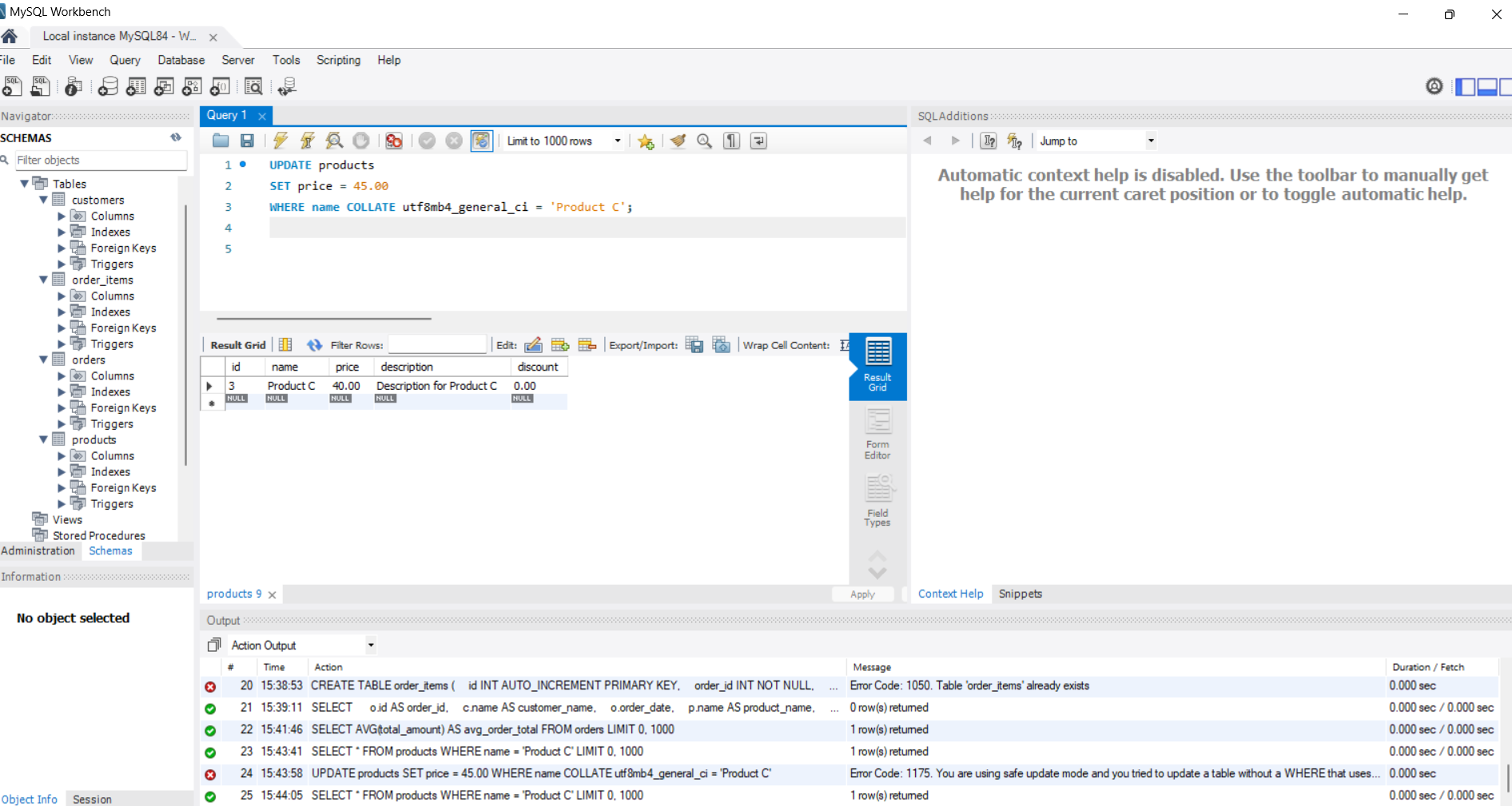
-- Update the price of Product C to 45.00

UPDATE products

SET price = 45.00

WHERE name = 'Product C';

