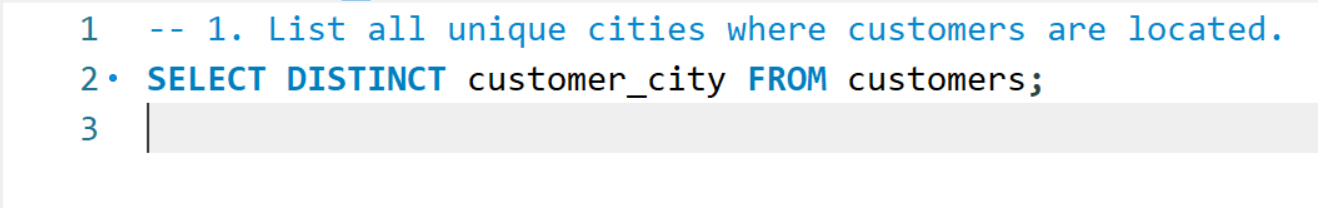
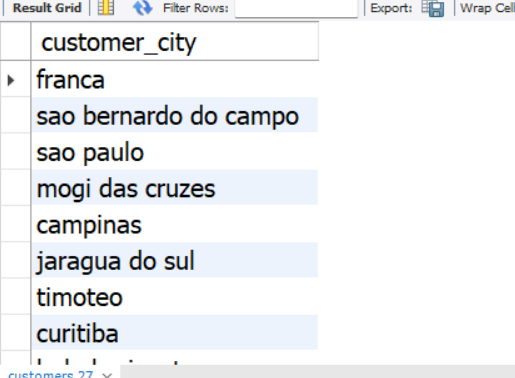
**1. List all unique cities where customers are located**

To get the list of all unique cities where customers are located, we use the DISTINCT function, which ensures that duplicate city names are removed. The SELECT statement fetches data from the customers table.

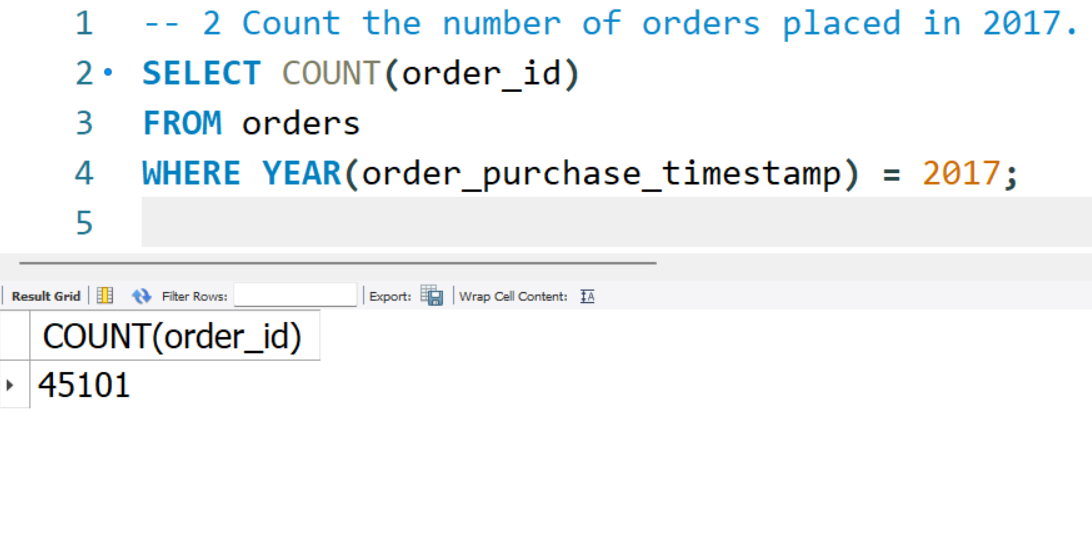




This helps in understanding the geographical distribution of customers, which is useful for targeted marketing and business expansion strategies.

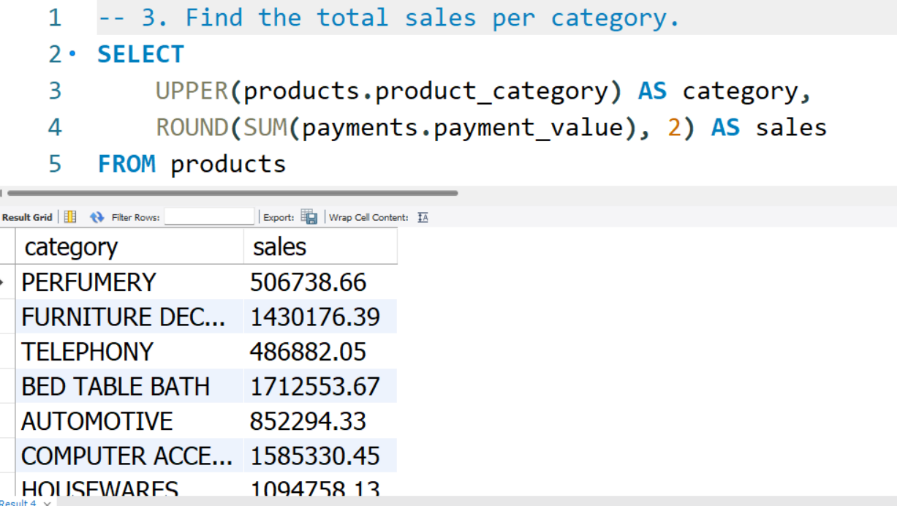
**2. Count the number of orders placed in 2017**

To count the total number of orders placed in 2017, we use the COUNT() function, which returns the total number of rows. The YEAR() function extracts the year from the order\_purchase\_timestamp, and the WHERE clause filters records for the year 2017.

This query helps businesses analyze yearly sales trends and evaluate performance for a specific year

**3. Find the total sales per category**

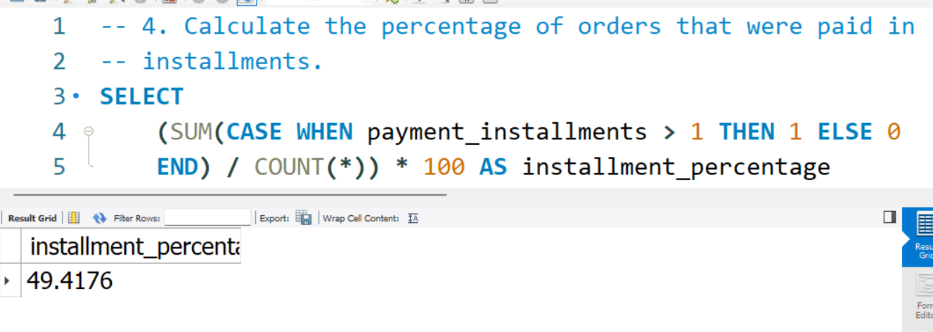
To calculate the total sales for each product category, we use the SUM() function to add up all payment values. The JOIN function links the products, order\_items, and payments tables. The GROUP BY function groups sales by category, and ROUND() is used to format the total sales value to two decimal places.



