**Experiment-5**

|  |  |
| --- | --- |
| **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!"

1. **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.
2. **DBMS Script :**

**Script 1:**

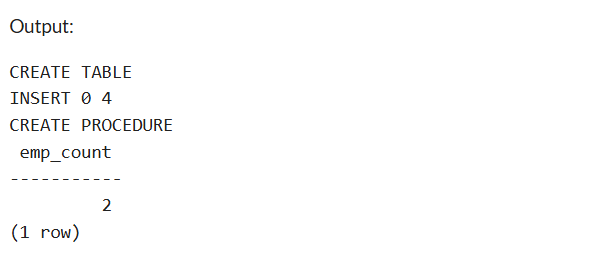
CREATE TABLE employees (  
 emp\_id SERIAL PRIMARY KEY,  
 emp\_name VARCHAR(50),  
 gender VARCHAR(10)  
);  
  
INSERT INTO employees (emp\_name, gender) VALUES  
('John', 'Male'),  
('Alice', 'Female'),  
('Robert', 'Male'),  
('Sophia', 'Female');  
  
CREATE OR REPLACE PROCEDURE get\_employee\_count\_by\_gender(  
 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:**

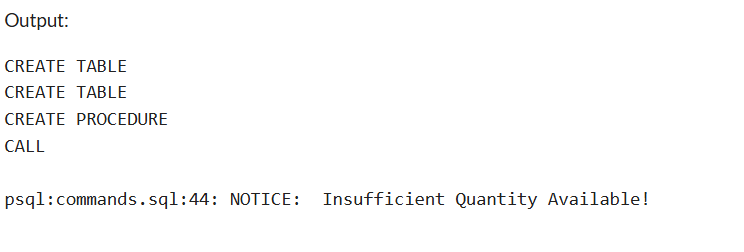
CREATE TABLE products (  
 product\_id SERIAL PRIMARY KEY,  
 product\_name VARCHAR(50),  
 price DECIMAL(10,2),  
 quantity\_remaining INT,  
 quantity\_sold INT DEFAULT 0  
);  
CREATE TABLE sales (  
 sale\_id SERIAL PRIMARY KEY,  
 product\_id INT REFERENCES products(product\_id),  
 quantity INT,  
 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);

1. **Output:**

**Output 1:**



**Output 2:**



1. **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.