/\*Q1. Rank employees by their total sales

(Total sales = Total no of orders handled, JOIN employees and orders table)\*/

select e.employee\_id, count(o.order\_id),

rank()over(order by count(o.order\_id) desc ) as Rank\_employee

from employees e

join orders o

on e.employee\_id = o.employee\_id

group by e.employee\_id

order by Rank\_employee;

A screenshot of a computer

AI-generated content may be incorrect.

/\*Q2.Compare current order's freight with previous and next order for each customer.

(Display order\_id, customer\_id, order\_date, freight,

Use lead(freight) and lag(freight).\*/

select order\_id, customer\_id,order\_date,freight,

lag(freight,1,freight) over (partition by customer\_id order by order\_date) as lag\_value,

lead(freight,1,freight) over (partition by customer\_id order by order\_date) as lead\_value,

case when freight > lag(freight) over (partition by customer\_id order by order\_date) then 'more than previous'

when freight < lag(freight) over (partition by customer\_id order by order\_date) then 'less than previous'

else 'no change'

end as lag\_comparision,

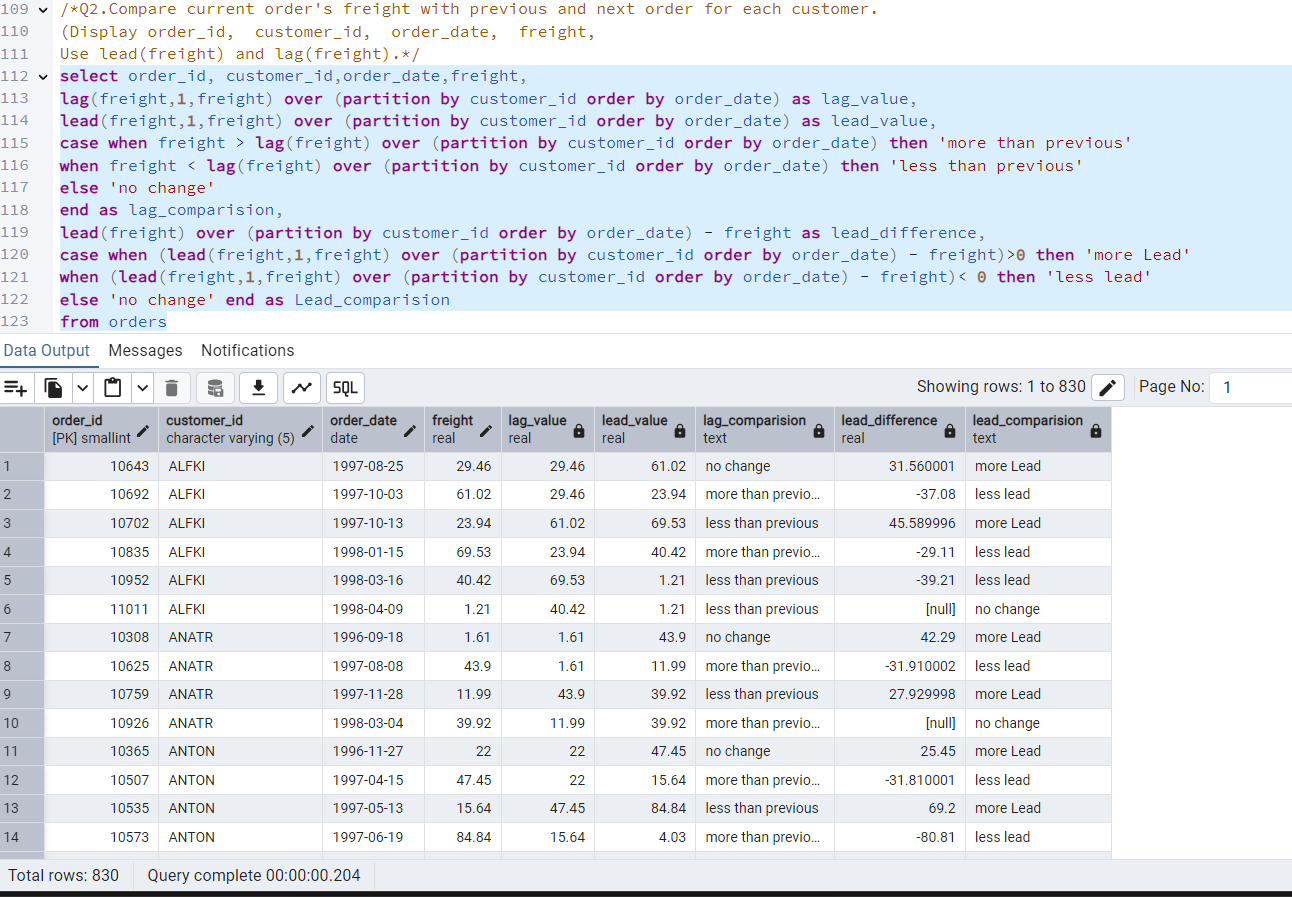
lead(freight) over (partition by customer\_id order by order\_date) - freight as lead\_difference,

case when (lead(freight,1,freight) over (partition by customer\_id order by order\_date) - freight)>0 then 'more Lead'

when (lead(freight,1,freight) over (partition by customer\_id order by order\_date) - freight)< 0 then 'less lead'

else 'no change' end as Lead\_comparision

from orders



/\* Q3. Show products and their price categories, product count in each category, avg price:

(HINT:

· Create a CTE which should have price\_category definition:

WHEN unit\_price < 20 THEN 'Low Price'

WHEN unit\_price < 50 THEN 'Medium Price'

ELSE 'High Price'

· In the main query display: price\_category, product\_count in each price\_category, ROUND(AVG(unit\_price)::numeric, 2) as avg\_price)\*/

WITH categorized\_products AS (

SELECT category\_id,product\_id, product\_name, unit\_price,

CASE

WHEN unit\_price < 20 THEN 'Low Price'

WHEN unit\_price < 50 THEN 'Medium Price'

ELSE 'High Price'

END AS price\_category

FROM products

)

SELECT category\_id, price\_category, COUNT(\*) AS product\_count,

ROUND(AVG(unit\_price)::numeric, 2) AS avg\_price

FROM categorized\_products

GROUP BY category\_id ,price\_category

ORDER BY category\_id;

A screenshot of a computer

AI-generated content may be incorrect.