Experiment - 5

Student Name: Riya Mehta **UID:** 23BCS14042
Branch: BE-CSE **Section/Group:** KRG_2B
Semester: 5th **Date of Performance:** 22/9/25
Subject Name: ADBMS **Subject Code:** 23CSP-333

1. Problem Description/Aim:

Medium-Problem Title: Generate 1 million records per ID in 'transaction_data' using generate_series() and random() ,create a normal view and a materialized view 'sales_summary' with aggregated metrics (total_quantity_sold , total_sales, total_orders) , and compare their performance and execution time.

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.

Sample Output Description:

The transaction_data table has 2 million rows (1 million per ID) with random values. The normal view sales_summary computes aggregates on the fly, while the materialized view sales_summary_mv stores precomputed results. Queries on the materialized view are much faster, but it needs refreshing when data changes, whereas the normal view always shows up-to-date results.

Hard-Problem Title: Create restricted views in the sales database to provide summarized, non-sensitive data to the reporting team, and control access using DCL commands(GRANT and REVOKE).

Procedure (Step-by-Step):

1. Create restricted views-
   - Define views that show only **aggregated sales data** (e.g., total_sales, total_orders) without exposing sensitive columns like customer details or payment info.

2. Assign access to reporting team(or client)-
   -Use "GRANT SELECT ON view_name TO reporting_user; " to give access.
3. Revoke access if needed.
   -Use "REVOKE SELECT ON view_name FROM reporting_user;" to remove access.
4. Verify access
   - Reporting users can query the view but cannot access base tables directly, ensuring security.

Sample Output Description:

The result shows the restricted view providing summarized sales data only like
- Columns shown are - product_id,total_quantity_sold, total_sales, total_orders
- Columns hidden are - Customer names, addresses, payment details

A reporting user querying the view sees something like :
- Product 101 - 5000 units sold, total sales Rs. 12,50,000,500 orders.
- Product 102 - 3200 units sold, total sales Rs. 8,60,000,320 orders.

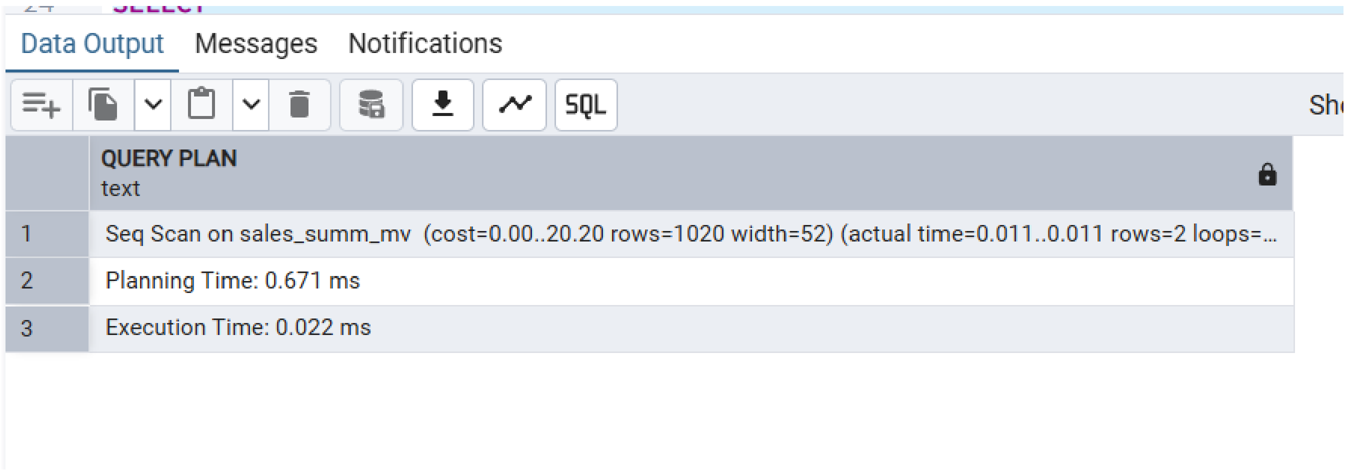
When the user tries to query the base "sales_transactions" table directly, access is denied, enforcing security.

