Task 4: SQL for Data Analysis

Objective: Use SQL queries to extract and analyze data from a database.

Tools: MySQL or PostgreSQL or SQLite

Deliverables: SQL queries in a SQL file + screenshots of output

Hints/Mini Guide:

a.Use SELECT, WHERE, ORDER BY, GROUP BY

b.Use JOINS (INNER, LEFT, RIGHT)

c.Write subqueries

d.Use aggregate functions (SUM, AVG)

e.Create views for analysis

f.Optimize queries with indexes

Dataset: Ecommerce\_SQL\_Database( or any data set of your choice)

Outcome: Learn to manipulate and query structured data using SQL.

Creating the connection in MYSql to export the data

Database created

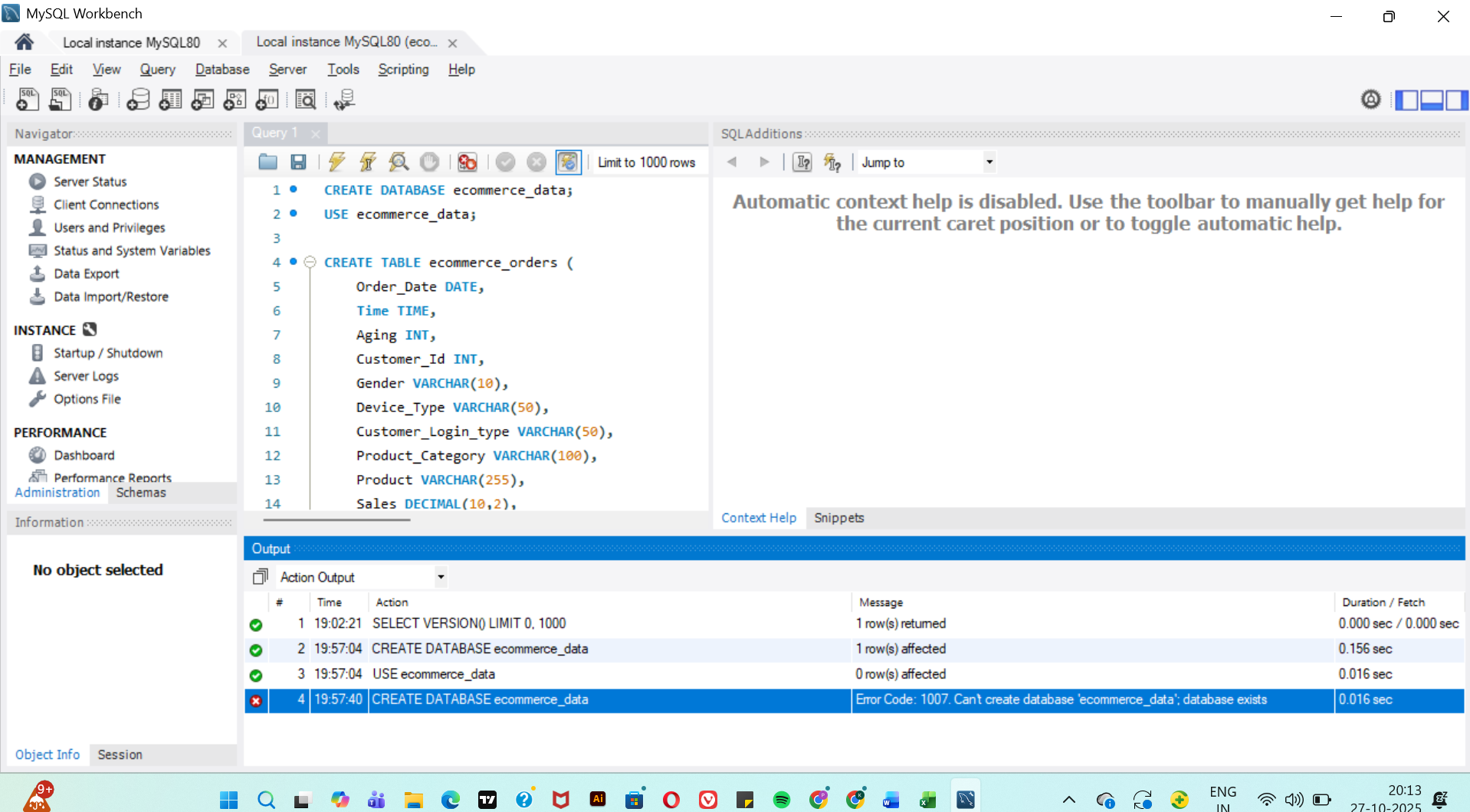
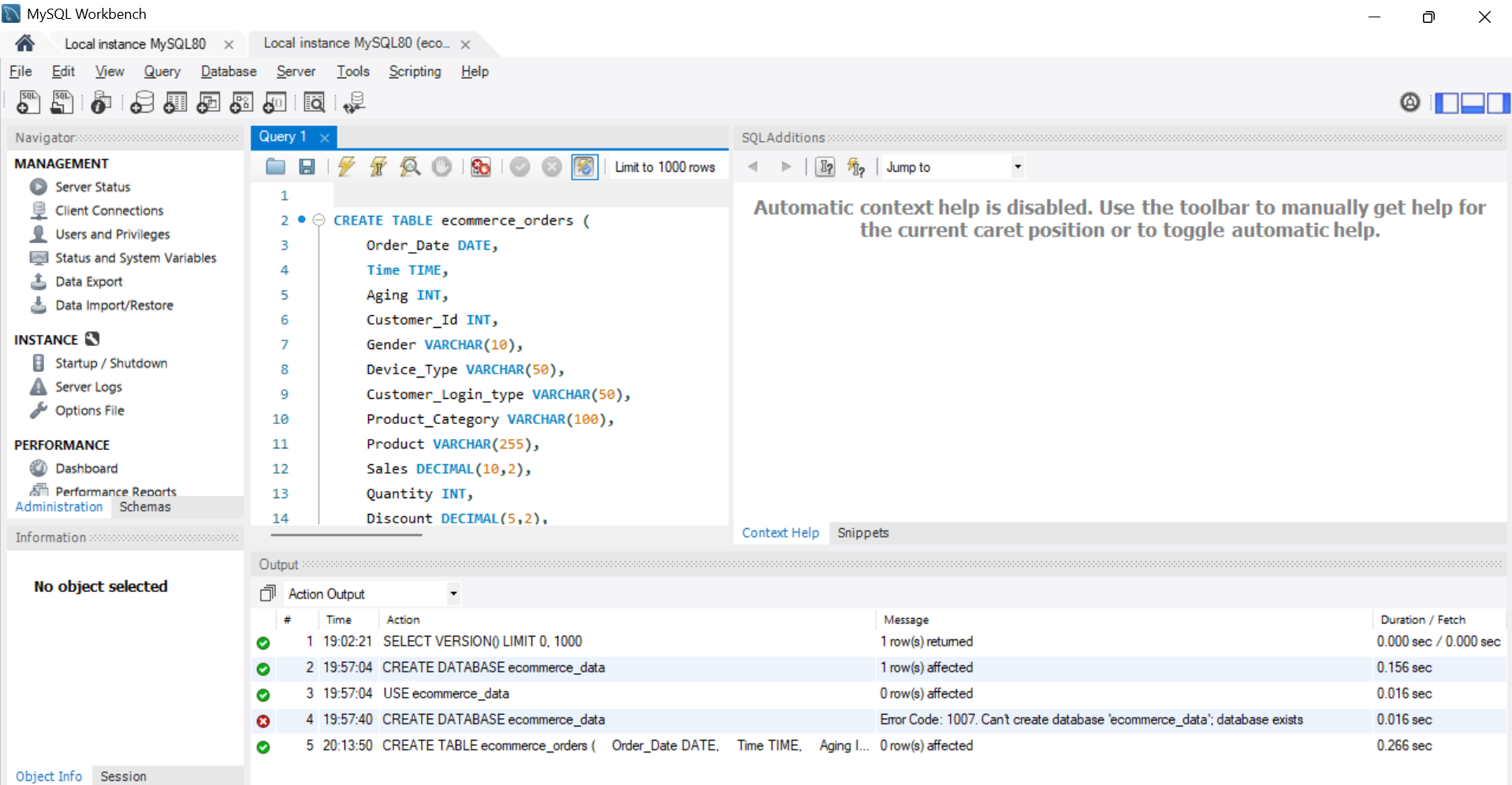


