

**Experiment - 6**

**Student Name:** Priyanka **UID:** 23BAI70303

**Branch:** BE-AIT-CSE **Section/Group:** 23AIT\_KRG-G1\_A

**Semester:** 5th **Date of Performance:** 24 Sept, 2025

**Subject Name:** ADBMS **Subject Code:** 23CSP-333

## Aim:

### MEDIUM LEVEL PROBLEM:

#### HR ANALYTICS:

To create a PostgreSQL stored procedure that dynamically counts the total number of

employees based on a given gender. This allows HR departments to instantly generate reports on workforce diversity and track gender representation efficiently.

### HARD LEVEL PROBLEM:

#### SMARTSTORE AUTOMATED PURCHASE SYSTEM:

To automate product ordering and inventory management in a retail database. The procedure ensures stock validation before processing orders, updates inventory accurately, logs sales transactions, and provides real-time confirmation messages to customers.

## Objective:

#### For HR Analytics:

* + Learn how to define and execute stored procedures in PostgreSQL.
  + Enable dynamic input handling to count employees by gender.
  + Provide HR with instant and accurate workforce analytics.
  + Understand the use of IN and OUT parameters and result display using RAISE NOTICE.

#### For SmartStore System:

* + Implement database-driven automation for retail operations.
  + Check product stock availability before order processing.
  + Update inventory (quantity\_remaining, quantity\_sold) correctly to prevent errors.
  + Log transactions in a sales table for accountability.
  + Provide feedback messages to users in real-time to improve the ordering experience.

## Theory:

#### Stored Procedures

A **stored procedure** is a precompiled set of SQL statements stored in the database that can perform operations like querying, updating, or inserting data. Advantages include:

* + Reusability: The procedure can be executed multiple times without rewriting SQL queries.
  + Security: Users can execute procedures without direct access to tables.
  + Efficiency: Reduces network traffic and increases performance by executing multiple SQL statements as one unit.

#### Input and Output Parameters

* + **IN parameter:** Accepts input data from the user (e.g., gender, product\_id).
  + **OUT parameter:** Returns output data after processing (e.g., total employee count).

#### RAISE NOTICE

* + A PostgreSQL command used to display messages during procedure execution.
  + Useful for logging information or providing real-time feedback without writing to a table.

#### Application in HR Analytics

* + HR often needs quick insights into workforce demographics.
  + A stored procedure with a gender parameter avoids repetitive query writing and allows for

#### dynamic reporting.

1. **Application in Retail Automation**
   * SmartShop wants **real-time automation** in sales and inventory.
   * The stored procedure validates stock before processing the order:
     + If sufficient: logs sale, updates inventory, displays confirmation.
     + If insufficient: rejects the order and shows an error.
   * This ensures **data integrity**, **avoids overselling**, and enhances **customer satisfaction**.

#### Transactions

* + Ensures that inventory updates and sales logging occur as a single atomic operation.
  + If any step fails, the database rolls back changes to maintain consistency.

## Procedure:

### Medium Level Solution:

* **Setup:** Create an employee\_info table and populate it with sample data, including employee names, genders, and other details.
* **Procedure Creation:** Develop a stored procedure named

sp\_get\_employees\_by\_gender. This procedure takes a gender as an input parameter and an integer output parameter.

* **Business Logic:** Inside the procedure, a SELECT COUNT query counts all employees that match the input gender. The result is then stored in the output parameter.
* **Execution:** The procedure is called with a specific gender value (e.g., 'Male'), and a RAISE NOTICE command is used to print the final count, demonstrating a simple yet powerful automated reporting feature.

### Hard Level Solution:

* **Setup:** Establish a database schema with products and sales tables to represent inventory and order history, respectively. Insert sample data into both tables.
* **Procedure Creation:** Create a stored procedure named pr\_buy\_products that accepts the product name and quantity as input.
* **Transactional Logic:** The procedure first checks if the requested quantity is available in the products table.

#### Conditional Processing:

* **If sufficient stock:** The procedure executes a series of steps within a transaction: it inserts a new record into the sales table, updates the products table to reflect the

reduced inventory (quantity\_remaining) and increased sales (quantity\_sold), and then prints a success message.

* **If insufficient stock:** The procedure immediately prints an "INSUFFICIENT QUANTITY" message without logging a sale or altering the inventory tables.
* **Execution:** Test the procedure with different values to demonstrate both a successful sale (when sufficient stock is available) and a failed transaction (when the quantity is too high), showcasing its transactional integrity and errorhandling capabilities.

