**Assignment 7**

1.     Rank employees by their total sales

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

SELECT

e.employee\_id,

e.first\_name || e.last\_name AS employeename,

COUNT(o.order\_id) AS total\_sales,

RANK() OVER (ORDER BY COUNT(o.order\_id) DESC) AS sales\_rank

FROM

employees e

JOIN

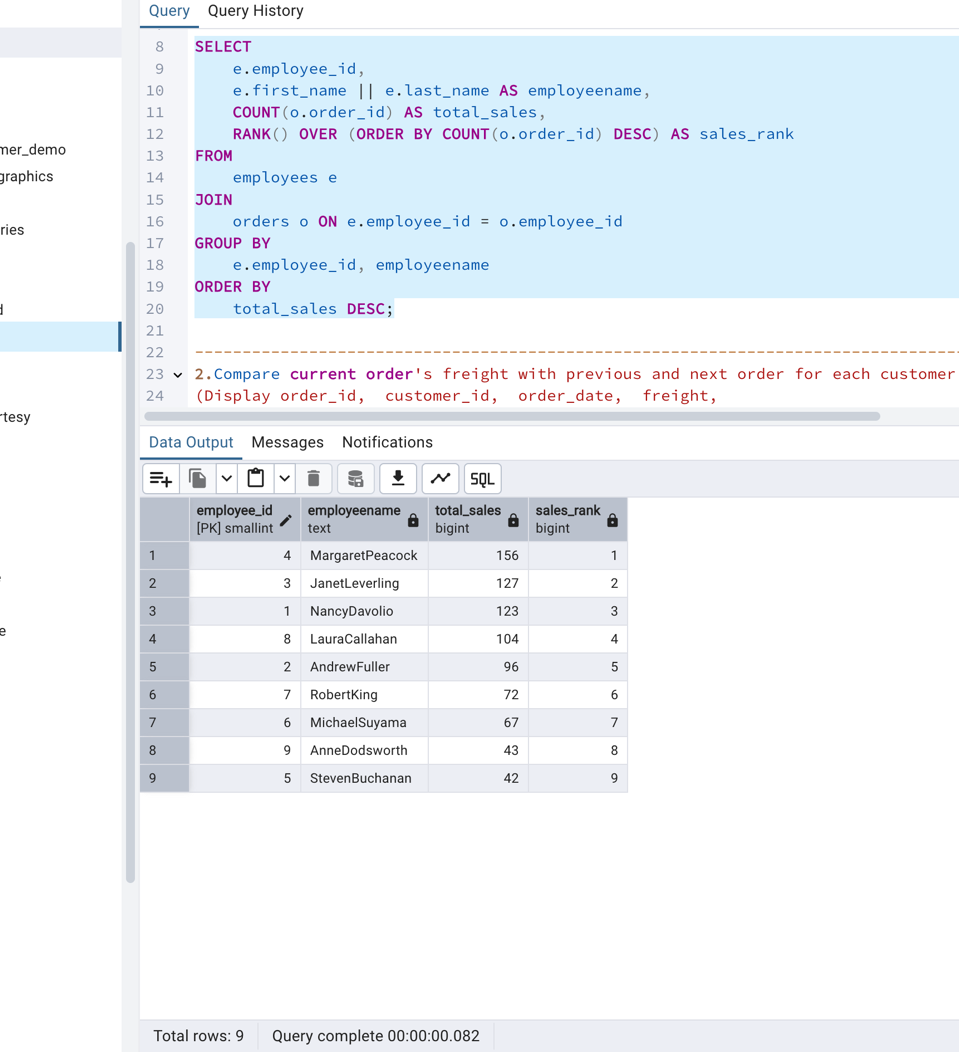
orders o ON e.employee\_id = o.employee\_id

GROUP BY

e.employee\_id, employeename

ORDER BY

total\_sales DESC;



