Experiment - 5

Student Name: HarshVardhan **UID:** 23BCS10363
Branch: BE-CSE **Section/Group:** KRG_1A
Semester: 5th **Date of Performance:** 22.09.2025
Subject Name: Advanced Database and Management System
Subject Code: 23CSP-333

Aim:

Medium-Problem:

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.

Hard-Problem

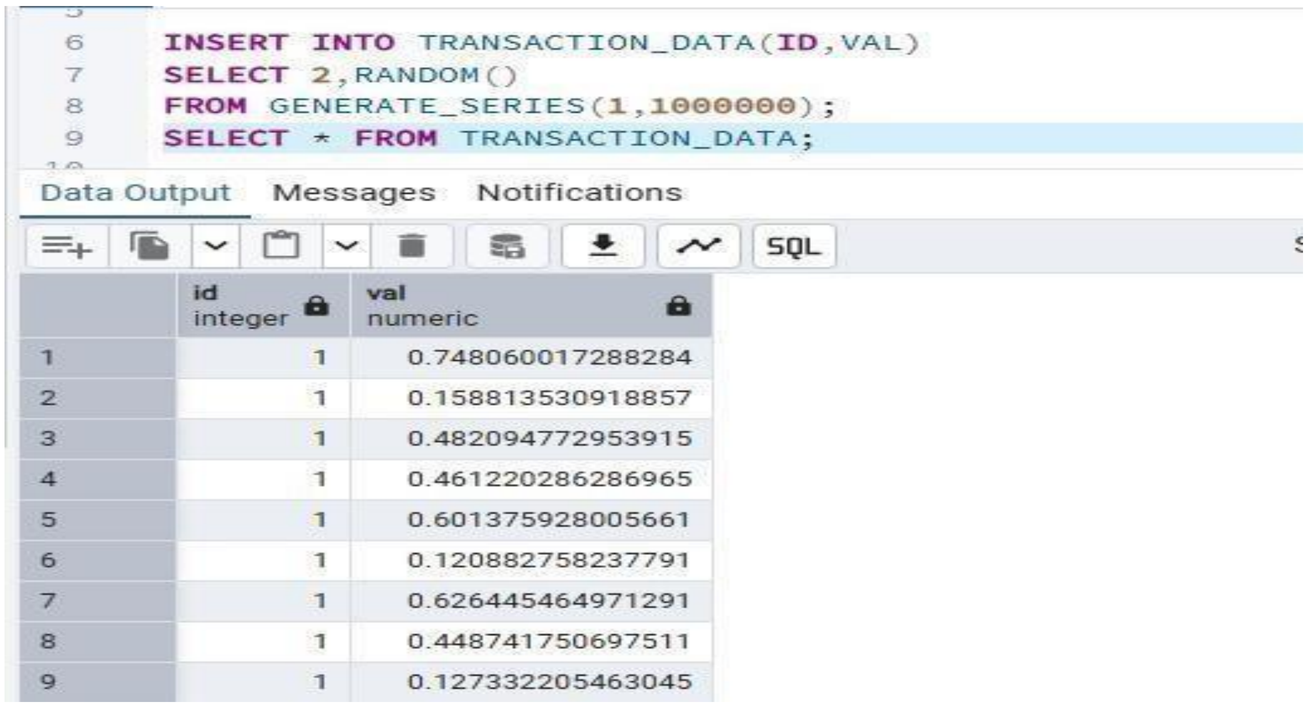
Create restricted views in the sales database to provide summarized, non-sensitive data to the reporting team, and control access using DCL commands(GRANT and REVOKE).

