Scenario-based SQL Interview Questions

1. Find Duplicate Records in a Table (Amazon)

```
SELECT column1, column2, COUNT(*)
FROM your_table
GROUP BY column1, column2
HAVING COUNT(*) > 1;
```

2. Retrieve the Second Highest Salary from Employee Table

```
SELECT MAX(salary) AS SecondHighestSalary
FROM Employee
WHERE salary < (SELECT MAX(salary) FROM Employee);
```

3. Find Employees Without Department (Uber)

```
SELECT e.*
FROM Employee e
LEFT JOIN Department d ON e.department_id = d.department_id
WHERE d.department_id IS NULL;
```

4. Calculate the Total Revenue Per Product (PayPal)

```
SELECT product_id, SUM(quantity * price) AS total_revenue
FROM Sales
GROUP BY product_id;
```

5. Get the Top 3 Highest-Paid Employees (Google)

```
SELECT *
FROM Employee
ORDER BY salary DESC LIMIT 3;
```

6. Customers Who Made Purchases but Never Returned Products (Walmart)

```
SELECT DISTINCT c.customer_id
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
WHERE c.customer_id NOT IN (SELECT customer_id FROM Returns);
```

7. Show the Count of Orders Per Customer (Meta)

```
SELECT customer_id, COUNT(*) AS order_count
FROM Orders
GROUP BY customer_id;
```

8. Retrieve All Employees Who Joined in 2023 (Amazon)

```
SELECT * FROM Employee
WHERE EXTRACT(YEAR FROM hire_date) = 2023;
```

9. Calculate Average Order Value Per Customer (Microsoft)

```
SELECT customer_id, AVG(total_amount) AS avg_order_value
FROM Orders
GROUP BY customer_id;
```

10. Get the Latest Order Placed by Each Customer (Uber)

```
SELECT customer_id, MAX(order_date) AS latest_order_date
FROM Orders
GROUP BY customer_id;
```

11. Find Products That Were Never Sold

```
SELECT p.product_id
FROM Products p
LEFT JOIN Sales s ON p.product_id = s.product_id
WHERE s.product_id IS NULL;
```

12. Identify the Most Selling Product (Adobe/Walmart)

```
SELECT product_id, SUM(quantity) AS total_qty
FROM Sales
GROUP BY product_id
ORDER BY total_qty DESC LIMIT 1;
```

13. Get Total Revenue and Number of Orders Per Region (Meta)

```
SELECT region, SUM(total_amount) AS total_revenue, COUNT(*) AS order_count
FROM Orders
GROUP BY region;
```

14. Count Customers with More Than 5 Orders (Amazon)

```
SELECT COUNT(*) AS customer_count
FROM (
    SELECT customer_id
    FROM Orders
    GROUP BY customer_id
    HAVING COUNT(*) > 5
) AS subquery;
```

15. Retrieve Customers with Orders Above Average Order Value (PayPal)

```
SELECT DISTINCT customer_id
FROM Orders
WHERE total_amount > (SELECT AVG(total_amount) FROM Orders);
```

16. Find All Employees Hired on Weekends (Google)

```
SELECT *
FROM Employee
WHERE EXTRACT(DOW FROM hire_date) IN (0, 6);
```

17. Find All Employees with Salary Between 50000 and 100000 (Microsoft)

```
SELECT *
FROM Employee
WHERE salary BETWEEN 50000 AND 100000;
```

18. Get Monthly Sales Revenue and Order Count (Google)

```
SELECT TO_CHAR(order_date, 'YYYY-MM') AS month,
 SUM(total_amount) AS total_revenue,
 COUNT(order_id) AS order_count
FROM Orders
GROUP BY TO_CHAR(order_date, 'YYYY-MM');
```

19. Rank Employees by Salary Within Each Department (Amazon)

```
SELECT employee_id, department_id, salary,
       RANK() OVER (PARTITION BY department_id ORDER BY salary DESC) AS salary_rk
FROM Employee;
```

20. Find Customers Who Placed Orders Every Month in 2023 (Meta)

```
SELECT customer_id
FROM Orders
WHERE EXTRACT(YEAR FROM order_date) = 2023
GROUP BY customer_id
HAVING COUNT(DISTINCT TO_CHAR(order_date, 'YYYY-MM')) = 12;
```

21. Find Moving Average of Sales Over the Last 3 Days (Microsoft)

```
SELECT order_date,
    AVG(total_amount) OVER (
        ORDER BY order_date
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
    )
FROM Orders;
```

22. Identify the First and Last Order Date for Each Customer (Uber)

```
SELECT customer_id, MIN(order_date) AS first_order, MAX(order_date) AS last_order
FROM Orders
GROUP BY customer_id;
```

23. Show Product Sales Distribution (Percent of Total Revenue) (PayPal)

```
WITH TotalRevenue AS (
    SELECT SUM(quantity * price) AS total FROM Sales
)
SELECT s.product_id,
       SUM(s.quantity * s.price) AS revenue,
       SUM(s.quantity * s.price) * 100 / t.total AS revenue_pct
FROM Sales s
CROSS JOIN TotalRevenue t
GROUP BY s.product_id, t.total;
```

