1. What is Window Frame in MySQL Window Functions?

a. A range of rows that define the subset of data to be used in a calculation.
b. A group of columns that define the data used in a calculation.
c. A set of criteria that define which rows are included in a calculation.
d. A set of functions that define how data is calculated.

Answer: a
Explanation:
In the context of MySQL Window Functions, a "Window Frame" refers to a range of rows that define the subset of data to be used in a calculation. This is the correct definition of a Window Frame. It determines which rows are included in the calculation.

2. Consider a table called "sales" with columns "id", "date", "amount", and "region". Which of the following SQL queries will calculate the average amount of sales for each region, including the previous and next rows in the result set?

a. SELECT region, AVG(amount) OVER (ORDER BY region ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) as avg_sales FROM sales;
b. SELECT region, AVG(amount) OVER (PARTITION BY region ORDER BY date ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) as avg_sales FROM sales;
c. SELECT region, AVG(amount) OVER (ORDER BY date ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) as avg_sales FROM sales;
d. SELECT region, AVG(amount) OVER (PARTITION BY region ORDER BY region ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) as avg_sales FROM sales;

Answer : b

Explanation: The correct query calculates the average amount of sales for each region(PARTITION on region), including the previous and next rows. It uses the ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING clause. This clause considers the rows immediately before and after the current row when calculating the average. This allows you to perform calculations on a window of data around each row.

3. Consider a table called "employees" with columns "id", "name", "department", and "salary". Which of the following SQL queries will calculate the running total of salary for each employee, ordered by department?

a. SELECT name, salary, SUM(salary) OVER (PARTITION BY department ORDER BY salary) as running_total FROM employees;

b. SELECT name, salary, SUM(salary) OVER (PARTITION BY department ORDER BY name) as running_total FROM employees;
c. SELECT name, salary, SUM(salary) OVER (ORDER BY department, name) as running_total FROM employees;
d. SELECT name, salary, SUM(salary) OVER (PARTITION BY name ORDER BY department) as running_total FROM employees;

Answer: c

Explanation: This query calculates the running total of salary for each employee, ordered by department. PARTITION is not needed as the question is not asked for running salary sum department-wise.

4. Consider a table called "products" with columns "id", "name", "category", and "price". Which of the following SQL queries will calculate the percentage of the total revenue generated by each category of products?

a. SELECT category, price, SUM(price) OVER (PARTITION BY category ORDER BY price) / SUM(SUM(price)) OVER () * 100 as percent_total_revenue FROM products;
b. SELECT category, SUM(price) OVER (PARTITION BY category) as revenue, SUM(price) / SUM(SUM(price)) OVER () * 100 as percent_total_revenue FROM products;
c. SELECT category, SUM(price) as revenue, SUM(price) / SUM(SUM(price)) OVER () * 100 as percent_total_revenue FROM products GROUP BY category;
d. SELECT category, SUM(price) OVER (PARTITION BY category) as revenue, SUM(SUM(price)) OVER () / SUM(price) * 100 as percent_total_revenue FROM products;

Answer: c

5. Consider a table called "customer_orders" with columns "id", "customer_id", "product_name", "price", and "date". Which of the following SQL queries will calculate the difference in price between each order and the previous order for each customer?

a. SELECT customer_id, product_name, price, LAG(price) OVER (PARTITION BY customer_id ORDER BY date) - price as price_difference FROM customer_orders;
b. SELECT customer_id, product_name, price, price - LAG(price) OVER (PARTITION BY customer_id ORDER BY date) as price_difference FROM customer_orders;
c. SELECT customer_id, product_name, price, price - LAG(price) OVER (PARTITION BY product_name ORDER BY date) as price_difference FROM customer_orders;

d. SELECT customer_id, product_name, price, LAG(price) OVER (PARTITION BY product_name ORDER BY date) - price as price_difference FROM customer_orders;

Answer: b

6. Consider a table called "employee_sales" with columns "employee_id", "sales_amount", and "date". Which of the following SQL queries will calculate the sum of sales for each employee over the last 7 days?

