Experiment 1.6

Student Name: Gagnesh Kakkar **UID:** 23BCS11196

Branch: B.E-C.S.E **Section/Group:** 23BCS_KRG-2B

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

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

MEDIUM - LEVEL

1. **Problem Title:** HR-Analytics: Employee count based on dynamic gender passing

2. **Procedure (Step-by-Step):**

   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.

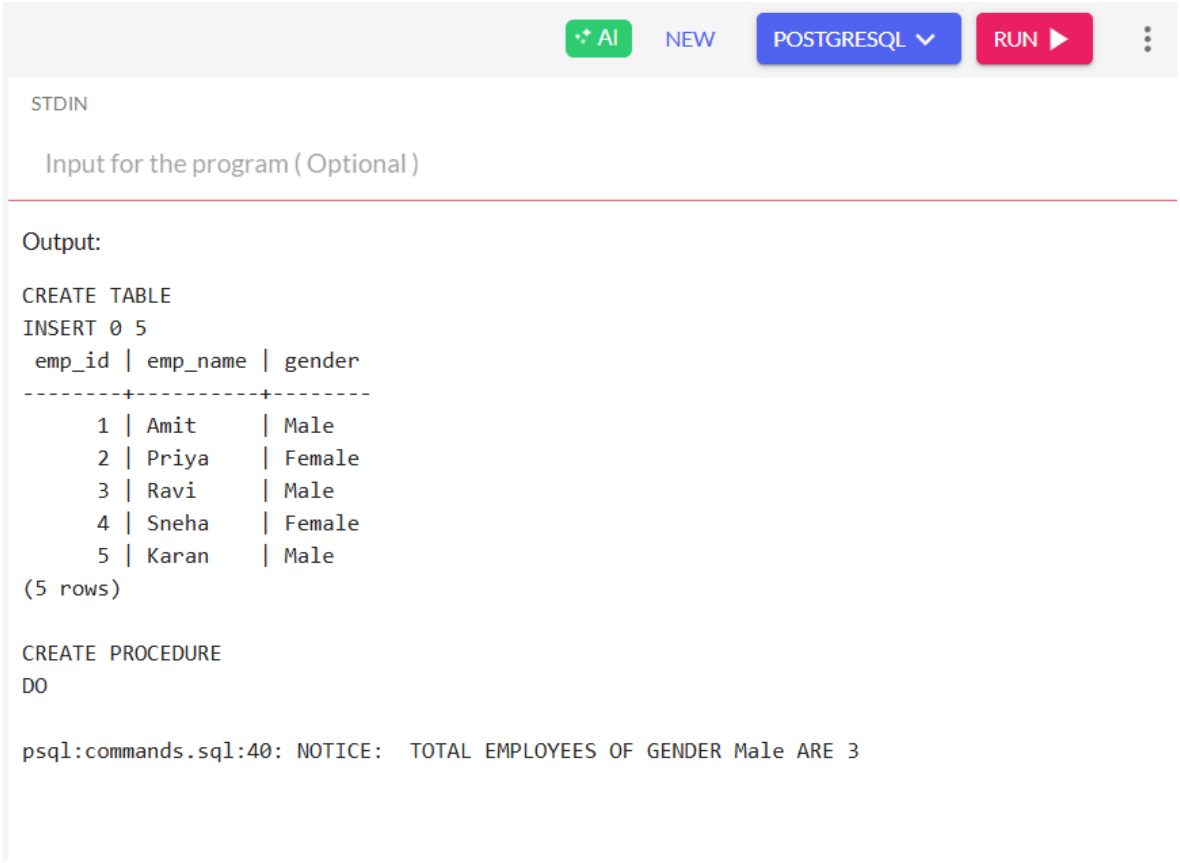
3. **SQL Commands:**

OneCompiler

commands.sql EXPERIMENT-6(MEDIUM)

```sql

-----------------------MEDIUM PROBLEM--------------------------
CREATE TABLE employees (
    emp_id SERIAL PRIMARY KEY,
    emp_name VARCHAR(100),
    gender VARCHAR(10)
);

-- Sample data
INSERT INTO employees (emp_name, gender) VALUES
('Amit', 'Male'),
('Priya', 'Female'),
('Ravi', 'Male'),
('Sneha', 'Female'),
('Karan', 'Male');

select * from EMPLOYEES;

----CREATING A PROCEDURE----
CREATE OR REPLACE PROCEDURE count_employees_by_gender(
    IN input_gender VARCHAR,
   OUT total_count int
)
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT COUNT(*) INTO total_count
    FROM employees
    WHERE gender = input_gender;
END;
$$;

---CALLING THE PROCEDURE-----
DO $$
DECLARE
    result INT;
BEGIN
    CALL count_employees_by_gender('Male', result);
    RAISE NOTICE 'TOTAL EMPLOYEES OF GENDER Male ARE %', result;
END;
$$;

```

4. **Output:**



HARD - LEVEL

1. **Problem Title:** SmartStore Automated Purchase System

2. **Procedure (Step-by-Step):**

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!"

