JENSONUSA.COM

JENSON ADVANCE SQL PROJECT

PRESENTED BY : KINJAN CHAUHAN

ABOUT COMPANY

JENSON USA IS A LEADING ONLINE RETAILER OF BICYCLES, PARTS, APPAREL, AND ACCESSORIES FOR CYCLING:

PRODUCTS

- JENSON USA SELLS OVER 30,000 PRODUCTS FOR ROAD, MOUNTAIN, TRIATHLON, BMX, GRAVEL, AND COMMUTER BIKES. THEY ALSO SELL CAMP EQUIPMENT, LUGGAGE, AND TRAVEL BAGS.

MISSION

- JENSON USA'S MISSION IS TO BE THE BEST COMPANY FOR CUSTOMERS TO BUY FROM, FOR VENDORS TO SELL TO, AND FOR EMPLOYEES TO WORK FOR.

PROJECT DESCRIPTION

1 As a data analyst at Jensen's, craft SQL queries to derive insights on customer behavior, staff performance, inventory management, and store operations.

2 This project involves designing and populating a relational database for a fictional BikeStore. It includes SQL scripts to create the database schema (tables, relationships, constraints) and load sample data. The database can be utilized for training, query practice, and demonstrations of retail and inventory management scenarios.

ANALYSIS BASED ON

Sales Performance

Analyze sales trends by product, store, and time period

Customer Insights

Identify key customer demographics and purchasing behavior

Inventory Management

Evaluate stock levels, fast-moving products, and restocking needs

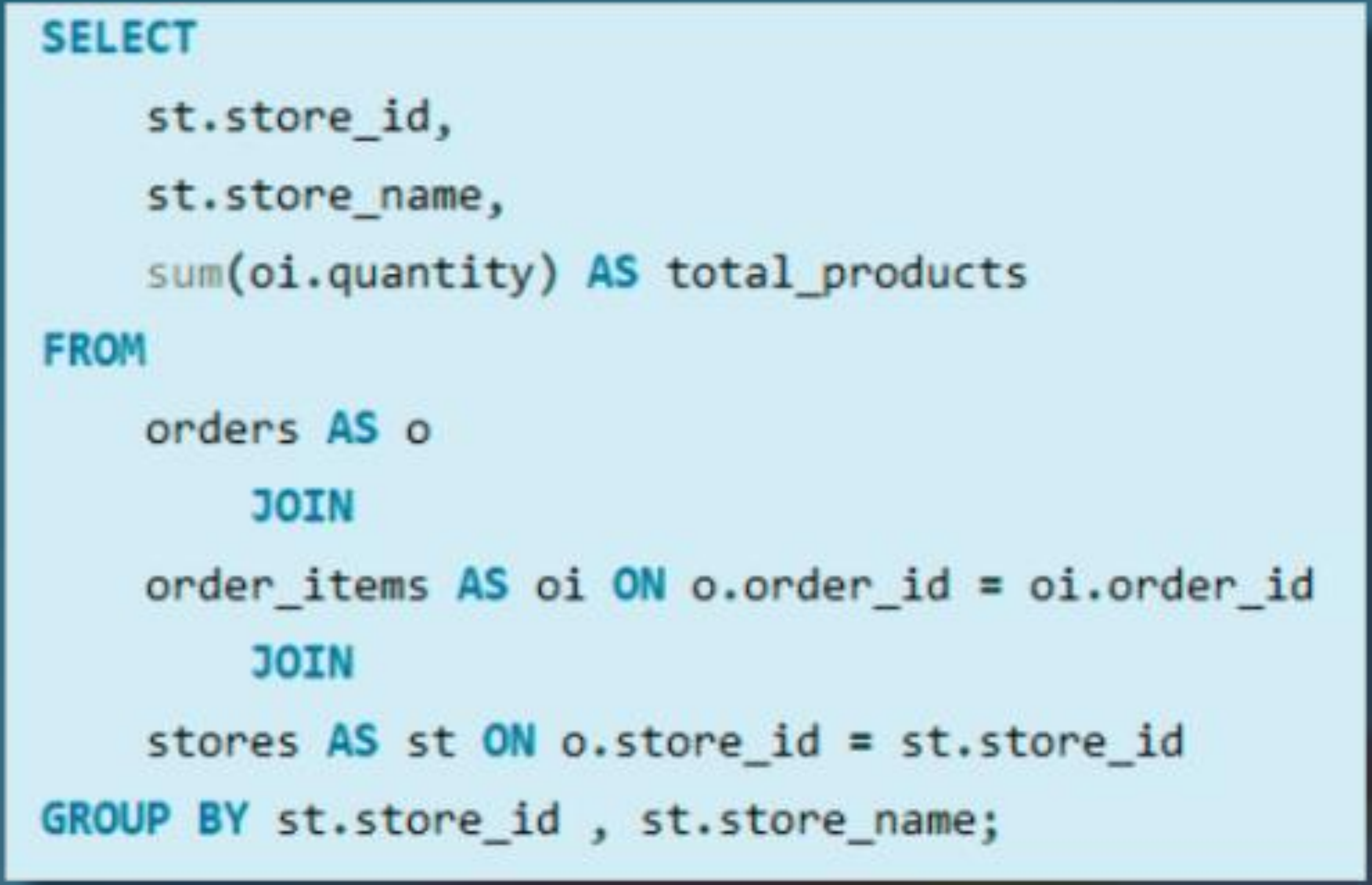
Store Comparisons

Compare performance metrics across different store locations

Employee Contribution

Assess sales performance by employee.

FIND THE TOTAL NUMBER OF PRODUCTS SOLD BY EACH STORE ALONG WITH THE STORE NAME.

