Experiment - 6

Student Name: Himanshu Gupta
UID: 23BCS10889
Branch: BE-CSE
Section/Group: KRG_2B
Semester: 5th
Date of Performance: 22/9/25
Subject Name: ADBMS
Subject Code: 23CSP-333

1. Problem Description/Aim:

Medium-Problem Title: **Gender Diversity Tracking**-Create a PostgreSQL stored procedure to track gender diversity in the workforce. The procedure takes a gender as input and returns the total number of employees of that gender, providing HR with instant and secure reporting.

Procedure (Step-by-Step):

1. Create a table employees with columns like emp_id,emp_name and gender.
2. Insert sample data with varying genders.
3. Create a stored procedure 'count_employees_by_gender' that:
   - Takes a gender as input.
   - Counts the number of employees with that gender.
   - Returns the result as an OUT parameter.
4. Call the procedure in a DO block to capture and display the result.

Sample Output Description:

- Input: 'Male' --- Output: 3
- Input: 'Female' --- Output: 2

-HR sees results instantly without accessing full employee data.

Hard-Problem Title: **Order Placement and Inventory Management**-Automate the ordering process in a retail company. The procedure validates stock availability, logs sales, updates inventory, and provides real-time confirmation or rejection messages.

Procedure (Step-by-Step):

1. Create products table with columns: product_id, product_name, price, quantity_remaining, quantity_sold.
2. Create sales table with columns: sale_id, product_id,

quantity, total_price, sale_date.

3. Create a stored procedure place_order that:
   - Takes product_id and quantity as input.
   - Checks if quantity_remaining is sufficient.
   - If yes:
     - Logs the sale in sales table.
       - Updates products(decrease quantity_remaining, increase quantity_sold).
     - Display “Product sold successfully!!”.
   - If no:
     - Display “Insufficient quantity available!!”
4. Call the procedure for different orders to validate functionality.

Sample Output Description:

- Order 5 units of Smartphone (stock available): "Product sold successfully!".
- Order 100 units of Tablet (insufficient stock): "Insufficient Quantity Available!".
- Inventory updates automatically for successful orders.

2. **Objective:** The objective is to automate critical business operations using PostgreSQL stored procedures. For HR, it tracks gender diversity by returning the total count of employees by gender. For retail, it manages orders by validating stock, logging sales, updating inventory, and providing real-time confirmation or rejection messages. This ensures efficiency, accuracy, and real-time insights in both workforce and inventory management.

3. SQL QUERY AND OUTPUTS

MEDEIUM PROBLEM

```
-- 1. Create the table
CREATE TABLE employees (
    emp_id SERIAL PRIMARY KEY,
    emp_name VARCHAR(100),
    gender VARCHAR(10)
);

-- 2. Insert sample data
INSERT INTO employees (emp_name, gender) VALUES
('Himanshu Gupta', 'M'),
('Jaskirat Singh', 'M'),
('Devjot Singh', 'M'),
('Kashish Mittal', 'F'),
('Dhruv Jadoo', 'M'),
('Hemant Narain', 'M');

-- 3. Create the procedure
CREATE OR REPLACE PROCEDURE count_employees_by_gender(
    IN input_gender VARCHAR,
    OUT total_count INT
)
LANGUAGE plpgsql AS $$
BEGIN
    SELECT COUNT(*) INTO total_count
    FROM employees
    WHERE gender = input_gender;
END;
$$;

-- 4. Call the procedure
DO $$
DECLARE
    result INT;
BEGIN
    CALL count_employees_by_gender('M', result);
    RAISE NOTICE 'TOTAL EMPLOYEES OF GENDER M ARE %', result;
END;
$$;
```

```
31  DO $$
32  DECLARE
33      result INT;
34  BEGIN
35      CALL count_employees_by_gender('M', result);
36      RAISE NOTICE 'TOTAL EMPLOYEES OF GENDER M ARE %', result;
37  END;
38  $$;
39
```

Data Output | Messages | Notifications

```
NOTICE:  TOTAL EMPLOYEES OF GENDER M ARE 5
DO

Query returned successfully in 59 msec.
```

```
40  select * from employees
```

