--Day 7

--1.Rank employees by their total sales

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

Select emp.employee\_id,

Count(ord.order\_id) As total\_sales,

Rank()Over(Order By Count(ord.order\_id) Desc) As Sales\_rank

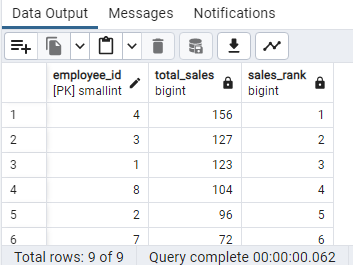
From employees emp

Join orders ord

On emp.employee\_id=ord.employee\_id

Group By emp.employee\_id

Order By total\_sales Desc;



--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 customer\_id,

order\_id,

order\_date,

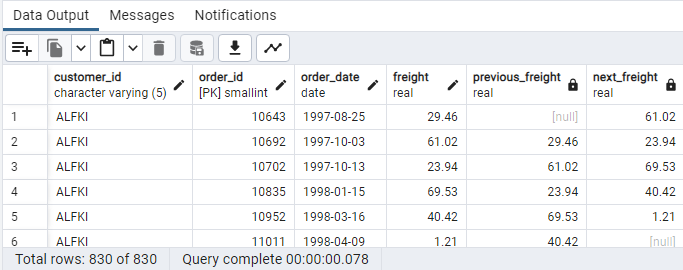
freight,

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 public.orders

Order By 1,2;



--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\_category As

(Select 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

price\_category,

Count(\*) As product\_count,

Round(Cast(Avg(unit\_price) As numeric), 2) As avg\_price

From price\_category

Group By 1

Order By 1;

