MySql Task

Create Database and Tables

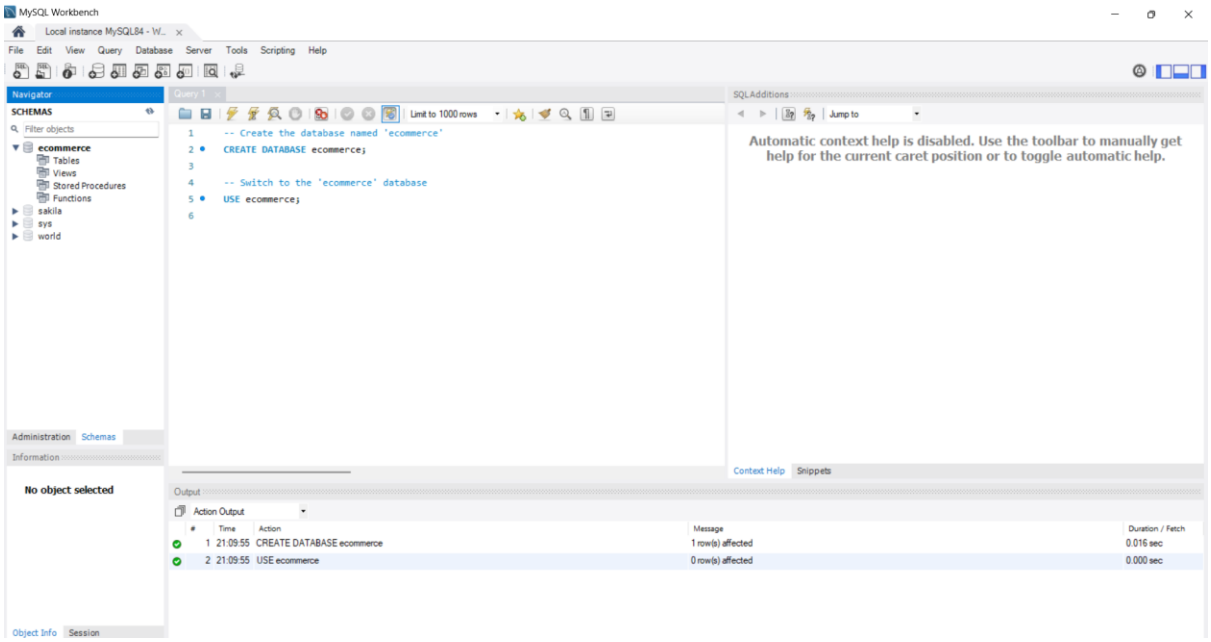
-- Create the database named 'ecommerce'

```
CREATE DATABASE ecommerce;
```

-- Switch to the 'ecommerce' database

```
USE ecommerce;
```

Output:



-- Create the 'customers' table to store customer details

```
CREATE TABLE customers (
    id INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each customer
    name VARCHAR(255) NOT NULL, -- Name of the customer
    email VARCHAR(255) NOT NULL UNIQUE, -- Email address of the customer
    address TEXT NOT NULL -- Address of the customer
);
```

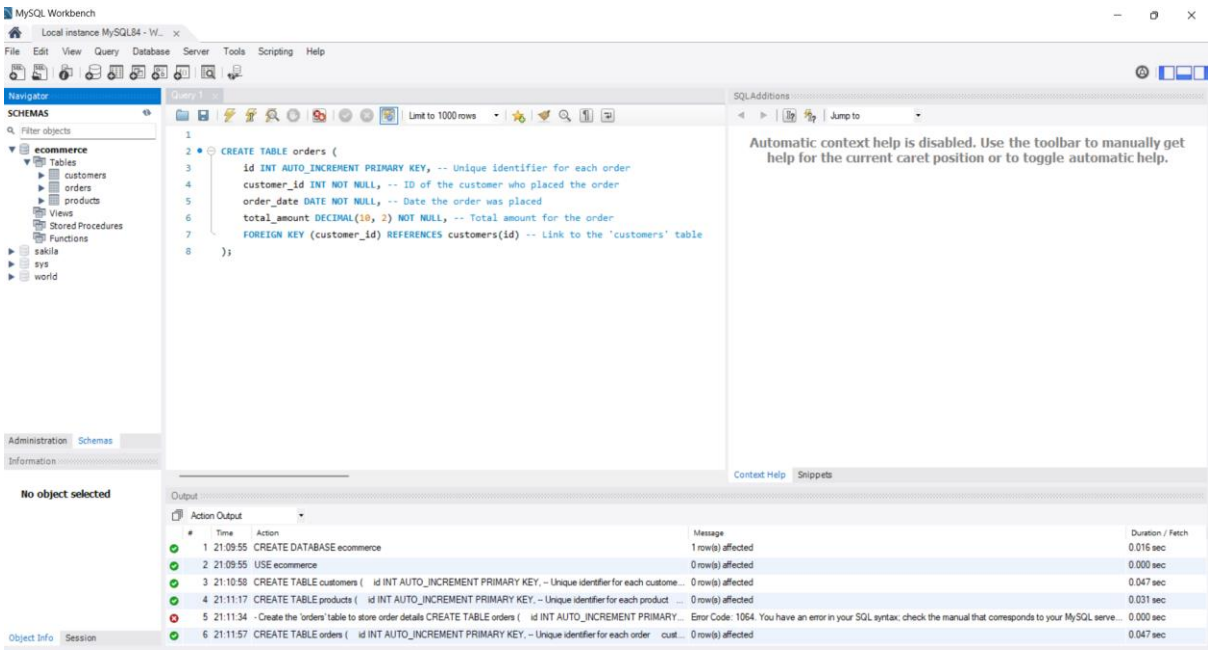
-- Create the 'products' table to store product details

```
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each product
    name VARCHAR(255) NOT NULL, -- Name of the product
    price DECIMAL(10, 2) NOT NULL, -- Price of the product
    description TEXT NOT NULL -- Description of the product
);
```