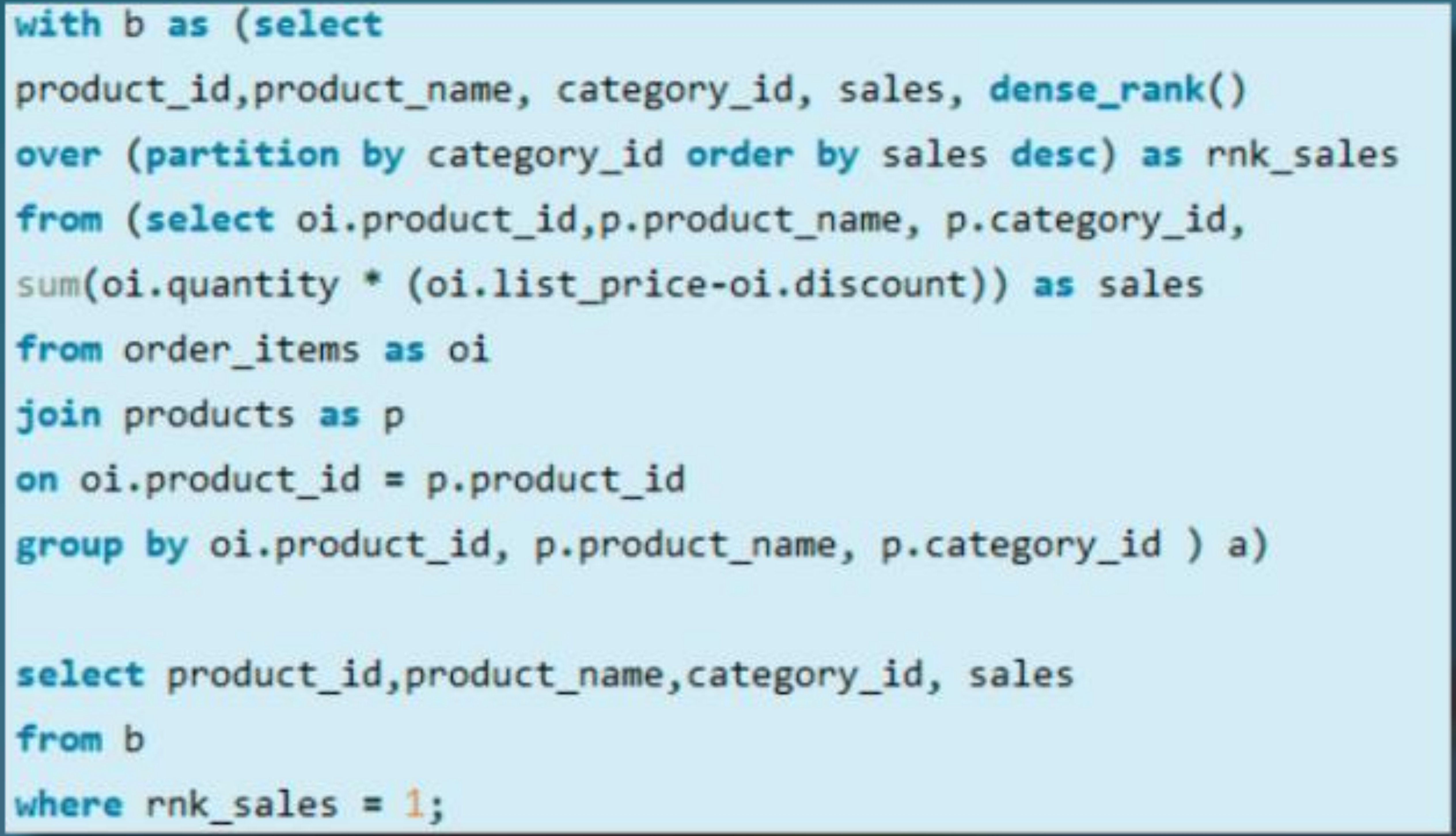


```
SELECT
    st.store_id,
    st.store_name,
    sum(oi.quantity) AS total_products
FROM
    orders AS o
        JOIN
    order_items AS oi ON o.order_id = oi.order_id
        JOIN
    stores AS st ON o.store_id = st.store_id
GROUP BY st.store_id , st.store_name;
```

CALCULATE THE CUMULATIVE SUM OF QUANTITIES SOLD FOR EACH PRODUCT OVER TIME.

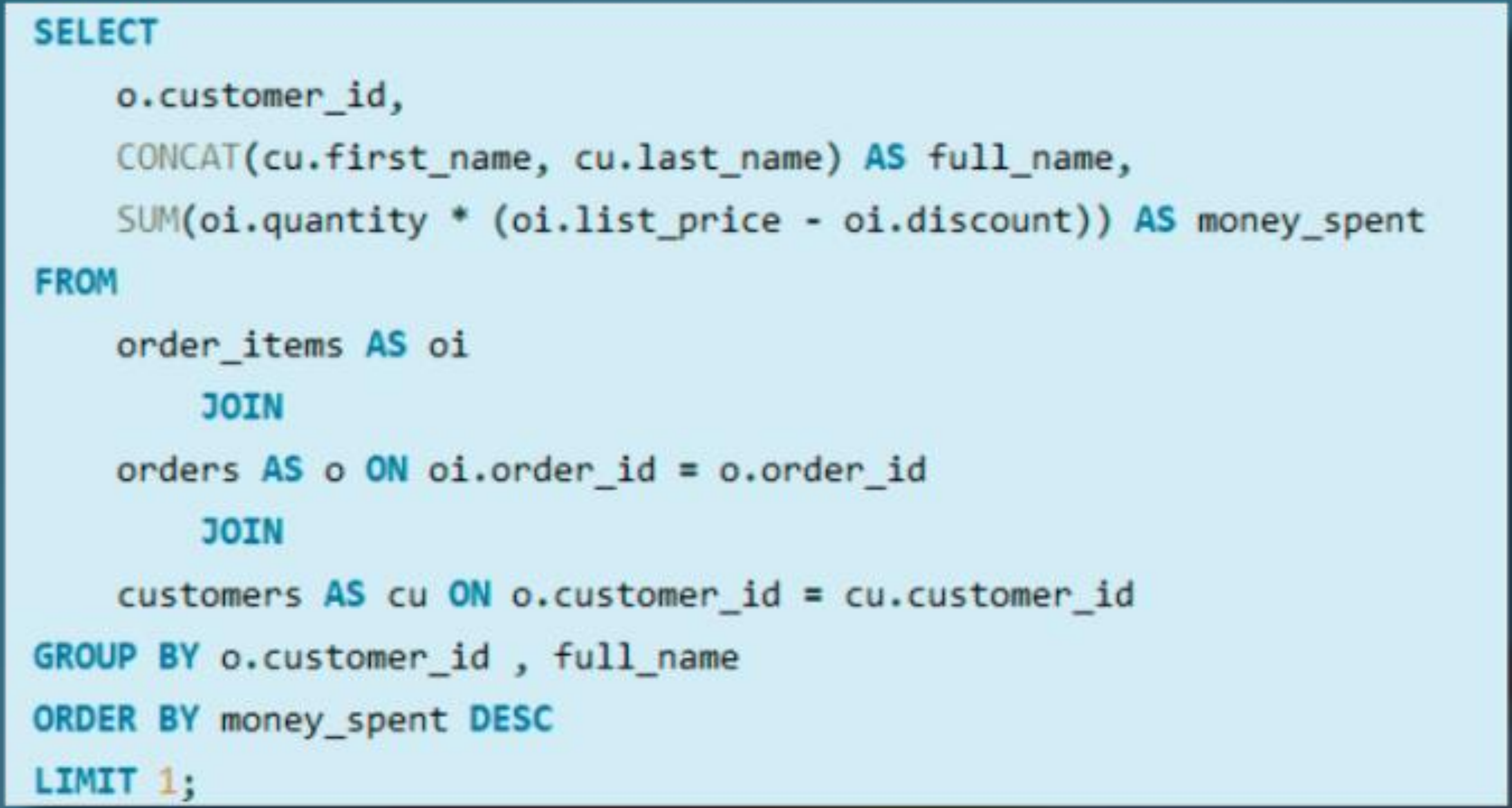
```sql
select oi.product_id,o.order_date, sum(oi.quantity)
over( partition by oi.product_id order by o.order_date) as cumulative_sum
from order_items as oi
join orders as o
on oi.order_id = o.order_id;
```