1. Using Dense rank

SELECT e.employee\_id,

COUNT(o.order\_id) AS total\_sales,

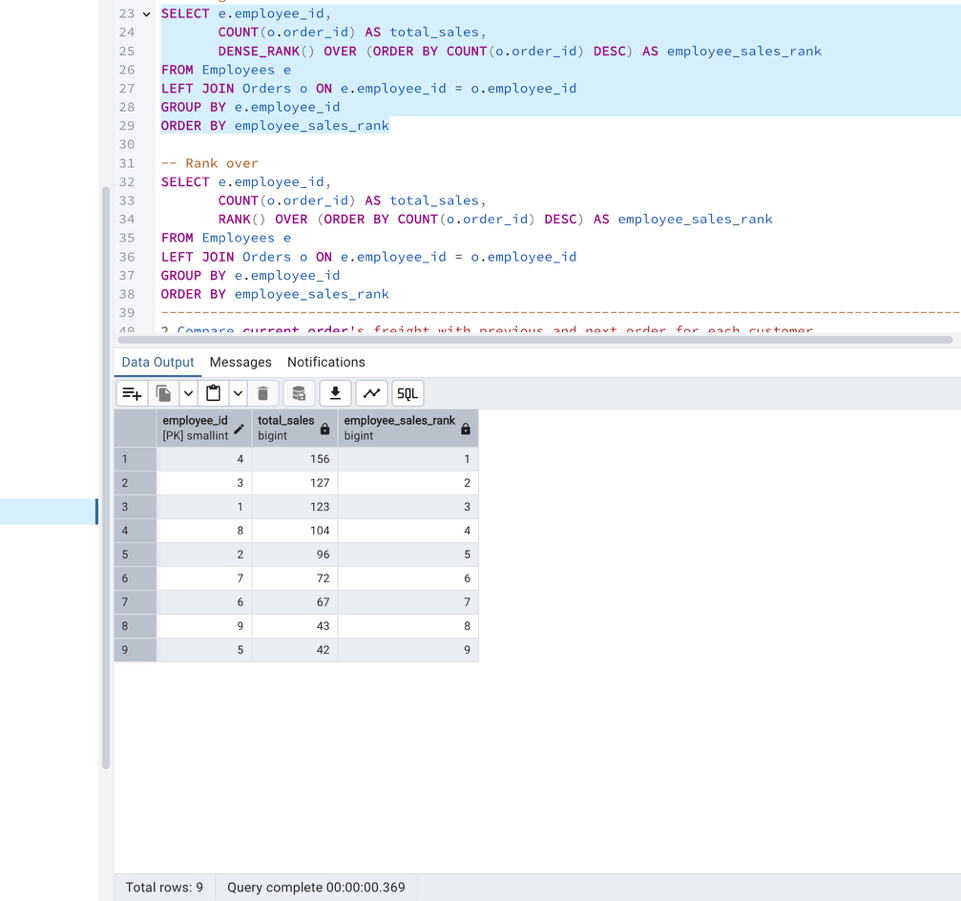
DENSE\_RANK() OVER (ORDER BY COUNT(o.order\_id) DESC) AS employee\_sales\_rank