**Output:**



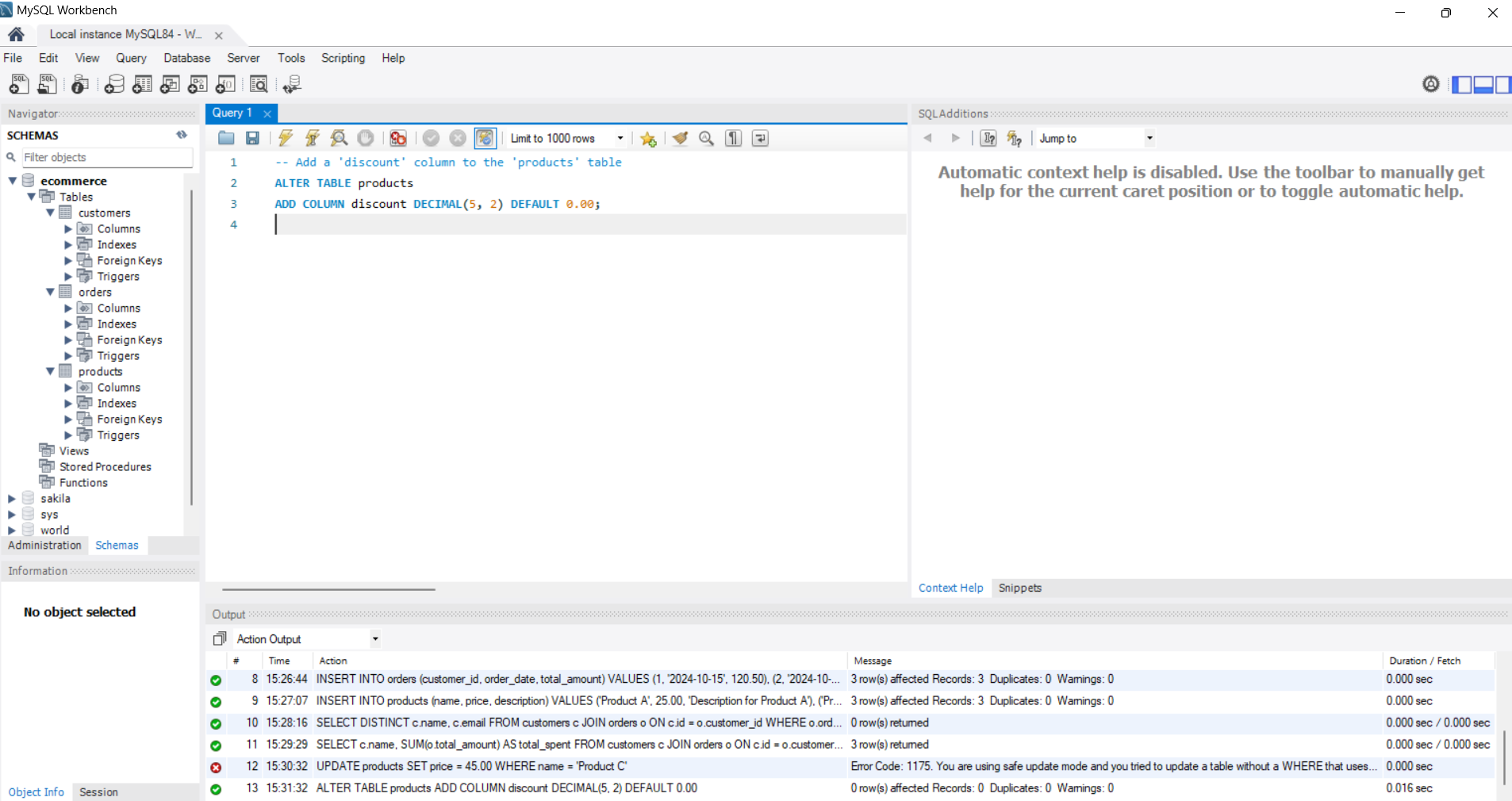
**Add a new column 'discount' to the products table**

-- Add a 'discount' column to the 'products' table

ALTER TABLE products

ADD COLUMN discount DECIMAL(5, 2) DEFAULT 0.00;

