Experiment-6

Student Name: Manjot Singh **UID:** 23BCS12549
Branch: CSE **Section/Group:** KRG-2B
Semester: 5th **Date of Performance:** 22/08/25
Subject Name: ADBMS **Subject Code:** 23CSP-333

1. Aim:

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.

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!"

2. Objective:

- Insert a new record into the sales table.
- The final output should clearly display the count for HR reporting.
- The result should be returned as an output parameter
- If stock is insufficient, it should Display the message: "Insufficient Quantity Available!"
- Use transactions to ensure data integrity.
- It should calculate the total number of employees for the specified gender.

3. DBMS Script :

Script 1:

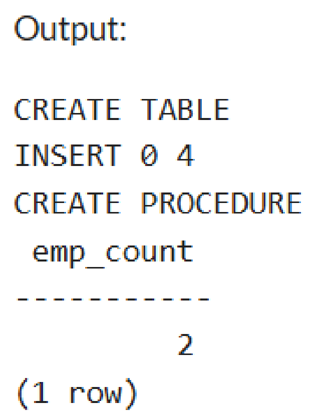
```
CREATE TABLE employees (
   emp_id SERIAL PRIMARY KEY,
   emp_name VARCHAR(50),
   gender VARCHAR(10)
);

INSERT INTO employees (emp_name, gender) VALUES
('John', 'Male'),
('Alice', 'Female'),
('Robert', 'Male'),
('Sophia', 'Female');

CREATE OR REPLACE PROCEDURE get_employee_count_by_gender(
   IN input_gender VARCHAR,
   OUT emp_count INT
)
LANGUAGE plpgsql
AS $$
BEGIN
   SELECT COUNT(*) INTO emp_count FROM employees WHERE gender = input_gender;
END;
$$;

CALL get_employee_count_by_gender('Male', emp_count);
```

Script 2:

```
CREATE TABLE products (
   product_id SERIAL PRIMARY KEY,
   product_name VARCHAR(50),
   price DECIMAL(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 DECIMAL(10,2)
);

CREATE OR REPLACE PROCEDURE process_order(
    IN p_product_id INT,
    IN p_quantity INT
)
LANGUAGE plpgsql
AS $$
DECLARE
    available_qty INT;
    unit_price DECIMAL(10,2);
BEGIN
    SELECT quantity_remaining, price INTO available_qty, unit_price
    FROM products WHERE product_id = p_product_id;

    IF available_qty >= p_quantity THEN
        INSERT INTO sales(product_id, quantity, total_price)
        VALUES (p_product_id, p_quantity, unit_price * p_quantity);

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

        RAISE NOTICE 'Product sold successfully!';
    ELSE
        RAISE NOTICE 'Insufficient Quantity Available!';
    END IF;
END;
$$;

CALL process_order(1, 2);
```

4. Output:

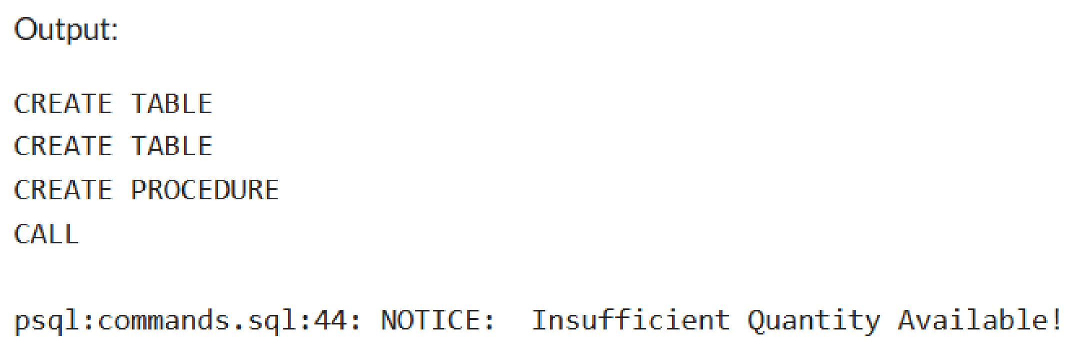
Output 1:



Output:

```
CREATE TABLE
INSERT 0 4
CREATE PROCEDURE
 emp_count
-----------
         2
(1 row)
```

Output 2:



Output:

```
CREATE TABLE
CREATE TABLE
CREATE PROCEDURE
CALL

psql:commands.sql:44: NOTICE:  Insufficient Quantity Available!
```

5. Learning Outcomes:

- Successfully implemented sub-queries to extract top salary earners by department.
- Successfully implemented stored procedures in PostgreSQL.
- Practiced handling input and output parameters in procedures.
- Automated HR analytics queries for gender-based employee counts.
- Developed an order-processing system with real-time stock validation.
- Enhanced SQL procedural programming skills for enterprise applications.