This query helps businesses analyze the performance of different product categories and make informed decisions regarding inventory and marketing strategies

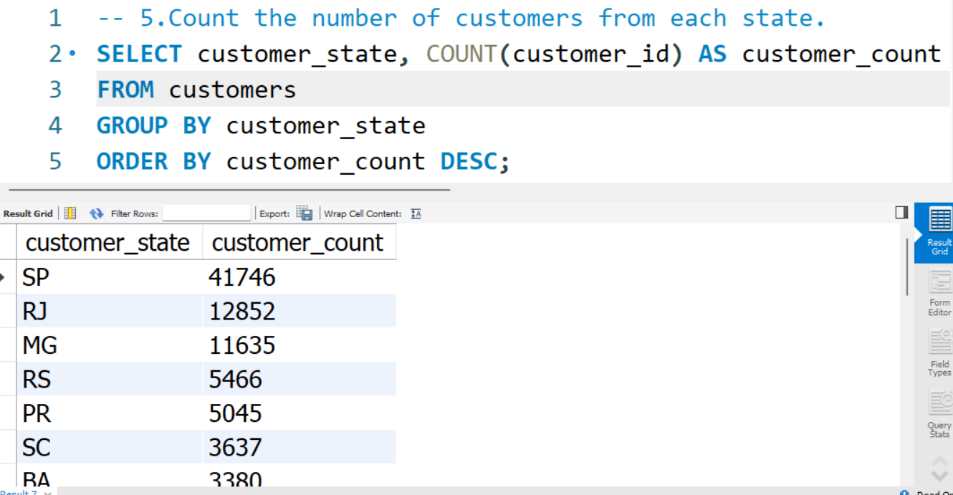
**4. Calculate the percentage of orders paid in installments**

To determine the percentage of orders paid in installments, we use the SUM() function to count orders where payment\_installments > 1. The CASE statement is used to classify installment orders, and COUNT(\*) is used to find the total number of orders. The formula calculates the percentage of installment orders

  
This analysis helps businesses understand customer payment preferences and optimize their installment plans accordingly.

**5. Count the number of customers from each state**

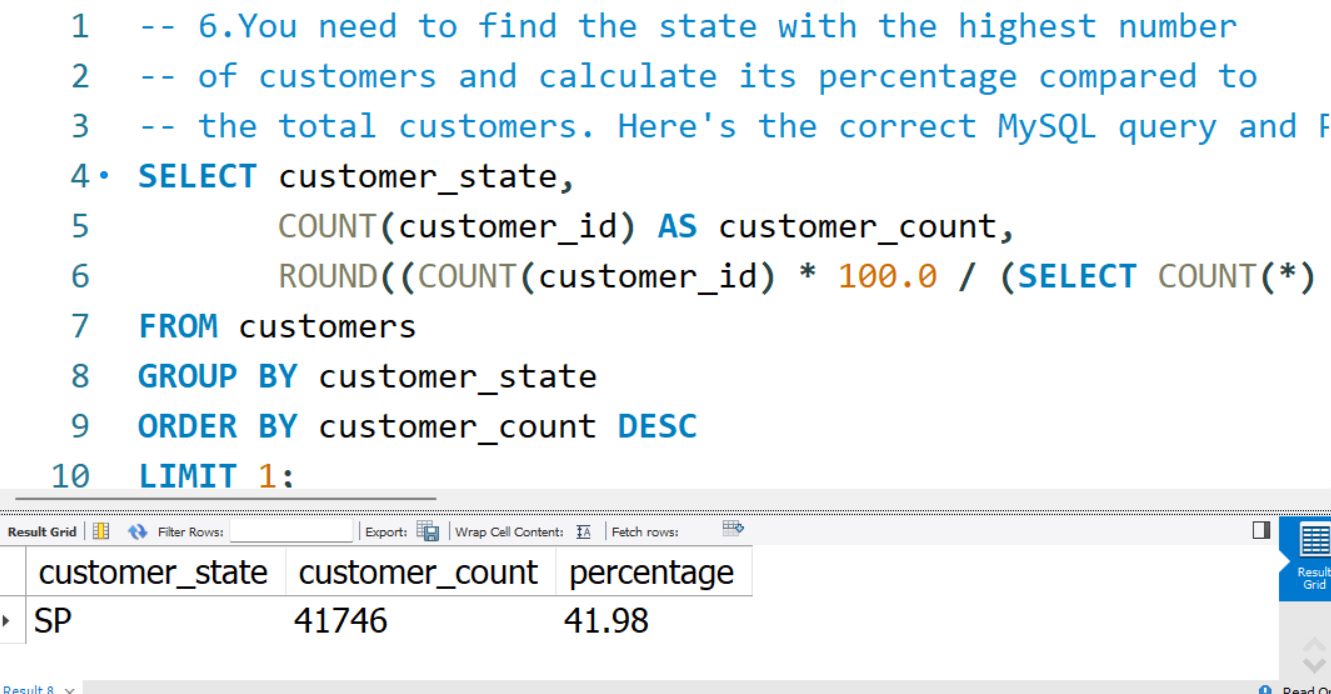
To determine the number of customers in each state, we use the COUNT() function to count customer IDs. The GROUP BY function groups data by state, and ORDER BY DESC ensures that the states with the highest customer count appear first.



This query is useful for identifying regions with the highest customer density, which can help in regional marketing and business expansion.

**6. Get the highest customer state and its percentage**

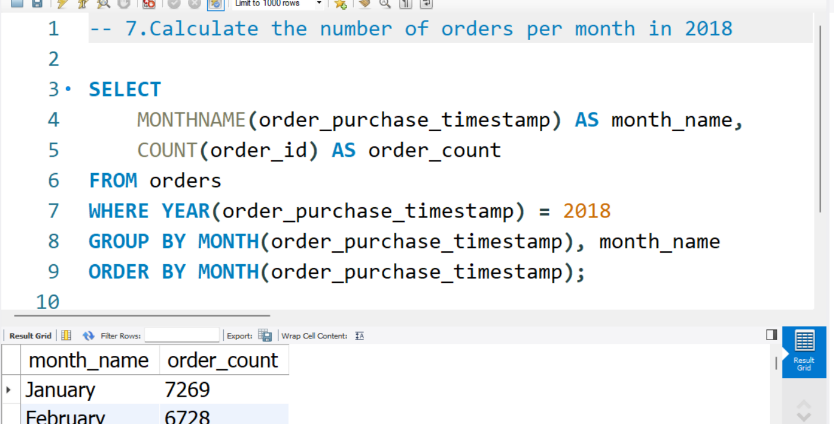
To find the state with the highest number of customers and its percentage contribution, we use COUNT() to count the number of customers per state. The total number of customers is retrieved using a subquery. ROUND() ensures the percentage is displayed with two decimal places. The LIMIT 1 function ensures that only the top state is displayed.



This query helps in identifying the most dominant customer region, which is useful for targeted marketing strategies.

**7. Calculate the number of orders per month in 2018**

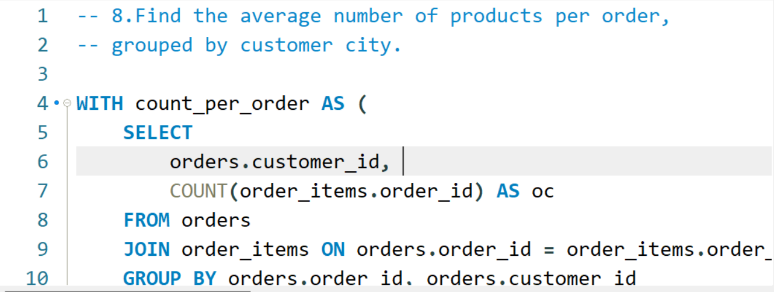
To find the number of orders placed each month in 2018, we use the MONTHNAME() function to extract the name of the month from the order\_purchase\_timestamp column. The COUNT(order\_id) function counts the total orders for each month, while the WHERE clause filters records for the year 2018. The GROUP BY function ensures that each row represents a month.