FIND THE PRODUCT WITH THE HIGHEST TOTAL SALES (QUANTITY * PRICE) FOR EACH CATEGORY.



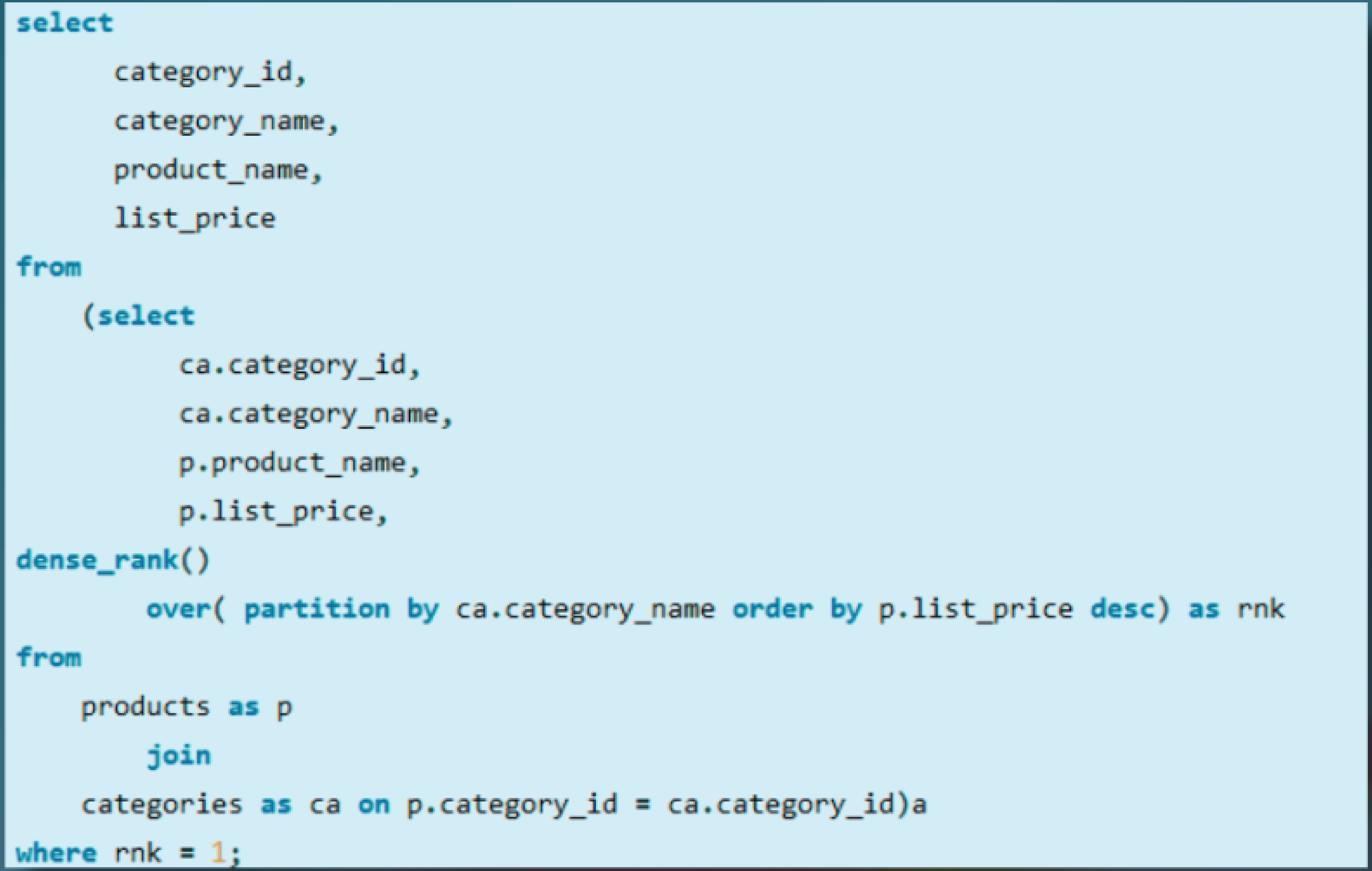
```sql
with b as (select
product_id,product_name, category_id, sales, dense_rank()
over (partition by category_id order by sales desc) as rnk_sales
from (select oi.product_id,p.product_name, p.category_id,
sum(oi.quantity * (oi.list_price-oi.discount)) as sales
from order_items as oi
join products as p
on oi.product_id = p.product_id
group by oi.product_id, p.product_name, p.category_id ) a)

select product_id,product_name,category_id, sales
from b
where rnk_sales = 1;
```

FIND THE CUSTOMER WHO SPENT THE MOST MONEY ON ORDERS.