FROM Employees e

LEFT JOIN Orders o ON e.employee\_id = o.employee\_id

GROUP BY e.employee\_id

ORDER BY employee\_sales\_rank



1. Using Rank Over

SELECT e.employee\_id,

COUNT(o.order\_id) AS total\_sales,

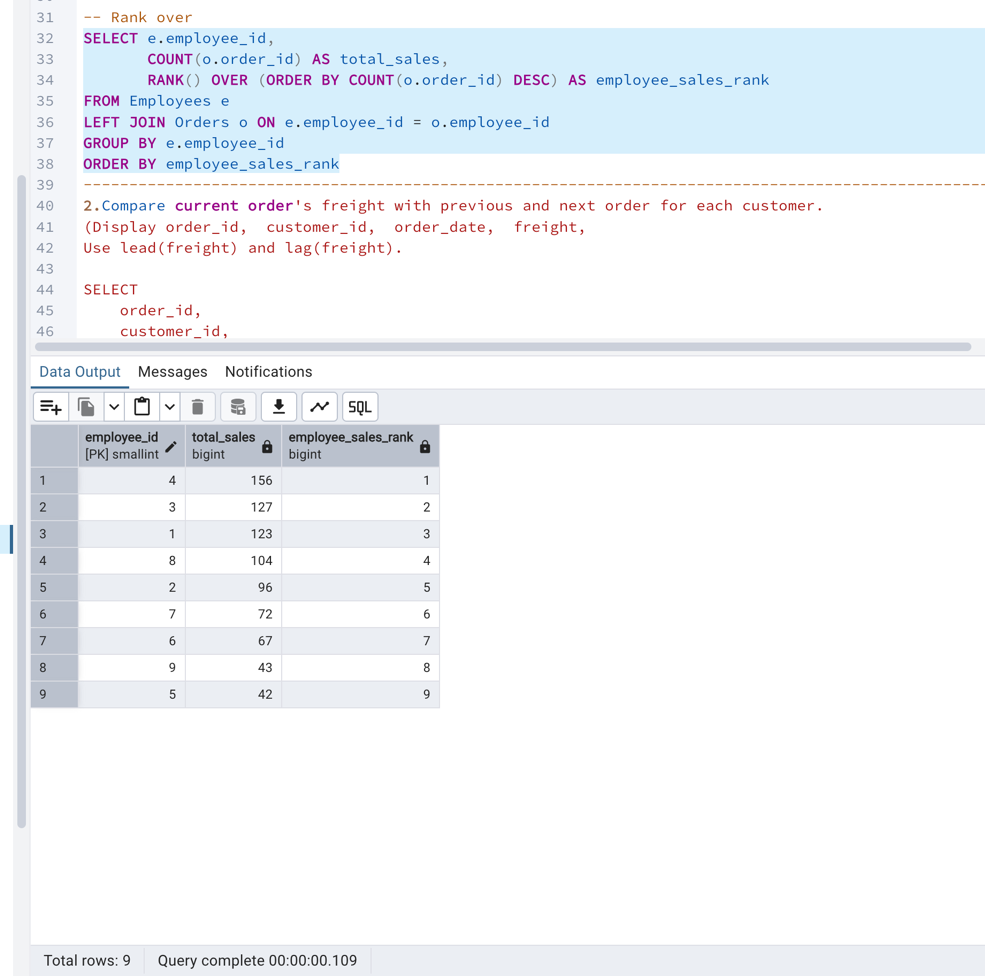
DENSE\_RANK() OVER (ORDER BY COUNT(o.order\_id) DESC) AS employee\_sales\_rank

FROM Employees e

LEFT JOIN Orders o ON e.employee\_id = o.employee\_id

GROUP BY e.employee\_id

ORDER BY employee\_sales\_rank



-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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).

SELECT

order\_id,

customer\_id,

order\_date,

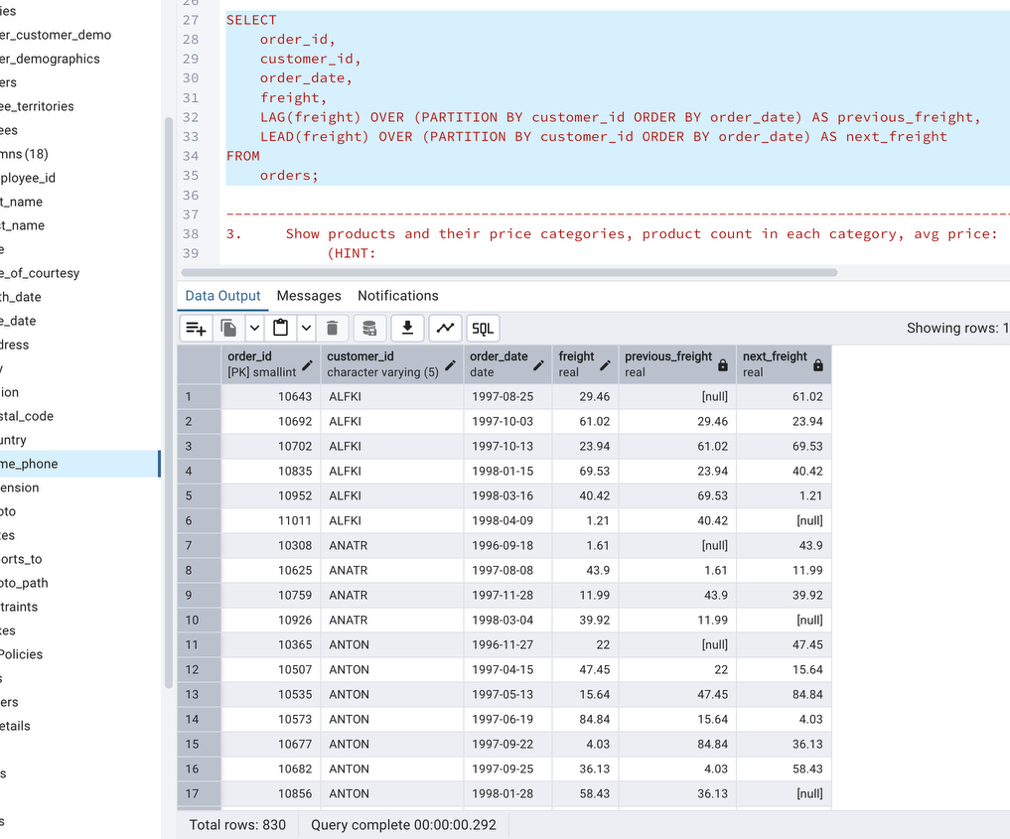
freight,

LAG(freight) OVER (PARTITION BY customer\_id ORDER BY order\_date) AS previous\_freight,