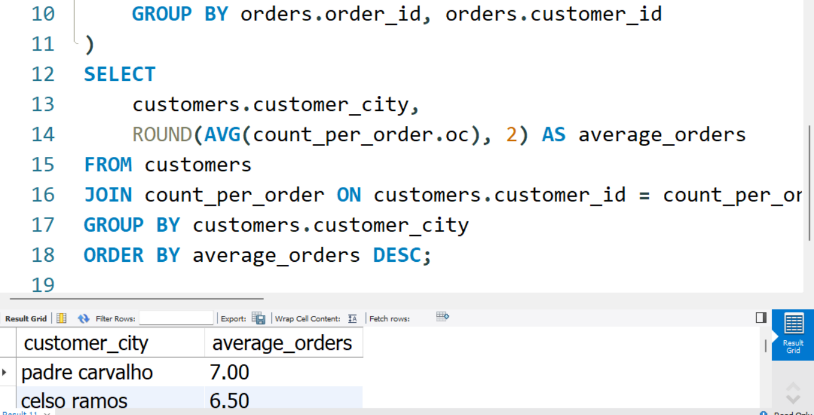


This query helps businesses analyze seasonal sales trends and adjust marketing campaigns accordingly.

**8. Find the average number of products per order, grouped by customer city**

To calculate the average number of products per order for each city, we use a Common Table Expression (CTE). The inner query (count\_per\_order) calculates the total products ordered per customer. The outer query uses AVG() to find the average number of products per order in each city.

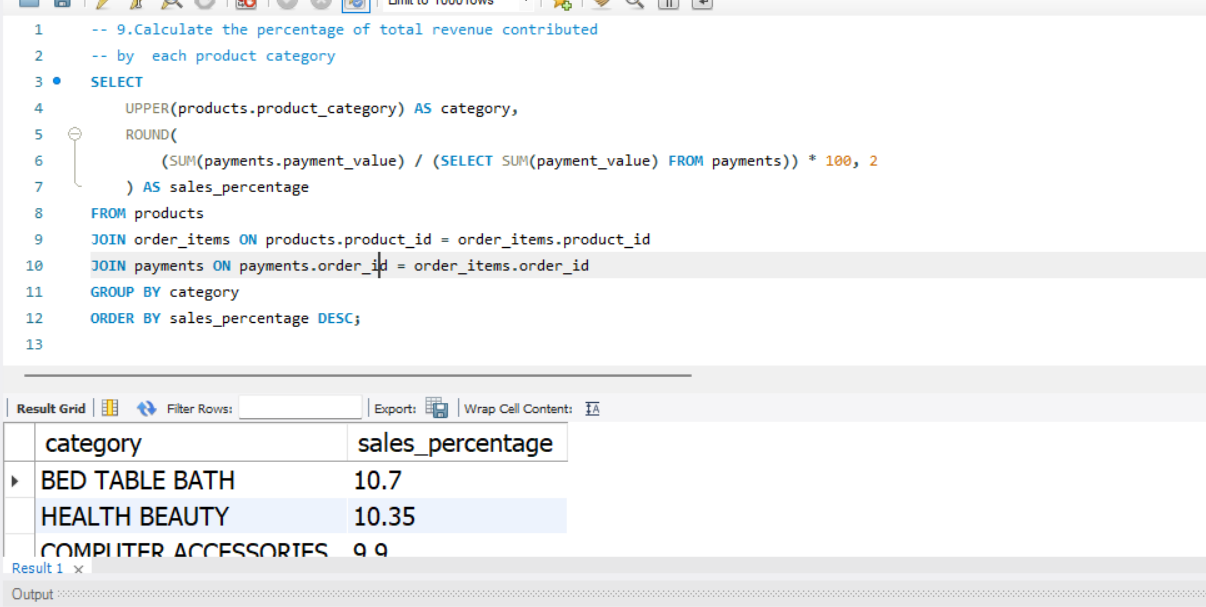




This query helps in understanding regional purchasing behavior, which is useful for inventory planning and demand forecasting

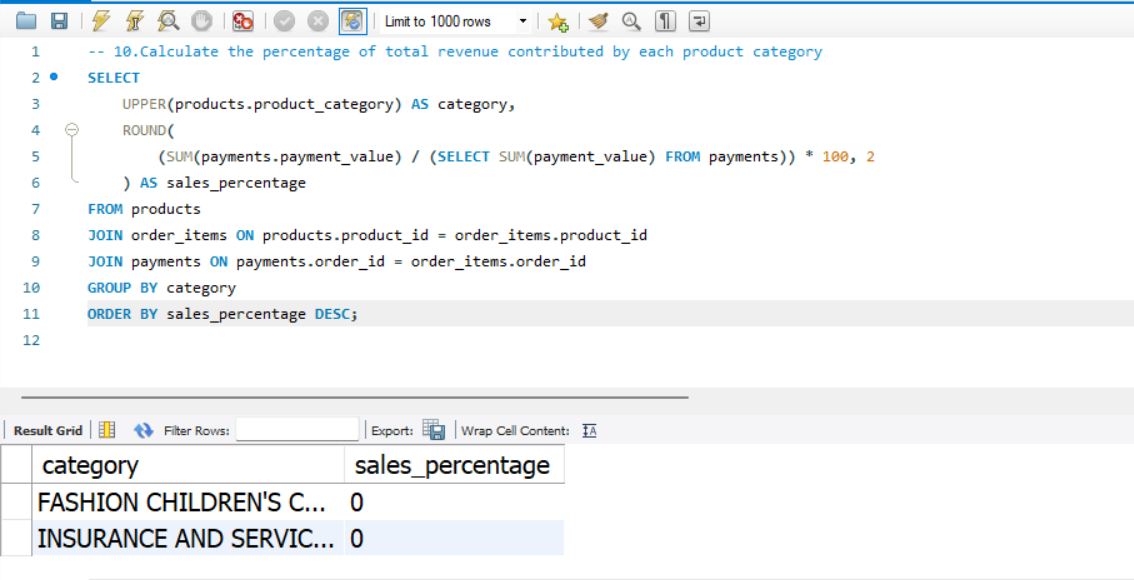
**9. Calculate the percentage of total revenue per product category**

To determine the percentage of total revenue contributed by each product category, we use SUM(payments.payment\_value) to calculate total sales per category. The total revenue is retrieved using a subquery. The percentage calculation is performed by dividing category revenue by total revenue and multiplying by 100.