Table created   
  


In MySQL Workbench top menu →  
**Server → Data Import Wizard**

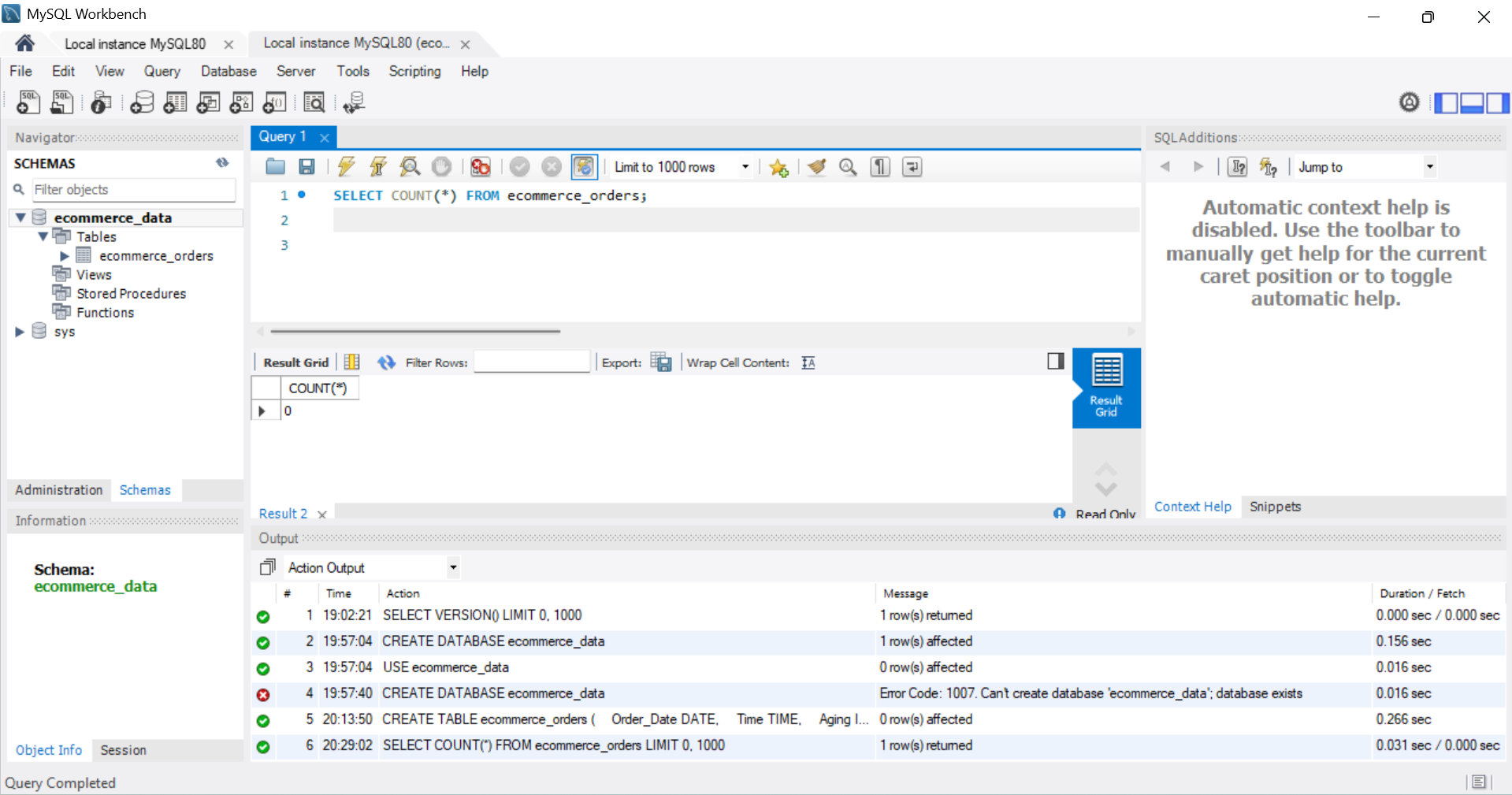
**Choose:  
Import from Self-Contained File → browse to your E-commerce Dataset.csv**