1. SQL QUERY AND OUTPUTS -

-------------------------------MEDIUM LEVEL PROBLEM-------------------------------------

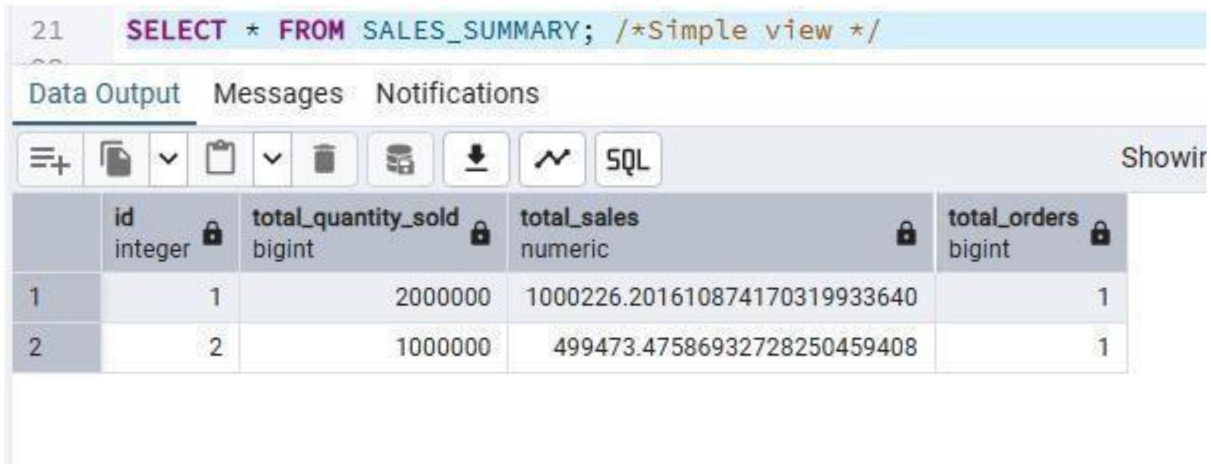
```
Create table TRANSACTION_DATA(id int,val decimal);
INSERT INTO TRANSACTION_DATA(ID,VAL)
SELECT 1,RANDOM()
FROM GENERATE_SERIES(1,1000000);

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;
```



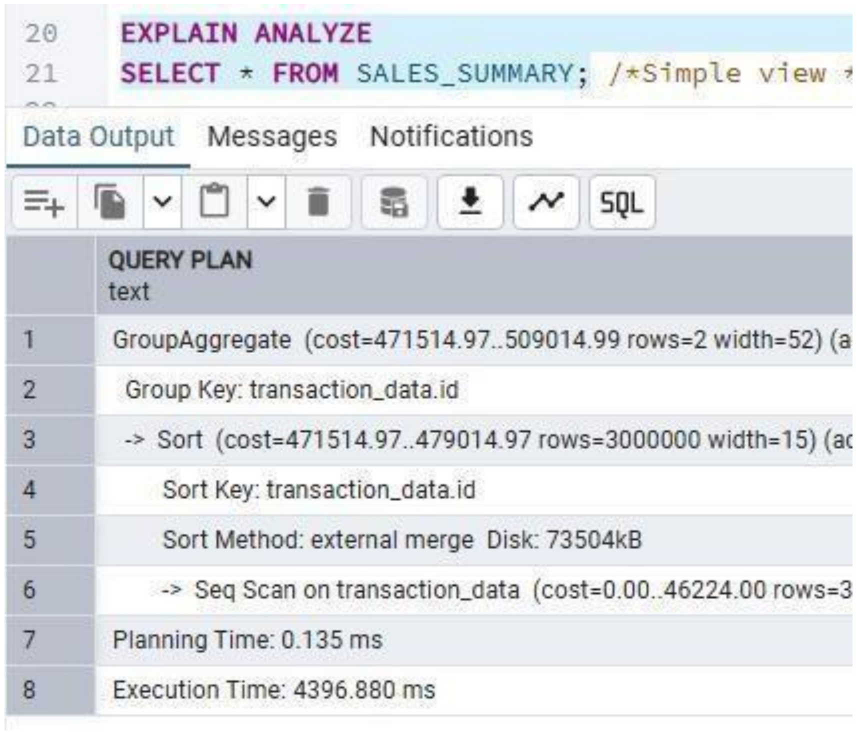
```
INSERT INTO TRANSACTION_DATA(ID,VAL)
SELECT 2,RANDOM()
FROM GENERATE_SERIES(1,1000000);
SELECT * FROM TRANSACTION_DATA;
```

	id integer	val numeric
1	1	0.748060017288284
2	1	0.158813530918857
3	1	0.482094772953915
4	1	0.461220286286965
5	1	0.601375928005661
6	1	0.120882758237791
7	1	0.626445464971291
8	1	0.448741750697511
9	1	0.127332205463045



