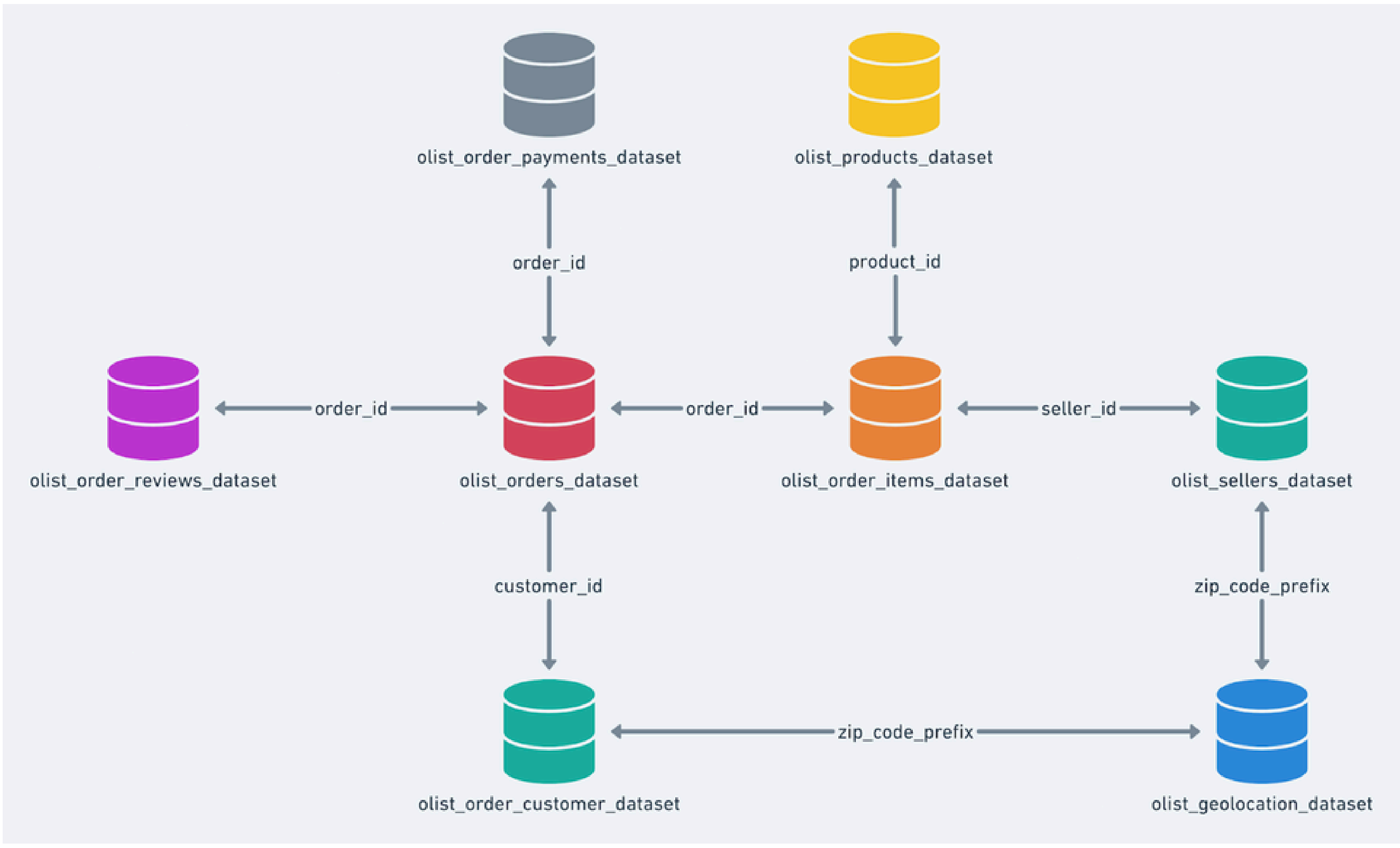
About Dataset

Target is a globally recognized brand and a leading retailer in the United States, known for offering exceptional value, inspiration, innovation, and a unique shopping experience.