2. **Objective:** To design and implement secure, efficient data access mechanisms by creating large-scale transaction datasets, summarizing them through normal and materialized views for performance comparison, and enforcing restricted access to sensitive data using views and DCL commands.

3. SQL QUERY AND OUTPUTS -

MEDIUM LEVEL PROBLEM

```
Create table TRANSACTION_DATA(id int,val decimal);
INSERT INTO TRANSACTION_DATA(ID,VAL)
SELECT 1,RANDOM()
FROM GENERATE_SERIES(1,1000000);

INSERT INTO TRANSACTION_DATA(ID,VAL)
SELECT 2,RANDOM()
FROM GENERATE_SERIES(1,1000000);
SELECT * FROM TRANSACTION_DATA;

CREATE or REPLACE VIEW SALES_SUMMARY AS
SELECT
ID,
COUNT(*) AS total_quantity_sold,
sum(val) AS total_sales,
count(distinct id) AS total_orders
FROM TRANSACTION_DATA
GROUP BY ID;

EXPLAIN ANALYZE
SELECT * FROM SALES_SUMMARY;

CREATE MATERIALIZED VIEW SALES_SUMM AS
SELECT
ID,
COUNT(*) AS total_quantity_sold,
sum(val) AS total_sales,
count(distinct id) AS total_orders
FROM TRANSACTION_DATA
GROUP BY ID;

EXPLAIN ANALYZE
SELECT * FROM SALES_SUMM;
```



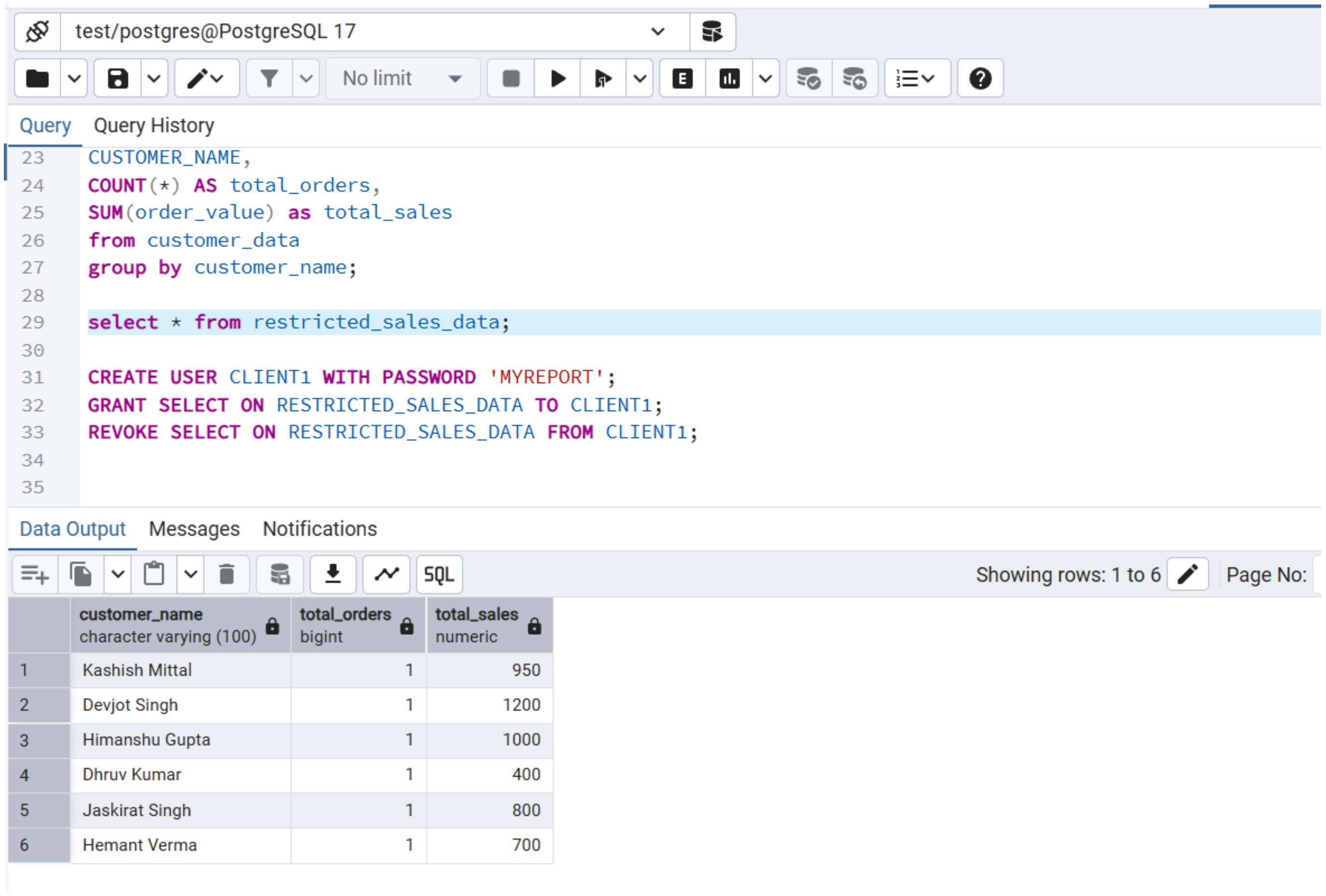
```
QUERY PLAN
text
1  Seq Scan on sales_summ_mv  (cost=0.00..20.20 rows=1020 width=52) (actual time=0.011..0.011 rows=2 loops=...
2  Planning Time: 0.671 ms
3  Execution Time: 0.022 ms
```