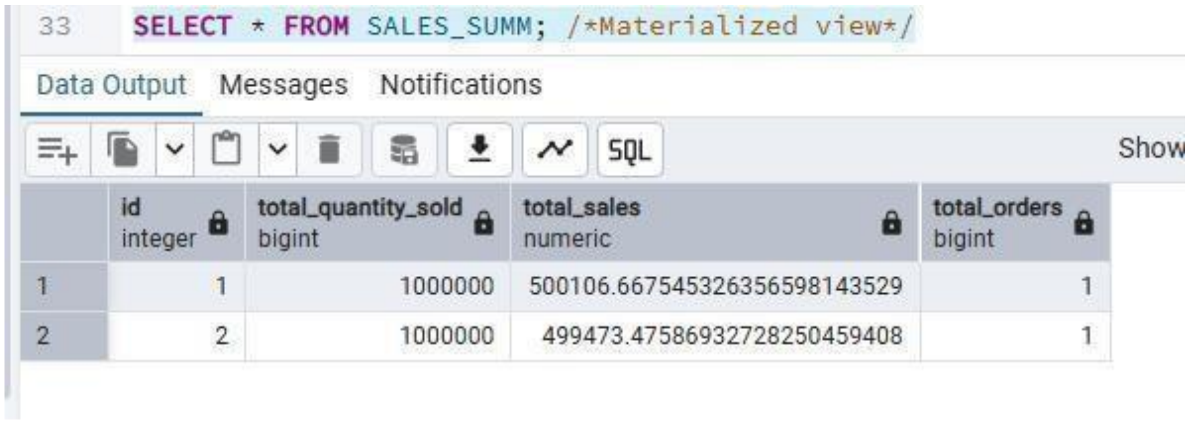
```sql
SELECT * FROM SALES_SUMMARY; /*Simple view */
```

	id integer	total_quantity_sold bigint	total_sales numeric	total_orders bigint
1	1	2000000	1000226.201610874170319933640	1
2	2	1000000	499473.47586932728250459408	1



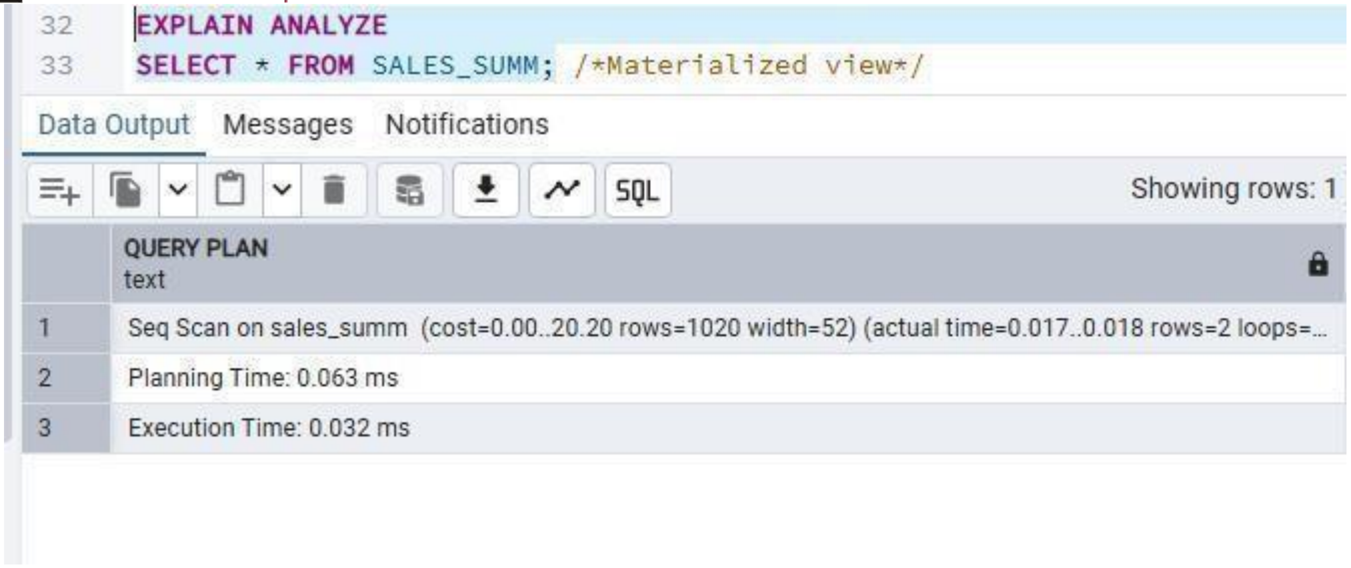
```sql
EXPLAIN ANALYZE
SELECT * FROM SALES_SUMMARY; /*Simple view *
```

