**Database Systems (CY-T)**

***Group members:***

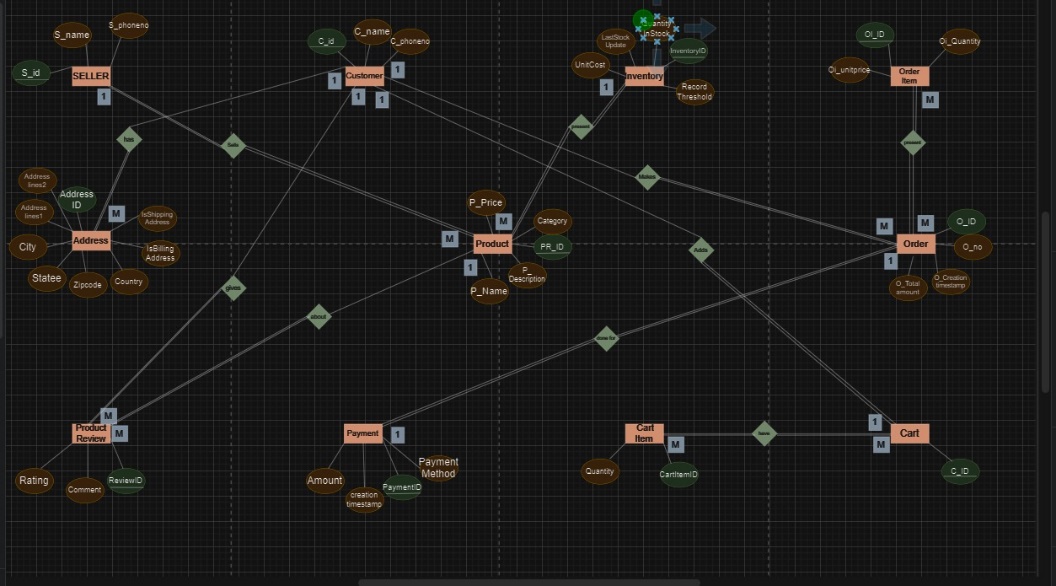
Minam Faisal (21i-1901)

Momenah Saif (21i-1909)

Assignment-2

Question\_1:

ERD Diagram:



**Explanation:**

The ERD diagram represents a simple e-commerce system. It has the following entities:

1. \*Customer:\* Customers have a name, address, and phone number.

2. \*Product:\* Products have a name, price, and category.

3. \*Order:\* Orders have a customer ID, product ID, and quantity ordered.

4. \*Payment:\* Payments have an order ID and payment amount.

5. \*Cart:\* Carts have a customer ID and a list of products that have been added to the cart.

The relationships between the entities are as follows:

\* A customer can place many orders.

\* An order can be placed by only one customer.

\* An order can contain one or more products.

\* A product can be included in many orders.

\* A customer can make many payments.

\* A payment is associated with only one order.

\* A customer can have many carts.

\* A cart is associated with only one customer.

Here is a simple explanation of the ERD diagram in 15 points:

1. \*Entities:\* The ERD diagram has five entities: Customer, Product, Order, Payment, and Cart.

2. \*Attributes:\* Each entity has a set of attributes. For example, the Customer entity has the attributes name, address, and phone number.

3. \*Primary keys:\* Each entity has a primary key that uniquely identifies each instance of the entity. For example, the Customer entity has the primary key customer ID.

4. \*Relationships:\* The ERD diagram shows the relationships between the entities. For example, the relationship between the Customer and Order entities is a one-to-many relationship, meaning that one customer can place many orders, but an order can be placed by only one customer.

5. \*Cardinality:\* the cardinality of a relationship specifies how many instances of one entity can be related to how many instances of another entity. For example, the cardinality of the relationship between the Customer and Order entities is 1: M, meaning that one customer can place many orders, but an order can be placed by only one customer.

Here are some additional points to explain the ERD diagram in more detail:

\* The \*Order\* entity has a foreign key that references the \*Customer\* entity. This means that each order must be associated with a valid customer ID.

\* The \*Order\* entity also has a foreign key that references the \*Product\* entity. This means that each order must be for at least one valid product.

\* The \*Payment\* entity has a foreign key that references the \*Order\* entity. This means that each payment must be associated with a valid order ID.

\* The \*Cart\* entity has a foreign key that references the \*Customer\* entity. This means that each cart must be associated with a valid customer ID.

The ERD diagram can be used to design a relational database for the e-commerce system. Each entity would be represented by a table in the database. The relationships between the entities would be represented by foreign keys in the tables.

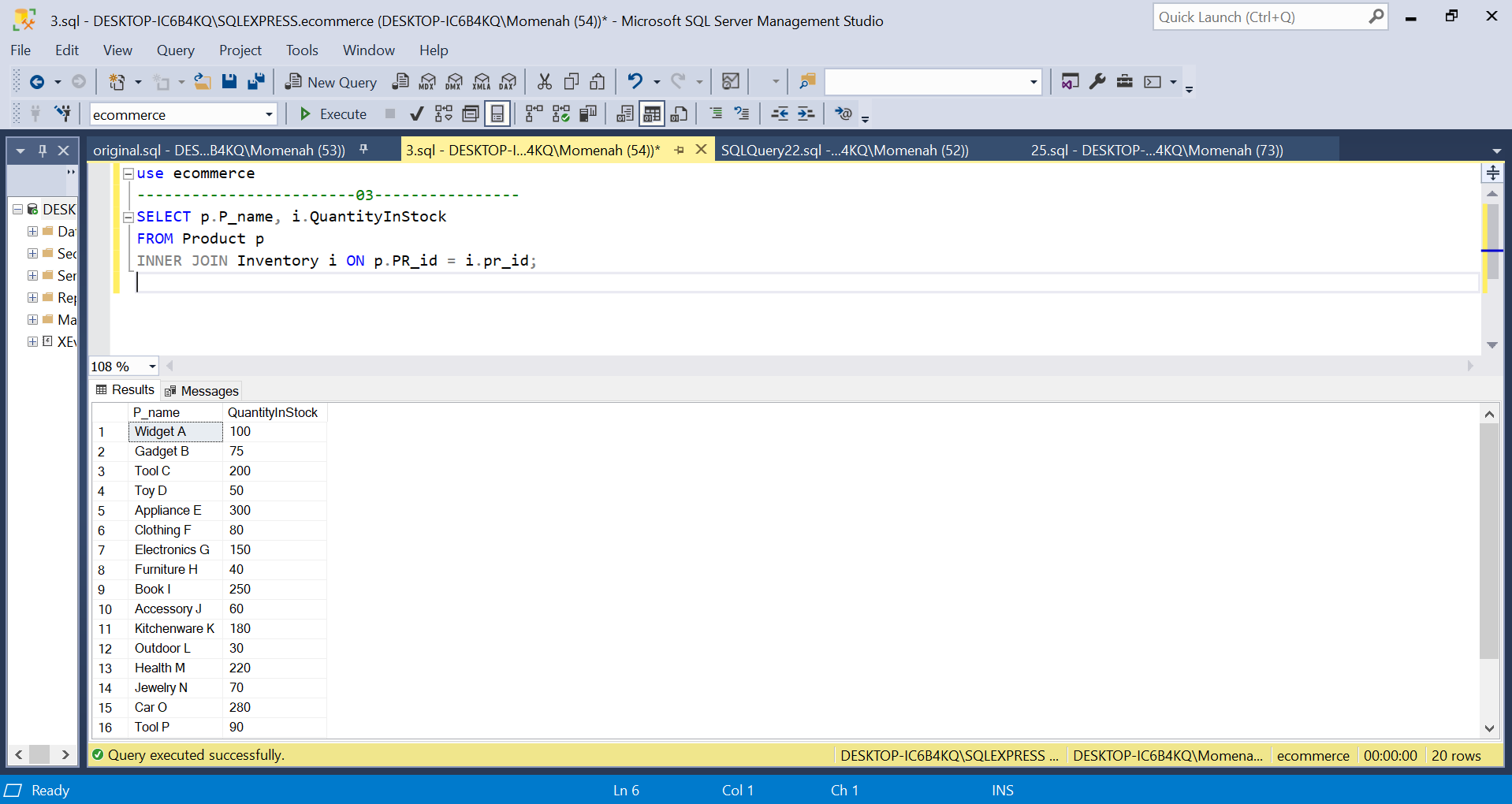
Question\_2:

Relational Model:

* SELLER (S\_id,S\_name,S\_phoneNo)
* Address (AddressID, Addressline1, Addressline2, City,Statee,Zipcode, Country,IsBillingAddress,IsShippingAddress)
* ProductReview(ReviewID,Comment,Rating,PR\_id,C\_id)
* Inventory (InventoryID, RecorfThreashold, QuantityInStock, LastStockUpdate,UnitCost)
* Orderitem(OI\_ID, Oi\_Quantity, Oi\_unitprice)
* Present(OI\_ID,O\_OD)
* Cart(C\_ID)
* Have(C\_ID,CartitemID)
* Cartitem(cartitemID, Quantity)
* OrderPayment(PaymentID, Amount,creationtimestamp, PayementMethod, O\_ID, O\_no, O\_creationtimestamp,O\_totalamount,C\_id)

Question\_3:

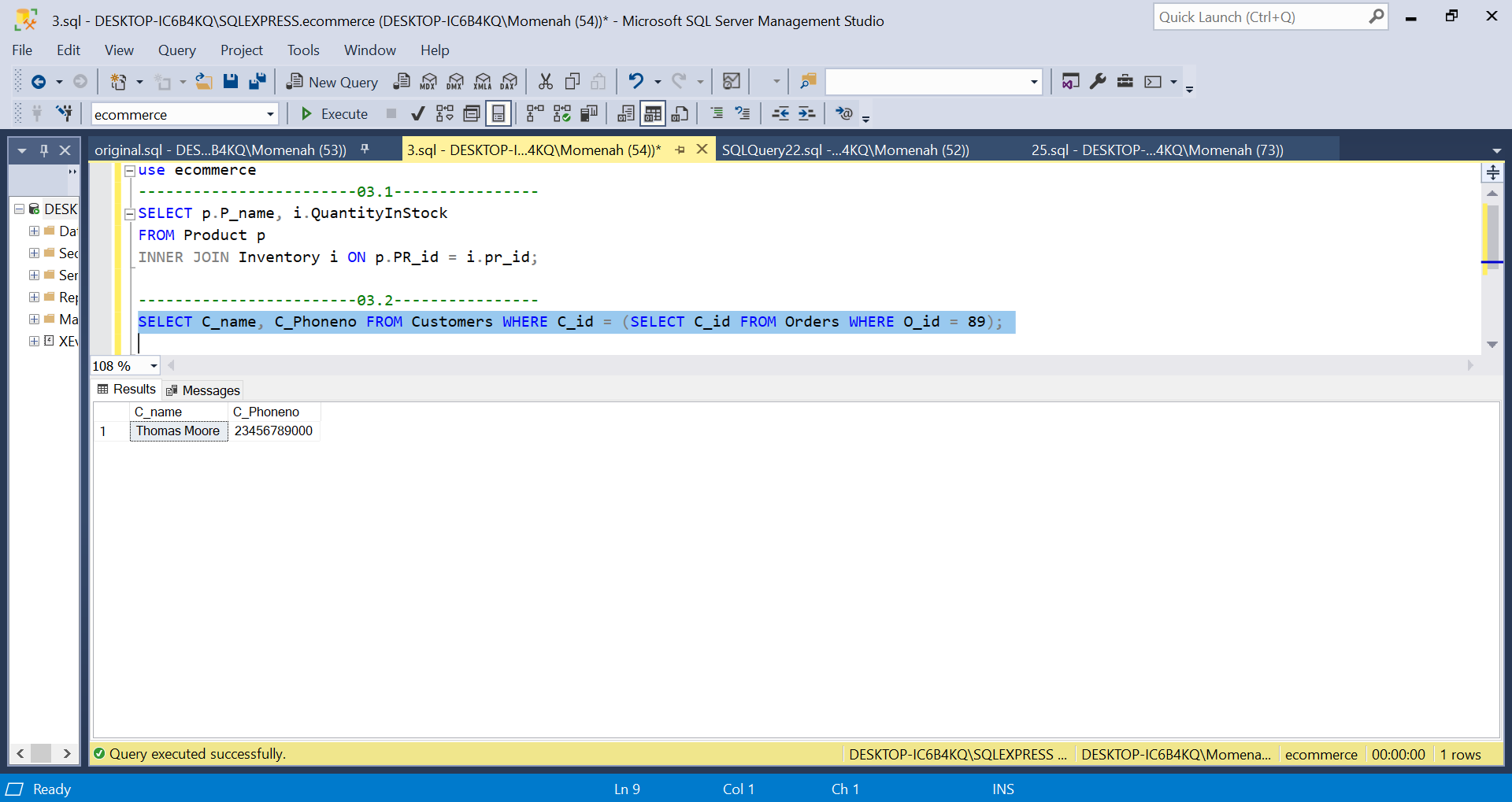
Query1:



Importance:

The "Product" table is crucial for managing product inventory. This query helps keep track of the available products and their current stock quantities. It is essential for maintaining accurate inventory information and ensuring products are available for customers to purchase.

