Assignments – Materialized Views in PostgreSQL

Scenario 1 – Sales Summary Materialized View (Basic)

1. Create a table sales with the following columns:

```
sale_id SERIAL PRIMARY KEY,
sale_date DATE,
customer_id INT,
product_id INT,
quantity INT,
price NUMERIC(10,2)
```

1. Insert at least **20 rows** of test data.
2. Create a **materialized view** sales_summary_mv showing:
    a. sale_date
    b. total sales amount per date (quantity * price)
    c. total number of orders per date
3. Refresh the materialized view manually.
4. Query the materialized view to verify results.

Scenario 2 – Monthly Sales by Product Category

1. Create a table products with product_id, product_name, and category.
2. Join sales with products to create a **materialized view** monthly_sales_category_mv that shows:
    - month (YYYY-MM)
    - category
    - total sales amount
3. Refresh the materialized view and query the top 3 categories per month.

Scenario 3 – Refresh Performance Check

1. Create a **materialized view** customer_sales_mv that shows:
    - customer_id
    - total orders placed
    - last purchase date
2. Insert additional sales data into the sales table.
3. Query the customer_sales_mv **before** and **after** a manual refresh to demonstrate stale data behavior.

Scenario 4 – Indexed Materialized View for Fast Queries

1. Create a **materialized view** daily_revenue_mv that shows:
    - sale_date
    - total revenue
2. Create an **index** on the sale_date column of the materialized view.
3. Compare query performance **with** and **without** the index using EXPLAIN ANALYZE.

Scenario 5 – Dependent Materialized Views

1. Use sales_summary_mv from Scenario 1 as the **source** for a new **materialized view** high_sales_days_mv that only includes days with total sales over ₹1,00,000.
2. Refresh sales_summary_mv first, then refresh high_sales_days_mv to ensure data consistency.