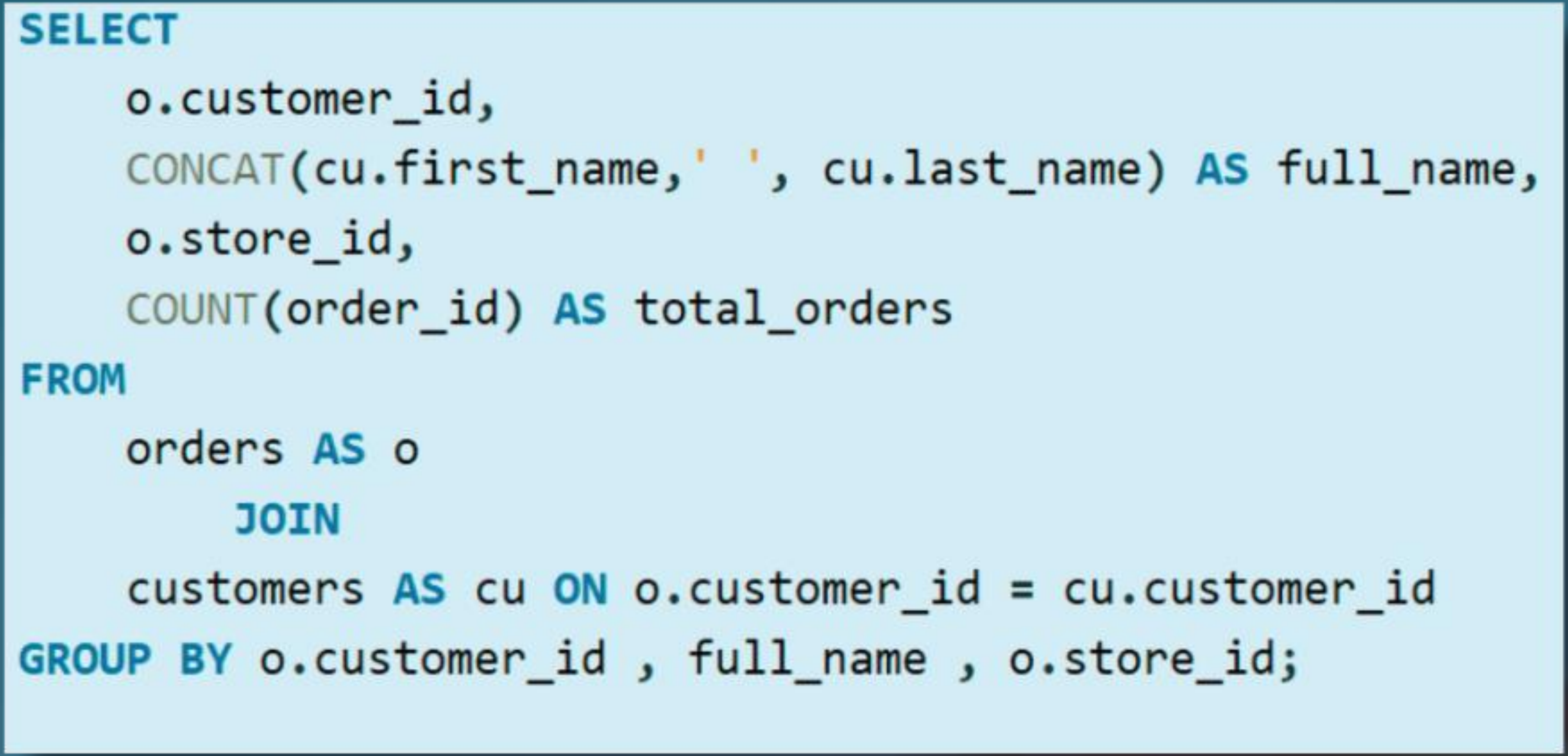
```sql
SELECT
    o.customer_id,
    CONCAT(cu.first_name, cu.last_name) AS full_name,
    SUM(oi.quantity * (oi.list_price - oi.discount)) AS money_spent
FROM
    order_items AS oi
        JOIN
    orders AS o ON oi.order_id = o.order_id
        JOIN
    customers AS cu ON o.customer_id = cu.customer_id
GROUP BY o.customer_id , full_name
ORDER BY money_spent DESC
LIMIT 1;
```

FIND THE HIGHEST-PRICED PRODUCT FOR EACH CATEGORY NAME.



