SQL Window Functions Assignment

This project showcases how to use SQL window functions — especially LAG() and LEAD() — on four tables: orders, order_items, products, and customers.

orders Table

```
CREATE TABLE orders (
   order_id INT,
   customer_name VARCHAR(50),
   order_date DATE,
   region VARCHAR(50),
   order_amount DECIMAL(10, 2)
);

INSERT INTO orders VALUES
(101, 'Alice', TO_DATE('2023-01-15', 'YYYY-MM-DD'), 'North',
1200.00),
(102, 'Bob', TO_DATE('2023-01-17', 'YYYY-MM-DD'), 'North',
850.00),
(103, 'Charlie', TO_DATE('2023-01-18', 'YYYY-MM-DD'),
'South', 1500.00),
(104, 'Diana', TO_DATE('2023-01-20', 'YYYY-MM-DD'), 'South',
950.00),
(105, 'Ethan', TO_DATE('2023-01-21', 'YYYY-MM-DD'), 'East',
1350.00);
```

---

order_items Table

```
CREATE TABLE order_items (
   item_id INT PRIMARY KEY,
   order_id INT,
```

```
    product_name VARCHAR(50),
    quantity INT,
    unit_price DECIMAL(10, 2)
);

INSERT INTO order_items VALUES
(1, 101, 'Laptop', 1, 800.00),
(2, 101, 'Mouse', 2, 20.00),
(3, 102, 'Keyboard', 1, 30.00),
(4, 103, 'Monitor', 2, 150.00),
(5, 104, 'Desk', 1, 200.00),
(6, 104, 'Chair', 2, 100.00),
(7, 105, 'Laptop', 1, 850.00),
(8, 105, 'Mouse', 1, 25.00);
```

---

products Table

```
CREATE TABLE products (
    product_id INT,
    product_name VARCHAR(50),
    category VARCHAR(50),
    price DECIMAL(10, 2),
    sales_count INT
);

INSERT INTO products VALUES
(1, 'Laptop', 'Electronics', 1000.00, 300),
(2, 'Mouse', 'Electronics', 25.00, 500),
(3, 'Chair', 'Furniture', 150.00, 200),
(4, 'Desk', 'Furniture', 250.00, 150),
(5, 'Monitor', 'Electronics', 200.00, 250),
(6, 'Keyboard', 'Electronics', 45.00, 350);
```

---

customers Table

```
CREATE TABLE customers (
   customer_id INT,
   customer_name VARCHAR(50),
   city VARCHAR(50),
   total_spent DECIMAL(10, 2),
   join_date DATE
);

INSERT INTO customers VALUES
(1, 'Alice', 'New York', 500.00, TO_DATE('2020-03-15', 'YYYY-MM-DD')),
(2, 'Bob', 'New York', 750.00, TO_DATE('2019-06-21', 'YYYY-MM-DD')),
(3, 'Charlie', 'Los Angeles', 900.00, TO_DATE('2021-01-10', 'YYYY-MM-DD')),
(4, 'Diana', 'Los Angeles', 1100.00, TO_DATE('2018-12-05', 'YYYY-MM-DD')),
(5, 'Ethan', 'Chicago', 650.00, TO_DATE('2020-09-30', 'YYYY-MM-DD')),
(6, 'Fiona', 'Chicago', 870.00, TO_DATE('2019-04-18', 'YYYY-MM-DD')),
(7, 'George', 'Chicago', 790.00, TO_DATE('2022-03-11', 'YYYY-MM-DD')),
(8, 'Hannah', 'New York', 1200.00, TO_DATE('2018-08-25', 'YYYY-MM-DD'));
```

LAG and LEAD on orders

```
SELECT
   order_id,
   customer_name,
   region,
```

```
    order_amount,
    LAG(order_amount) OVER (PARTITION BY region ORDER
BY order_date) AS prev_order_amount,
    LEAD(order_amount) OVER (PARTITION BY region ORDER
BY order_date) AS next_order_amount
FROM orders;
```

LAG and LEAD on order_items

```
SELECT
    item_id,
    product_name,
    quantity,
    unit_price,
    (quantity * unit_price) AS total_price,
    LAG(quantity * unit_price) OVER (PARTITION BY
product_name ORDER BY item_id) AS prev_price,
    LEAD(quantity * unit_price) OVER (PARTITION BY
product_name ORDER BY item_id) AS next_price
FROM order_items;
```

LAG and LEAD on products

```
SELECT
    product_id,
    product_name,
    category,
    sales_count,
    LAG(sales_count) OVER (PARTITION BY category ORDER
BY sales_count) AS prev_sales,
    LEAD(sales_count) OVER (PARTITION BY category ORDER
BY sales_count) AS next_sales
```

```
FROM products;
```

LAG and LEAD on customers

```
SELECT
   customer_id,
   customer_name,
   city,
   total_spent,
   LAG(total_spent) OVER (PARTITION BY city ORDER BY
total_spent) AS prev_spent,
   LEAD(total_spent) OVER (PARTITION BY city ORDER BY
total_spent) AS next_spent
FROM customers;
```
