Experiment 5

Student Name: Muskan

UID:23BAI70172

Branch: BE-AIT-CSE

Section/Group:23AML-1(A)

Semester:5th

Date of Performance:25 September ,2025

Subject Name:ADBMS

Subject Code:23CSP-333

Medium-Level

1. **Problem Title:** Performance Benchmarking: Normal view vs Materialised view
2. **Problem Tasks and Description:**
   a) Create a large dataset:
   Create a table named 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.
   c) Compare the performance and execution time of both.

SQL Commands:

a.Creating the table Employee and generating 1 million records for both ids:

```
------------------------------Medium-----------------------------------------------
-----------------------------------------------------------------------------------
CREATE TABLE transaction_data (
    id INT,
    value INT
);
-- For id = 1
INSERT INTO transaction_data (id, value)
SELECT 1, (random() * 1000)::INT
FROM generate_series(1, 1000000);
-- For id = 2
INSERT INTO transaction_data (id, value)
SELECT 2, (random() * 1000)::INT
FROM generate_series(1, 1000000);

-----------------------------------------------------------------------------------
```

b. Creating both the Normal view and the Materialised view:

c. Using the “Explain Analyze” to compare both their performance:

```
---------------------------------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;

------------------------------ 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;
```

Output:

	QUERY PLAN text
1	Finalize GroupAggregate (cost=26399.39..26399.93 rows=2 width=76) (actual time=901.979..926.396 rows=2 loops=1)
2	Group Key: tbl_transaction_data.id
3	-> Gather Merge (cost=26399.39..26399.86 rows=4 width=76) (actual time=900.649..924.978 rows=6 loops=1)
4	Workers Planned: 2
5	Workers Launched: 2
6	-> Sort (cost=25399.37..25399.37 rows=2 width=76) (actual time=752.027..752.029 rows=2 loops=3)
7	Sort Key: tbl_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=25399.33..25399.36 rows=2 width=76) (actual time=750.982..751.294 rows=2 loops=3)
12	Group Key: tbl_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 tbl_transaction_data (cost=0.00..19149.33 rows=833333 width=15) (actual time=0.032..137.191 rows=66...
17	Planning Time: 2.855 ms
18	Execution Time: 928.475 ms

Explain Analyze of Normal View

	QUERY PLAN text
1	Seq Scan on vw_materialisedview_salessummary (cost=0.00..17.80 rows=780 width=76) (actual time=0.026..0.028 rows=2 loops=...
2	Planning Time: 0.104 ms
3	Execution Time: 0.045 ms

Expalin Analyze of Materialized View

Learning Outcome:

a. I learnt the practical uses of views
b. I learnt about different types of views and their applications
c. I learnt the advantage of materialized views for large amounts of data.

Hard - Level

1. **Problem Title:** Securing Data Access with views and Role Based Permissions
2. **Problem Task and Description:**
   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 basetables 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).

SQL Commands:

```
--create the user
CREATE USER CLIENT_1;
WITH PASSWORD '123';
--Grant user ceratin permissions as required
GRANT SELECT ON sales_summary_view TO CLIENT_1;
GRANT SELECT ON mv_random_tabl TO CLIENT_1;
--Rekove any permission if required

REVOKE SELECT ON sales_summary_view FROM CLIENT_1;
```

Learning Outcome:

Learnt about the views and how to use them for security purpose.

Learnt how to assign access to users using dcl commands.