**Choose your schema: ecommerce\_data**

**Target Table: ecommerce\_orders**

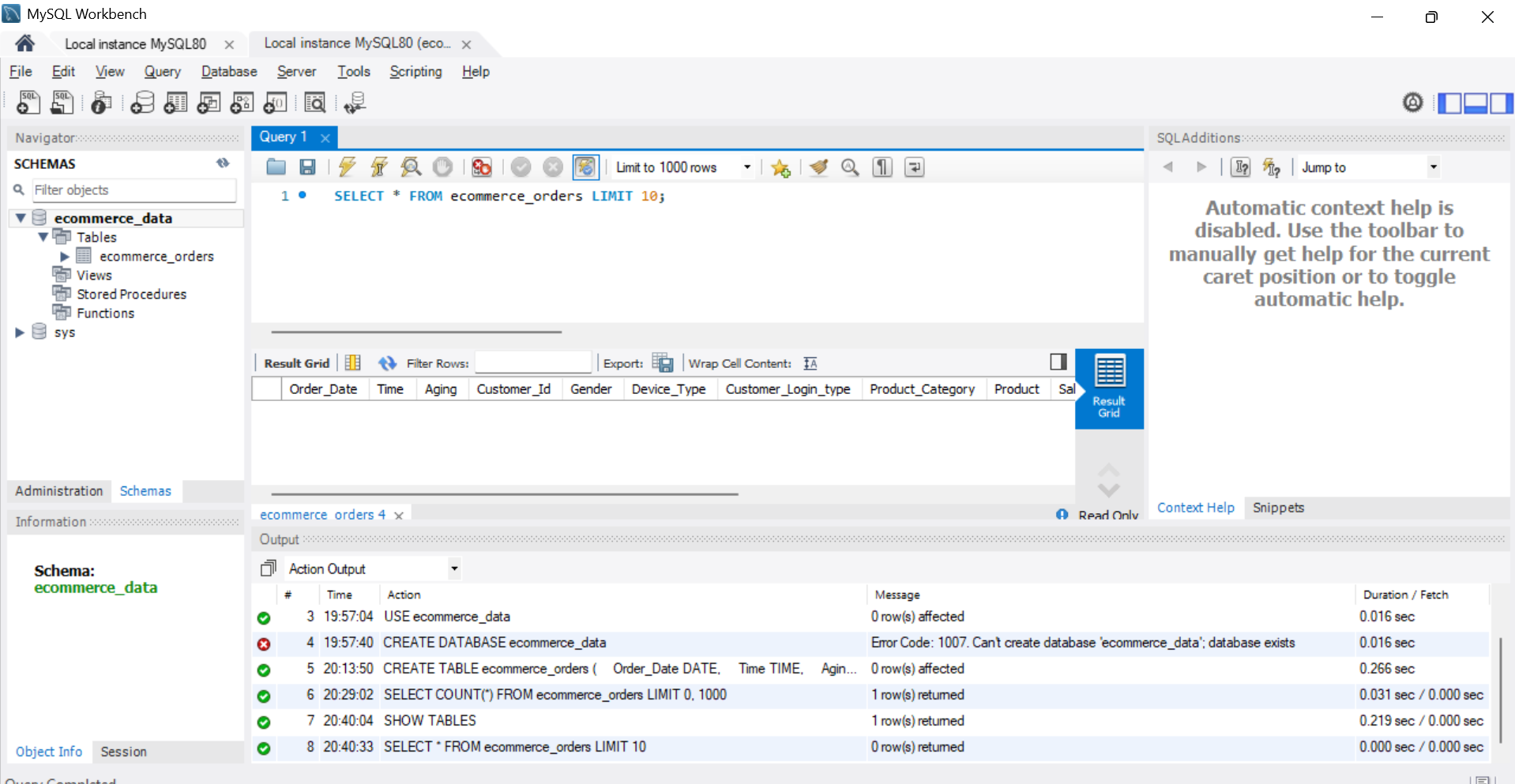
**Match columns automatically → click Next, then Finish.**

**Successfully imported the excel**

**To check the connection and the table creation   
  
**

Task (a): Use SELECT, WHERE, ORDER BY, GROUP BY

SELECT \* FROM ecommerce\_orders LIMIT 10;



