

# **University Institute of Engineering**

## **Department of Computer Science & Engineering**

### **EXPERIMENT: 6**

NAME: AAYUSH AMAN UID: 23BCS14046

BRANCH: BE-CSE SECTION / GROUP: KRG 3B

SEMESTER: 5<sup>TH</sup> SUBJECT CODE: 23CSP-339

SUBJECT NAME: ADBMS

## 1. Aim of the practical:

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. Takes a gender (e.g., 'Male' or 'Female') as input.
- 2. Calculates the total count of employees for that gender.
- 3. Returns the result as an output parameter.
- 4. Displays the result clearly for HR reporting purposes.

SmartStore Automated Purchase System [HARD]

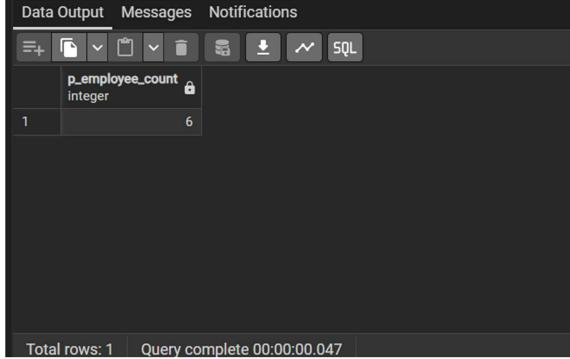
SmartShop is a modern retail company that sells electronic gadgets like smartphones, tablets, and laptops.

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!"
- 5. Tools Used: PgAdmin, PostgresSQL

### 6. Code:

```
-- Database: krg 3b
--- Medium
CREATE TABLE employee info (
  id SERIAL 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 info (name, gender, salary, city)
VALUES
('Alok', 'Male', 50000.00, 'Delhi'),
('Priya', 'Male', 60000.00, 'Mumbai'),
('Rajesh', 'Female', 45000.00, 'Bangalore'),
('Sneha', 'Male', 55000.00, 'Chennai'),
('Anil', 'Male', 52000.00, 'Hyderabad'),
('Sunita', 'Female', 48000.00, 'Kolkata'),
('Vijay', 'Male', 47000.00, 'Pune'),
('Ritu', 'Male', 62000.00, 'Ahmedabad'),
('Amit', 'Female', 51000.00, 'Jaipur');
--SOLUTION
CREATE OR REPLACE PROCEDURE sp get employees by gender(
  IN p gender VARCHAR(50),
  OUT p employee count INT
)
LANGUAGE plpgsql
AS $$
BEGIN
  SELECT COUNT(id)
  INTO p employee count
  FROM employee info
  WHERE gender = p gender;
  RAISE NOTICE 'Total employees with gender %: %', p gender, p employee count;
END;
$$;
CALL sp get employees by gender('Male', NULL);
```

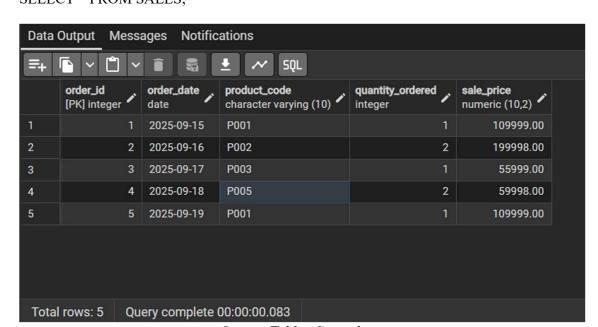


Output: Medium level question

#### --- Hard

```
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;
```



Output: Tables Created

---SOLUTION:

```
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
  -- Step 1: Check if product exists and has enough quantity
  SELECT COUNT(*)
 INTO v count
 FROM products
  WHERE product name = p product name
  AND quantity remaining >= p quantity;
```

```
-- Step 2: If sufficient stock
  IF v count > 0 THEN
    -- Fetch product code and price
    SELECT product code, price
    INTO v_product_code, v price
    FROM products
    WHERE product name = p product name;
    -- Insert a new record into the sales table
    INSERT INTO sales (order date, product code, quantity ordered, sale price)
    VALUES (CURRENT DATE, v product code, p quantity, (v price *
p quantity));
    -- Update stock details
    UPDATE products
    SET quantity remaining = quantity remaining - p quantity,
       quantity sold = quantity sold + p quantity
    WHERE product code = v product code;
    -- Confirmation message
    RAISE NOTICE 'PRODUCT SOLD..! Order placed successfully for % unit(s) of
%.', p quantity, p product name;
  ELSE
    -- Step 3: If stock is insufficient
    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);
```

```
Data Output Messages Notifications

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

CALL

Query returned successfully in 56 msec.
```

## 7. Learning Outcomes:

- 1. Learned how to create and invoke stored procedures using PL/pgSQL in PostgreSQL.Practiced using input and output parameters to dynamically filter and return results.
- 2. Implemented business logic for inventory validation and transactional updates.
- 3. Gained experience with conditional control flow using IF statements inside procedures.
- 4. Understood how to use RAISE NOTICE for user-friendly feedback during procedure execution.