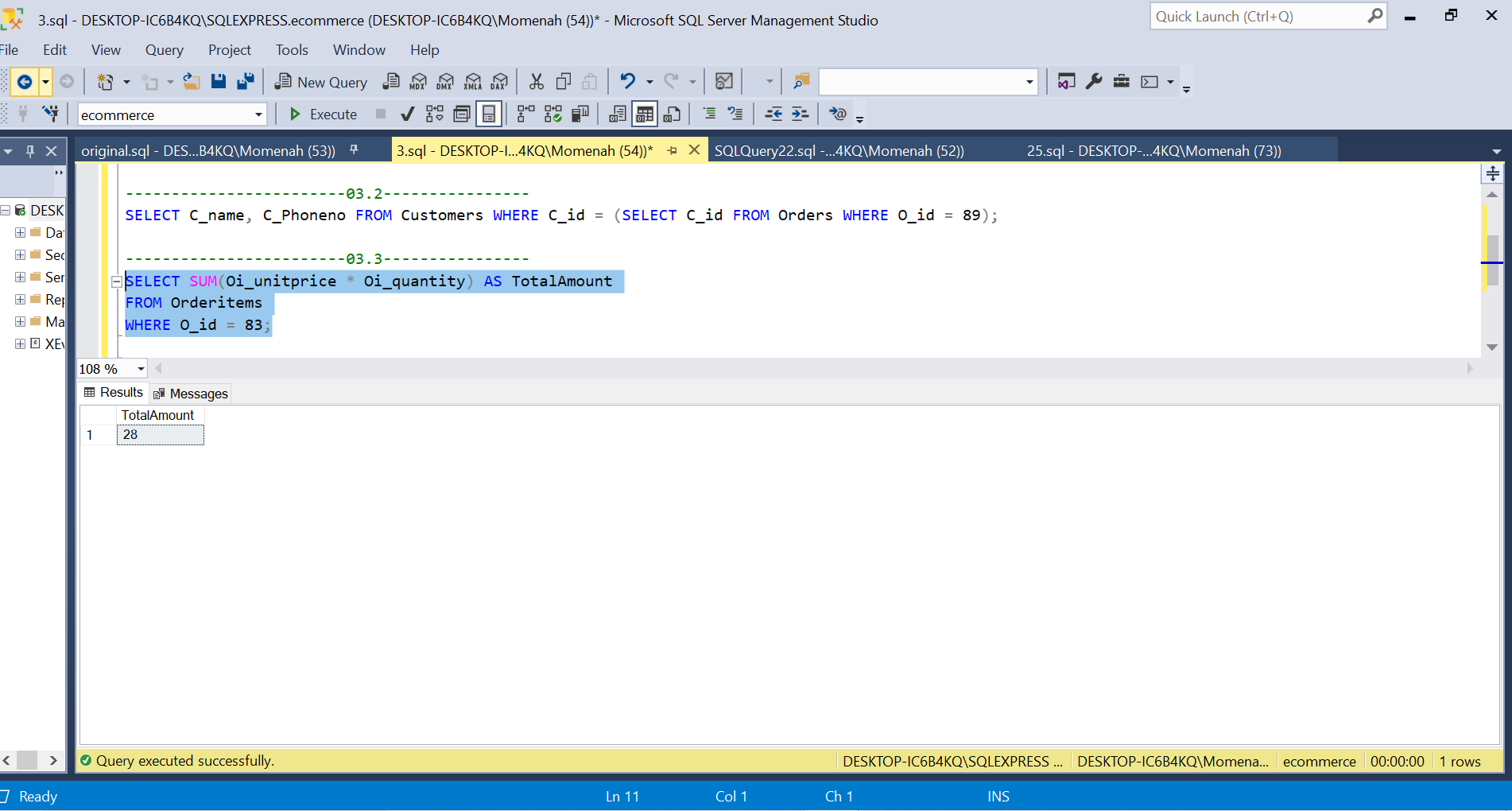
Query2:



Importance:

The "Customer" table stores customer details such as names and contact information. This query helps in associating customers with their orders, allowing you to provide customer support, track order history, and improve customer experience.

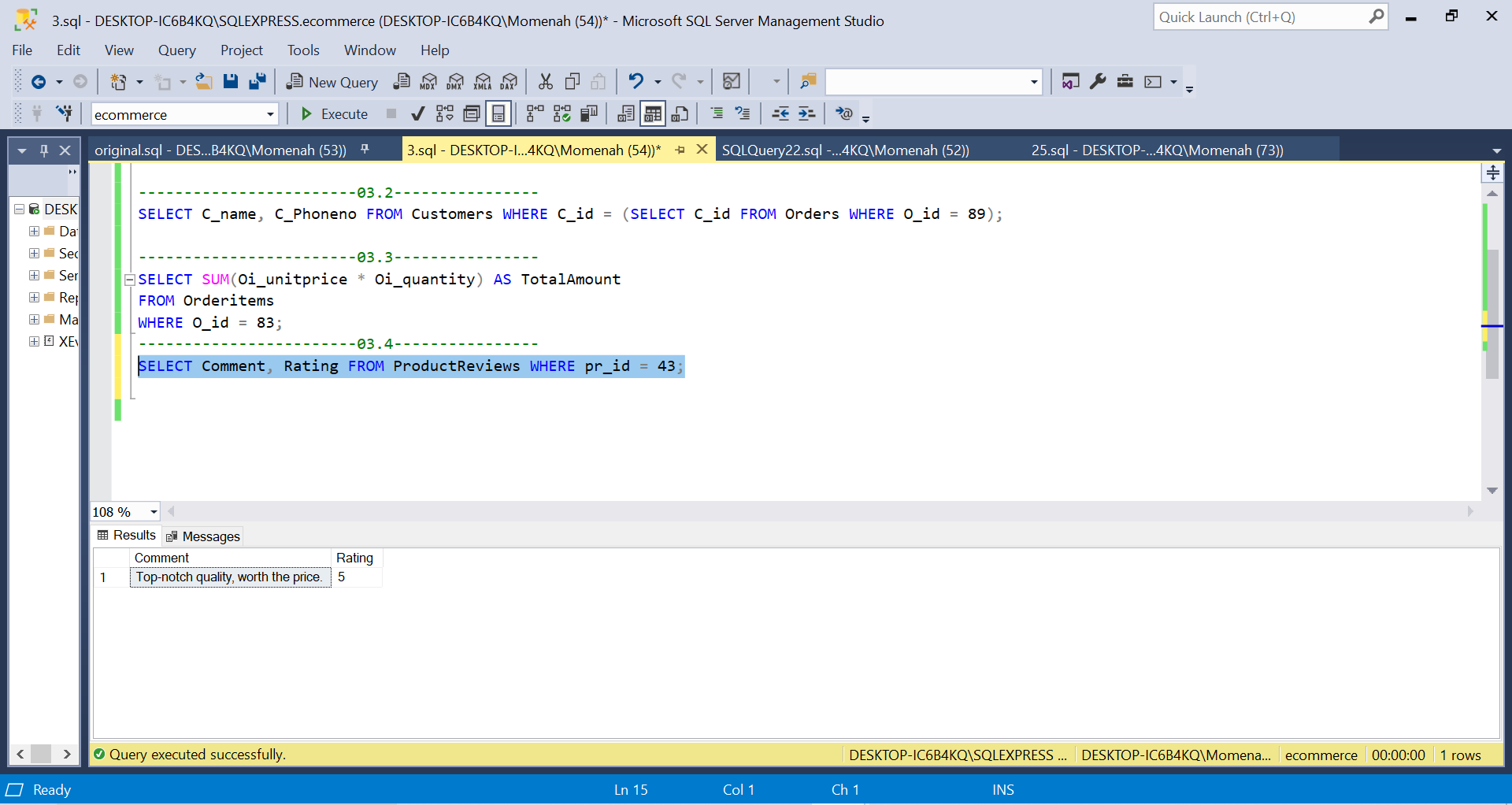
Query3:



Importance:

The "OrderItem" table records individual products within an order and their quantities. This query helps in calculating the total cost of a specific order, which is crucial for billing, accounting, and financial management.

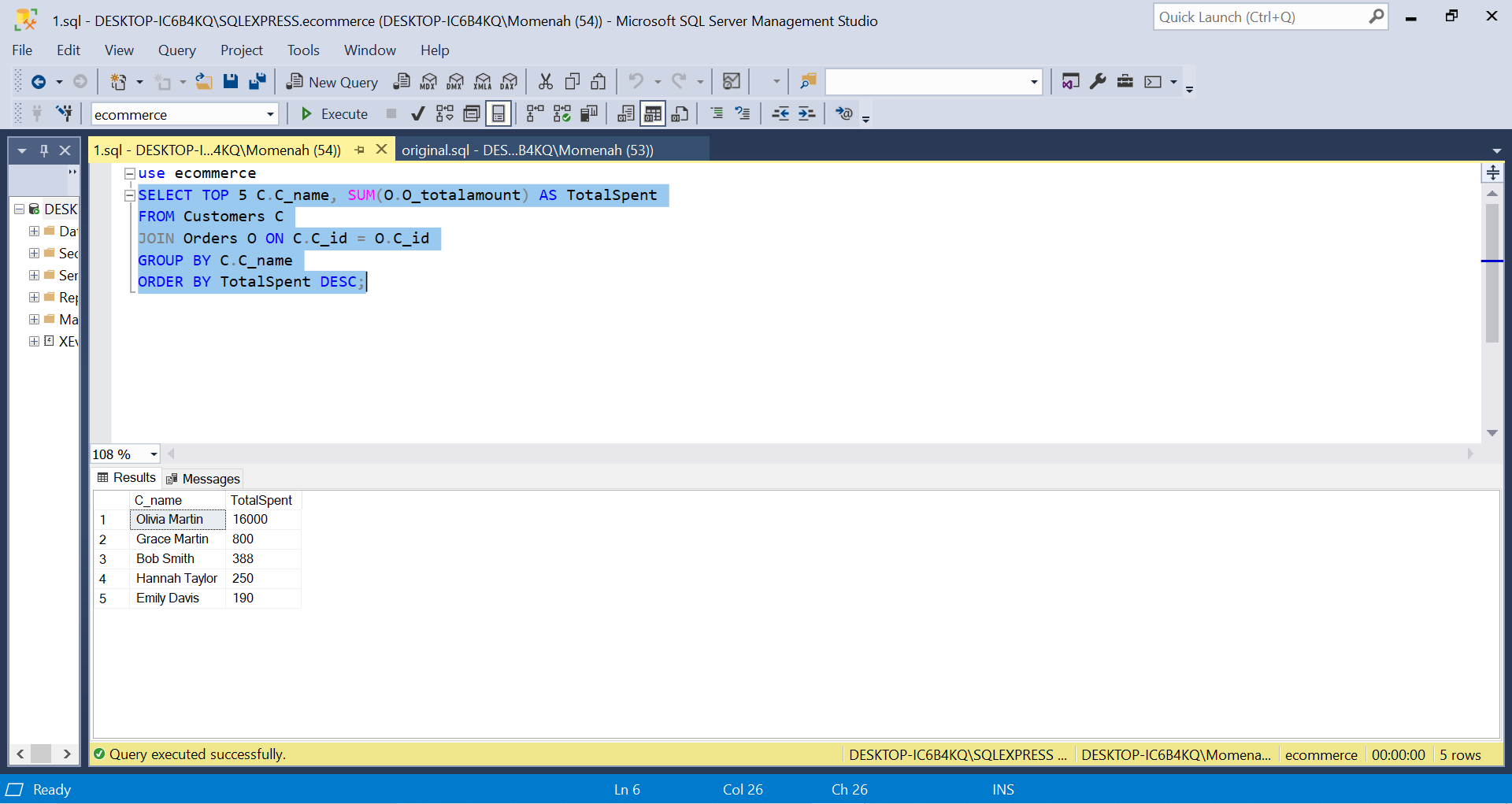
Query4:



Importance:

The "ProductReviews" table stores reviews and ratings provided by customers. This query allows customers to make informed purchase decisions and provides valuable feedback to improve product quality and customer satisfaction.

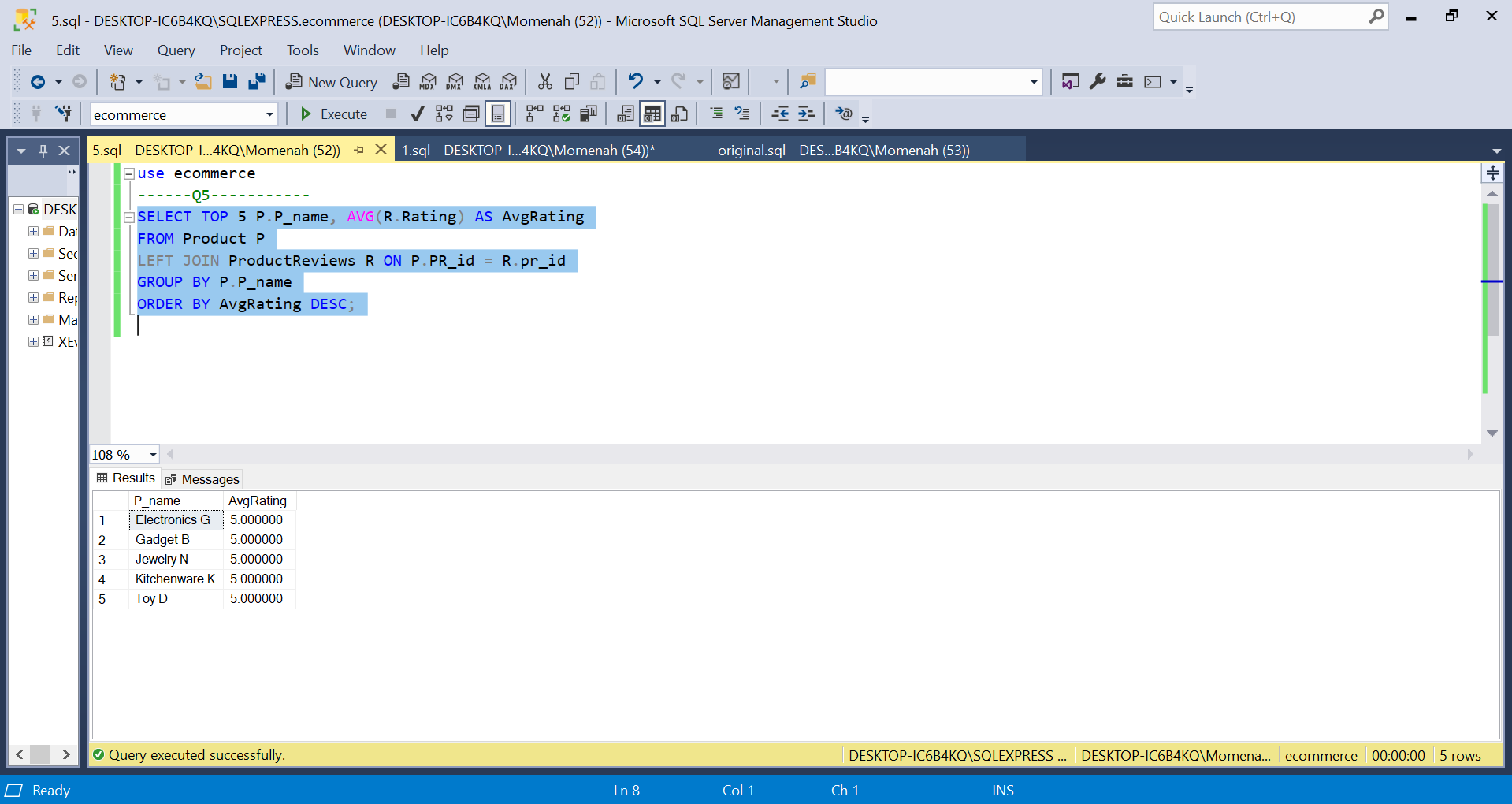
Question\_4:



Explanation:

1. The C\_id field is used in this code to link the Customers and Orders tables. This implies that the code will locate all of a customer's orders for that customer.
2. The results are then grouped by the C\_name column in the code. This indicates that the code will figure out how much money each consumer has spent overall.
3. The results are then sorted in reverse order by the TotalSpent column. The top 5 clients who have spent the most money will be displayed by the code as a result.

