```sql
-- Regions with the highest canceled rate
select order_status,count(*)
From orders
group by 1;

SELECT
    C.customer_state,
    COUNT(CASE
        WHEN O.order_status = 'canceled' THEN 1
    END) * 100 / COUNT(*) AS canceled_rate
FROM
    customers C
        JOIN
    orders O ON C.customer_id = O.customer_id
Group by C.customer_state
Order by canceled_rate DESC


-- Write a query to calculate the total revenue per-category, sub-category and region
   SELECT
   products.product_category,
   customers.customer_state AS region,
   ROUND(SUM(payments.payment_value), 2) AS Total_Revenue
FROM
   order_items
       JOIN
   products ON order_items.product_id = products.product_id
       JOIN
   orders ON order_items.order_id = orders.order_id
       JOIN
   customers ON orders.customer_id = customers.customer_id
       JOIN
   payments ON orders.order_id = payments.order_id
GROUP BY 1 , 2;
```

```sql
-- 4. Identify the top 3 customers who spent the most money in each year.
With top_customers As(
Select *,
    Rank() over(partition by year order by year, sales DESC) as cust_rank
From(
Select year(orders.order_purchase_timestamp) as year,
    orders.customer_id,
   sum(payments.payment_value) as sales
From orders
join payments
on orders.order_id = payments.order_id
group by 1,2) As a
)
SELECT
    *
FROM
    top_customers
WHERE
    cust_rank <= 3;
```

```sql
-- 3. Calculate the year-over-year growth rate of total sales
With growth_rate As(
SELECT
    YEAR(orders.order_purchase_timestamp) AS year,
    ROUND(SUM(payments.payment_value), 2) AS sales
FROM
    orders
        JOIN
    payments ON orders.order_id = payments.order_id
GROUP BY 1
)
Select year, sales,
((sales - lag(sales,1) over(order by year)) / lag(sales,1) over(order by year)) * 100 as year_growth
From growth_rate;
```

```sql
-- 2. Calculate the cumulative sales per month for each month
Select year, Month, total_sales,
        sum(total_sales) over(order by year, month) as cumulative_sales
 From(
SELECT
    YEAR(orders.order_purchase_timestamp) AS year,
    MONTH(orders.order_purchase_timestamp) AS Month,
    ROUND(SUM(payments.payment_value), 2) AS total_sales
FROM
    orders
        JOIN
    payments ON orders.order_id = payments.order_id
GROUP BY 1 , 2
ORDER BY 1 , 2) As a;



    # 1. Find the average number of products per order, grouped by customer city.
        With count_order as(
       SELECT
    orders.order_id,
    orders.customer_id,
    COUNT(orders.order_id) AS ordered_count
FROM
    orders
        JOIN
    order_items ON orders.order_id = order_items.order_id
GROUP BY 1 , 2
        )
   SELECT
    customers.customer_city,
    AVG(count_order.ordered_count) AS avg_order
FROM
    count_order
        JOIN
    customers ON count_order.customer_id = customers.customer_id
GROUP BY customers.customer_city
ORDER BY avg_order DESC;
```

```sql
-- 1. Calculate the moving average of order values for each customer over their order history.
Select customer_id,
    order_purchase_timestamp, payment,
    avg(payment) over (partition by customer_id order by order_purchase_timestamp rows between 2 preced
From(
SELECT
orders.customer_id,
orders.order_purchase_timestamp,
payments.payment_value AS payment
FROM
orders
    JOIN
payments ON orders.order_id = payments.order_id) as A;


-- Analyze the seasonality of sales to identify peak month
SELECT
    YEAR(O.order_purchase_timestamp) AS Year,
    MONTH(O.order_purchase_timestamp) AS Month,
    ROUND(SUM(P.payment_value), 2) AS total_sales
FROM
    orders O
        JOIN
    payments P ON O.order_id = P.order_id
Where O.order_status = 'delivered'
GROUP BY 1 , 2
ORDER BY 1 , 2;


-- Identify the top 5 best-selling products on both revenue and quantity sold
SELECT
    products.product_id,
    products.product_category,
    ROUND(SUM(order_items.price * order_items.order_item_id),
            2) AS total_revenue
FROM
    order_items
        JOIN
    products ON order_items.product_id = products.product_id
GROUP BY 1 , 2
ORDER BY 3 DESC
LIMIT 5;
```

```sql
-- 4. Calculate the total revenue generated by each seller, and rank them by revenue.

    With top_seller As(
    SELECT
    order_items.seller_id,
    ROUND(SUM(payments.payment_value), 2) total_revenue
FROM
    order_items
        JOIN
    payments ON order_items.order_id = payments.order_id
GROUP BY order_items.seller_id
    )
    Select *,
        Rank()over(order by total_revenue DESC) as top_seller_rank
        From top_seller;


   # 2. Calculate the percentage of total revenue contributed by each product category.
           SELECT
    products.product_category,
    ROUND((SUM(payments.payment_value) / (SELECT
                    SUM(payment_value)
                FROM
                    payments)) * 100,
            2) AS revenue_percentage
FROM
    products
        JOIN
    order_items ON products.product_id = order_items.product_id
        JOIN
    payments ON order_items.order_id = payments.order_id
GROUP BY products.product_category
ORDER BY revenue_percentage DESC;
```