Experiment 5

Student Name: Bharat **UID: 23BCS13947**
Branch: CSE **Section/Group: KRG 3-A**
Semester: 5th **Date of Performance:25/09/2025**
Subject Name: ADBMS **Subject Code: 23CSP-333**

1. Aim: **Problem 1:**

a) Create a large dataset:
   - Create a table names transaction_data (id , value) with 1 million records.
   - take id 1 and 2, and for each id, generate 1 million records in value column.
   - Use Generate_series () and random() to populate the data.

b) Create a normal view and materialized view to for sales_summary, which includes total_quantity_sold, total_sales, and total_orders with aggregation.

c) Compare the performance and execution time of both.

Problem 2:

The company TechMart Solutions stores all sales transactions in a central database.
A new reporting team has been formed to analyze sales but they should not have direct access to the base tables for security reasons.
The database administrator has decided to:

- Create restricted views to display only summarized, non-sensitive data.
- Assign access to these views to specific users using DCL commands (GRANT, REVOKE).

2. Objective:

- To learn how to create large datasets in SQL using generate_series() and random().
- To practice creating and populating tables with millions of records efficiently.
- To understand how to create normal and materialized views for aggregated data.
- To analyze sales data using aggregate functions like SUM(), COUNT(), and AVG().
- To compare the performance and execution time of normal views versus materialized views for large datasets.

3. DBMS script and output:

----------Problem 1----------

```
CREATE TABLE transaction_data (
   id INT,
   value NUMERIC
```

```
);

INSERT INTO transaction_data (id, value)
SELECT 1, random() * 1000
FROM generate_series(1, 1000000);

INSERT INTO transaction_data (id, value)
SELECT 2, random() * 1000
FROM generate_series(1, 1000000);

CREATE OR REPLACE VIEW sales_summary_view AS SELECT
   id,
   COUNT(*) AS total_orders,
   SUM(value) AS total_sales,
   AVG(value) AS avg_transaction
FROM transaction_data
GROUP BY id;

SELECT * FROM sales_summary_view;

CREATE MATERIALIZED VIEW sales_summary_mv AS SELECT
   id,
   COUNT(*) AS total_orders,
   SUM(value) AS total_sales,
   AVG(value) AS avg_transaction
FROM transaction_data
GROUP BY id;

SELECT * FROM sales_summary_mv;

EXPLAIN ANALYZE
SELECT * FROM sales_summary_view;

EXPLAIN ANALYZE
SELECT * FROM sales_summary_mv;

REFRESH MATERIALIZED VIEW sales_summary_mv;
```

Result Grid | Filter Rows: | Export:

id	total_orders	total_sales	avg_transaction
1	15625	7805937	499.5800
2	15625	7811693	499.9484

----------Problem 2----------

```
CREATE TABLE customer_master (
customer_id VARCHAR(5) PRIMARY KEY,
full_name VARCHAR(50) NOT NULL,
phone VARCHAR(15),     email
VARCHAR(50),     city VARCHAR(30)
);

CREATE TABLE product_catalog (
product_id VARCHAR(5) PRIMARY KEY,
product_name VARCHAR(50) NOT NULL,
brand VARCHAR(30),     unit_price
NUMERIC(10,2) NOT NULL
);

CREATE TABLE sales_orders (     order_id SERIAL PRIMARY KEY,
product_id VARCHAR(5) REFERENCES product_catalog(product_id),
quantity INT NOT NULL,     customer_id VARCHAR(5) REFERENCES
customer_master(customer_id),     discount_percent NUMERIC(5,2),
order_date DATE NOT NULL
);

INSERT INTO customer_master (customer_id, full_name, phone, email, city) VALUES
('C1', 'Amit Sharma', '9876543210', 'amit.sharma@example.com', 'Delhi'),
('C2', Bharat Sharma, '9876501234', bharar.sharma@example.com', 'Mumbai'),
('C3', 'Ravi Kumar', '9988776655', 'ravi.kumar@example.com', 'Bangalore');

INSERT INTO product_catalog (product_id, product_name, brand, unit_price) VALUES
('P1', 'Smartphone X100', 'Samsung', 25000.00),
('P2', 'Laptop Pro 15', 'Dell', 65000.00),
('P3', 'Wireless Earbuds', 'Sony', 5000.00);

INSERT INTO sales_orders (product_id, quantity, customer_id, discount_percent, order_date)
VALUES
```

```
('P1', 2, 'C1', 5.00, '2025-09-01'),
('P2', 1, 'C2', 10.00, '2025-09-02'),
('P3', 3, 'C3', 0.00, '2025-09-03'),
('P1', 1, 'C2', 5.00, '2025-09-04');

CREATE VIEW v_sales_summary AS
SELECT
    O.order_date,
    P.product_name,
    SUM(O.quantity) AS total_quantity_sold,
    SUM((P.unit_price * O.quantity) - ((P.unit_price * O.quantity) * O.discount_percent / 100)) AS
total_sales,
    COUNT(O.order_id) AS total_orders
FROM sales_orders O
JOIN product_catalog P ON O.product_id = P.product_id
GROUP BY O.order_date, P.product_name;

CREATE ROLE reporting_user
LOGIN
PASSWORD 'report123';

GRANT SELECT ON v_sales_summary TO reporting_user;

SELECT * FROM v_sales_summary;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_date	product_name	total_quantity_sold	total_sales	total_orders
2025-09-01	Smartphone X100	2	47500.00000000	1
2025-09-04	Smartphone X100	1	23750.00000000	1
2025-09-02	Laptop Pro 15	1	58500.00000000	1
2025-09-03	Wireless Earbuds	3	15000.00000000	1

4. Learning Outcomes (What I have Learnt):

- Gained hands-on experience in creating large datasets and defining relational tables in PostgreSQL. ○ Learned to create normal views, materialized views, and aggregate transactional data efficiently.
- Understood performance differences between views and materialized views and how to refresh materialized views. ○ Acquired skills to secure data using restricted views and control access with GRANT and REVOKE commands.
- Practiced joining multiple tables, calculating totals, and providing summarized insights while protecting sensitive information.