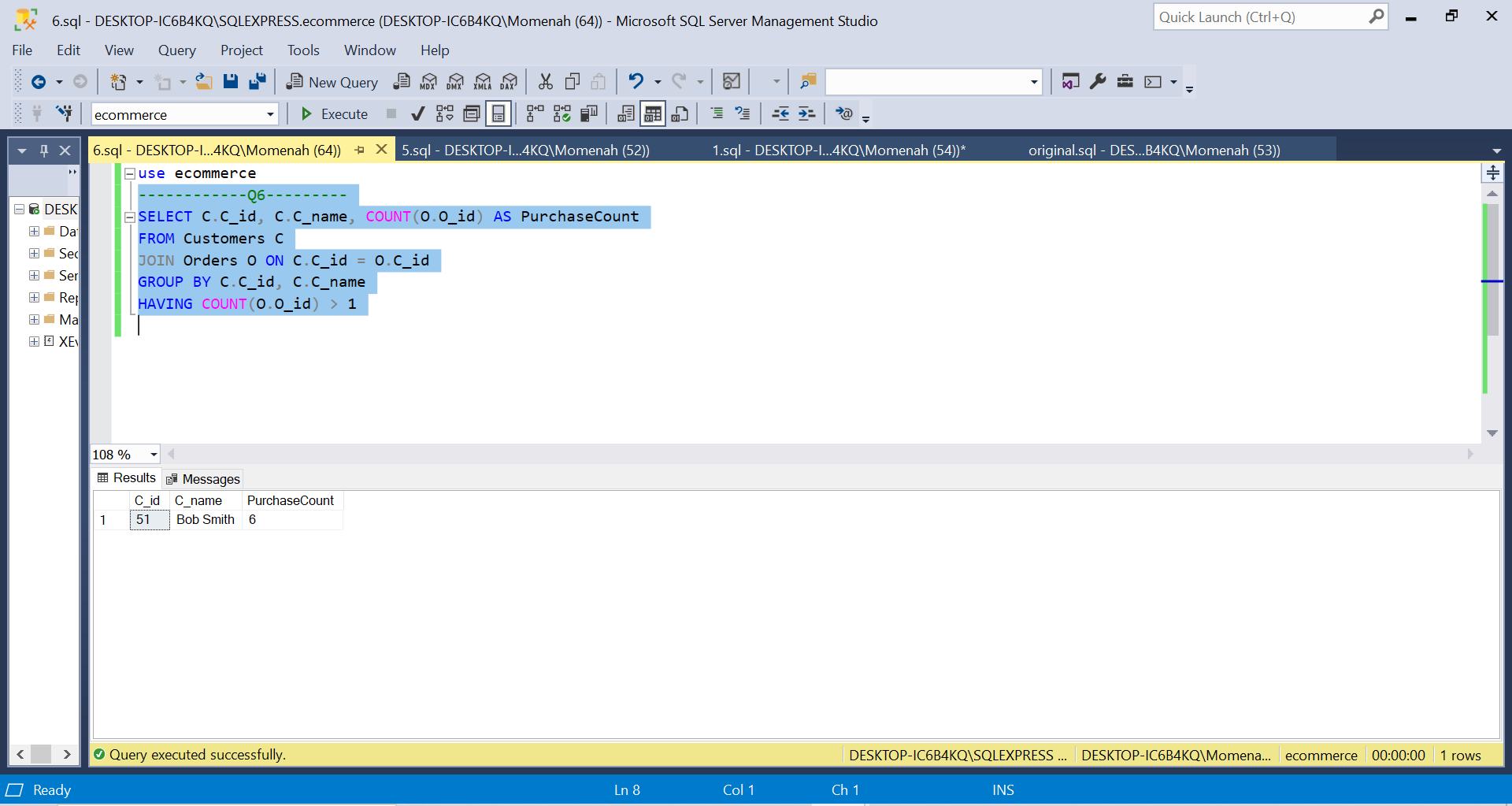
Question\_5:



Explanation:

1. The PR\_id field is used to link the Product and ProductReviews tables. This implies that the algorithm will locate all of a product's reviews.
2. The P\_name field is then used to group the results. This means that the average rating for each product will be determined by the code.
3. The results are then sorted by the AvgRating column in decreasing order by the code. This indicates that the top 5 goods with the highest average ratings will be displayed by the code.

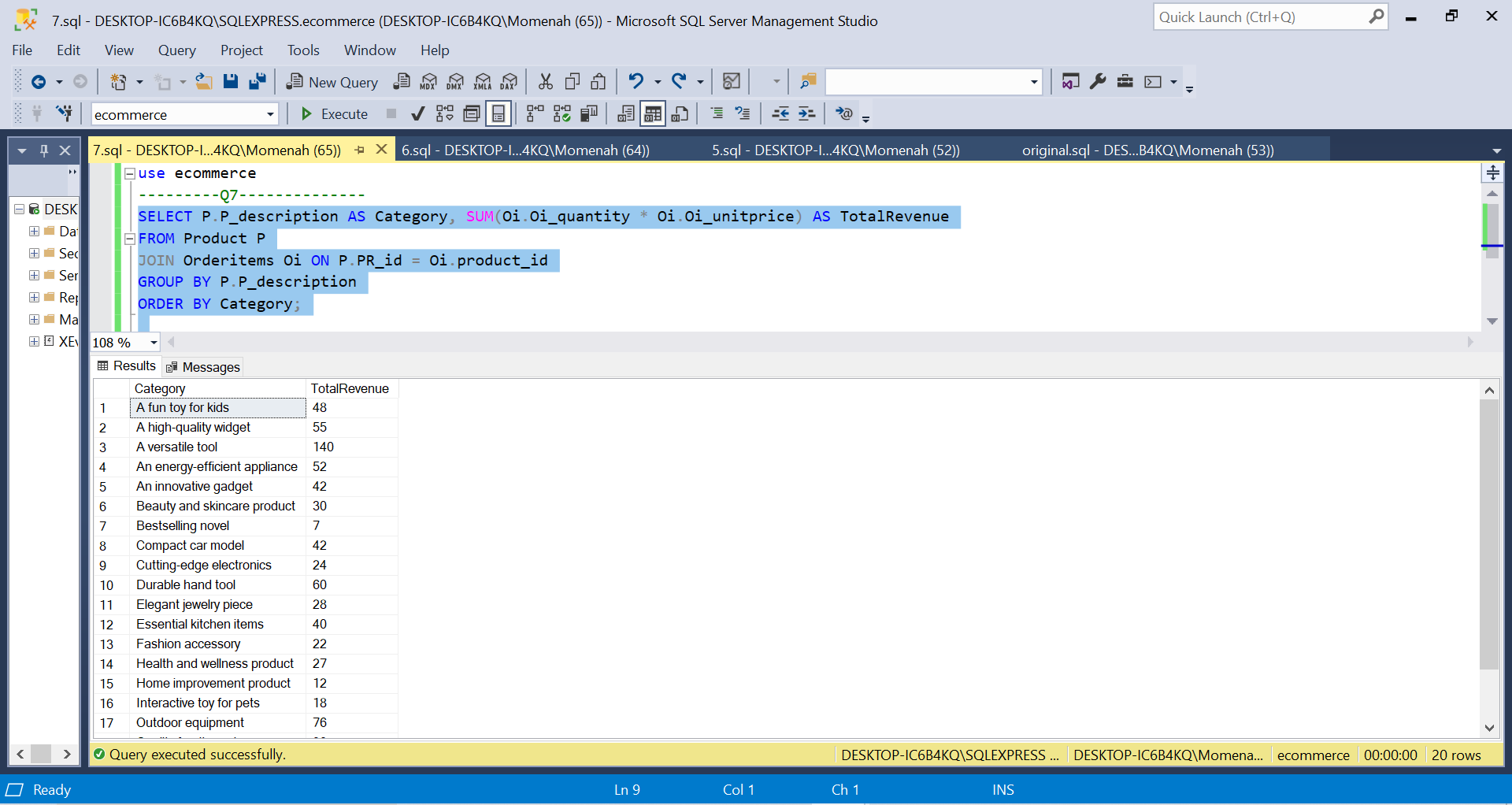
Question\_6:



Explanation:

1. The C\_id field is used in this code to link the Customers and Orders tables. This implies that the code will locate all of a customer's orders for that customer.
2. The C\_id and C\_name columns are then used to group the results. This indicates that the code will tally how many orders each customer has placed.
3. Then, the results are filtered such that they only include clients who have made many purchases. The filter is applied using the HAVING clause.
4. The results are then sorted by the AvgRating column in decreasing order by the code. The code will, therefore, display the clients with the highest average ratings first.

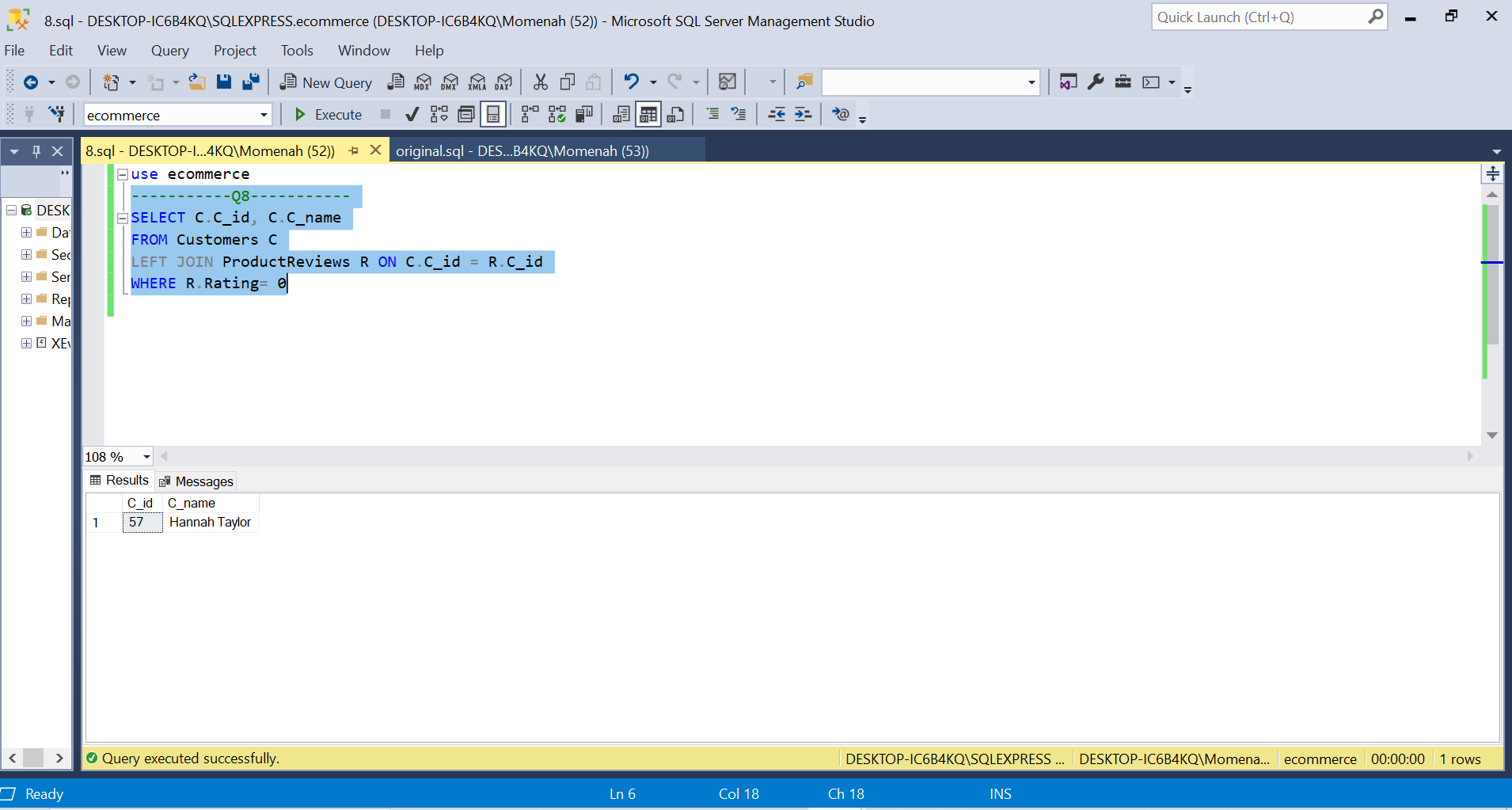
Question\_7:



Explanation:

1. With the use of the product\_id field, this code combines the OrderItems and Product databases. This implies that the code will discover every order item for every product.
2. The P\_description column is then used to group the results. The total income for each product category will therefore be determined by the code.
3. The results are then ordered by the Category column in ascending order by the code. This indicates that the product categories will be displayed by the code in alphabetical order.