Data Output | Messages | Notifications

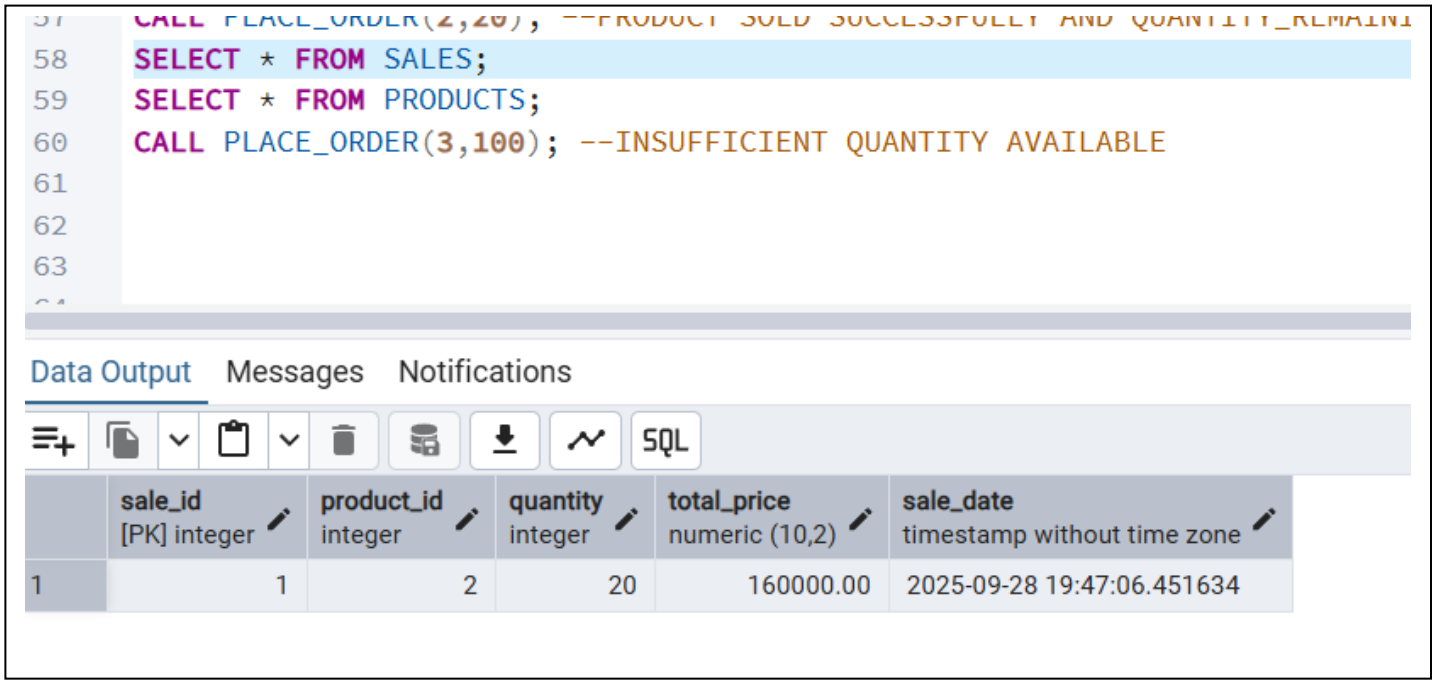
	emp_id [PK] integer	emp_name character varying (100)	gender character varying (10)
1	1	Himanshu Gupta	M
2	2	Jaskirat Singh	M
3	3	Devjot Singh	M
4	4	Kashish Mittal	F
5	5	Dhruv Jadoo	M
6	6	Hemant Narain	M

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
('Headphones', 2500, 100),
('Smartwatch', 8000, 40),
('Desktop', 45000, 15);

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 AND DATA LOGGED TO SALES TABLE
SELECT * FROM SALES;
SELECT * FROM PRODUCTS;
CALL PLACE_ORDER(3,100); --INSUFFICIENT QUANTITY AVAILABLE
```



63

Data Output | Messages | Notifications

Showing

	product_id [PK] integer	product_name character varying (100)	price numeric (10,2)	quantity_remaining integer	quantity_sold integer
1	1	Headphones	2500.00	100	0
2	3	Desktop	45000.00	15	0
3	2	Smartwatch	8000.00	20	20

```
59  SELECT * FROM PRODUCTS;
60  CALL PLACE_ORDER(3,100); --INSUFFICIENT QUANTITY AVAILABLE
61
62
63
```

Data Output | Messages | Notifications

```
NOTICE:  Insufficient Quantity Available!
CALL

Query returned successfully in 74 msec.
```