



EXPERIMENT- 06

Student Name: Milan Kumar

UID: 23bcs14208

Branch: BE-CSE

**Section/Group: KRG 3-AG
3(A)**

Semester: 05

Subject Name: ADBMS

Date of Performance: 23/09/25

Subject Code: 23CSP-333

HR-Analytics: Employee count based on dynamic gender passing (Medium Level)

1. Aim:

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:

- i. Create a PostgreSQL stored procedure that:
- ii. Takes a gender (e.g., 'Male' or 'Female') as input.
- iii. Calculates the total count of employees for that gender.
- iv. Returns the result as an output parameter.
- v. Displays the result clearly for HR reporting purposes.

2. Objective:

- To understand how to create and use stored procedures in PostgreSQL for real-world business requirements.
- To learn how to use input and output parameters in a stored procedure for dynamic data retrieval.
- To implement a database-driven solution that automatically calculates and returns employee count by gender for HR reporting and decision-making.

3. DBMS script and output:

```
-- Create employee table
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
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 sample data

```
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');
```

-- Create procedure to count employees by gender

```
CREATE OR REPLACE PROCEDURE sp_get_employees_by_gender(  
    IN p_gender VARCHAR(50),  
    OUT p_employee_count INT  
)  
LANGUAGE plpgsql  
AS $$  
  
BEGIN  
    -- Count employees of given gender  
    SELECT COUNT(id)
```

```
INTO      p_employee_count
FROM employee_info

WHERE gender = p_gender;
-- Display result
RAISE NOTICE 'Total employees with gender %: %', p_gender, p_employee_count;

END;

$$;

-- Call procedure
CALL sp_get_employees_by_gender('Male', NULL);
```

4. Output:

		
	p_employee_count integer	
1		6

SmartStore Automated Purchase System (Hard Level)

1. Aim:

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 and implement a **stored procedure** that automates order processing and inventory management.
- To ensure **real-time validation of stock availability** before completing a sales transaction.
- To provide **automated feedback messages** confirming successful sales or notifying insufficient stock for better customer experience.

3. DBMS script and output:

-- Create products table

```
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 sales table

```
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 sample products

```
INSERT INTO products (product_code, product_name, price, quantity_remaining, quantity_sold)
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

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 sample sales

```
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);
```

-- View tables

```
SELECT * FROM products;
```

```
SELECT * FROM sales;
```

-- Create procedure to buy products

```
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
```

-- Check if product exists and has enough stock

```
SELECT COUNT(*)  
INTO v_count  
FROM products  
WHERE product_name = p_product_name  
AND quantity_remaining >= p_quantity;
```

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;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
-- Insert new sale
INSERT INTO sales (order_date, product_code, quantity_ordered, sale_price) VALUES (CURRENT_DATE,
v_product_code, p_quantity, (v_price * p_quantity));

-- Update product stock
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
  -- Not enough stock
  RAISE NOTICE 'INSUFFICIENT QUANTITY..! Order cannot be processed for % unit(s) of %.', p_quantity,
p_product_name;
END IF;
END;
$$;

-- Call procedure to buy product
CALL pr_buy_products('MacBook Pro 14"', 1);
```

5. Output:

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

```
Query returned successfully in 241 msec.
```