

# **Experiment 1.5**

Student Name: Gagnesh Kakkar UID: 23BCS11196

Branch: B.E-C.S.E Section/Group: 23BCS KRG-2B

Semester: 5<sup>th</sup> Date of Performance: 24 Sept, 2025

Subject Name: ADBMS Subject Code: 23CSP-333

#### **MEDIUM - LEVEL**

1. **Problem Title:** Views: Performance Benchmarking : Normal View vs. Materialized View

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

- 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.

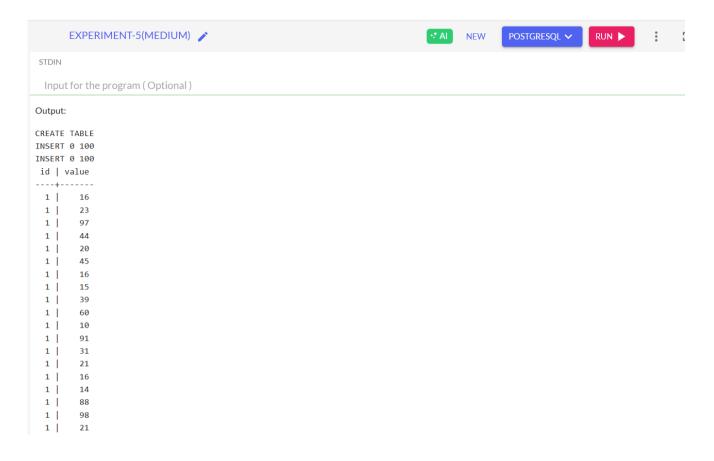
## 3. SQL Commands:

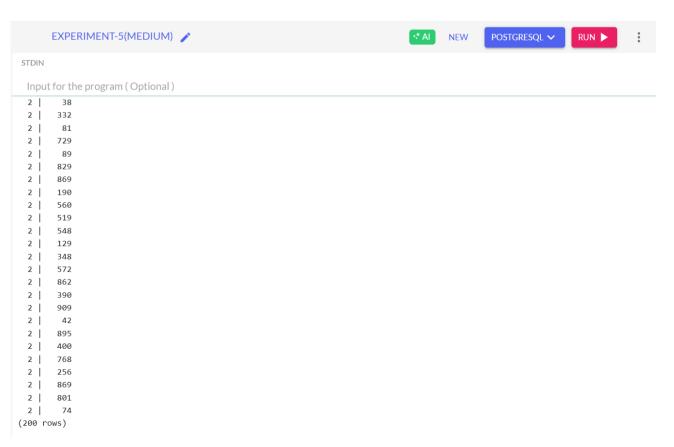
commands.sql

EXPERIMENT-5(MEDIUM)

```
1 - CREATE TABLE transaction data (
         id INT,
          value INT
 7 -- For id = 1
Instrt into transaction_data (id, value)
SELECT 1, random() * 100 -- simulate transaction amounts 0-1000
FROM generate_series(1, 100);
13 INSERT INTO transaction_data (id, value)
    SELECT 2, random() * 1000
15 FROM generate_series(1, 100);
17     SELECT *FROM transaction_data;
18
19
20
      --WITH NORMAL VIEW
    CREATE VIEW sales_summary_view AS
     SELECT
       id,
          COUNT(*) AS total_orders,
         SUM(value) AS total_sales,
AVG(value) AS avg_transaction
26
    FROM transaction_data
28
    GROUP BY id;
29
30
     EXPLAIN ANALYZE
    SELECT * FROM sales_summary_view;
34
35
36
37
    --WITH MATERIALIZED VIEW
38
    CREATE MATERIALIZED VIEW sales_summary_mv AS
39
    SELECT
40
         id,
10, COUNT(*) AS total_orders,
2 SUM(value) AS total_sales,
43 AVG(value) AS avg_transaction
44 FROM transaction_data
    GROUP BY id;
47
48
49 EXPLAIN ANALYZE
50 SELECT * FROM sales_summary_mv;
     REFRESH MATERIALIZED VIEW sales summary mv;
```

## 5. Output:





```
HashAggregate (cost=55.20..57.70 rows=200 width=52) (actual time=0.266..0.270 rows=2 loops=1)
Group Key: transaction_data.id
Batches: 1 Memory Usage: 40kB
-> Seq Scan on transaction_data (cost=0.00..32.60 rows=2260 width=8) (actual time=0.005..0.122 rows=200 loops=1)
Planning Time: 0.368 ms
Execution Time: 0.355 ms
(6 rows)

SELECT 2

QUERY PLAN

Seq Scan on sales_summary_mv (cost=0.00..20.20 rows=1020 width=52) (actual time=0.003..0.005 rows=2 loops=1)
Planning Time: 0.035 ms
Execution Time: 0.013 ms
(3 rows)

REFRESH MATERIALIZED VIEW
```

