Amazon SQL interview Q&A

Here are all 15 SQL questions and answers in plain text format, with increased difficulty for specified questions. The first letter of each question is capitalized.

Q1: Identify customers who made purchases on exactly three different days in the last month.

Tables: purchases (customer_id, purchase_date)

Answer:

```
WITH purchases_summary AS (
SELECT customer_id, COUNT(DISTINCT purchase_date) AS purchase_days
FROM purchases
WHERE purchase_date >= DATEADD(month, -1, CURRENT_DATE)
GROUP BY customer_id
)
SELECT customer_id
FROM purchases_summary
WHERE purchase_days = 3;
```

Q2: Find the top 2 highest-selling products for each category.

Tables: sales (product_id, sale_amount), products (product_id, category)

Answer:

```
WITH ranked_sales AS (
SELECT
p.category,
s.product_id,
SUM(s.sale_amount) AS total_sales,
```

```
 RANK() OVER (PARTITION BY p.category ORDER BY SUM(s.sale_amount)
DESC) AS rank
 FROM sales s
 JOIN products p ON s.product_id = p.product_id
 GROUP BY p.category, s.product_id
 )
 SELECT category, product_id, total_sales
 FROM ranked_sales
 WHERE rank <= 2;
```

Q3: Detect anomalies where sales for a product are 50% lower than the average for that product.

Tables: sales (product_id, sale_amount)

Answer:

```
 WITH product_stats AS (
 SELECT product_id, AVG(sale_amount) AS avg_sales
 FROM sales
 GROUP BY product_id
 )
 SELECT s.product_id, s.sale_amount
 FROM sales s
 JOIN product_stats ps ON s.product_id = ps.product_id
 WHERE s.sale_amount < 0.5 * ps.avg_sales;
```

Q4: Find employees who have never been a manager and have worked in more than one department.

Tables: employees (employee_id, name, manager_id, department_id)

Answer:

```
 WITH manager_list AS (
```

```
SELECT DISTINCT manager_id
FROM employees
WHERE manager_id IS NOT NULL
),
department_count AS (
SELECT employee_id, COUNT(DISTINCT department_id) AS
department_count
FROM employees
GROUP BY employee_id
)
SELECT e.employee_id, e.name
FROM employees e
JOIN department_count dc ON e.employee_id = dc.employee_id
WHERE e.employee_id NOT IN (SELECT manager_id FROM manager_list)
AND dc.department_count > 1;
```

Q5: Calculate the median salary in each department.

Tables: employees (employee_id, department_id, salary)

Answer:

```
WITH ranked_salaries AS (
SELECT
department_id,
salary,
ROW_NUMBER() OVER (PARTITION BY department_id ORDER BY salary)
AS row_num,
COUNT(*) OVER (PARTITION BY department_id) AS total_rows
FROM employees
)
SELECT department_id, AVG(salary) AS median_salary
FROM ranked_salaries
WHERE row_num IN (FLOOR((total_rows + 1) / 2), CEIL((total_rows + 1) / 2))
GROUP BY department_id;
```

Q6: Identify customers who purchased products from all available categories.

Tables: purchases (customer_id, product_id), products (product_id, category)

Answer:

```
WITH categories_per_customer AS (
SELECT customer_id, COUNT(DISTINCT p.category) AS customer_categories
FROM purchases pu
JOIN products p ON pu.product_id = p.product_id
GROUP BY customer_id
),
total_categories AS (
SELECT COUNT(DISTINCT category) AS total_categories
FROM products
)
SELECT customer_id
FROM categories_per_customer, total_categories
WHERE customer_categories = total_categories;
```

Q7: Calculate the cumulative sales for each store, but only include dates where the daily sales exceeded the store's average daily sales.

Tables: sales (store_id, sale_amount, sale_date)

Answer:

```
WITH store_avg AS (
SELECT store_id, AVG(sale_amount) AS avg_sales
FROM sales
GROUP BY store_id
),
filtered_sales AS (
SELECT s.store_id, s.sale_date, s.sale_amount
FROM sales s
```

```
JOIN store_avg sa ON s.store_id = sa.store_id
WHERE s.sale_amount > sa.avg_sales
)
SELECT store_id, sale_date,
SUM(sale_amount) OVER (PARTITION BY store_id ORDER BY sale_date) AS cumulative_sales
FROM filtered_sales;
```

Q8: List employees who earn more than their department average.

