



Worksheet 5

Student Name: Shivanshu Ranjan

Branch: CSE

Semester: 5th

Subject Name: ADBMS

UID: 23BCS10193

Section/Group: KRG 3-A

Date of Performance: 25/09/2025

Subject Code: 23CSP-333

1. Aim:

- 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.

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

- Compare the performance and execution time of both.

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 create and populate a large dataset using SQL functions like generate_series() and random() for performance testing.
- To design normal views and materialized views for summarizing sales data, and to compare their performance in terms of execution time.
- To implement a secure reporting mechanism by restricting direct access to base tables and providing summarized, non-sensitive data through views.
- To apply Data Control Language (DCL) commands such as GRANT and REVOKE for assigning controlled access rights to specific users.
- To demonstrate how views and materialized views can improve query efficiency, security, and reporting in a sales transaction database.



3. Code:

1.

-- Performance Benchmarking (Medium Level)

```
CREATE TABLE transaction_data (
```

```
    id INT,
```

```
    value INT
```

```
);
```

-- Insert 1 million rows for id=1

```
INSERT INTO transaction_data (id, value)
```

```
SELECT 1, (random() * 1000)
```

```
FROM generate_series(1, 1000000);
```

-- Insert 1 million rows for id=2

```
INSERT INTO transaction_data (id, value)
```

```
SELECT 2, (random() * 1000)
```

```
FROM generate_series(1, 1000000);
```

```
SELECT COUNT(*) FROM transaction_data;
```

-- Normal View

```
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;

-- Check Performance (Normal View)

EXPLAIN ANALYZE SELECT * FROM sales_summary_view;

-- Materialized 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;

-- Check Performance (Materialized View)

EXPLAIN ANALYZE SELECT * FROM sales_summary_mv;
```

2.

-- Securing Data Access with Views and Role-Based Permissions (Hard Level)

```
CREATE TABLE customer_master (

    customer_id SERIAL PRIMARY KEY,

    full_name VARCHAR(100)

);

CREATE TABLE product_catalog (

    product_id SERIAL PRIMARY KEY,

    product_name VARCHAR(100),

    unit_price NUMERIC(10,2)
```



CREATE TABLE *sales_orders* (

order_id SERIAL *PRIMARY KEY*,

customer_id INT *REFERENCES* customer_master(customer_id),

product_id INT *REFERENCES* product_catalog(product_id),

order_date DATE,

quantity INT,

discount_percent NUMERIC(5,2)

);

INSERT INTO customer_master (full_name) VALUES

('Shivanshu Ranjan'),

('Tanya Verma'),

('Alok Kumar'),

('Neha Sharma');

INSERT INTO product_catalog (product_name, unit_price) VALUES

('Laptop', 60000),

('Keyboard', 1200),

('Monitor', 15000),

('Mouse', 800);

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

(1, 1, '2025-09-01', 1, 10),

(2, 2, '2025-09-02', 2, 5),

(3, 3, '2025-09-03', 1, 20),

(4, 4, '2025-09-05', 3, 15);



-- Create View

CREATE OR REPLACE VIEW vW_ORDER_SUMMARY AS

SELECT

O.order_id,

O.order_date,

P.product_name,

C.full_name,

(P.unit_price * O.quantity)

- ((P.unit_price * O.quantity) * O.discount_percent / 100) AS final_cost

FROM customer_master AS C

JOIN sales_orders AS O

ON O.customer_id = C.customer_id

JOIN product_catalog AS P

ON P.product_id = O.product_id;

-- Create Restricted role/user (shivanshu)

CREATE ROLE shivanshu LOGIN PASSWORD '1234';

-- shivanshu logs in and runs (In new query window)

SELECT * FROM vW_ORDER_SUMMARY; -- permission denied for view vW_ORDER_SUMMARY

-- Grant access to shivanshu

GRANT SELECT ON vW_ORDER_SUMMARY TO shivanshu;

SELECT * FROM vW_ORDER_SUMMARY; -- now shivanshu can view vW_ORDER_SUMMARY

SELECT * FROM customer_master; -- shivanshu can't view base tables(direct access)

-- Revoke access from shivanshu

REVOKE SELECT ON vW_ORDER_SUMMARY FROM shivanshu;

SELECT * FROM vW_ORDER_SUMMARY; -- now shivanshu can't view vW_ORDER_SUMMARY

4. Output:

```
count
-----
2000000
(1 row)
```

```
CREATE VIEW
id | total_orders | total_sales | avg_transaction
-----+-----+-----+-----
1 | 1000000 | 499566284 | 499.5662840000000000
2 | 1000000 | 499758617 | 499.7586170000000000
(2 rows)
```

QUERY PLAN

```
Finalize GroupAggregate (cost=26576.79..26584.29 rows=200 width=52) (actual time=587.379..593.193 rows=2)
  Group Key: transaction_data.id
    -> Sort (cost=26576.79..26577.79 rows=400 width=52) (actual time=587.353..593.164 rows=6 loops=1)
      Sort Key: transaction_data.id
      Sort Method: quicksort Memory: 25kB
        -> Gather (cost=26517.50..26559.50 rows=400 width=52) (actual time=586.883..593.150 rows=6 loops=1)
          Workers Planned: 2
          Workers Launched: 2
            -> Partial HashAggregate (cost=25517.50..25519.50 rows=200 width=52) (actual time=578.705..586.883 rows=2 loops=1)
              Group Key: transaction_data.id
              -> Parallel Seq Scan on transaction_data (cost=0.00..17183.75 rows=833375 width=8)

Planning time: 0.106 ms
Execution time: 593.269 ms
(13 rows)
```

```
id | total_orders | total_sales | avg_transaction
-----+-----+-----+-----
2 | 1000000 | 499758617 | 499.7586170000000000
1 | 1000000 | 499566284 | 499.5662840000000000
(2 rows)
```

QUERY PLAN

```
Seq Scan on sales_summary_mv (cost=0.00..20.20 rows=1020 width=52) (actual time=0.004..0.005 rows=2 loops=1)
Planning time: 0.022 ms
Execution time: 0.016 ms
(3 rows)
```

(1)

order_id	order_date	product_name	full_name	final_cost
1	2025-09-01	Laptop	Shivanshu Ranjan	54000.0000000000000000
2	2025-09-02	Keyboard	Tanya Verma	2280.0000000000000000
3	2025-09-03	Monitor	Alok Kumar	12000.0000000000000000
4	2025-09-05	Mouse	Neha Sharma	2040.0000000000000000

(4 rows)

ERROR: permission denied for view vw_order_summary

ERROR: permission denied for table customer_master

(2)

5. Learning Outcomes:

- Generate and manage large datasets using SQL functions like `generate_series()` and `random()`.
- Differentiate between normal views and materialized views with performance analysis.
- Secure data by allowing access only to summarized, non-sensitive information through views.
- Apply DCL commands (`GRANT`, `REVOKE`) to control user access and permissions.