Finding top selling products

SELECT

Product\_Category,

SUM(Quantity) AS Total\_Quantity,

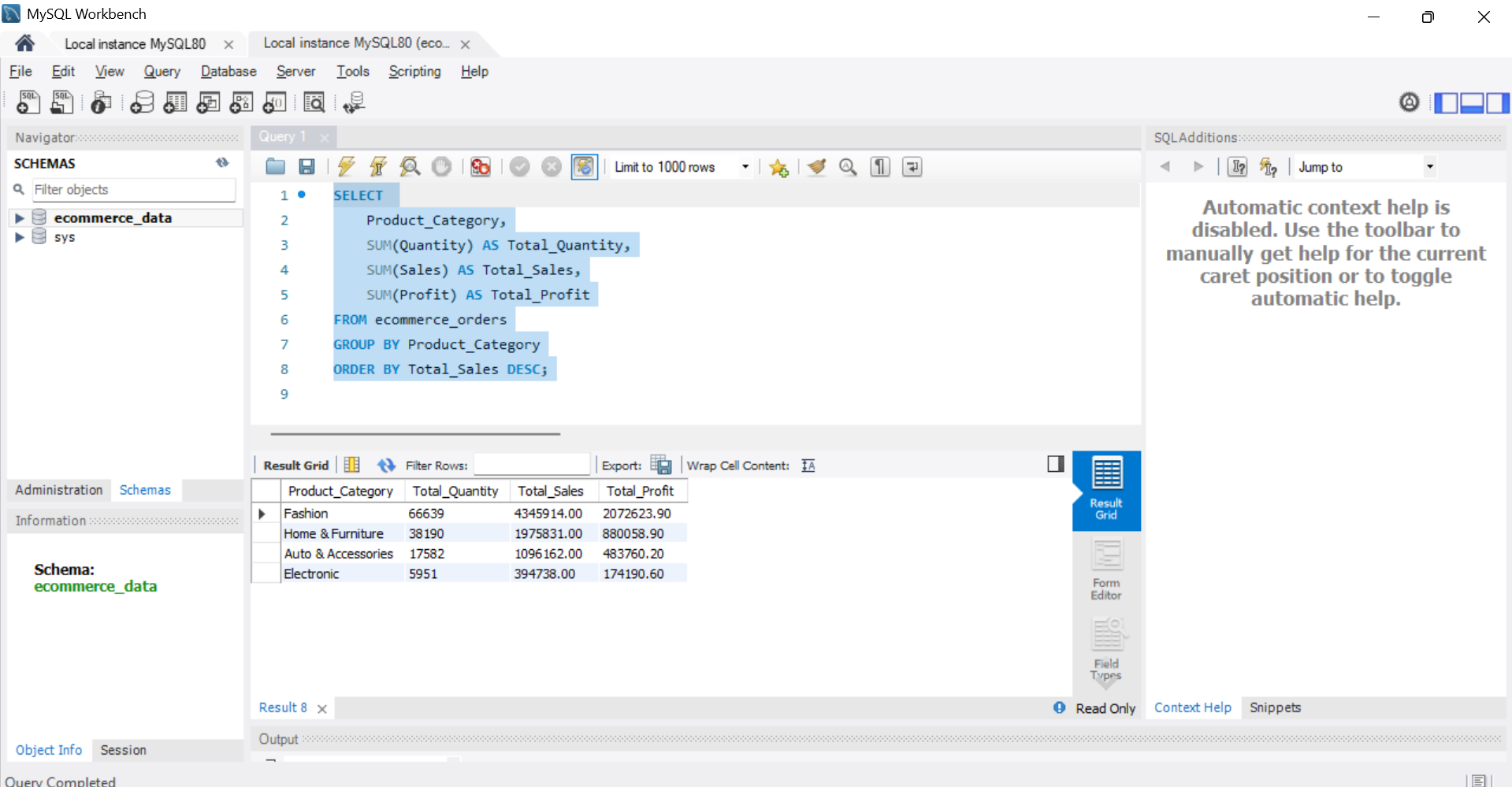
SUM(Sales) AS Total\_Sales,

SUM(Profit) AS Total\_Profit

FROM ecommerce\_orders

GROUP BY Product\_Category

ORDER BY Total\_Sales DESC;