-- Create the 'orders' table to store order details

```
CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each order
    customer_id INT NOT NULL, -- ID of the customer who placed the order
    order_date DATE NOT NULL, -- Date the order was placed
    total_amount DECIMAL(10, 2) NOT NULL, -- Total amount for the order
    FOREIGN KEY (customer_id) REFERENCES customers(id) -- Link to the 'customers' table
);
```

Output:

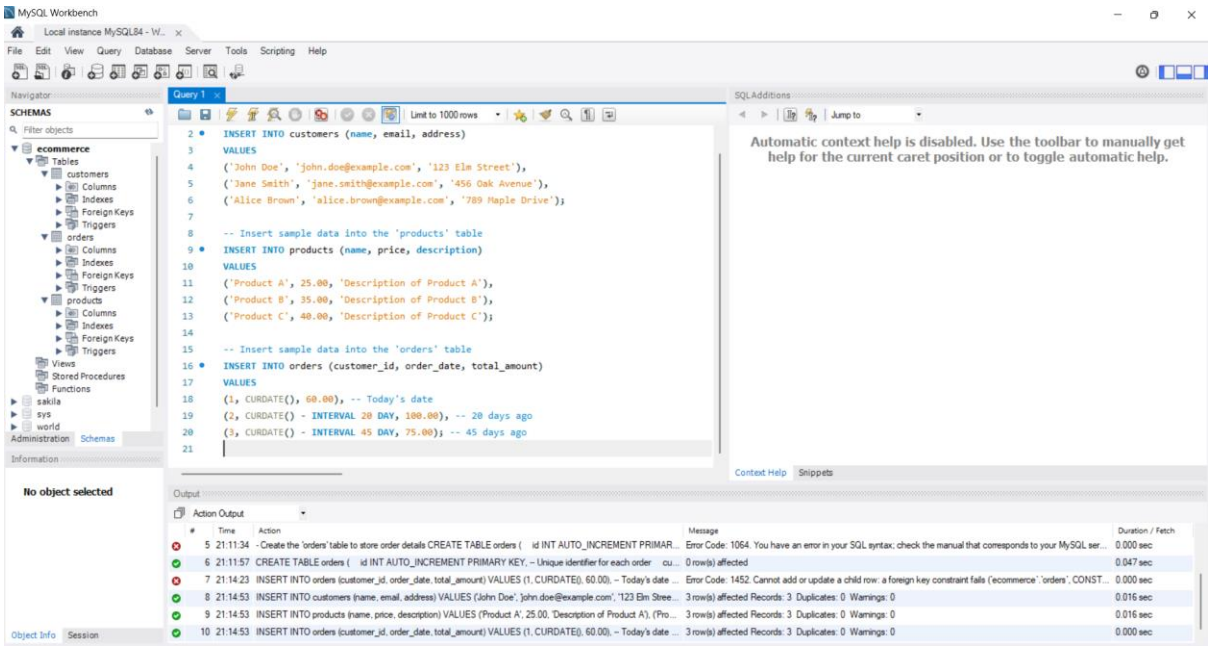


---

2. Insert Sample Data

```
-- Insert sample data into the 'customers' table
INSERT INTO customers (name, email, address)
VALUES
('John Doe', 'john.doe@example.com', '123 Elm Street'),
('Jane Smith', 'jane.smith@example.com', '456 Oak Avenue'),
('Alice Brown', 'alice.brown@example.com', '789 Maple Drive');

-- Insert sample data into the 'products' table
INSERT INTO products (name, price, description)
VALUES
('Product A', 25.00, 'Description of Product A'),
('Product B', 35.00, 'Description of Product B'),
('Product C', 40.00, 'Description of Product C');

-- Insert sample data into the 'orders' table
INSERT INTO orders (customer_id, order_date, total_amount)
VALUES
(1, CURDATE(), 60.00), -- Today's date
(2, CURDATE() - INTERVAL 20 DAY, 100.00), -- 20 days ago
(3, CURDATE() - INTERVAL 45 DAY, 75.00); -- 45 days ago
```

