```
Authors: Ingabire Mugisha Fiston & Muyoboke Emanuele
 Assignment: SQL Window Functions using Product Sales


-- Drop the table if it exists (optional)
BEGIN
    EXECUTE IMMEDIATE 'DROP TABLE product_sales';
EXCEPTION
    WHEN OTHERS THEN NULL;
END;
/

-- Create the product_sales table
CREATE TABLE product_sales (
    sale_id      NUMBER PRIMARY KEY,
    product_name VARCHAR2(50),
    category     VARCHAR2(30),
    region       VARCHAR2(30),
    sale_amount  NUMBER,
    sale_date    DATE
);

-- Insert sample data
INSERT INTO product_sales VALUES (1, 'Notebook',    'Stationery',
'Kigali',     1000, TO_DATE('2023-01-01','YYYY-MM-DD'));
INSERT INTO product_sales VALUES (2, 'Pen',         'Stationery',
'Kigali',      300, TO_DATE('2023-01-03','YYYY-MM-DD'));
INSERT INTO product_sales VALUES (3, 'Eraser',      'Stationery', 'Huye',
150, TO_DATE('2023-01-10','YYYY-MM-DD'));
INSERT INTO product_sales VALUES (4, 'Backpack',    'Accessories','Huye',
2500, TO_DATE('2023-02-01','YYYY-MM-DD'));
INSERT INTO product_sales VALUES (5, 'Ruler',       'Stationery',
'Musanze',     500, TO_DATE('2023-02-05','YYYY-MM-DD'));
INSERT INTO product_sales VALUES (6, 'Water
Bottle','Accessories','Musanze',    1200, TO_DATE('2023-02-10','YYYY-MM-
DD'));
INSERT INTO product_sales VALUES (7, 'Stapler',     'Office',
'Rubavu',     1800, TO_DATE('2023-03-01','YYYY-MM-DD'));
INSERT INTO product_sales VALUES (8, 'Desk Lamp',   'Office',
'Rubavu',     3200, TO_DATE('2023-03-05','YYYY-MM-DD'));
INSERT INTO product_sales VALUES (9, 'Folder',      'Office',
'Kigali',      800, TO_DATE('2023-03-10','YYYY-MM-DD'));

COMMIT;

-- Task 1: Compare Sale Amount with Previous/Next Using LAG() and LEAD()
PROMPT === Task 1: LAG / LEAD Comparison ===
SELECT
    sale_id,
    product_name,
    sale_amount,
    LAG(sale_amount)  OVER (ORDER BY sale_amount) AS previous_sale,
    LEAD(sale_amount) OVER (ORDER BY sale_amount) AS next_sale,
    CASE
        WHEN sale_amount > LAG(sale_amount) OVER (ORDER BY sale_amount)
THEN 'HIGHER'
        WHEN sale_amount < LAG(sale_amount) OVER (ORDER BY sale_amount)
THEN 'LOWER'
```

```
        ELSE 'EQUAL'
    END AS comparison
FROM product_sales;

-- Task 2: RANK vs DENSE_RANK by category
PROMPT === Task 2: RANK vs DENSE_RANK ===
SELECT
    sale_id,
    product_name,
    category,
    sale_amount,
    RANK()       OVER (PARTITION BY category ORDER BY sale_amount DESC)
AS rank_amount,
    DENSE_RANK() OVER (PARTITION BY category ORDER BY sale_amount DESC)
AS dense_rank_amount
FROM product_sales;

-- Task 3: Top 3 Sales per Category
PROMPT === Task 3: Top 3 Sales per Category ===
SELECT *
FROM (
    SELECT
        sale_id,
        product_name,
        category,
        sale_amount,
        RANK() OVER (PARTITION BY category ORDER BY sale_amount DESC) AS
rank_amount
    FROM product_sales
)
WHERE rank_amount <= 3;

-- Task 4: First 2 Sales per Region
PROMPT === Task 4: First 2 Sales per Region ===
SELECT *
FROM (
    SELECT
        sale_id,
        product_name,
        region,
        sale_date,
        ROW_NUMBER() OVER (PARTITION BY region ORDER BY sale_date) AS
row_num
    FROM product_sales
)
WHERE row_num <= 2;

-- Task 5: Window Aggregation
PROMPT === Task 5: Max Sale per Category and Overall ===
SELECT
    sale_id,
    product_name,
    category,
    sale_amount,
    MAX(sale_amount) OVER (PARTITION BY category) AS max_in_category,
    MAX(sale_amount) OVER () AS overall_max
FROM product_sales;
```