OUTPUT -
As we can see that the execution time using the materialized view is very less as compared to the simple view’s execution time.

HARD PROBLEM

```
CREATE TABLE customer_data (
    transaction_id SERIAL PRIMARY KEY,
    customer_name VARCHAR(100),
    email VARCHAR(100),
    phone VARCHAR(15),
    payment_info VARCHAR(50),  -- sensitive
    order_value DECIMAL,
    order_date DATE DEFAULT CURRENT_DATE
);

INSERT INTO customer_data (customer_name, email, phone, payment_info, order_value)
VALUES

('Himanshu Gupta', 'himanshu@example.com', '9876543210', '1234-5678-9012-3456', 1000),
('Jaskirat Singh', 'jaskirat@example.com', '9123456789', '2345-6789-0123-4567', 800),
('Devjot Singh', 'devjot@example.com', '9988776655', '3456-7890-1234-5678', 1200),
('Kashish Mittal', 'kashish@example.com', '9012345678', '4567-8901-2345-6789', 950),
('Dhruv Kumar', 'dhruv@example.com', '9876501234', '5678-9012-3456-7890', 400),
('Hemant Verma', 'hemant@example.com', '9765432109', '6789-0123-4567-8901', 700);

CREATE OR REPLACE VIEW RESTRICTED_SALES_DATA AS
SELECT
CUSTOMER_NAME,
COUNT(*) AS total_orders,
SUM(order_value) as total_sales
from customer_data
group by customer_name;

select * from restricted_sales_data;

CREATE USER CLIENT1 WITH PASSWORD MYPASSWORD;
GRANT SELECT ON RESTRICTED_SALES_DATA TO CLIENT1;
REVOKE SELECT ON RESTRICTED_SALES_DATA FROM CLIENT1;
```

