

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment No: 6

Student Name: Himanshu Dubey

UID: 23BCS12967

Branch: B.E./C.S.E.

Section/Group: KRG_2-A

Semester: 5th

Subject Code: 23CSP-333

Subject Name: ADBMS

Medium Level Problem

Question: HR-Analytics: Employee count based on dynamic gender passing

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) .

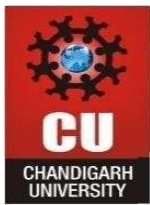
To solve this problem, the company needs an automated database-driven solution that can instantly return the count of employees by gender through a stored procedure that:

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.

Solution:

--INPUT TABLES:

```
CREATE TABLE employee_info (  
  id SERIAL PRIMARY KEY,  name  
  VARCHAR(50)    NOT    NULL,
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
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');
```

```
CREATE OR REPLACE PROCEDURE sp_get_employees_by_gender(  
    IN p_gender VARCHAR(50),  
    OUT p_employee_count INT  
)  
LANGUAGE plpgsql  
AS $$  
BEGIN  
    -- Count total employees by gender  
    SELECT COUNT(id)  
    INTO p_employee_count  
    FROM employee_info  
    WHERE gender = p_gender;  
  
    -- Display the result  
    RAISE NOTICE 'Total employees with gender %: %', p_gender, p_employee_count; END;  
$$;  
  
CALL sp_get_employees_by_gender('Male', NULL);
```

Output:

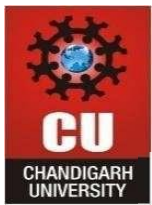
| Data Output | | Messages | I |
|---|------------------|----------|---|
| <div><div>≡+</div><div>▼</div><div>▼</div><div></div></div> | | | |
| | p_employee_count | | |
| | integer | | |
| 1 | | | 6 |

Hard Level Problem

Question : Smart Store Automated Purchase System

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!"



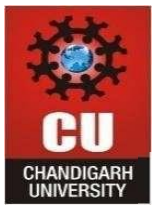
Solution:

--INPUT TABLES:

```
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),
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

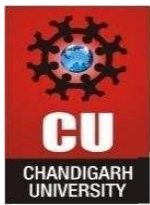
Discover. Learn. Empower.

```
('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;
```

```
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  
    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;

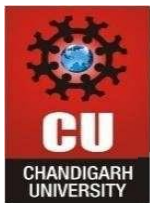
\$\$;

CALL pr_buy_products ('MacBook Pro 14"', 1);

Output :

| | order_id [PK] integer | order_date date | product_code character varying (10) | quantity_ordered integer | sale_price numeric (10,2) |
|---|--------------------------|--------------------|--|-----------------------------|------------------------------|
| 1 | 1 | 2025-09-15 | P001 | 1 | 109999.00 |
| 2 | 2 | 2025-09-16 | P002 | 2 | 199998.00 |
| 3 | 3 | 2025-09-17 | P003 | 1 | 55999.00 |
| 4 | 4 | 2025-09-18 | P005 | 2 | 59998.00 |
| 5 | 5 | 2025-09-19 | P001 | 1 | 109999.00 |
| 6 | 6 | 2025-09-24 | P004 | 1 | 189999.00 |

| | product_code [PK] character varying (10) | product_name character varying (100) | price numeric (10,2) | quantity_remaining integer | quantity_sold integer |
|---|---|---|-------------------------|-------------------------------|--------------------------|
| 1 | P001 | iPhone 13 Pro Max | 109999.00 | 10 | 0 |
| 2 | P002 | Samsung Galaxy S23 Ultra | 99999.00 | 8 | 0 |
| 3 | P003 | iPad Air | 55999.00 | 5 | 0 |
| 4 | P005 | Sony WH-1000XM5 Headphones | 29999.00 | 15 | 0 |
| 5 | P004 | MacBook Pro 14" | 189999.00 | 2 | 1 |



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Data Output Messages Notifications

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

Query returned successfully in 79 msec.