1. **Code:**

# MEDIUM PROBLEM

CREATE TABLE Employeess (

emp\_id SERIAL PRIMARY KEY,

emp\_name VARCHAR(50),

gender VARCHAR(10)

);

INSERT INTO Employeess (emp\_name, gender) VALUES

('Alice', 'Female'),

('Bob', 'Male'),

('Charlie', 'Male'),

('Diana', 'Female');

drop procedure if exists get\_employee\_count\_by\_gender

CREATE OR REPLACE PROCEDURE get\_employee\_count\_by\_gender(

IN input\_gender VARCHAR,

OUT total\_count INT

)

LANGUAGE plpgsql

AS $$

BEGIN

SELECT COUNT(\*)

INTO total\_count

FROM Employeess

WHERE gender = input\_gender;

RAISE NOTICE 'Gender: %, Total Employees: %', input\_gender, total\_count;

END;

$$;

CALL get\_employee\_count\_by\_gender('Male', total\_count => 0);

CALL get\_employee\_count\_by\_gender('Female', total\_count => 0);

# HARD PROBLEM

CREATE TABLE Items (

item\_id SERIAL PRIMARY KEY,

item\_name VARCHAR(50),

price DECIMAL(10,2),

quantity\_remaining INT,

quantity\_sold INT DEFAULT 0

);

-- Orders Table

CREATE TABLE Orders (

order\_id SERIAL PRIMARY KEY,

item\_id INT REFERENCES Items(item\_id),

quantity\_ordered INT,

total\_price DECIMAL(10,2),

order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE OR REPLACE PROCEDURE process\_order(

IN input\_item\_id INT,

IN input\_quantity INT

)

LANGUAGE plpgsql

AS $$

DECLARE

available\_stock INT;

item\_price DECIMAL(10,2);

total\_cost DECIMAL(10,2);

BEGIN

-- Check available stock and price

SELECT quantity\_remaining, price

INTO available\_stock, item\_price

FROM Items

WHERE item\_id = input\_item\_id;

-- If enough stock available

IF available\_stock >= input\_quantity THEN

total\_cost := item\_price \* input\_quantity;

-- Insert order into Orders table

INSERT INTO Orders (item\_id, quantity\_ordered, total\_price)

VALUES (input\_item\_id, input\_quantity, total\_cost);

-- Update inventory in Items table

UPDATE Items

SET quantity\_remaining = quantity\_remaining - input\_quantity,

quantity\_sold = quantity\_sold + input\_quantity

WHERE item\_id = input\_item\_id;

RAISE NOTICE 'Product sold successfully!';

ELSE

RAISE NOTICE 'Insufficient Quantity Available!';

END IF;

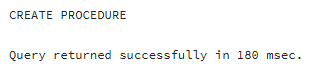
END;

$$;

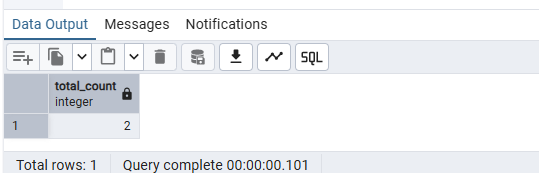
CALL process\_order(1, 2);

## Output:

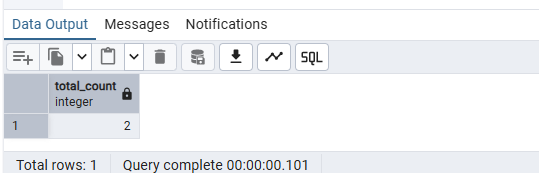
## Medium:



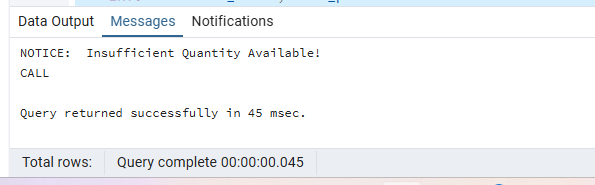
For males:



For Females:



**Hard:**

****

1. **Learning Outcomes:**

#### Stored Procedure Implementation:

* + Learned how to create, execute, and manage stored procedures in PostgreSQL.
  + Understood the use of IN and OUT parameters for dynamic input and output handling.

#### Dynamic Querying:

* + Gained the ability to write procedures that count records based on dynamic input, such as gender.
  + Learned how to avoid repetitive queries by automating common HR analytics tasks.

#### Result Display:

* + Learned to use RAISE NOTICE for real-time feedback in pgAdmin.
  + Understood how to present calculated results clearly for reporting purposes.

#### Database Management Skills:

* + Practiced working with tables, inserting data, and validating results.
  + Developed analytical skills for HR reporting and workforce diversity tracking.

#### Transaction Automation:

* + Learned to automate retail operations using stored procedures.
  + Understood how to validate stock before processing orders.

#### Inventory Management:

* + Gained experience in updating multiple tables (products and sales) in a single procedure.
  + Learned how to maintain data integrity by adjusting quantity\_remaining and quantity\_sold.

#### Conditional Logic in Procedures:

* + Learned to implement IF-ELSE logic to handle sufficient and insufficient stock scenarios.
  + Practiced providing real-time notifications to the user.

#### Dynamic Input Handling:

* + Developed the skill to take dynamic product name and quantity as input for automated processing.
  + Learned to calculate total sale price dynamically using stored values.

#### Practical Application:

* + Understood how database procedures can simulate real-world business operations, like inventory control and order management.
  + Enhanced ability to solve complex database problems with procedural programming.