  
This query helps businesses identify their most profitable product categories and optimize their inventory accordingly

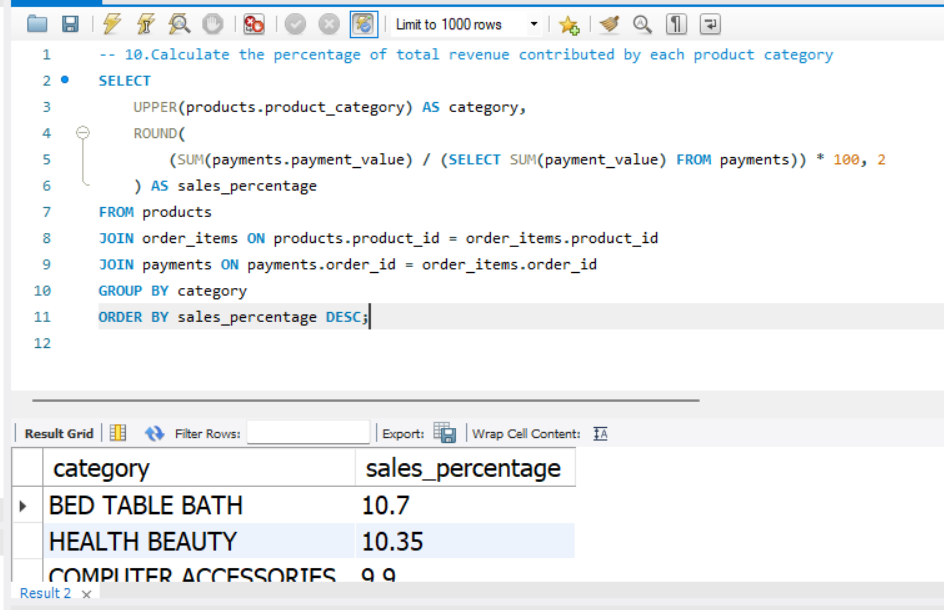
**10. Identify the most frequently bought-together products**

To find which products are frequently purchased together, we use a self-join on the order\_items table. The ON condition ensures that the same order\_id is used while selecting different product\_id values. The COUNT(\*) function counts the number of times each product pair appears together.

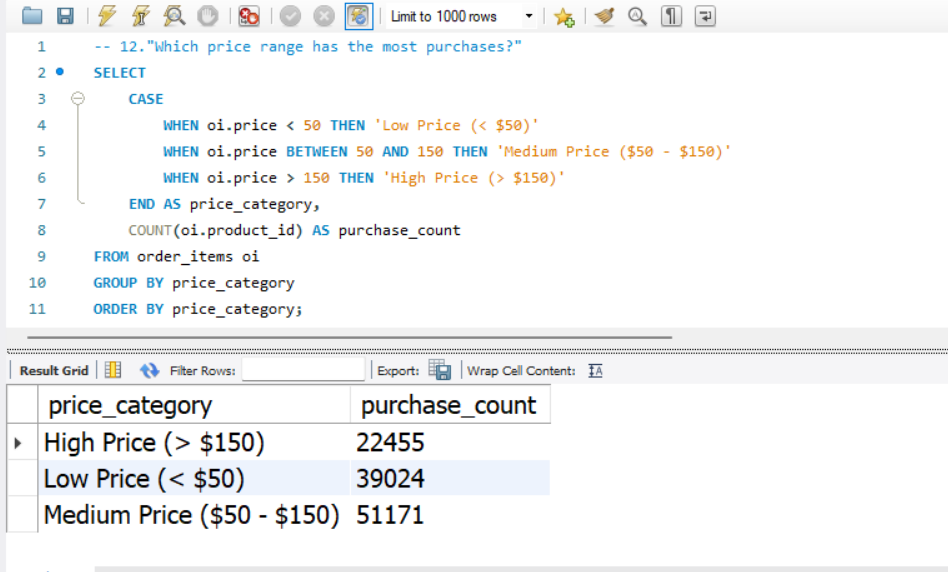
  
This query is useful for cross-selling strategies, as it helps businesses identify which products should be bundled together for promotions.

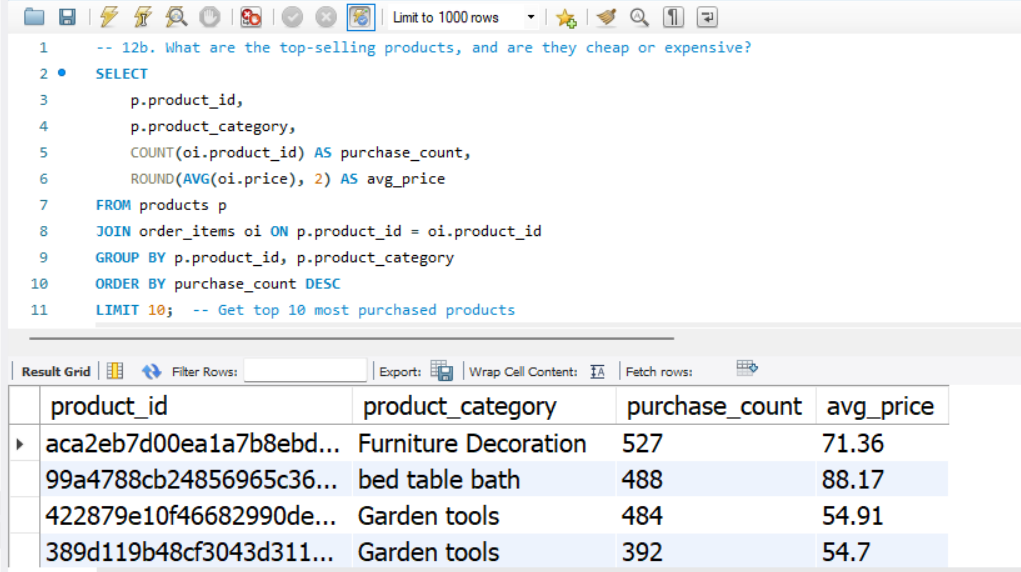
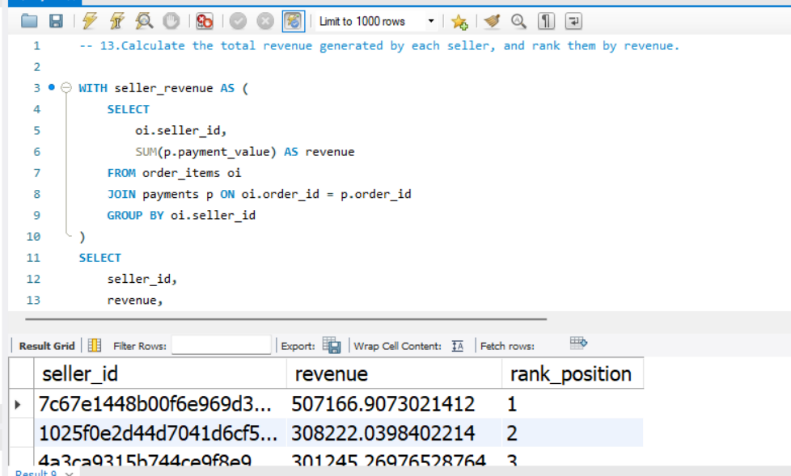
**11. Calculate the total revenue generated by each seller and rank them**

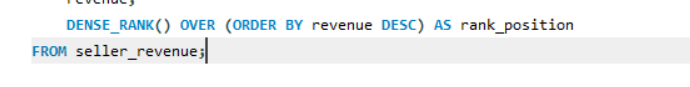
To find the total revenue generated by each seller, we use the SUM() function to calculate the total payments received by each seller. We use DENSE\_RANK() to rank sellers based on their revenue in descending order.



