Experiment 5

Student Name: Diksha	UID: 23BCS10994
Branch: CSE	Section/Group: KRG_2B
Semester: 5th	Date : 22/09/25
Subject Name: ADBMS	Subject Code: 23CSP-333

1. Aim:

A) Medium Level:

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.

B) Hard Level:

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)

2. Objective:

Medium-Level Problem:

- **Data Generation:** Generate 1 million transaction records per ID in the transaction_data table using PostgreSQL functions generate_series() and random() to simulate realistic sales data.
- **View Creation:** Create a normal view to summarize sales metrics such as total_quantity_sold, total_sales, and total_orders.
- **Performance Comparison:** Compare the execution time and query performance between the normal view and the materialized view to demonstrate the benefits of materialized views in large datasets.
- **Query Optimization:** Understand how pre-aggregation in materialized views can optimize reporting queries on large datasets.

Hard-Level Problem:

- **Restricted Views:** Create restricted or filtered views in the sales database that provide only non-sensitive aggregated data to the reporting team.

- **Audit & Compliance:** Demonstrate how database security features can enforce organizational data privacy and compliance policies.

- **Access Control:** Implement Data Control Language (DCL) commands such as GRANT and REVOKE to manage user permissions and restrict access to sensitive transactional data.

3. ADBMS script and output:

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; /*Simple view */

CREATE MATERIALIZED VIEW SALES_SUMM_MV AS

SELECT ID,

COUNT(*) AS

total_quantity_sold, sum(val)

AS total_sales, count(distinct

id) AS total_orders

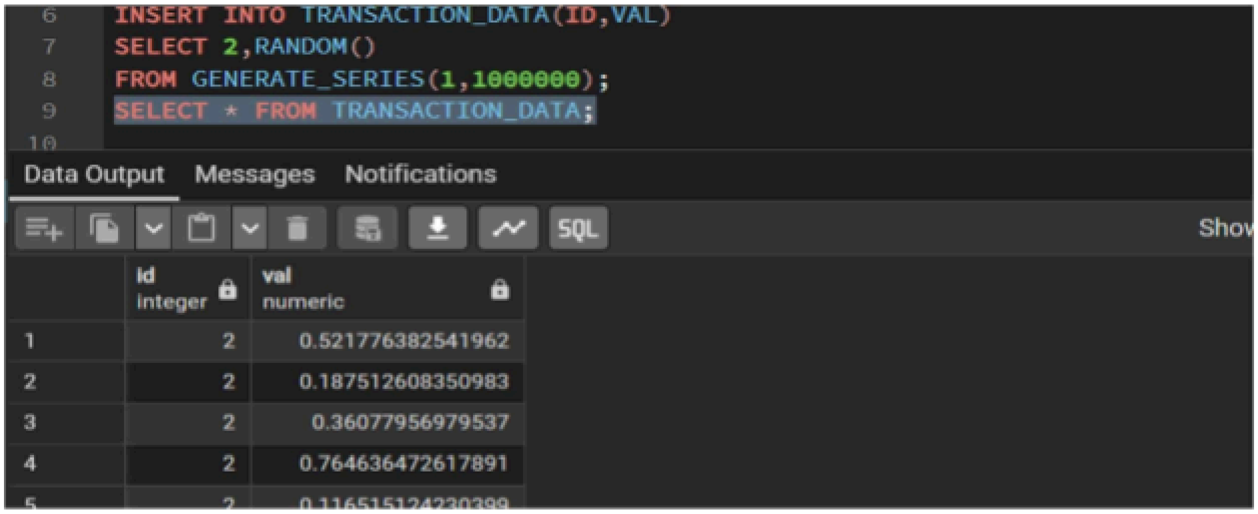
FROM TRANSACTION_DATA

GROUP BY ID;

