Experiment 1.5

Student Name: Gagnesh Kakkar **UID:** 23BCS11196

Branch: B.E-C.S.E **Section/Group:** 23BCS_KRG-2B

Semester: 5th **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:**

OneCompiler

commands.sql

EXPERIMENT-5(MEDIUM)

```sql
CREATE TABLE transaction_data (
    id INT,
    value INT
);


-- For id = 1
INSERT INTO transaction_data (id, value)
SELECT 1, random() * 100  -- simulate transaction amounts 0-1000
FROM generate_series(1, 100);

-- For id = 2
INSERT INTO transaction_data (id, value)
SELECT 2, random() * 1000
FROM generate_series(1, 100);

SELECT *FROM transaction_data;



--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
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;


REFRESH MATERIALIZED VIEW sales_summary_mv;
```

5. **Output:**

EXPERIMENT-5(MEDIUM) AI NEW POSTGRESQL RUN

STDIN

Input for the program (Optional)

Output:

```
CREATE TABLE
INSERT 0 100
INSERT 0 100
 id | value
----+-------
  1 |    16
  1 |    23
  1 |    97
  1 |    44
  1 |    20
  1 |    45
  1 |    16
  1 |    15
  1 |    39
  1 |    60
  1 |    10
  1 |    91
  1 |    31
  1 |    21
  1 |    16
  1 |    14
  1 |    88
  1 |    98
  1 |    21
```

EXPERIMENT-5(MEDIUM) AI NEW POSTGRESQL RUN

STDIN

Input for the program (Optional)

```
  2 |    38
  2 |   332
  2 |    81
  2 |   729
  2 |    89
  2 |   829
  2 |   869
  2 |   190
  2 |   560
  2 |   519
  2 |   548
  2 |   129
  2 |   348
  2 |   572
  2 |   862
  2 |   390
  2 |   909
  2 |    42
  2 |   895
  2 |   400
  2 |   768
  2 |   256
  2 |   869
  2 |   801
  2 |    74
(200 rows)
```

```
CREATE VIEW
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 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:**

OneCompiler

commands.sql

EXPERIMENT-5(HARD)

```sql
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', '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', '9898989898', 'pooja.nair@example.com', 'Kochi'),
('C10', 'Ankit Yadav', '9345678901', 'ankit.yadav@example.com', 'Ahmedabad');


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),
('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);



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'),
('P4', 1, 'C4', 8.00, '2025-09-04'),
('P5', 2, 'C5', 5.00, '2025-09-05'),
('P6', 1, 'C1', 12.00, '2025-09-06'),
('P7', 2, 'C2', 0.00, '2025-09-07'),
('P8', 1, 'C3', 10.00, '2025-09-08'),
('P9', 1, 'C6', 15.00, '2025-09-09'),
('P10', 4, 'C7', 0.00, '2025-09-10'),
('P1', 1, 'C8', 5.00, '2025-09-11'),
('P2', 2, 'C9', 10.00, '2025-09-12'),
('P3', 2, 'C10', 0.00, '2025-09-13'),
('P4', 1, 'C5', 8.00, '2025-09-14'),
('P5', 3, 'C6', 5.00, '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'),
('P10', 5, 'C4', 0.00, '2025-09-20');
```

```sql

SELECT * FROM customer_master;
SELECT * FROM product_catalog;
SELECT * FROM sales_orders;



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;



CREATE ROLE CLIENT_ABC
LOGIN
PASSWORD '1234';

GRANT SELECT ON vW_ORDER_SUMMARY TO CLIENT_ABC;

REVOKE SELECT ON vW ORDER SUMMARY FROM CLIENT ABC;
```

4. **Output:**

AI NEW POSTGRESQL RUN

STDIN

Input for the program (Optional)

Output:

```
CREATE TABLE
CREATE TABLE
CREATE TABLE
INSERT 0 10
INSERT 0 10
INSERT 0 20
 customer_id |  full_name  |   phone    |          email          |   city
-------------+-------------+------------+-------------------------+-----------
 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  | 9898989898 | pooja.nair@example.com  | Kochi
 C10         | Ankit Yadav | 9345678901 | ankit.yadav@example.com | Ahmedabad
(10 rows)
```

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