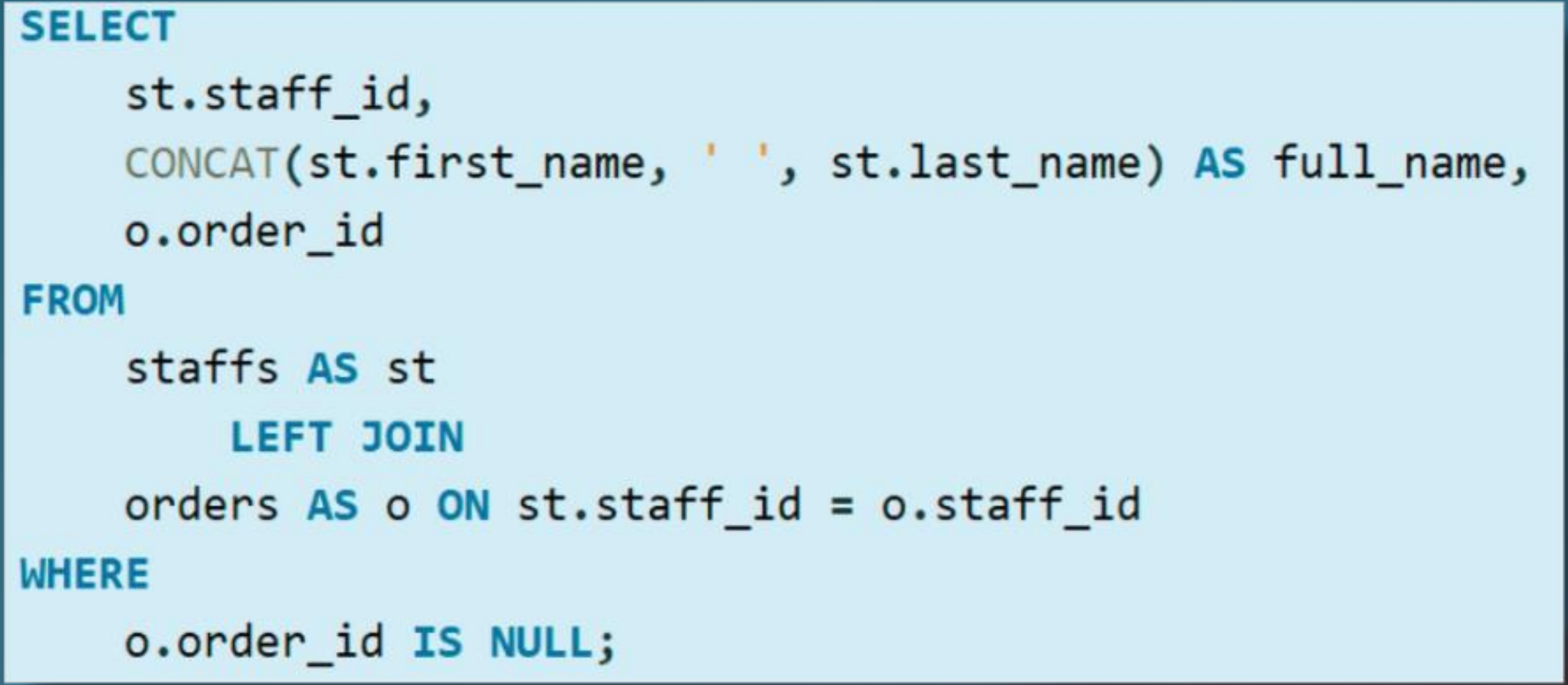
```
select
      category_id,
      category_name,
      product_name,
      list_price
from
    (select
          ca.category_id,
          ca.category_name,
          p.product_name,
          p.list_price,
dense_rank()
        over( partition by ca.category_name order by p.list_price desc) as rnk
from
    products as p
        join
    categories as ca on p.category_id = ca.category_id)a
where rnk = 1;
```

FIND THE TOTAL NUMBER OF ORDERS PLACED BY EACH CUSTOMER PER STORE.



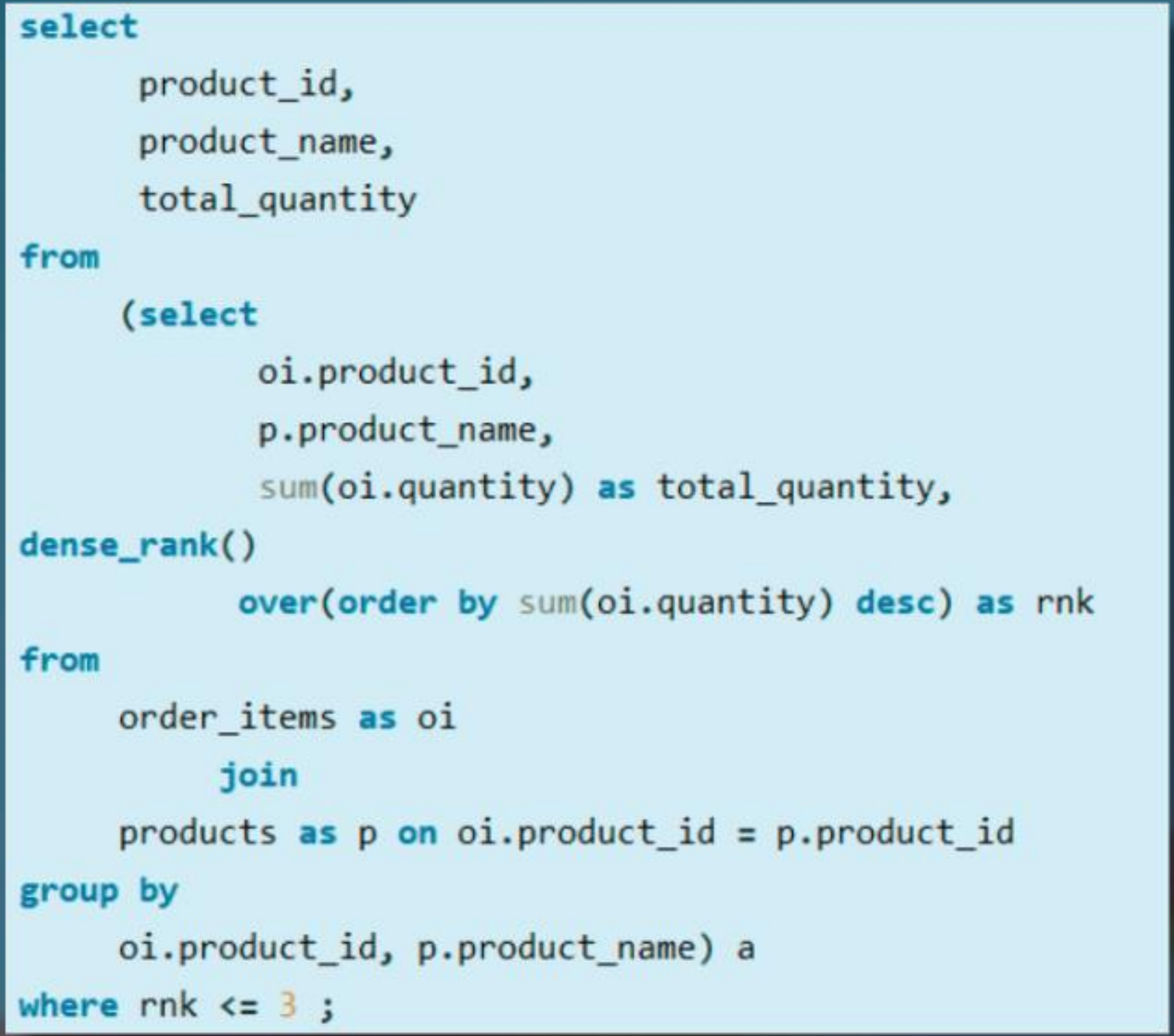
```sql
SELECT
    o.customer_id,
    CONCAT(cu.first_name,' ', cu.last_name) AS full_name,
    o.store_id,
    COUNT(order_id) AS total_orders
FROM
    orders AS o
        JOIN
    customers AS cu ON o.customer_id = cu.customer_id
GROUP BY o.customer_id , full_name , o.store_id;
```