EXPLAIN ANALYZE

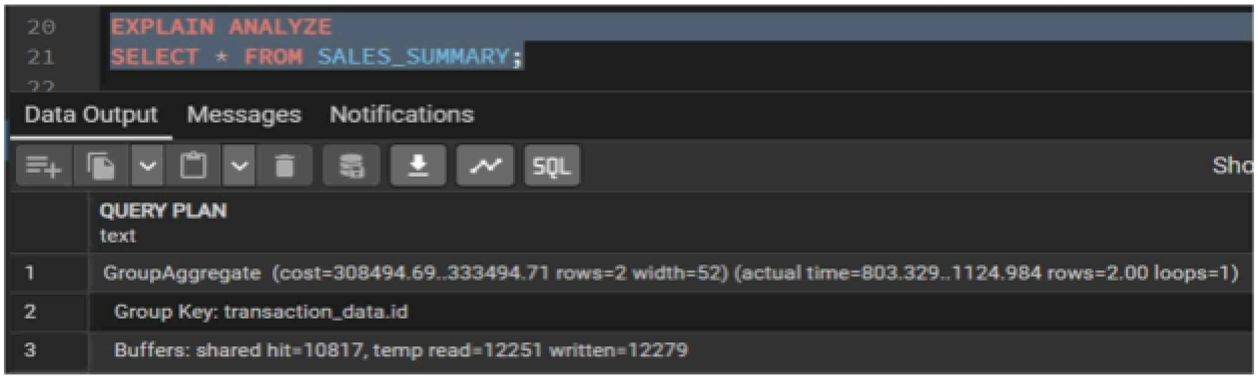
SELECT * FROM SALES_SUMM_MV;

OUTPUT:-



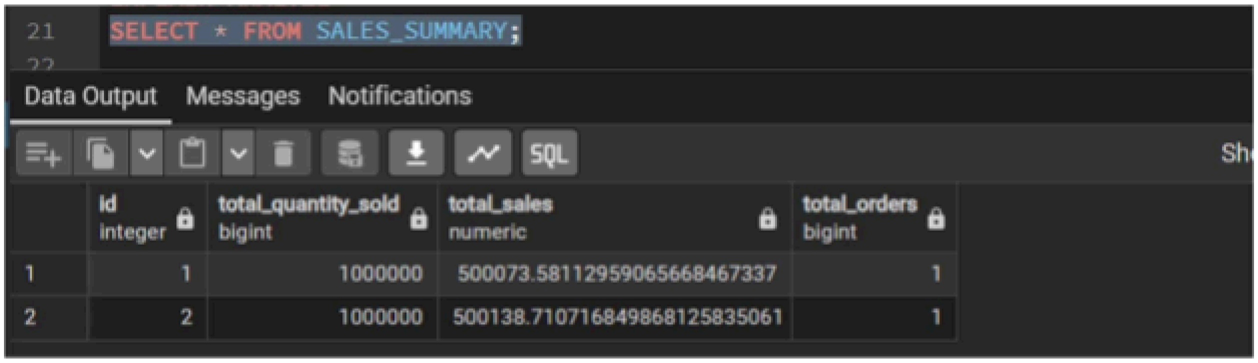
```
INSERT INTO TRANSACTION_DATA(ID,VAL)
SELECT 2,RANDOM()
FROM GENERATE_SERIES(1,1000000);
SELECT * FROM TRANSACTION_DATA;
```

	id integer	val numeric
1	2	0.521776382541962
2	2	0.187512608350983
3	2	0.36077956979537
4	2	0.764636472617891
5	2	0.116515124230399



```
EXPLAIN ANALYZE
SELECT * FROM SALES_SUMMARY;
```

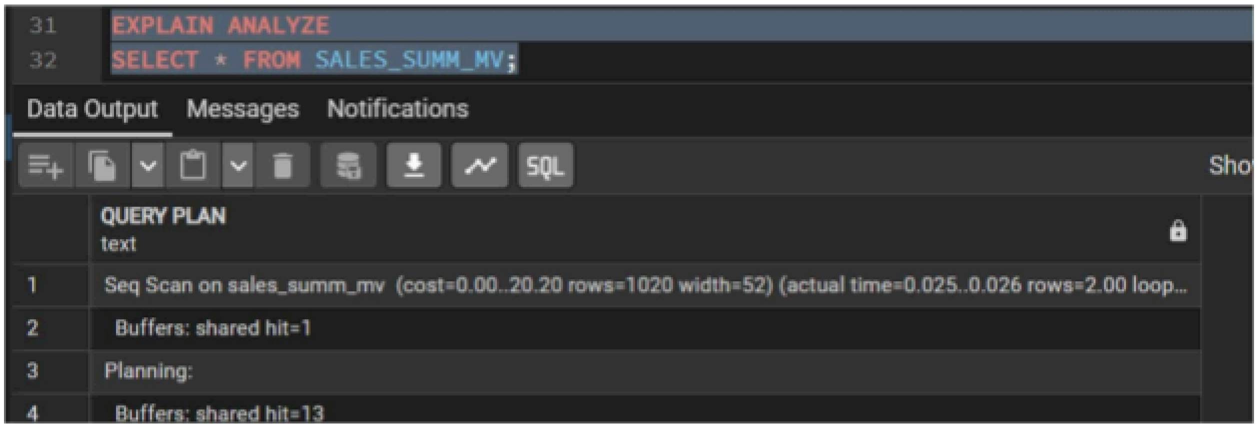
	QUERY PLAN text
1	GroupAggregate (cost=308494.69..333494.71 rows=2 width=52) (actual time=803.329..1124.984 rows=2.00 loops=1)
2	Group Key: transaction_data.id
3	Buffers: shared hit=10817, temp read=12251 written=12279



```
SELECT * FROM SALES_SUMMARY;
```