This query helps businesses identify top-performing sellers, which is useful for partnerships and reward programs

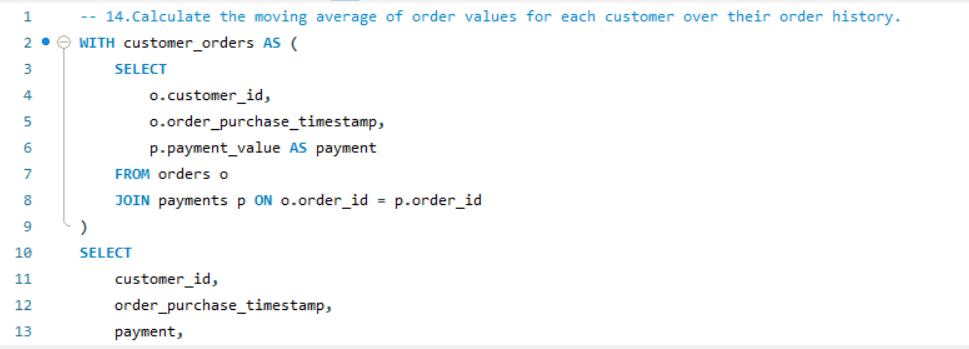


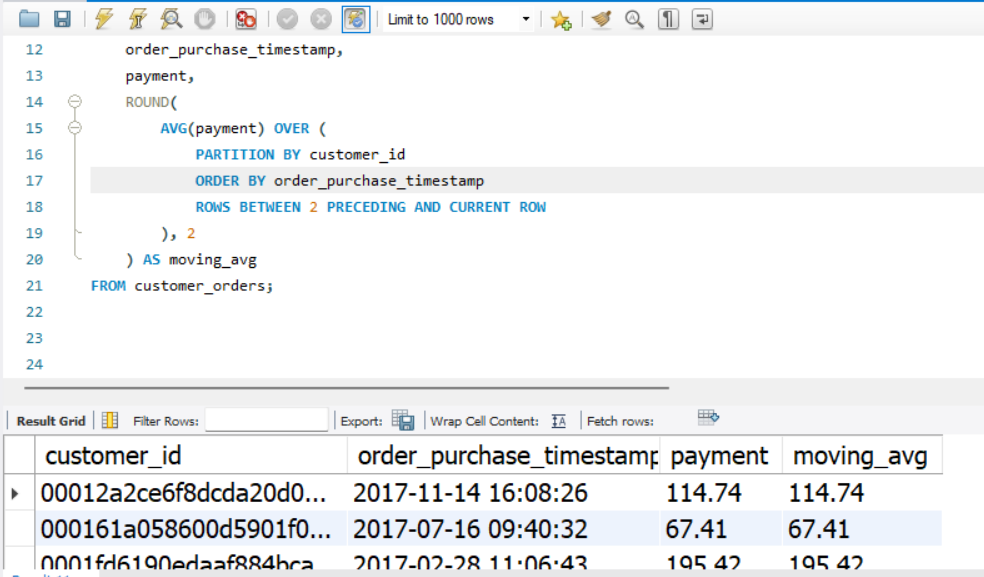
  
  




**12. Calculate the moving average of order values for each customer over their order history**

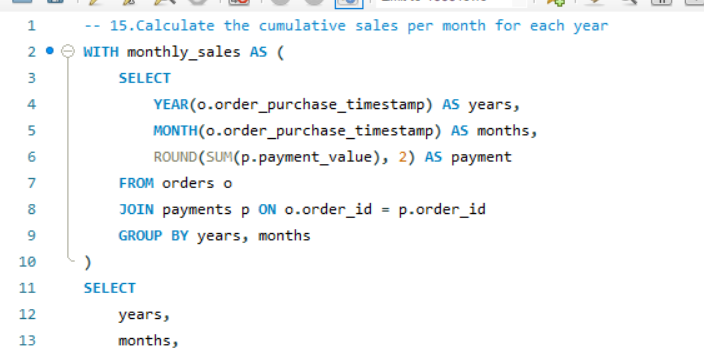
To calculate the moving average of order values for each customer, we use AVG() as a window function. The PARTITION BY customer\_id ensures that calculations happen separately for each customer. The ROWS BETWEEN 2 PRECEDING AND CURRENT ROW clause ensures the moving average includes only the last 3 orders

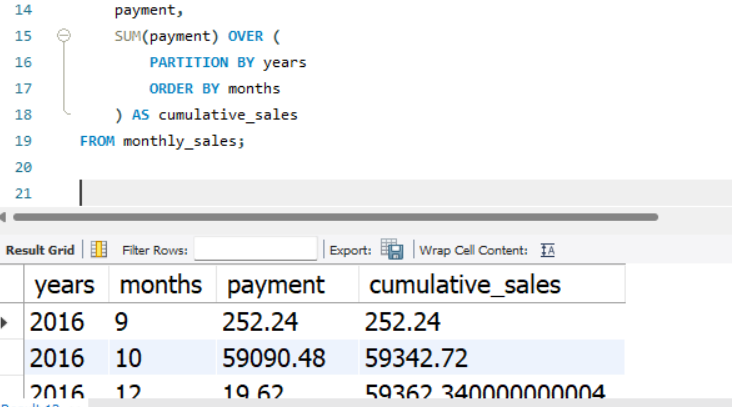


  
This query helps businesses track customer spending trends over time.

**13. Calculate the cumulative sales per month for each year**

To calculate cumulative monthly sales, we first sum up monthly revenue using SUM(). Then, the SUM() window function with PARTITION BY ensures that cumulative sales are computed per year.

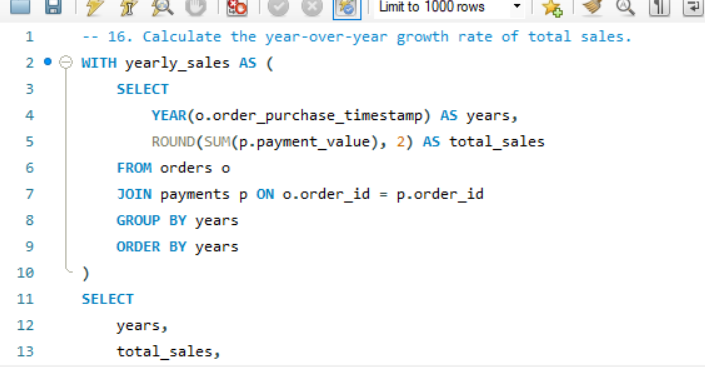


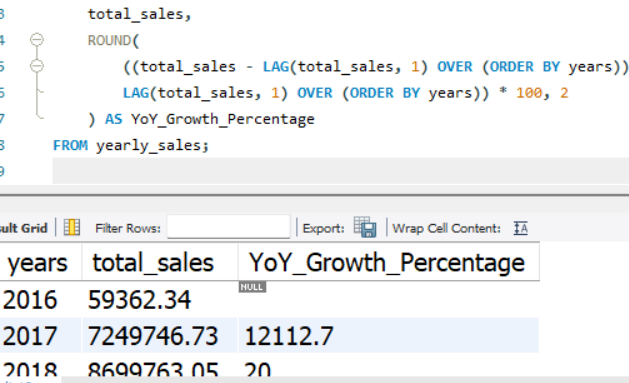


This query is useful for analyzing sales growth within each year.

