Nitya CloudTech Pvt Ltd.

SQL Scenario-Based Interview Questions & Answers

1. Retrieve the Second Highest Salary for Each Department

Q: Find the second highest salary in each department.

A:

```
SELECT department_id, MAX(salary) AS second_highest_salary
FROM (
    SELECT department_id, salary,
           DENSE_RANK() OVER (PARTITION BY department_id ORDER BY
salary DESC) AS salary_rank
    FROM employees
) AS ranked_salaries
WHERE salary_rank = 2
GROUP BY department_id;
```

2. Calculate Cumulative Sales Percentage Contribution

Q: Calculate the cumulative percentage of total sales for each product.

A:

```
SELECT product_id, sales_amount,
       SUM(sales_amount) OVER (ORDER BY sales_amount DESC) * 100.0 /
SUM(sales_amount) OVER () AS cumulative_percentage
FROM product_sales;
```

3. Identify Orders with at Least One Product Above $100

Q: Find all orders where at least one product has a price greater than $100.

A:

```
SELECT DISTINCT order_id
FROM order_items
WHERE price > 100;
```

4. Rolling 3-Month Total Sales for Each Product

Q: Calculate a rolling 3-month sales total for each product.

A:

```
SELECT product_id, month, sales_amount,
       SUM(sales_amount) OVER (PARTITION BY product_id ORDER BY month
ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS rolling_3_month_sales
FROM monthly_sales;
```

5. Identify Customers with Multiple Purchases on the Same Day

Q: Find customers who made more than one purchase on the same day.

A:

```
SELECT customer_id, purchase_date
FROM purchases
GROUP BY customer_id, purchase_date
HAVING COUNT(*) > 1;
```

6. Calculate Total Duration Between Consecutive Orders for Each Customer

Q: For each customer, calculate the total time elapsed between consecutive orders.

A:

```
SELECT customer_id,
       SUM(DATEDIFF(day, LAG(order_date) OVER (PARTITION BY
customer_id ORDER BY order_date), order_date)) AS
total_time_between_orders
FROM orders
GROUP BY customer_id;
```

7. Identify Products Sold in All Regions

Q: List all products that have been sold in every region.

A:

```
SELECT product_id
FROM sales
GROUP BY product_id
HAVING COUNT(DISTINCT region_id) = (SELECT COUNT(DISTINCT region_id)
FROM sales);
```

8. Generate Monthly Sales Growth as a Percentage

Q: Calculate the month-over-month sales growth as a percentage.

A:

```
SELECT month, sales_amount,
       (sales_amount - LAG(sales_amount) OVER (ORDER BY month)) *
100.0 / LAG(sales_amount) OVER (ORDER BY month) AS
monthly_growth_percentage
FROM monthly_sales;
```

9. Retrieve Customers with the Longest Period of Inactivity

Q: Find the customers with the longest gap between two consecutive purchases.

A:

```
SELECT customer_id, MAX(DATEDIFF(day, LAG(purchase_date) OVER
(PARTITION BY customer_id ORDER BY purchase_date), purchase_date)) AS
max_inactivity_period
FROM purchases
GROUP BY customer_id;
```

10. Detect Cyclic Dependencies in Project Tasks

Q: Identify tasks with cyclic dependencies in a project.

A:

```
WITH RECURSIVE TaskHierarchy AS (
    SELECT task_id, dependency_id, ARRAY[task_id] AS path
    FROM tasks
    UNION ALL
    SELECT t.task_id, t.dependency_id, th.path || t.task_id
    FROM tasks t
    JOIN TaskHierarchy th ON t.dependency_id = th.task_id
    WHERE t.task_id <> ALL(th.path)
)
SELECT * FROM TaskHierarchy WHERE task_id = ANY(path);
```

11. Filter Orders with Most Recent Price for Each Product

Q: Retrieve orders showing only the latest price for each product.

A:

```
SELECT product_id, price, updated_at
FROM (
    SELECT product_id, price, updated_at,
           ROW_NUMBER() OVER (PARTITION BY product_id ORDER BY
updated_at DESC) AS rn
    FROM product_prices
) AS latest_price
WHERE rn = 1;
```

12. Calculate Total Sales of Top 10% Customers by Spend

Q: Find the total sales amount for the top 10% of customers by spending.

A:

```
WITH CustomerSpending AS (
    SELECT customer_id, SUM(sales_amount) AS total_spend
    FROM sales
    GROUP BY customer_id
),
TopCustomers AS (
    SELECT customer_id, total_spend,
           PERCENT_RANK() OVER (ORDER BY total_spend DESC) AS
percentile
    FROM CustomerSpending
)
SELECT SUM(total_spend) AS top_10_percent_total_sales
FROM TopCustomers
WHERE percentile <= 0.1;
```

13. Identify Days with Sales Deviating Significantly from Average

Q: Find the days when daily sales were 50% above or below the average.

A:

```
WITH AverageSales AS (
    SELECT AVG(sales_amount) AS avg_sales
    FROM daily_sales
)
SELECT sales_date, sales_amount
FROM daily_sales, AverageSales
WHERE sales_amount > 1.5 * avg_sales OR sales_amount < 0.5 *
avg_sales;
```

14. Rank Customers Based on Frequency of Purchases

Q: Rank customers by their purchase frequency.

A:

```
SELECT customer_id, COUNT(*) AS purchase_count,
       DENSE_RANK() OVER (ORDER BY COUNT(*) DESC) AS purchase_rank
FROM purchases
GROUP BY customer_id;
```

15. Find Orders with a Non-Matching Shipment Address

Q: Find orders where the shipping address doesn't match the customer's registered address.

A:

```
SELECT order_id
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
WHERE o.shipping_address <> c.registered_address;
```

16. Generate a Rank on Sales by Product for Each Month

Q: Rank products by sales in each month.

A:

```
SELECT month, product_id, sales_amount,
       RANK() OVER (PARTITION BY month ORDER BY sales_amount DESC) AS
sales_rank
FROM monthly_sales;
```

17. Calculate the Median Sales Amount for Each Product

Q: Calculate the median sales amount for each product.

A:

```
SELECT product_id,
```

```
       PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY sales_amount) AS
median_sales
FROM product_sales
GROUP BY product_id;
```

18. Identify Employees with a Pay Raise Every Year

Q: List employees who have received a pay raise every year.

A:

```
WITH YearlyRaises AS (
    SELECT employee_id, year, salary,
           LAG(salary) OVER (PARTITION BY employee_id ORDER BY year)
AS prev_salary
    FROM salaries
)
SELECT employee_id
FROM YearlyRaises
GROUP BY employee_id
HAVING COUNT(*) = COUNT(CASE WHEN salary > prev_salary THEN 1 END);
```

19. Calculate 6-Month Moving Average of Revenue

Q: Calculate the 6-month moving average of revenue.

A:

```
SELECT month, revenue,
       AVG(revenue) OVER (ORDER BY month ROWS BETWEEN 5 PRECEDING AND
CURRENT ROW) AS moving_avg_6_months
FROM monthly_revenue;
```

20. Detect Anomalies in Sales Data Based on Z-Score

Q: Identify sales amounts that are more than 3 standard deviations from the mean (outliers).

A:

```
WITH SalesStats AS (
    SELECT AVG(sales_amount) AS avg_sales, STDDEV(sales_amount) AS
stddev_sales
    FROM sales
)
SELECT sales_id, sales_amount
```

```
FROM sales, SalesStats
WHERE ABS(sales_amount - avg_sales) > 3 * stddev_sales;
```
