

## Experiment 5

**Student Name:** Jayaprakash

**UID:** 23BAI70240

**Branch:** BE-AIT-CSE

**Section/Group:** 23AIT-KRG-G2

**Semester:** 5th

**Date of Performance:** 23<sup>rd</sup> Sept, 2025

**Subject Name:** ADBMS

**Subject Code:** 23CSP-333

### 1. **AIM:**

**Medium:** Performance Benchmarking: Normal View vs. Materialized View.

**Hard:** Securing Data Access with Views and Role-Based Permissions.

### 2. **Tools Used:** pgAdmin4

### 3. **Experiment:**

**Medium: -**

1. 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.
2. Create a normal view and materialized view to for sales\_summary, which includes total\_quantity\_sold, total\_sales, and total\_orders with aggregation.
3. Compare the performance and execution time of both.

**Hard: -**

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:

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

#### 4. Solution:

##### Medium: -

```
CREATE TABLE transaction_data (  
    id INT,  
    value INT  
);  
  
-- For id = 1  
INSERT INTO transaction_data (id, value)  
SELECT 1, random() * 1000 -- simulate transaction amounts 0-1000  
FROM generate_series(1, 1000000);  
  
-- For id = 2  
INSERT INTO transaction_data (id, value)  
SELECT 2, random() * 1000  
FROM generate_series(1, 1000000);  
  
SELECT * FROM transaction_data  
  
--WITH 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;  
  
EXPLAIN ANALYZE  
SELECT * FROM sales_summary_view;  
  
--WITH 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;  
  
EXPLAIN ANALYZE  
SELECT * FROM sales_summary_mv;
```



## Hard: -

```
CREATE 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;

SELECT * FROM vw_ORDER_SUMMARY;

--1. CREATE USER
CREATE ROLE ALOK
LOGIN
PASSWORD 'alok';
GRANT SELECT ON vw_ORDER_SUMMARY TO ALOK;
--client will only be able to do the select, no alteration, and he can not see the sql
REVOKE SELECT ON vw_ORDER_SUMMARY FROM ALOK;
```

## 5. Output:

	id integer 	value integer 
994	1	123
995	1	535
996	1	816
997	1	235
998	1	645
999	1	643
1000	1	919

## Normal View

	QUERY PLAN text	
1	Finalize GroupAggregate (cost=39275.05..39275.59 rows=2 width=52) (actual time=256.541..261.581 rows=2 loops=1)	
2	Group Key: transaction_data.id	
3	-> Gather Merge (cost=39275.05..39275.52 rows=4 width=52) (actual time=256.525..261.563 rows=5 loops=1)	
4	Workers Planned: 2	
5	Workers Launched: 2	
6	-> Sort (cost=38275.03..38275.03 rows=2 width=52) (actual time=183.277..183.278 rows=2 loops=3)	
7	Sort Key: transaction_data.id	
8	Sort Method: quicksort Memory: 25kB	
9	Worker 0: Sort Method: quicksort Memory: 25kB	
10	Worker 1: Sort Method: quicksort Memory: 25kB	
11	-> Partial HashAggregate (cost=38275.00..38275.02 rows=2 width=52) (actual time=182.192..182.199 rows=2 loops=3)	
12	Group Key: transaction_data.id	
13	Batches: 1 Memory Usage: 24kB	
14	Worker 0: Batches: 1 Memory Usage: 24kB	
15	Worker 1: Batches: 1 Memory Usage: 24kB	
16	-> Parallel Seq Scan on transaction_data (cost=0.00..25775.00 rows=1250000 width=8) (actual time=0.009..45.880 rows=100...	
17	Planning Time: 1.011 ms	
18	Execution Time: 262.380 ms	

## Materialized View

	QUERY PLAN text	
1	Seq Scan on sales_summary_mv (cost=0.00..20.20 rows=1020 width=52) (actual time=0.006..0.007 rows=2 loops=...	
2	Planning Time: 0.627 ms	
3	Execution Time: 0.015 ms	

## 6. Learning Outcomes:

- Learn't how to create a view.
- Learn't about the various types of views and their use cases.
- Learn't how to apply DCL Commands such as GRANT & REVOKE.
- Learn't how to use random() & generate\_series() functions.