**14. Calculate the year-over-year growth rate of total sales**

To calculate the percentage increase or decrease in sales compared to the previous year, we use the LAG() function to retrieve the previous year's sales and perform a percentage difference calculation

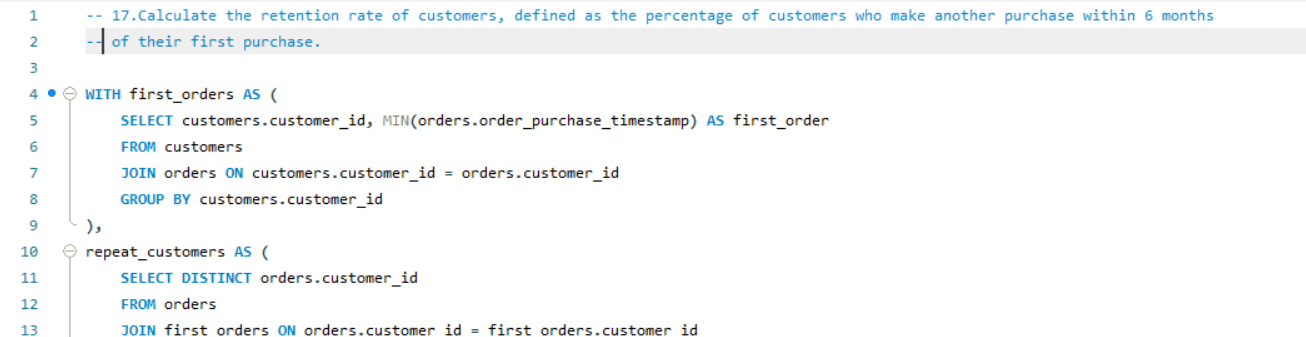


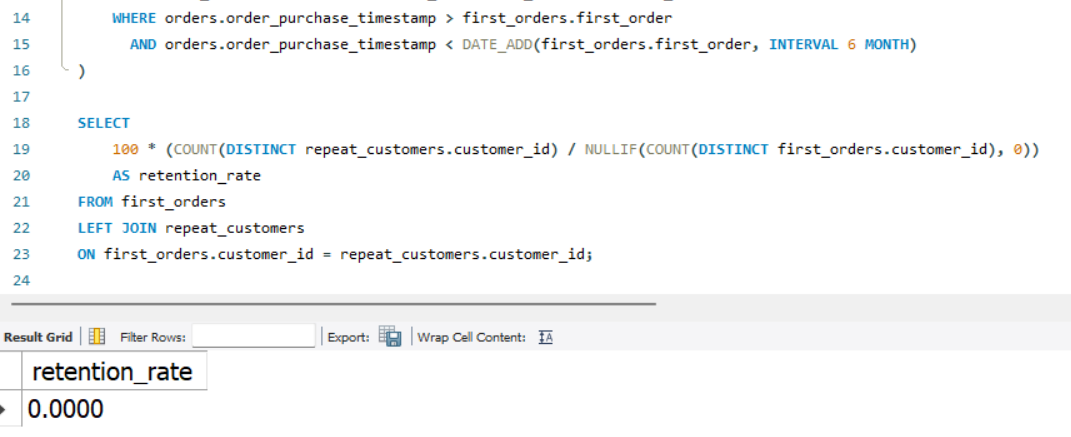


This query helps businesses measure yearly revenue growth.

**15. Calculate the retention rate of customers**

To find the percentage of customers who make a repeat purchase within 6 months, we first determine each customer’s first purchase date and check if they made another order within 6 months.

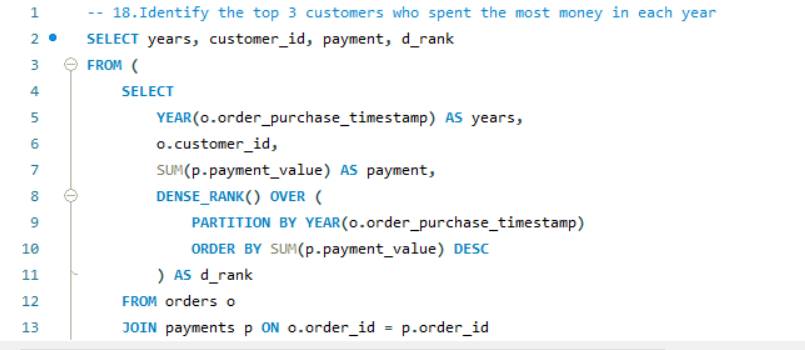


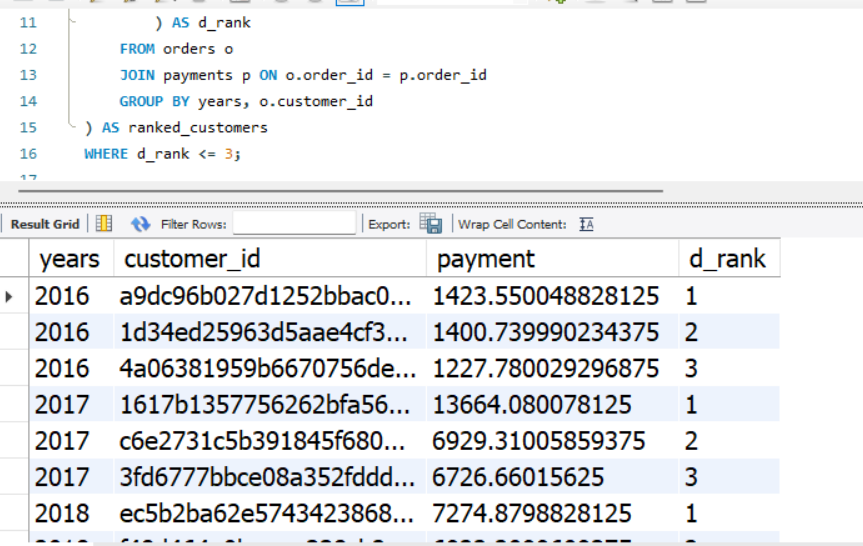


This query helps track customer loyalty and retention rates.

**16. Identify the top 3 customers who spent the most money in each year**

To find the highest spending customers per year, we use SUM() to calculate total spending and DENSE\_RANK() to rank customers within each year.

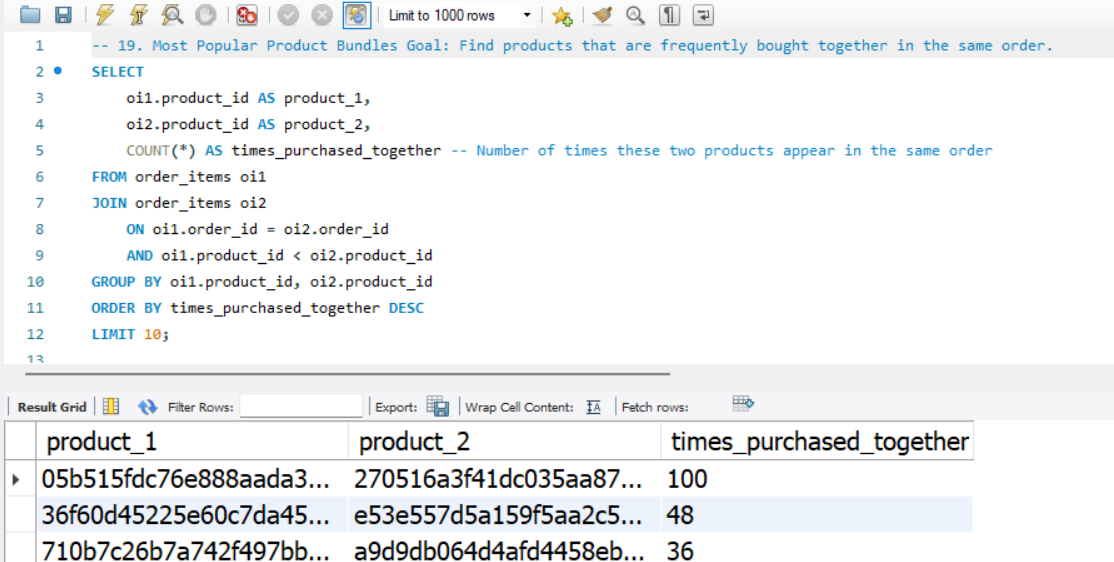




This query helps businesses identify high-value customers.