LEAD(freight) OVER (PARTITION BY customer\_id ORDER BY order\_date) AS next\_freight

FROM

orders;



-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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

WITH price\_categories AS (

SELECT

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

price\_category,

COUNT(\*) AS product\_count,

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

FROM

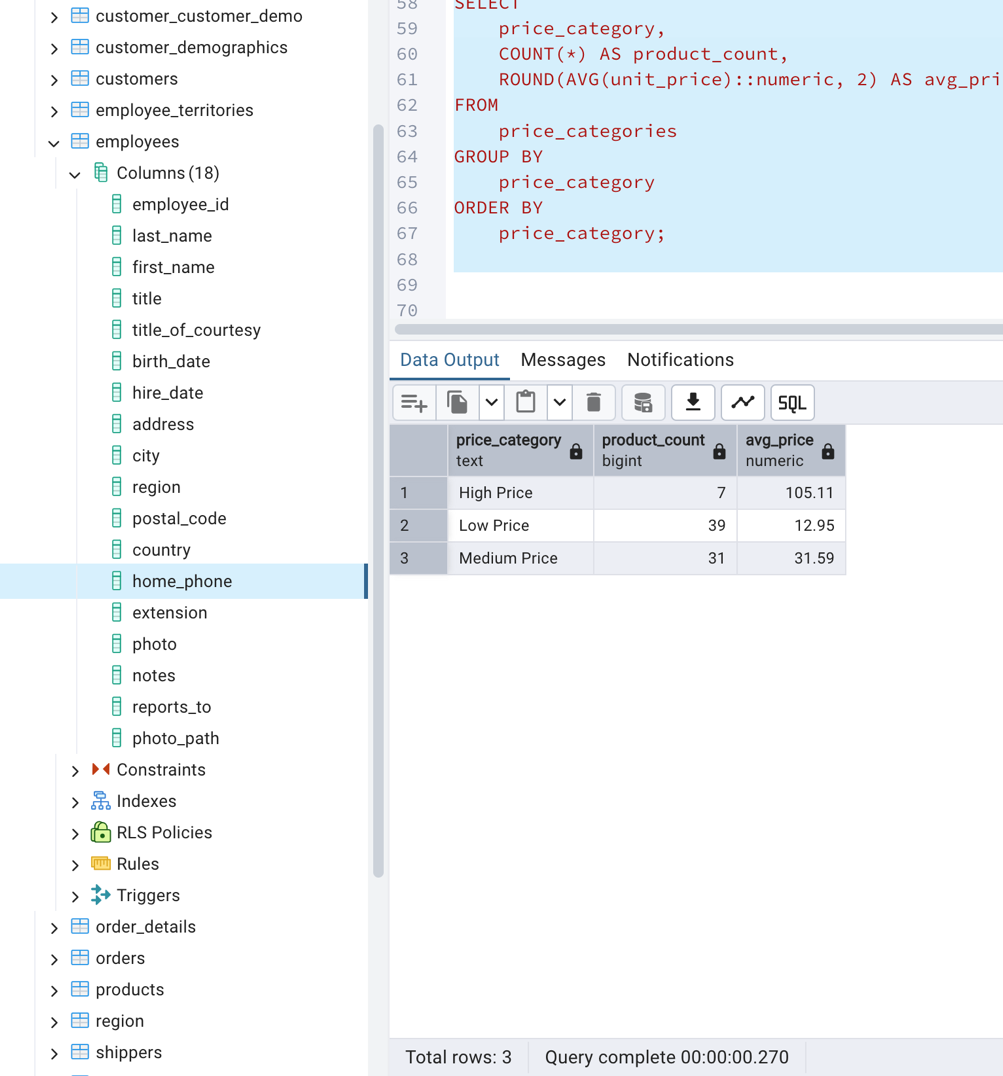
price\_categories

GROUP BY

price\_category

ORDER BY

price\_category;



-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------