This dataset focuses on Target's operations in Brazil, covering 100,000 orders placed between 2016 and 2018. It includes detailed information on order status, pricing, payment and shipping performance, customer locations, product attributes, and customer reviews.

Dataset Schema

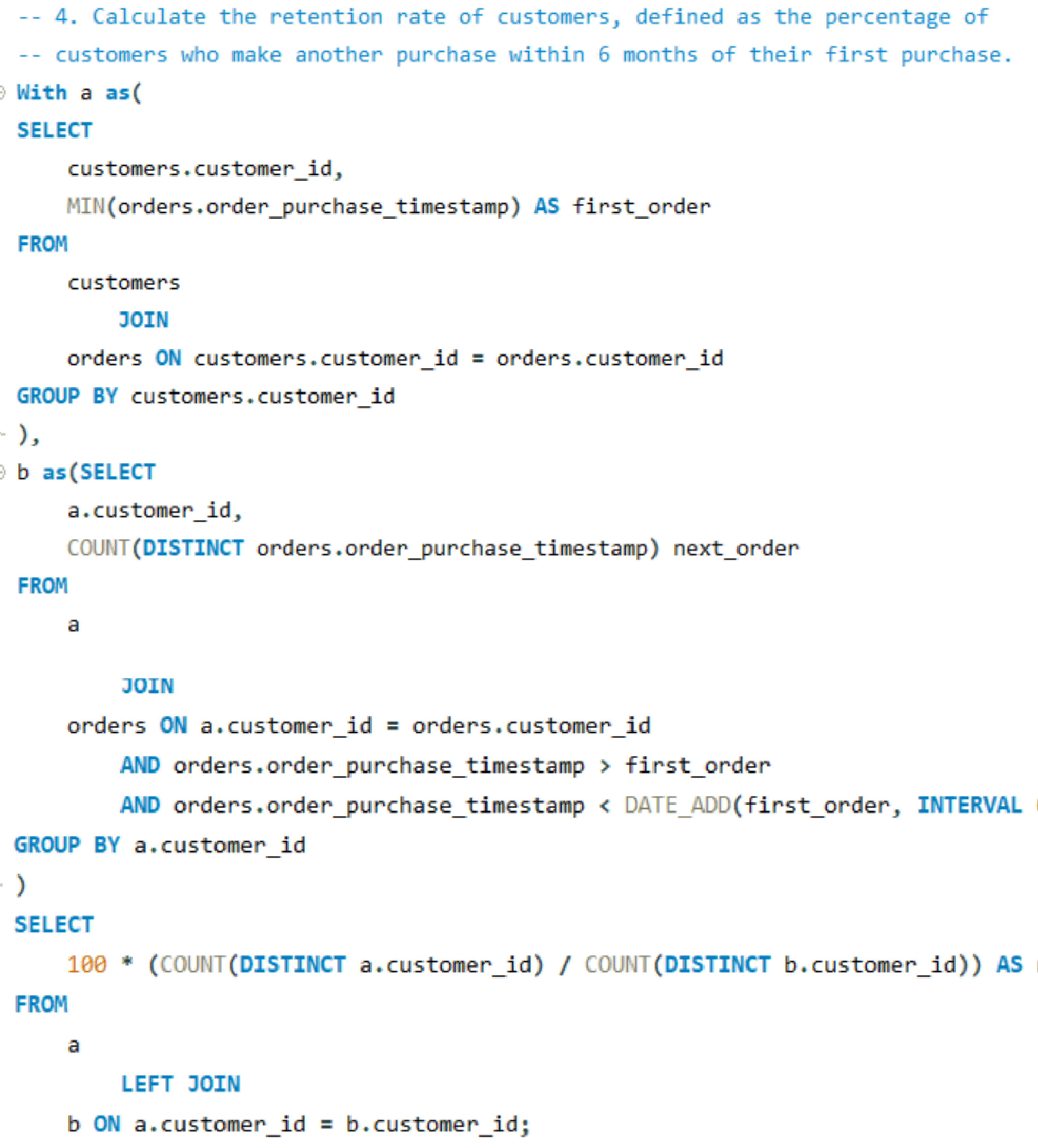


```sql
-- Advanced Queries

-- 1. Calculate the moving average of order values for each customer over their order histor

