University Institute of Engineering

Department of Computer Science & Engineering

EXPERIMENT: 6

NAME : Johnson Kumar **UID** : 23BCS12654

SECTION : KRG_2A **SEMESTER**: 5TH

SUBJECT CODE: 23CSP-339 **SUBJECT** : ADBMS

Problem Statement:

SmartShop is a modern retail company that sells electronic gadgets like smartphones, tablets, and laptops.

The company wants to automate its ordering and inventory management process. Whenever a customer places an order, the system must:

1. Verify stock availability for the requested product and quantity.
2. If sufficient stock is available:
   - Log the order in the sales table with the ordered quantity and total price.
   - Update the inventory in the products table by reducing quantity_remaining and increasing quantity_sold.
   - Display a real-time confirmation message: "Product sold successfully!“
3. If there is insufficient stock, the system must:
   - Reject the transaction and display: Insufficient Quantity Available!"

Solution:

```
------------------------HARD LEVEL-----------------------------
CREATE TABLE products (
    product_code VARCHAR(10) PRIMARY KEY,
    product_name VARCHAR(100) NOT NULL,
    price NUMERIC(10,2) NOT NULL,
    quantity_remaining INT NOT NULL,
    quantity_sold INT DEFAULT 0
);
CREATE TABLE sales (
    order_id SERIAL PRIMARY KEY,
    order_date DATE NOT NULL,
    product_code VARCHAR(10) NOT NULL,
```

```
   quantity_ordered INT NOT NULL,
   sale_price NUMERIC(10,2) NOT NULL,
   FOREIGN KEY (product_code) REFERENCES products(product_code)
);

INSERT INTO products (product_code, product_name, price, quantity_remaining, quantity_sold)
VALUES
('P001', 'iPHONE 13 PRO MAX', 109999.00, 10, 0),
('P002', 'Samsung Galaxy S23 Ultra', 99999.00, 8, 0),
('P003', 'iPAD AIR', 55999.00, 5, 0),
('P004', 'MacBook Pro 14"', 189999.00, 3, 0),
('P005', 'Sony WH-1000XM5 Headphones', 29999.00, 15, 0);

INSERT INTO sales (order_date, product_code, quantity_ordered, sale_price)
VALUES
('2025-09-15', 'P001', 1, 109999.00),
('2025-09-16', 'P002', 2, 199998.00),
('2025-09-17', 'P003', 1, 55999.00),
('2025-09-18', 'P005', 2, 59998.00),
('2025-09-19', 'P001', 1, 109999.00);

SELECT * FROM PRODUCTS;
SELECT * FROM SALES;

CREATE OR REPLACE PROCEDURE pr_buy_products(
   IN p_product_name VARCHAR,
   IN p_quantity INT
)
LANGUAGE plpgsql
AS $$
DECLARE
   v_product_code VARCHAR(20);
   v_price FLOAT;
   v_count INT;
BEGIN

   SELECT COUNT(*)
   INTO v_count
   FROM products
   WHERE product_name = p_product_name
```

```
    AND quantity_remaining >= p_quantity;

    IF v_count > 0 THEN
        SELECT product_code, price
        INTO v_product_code, v_price
        FROM products
        WHERE product_name = p_product_name;

        INSERT INTO sales (order_date, product_code, quantity_ordered, sale_price)
        VALUES (CURRENT_DATE, v_product_code, p_quantity, (v_price * p_quantity));
        UPDATE products
        SET quantity_remaining = quantity_remaining - p_quantity,
            quantity_sold = quantity_sold + p_quantity
        WHERE product_code = v_product_code;

        RAISE NOTICE 'PRODUCT SOLD..! Order placed successfully for % unit(s) of %.', p_quantity,
        p_product_name;

    ELSE
      RAISE NOTICE 'INSUFFICIENT QUANTITY..! Order cannot be processed for % unit(s) of
      %.', p_quantity, p_product_name;
    END IF;
END;
$$;
CALL pr_buy_products ('MacBook Pro 14"', 1);
```

Output:

Data Output | Messages | Notifications

```
NOTICE:  PRODUCT SOLD..! Order placed successfully for 1 unit(s) of MacBook Pro 14".
CALL

Query returned successfully in 73 msec.
```

University Institute of Engineering

Department of Computer Science & Engineering

Learning outcomes:

- ✓ I learned how to implement conditional flow in SQL procedures to verify stock availability before processing an order, ensuring business rules are enforced.
- ✓ I applied multiple updates (sales logging and inventory adjustment) within a single procedure, reinforcing the concept of atomicity to prevent partial transactions.
- ✓ I practiced updating product records by reducing quantity_remaining and increasing quantity_sold, simulating real-time inventory tracking.
- ✓ I inserted order details into the sales table, dynamically calculating total_price based on quantity and unit price, which supports accurate financial reporting.
- ✓ I used output messages like "Product sold successfully!" or "Insufficient Quantity Available!" to provide immediate, user-friendly transaction feedback.
- ✓ I translated a retail workflow into a modular, reusable stored procedure that aligns with SmartShop's operational goals—automating order validation and inventory updates.