**17. Identify the most popular product bundles**

To find which products are most often bought together, we use a self-join on order\_items, ensuring products from the same order are paired.

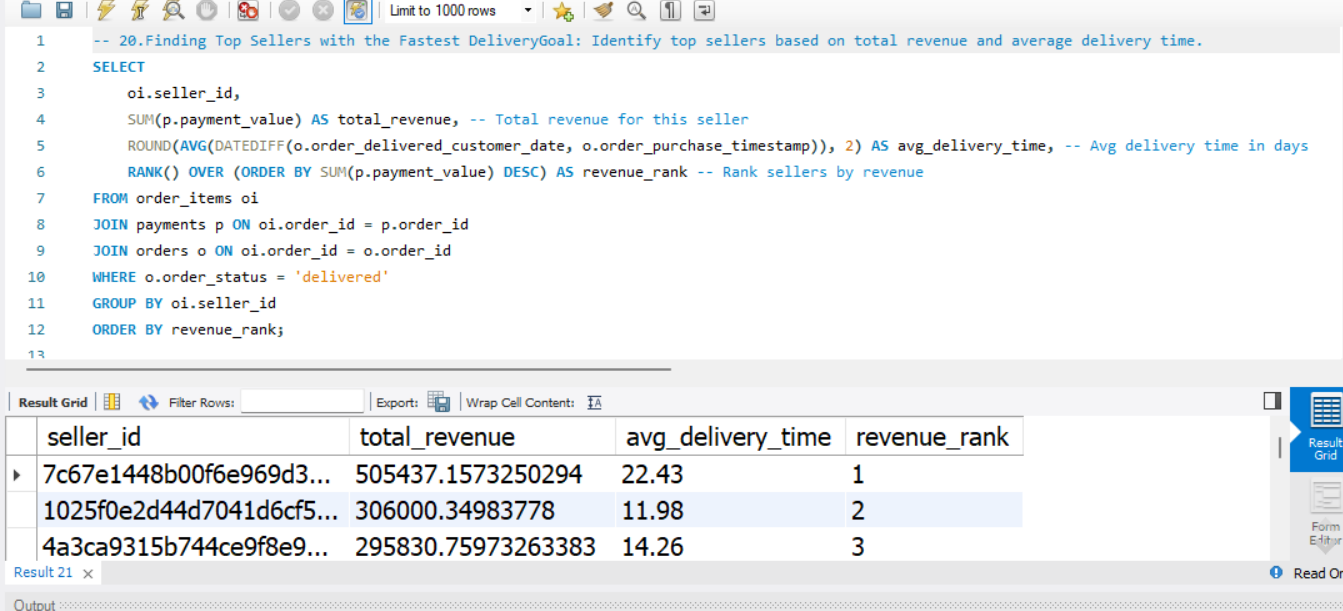


This query helps in cross-selling and bundle recommendations.

**18. Find the top-performing sellers based on revenue and delivery speed**

**18. Find the Top-Performing Sellers Based on Revenue and Delivery Speed**

To determine the best-performing sellers, we need to calculate their total revenue and average delivery time. We use the SUM() function to compute the total revenue each seller has earned, and the AVG() function along with DATEDIFF() to calculate the average time it takes for them to deliver orders. The RANK() function ranks sellers based on their total revenue in descending order.

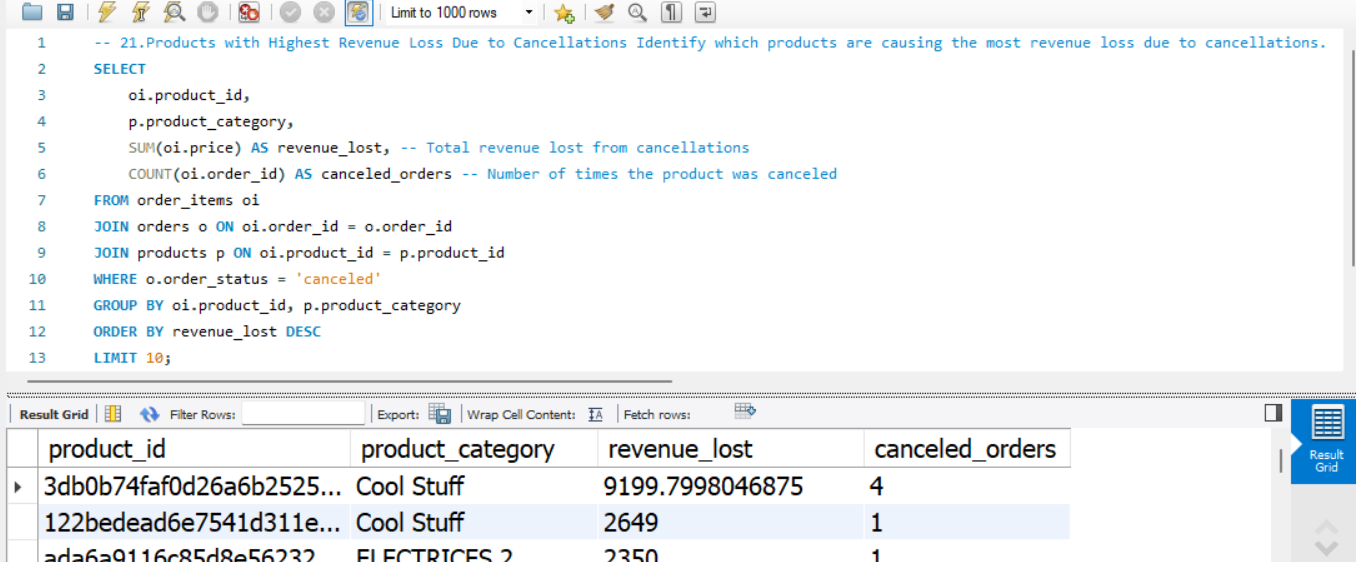


This query helps in recognizing reliable and high-earning sellers.