FIND THE NAMES OF STAFF MEMBERS WHO HAVE NOT MADE ANY SALES.



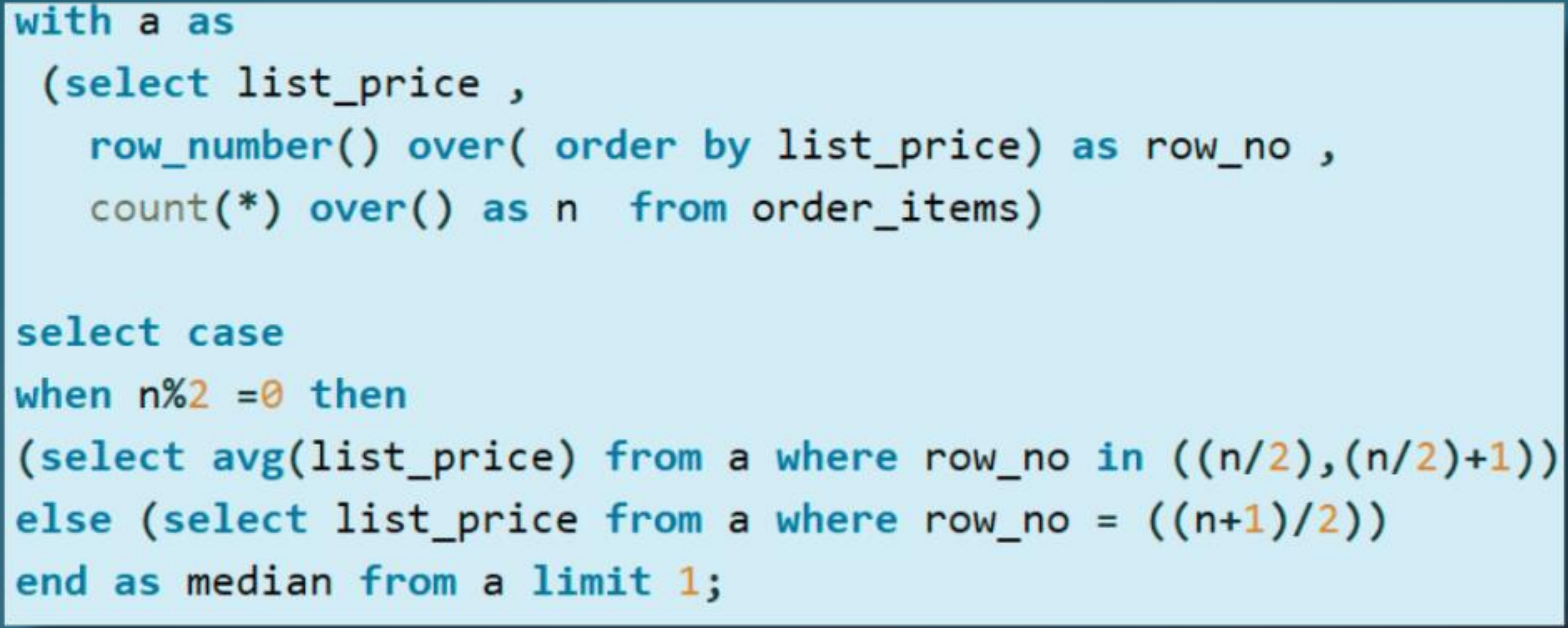
```sql
SELECT
    st.staff_id,
    CONCAT(st.first_name, ' ', st.last_name) AS full_name,
    o.order_id
FROM
    staffs AS st
        LEFT JOIN
    orders AS o ON st.staff_id = o.staff_id
WHERE
    o.order_id IS NULL;
```

FIND THE TOP 3 MOST SOLD PRODUCTS IN TERMS OF QUANTITY.