Data Output Messages Notifications

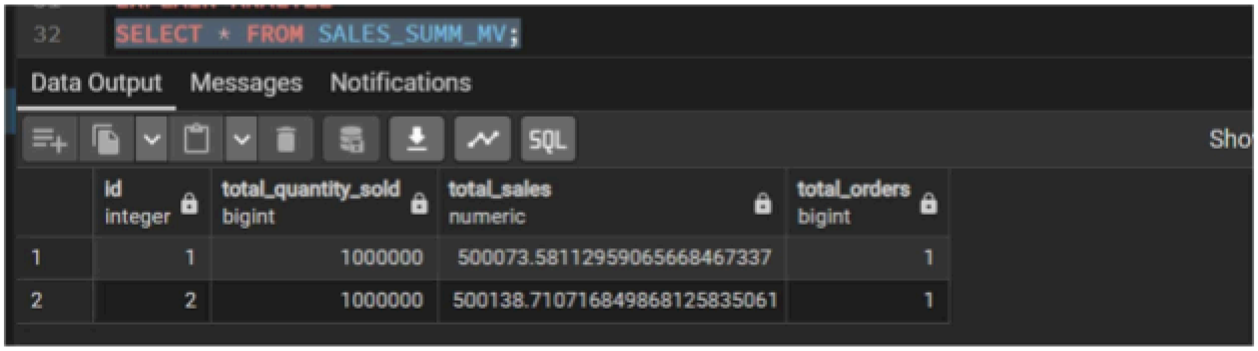
	id integer	total_quantity_sold bigint	total_sales numeric	total_orders bigint
1	1	1000000	500073.5811295906566846 7337	1
2	2	1000000	500138.7107168498681258 35061	1



```
EXPLAIN ANALYZE
SELECT * FROM SALES_SUMM_MV;
```

Data Output Messages Notifications

	QUERY PLAN text
1	Seq Scan on sales_summ_mv (cost=0.00..20.20 rows=1020 width=52) (actual time=0.025..0.026 rows=2.00 loop...
2	Buffers: shared hit=1
3	Planning:
4	Buffers: shared hit=13



```
SELECT * FROM SALES_SUMM_MV;
```

Data Output Messages Notifications

	id integer	total_quantity_sold bigint	total_sales numeric	total_orders bigint
1	1	1000000	500073.5811295906566846 7337	1
2	2	1000000	500138.7107168498681258 35061	1

Hard Level Problem:

CREATE TABLE customer_data

(transaction_id SERIAL PRIMARY

KEY, customer_name

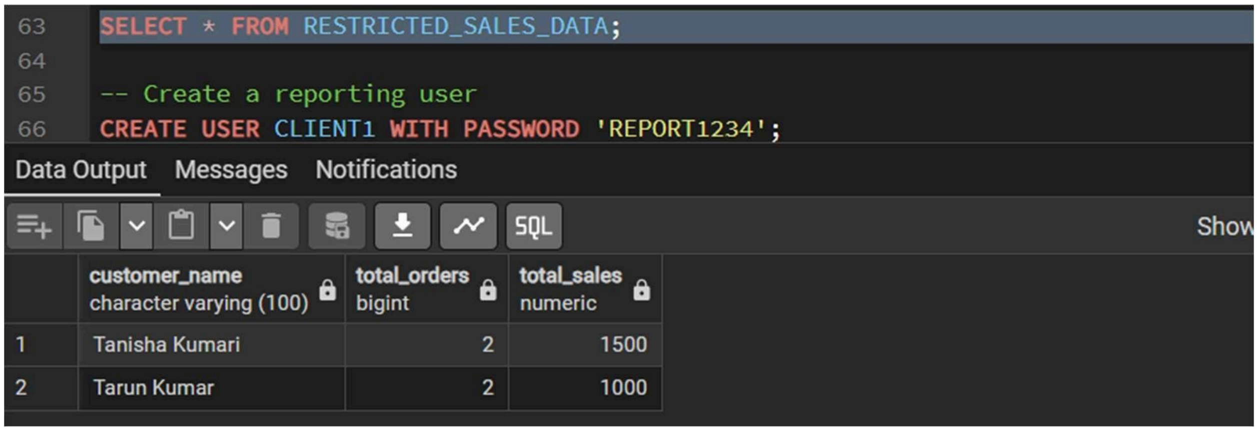
VARCHAR(100), email

VARCHAR(100), phone

```
VARCHAR(15), payment_info
VARCHAR(50), order_value
DECIMAL,
    order_date DATE DEFAULT CURRENT_DATE
);
INSERT INTO customer_data (customer_name, email, phone, payment_info,
order_value)
VALUES
('Tanisha Kumari', 'tanisha.pankajj@gmail.com', '987654321',
'1234-5678-9012-3456', 500),
('Tanisha Kumari', 'tanisha.pankajj@gmail.com', '987654321',
'1234-5678-9012-3456', 1000),
('Tarun Kumar', 'tarun3008@gmail.com', '123456789', '9876-5432-1098-7654',
700),
('Tarun Kumar', 'tarun3008@gmail.com', '123456789', '9876-5432-1098-7654',
300);

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 'REPORT1234';
GRANT SELECT ON RESTRICTED_SALES_DATA TO CLIENT1;
```

REVOKE SELECT ON RESTRICTED_SALES_DATA FROM CLIENT1;

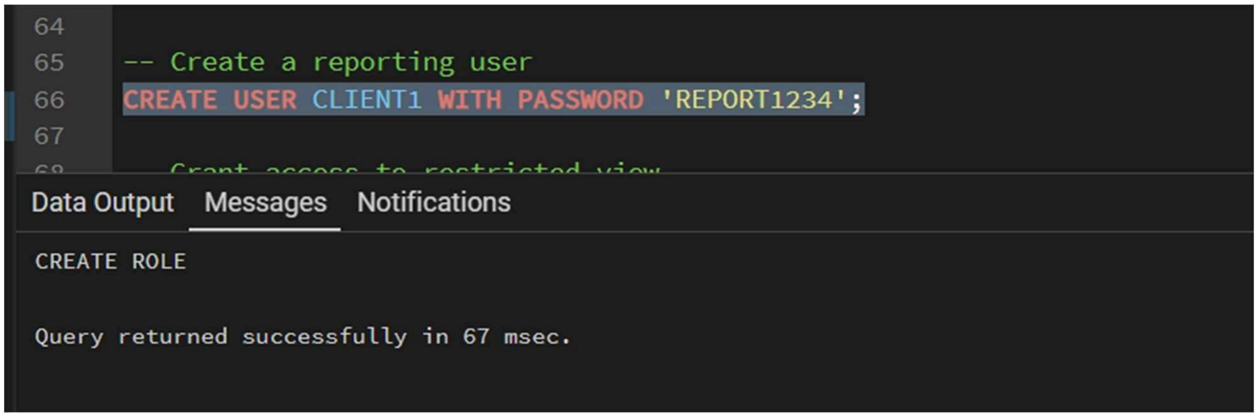
OUTPUT:



```
SELECT * FROM RESTRICTED_SALES_DATA;

-- Create a reporting user
CREATE USER CLIENT1 WITH PASSWORD 'REPORT1234';
```