Output:



---

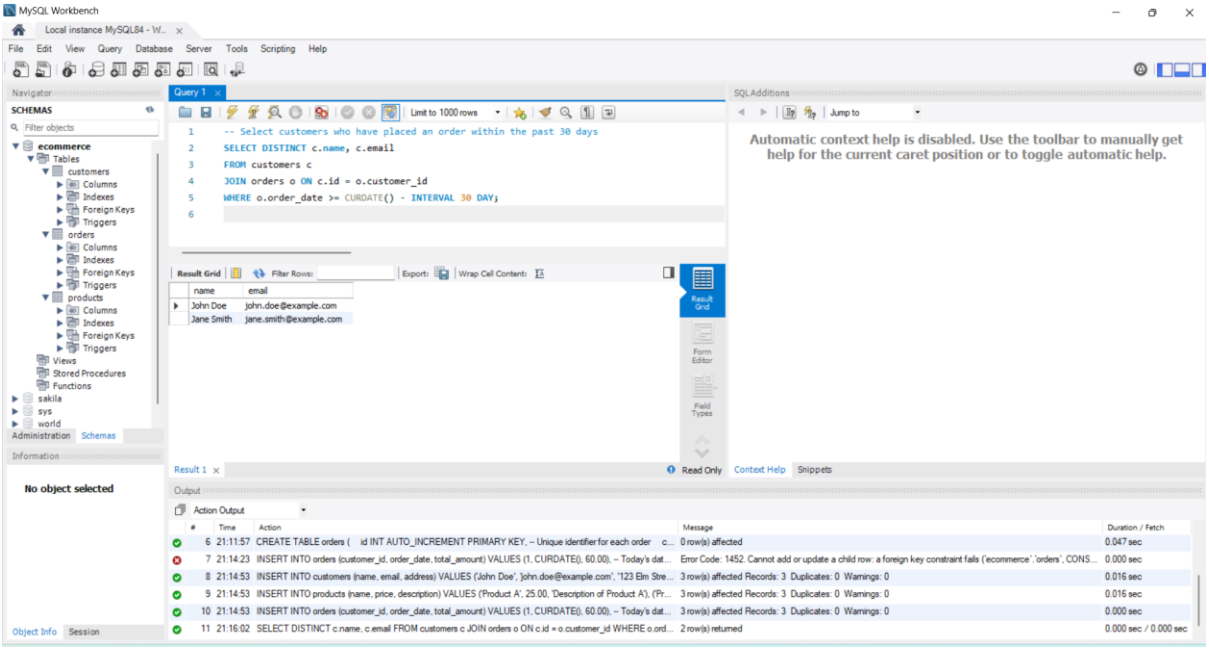
3. Queries

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

-- Select customers who have placed an order within the past 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:

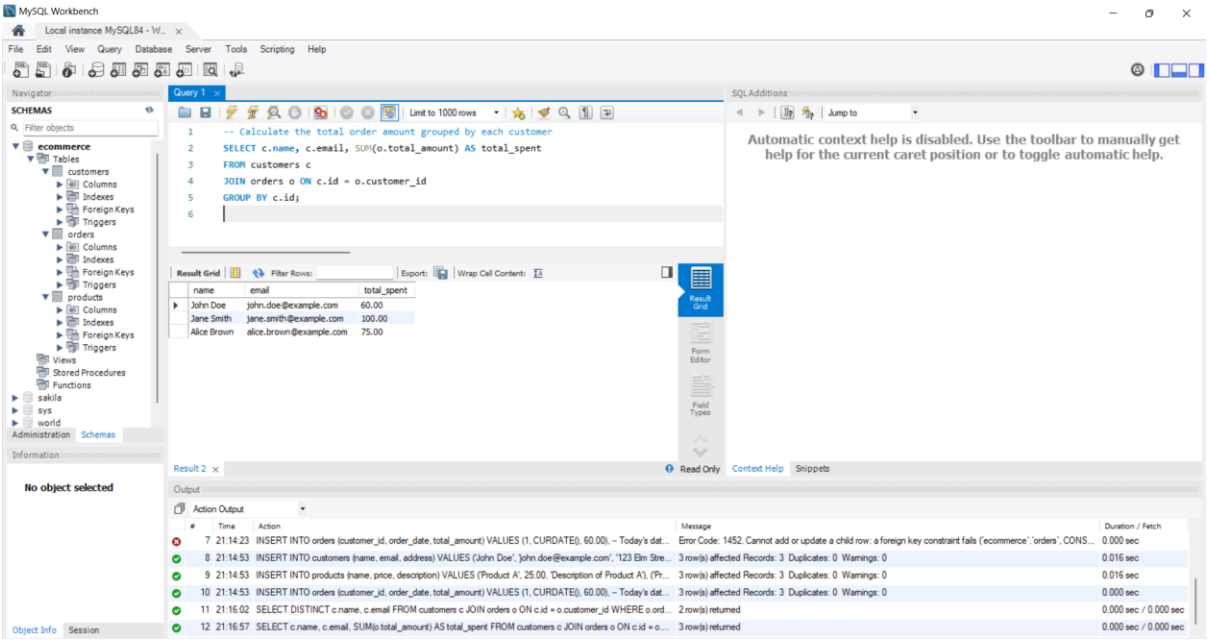


b. Get the total amount of all orders placed by each customer

-- Calculate the total order amount grouped by each customer

```
SELECT c.name, c.email, SUM(o.total_amount) AS total_spent
FROM customers c
JOIN orders o ON c.id = o.customer_id
GROUP BY c.id;
```

Output:



c. Update the price of Product C to 45.00

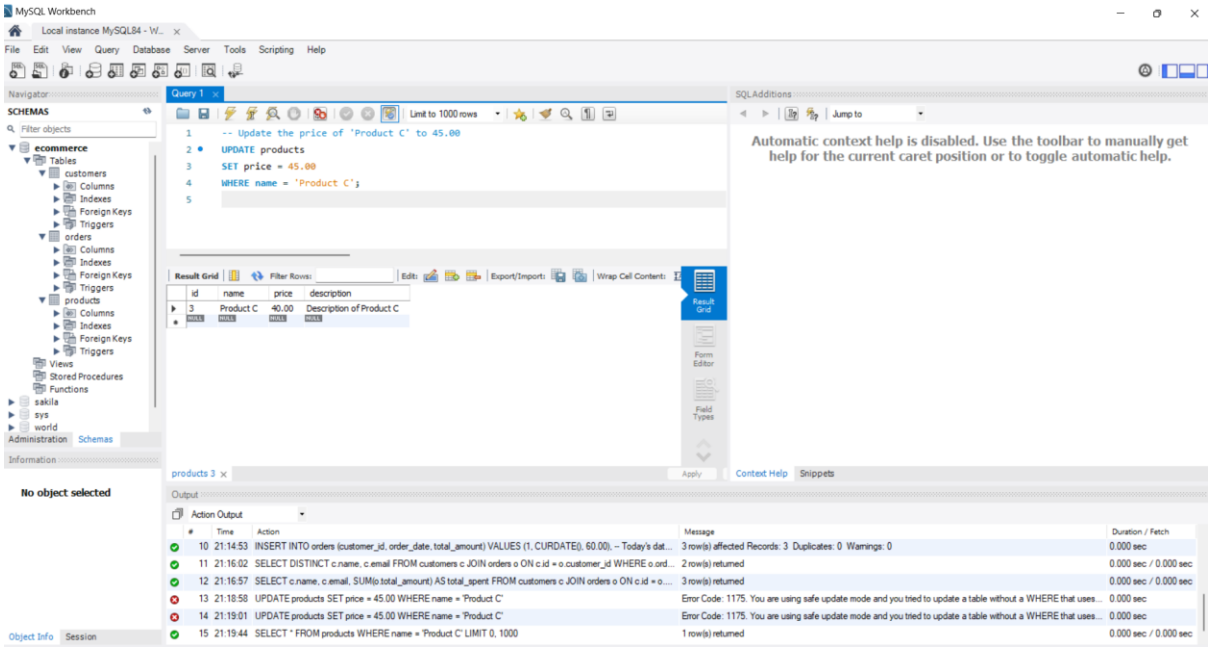
-- Update the price of 'Product C' to 45.00

UPDATE products

SET price = 45.00

WHERE name = 'Product C';