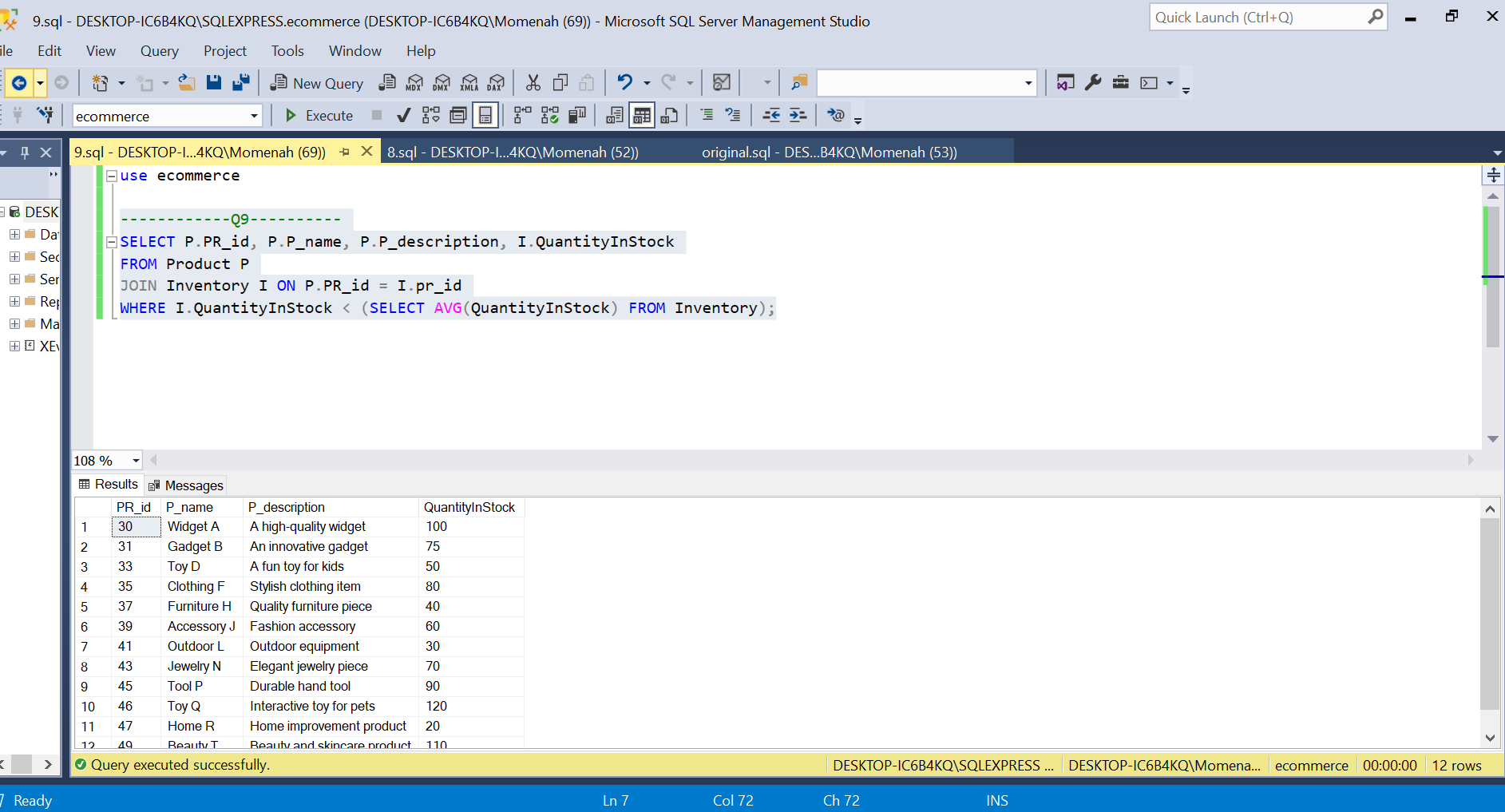
Question\_8:



Explanation:

1. The C\_id field is used in this code to link the Customers and ProductReviews tables. This implies that the code will locate all of the reviews for each client.
2. The code then limits the output to only include reviews with a score of 0 from clients. This filter is applied via the WHERE clause.
3. The C\_id and C\_name columns from the Customers table are then chosen by the code. This means that each user who has submitted a review with a score of 0 will see their customer ID and name in the results.

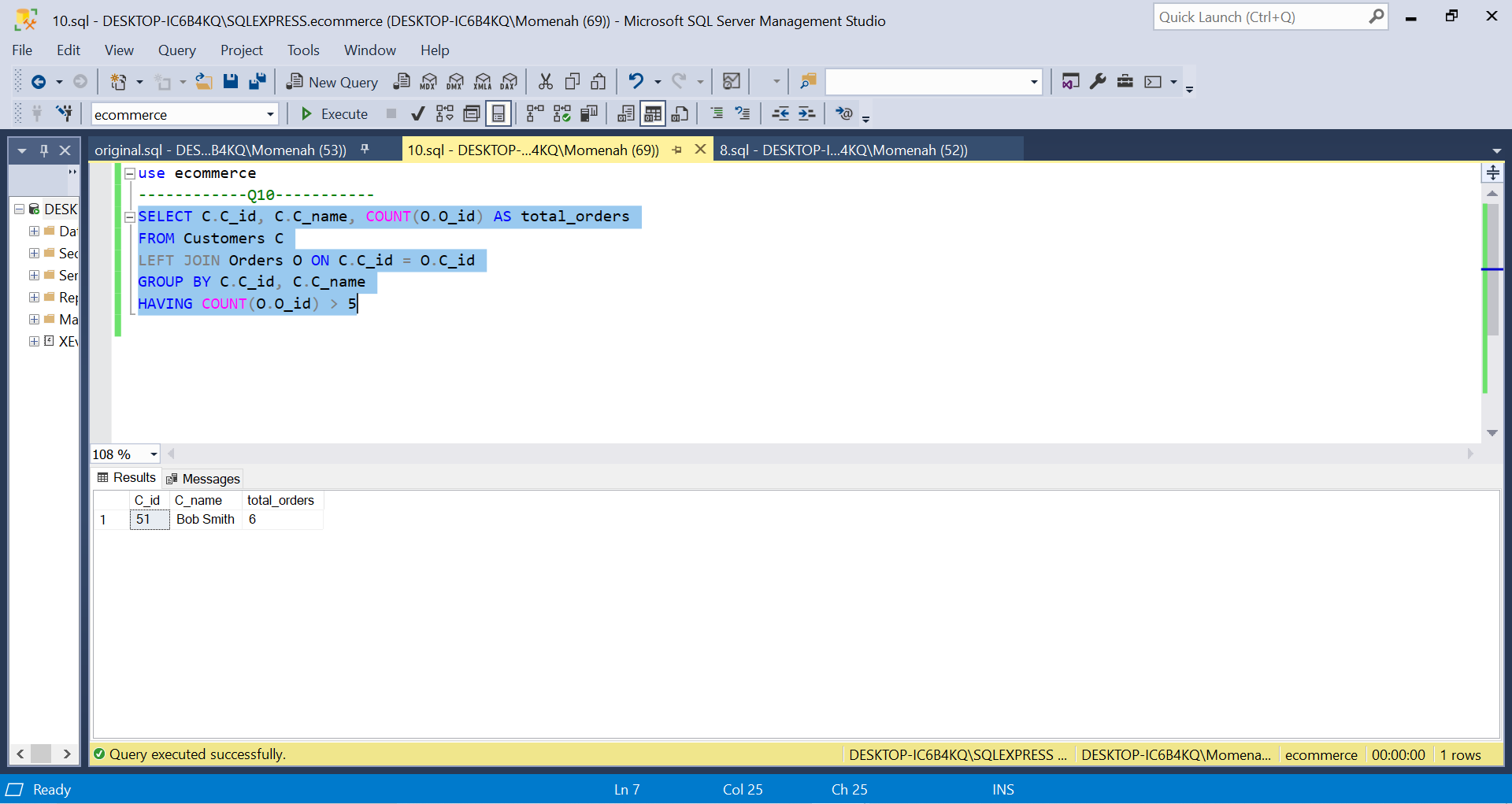
Question\_9:



Explanation:

1. Utilizing the pr\_id column, this code joins the Product and Inventory tables. This indicates that the code will locate the corresponding inventory record for each product.
2. The average amount of stock for all products is then determined by the code. The WHERE clause's subquery is used to accomplish this.
3. The code then refines the outcomes to only display goods that have a quantity in stock that is below the industry average.

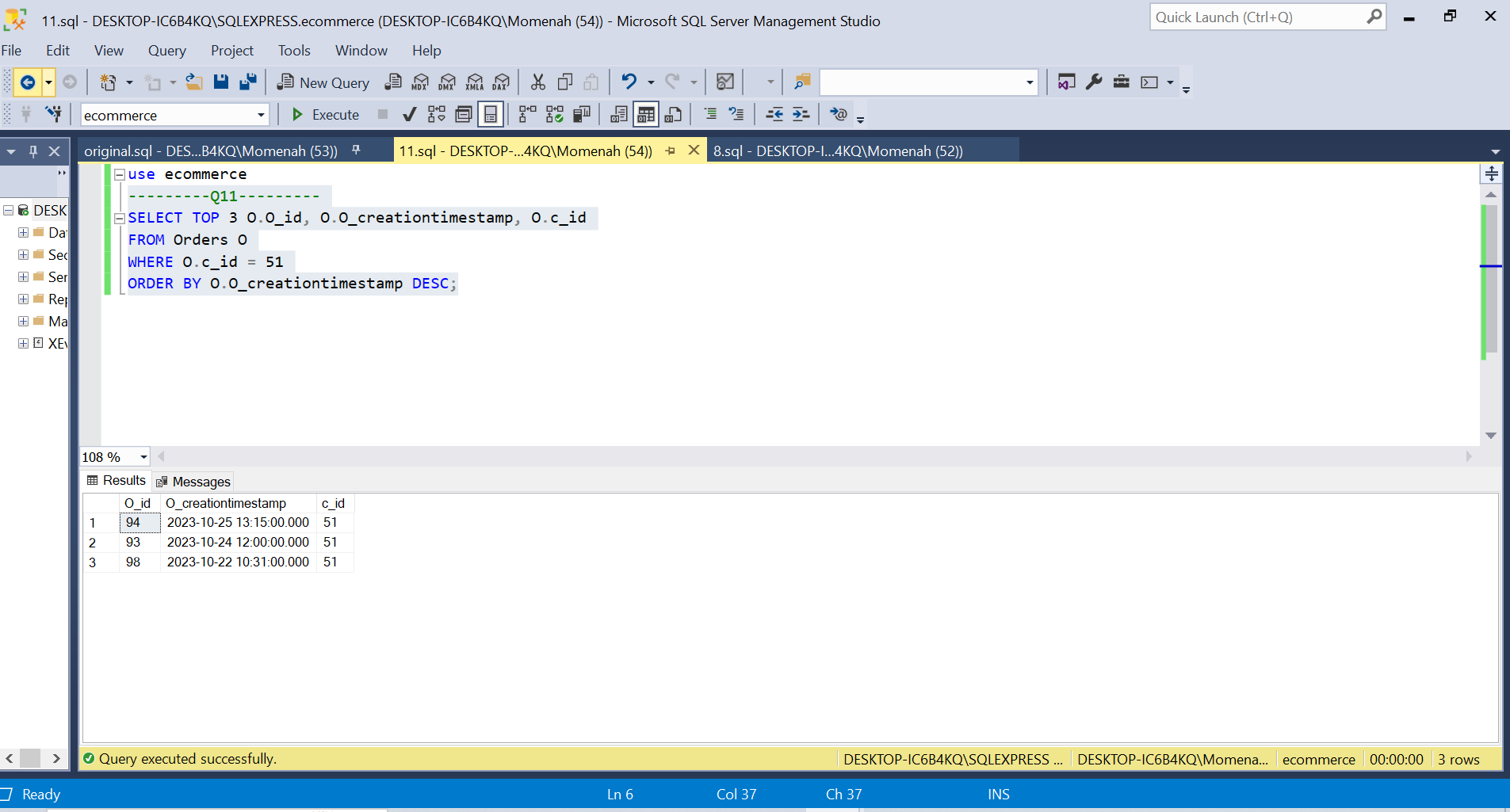
Question\_10:



Explanation:

1. The C\_id column is used in this code to join the Customers and Orders tables. In other words, even if a customer doesn't have any orders, the code will still find all of them.
2. The C\_id and C\_name columns are then used to group the results. This indicates that the code will tally how many orders each customer has placed.
3. The results are then filtered by the code to only display clients who have placed more than five orders. The filter is applied using the HAVING clause.