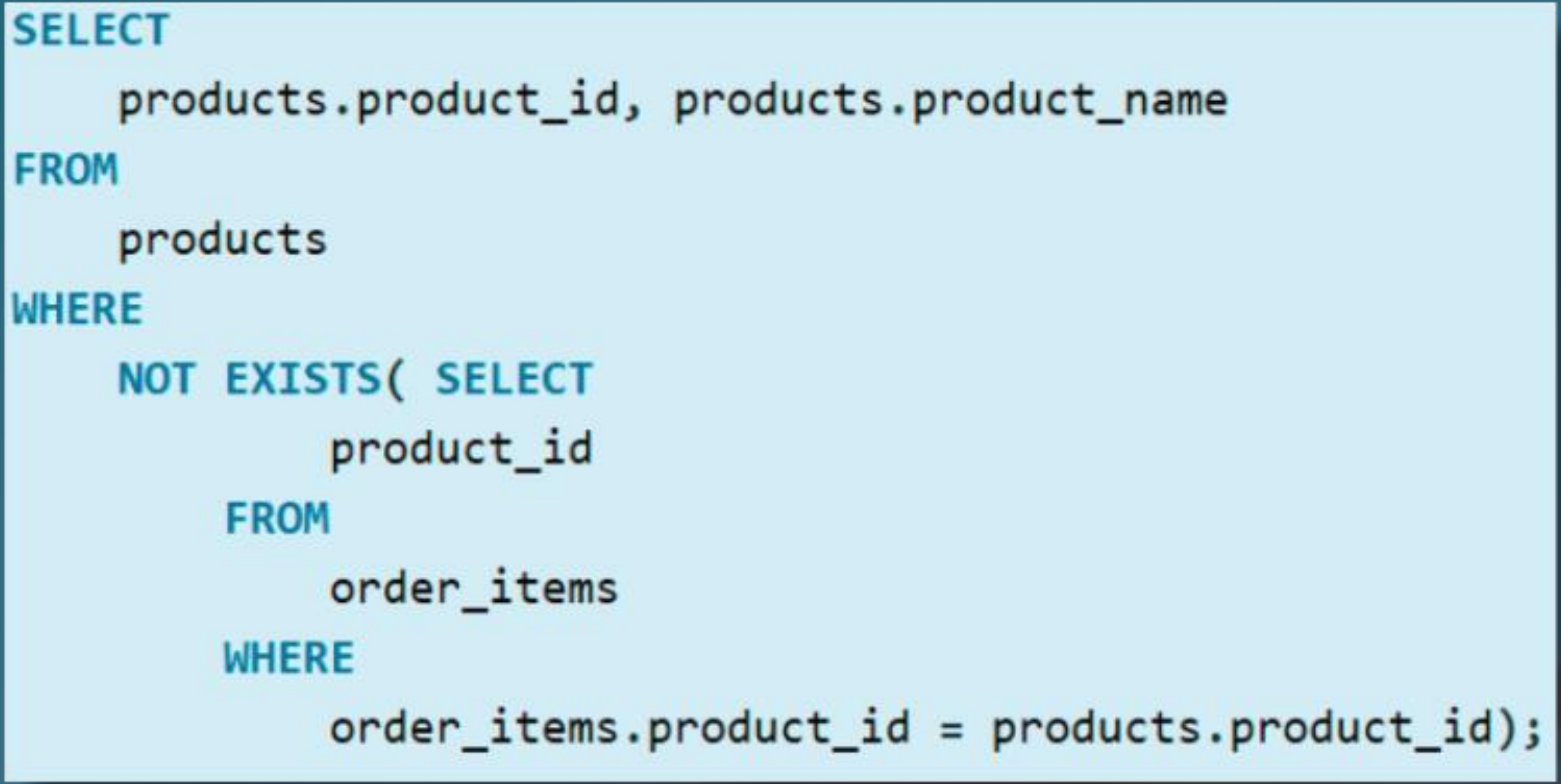
```
select
      product_id,
      product_name,
      total_quantity
from
     (select
            oi.product_id,
            p.product_name,
            sum(oi.quantity) as total_quantity,
dense_rank()
           over(order by sum(oi.quantity) desc) as rnk
from
     order_items as oi
          join
     products as p on oi.product_id = p.product_id
group by
     oi.product_id, p.product_name) a
where rnk <= 3 ;
```

FIND THE MEDIAN VALUE OF THE PRICE LIST.



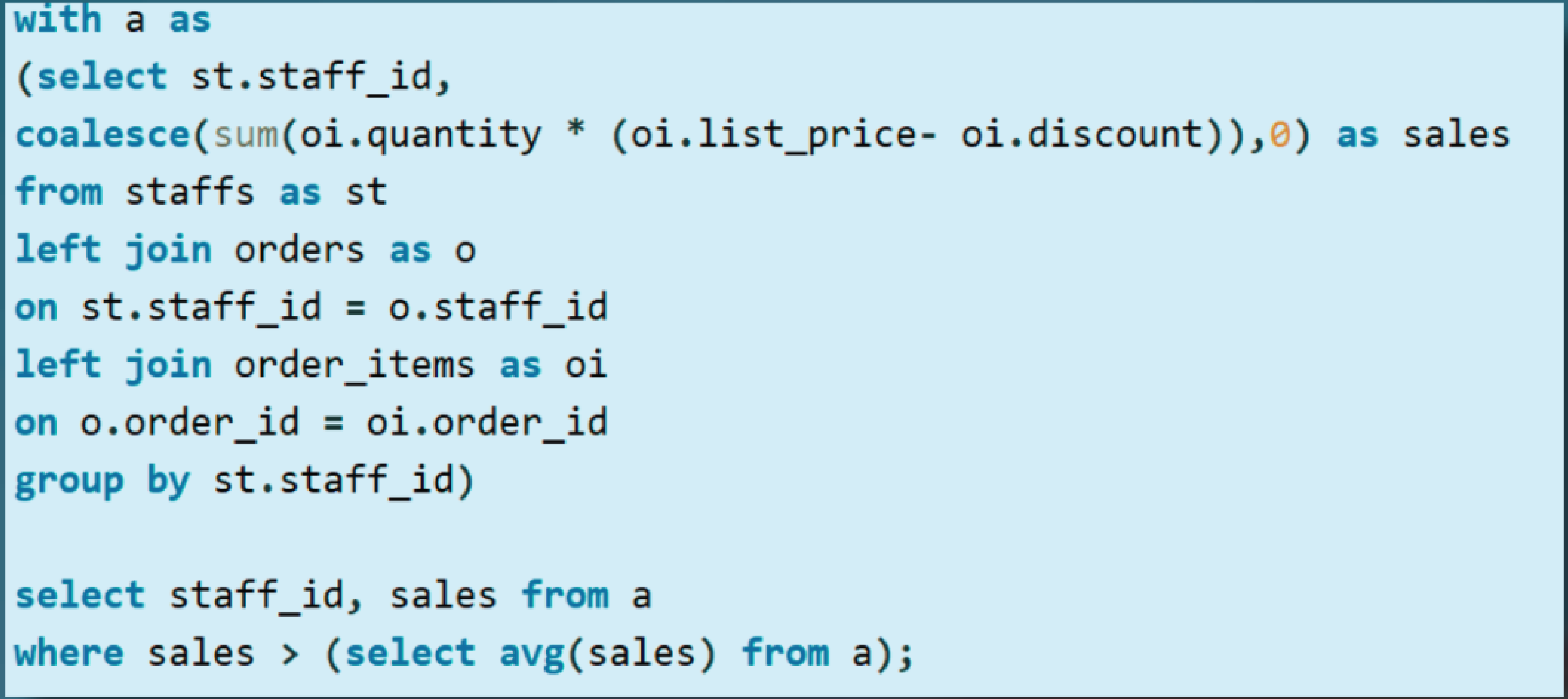
```
with a as
 (select list_price ,
   row_number() over( order by list_price) as row_no ,
   count(*) over() as n  from order_items)

select case
when n%2 =0 then
(select avg(list_price) from a where row_no in ((n/2),(n/2)+1))
else (select list_price from a where row_no = ((n+1)/2))
end as median from a limit 1;
```