Output:



d. Add a new column discount to the products table

-- Add a new column named 'discount' with a default value of 0.00

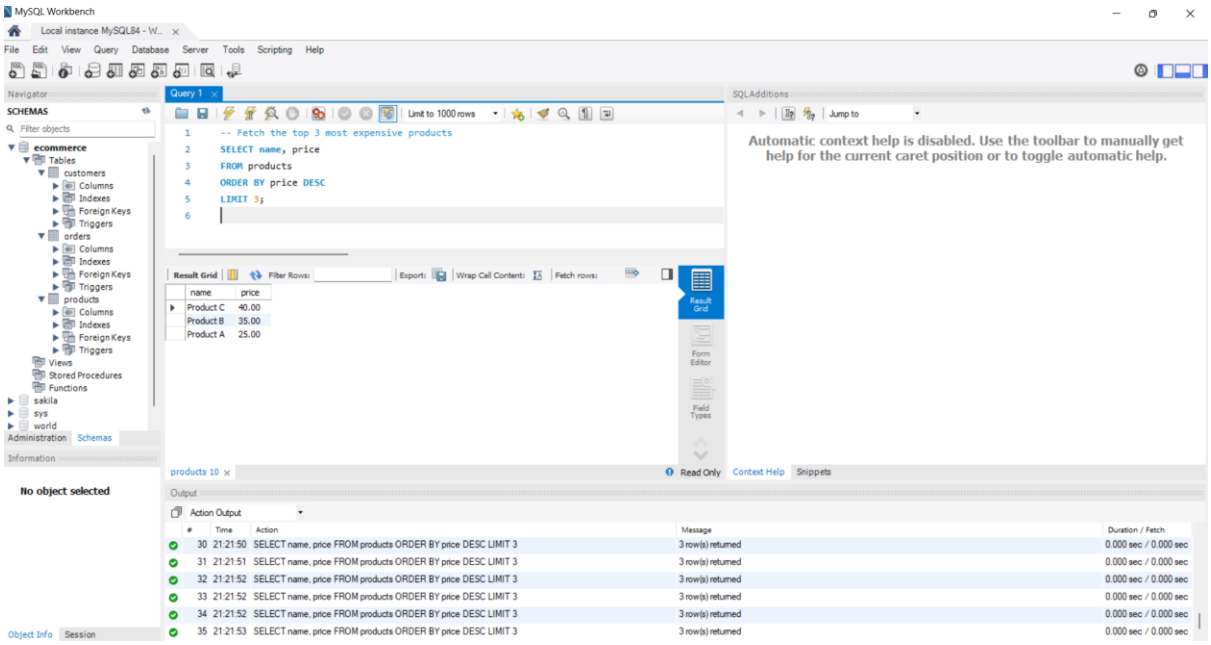
```
ALTER TABLE products

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

e. Retrieve the top 3 products with the highest price

-- Fetch the top 3 most expensive products

```
SELECT name, price

FROM products

ORDER BY price DESC

LIMIT 3;
```

Output:

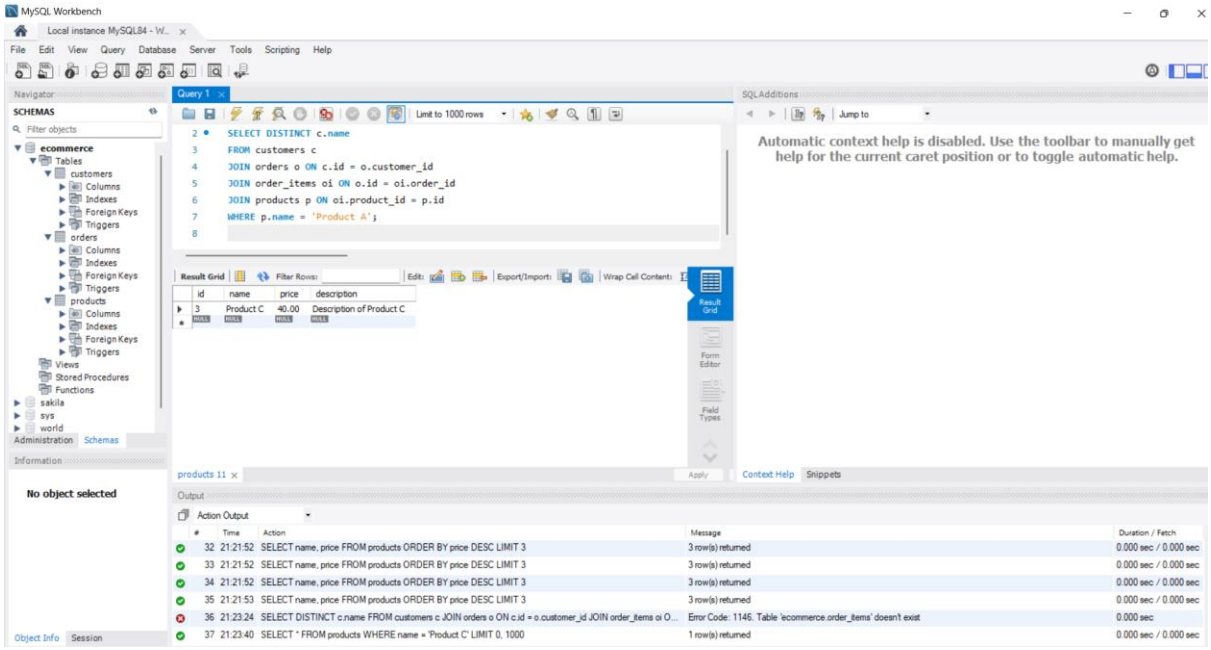


f. Get the names of customers who have ordered Product A

-- Retrieve names of customers who have ordered 'Product A'