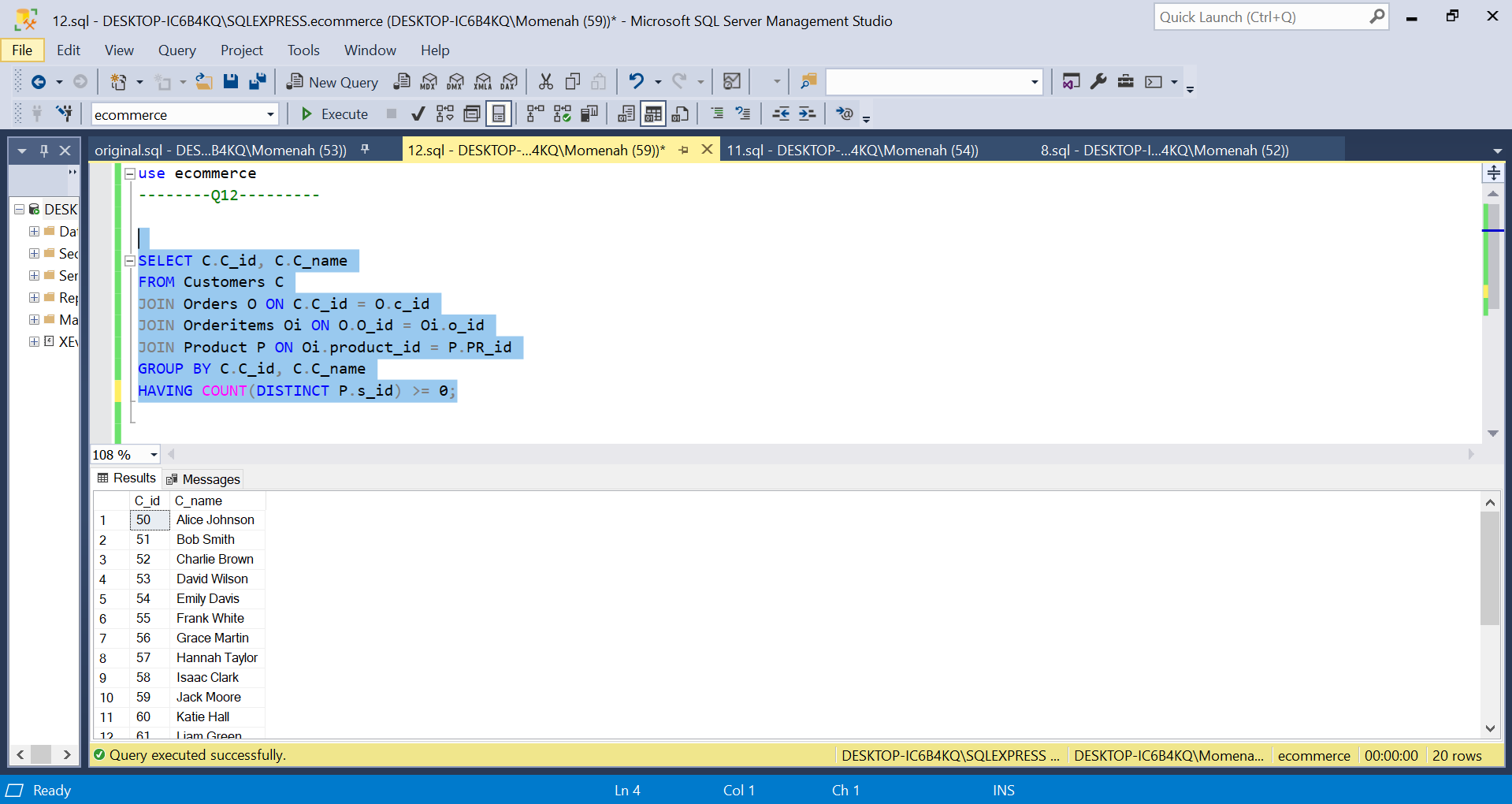
Question\_11:



Explanation:

1. The Orders table's O\_id, O\_creationtimestamp, and c\_id columns are chosen by this code.
2. Then, the code limits the results to only display orders with a c\_id of 51. This means that only orders placed by customer 51 will be displayed by the code.
3. The O\_creationtimestamp column is then used to order the results in reverse chronological order. The code will, therefore, display the most recent orders first.

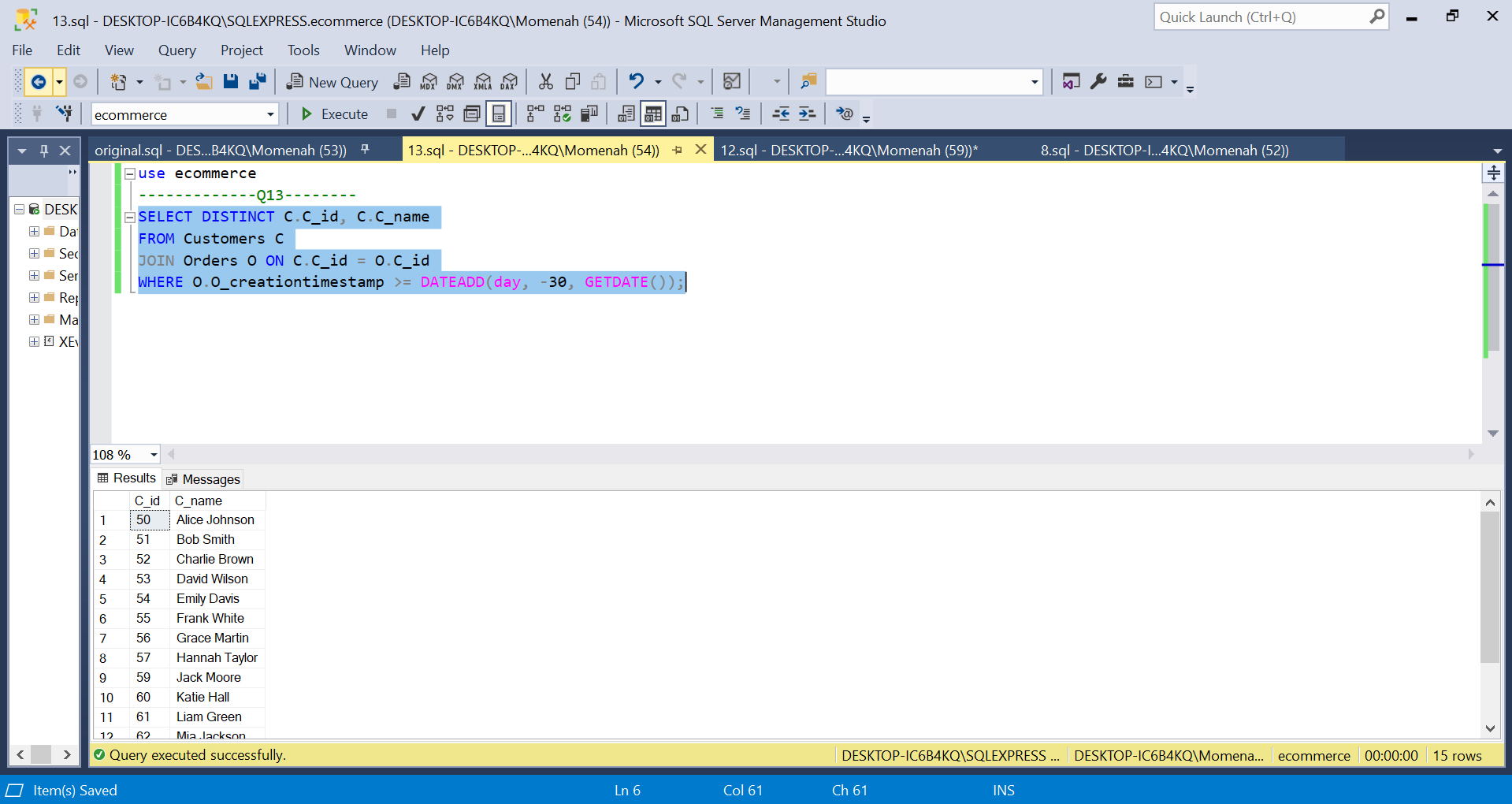
Question\_12:



Explanation:

1. The Customers, Orders, OrderItems, and Product tables are joined together by this code. This means that for each customer, the code will locate all of their orders, along with the order items and products associated with those orders.
2. The C\_id and C\_name columns are then used to group the results. This means that for each customer, the code will count the number of unique suppliers.
3. Then, the results are filtered so that only customers who have made purchases from at least two different suppliers are displayed. The filter is applied using the HAVING clause.

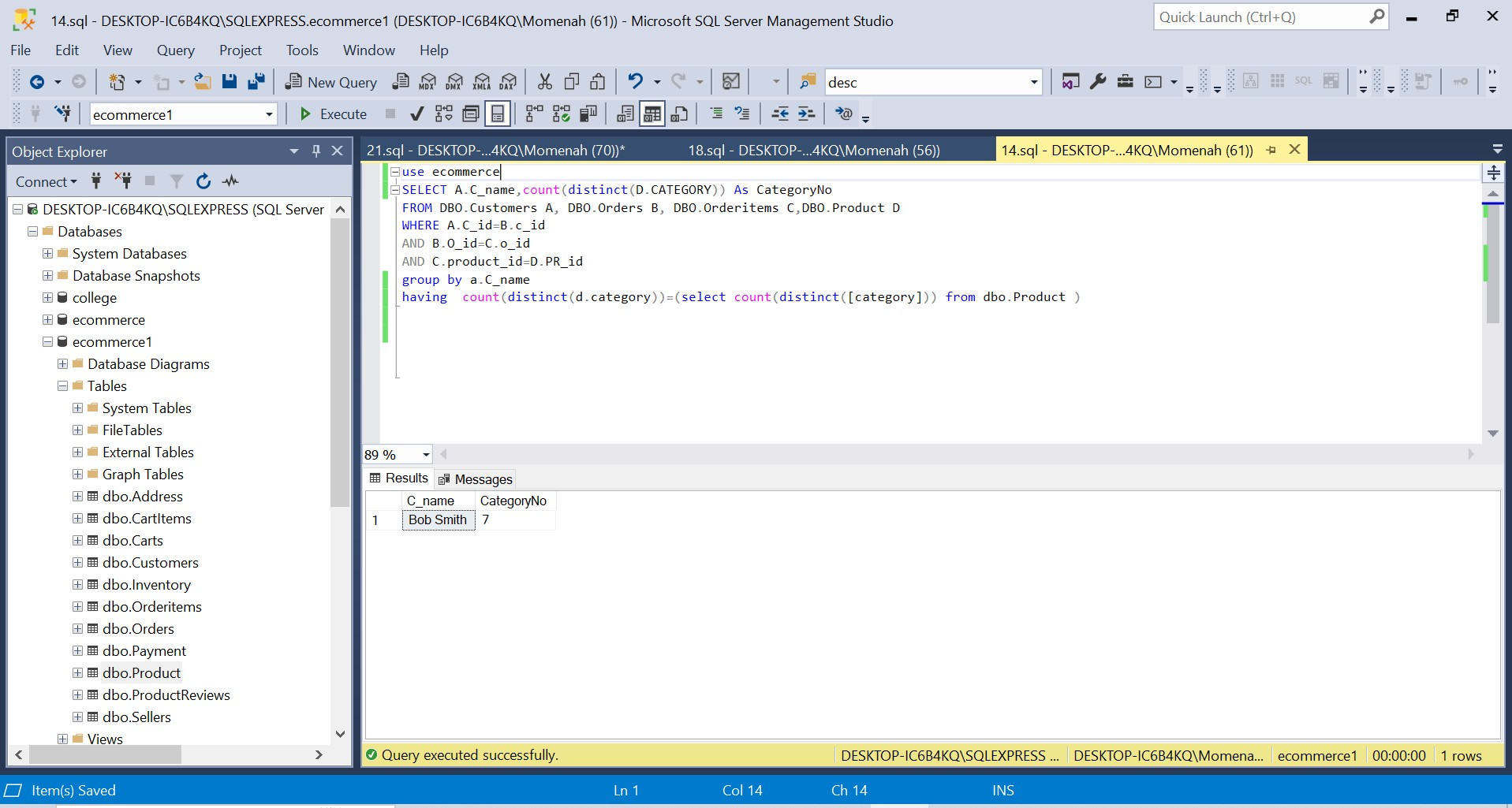
Question\_13:



Explanation:

1. The C\_id column is used in this code to join the Customers and Orders tables. This implies that the code will locate all of a customer's orders for that customer.
2. The results are then filtered by the code to only display recent (30 days) orders. The date 30 days ago is determined using the DATEADD function. This filter is applied via the WHERE clause.
3. The distinct C\_id and C\_name columns from the Customers table are then chosen by the code. This means that even if a customer placed multiple orders during the past 30 days, their information will only appear once in the results.