select customer_id, order_purchase_timestamp, payment,
avg(payment) over(partition by customer_id order by order_purchase_timestamp
rows between 2 preceding and current row) as moving_avg
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;
```

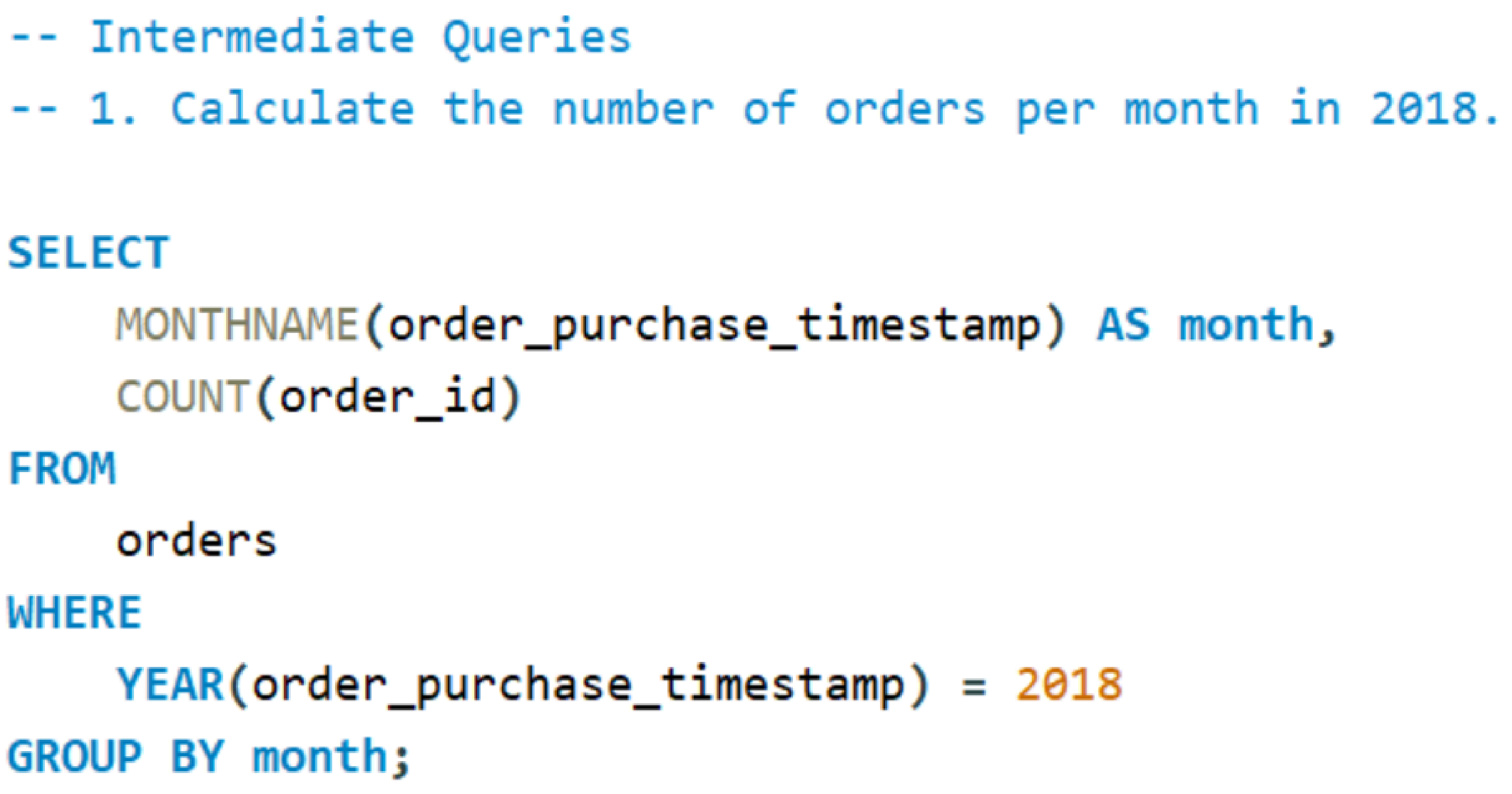
```sql
-- 2. Calculate the cumulative sales per month for each year.