a. SELECT employee_id, SUM(sales_amount) OVER (PARTITION BY employee_id) as total_sales FROM employee_sales WHERE date BETWEEN NOW() - INTERVAL 7 DAY AND NOW();
b. SELECT employee_id, SUM(sales_amount) OVER (PARTITION BY employee_id ORDER BY date RANGE BETWEEN INTERVAL 7 DAY PRECEDING AND CURRENT ROW) as total_sales FROM employee_sales;
c. SELECT employee_id, SUM(sales_amount) OVER (PARTITION BY employee_id ORDER BY date ROWS BETWEEN 7 PRECEDING AND CURRENT ROW) as total_sales FROM employee_sales;
d. SELECT employee_id, SUM(sales_amount) OVER (PARTITION BY employee_id ORDER BY date RANGE BETWEEN 7 PRECEDING AND CURRENT ROW) as total_sales FROM employee_sales;

Answer: b

7. Consider a table called "product_sales" with columns "product_name", "sales_amount", and "date". Which of the following SQL queries will calculate the moving average of sales for each product over the last 30 days?

a. SELECT product_name, AVG(sales_amount) OVER (PARTITION BY product_name) as moving_average FROM product_sales WHERE date BETWEEN NOW() - INTERVAL 30 DAY AND NOW();
b. SELECT product_name, AVG(sales_amount) OVER (PARTITION BY product_name ORDER BY date RANGE BETWEEN INTERVAL 30 DAY PRECEDING AND CURRENT ROW) as moving_average FROM product_sales;
c. SELECT product_name, AVG(sales_amount) OVER (PARTITION BY product_name ORDER BY date ROWS BETWEEN 30 PRECEDING AND CURRENT ROW) as moving_average FROM product_sales;
d. SELECT product_name, AVG(sales_amount) OVER (PARTITION BY product_name ORDER BY date RANGE BETWEEN 30 PRECEDING AND CURRENT ROW) as moving_average FROM product_sales;

Answer: b

8. Consider a table called "order_details" with columns "order_id", "product_name", "quantity", and "price". Which of the following SQL queries will calculate the total revenue for each order?

a. SELECT order_id, SUM(quantity * price) OVER (PARTITION BY order_id) as total_revenue FROM order_details;
b. SELECT order_id, SUM(quantity * price) OVER (PARTITION BY product_name) as total_revenue FROM order_details;
c. SELECT order_id, SUM(price) OVER (PARTITION BY order_id) as total_revenue FROM order_details;
d. SELECT order_id, SUM(price) OVER (PARTITION BY product_name) as total_revenue FROM order_details;

Answer: a

9. Consider a table called "product_inventory" with columns "product_name", "quantity", and "date". Which of the following SQL queries will calculate the difference in inventory levels between each day for each product?

a. SELECT product_name, quantity, quantity - LAG(quantity) OVER (PARTITION BY product_name ORDER BY date) as inventory_difference FROM product_inventory;
b. SELECT product_name, quantity, LAG(quantity) OVER (PARTITION BY product_name ORDER BY date) - quantity as inventory_difference FROM product_inventory;
c. SELECT product_name, quantity, quantity - LAG(quantity) OVER (PARTITION BY date ORDER BY product_name) as inventory_difference FROM product_inventory;
d. SELECT product_name, quantity, LAG(quantity) OVER (PARTITION BY date ORDER BY product_name) - quantity as inventory_difference FROM product_inventory;

Answer: a

10. Consider a table called "customer_transactions" with columns "customer_id", "transaction_amount", and "date". Which of the following SQL queries will calculate the running total of transactions for each customer?

a. SELECT customer_id, transaction_amount, SUM(transaction_amount) OVER (PARTITION BY customer_id) as running_total FROM customer_transactions;
b. SELECT customer_id, transaction_amount, SUM(transaction_amount) OVER (ORDER BY date) as running_total FROM customer_transactions;
c. SELECT customer_id, transaction_amount, SUM(transaction_amount) OVER (ORDER BY customer_id) as running_total FROM customer_transactions;

d. SELECT customer_id, transaction_amount, SUM(transaction_amount) OVER (ORDER BY date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as running_total FROM customer_transactions;