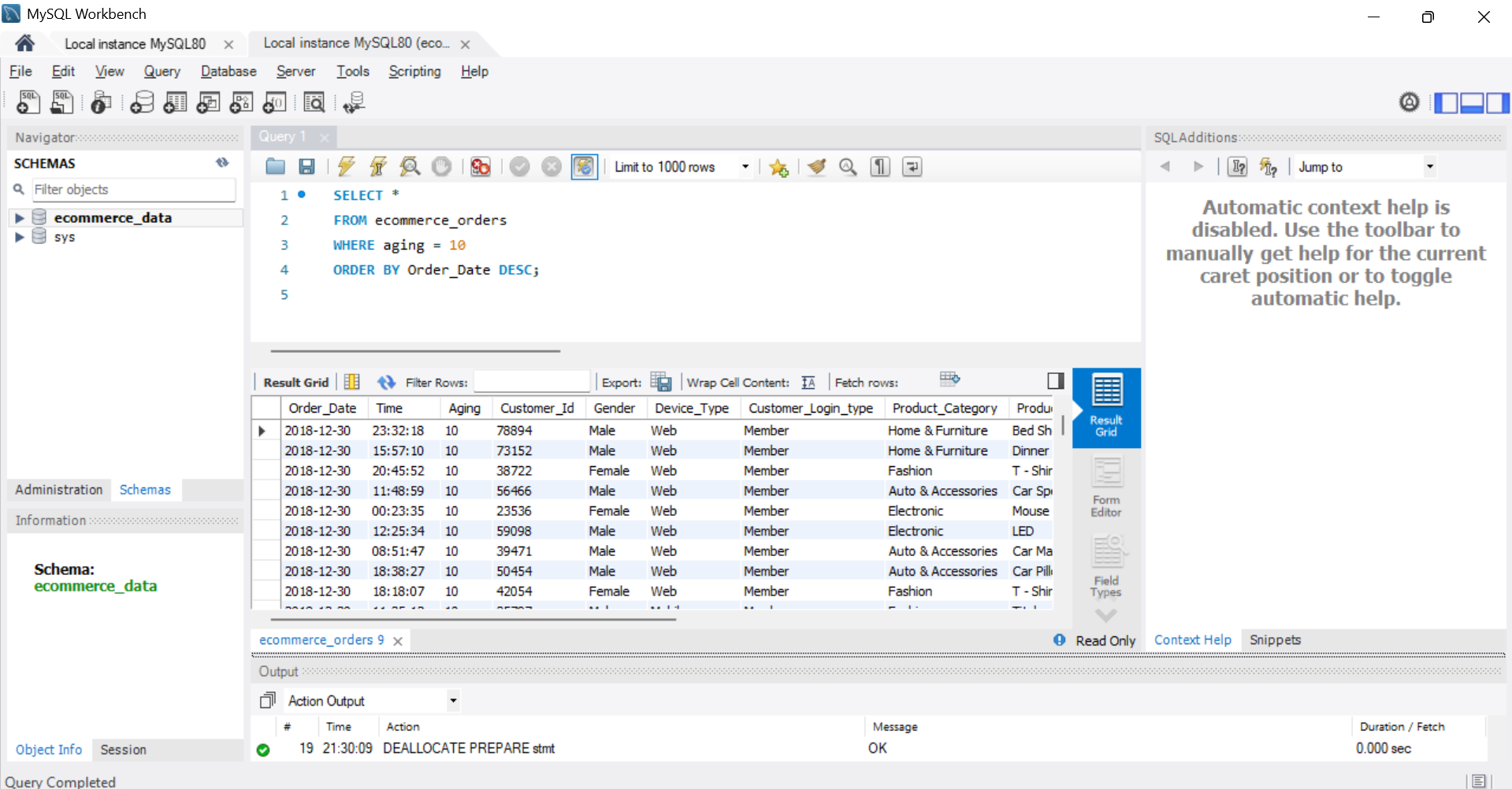
**Filter by age**

SELECT \*

FROM ecommerce\_orders

WHERE aging = 10

ORDER BY Order\_Date DESC;



Task (b): Use JOINS (INNER, LEFT, RIGHT)

SELF JOIN

SELECT a.Customer\_Id, a.Product AS Product\_1, b.Product AS Product\_2, a.Product\_Category

FROM ecommerce\_orders a

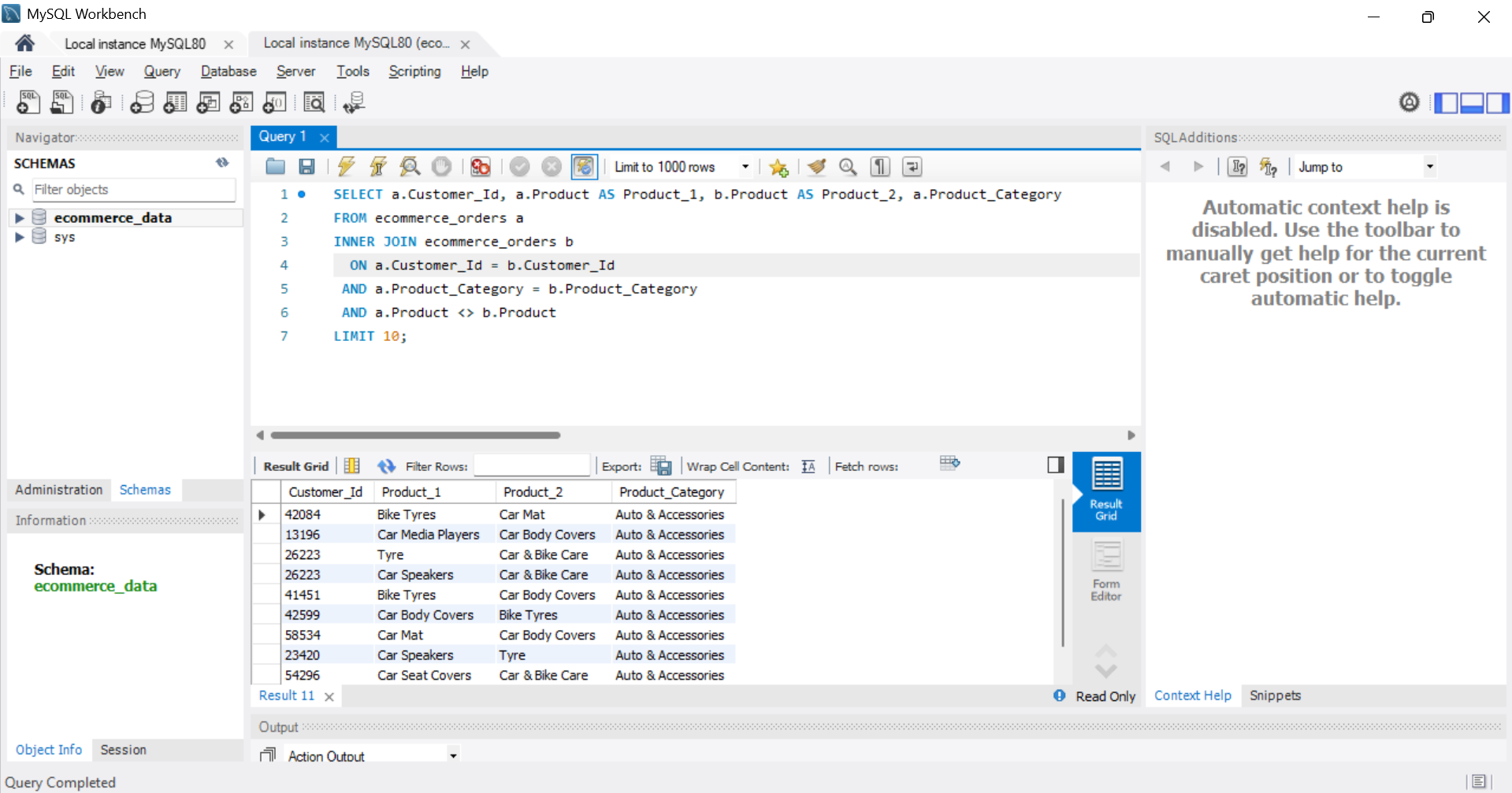
INNER JOIN ecommerce\_orders b

ON a.Customer\_Id = b.Customer\_Id

AND a.Product\_Category = b.Product\_Category

AND a.Product <> b.Product

LIMIT 10;