select year, month, value,
sum(value) 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 value
FROM
    orders
join payments
on orders.order_id = payments.order_id
group by year, month
order by year, month) as a;
```

```sql
-- 3. Calculate the year-over-year growth rate of total sales.

WITH sales_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 year
)
select year, sales, ((sales - lag(sales,1)
over (order by year)) / lag(sales,1) over(order by year))*100 as year_growth_rate
from sales_rate;
```

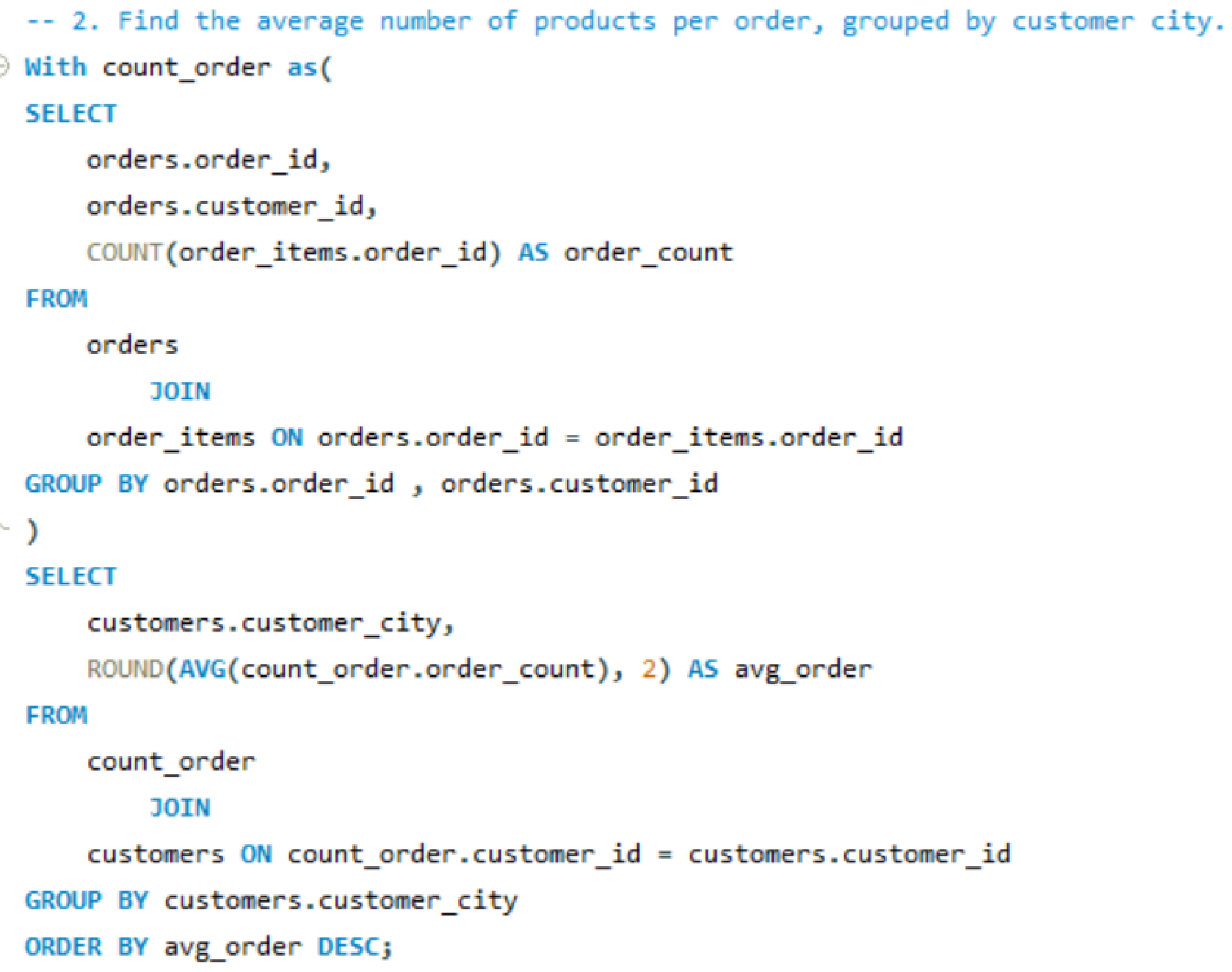
```sql
-- 5. Identify the top 3 customers who spent the most money in each year.