LIST ALL PRODUCTS THAT HAVE NEVER BEEN ORDERED.(USE EXISTS)



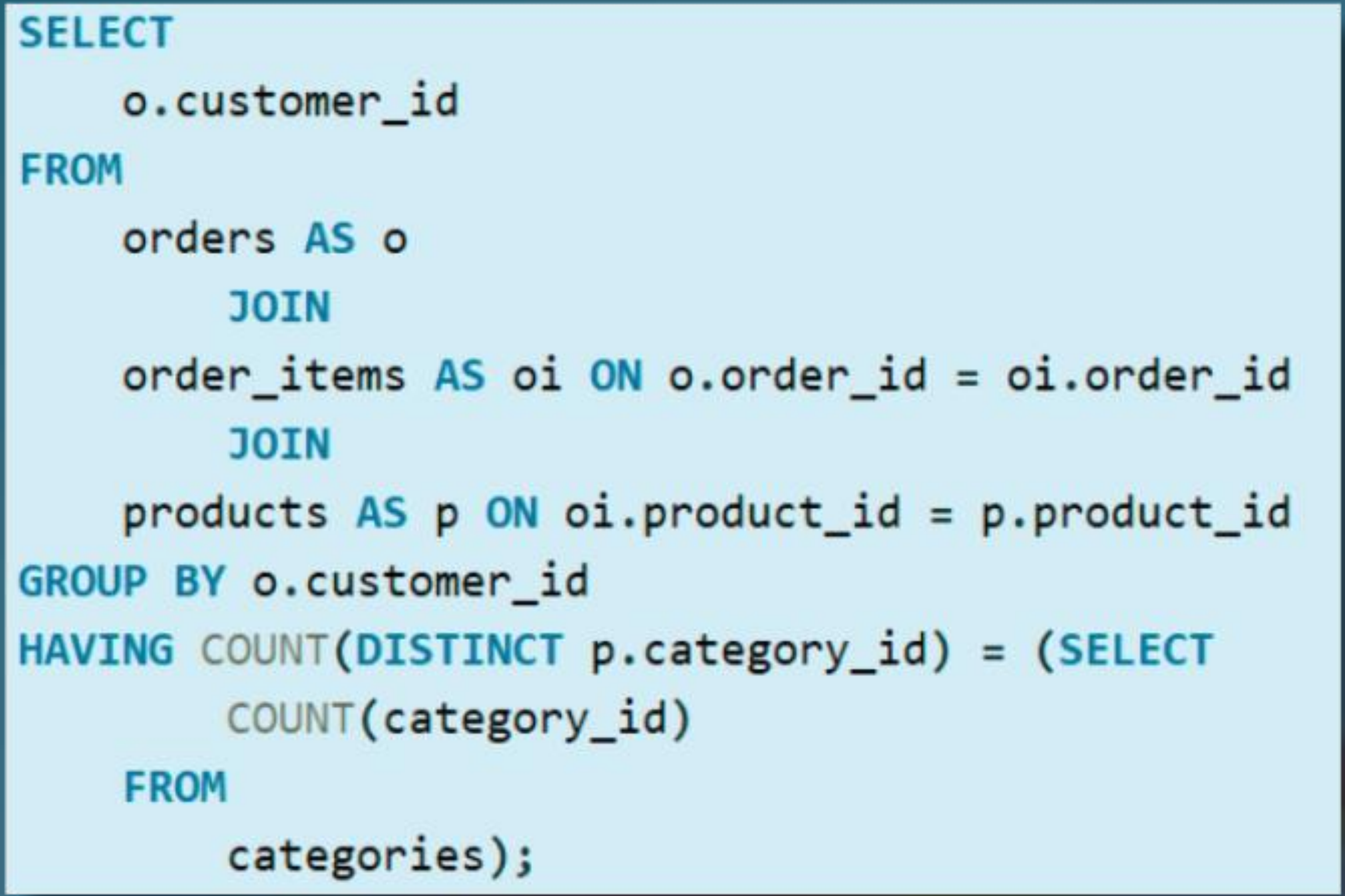
```sql
SELECT
    products.product_id, products.product_name
FROM
    products
WHERE
    NOT EXISTS( SELECT
            product_id
        FROM
            order_items
        WHERE
            order_items.product_id = products.product_id);
```

LIST THE NAMES OF STAFF MEMBERS WHO HAVE MADE MORE SALES THAN THE AVERAGE NUMBER OF SALES BY ALL STAFF MEMBERS.



```sql
with a as
(select st.staff_id,
coalesce(sum(oi.quantity * (oi.list_price- oi.discount)),0) as sales
from staffs as st
left join orders as o
on st.staff_id = o.staff_id
left join order_items as oi
on o.order_id = oi.order_id
group by st.staff_id)

select staff_id, sales from a
where sales > (select avg(sales) from a);
```

IDENTIFY THE CUSTOMERS WHO HAVE ORDERED ALL TYPES OF PRODUCTS (I.E., FROM EVERY CATEGORY).



```
SELECT
    o.customer_id
FROM
    orders AS o
        JOIN
    order_items AS oi ON o.order_id = oi.order_id
        JOIN
    products AS p ON oi.product_id = p.product_id
GROUP BY o.customer_id
HAVING COUNT(DISTINCT p.category_id) = (SELECT
        COUNT(category_id)
    FROM
        categories);
```

THANK YOU

PRESENTED BY : KINJAN CHAUHAN