**Output:**



**Retrieve the top 3 products with the highest price**

-- Retrieve top 3 products with the highest price

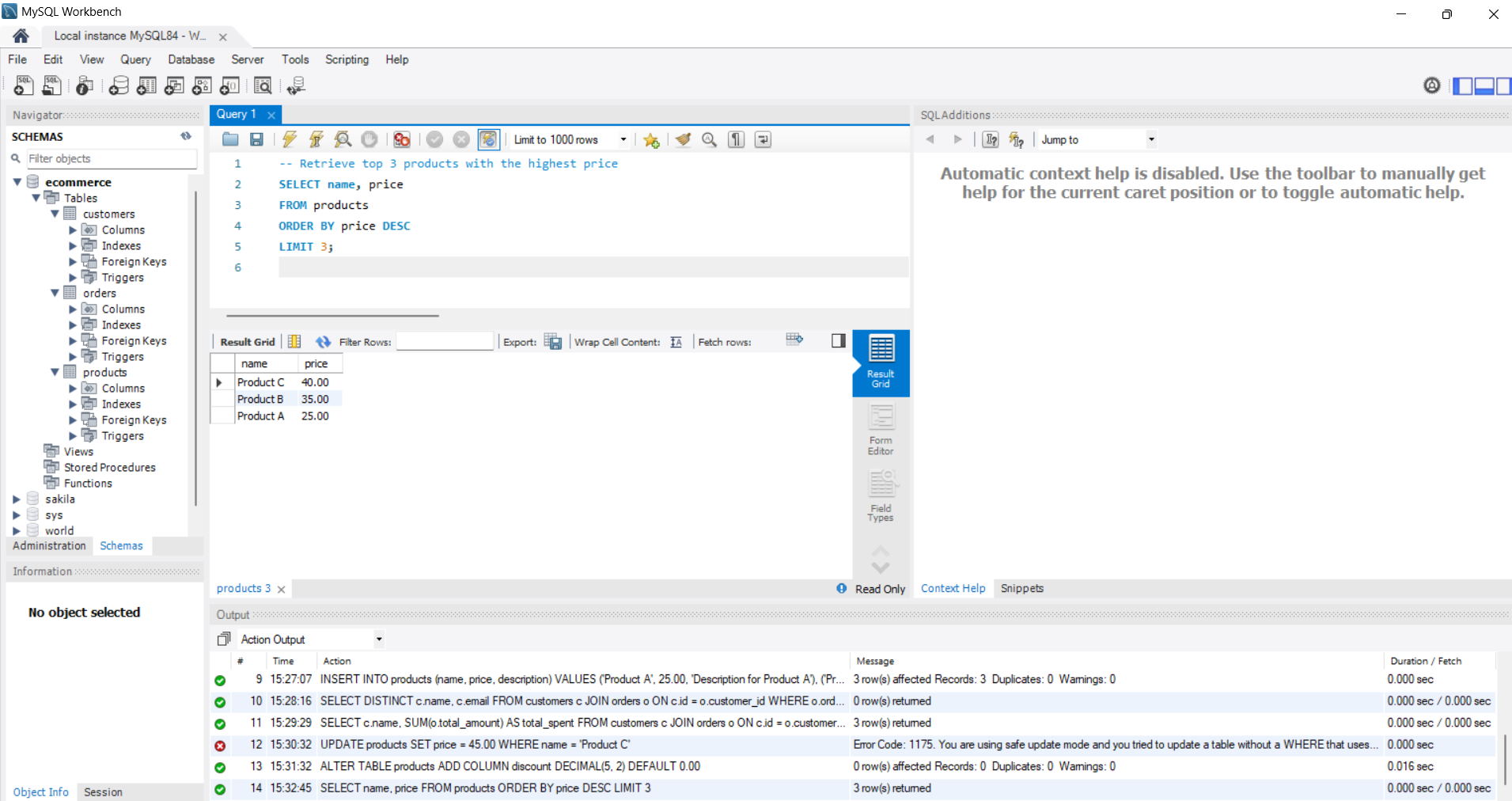
SELECT name, price

FROM products

ORDER BY price DESC

LIMIT 3;