With top_customers as(
select  *,
rank() over(partition by year order by year, indi_purchase DESC) as cust_rank
from(
SELECT
    YEAR(orders.order_purchase_timestamp) year,
    orders.customer_id,
    SUM(payments.payment_value) AS indi_purchase
FROM
    orders
join payments
on orders.order_id = payments.order_id
group by year, orders.customer_id) as a
)
SELECT
    *
FROM
    top_customers
WHERE
    cust_rank <= 3;
```



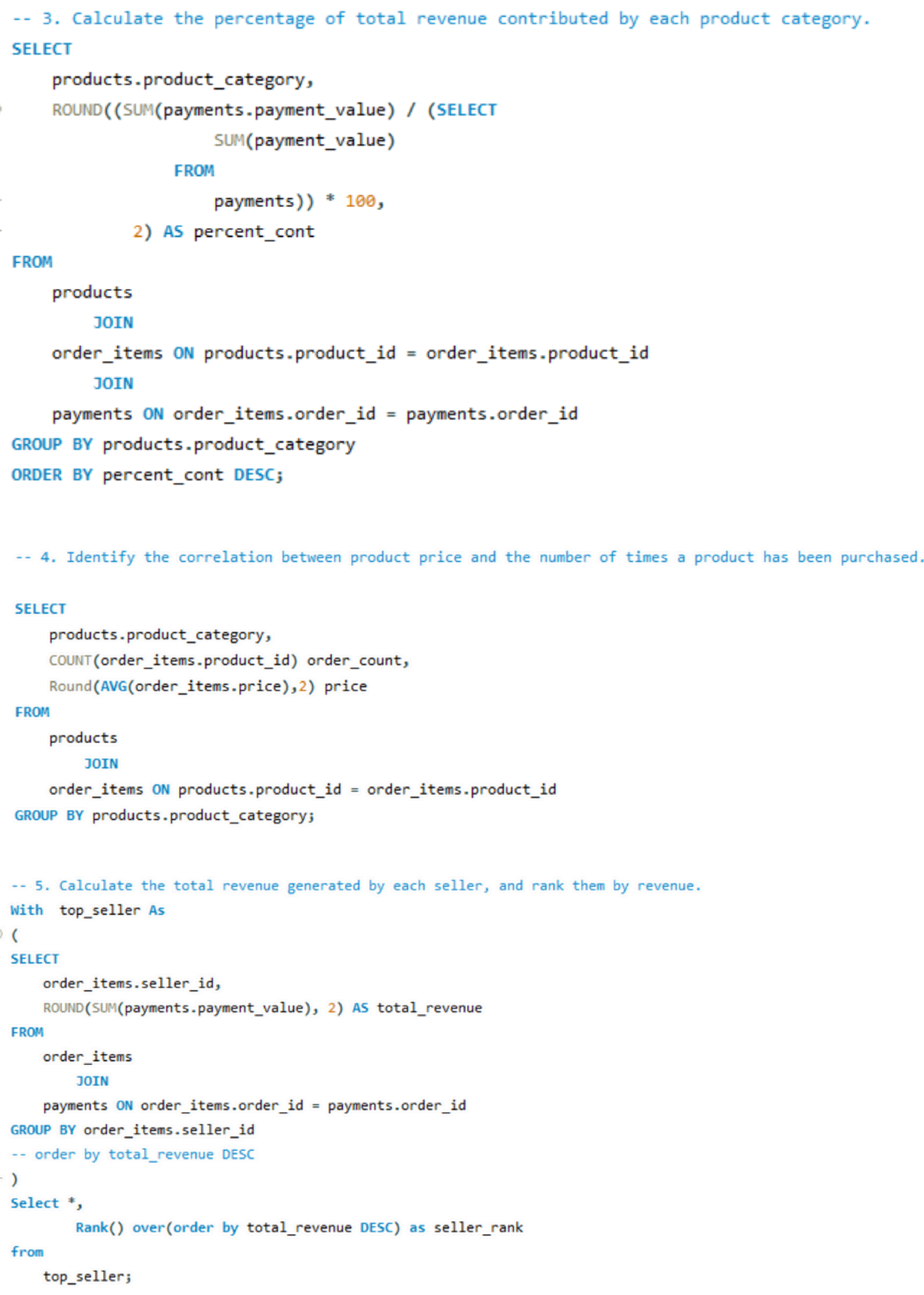
```
-- 4. Calculate the retention rate of customers, defined as the percentage of
-- customers who make another purchase within 6 months of their first purchase.