#### HARD - LEVEL

1. Problem Title: Views: Securing Data Access with Views and Role-Based Permissions

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

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:

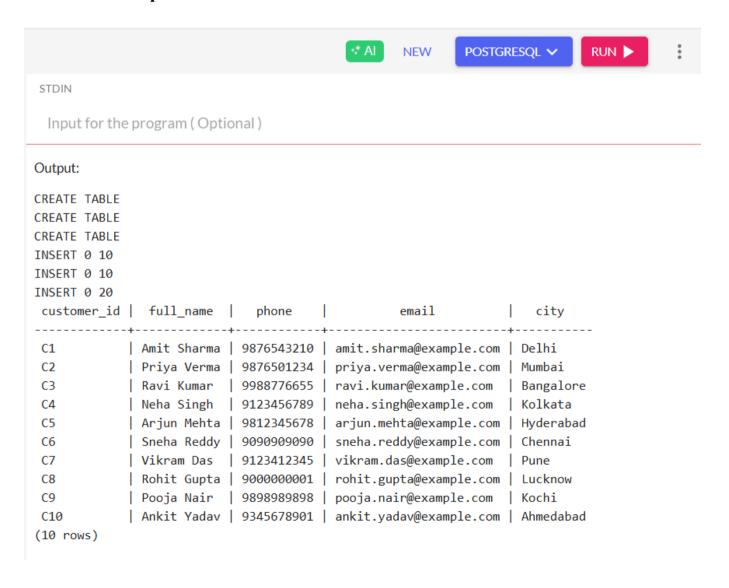
- 1. Create restricted views to display only summarized, non-sensitive data.
- 2. Assign access to these views to specific users using **DCL commands** (GRANT, REVOKE).
- 3. SQL Commands:

commands.sql EXPERIMENT-5(HARD) 💉

```
1 - 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)
  6
          ):
 8
 9 - CREATE TABLE product_catalog (
                      product_id VARCHAR(5) PRIMARY KEY,
10
                      product_name VARCHAR(50) NOT NULL,
                      brand VARCHAR(30)
                     unit_price NUMERIC(10,2) NOT NULL
14
         );
16 • 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
          );
27
          INSERT INTO customer_master (customer_id, full_name, phone, email, city) VALUES
         INSERT INTO customer_master (customer_id, full_name, phone, email, city) VALUI
('C1', 'Amit Sharma', '9876543210', 'amit.sharma@example.com', 'Delhi'),
('C2', 'Priya Verma', '9876501234', 'priya.verma@example.com', 'Mumbai'),
('C3', 'Ravi Kumar', '9988776655', 'ravi.kumar@example.com', 'Bangalore'),
('C4', 'Neha Singh', '9123456789', 'neha.singh@example.com', 'Kolkata'),
('C5', 'Arjun Mehta', '9812345678', 'arjun.mehta@example.com', 'Hyderabad'),
('C6', 'Sneha Reddy', '9090909090', 'sneha.reddy@example.com', 'Chennai'),
('C7', 'Vikram Das', '9123412345', 'vikram.das@example.com', 'Pune'),
('C8', 'Rohit Gupta', '9000000001', 'rohit.gupta@example.com', 'Lucknow'),
('C9', 'Pooja Nair', '988989898', 'pooja.nair@example.com', 'Kochi'),
('C10', 'Ankit Yadav', '9345678901', 'ankit.yadav@example.com', 'Ahmedabad');
38
40
          INSERT INTO product_catalog (product_id, product_name, brand, unit_price) VALUES
                                'Smartphone X100', 'Samsung', 25000.00), 
'Laptop Pro 15', 'Dell', 65000.00),
          ('P2', 'Laptop Pro 15', 'Dell', 65000.00),
('P3', 'Wireless Earbuds', 'Sony', 5000.00),
('P4', 'Smartwatch Fit', 'Apple', 30000.00),
('P5', 'Tablet 10.5', 'Lenovo', 22000.00),
('P6', 'Gaming Console', 'Sony', 45000.00),
('P7', 'Bluetooth Speaker', 'JBL', 7000.00),
('P8', 'Digital Camera', 'Canon', 55000.00),
('P9', 'LED TV 55 inch', 'LG', 60000.00),
('P10', 'Power Bank 20000mAh', 'Mi', 2500.00);
44
45
46
47
       INSERT INTO sales_orders (product_id, ('P1', 2, 'C1', 5.00, '2025-09-01'), ('P2', 1, 'C2', 10.00, '2025-09-02'), ('P3', 3, 'C3', 0.00, '2025-09-02'), ('P4', 1, 'C4', 8.00, '2025-09-04'), ('P5', 2, 'C5', 5.00, '2025-09-06'), ('P6', 1, 'C1', 12.00, '2025-09-06'), ('P7', 2, 'C2', 0.00, '2025-09-06'), ('P8', 1, 'C3', 10.00, '2025-09-08'), ('P9', 1, 'C6', 15.00, '2025-09-08'), ('P10', 4, 'C7', 0.00, '2025-09-10'), ('P1', 1, 'C8', 5.00, '2025-09-11'), ('P2', 2, 'C9', 10.00, '2025-09-11'), ('P3', 2, 'C10', 0.00, '2025-09-12'), ('P4', 1, 'C5', 8.00, '2025-09-13'), ('P6', 1, 'C7', 12.00, '2025-09-15'), ('P6', 1, 'C7', 12.00, '2025-09-16'), ('P7', 2, 'C8', 0.00, '2025-09-17'), ('P8', 1, 'C9', 10.00, '2025-09-18'), ('P9', 1, 'C10', 15.00, '2025-09-19'), ('P9', 1, 'C10', 15.00, '2025-09-19'), ('P10', 5, 'C4', 0.00, '2025-09-20');
          INSERT INTO sales_orders (product_id, quantity, customer_id, discount_percent, order_date) VALUES
64
67
70
71
76
```

```
76
 77
78
     SELECT * FROM customer_master;
     SELECT * FROM product_catalog;
     SELECT * FROM sales_orders;
 80
 82
 83
 84
     CREATE VIEW vW_ORDER_SUMMARY
 85
     SELECT
 86
         O.order_id,
 87
 88
         O.order_date,
 89
         P product name
 90
         C.full name.
         (P.unit_price * O.quantity) - ((P.unit_price * O.quantity) * O.discount_percent / 100) AS final_cost
 91
     FROM customer_master AS C
 93
     JOIN sales orders AS 0
         ON 0.customer_id = C.customer_id
 95
     JOIN product_catalog AS P
 96
         ON P.product_id = 0.product_id;
 97
     CREATE ROLE CLIENT_ABC
100
101
     LOGIN
102
     PASSWORD '1234';
103
104
     GRANT SELECT ON VW_ORDER_SUMMARY TO CLIENT_ABC;
106 REVOKE SELECT ON VW ORDER SUMMARY FROM CLIENT ABC;
```

#### 4. Output:



product_id	product_name	brand	unit_price
P1	Smartphone X100	Samsung	25000.00
P2	Laptop Pro 15	Dell	65000.00
P3	Wireless Earbuds	Sony	5000.00
P4	Smartwatch Fit	Apple	30000.00
P5	Tablet 10.5	Lenovo	22000.00
P6	Gaming Console	Sony	45000.00
P7	Bluetooth Speaker	JBL	7000.00
P8	Digital Camera	Canon	55000.00
P9	LED TV 55 inch	LG	60000.00
P10	Power Bank 20000mAh	Mi	2500.00
(10 rows)			

order_id	product_id	quantity	customer_id	discount_percent	order_date
1	P1	2	C1	5.00	2025-09-01
2	P2	1	C2	10.00	2025-09-02
3	P3	3	C3	0.00	2025-09-03
4	P4	1	C4	8.00	2025-09-04
5	P5	2	C5	5.00	2025-09-05
6	P6	1	C1	12.00	2025-09-06
7	P7	2	C2	0.00	2025-09-07
8	P8	1	C3	10.00	2025-09-08
9	P9	1	C6	15.00	2025-09-09
10	P10	4	C7	0.00	2025-09-10
11	P1	1	C8	5.00	2025-09-11
12	P2	2	C9	10.00	2025-09-12
13	P3	2	C10	0.00	2025-09-13
14	P4	1	C5	8.00	2025-09-14
15	P5	3	C6	5.00	2025-09-15
16	P6	1	C7	12.00	2025-09-16
17	P7	2	C8	0.00	2025-09-17
18	P8	1	C9	10.00	2025-09-18
19	P9	1	C10	15.00	2025-09-19
20	P10	5	C4	0.00	2025-09-20
(20 rows)					

CREATE VIEW