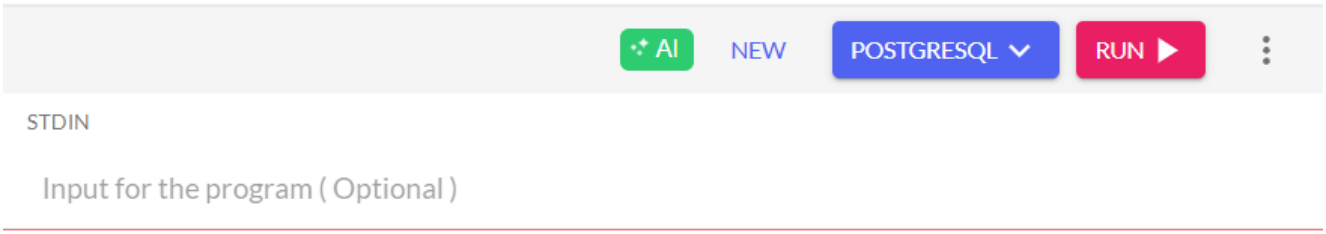
3. SQL Commands:

OneCompiler

commands.sql EXPERIMENT-6(HARD)

```sql
----------HARD PROBLEM ----------
CREATE TABLE products (
    product_id SERIAL PRIMARY KEY,
    product_name VARCHAR(100),
    price NUMERIC(10,2),
    quantity_remaining INT,
    quantity_sold INT DEFAULT 0
);

INSERT INTO products (product_name, price, quantity_remaining) VALUES
('Smartphone', 30000, 50),
('Tablet', 20000, 30),
('Laptop', 60000, 20);

CREATE TABLE sales (
    sale_id SERIAL PRIMARY KEY,
    product_id INT REFERENCES products(product_id),
    quantity INT,
    total_price NUMERIC(10,2),
    sale_date TIMESTAMP DEFAULT NOW()
);

CREATE OR REPLACE PROCEDURE place_order(
    IN p_product_id INT,
    IN p_quantity INT
)
LANGUAGE plpgsql
AS $$
DECLARE
    available_stock INT;
    product_price NUMERIC(10,2);
BEGIN
    SELECT quantity_remaining, price
    INTO available_stock, product_price
    FROM products
    WHERE product_id = p_product_id;

    IF available_stock IS NULL THEN
        RAISE NOTICE 'Product ID % does not exist!', p_product_id;
    ELSIF available_stock >= p_quantity THEN
        -- LOGGING THE ORDER
        INSERT INTO sales (product_id, quantity, total_price)
        VALUES (p_product_id, p_quantity, p_quantity * product_price);

        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 PLACE_ORDER(2,20); --PRODUCT SOLD SUCCESSFULLY AND QUANTITY_REMAINING COLUMN SET TO -20 AND
SELECT * FROM SALES;
SELECT * FROM PRODUCTS;
CALL PLACE_ORDER(3,100); --INSUFFICIENT QUANTITY AVAILABLE
```

4. Output:



Output:

```
CREATE TABLE
INSERT 0 3
CREATE TABLE
CREATE PROCEDURE
CALL
 sale_id | product_id | quantity | total_price |         sale_date
---------+------------+----------+-------------+----------------------------
       1 |          2 |       20 |   400000.00 | 2025-09-28 15:08:21.159238
(1 row)

 product_id | product_name |  price   | quantity_remaining | quantity_sold
------------+--------------+----------+--------------------+---------------
          1 | Smartphone   | 30000.00 |                 50 |             0
          3 | Laptop       | 60000.00 |                 20 |             0
          2 | Tablet       | 20000.00 |                 10 |            20
(3 rows)

CALL

psql:commands.sql:58: NOTICE:  Product sold successfully!
psql:commands.sql:61: NOTICE:  Insufficient Quantity Available!
```