With a as(
SELECT
    customers.customer_id,
    MIN(orders.order_purchase_timestamp) AS first_order
FROM
    customers
        JOIN
    orders ON customers.customer_id = orders.customer_id
GROUP BY customers.customer_id
),
b as(SELECT
    a.customer_id,
    COUNT(DISTINCT orders.order_purchase_timestamp) next_order
FROM
    a

        JOIN
    orders ON a.customer_id = orders.customer_id
        AND orders.order_purchase_timestamp > first_order
        AND orders.order_purchase_timestamp < DATE_ADD(first_order, INTERVAL
GROUP BY a.customer_id
)
SELECT
    100 * (COUNT(DISTINCT a.customer_id) / COUNT(DISTINCT b.customer_id)) AS
FROM
    a
        LEFT JOIN
    b ON a.customer_id = b.customer_id;
```



```sql
-- Intermediate Queries
-- 1. Calculate the number of orders per month in 2018.

SELECT
    MONTHNAME(order_purchase_timestamp) AS month,
    COUNT(order_id)
FROM
    orders
WHERE
    YEAR(order_purchase_timestamp) = 2018
GROUP BY month;
```



```sql
-- 2. Find the average number of products per order, grouped by customer city.
With count_order as(
SELECT
    orders.order_id,
    orders.customer_id,
    COUNT(order_items.order_id) AS order_count
FROM
    orders
        JOIN
    order_items ON orders.order_id = order_items.order_id
GROUP BY orders.order_id , orders.customer_id
)
SELECT
    customers.customer_city,
    ROUND(AVG(count_order.order_count), 2) 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
-- 3. 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 percent_cont
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 percent_cont DESC;


-- 4. Identify the correlation between product price and the number of times a product has been purchased.

SELECT
    products.product_category,
    COUNT(order_items.product_id) order_count,
    Round(AVG(order_items.price),2) price
FROM
    products
        JOIN
    order_items ON products.product_id = order_items.product_id
GROUP BY products.product_category;


-- 5. 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) AS total_revenue
FROM
    order_items
        JOIN
    payments ON order_items.order_id = payments.order_id
GROUP BY order_items.seller_id
-- order by total_revenue DESC
)
Select *,
        Rank() over(order by total_revenue DESC) as seller_rank
from
    top_seller;
```

```sql
-- 1. List all unique cities where customers are located.
SELECT DISTINCT
    customer_city
FROM
    customers;

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

SELECT
    COUNT(order_id) AS orders_places
FROM
    orders
WHERE
    year(order_purchase_timestamp) = 2017;



 -- 3. Find the total sales per category.

 SELECT
     P.product_category,
     ROUND(SUM(Q.payment_value), 2) AS total_sales
 FROM
     products P
         JOIN
     order_items O ON P.product_id = O.product_id
         JOIN
     payments Q ON Q.order_id = O.order_id
 GROUP BY P.product_category
 order by total_sales DESC;
```

```sql
-- 4. Calculate the percentage of orders that were paid in installments.

SELECT
    ((SUM(CASE
        WHEN payment_installments >= 1 THEN 1
        ELSE 0
    END)) / COUNT(*)) * 100 AS order_percentage
FROM
    payments;


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

SELECT
    customer_state, COUNT(customer_id) AS no_of_customers
FROM
    customers
GROUP BY customer_state;
```