OUTPUT



test/postgres@PostgreSQL 17

No limit

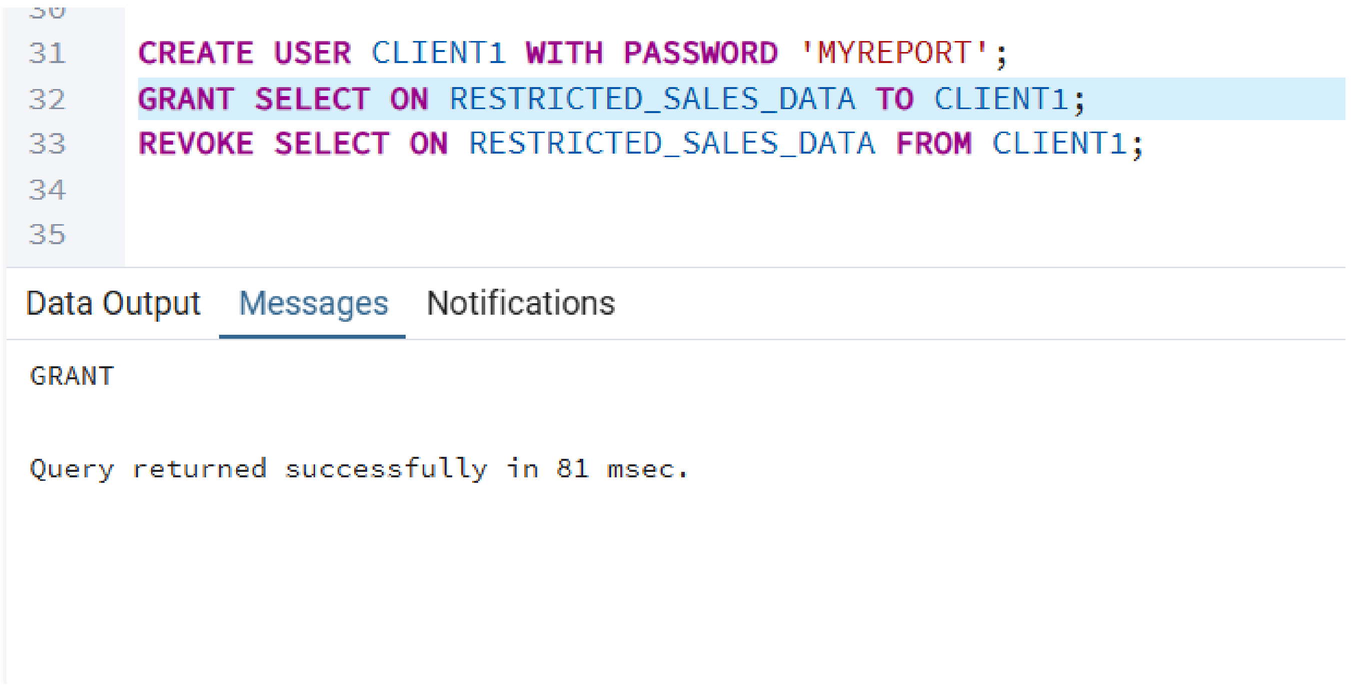
Query | Query History

```
CUSTOMER_NAME,
COUNT(*) AS total_orders,
SUM(order_value) as total_sales
from customer_data
group by customer_name;

select * from restricted_sales_data;

CREATE USER CLIENT1 WITH PASSWORD 'MYREPORT';
GRANT SELECT ON RESTRICTED_SALES_DATA TO CLIENT1;
REVOKE SELECT ON RESTRICTED_SALES_DATA FROM CLIENT1;
```

Data Output | Messages | Notifications

Showing rows: 1 to 6 Page No:

	customer_name character varying (100)	total_orders bigint	total_sales numeric
1	Kashish Mittal	1	950
2	Devjot Singh	1	1200
3	Himanshu Gupta	1	1000
4	Dhruv Kumar	1	400
5	Jaskirat Singh	1	800
6	Hemant Verma	1	700



```
CREATE USER CLIENT1 WITH PASSWORD 'MYREPORT';
GRANT SELECT ON RESTRICTED_SALES_DATA TO CLIENT1;
REVOKE SELECT ON RESTRICTED_SALES_DATA FROM CLIENT1;
```

Data Output | Messages | Notifications

```
GRANT

Query returned successfully in 81 msec.
```