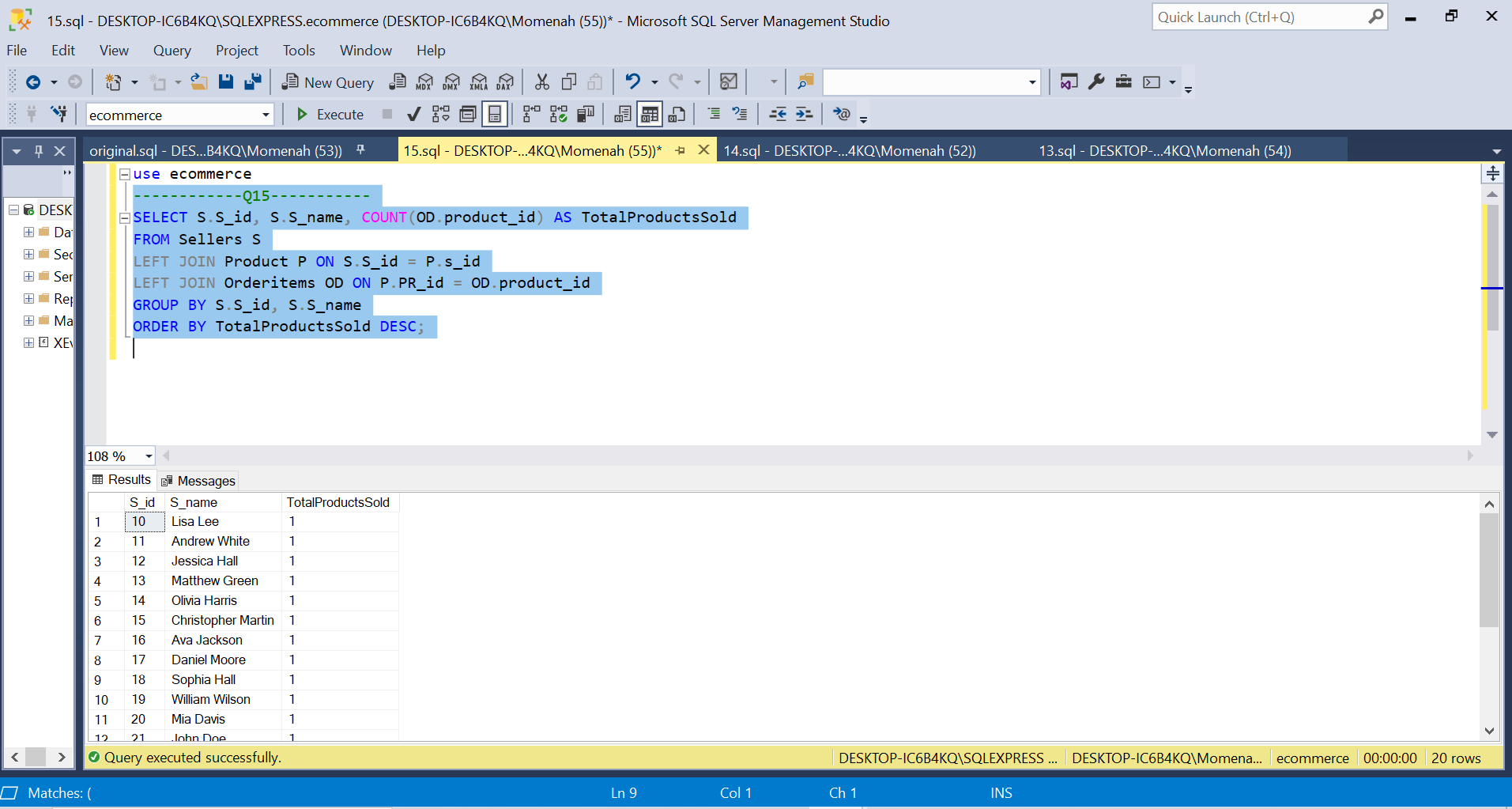
Question\_14:



Explanation:

1. 1. The query is selecting the customer name (A.C\_name) and the number of product categories (count(D.CATEGORY)) for each customer.
2. 2. The query is joining the Customers (A), Orders (B), Orderitems (C), and Product (D) tables together.
3. 3. The query is using the HAVING clause to filter the results to only include customers who have purchased all product categories.

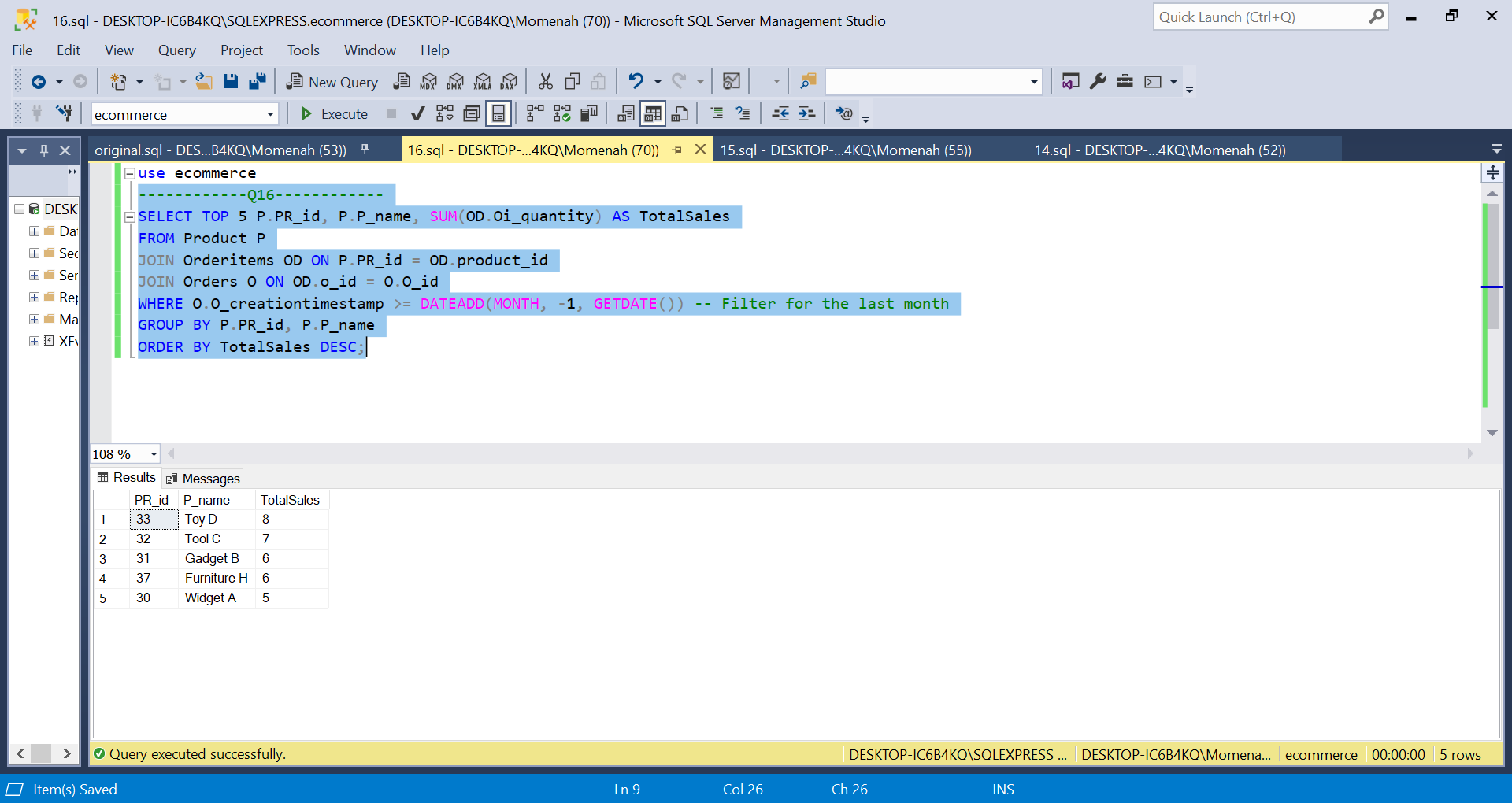
Question\_15:



Explanation:

1. This code uses the S\_id column to perform a left join between the Sellers (S) and Product (P) tables. This implies that the code will locate every product for every seller, even if they don't have any products.
2. The PR\_id column is then used in a left join operation between the Product (P) and OrderItems (OD) tables. In other words, even if a product doesn't have any order items, the code will still find all of them for that product.
3. The S\_id and S\_name columns are then used to group the results. In other words, the code will tally the quantity of goods that were sold by each seller.
4. The results are then sorted in reverse order by the TotalProductsSold column. This means that the code will show the sellers who have sold the most products first.

Question\_16:



Explanation:

1. The Product (P), OrderItems (OD), and Orders (O) tables are joined by the following code. This implies that the code will locate every order item for every product as well as the order for each order item.
2. The results are then filtered by the code to only display recent orders. The date one month ago is determined using the DATEADD function. This filter is applied via the WHERE clause.
3. The PR\_id and P\_name columns are then used to group the results. This indicates that the code will figure out the overall quantity of units sold for each good.
4. The results are then sorted in reverse order by the TotalSales column. As a result, the code will display the top 5 products by number of units sold first.

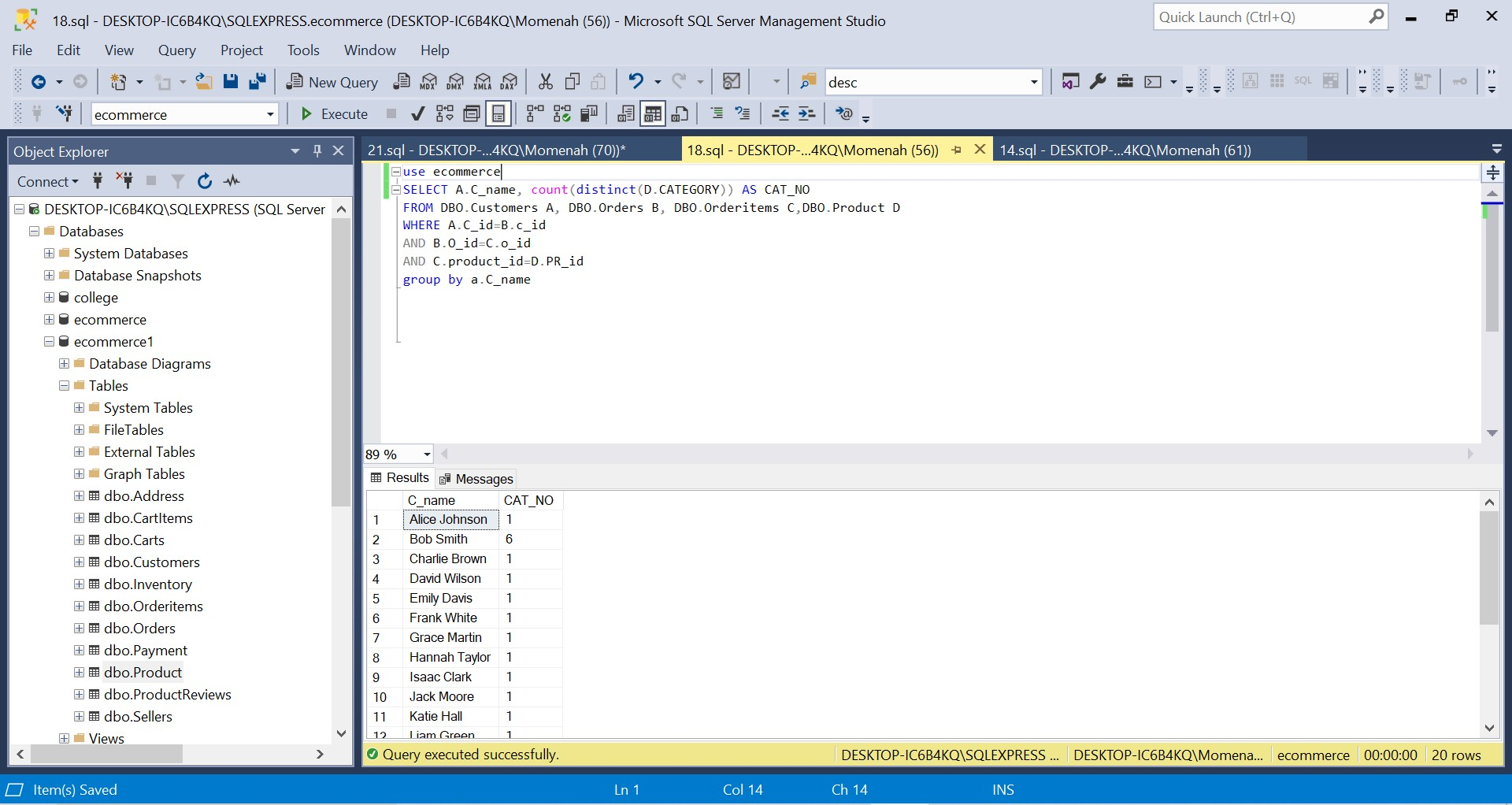
Question\_17:



Explanation:

1. The Customer (C), Order (O), OrderItem (Oi), and Product (P) tables are all joined together by this code. This means that for each customer, the code will locate all of their orders, along with the order items and products associated with those orders.
2. The O\_creationtimestamp column is then used to order the results in reverse chronological order. The code will, therefore, display the most recent orders first.
3. The top 5 rows of the results are then chosen by the code. As a result, only the five most recent orders will be displayed in the results.