**Join with a Derived Table**

Join with a summary (created on the fly):

SELECT e.Customer\_Id, e.Product\_Category, e.Sales, s.Total\_Sales

FROM ecommerce\_orders e

INNER JOIN (

SELECT Customer\_Id, SUM(Sales) AS Total\_Sales

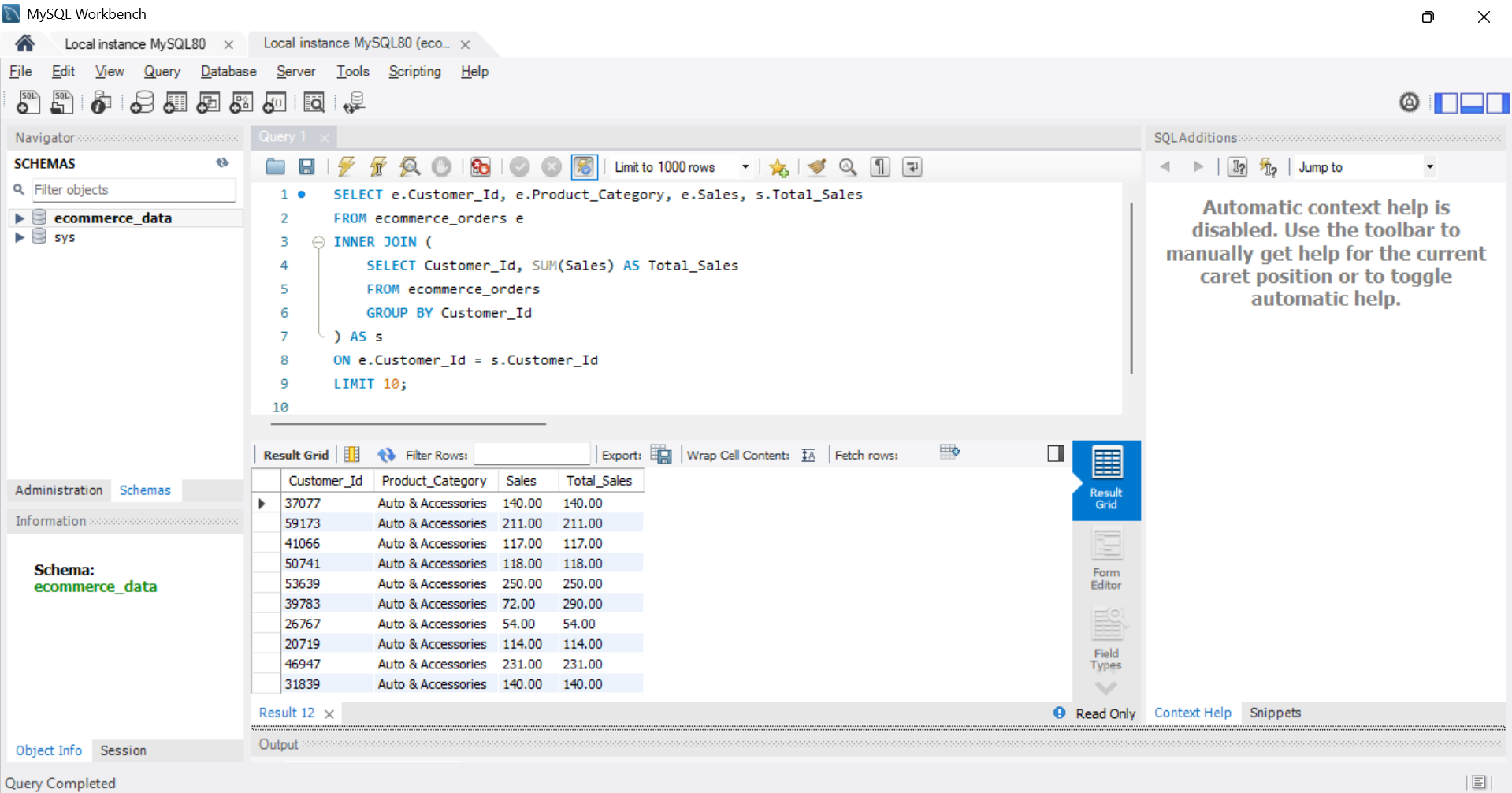
FROM ecommerce\_orders

GROUP BY Customer\_Id

) AS s

ON e.Customer\_Id = s.Customer\_Id

LIMIT 10;