Data Output Messages Notifications

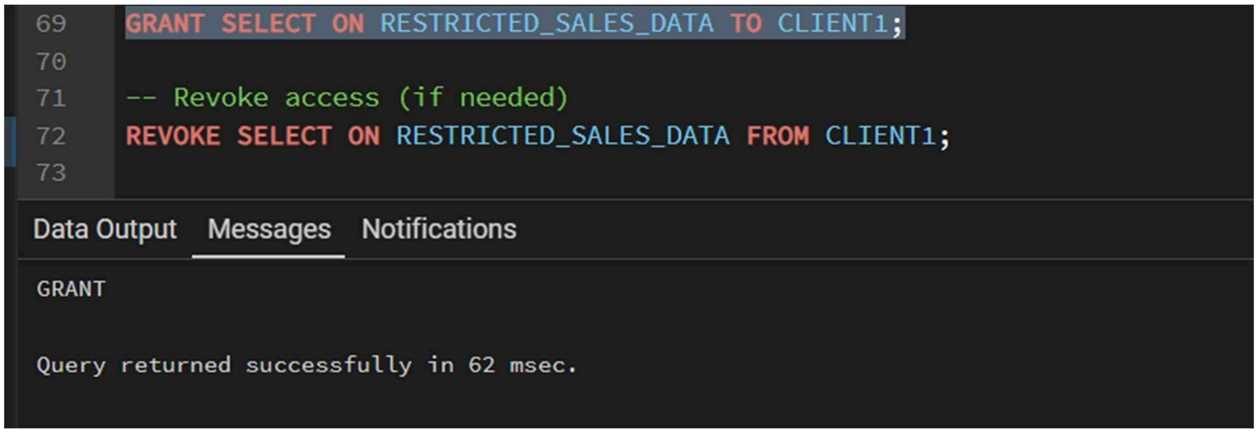
	customer_name character varying (100)	total_orders bigint	total_sales numeric
1	Tanisha Kumari	2	1500
2	Tarun Kumar	2	1000



```

-- Create a reporting user
CREATE USER CLIENT1 WITH PASSWORD 'REPORT1234';

```

Data Output Messages Notifications

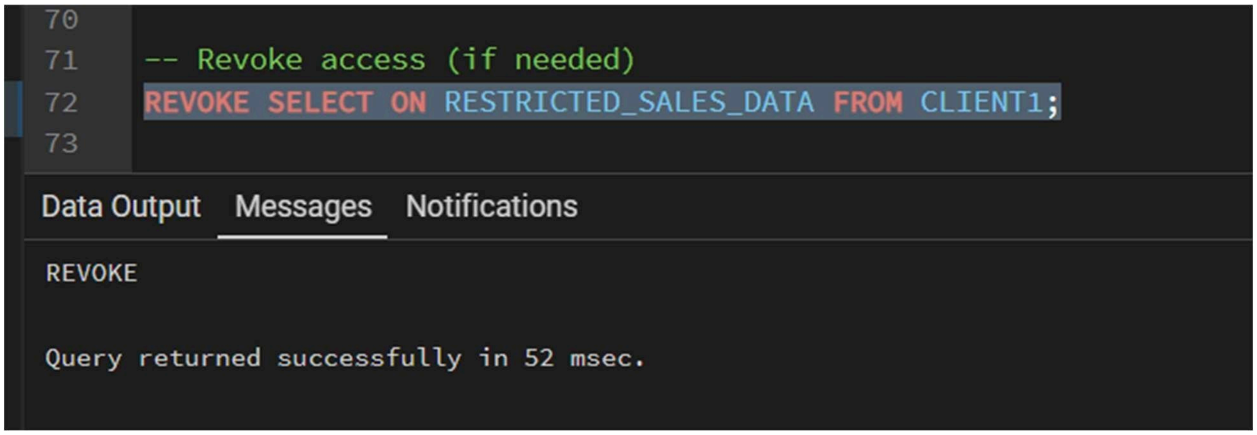
```
CREATE ROLE

Query returned successfully in 67 msec.
```



```
GRANT SELECT ON RESTRICTED_SALES_DATA TO CLIENT1;

-- Revoke access (if needed)
REVOKE SELECT ON RESTRICTED_SALES_DATA FROM CLIENT1;

```

Data Output Messages Notifications

```
GRANT

Query returned successfully in 62 msec.
```



```sql
-- Revoke access (if needed)
REVOKE SELECT ON RESTRICTED_SALES_DATA FROM CLIENT1;
```

```
REVOKE

Query returned successfully in 52 msec.
```