**Output:**



**Join the orders and customers tables to retrieve the customer's name and order date for each order**

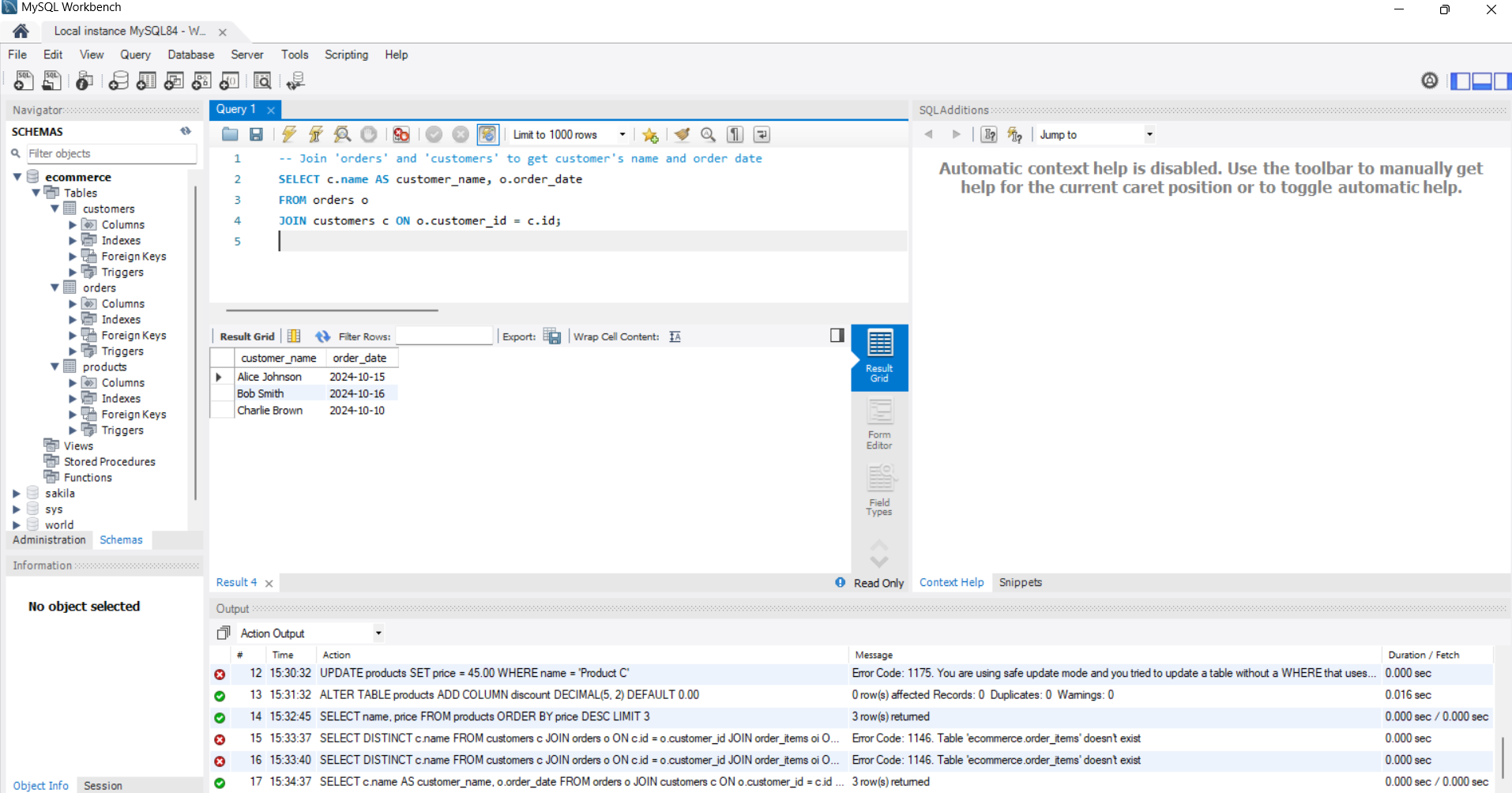
-- Join 'orders' and 'customers' to get customer's name and order date

SELECT c.name AS customer\_name, o.order\_date

FROM orders o

JOIN customers c ON o.customer\_id = c.id;