	QUERY PLAN text
1	GroupAggregate (cost=471514.97..509014.99 rows=2 width=52) (a
2	Group Key: transaction_data.id
3	-> Sort (cost=471514.97..479014.97 rows=3000000 width=15) (ac
4	Sort Key: transaction_data.id
5	Sort Method: external merge Disk: 73504kB
6	-> Seq Scan on transaction_data (cost=0.00..46224.00 rows=3
7	Planning Time: 0.135 ms
8	Execution Time: 4396.880 ms



```sql
SELECT * FROM SALES_SUMM; /*Materialized view*/
```

	id integer	total_quantity_sold bigint	total_sales numeric	total_orders bigint
1	1	1000000	500106.667545326356598143529	1
2	2	1000000	499473.47586932728250459408	1



```
EXPLAIN ANALYZE
SELECT * FROM SALES_SUMM; /*Materialized view*/
```

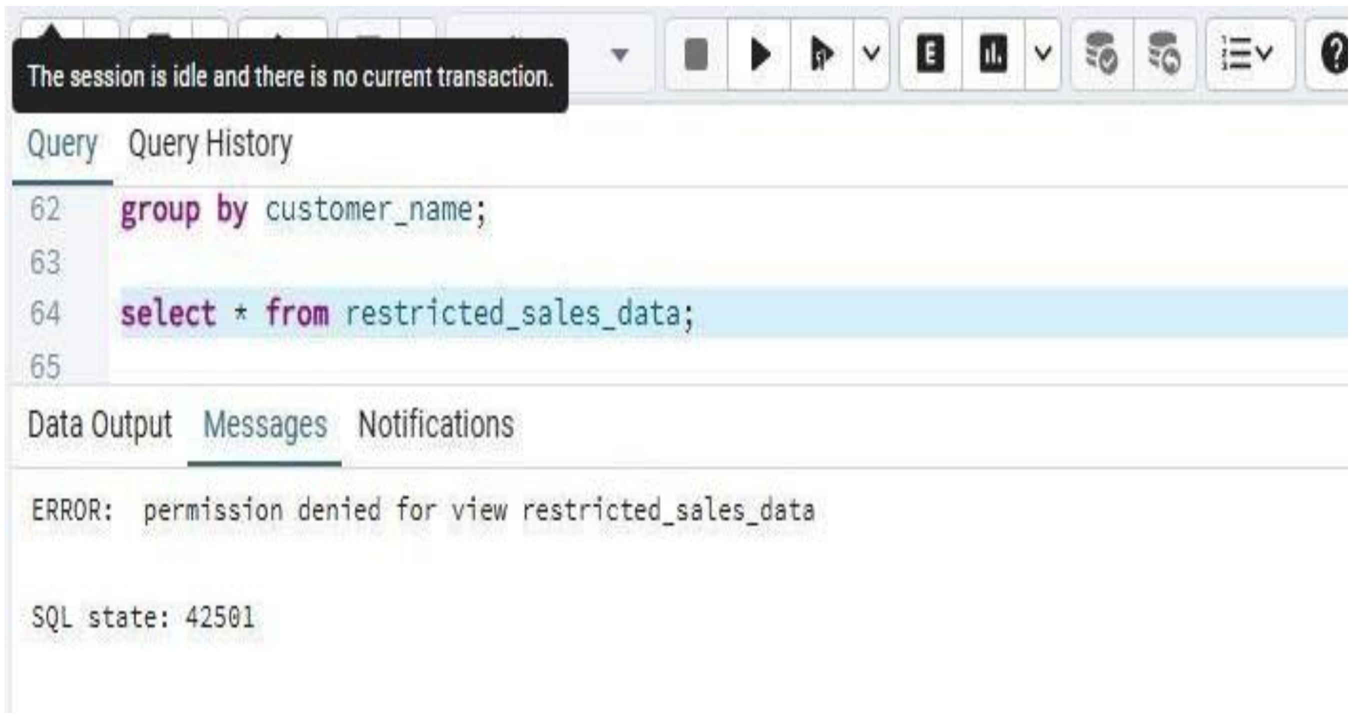
```
QUERY PLAN
text
1  Seq Scan on sales_summ  (cost=0.00..20.20 rows=1020 width=52) (actual time=0.017..0.018 rows=2 loops=...
2  Planning Time: 0.063 ms
3  Execution Time: 0.032 ms
```

