DAY 7 ASSIGNMENT

1.Rank employees by their total sales

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

```
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;
```

```
/* 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
      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

	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

Total rows: 9 | Query complete 00:00:00.088

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 freight) as previous_order_freight,
lead(freight) over (partition by customer_id order by freight) as next_order_freight
from orders;

```
21  /* 2.Compare current order's freight with previous and next order for each customer.
22  (Display order_id,  customer_id,  order_date,  freight,
23   Use lead(freight) and lag(freight). */
24
25  select order_id, customer_id, order_date,freight,
26  lag(freight) over (partition by customer_id order by freight) as previous_order_freight,
27  lead(freight) over (partition by customer_id order by freight) as next_order_freight
28  from orders;
29
```

Data Output | Messages | Notifications

	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	10535	ANTON	1997-05-13	15.64	4.03	22
13	10365	ANTON	1996-11-27	22	15.64	36.13
14	10682	ANTON	1997-09-25	36.13	22	47.45
15	10507	ANTON	1997-04-15	47.45	36.13	58.43

Total rows: 830 | Query complete 00:00:00.187

select order_id, customer_id, order_date,freight,
lag(freight) over (partition by customer_id order by order_id) as previous_order_freight,
lead(freight) over (partition by customer_id order by order_id) as next_order_freight
from orders;

```
/* 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_id) as previous_order_freight
lead(freight) over (partition by customer_id order by order_id) as next_order_freight
from orders;
```

Data Output | Messages | Notifications

	order_id [PK] smallint	customer_id character varying (5)	order_date date	freight real	previous_order_freight real	next_order_freight real
1	10643	ALFKI	1997-08-25	29.46	[null]	61.02
2	10692	ALFKI	1997-10-03	61.02	29.46	23.94
3	10702	ALFKI	1997-10-13	23.94	61.02	69.53
4	10835	ALFKI	1998-01-15	69.53	23.94	40.42
5	10952	ALFKI	1998-03-16	40.42	69.53	1.21
6	11011	ALFKI	1998-04-09	1.21	40.42	[null]
7	10308	ANATR	1996-09-18	1.61	[null]	43.9
8	10625	ANATR	1997-08-08	43.9	1.61	11.99
9	10759	ANATR	1997-11-28	11.99	43.9	39.92
10	10926	ANATR	1998-03-04	39.92	11.99	[null]
11	10365	ANTON	1996-11-27	22	[null]	47.45
12	10507	ANTON	1997-04-15	47.45	22	15.64
13	10535	ANTON	1997-05-13	15.64	47.45	84.84
14	10573	ANTON	1997-06-19	84.84	15.64	4.03
15	10677	ANTON	1997-09-22	4.03	84.84	36.13

Total rows: 830 | Query complete 00:00:00.152

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 cte_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
)

--main query
select price_category,
count(*) as product_count,
ROUND(AVG(unit_price)::numeric, 2) as avg_price
from cte_price_category
group by price_category
order by price_category;
```

```sql
/* 3.Show products and their price categories, product c
(HINT: ·Create a CTE which should have price_category de
        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_cou


With cte_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
)

--main query
select price_category,
count(*) as product_count,
ROUND(AVG(unit_price)::numeric, 2) as avg_price
from cte_price_category
group by price_category
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

Total rows: 3 | Query complete 00:00:00.074