# Day 7 Assignment SQL

## 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,
   COUNT(O.order_id) AS total_orders,
   RANK() OVER (ORDER BY COUNT(O.order_id) DESC) AS rank
FROM
   Employees E
LEFT JOIN
   Orders O ON E.employee_id = O.employee_id
GROUP BY
   E.employee_id
ORDER BY
   total_orders DESC;
```

Data Output | Messages | Notifications

| | employee_id [PK] smallint | first_name character varying (10) | last_name character varying (20) | total_orders bigint | rank bigint |
|---|---|---|---|---|---|
| 1 | 4 | Margaret | Peacock | 156 | 1 |
| 2 | 3 | Janet | Leverling | 127 | 2 |
| 3 | 1 | Nancy | Davolio | 123 | 3 |
| 4 | 8 | Laura | Callahan | 104 | 4 |
| 5 | 2 | Andrew | Fuller | 96 | 5 |
| 6 | 7 | Robert | King | 72 | 6 |
| 7 | 6 | Michael | Suyama | 67 | 7 |
| 8 | 9 | Anne | Dodsworth | 43 | 8 |
| 9 | 5 | Steven | Buchanan | 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).**

```
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
ORDER BY
   customer_id,
   order_date;
```

Data Output | Messages | Notifications

| | order_id [PK] smallint | customer_id character varying (5) | order_date date | freight real | previous_freight real | next_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 |

**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 PriceCategories 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(product_id) AS product_count,
  ROUND(AVG(unit_price)::numeric, 2) AS avg_price
FROM
  PriceCategories
GROUP BY
  price_category
ORDER BY
  price_category;
```

Data Output | Messages | Notifications

| | price_category text | product_count bigint | avg_price numeric |
|---|---|---|---|
| 1 | High Price | 6 | 109.13 |
| 2 | Low Price | 34 | 13.30 |
| 3 | Medium Price | 27 | 30.92 |