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 ---

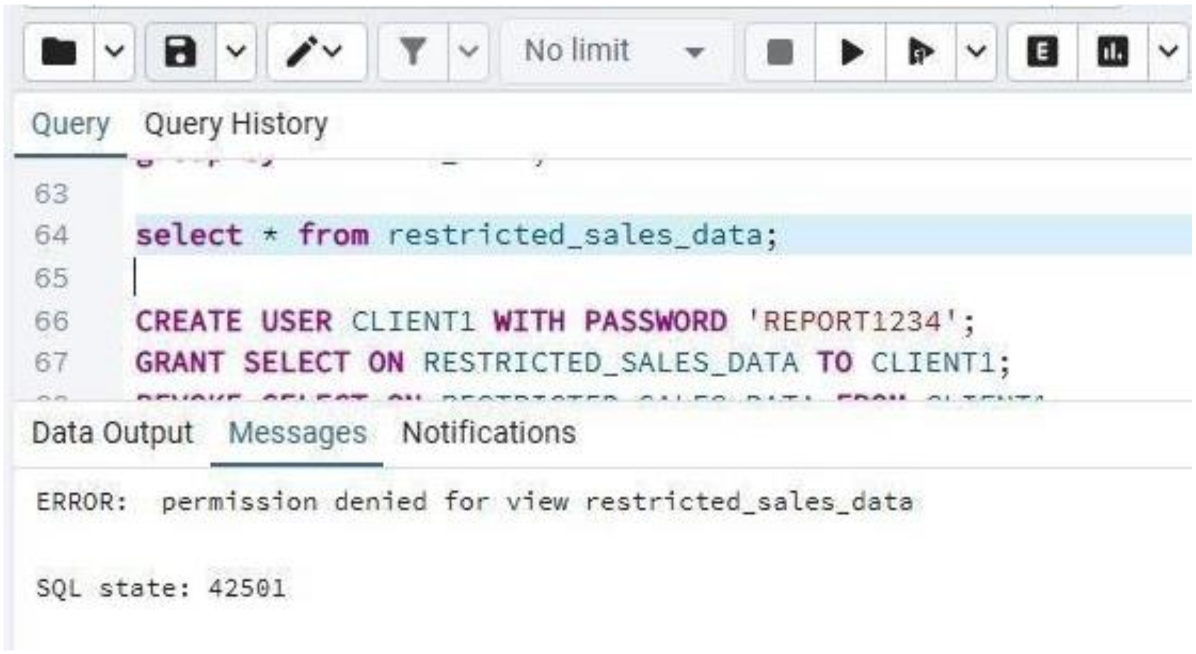
```
CREATE TABLE customer_data ( transaction_id
    SERIAL PRIMARY KEY, customer_name
    VARCHAR(100), email VARCHAR(100), phone
    VARCHAR(15),
    payment_info VARCHAR(50), -- sensitive order_value
    DECIMAL, order_date DATE DEFAULT
    CURRENT_DATE
);

-- Insert sample data
INSERT INTO customer_data (customer_name, email, phone, payment_info, order_value) VALUES
('John ', 'John@example.com', '9040122324', '1234-5678-9012-3456', 500),
('John ', 'John@example.com', '9040122324', '1234-5678-9012-3456', 1000),
('Alice Singh', 'Alice@example.com', '9876543210', '9876-5432-1098-7654', 700),
('Alice Singh', 'Alice@example.com', '9876543210', '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;
```



```
group by customer_name;

select * from restricted_sales_data;
```

```
ERROR:  permission denied for view restricted_sales_data

SQL state: 42501
```



```
select * from restricted_sales_data;

CREATE USER CLIENT1 WITH PASSWORD 'REPORT1234';
GRANT SELECT ON RESTRICTED_SALES_DATA TO CLIENT1;
```

```
ERROR:  permission denied for view restricted_sales_data

SQL state: 42501
```