**Output:**



**Retrieve the orders with a total amount greater than 150.00**

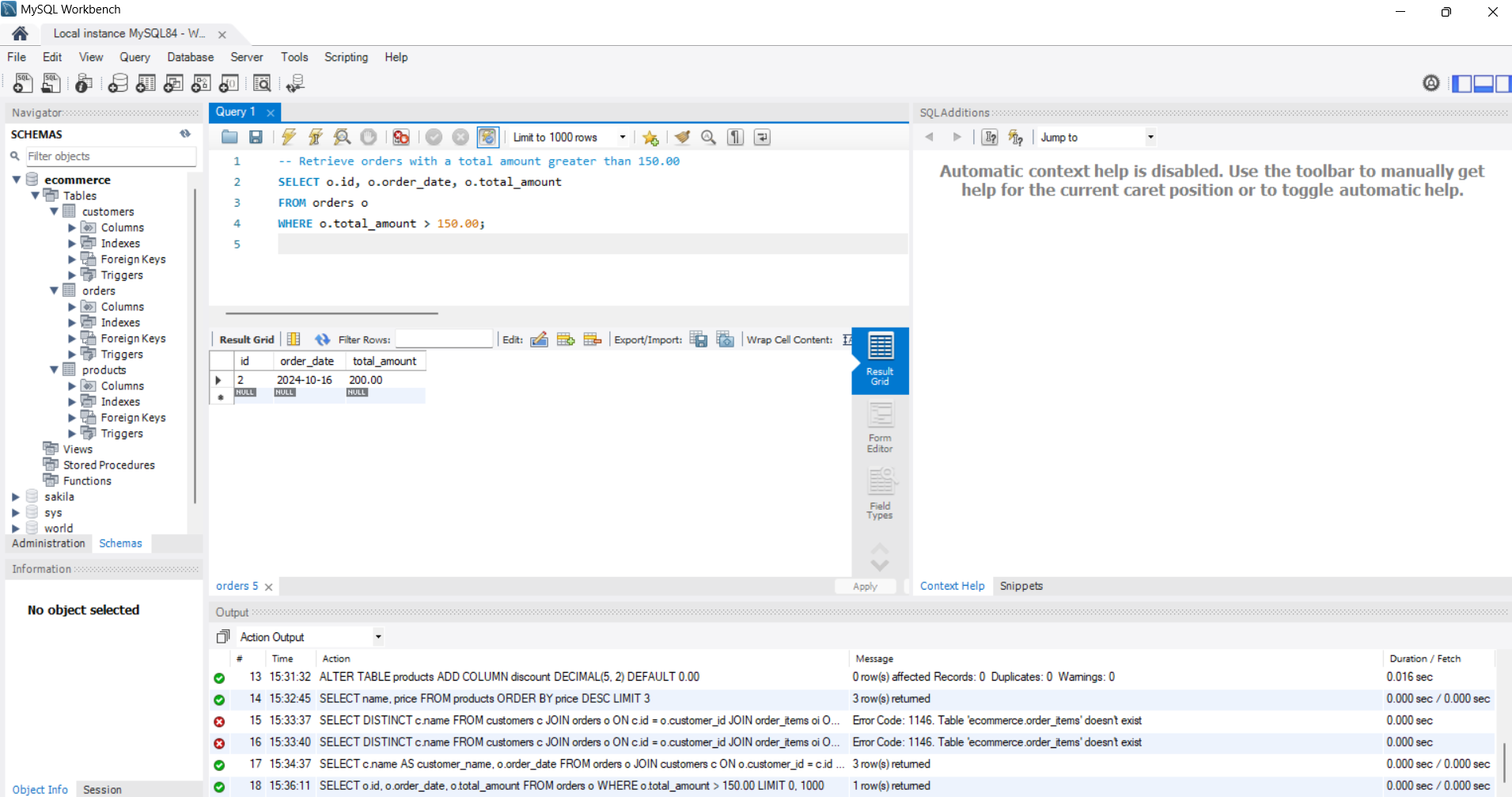
-- Retrieve orders with a total amount greater than 150.00

SELECT o.id, o.order\_date, o.total\_amount

FROM orders o

WHERE o.total\_amount > 150.00;

**Output:**



**Normalize the database by creating a separate table for order items and updating the orders table**

-- Create the 'order\_items' table to normalize the database

CREATE TABLE order\_items (

id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT,

product\_id INT,

quantity INT,

FOREIGN KEY (order\_id) REFERENCES orders(id),

FOREIGN KEY (product\_id) REFERENCES products(id)

);

**Output:**



**Query to Join All Related Tables (Example)**

Retrieve full details for each order, including customer name, order date, product name, and quantity:

SELECT

o.id AS order\_id,

c.name AS customer\_name,

o.order\_date,

p.name AS product\_name,

oi.quantity,

p.price AS product\_price,

(oi.quantity \* p.price) AS line\_total

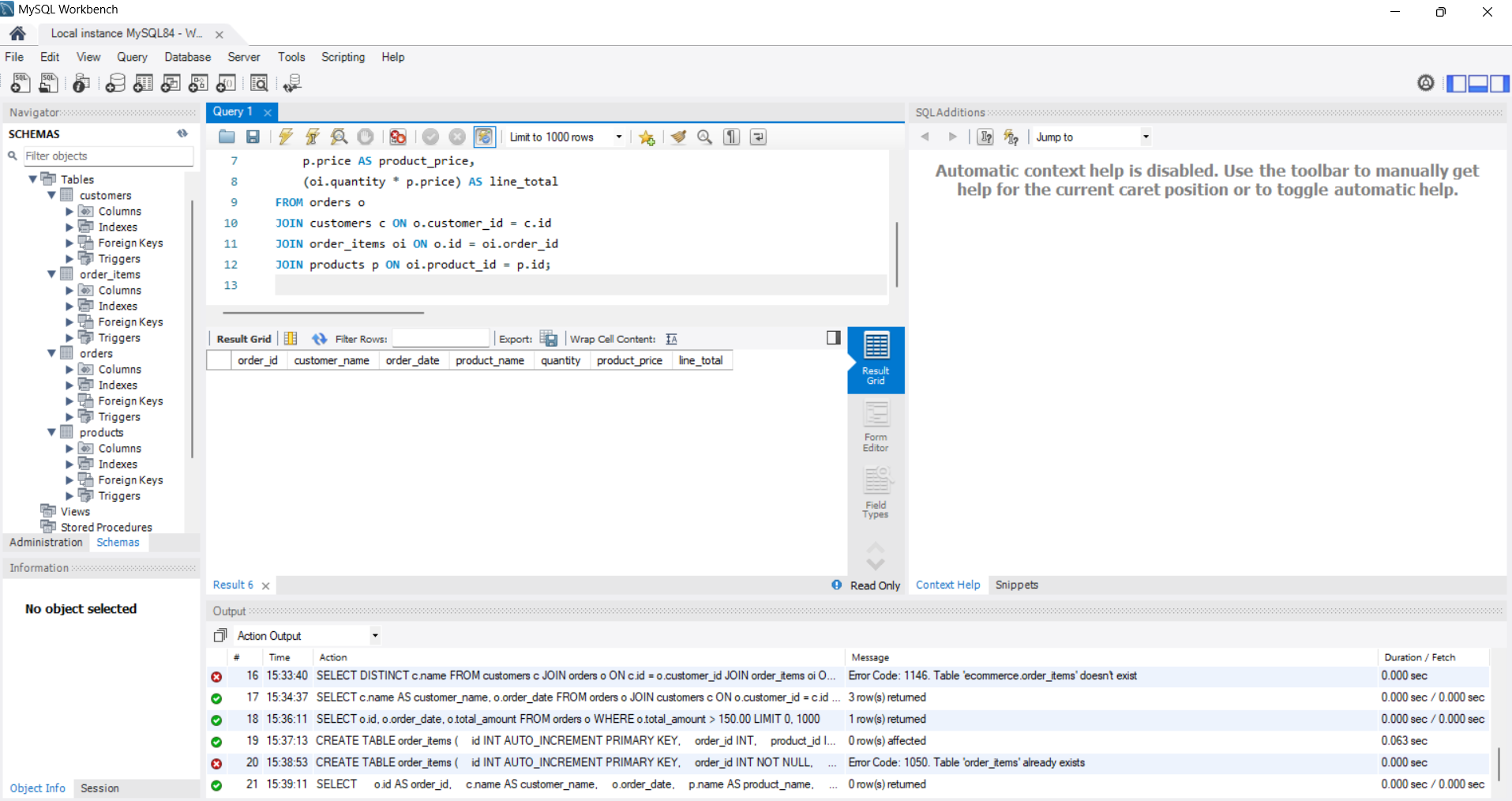
FROM orders o

JOIN customers c ON o.customer\_id = c.id

JOIN order\_items oi ON o.id = oi.order\_id

JOIN products p ON oi.product\_id = p.id;

**Output:**



**Retrieve the average total of all orders**

-- Retrieve the average total of all orders

SELECT AVG(total\_amount) AS avg\_order\_total

FROM orders;

**Output:**