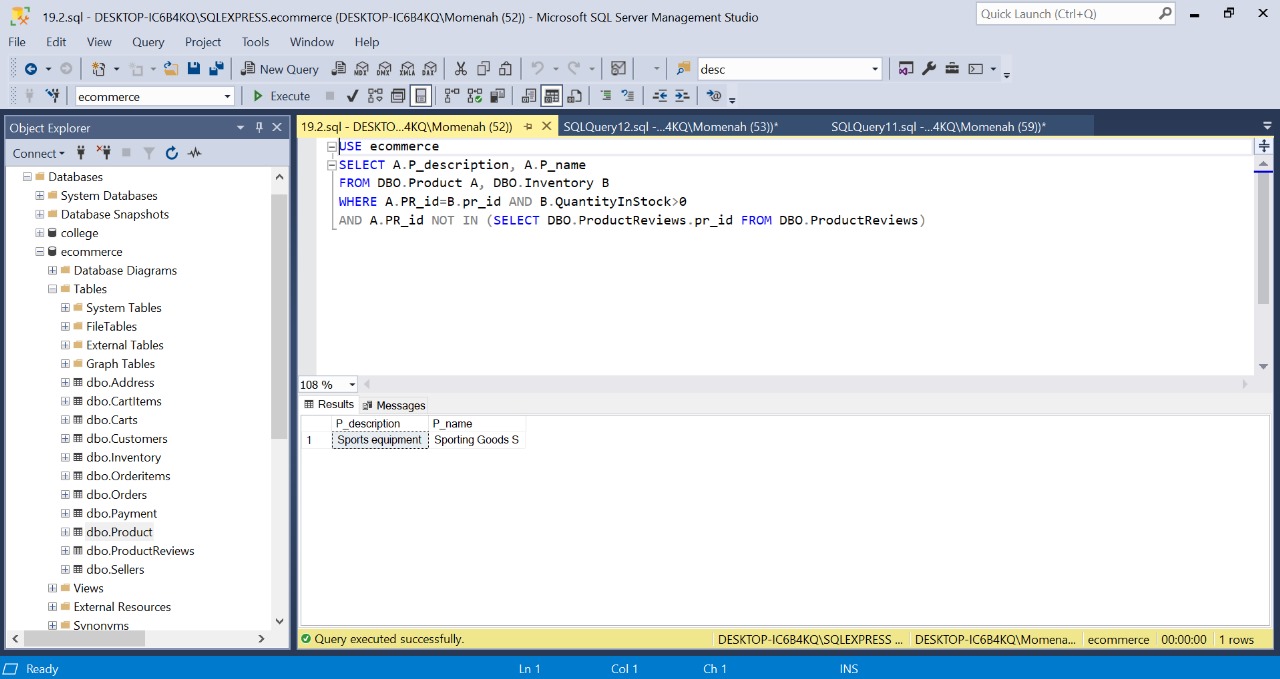
Question\_18:



Explanation:

1. 1. The query is selecting the customer name (A.C\_name) and the number of product categories purchased by each customer (count(D.CATEGORY) AS CAT\_NO).
2. 2. The query is joining the Customers (A), Orders (B), Orderitems (C), and Product (D) tables together.
3. 3. The query is grouping the results by customer name (a.C\_name).

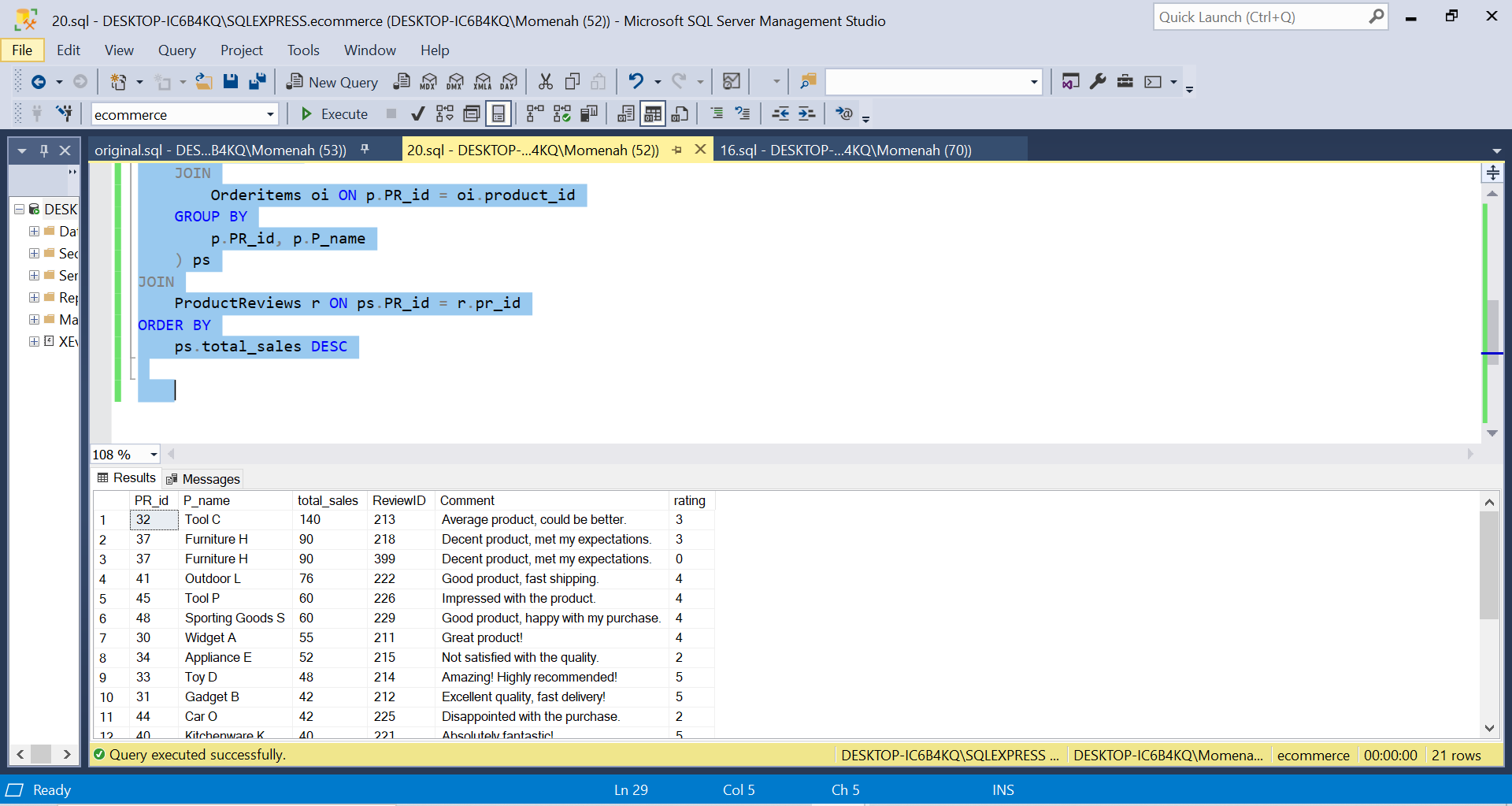
Question\_19:



Explanation:

1. The Product and Inventory tables are first joined using the PR\_id column in this code. This indicates that the code will locate the corresponding inventory record for each product.
2. The code then refines the outcomes to only display items with a stock quantity greater than 0. The QuantityInStock column from the Inventory table is used for this.
3. Then, the code filters the outcomes to only display items without reviews. To accomplish this, a NOT IN clause is used to exclude items from consideration that have a PR\_id in the ProductReviews table.

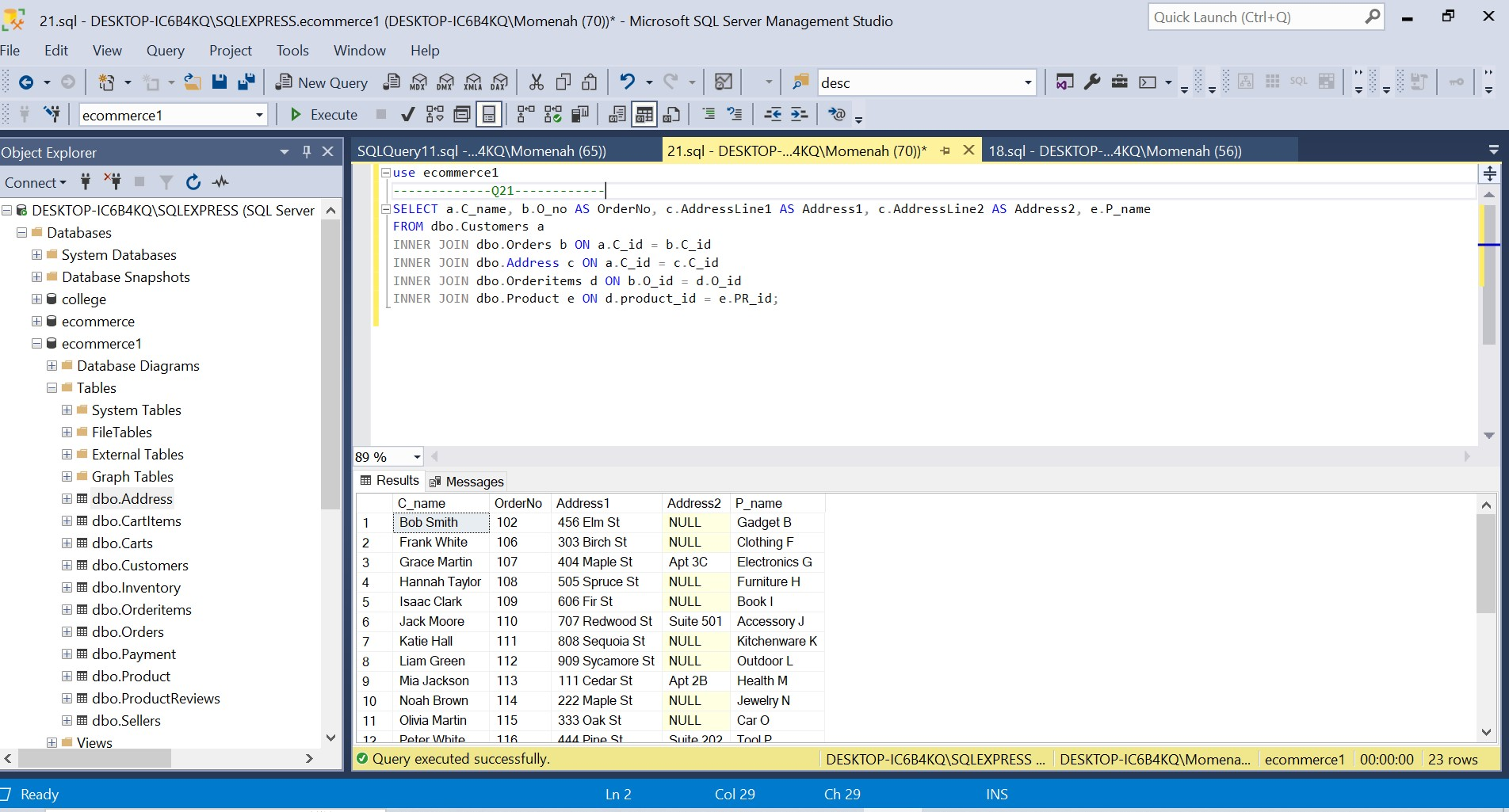
Question\_20:



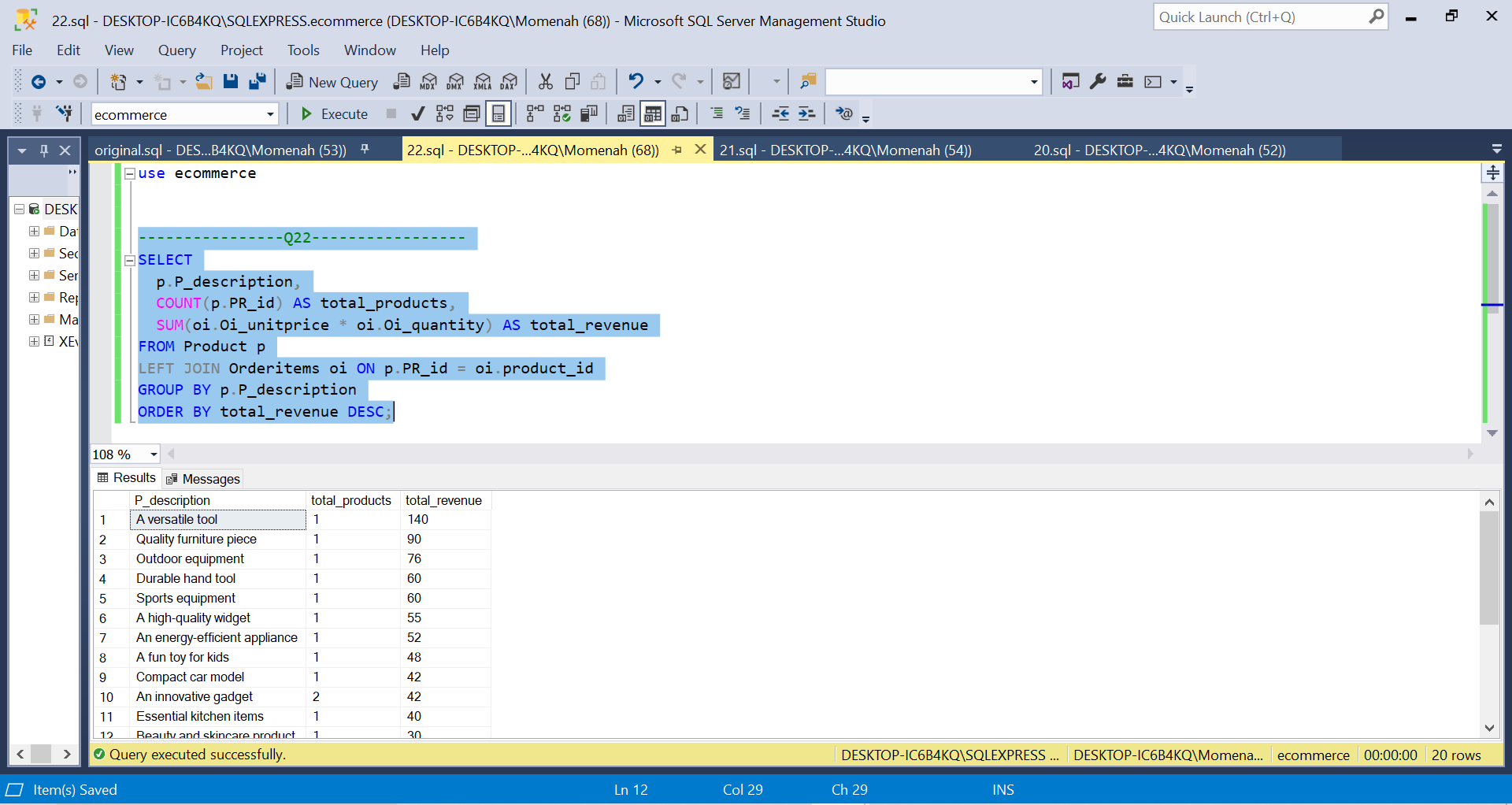
Explanation:

1. The Product and OrderItems tables are joined using the PR\_id column to determine the total sales for each product first. The sum of each product's sales is determined using the SUM function. The outcomes are kept in a transient table called ps.
2. The ProductReviews table is then joined to the ps table using the PR\_id column. This implies that the code will locate all of a product's reviews.
3. The results are then sorted by the total\_sales column in reverse chronological order. As a result, the code will display first the items with the highest cumulative sales.