```
SELECT DISTINCT c.name

FROM customers c

JOIN orders o ON c.id = o.customer_id

JOIN order_items oi ON o.id = oi.order_id

JOIN products p ON oi.product_id = p.id

WHERE p.name = 'Product A';
```

Output:



g. Join orders and customers tables to retrieve customer name and order date

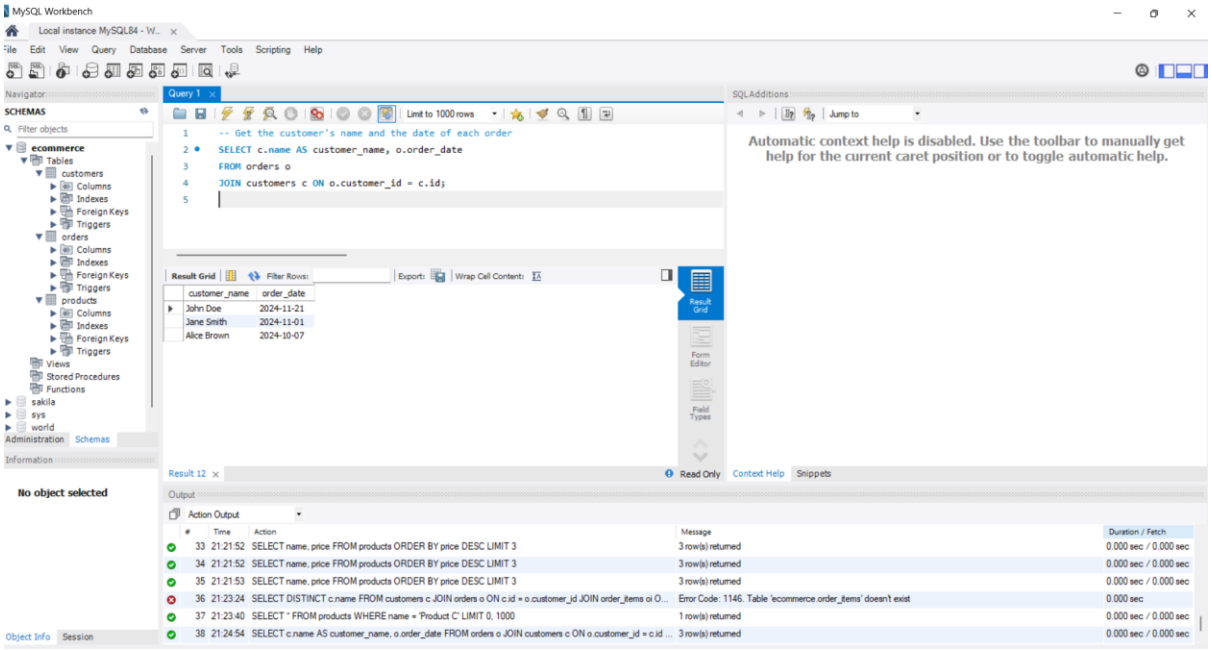
-- Get the customer's name and the date of each order

SELECT c.name AS customer_name, o.order_date

FROM orders o

JOIN customers c ON o.customer_id = c.id;

Output:



h. Retrieve orders with a total amount greater than 150.00

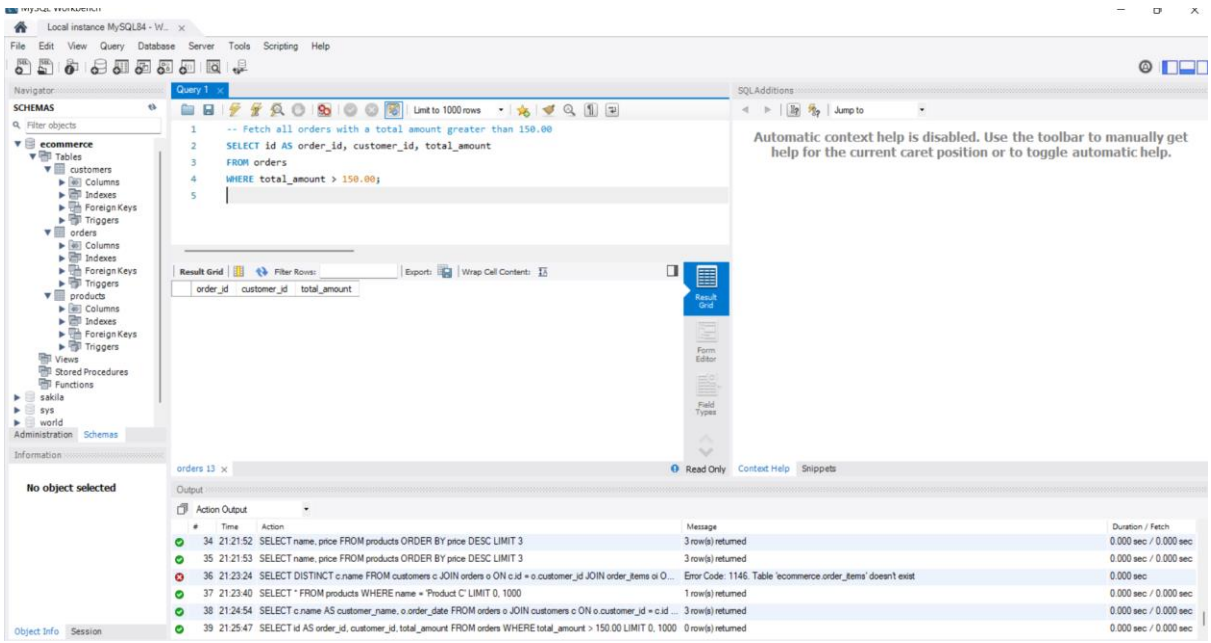
-- Fetch all orders with a total amount greater than 150.00

SELECT id AS order_id, customer_id, total_amount

FROM orders

WHERE total_amount > 150.00;

Output:



i. Normalize the database (add order_items table)

-- Create the 'order_items' table to store details of individual items in an order

CREATE TABLE order_items (

id INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each order item

order_id INT NOT NULL, -- ID of the order

product_id INT NOT NULL, -- ID of the product

quantity INT NOT NULL, -- Quantity of the product ordered

