EXPERIMENT - 6

Student Name: Tanay Manish Nesari **UID: 23BCS13761**
Branch: BE-CSE **Section/Group: KRG 1-B**
Semester: 5th **Date of Performance: 23/09/2025**
Subject Name: ADBMS **Subject Code: 23CSP-333**

Aim: To understand and implement Stored procedures.

Objective:

1) HR-Analytics: Employee count based on dynamic gender passing (Medium)
TechSphere Solutions, a growing IT services company with offices across India, wants to track and monitor gender diversity within its workforce. The HR department frequently needs to know the total number of employees by gender (Male or Female) .
To solve this problem, the company needs an automated database-driven solution that can instantly return the count of employees by gender through a stored procedure that:
1. Create a PostgreSQL stored procedure that:
2. Takes a gender (e.g., 'Male' or 'Female') as input.
3. Calculates the total count of employees for that gender.
4. Returns the result as an output parameter.
5. Displays the result clearly for HR reporting purposes.

2) SmartStore Automated Purchase System (Hard)
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!"

Code:

1)
```
drop table if exists employee;

CREATE TABLE Employee (
   id int PRIMARY KEY,
   name VARCHAR(50) NOT NULL,
   gender VARCHAR(10) NOT NULL,
   salary NUMERIC(10,2) NOT NULL,
   city VARCHAR(50) NOT NULL
);

INSERT INTO Employee (id, name, gender, salary, city)
VALUES
(10, 'Karan', 'Male', 70000.00, 'Gurgaon'),
(11, 'Seema', 'Female', 75000.00, 'Mumbai'),
(12, 'Zahra', 'Female', 65000.00, 'Hyderabad'),
(13, 'Leo', 'Other', 50000.00, 'Bangalore'),
(14, 'Raj', 'Male', 68000.00, 'Delhi'),
(15, 'Sonia', 'Female', 72000.00, 'Chennai');


create or replace procedure count_employee(in gender varchar(50), out tCount int)
language plpgsql as
$$
        Begin
        select count(*) into tCount
        from
        Employee as e
        where e.gender = count_employee.gender;

        raise notice 'Total number of % employees: %',gender,tCount;
        end;
$$;

call count_employee('Female',0);
call count_employee('Other',0);
```

2)

```
drop table if exists sales;
drop table if exists products;

CREATE TABLE products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100) NOT NULL,
    unit_price NUMERIC(10, 2) NOT NULL,
    quantity_remaining INT NOT NULL DEFAULT 0 CHECK (quantity_remaining >= 0),
    quantity_sold INT NOT NULL DEFAULT 0 CHECK (quantity_sold >= 0)
);

CREATE TABLE sales (
    sale_id SERIAL PRIMARY KEY,
    product_id INT REFERENCES products(product_id),
    order_quantity INT NOT NULL,
    total_price NUMERIC(10, 2) NOT NULL,
    sale_timestamp TIMESTAMP WITHOUT TIME ZONE DEFAULT NOW()
);

INSERT INTO products (product_id, product_name, unit_price, quantity_remaining, quantity_sold) VALUES
(2001, 'Bluetooth Headset', 49.99, 100, 20),
(2002, 'Gaming Mouse', 25.00, 30, 5),
(2003, 'External HDD 1TB', 60.00, 15, 3);


CREATE OR REPLACE FUNCTION process_sale(
    p_product_id INT,
    p_order_quantity INT
)
RETURNS TEXT
```

```
LANGUAGE plpgsql
AS $$
DECLARE
    v_unit_price NUMERIC(10, 2);
    v_total_price NUMERIC(10, 2);
    v_stock_available INT;
BEGIN
    SELECT unit_price, quantity_remaining INTO v_unit_price, v_stock_available
    FROM products
    WHERE product_id = p_product_id
    FOR UPDATE;

    IF NOT FOUND THEN
        RETURN 'Error: Product ID not found!';
    END IF;

    IF v_stock_available < p_order_quantity THEN
        RAISE EXCEPTION 'Insufficient Quantity Available! Requested: %; Available: %', p_order_quantity,
v_stock_available;
    ELSE
        v_total_price := v_unit_price * p_order_quantity;

        INSERT INTO sales (product_id, order_quantity, total_price)
        VALUES (p_product_id, p_order_quantity, v_total_price);

        UPDATE products
        SET
            quantity_remaining = quantity_remaining - p_order_quantity,
            quantity_sold = quantity_sold + p_order_quantity
        WHERE
            product_id = p_product_id;

        RETURN 'Product sold successfully!';
    END IF;
END;
$$;

SELECT process_sale(2001, 10) AS Transaction_Result;

SELECT process_sale(2003, 20) AS Transaction_Result;

SELECT product_id, quantity_remaining, quantity_sold FROM products WHERE product_id IN (2001, 2003);
```