c.Write subqueries

SELECT Customer\_Id, SUM(Sales) AS TotalSales

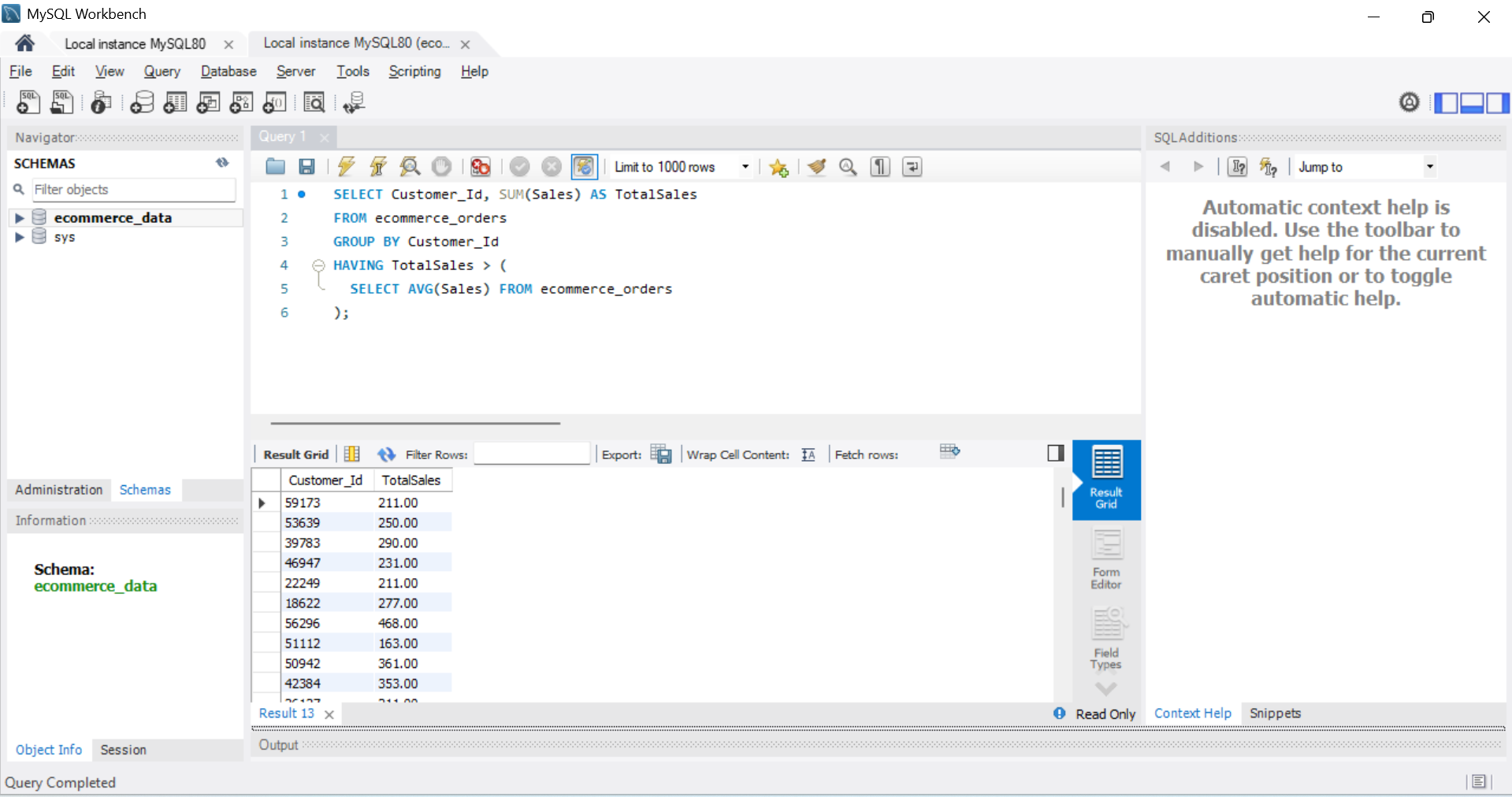
FROM ecommerce\_orders

GROUP BY Customer\_Id

HAVING TotalSales > (

SELECT AVG(Sales) FROM ecommerce\_orders

);