Question\_21:



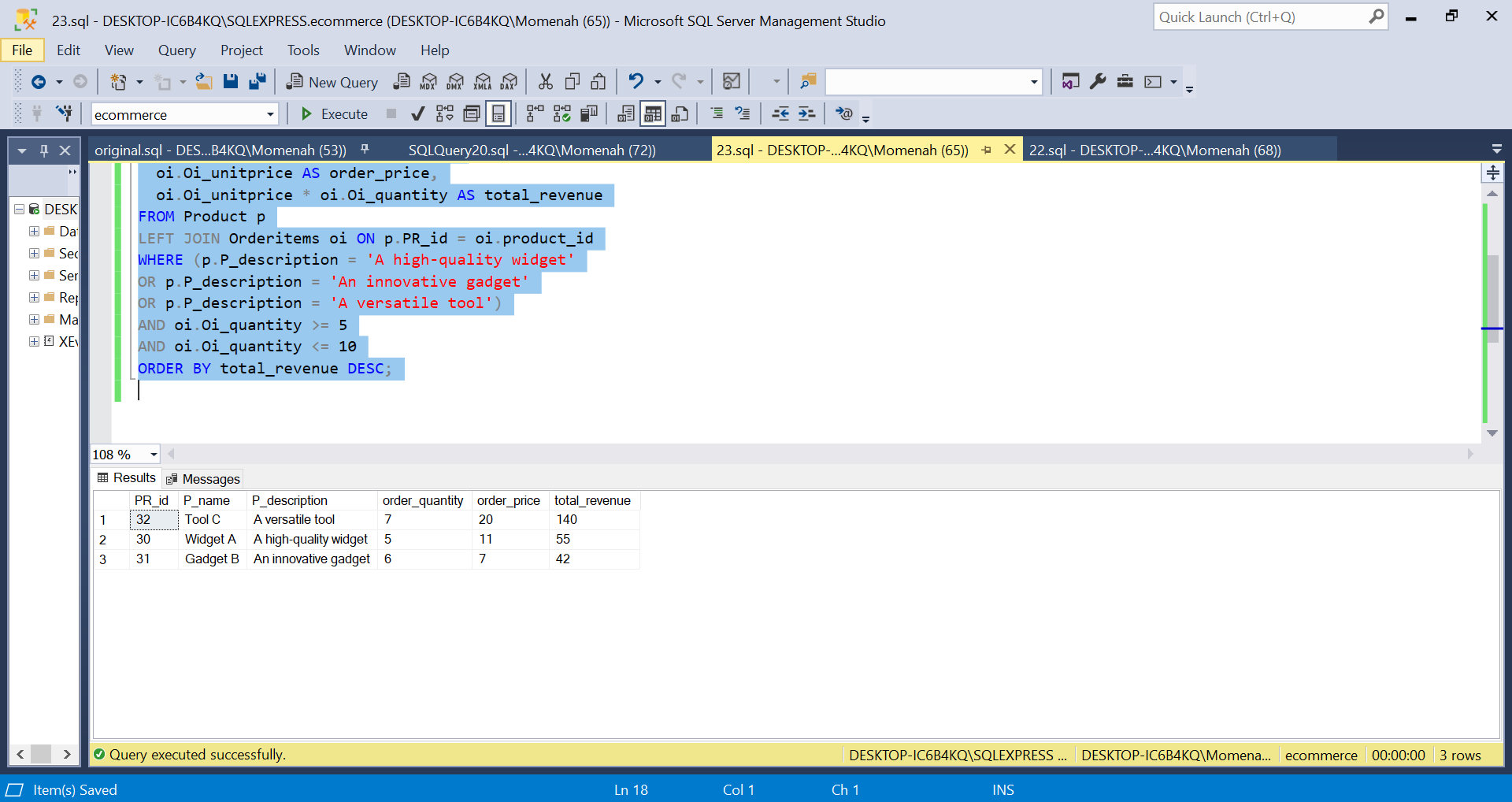
Question\_22:



Explanation:

1. The PR\_id column is used in this code to join the Product and OrderItems tables. In other words, even if a product doesn't have any order items, the code will still find all of them for that product.
2. The P\_description column is then used to group the results. This means that the code will determine the total revenue for each category and count the number of products in each category.
3. The results are then sorted in reverse order by the total\_revenue column. This means that the product categories with the highest overall revenue will be displayed first by the code.

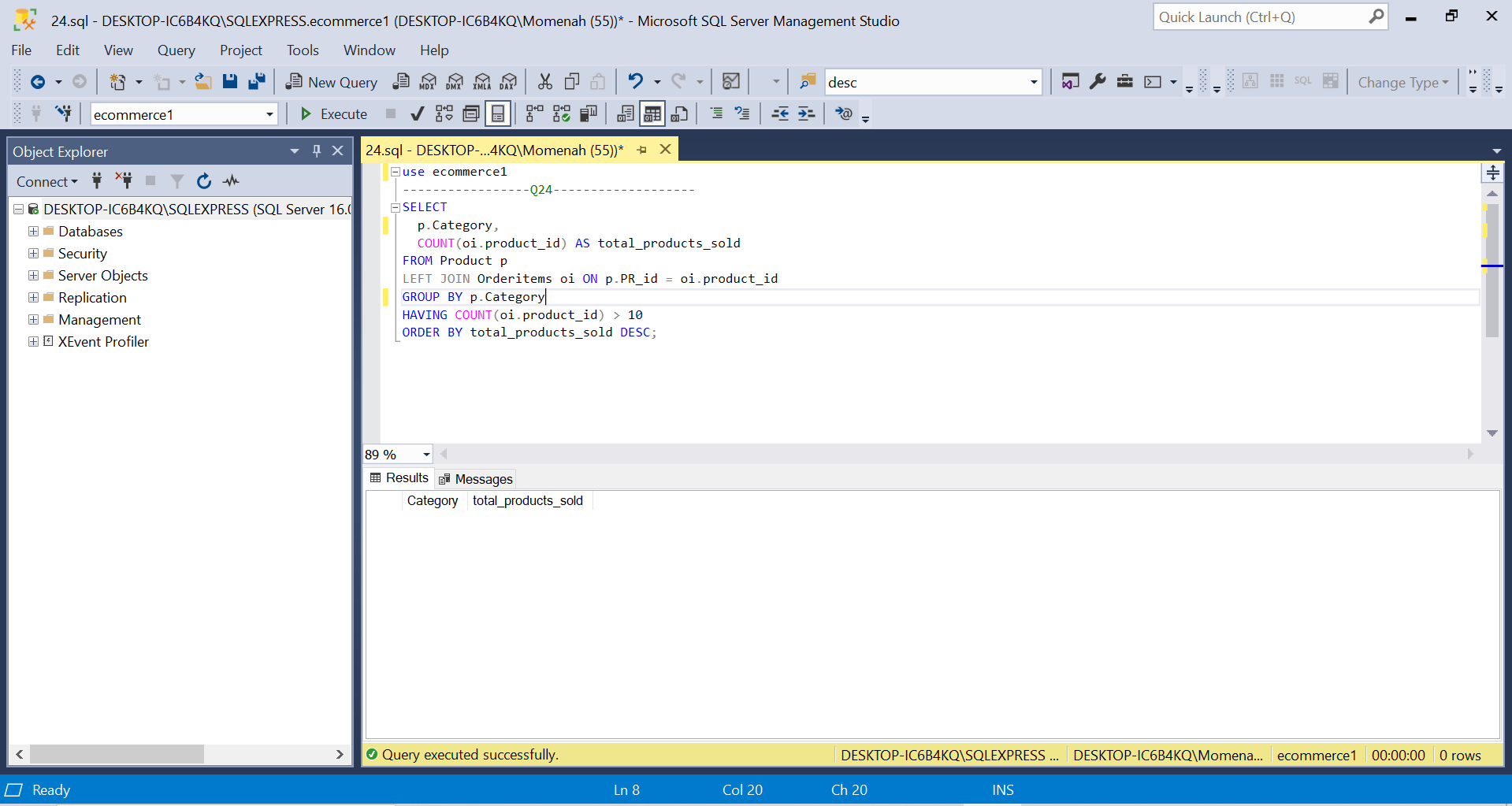
Question\_23:



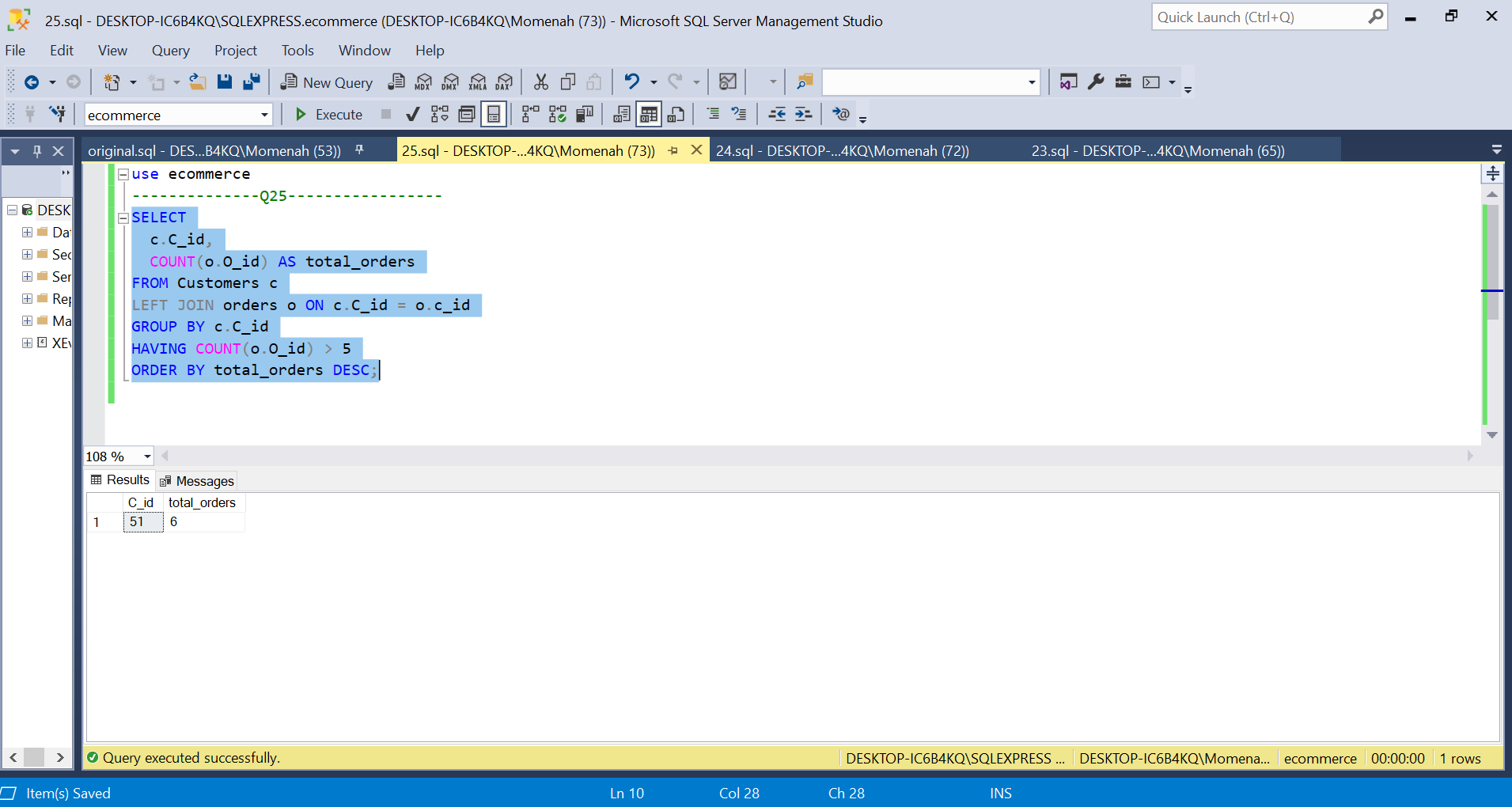
Explanation:

1. The PR\_id column is used in this code to join the Product and OrderItems tables. In other words, even if a product doesn't have any order items, the code will still find all of them for that product.
2. The code then limits the products that are displayed to those that are described as "A high-quality widget", "An innovative gadget", or "A versatile tool". Additionally, the code limits the results to only display order items with quantities between 5 and 10.
3. The results are then sorted in reverse order by the total\_revenue column. As a result, the code will display the items with the highest overall revenue first.

Question\_24:



Question\_25:



Explanation:

1. This code joins the C\_id column of the Customers (c) and Orders (o) tables. In other words, even if a customer doesn't have any orders, the code will still find all of them.
2. The C\_id column is then used to group the results in the code. This indicates that the code will tally how many orders each customer has placed.
3. The results are then filtered by the code to only display clients who have placed more than five orders. The filter is applied using the HAVING clause.
4. The results are then sorted by the total\_orders column in reverse chronological order. As a result, the code will display the customers who have placed the most orders first.

Link To GITHUB:

https://github.com/minam121/i211901\_i211909