Answer: a

11. Consider a table called "employee_salaries" with columns "employee_name", "salary", and "date". Which of the following SQL queries will calculate the average salary for each employee over the last 12 months?

a. SELECT employee_name, AVG(salary) OVER (PARTITION BY employee_name) as avg_salary FROM employee_salaries WHERE date BETWEEN NOW() - INTERVAL 12 MONTH AND NOW();
b. SELECT employee_name, AVG(salary) OVER (PARTITION BY employee_name ORDER BY date RANGE BETWEEN INTERVAL 12 MONTH PRECEDING AND CURRENT ROW) as avg_salary FROM employee_salaries;
c. SELECT employee_name, AVG(salary) OVER (PARTITION BY employee_name ORDER BY date ROWS BETWEEN 12 PRECEDING AND CURRENT ROW) as avg_salary FROM employee_salaries;
d. SELECT employee_name, AVG(salary) OVER (PARTITION BY employee_name ORDER BY date RANGE BETWEEN 12 PRECEDING AND CURRENT ROW) as avg_salary FROM employee_salaries;

Answer: b

12. Consider a table called "orders" with columns "order_id", "order_date", and "order_total". Which of the following SQL queries will rank orders by their order total within each month in descending order?

a. SELECT order_id, order_date, order_total, RANK() OVER (PARTITION BY MONTH(order_date) ORDER BY order_total DESC) as order_rank FROM orders;
b. SELECT order_id, order_date, order_total, RANK() OVER (ORDER BY order_total DESC) as order_rank FROM orders;
c. SELECT order_id, order_date, order_total, RANK() OVER (PARTITION BY MONTH(order_date) ORDER BY order_total ASC) as order_rank FROM orders;
d. SELECT order_id, order_date, order_total, RANK() OVER (PARTITION BY MONTH(order_date) ORDER BY order_total) as order_rank FROM orders;

Answer: a

13. Consider a table called "customer_orders" with columns "customer_id", "order_date", and "order_total". Which of the following SQL queries will calculate the percent of each customer's total orders that are over $1000?

Select all queries which will give the correct Result.

a. SELECT customer_id, SUM(CASE WHEN order_total > 1000 THEN 1 ELSE 0 END) / COUNT(*) * 100 as percent_over_1000 FROM customer_orders GROUP BY customer_id;

b. SELECT customer_id, order_date, order_total, PERCENT_RANK() OVER (PARTITION BY customer_id) as order_percent FROM customer_orders WHERE order_total > 1000;

c. SELECT customer_id,, SUM(CASE WHEN order_total> 1000 THEN 1 ELSE 0 END) / COUNT(*) * 100 as percent_over_1000 FROM customer_orders GROUP BY customer_id;

d. SELECT customer_id, order_date, order_total, PERCENT_RANK() OVER (PARTITION BY customer_id ORDER BY order_total DESC) as order_percent FROM customer_orders WHERE order_total > 1000;

Answer: a, c

14. Consider a table called "website_traffic" with columns "page_url", "visit_date", and "visitors". Which of the following SQL queries will calculate the percent change in visitors to each page from the previous day?

a. SELECT page_url, visit_date, visitors, visitors - LAG(visitors) OVER (PARTITION BY page_url ORDER BY visit_date) / LAG(visitors) OVER (PARTITION BY page_url ORDER BY visit_date) * 100 as visitors_percent_change FROM website_traffic;

b. SELECT page_url, visit_date, visitors, (visitors - LAG(visitors) OVER (PARTITION BY page_url ORDER BY visit_date)) / visitors * 100 as visitors_percent_change FROM website_traffic;

c. SELECT page_url, visit_date, visitors, (visitors - LAG(visitors) OVER (ORDER BY visit_date)) / visitors * 100 as visitors_percent_change FROM website_traffic;

d. SELECT page_url, visit_date, visitors, (visitors - LAG(visitors) OVER (ORDER BY visit_date)) / LAG(visitors) OVER (ORDER BY visit_date) * 100 as visitors_percent_change FROM website_traffic;

Answer : d