**Day 7**

1. Rank employees by their total sales

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

**Query:**

SELECT

E.EMPLOYEE\_ID,

COUNT(ORDER\_ID) AS TOTAL\_SALES,

RANK() OVER (ORDER BY COUNT(O.ORDER\_ID)) AS SALES\_RANK

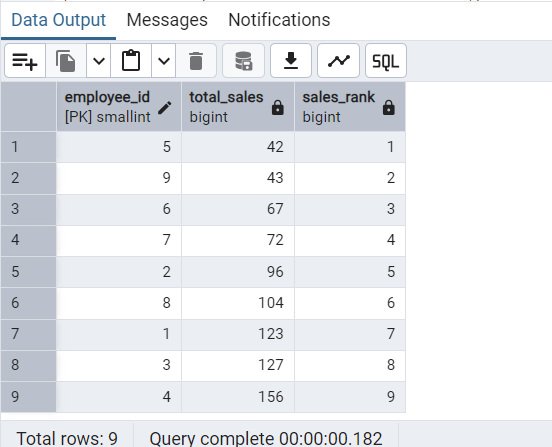
FROM

ORDERS O

JOIN EMPLOYEES E ON E.EMPLOYEE\_ID = O.EMPLOYEE\_ID

GROUP BY E.EMPLOYEE\_ID;

**Output:**

****

2. 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).

**Query:**

SELECT

ORDER\_ID,

CUSTOMER\_ID,

ORDER\_DATE,

FREIGHT,

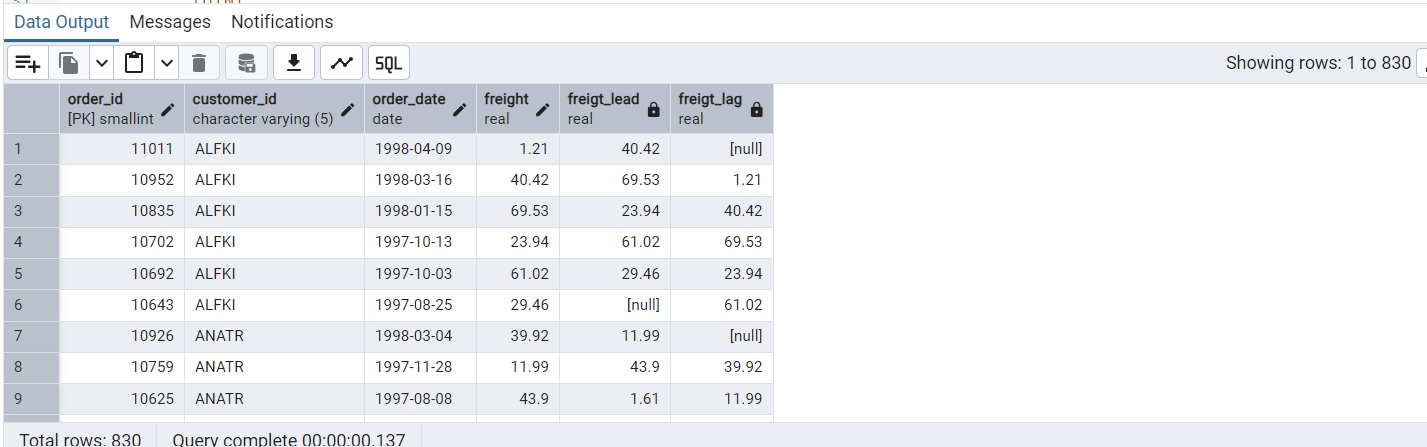
LEAD(FREIGHT) OVER (PARTITION BY CUSTOMER\_ID ORDER BY ORDER\_DATE DESC ) AS FREIGT\_LEAD,

LAG(FREIGHT) OVER (PARTITION BY CUSTOMER\_ID ORDER BY ORDER\_DATE DESC) AS FREIGT\_LAG

FROM

ORDERS;

**Output:**

****

3. 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)

**Query:**

WITH cte\_price\_category as(

SELECT

product\_name,product\_id,category\_id,

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

price\_category,category\_id,

COUNT(product\_id),

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

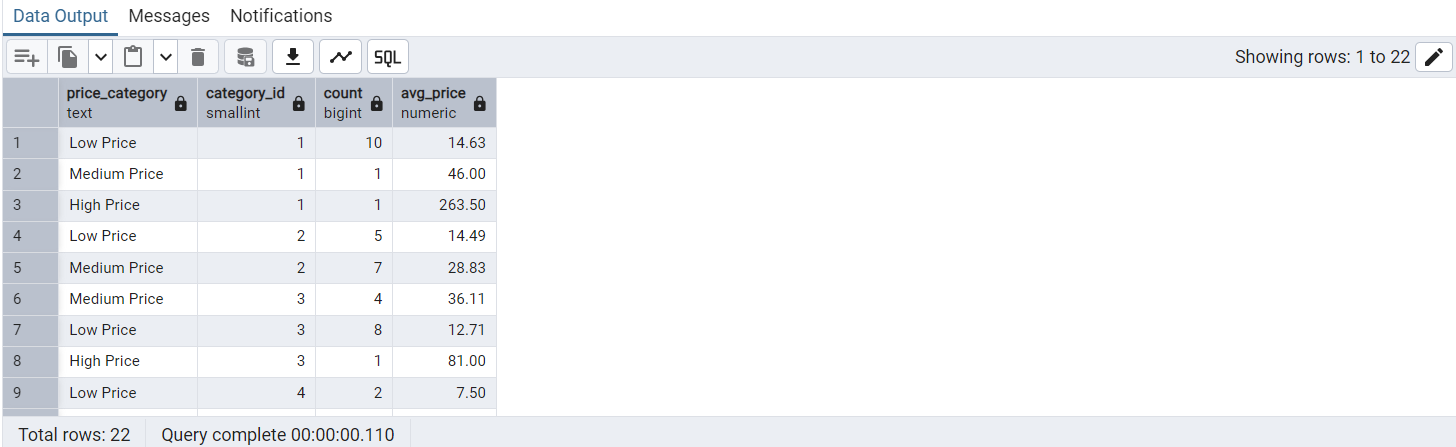
FROM cte\_price\_category

group by

category\_id,price\_category

order by category\_id;

**Output:**

****