FOREIGN KEY (order_id) REFERENCES orders(id), -- Link to the 'orders' table

FOREIGN KEY (product_id) REFERENCES products(id) -- Link to the 'products' table

);

-- Remove the 'total_amount' column from the 'orders' table to calculate it dynamically

ALTER TABLE orders DROP COLUMN total_amount;

-- Insert sample data into the 'order_items' table

INSERT INTO order_items (order_id, product_id, quantity)

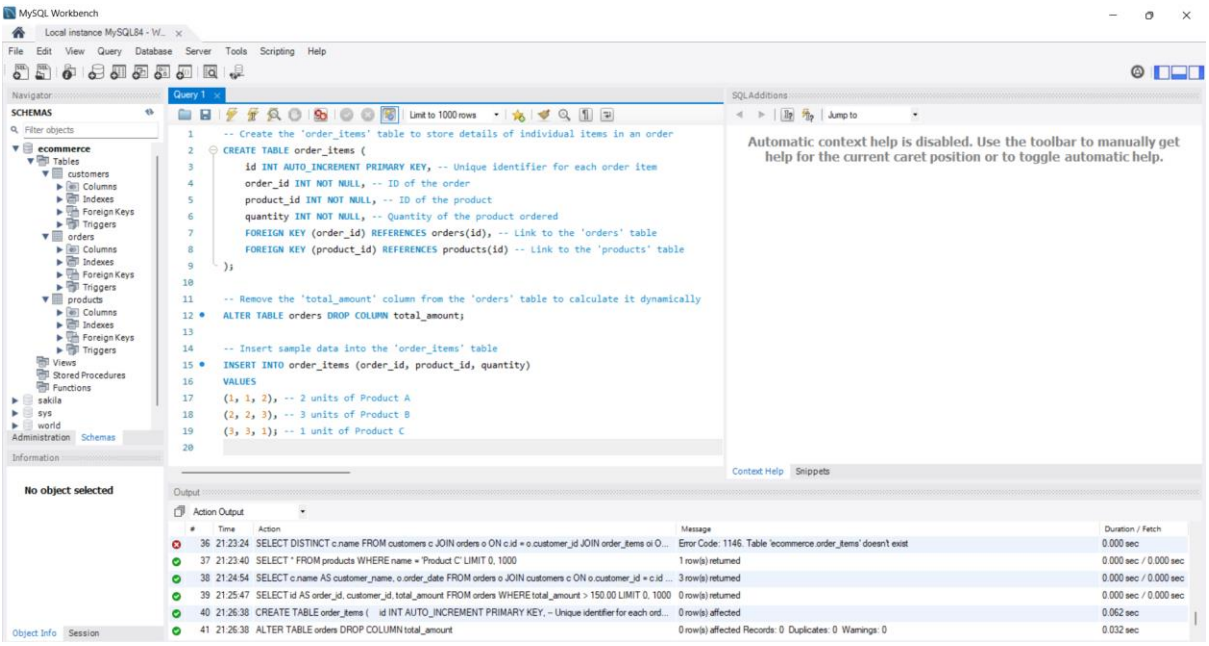
VALUES

(1, 1, 2), -- 2 units of Product A

(2, 2, 3), -- 3 units of Product B

(3, 3, 1); -- 1 unit of Product C

Output:



j. Retrieve the average total of all orders

-- Calculate the average total amount for all orders dynamically

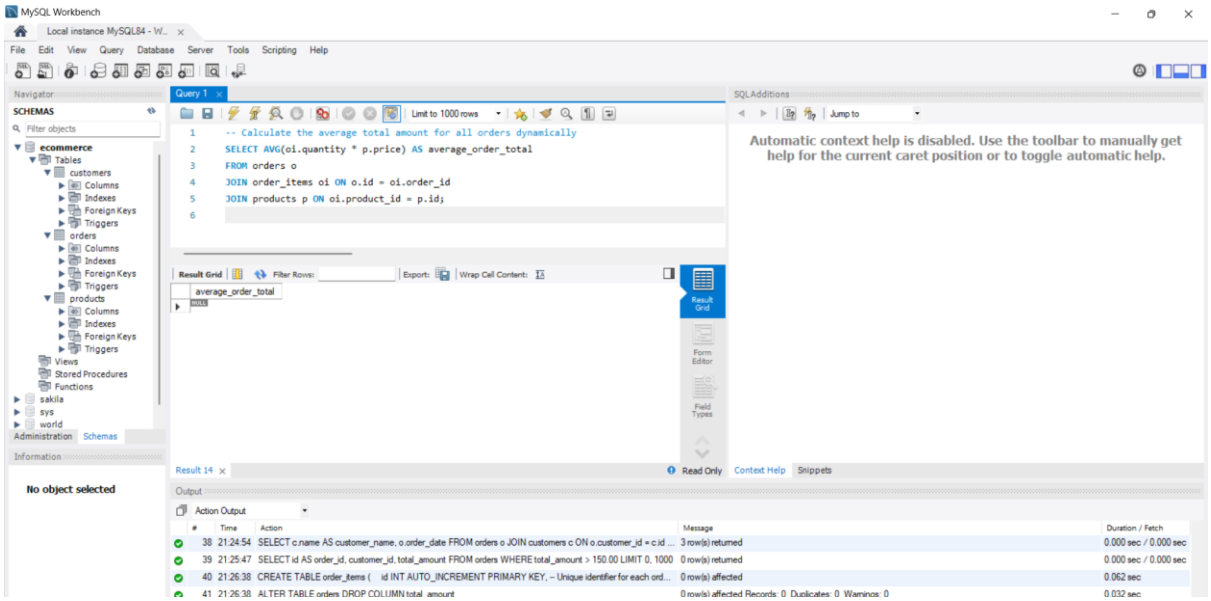
SELECT AVG(oi.quantity * p.price) AS average_order_total

FROM orders o

JOIN order_items oi ON o.id = oi.order_id

JOIN products p ON oi.product_id = p.id;

Output:


MySQL Workbench
Local instance MySQL84 - W...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
ecommerce
Tables
customers
Columns
Indexes
Foreign Keys
Triggers
orders
Columns
Indexes
Foreign Keys
Triggers
products
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
sakila
sys
world
Administration Schemas
Information
No object selected
Query 1
Limit to 1000 rows
1 -- Calculate the average total amount for all orders dynamically
2 SELECT AVG(oi.quantity * p.price) AS average_order_total
3 FROM orders o
4 JOIN order_items oi ON o.id = oi.order_id
5 JOIN products p ON oi.product_id = p.id;
6
Result Grid Filter Rows: Export: Wrap Cell Content:
average_order_total
NULL
Result Grid
Form Editor
Field Types
Result 14
Read Only
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
38 21:24:54 SELECT c.name AS customer_name, o.order_date FROM orders o JOIN customers c ON o.customer_id = c.id ... 3 row(s) returned 0.000 sec / 0.000 sec
39 21:25:47 SELECT id AS order_id, customer_id, total_amount FROM orders WHERE total_amount > 150.00 LIMIT 0, 1000 0 row(s) returned 0.000 sec / 0.000 sec
40 21:26:38 CREATE TABLE order_items (id INT AUTO_INCREMENT PRIMARY KEY, -- Unique identifier for each ord... 0 row(s) affected 0.062 sec
41 21:26:38 ALTER TABLE orders DROP COLUMN total_amount 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 0.032 sec