

WORKSHEET 5

Student Name: Supriya Dutta UID: 23BCS13291

Branch: CSE(3rd Year) Section/Group: Krg-1-A

Semester: 5th Date of Performance: 25/09/25

Subject Name: ADBMS Subject Code: 23CSP-333

1. AIM:

i) Performance Benchmarking: Normal View vs. Materialized View (Medium)

- 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.

ii)Views: Securing Data Access with Views and Role-Based Permissions (Hard)

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).

2. Tools Used: SQL Server Management Studio

Solutions:

Q1) --EXP 05(MEDIUM LEVEL)

CREATE TABLE transaction_data (
id INT,
value INT

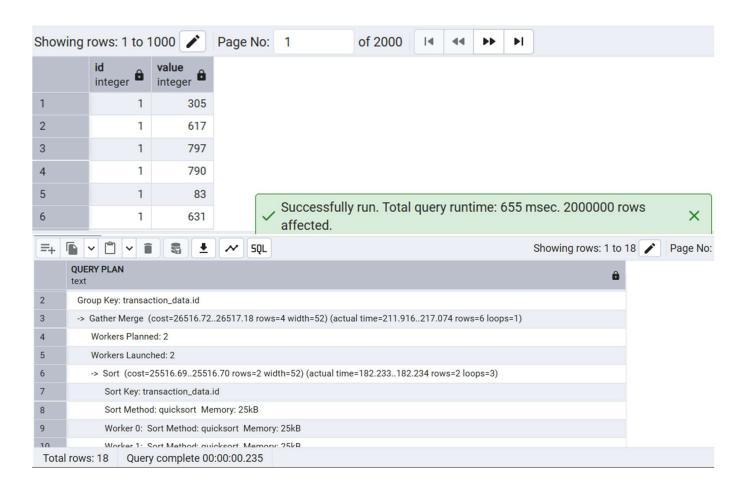
```
-- For id = 1
INSERT INTO transaction data (id, value)
SELECT 1, random() * 1000
FROM generate series(1, 1000000);
-- For id = 2
INSERT INTO transaction data (id, value)
SELECT 2, random() * 1000
FROM generate series(1, 1000000);
SELECT *FROM transaction data;
--WITH NORMAL VIEW
CREATE OR REPLACE 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;
--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;
REFRESH MATERIALIZED VIEW sales_summary_mv;
EXPLAIN ANALYZE SELECT * FROM sales summary view;
EXPLAIN ANALYZE SELECT * FROM sales_summary_mv;
Q2) --EXP(05) HARD LEVEL
CREATE TABLE customer master (
customer id VARCHAR(5) PRIMARY KEY,
full name VARCHAR(50) NOT NULL,
phone VARCHAR(15),
```

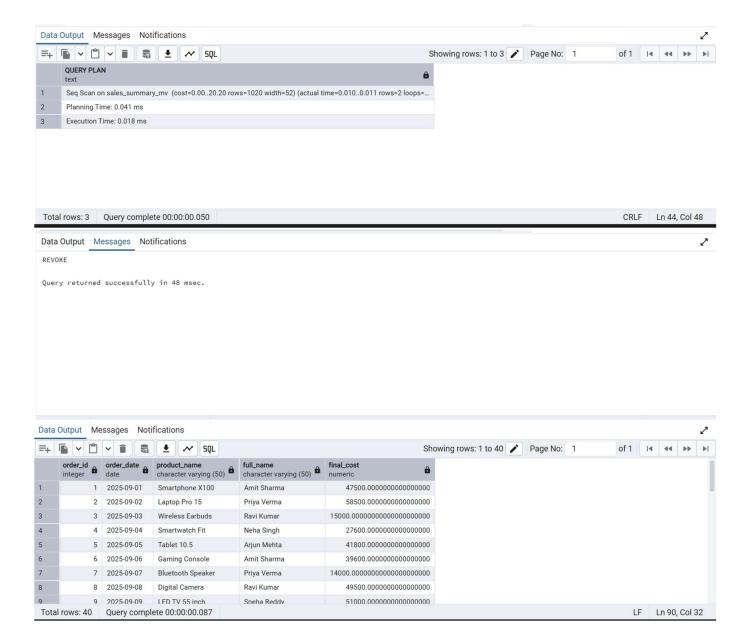
```
email VARCHAR(50),
city VARCHAR(30)
);
CREATE TABLE product catalog (
product id VARCHAR(5) PRIMARY KEY,
product name VARCHAR (50) NOT NULL,
brand VARCHAR(30),
unit price NUMERIC(10,2) NOT NULL
);
CREATE TABLE sales orders (
order id SERIAL PRIMARY KEY,
product id VARCHAR(5) REFERENCES product catalog(product 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');
-- 1. CREATE VIEW (only summarized, non-sensitive data)
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;
-- 2. TESTING THE VIEW
SELECT * FROM vW ORDER SUMMARY;
-- 3. CREATE A USER / ROLE FOR THE REPORTING TEAM
-- (syntax may vary depending on DB: PostgreSQL, SQL Server, MySQL etc.)
-- Example for PostgreSQL:
CREATE ROLE alok LOGIN PASSWORD 'alok123';
```

- -- Example for SQL Server:
- -- CREATE LOGIN alok WITH PASSWORD = 'alok123';
- -- CREATE USER alok FOR LOGIN alok;
- -- 4. GRANT ACCESS TO THE VIEW ONLY GRANT SELECT ON vW_ORDER_SUMMARY TO alok;
- -- 5. IF YOU WANT TO REMOVE ACCESS LATER REVOKE SELECT ON vW_ORDER_SUMMARY FROM alok;

3. Output:





4. Learning Outcomes:

i) Understanding View Creation & Data Security

• Learned how to create database **views** to present summarized and non-sensitive data while hiding underlying table details, improving **data security** for external/reporting users.

ii) Role & Access Control

 Gained practical knowledge of how to create users/roles and grant or revoke permissions on views to ensure controlled access in multi-user database environments.

iii) Performance Benchmarking of Views

 Understood the difference between Normal Views (executed dynamically) and Materialized Views (pre-computed and stored), and analyzed their execution time using EXPLAIN ANALYZE.

iv) Efficient Data Aggregation

• Learned to perform **aggregation operations** (SUM, COUNT, AVG) on large datasets and summarize transactions for analytical reporting.

v) Practical Exposure to Large Dataset Handling

• Acquired experience in generating and managing large-scale synthetic data (using generate_series() and random()) and applying optimization techniques for query performance improvement.