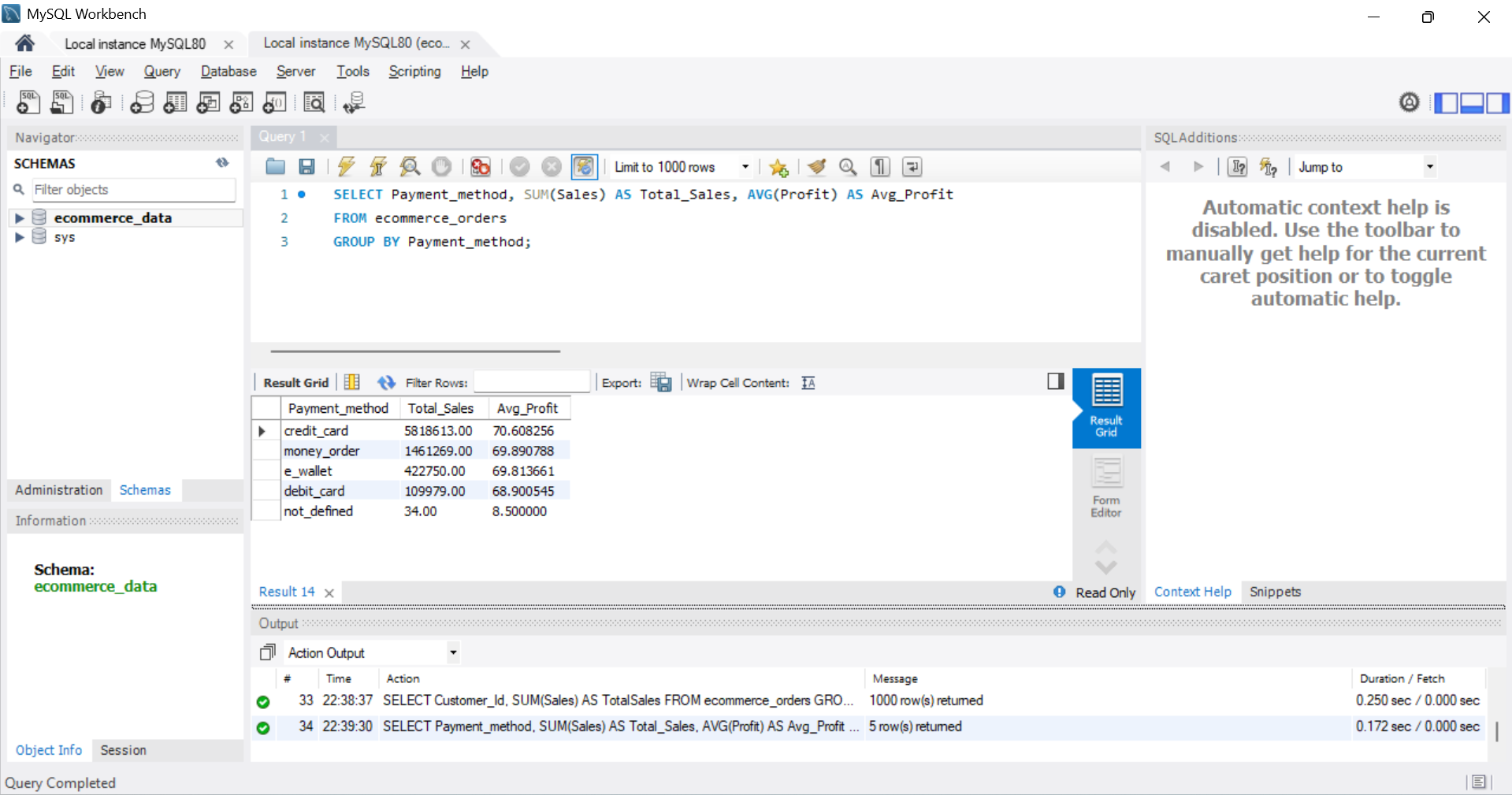


d.Use aggregate functions (SUM, AVG)

SELECT Payment\_method, SUM(Sales) AS Total\_Sales, AVG(Profit) AS Avg\_Profit

FROM ecommerce\_orders

GROUP BY Payment\_method;



e.Create views for analysis

CREATE VIEW profit\_by\_category AS

SELECT Product\_Category,

SUM(Sales) AS Total\_Sales,

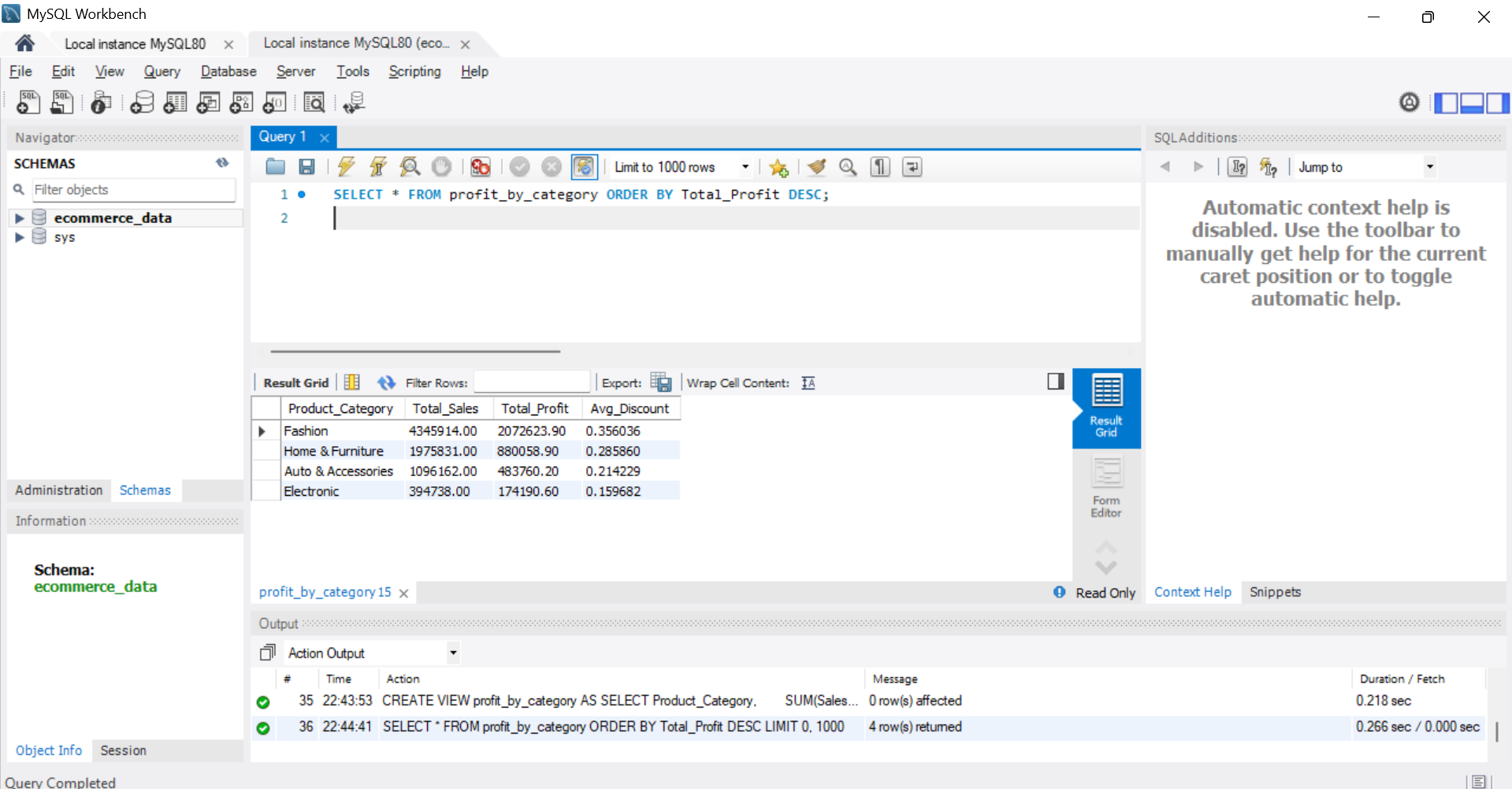
SUM(Profit) AS Total\_Profit,

AVG(Discount) AS Avg\_Discount

FROM ecommerce\_orders

GROUP BY Product\_Category;

SELECT \* FROM profit\_by\_category ORDER BY Total\_Profit DESC;



f.Optimize queries with indexes

CREATE INDEX idx\_customer\_id ON ecommerce\_orders (Customer\_Id);

CREATE INDEX idx\_product\_category ON ecommerce\_orders (Product\_Category);

CREATE INDEX idx\_order\_priority ON ecommerce\_orders (Order\_Priority);