Tables: employees (employee_id, department_id, salary)

Answer:

```
WITH department_avg AS (
SELECT department_id, AVG(salary) AS avg_salary
FROM employees
GROUP BY department_id
)
SELECT e.employee_id, e.salary
FROM employees e
JOIN department_avg da ON e.department_id = da.department_id
WHERE e.salary > da.avg_salary;
```

Q9: Identify products that have been sold but have no record in the products table and also calculate how many times each missing product has been sold.

Tables: sales (product_id), products (product_id)

Answer:

```
SELECT s.product_id, COUNT(*) AS times_sold
FROM sales s
LEFT JOIN products p ON s.product_id = p.product_id
```

```
WHERE p.product_id IS NULL
GROUP BY s.product_id;
```

Q10: Identify suppliers whose average delivery time is less than 2 days, but only consider deliveries with quantities greater than 100 units.

Tables: deliveries (supplier_id, delivery_date, order_date, quantity)

Answer:

```
SELECT supplier_id
FROM deliveries
WHERE quantity > 100
GROUP BY supplier_id
HAVING AVG(DATEDIFF(day, order_date, delivery_date)) < 2;
```

Q11: Find customers who made no purchases in the last 6 months but made at least one purchase in the 6 months prior to that.

Tables: customers (customer_id), purchases (customer_id, purchase_date)

Answer:

```
WITH six_months_ago AS (
SELECT customer_id
FROM purchases
WHERE purchase_date BETWEEN DATEADD(month, -12, CURRENT_DATE)
AND DATEADD(month, -6, CURRENT_DATE)
),
recent_purchases AS (
SELECT customer_id
FROM purchases
WHERE purchase_date >= DATEADD(month, -6, CURRENT_DATE)
)
SELECT DISTINCT c.customer_id
```

```
FROM customers c
JOIN six_months_ago sm ON c.customer_id = sm.customer_id
LEFT JOIN recent_purchases rp ON c.customer_id = rp.customer_id
WHERE rp.customer_id IS NULL;
```

Q12: Find the top 3 most frequent product combinations bought together.

Tables: order_details (order_id, product_id)

Answer:

```
WITH product_pairs AS (
SELECT
od1.product_id AS product1,
od2.product_id AS product2,
COUNT(*) AS pair_count
FROM order_details od1
JOIN order_details od2 ON od1.order_id = od2.order_id AND od1.product_id <
od2.product_id
GROUP BY od1.product_id, od2.product_id
)
SELECT product1, product2, pair_count
FROM product_pairs
ORDER BY pair_count DESC
LIMIT 3;
```

Q13: Calculate the moving average of sales for each product over a 7-day window.

Tables: sales (product_id, sale_amount, sale_date)

Answer:

```
SELECT
product_id,
sale_date,
```

```
 AVG(sale_amount) OVER (PARTITION BY product_id ORDER BY sale_date
ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS moving_avg
 FROM sales;
```

Q14: Rank stores by their monthly sales performance.

Tables: sales (store_id, sale_amount, sale_date)

Answer:

```
 WITH monthly_sales AS (
 SELECT
 store_id,
 DATE_TRUNC('month', sale_date) AS sale_month,
 SUM(sale_amount) AS total_sales
 FROM sales
 GROUP BY store_id, DATE_TRUNC('month', sale_date)
 )
 SELECT
 store_id,
 sale_month,
 total_sales,
 RANK() OVER (PARTITION BY sale_month ORDER BY total_sales DESC) AS
rank
 FROM monthly_sales;
```

Q15: Find customers who placed more than 50% of their orders in the last month.

Tables: orders (customer_id, order_id, order_date)

Answer:

```
 WITH order_stats AS (
 SELECT
 customer_id,
```

```
 COUNT(*) AS total_orders,
 SUM(CASE WHEN order_date >= DATEADD(month, -1, CURRENT_DATE)
THEN 1 ELSE 0 END) AS last_month_orders
 FROM orders
 GROUP BY customer_id
 )
 SELECT customer_id
 FROM order_stats
 WHERE last_month_orders > 0.5 * total_orders;
```
