SQL Test - Views in PostgreSQL

Part 1: Creating and Querying Views

1. Create a Basic View

Create a view called `active_customers` that includes only customers with the `active` status from a table `customers`.

2. Add Additional Filters to the View

Modify or recreate the `active_customers` view to include only active customers who reside in specific cities (`city IN ('São Paulo', 'Rio de Janeiro')`).

3. View with Selected Columns

Create a view called `customers_summary` that shows only the columns `id`, `name`, and `city` from the `customers` table.

Part 2: Views with Transformations

4. View with Calculation

From a table `orders` (containing `order_id`, `customer_id`, `total_value`, `order_date`), create a view called `orders_with_tax` that includes a new column named `value_with_tax` (considering a 10% increase).

5. View with Table Join

Consider the tables `customers` and `orders`. Create a view called `customer_orders_summary` that shows `customer_id`, `name`, `city`, and the total number of orders placed by each customer.

6. View with Advanced Filtering

SQL Test - Views in PostgreSQL

Create a view called `recent_orders` that lists the orders placed in the last 30 days.

Part 3: Modification and Deletion

7. Indirect Update

Consider a view called `available_stock` based on a table `products`. Test the ability to perform updates on the table through this view. Explain why the modification is not allowed if applicable.

8. Delete a View

Drop a view you created, ensuring that it does not affect the original tables' data.

Part 4: Advanced Challenges

9. View with Window Functions

Create a view called `customers_ranking` that ranks customers based on the total value of orders placed (use a window function like `RANK`).

10. Materialized View

Create a materialized view called `orders_analysis` to summarize the number of orders per month. Refresh it after inserting new data into the orders table.

Part 5: Advanced Exercises

11. View with Subquery

Create a view called `top_customers` that shows the customers with the top 5 highest total order values. Use a subquery to calculate the total order value per customer and sort the results.

SQL Test - Views in PostgreSQL

12. View with Grouping and Filters

Create a view called `orders_by_status` that groups orders by status (e.g., `pending`, `shipped`, `completed`), counts the number of orders in each status, and filters to show only statuses with more than 10 orders.

13. View with Joins and Aggregation

From the tables `customers`, `orders`, and `products`, create a view called `sales_summary` that shows the total sales per customer, including the columns `customer_id`, `name`, and `total_sales_value`. Use a `JOIN` between the tables and an aggregation function like `SUM()`.

14. View with Case When

Create a view called `orders_status` that shows the status of each order, with an additional column called `delayed_status`, indicating whether the order is 'delayed' (when `order_date` is earlier than the current date and the status is 'pending').

15. View with Temporary Tables

Create a view that uses a temporary table. The temporary table should store an intermediate calculation, such as daily sales totals, before presenting the final view showing total sales per month.