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;