SQL OF THE DAY

💻 HIGHEST COST ORDERS

NIVAN R. SUGIANTORO

PROBLEMS

🔗 https://platform.stratascratch.com/coding/10352-users-by-avg-session-time?code_type=1

Problem:
Find the customers with the highest daily total order cost between 2019-02-01 and 2019-05-01. If a customer had more than one order on a certain day, sum the order costs on a daily basis. Output each customer's first name, total cost of their items, and the date. If multiple customers tie for the highest daily total on the same date, return all of them.

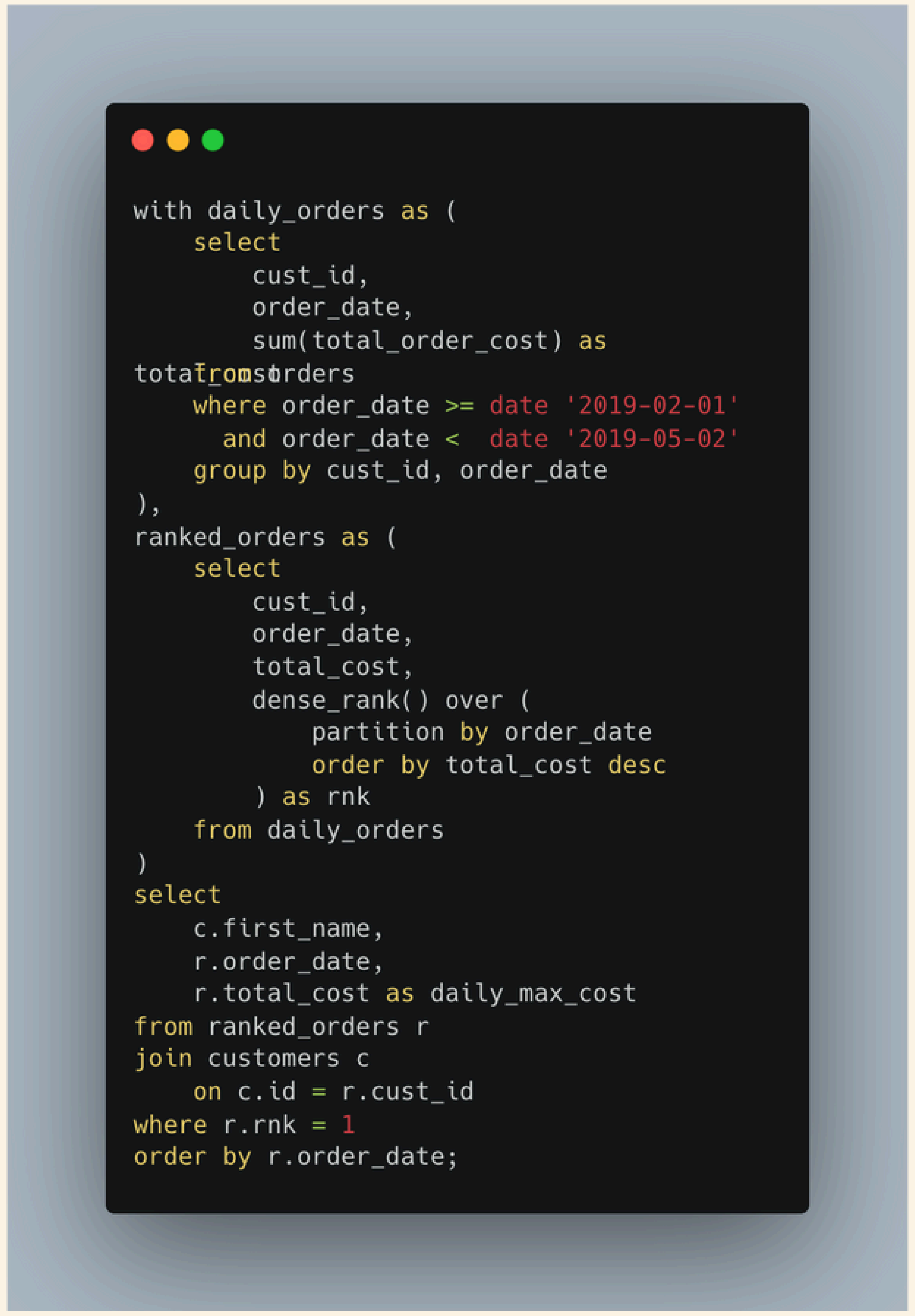
SOLUTION

🧩 Logic breakdown

daily_orders
→ total spending per customer per day

ranked_orders
→ rank customers per day by total spending

final select
→ pick top spender(s) each day
→ dense_rank() handles ties



```
with daily_orders as (
    select
        cust_id,
        order_date,
        sum(total_order_cost) as
total_cost
    from orders
    where order_date >= date '2019-02-01'
      and order_date <  date '2019-05-02'
    group by cust_id, order_date
),
ranked_orders as (
    select
        cust_id,
        order_date,
        total_cost,
        dense_rank() over (
            partition by order_date
            order by total_cost desc
        ) as rnk
    from daily_orders
)
select
    c.first_name,
    r.order_date,
    r.total_cost as daily_max_cost
from ranked_orders r
join customers c
    on c.id = r.cust_id
where r.rnk = 1
order by r.order_date;
```

KEY TAKEAWAYS

- Be careful with date ranges. between includes both ends; using >= and < is safer for time-based data.
- dense_rank() is important when ties matter. Without it, you may reward the wrong customer. With it, all true top contributors are captured.
- This pattern shows up a lot in marketing, loyalty, fraud detection, and rewards logic.

Let's Connect

nivanrs@gmail.com