

Assignment 7










Day 7

1. Rank employees by their total sales

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

```
SELECT * FROM employees;
SELECT * FROM orders;
SELECT * FROM products;

SELECT
    E.employee_id,
    COUNT (*) AS total_sales,
    Rank() over(order by count(order_id) desc) AS sales_rank
FROM orders O
JOIN employees E on O.employee_id = E.employee_id
GROUP BY E.employee_id
ORDER BY 2 DESC;
```

Data Output Messages Notifications			
         SQL			
	employee_id [PK] smallint	total_sales bigint	sales_rank bigint
1	4	156	1
2	3	127	2
3	1	123	3
4	8	104	4
5	2	96	5
6	7	72	6
7	6	67	7
8	9	43	8
9	5	42	9

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)).

```
21 SELECT order_id, customer_id, order_date, freight,
22 LAG(freight) OVER (partition by customer_id order by freight)AS previous_order_freight,
23 LEAD(freight) OVER (partition by customer_id order by freight) AS next_order_freight
24 FROM orders;
```

Data OutputMessagesNotifications

SQL

	order_id [PK] smallint	customer_id character varying (5)	order_date date	freight real	previous_order_freight real	next_order_freight real
1	11011	ALFKI	1998-04-09	1.21	[null]	23.94
2	10702	ALFKI	1997-10-13	23.94	1.21	29.46
3	10643	ALFKI	1997-08-25	29.46	23.94	40.42
4	10952	ALFKI	1998-03-16	40.42	29.46	61.02
5	10692	ALFKI	1997-10-03	61.02	40.42	69.53
6	10835	ALFKI	1998-01-15	69.53	61.02	[null]
7	10308	ANATR	1996-09-18	1.61	[null]	11.99
8	10759	ANATR	1997-11-28	11.99	1.61	39.92
9	10926	ANATR	1998-03-04	39.92	11.99	43.9
10	10625	ANATR	1997-08-08	43.9	39.92	[null]
11	10677	ANTON	1997-09-22	4.03	[null]	15.64
12	10505	ANTON	1997-05-10	15.64	4.03	80.00

Total rows: 830Query complete 00:00:00.126

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)

```
33 WITH cte_price_category As(  
34 SELECT product_id,product_name, unit_price,  
35 CASE  
36     WHEN unit_price < 20 THEN 'Low Price'  
37     WHEN unit_price < 50 THEN 'Medium Price'  
38     ELSE 'High Price'  
39 END AS price_category  
40 FROM products  
41 )  
42 --main query  
43 SELECT price_category,  
44 COUNT (*) AS product_count,  
45 ROUND(AVG(unit_price)::numeric, 2) as avg_price  
46 FROM cte_price_category  
47 GROUP BY price_category  
48 ORDER BY price_category;
```

Data Output Messages Notifications

	price_category text	product_count bigint	avg_price numeric
1	High Price	7	105.11
2	Low Price	39	12.95
3	Medium Price	31	31.59

