Worksheet 6

Student Name: Shivanshu Ranjan **UID: 23BCS10193**
Branch: CSE **Section/Group: KRG 3-A**
Semester: 5th **Date of Performance:25/09/2025**
Subject Name: ADBMS **Subject Code: 23CSP-333**

1. **Aim: 1.** 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:

   a) Create a PostgreSQL stored procedure that:
   b) Takes a **gender** (e.g., 'Male' or 'Female') as input.
   c) Calculates the **total count of employees** for that gender.
   d) Returns the result as an **output parameter**.

2. 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:
   i. **Verify stock availability** for the requested product and quantity.

   ii. 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!“
   iii. If there is **insufficient stock**, the system must:

   - **Reject the transaction** and display: Insufficient Quantity Available!"

2. Objective:

- To design database-driven stored procedures that automate business processes and reduce manual effort.
- To provide accurate and real-time information (such as employee gender count or stock availability) for better decision-making.

- To ensure efficient handling of company operations like HR diversity tracking and retail order management.
- To enhance user experience by offering instant responses, whether in reporting (gender diversity) or transaction processing (order confirmation/rejection).

3. Code:

1.

```
CREATE TABLE employees (
    emp_id SERIAL PRIMARY KEY,
    emp_name VARCHAR(100),
    gender VARCHAR(10)
);
INSERT INTO employees (emp_name, gender) VALUES
('Shivanshu Ranjan', 'Male'),
('Tanya Verma', 'Female'),
('Alok Kumar', 'Male'),
('Neha Singh', 'Female'),
('Devanshu Ranjan', 'Male');
-- Stored procedure
CREATE OR REPLACE PROCEDURE get_employee_count_by_gender(IN input_gender
VARCHAR,OUT gender_count INT)
LANGUAGE plpgsql
AS
$$
BEGIN
    SELECT COUNT(*)
    INTO gender_count
```

```
    FROM employees
    WHERE LOWER(gender) = LOWER(input_gender);
RAISE NOTICE 'Total employees with gender % are: %', input_gender, gender_count;
END;
$$;
CALL get_employee_count_by_gender('Male', NULL); -- Call for Male
CALL get_employee_count_by_gender('Female', NULL); -- Call for Female
```

2.

```
-- SmartStore Automated Purchase System (Hard Level)
CREATE TABLE products (
    product_id SERIAL PRIMARY KEY,
    product_name VARCHAR(100),
    unit_price NUMERIC(10,2),
    quantity_remaining INT,
    quantity_sold INT DEFAULT 0
);
CREATE TABLE sales (
    sale_id SERIAL PRIMARY KEY,
    product_id INT REFERENCES products(product_id),
    quantity INT,
    total_price NUMERIC(10,2),
    sale_date TIMESTAMP DEFAULT NOW()
);
INSERT INTO products (product_name, unit_price, quantity_remaining)
VALUES
```

```
('Smartphone', 25000, 10),
('Tablet', 18000, 5),
('Laptop', 55000, 3);
CREATE OR REPLACE PROCEDURE process_order(IN p_product_id INT,IN p_quantity INT)
LANGUAGE plpgsql
AS $$
DECLARE
    available_qty INT;
    product_price NUMERIC(10,2);
    total NUMERIC(10,2);
BEGIN
    -- Get available stock and price
    SELECT quantity_remaining, unit_price
    INTO available_qty, product_price
    FROM products
    WHERE product_id = p_product_id;
  -- If no product found
    IF available_qty IS NULL THEN
        RAISE NOTICE 'Product not found!';
        RETURN;
    END IF;
-- Check stock availability
    IF available_qty >= p_quantity THEN
        -- Calculate total price
        total := product_price * p_quantity;
```

```
        -- Log the order in sales
        INSERT INTO sales(product_id, quantity, total_price)
        VALUES (p_product_id, p_quantity, total);
 -- Update inventory
        UPDATE products
        SET quantity_remaining = quantity_remaining - p_quantity,
            quantity_sold = quantity_sold + p_quantity
        WHERE product_id = p_product_id;
-- Confirmation message
        RAISE NOTICE 'Product sold successfully!';
    ELSE
        -- Reject order
        RAISE NOTICE 'Insufficient Quantity Available!';
    END IF;
END;
$$;
select * from products;
CALL process_order(1, 2);
select * from products;
CALL process_order(3, 10);
-- select * from products;
```

4. Output:

```
 gender_count
--------------
            3
(1 row)

 gender_count
--------------
            2
(1 row)


psql:commands.sql:33: NOTICE:  Total employees with gender Male are: 3
psql:commands.sql:35: NOTICE:  Total employees with gender Female are: 2
```

(1)

```
 product_id | product_name | unit_price | quantity_remaining | quantity_sold
------------+--------------+------------+--------------------+---------------
          1 | Smartphone   |   25000.00 |                 10 |             0
          2 | Tablet       |   18000.00 |                  5 |             0
          3 | Laptop       |   55000.00 |                  3 |             0
(3 rows)

CALL
 product_id | product_name | unit_price | quantity_remaining | quantity_sold
------------+--------------+------------+--------------------+---------------
          2 | Tablet       |   18000.00 |                  5 |             0
          3 | Laptop       |   55000.00 |                  3 |             0
          1 | Smartphone   |   25000.00 |                  8 |             2
(3 rows)

CALL

psql:commands.sql:74: NOTICE:  Product sold successfully!
psql:commands.sql:77: NOTICE:  Insufficient Quantity Available!
```

(2)

5. Learning Outcomes:

- Students will be able to design and implement **stored procedures** in PostgreSQL for automating organizational tasks.
- Learners will understand how to **use input and output parameters** in stored procedures for dynamic queries.
- They will gain hands-on experience in **real-time business applications** like HR diversity tracking and retail inventory/order management.
- They will be able to apply **transactional logic with conditions** (e.g., stock verification, sales updates) to ensure data integrity and efficiency.