This query is useful for **e-commerce platforms and online marketplaces** to evaluate seller performance. It helps in identifying top sellers who generate the highest revenue while maintaining efficient delivery times. Businesses can use this data to provide **incentives, better visibility, or promotional benefits** to high-performing sellers. Additionally, sellers with slow delivery times but high revenue can be targeted for **logistics improvements** to enhance customer satisfaction

**19. Identify Products with the Highest Revenue Loss Due to Cancellations**

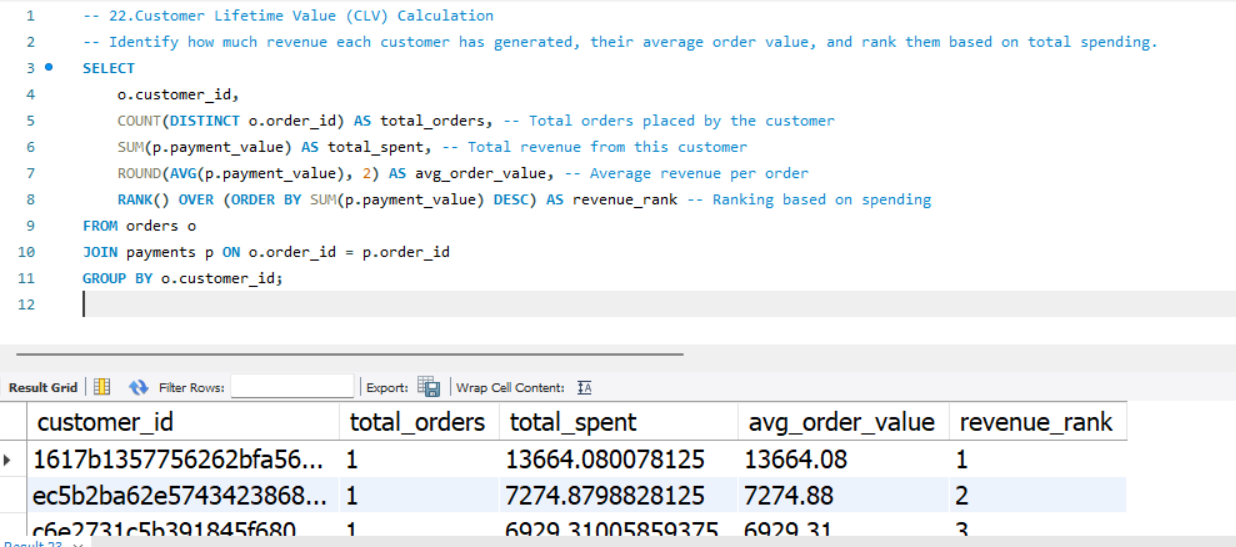
To analyze revenue loss from canceled orders, we sum up the price of all products that belong to canceled orders. We use the SUM() function to calculate the total lost revenue and COUNT() to determine how many times each product was canceled. The ORDER BY clause sorts products in descending order to highlight the most affected ones.



This query is beneficial for **inventory managers and business analysts** to identify products that frequently get canceled, leading to revenue loss. The insights gained from this query can help businesses determine whether cancellations are due to **poor product quality, pricing issues, or supply chain problems**. Companies can take corrective actions like **adjusting product descriptions, improving supplier quality, or offering better customer support** to reduce cancellations.

**20. Calculate Customer Lifetime Value (CLV)**

Customer Lifetime Value (CLV) is a crucial metric for businesses to estimate the total revenue a customer will generate over their lifetime. We use SUM() to calculate the total money spent by each customer and AVG() to determine their average spending per order. The RANK() function is used to rank customers based on total spending.



Businesses use CLV analysis for **customer segmentation, targeted marketing, and loyalty programs**. High CLV customers can be offered **exclusive discounts and personalized recommendations**, while businesses can work on retention strategies for low CLV customers to increase their spending.

**21. Calculate Monthly Revenue Growth Rate**

To measure how revenue grows or declines month-over-month, we use SUM() to calculate total revenue per month. The LAG() function retrieves revenue from the previous month, and a percentage growth formula is applied.

The query begins by calculating the total revenue for each month. This is done using a **Common Table Expression (CTE)** called monthly\_sales. The YEAR(order\_purchase\_timestamp) and MONTH(order\_purchase\_timestamp) functions extract the year and month from the order date, allowing revenue to be grouped accordingly. The SUM(payment\_value) function then calculates the total revenue for each month. The data is grouped using GROUP BY year, month, ensuring that each row in the result represents the total revenue for a particular month in a given year.

After calculating the monthly revenue, the main query retrieves data from monthly\_sales and calculates the month-over-month growth rate. The LAG(revenue) OVER (ORDER BY year, month) function retrieves the revenue from the **previous month** for comparison. By subtracting this previous month's revenue from the current month's revenue and dividing by the previous month's revenue, the query calculates the **percentage change in revenue**. The formula is then wrapped in the ROUND() function to format the percentage to two decimal places.

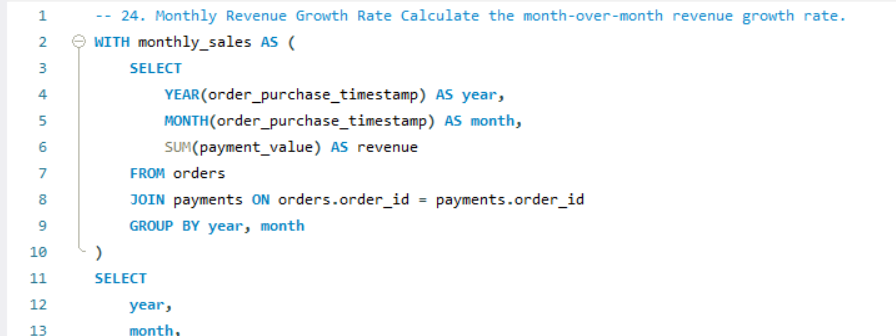
The ORDER BY year, month ensures that the growth rate is calculated sequentially for each month. The first month in the dataset will have a **NULL** growth rate since there is no previous month for comparison.

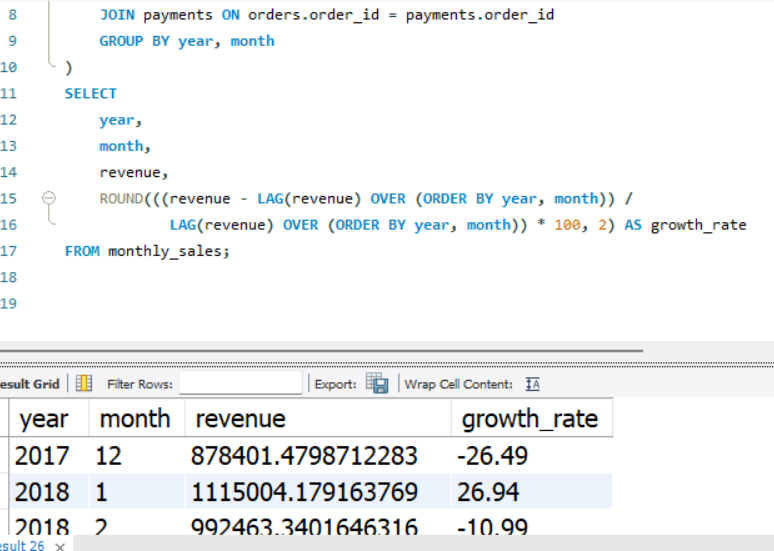
**Use Case & Business Impact:**

This query helps businesses track monthly revenue trends and identify growth patterns or warning signs. If revenue growth is positive, it confirms that sales strategies are working. If revenue declines, businesses can investigate potential issues such as seasonal slowdowns, product performance, or customer churn.

Businesses can use these insights for demand forecasting, pricing strategies, and marketing optimizations. If a business observes consistent revenue declines, it may need to introduce promotions, new product lines, or expand into new markets. Conversely, if certain months show rapid revenue growth, companies can analyze what factors contributed to the success and replicate those strategies in future months.

This query is especially valuable for finance teams, sales analysts, and executives who need to monitor financial performance regularly. By automating this analysis, companies can stay proactive in responding to revenue trends instead of reacting too late.





`This query helps businesses monitor **monthly sales trends, identify seasonal patterns, and detect periods of decline**. Companies can use this information to **adjust pricing, launch promotions, or optimize inventory levels** to maintain consistent growth