24. Retrieve Customers Who Made Consecutive Purchases (2 Days) (Walmart)

```
WITH cte AS (
    SELECT
        customer_id,
        order_date,
        LAG(order_date) OVER (
            PARTITION BY customer_id
            ORDER BY order_date
        ) AS prev_order_date
    FROM Orders
)
SELECT customer_id, order_date, prev_order_date
FROM cte
WHERE order_date - prev_order_date = INTERVAL '1' DAY;
```

25. Find Churned Customers (No Orders in the Last 6 Months) (Amazon)

```
SELECT customer_id
FROM Orders
GROUP BY customer_id
HAVING MAX(order_date) < (NOW() - INTERVAL '6 months');
```

26. Calculate Cumulative Revenue by Day (Adobe)

```
SELECT order_date,
       SUM(total_amount) OVER (ORDER BY order_date) AS cumulative_revenue
FROM Orders;
```

27. Identify Top-Performing Departments by Average Salary (Google)

```
SELECT department_id, AVG(salary) AS avg_salary
FROM Employee
GROUP BY department_id
ORDER BY avg_salary DESC;
```

28. Find Customers Who Ordered More Than the Average Number of Orders Per Customer (Meta)

```
WITH customer_orders AS (
    SELECT customer_id, COUNT(*) AS order_count
    FROM Orders
    GROUP BY customer_id
)
SELECT *
FROM customer_orders
WHERE order_count > (SELECT AVG(order_count) FROM customer_orders);
```

29. Calculate Revenue Generated from New Customers (First-Time Orders) (Microsoft)

```
WITH first_orders AS (
    SELECT customer_id, MIN(order_date) AS first_order_date
    FROM Orders
    GROUP BY customer_id
)
SELECT SUM(o.total_amount) AS new_revenue
FROM Orders o
JOIN first_orders f ON o.customer_id = f.customer_id
WHERE o.order_date = f.first_order_date;
```

30. Find the Percentage of Employees in Each Department (Uber)

```
SELECT department_id, COUNT(*) AS emp_count,
       COUNT(*) * 100.0 / (SELECT COUNT(*) FROM Employee) AS pct
FROM Employee
GROUP BY department_id;
```

31. Retrieve the Maximum Salary Difference Within Each Department (PayPal)

```
SELECT department_id,
       MAX(salary) - MIN(salary) AS salary_diff
FROM Employee
GROUP BY department_id;
```

32. Find Products That Contribute to 80% of the Revenue (Pareto Principle) (Walmart)

```
WITH sales_cte AS (
    SELECT product_id, SUM(quantity * price) AS revenue
    FROM Sales
    GROUP BY product_id
),
total_revenue AS (SELECT SUM(revenue) AS total FROM sales_cte)
SELECT product_id, revenue, cumulative_revenue
FROM (SELECT s.product_id, s.revenue,
           SUM(s.revenue) OVER (ORDER BY s.revenue DESC) AS cumulative_revenue,
           t.total
    FROM sales_cte s CROSS JOIN total_revenue t)
WHERE cumulative_revenue <= total * 0.8;
```

33. Show Last Purchase for Each Customer Along with Order Amount (Google)

```
WITH ranked_orders AS (
    SELECT customer_id, order_id, total_amount,
           ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY order_date DESC)
AS rn
    FROM Orders
)
SELECT customer_id, order_id, total_amount
FROM ranked_orders
WHERE rn = 1;
```

34. Calculate Average Time Between Two Purchases for Each Customer (Meta)

```
WITH cte AS (
    SELECT
        customer_id,
        order_date,
        LAG(order_date) OVER (
            PARTITION BY customer_id
            ORDER BY order_date
        ) AS prev_date
    FROM Orders
)
SELECT
    customer_id,
    AVG(DATE_DIFF(DAY, prev_date, order_date)) AS avg_gap_days
FROM cte
WHERE prev_date IS NOT NULL
GROUP BY customer_id;
```

35. Calculate Year-Over-Year Growth in Revenue (Microsoft)

```
WITH yearly AS (
    SELECT EXTRACT(YEAR FROM order_date) AS year,
           SUM(total_amount) AS revenue
    FROM Orders
    GROUP BY EXTRACT(YEAR FROM order_date)
)
SELECT year,
       revenue,
       revenue - LAG(revenue) OVER (ORDER BY year) AS yoy_growth
FROM yearly;
```

36. Detect Customers Whose Purchase Amount Is Higher Than Their Historical 90th Percentile (Amazon)

```
WITH ranked_orders AS (
    SELECT customer_id, order_id, total_amount,
           NTILE(10) OVER (PARTITION BY customer_id ORDER BY total_amount) AS
decile
    FROM Orders
)
SELECT customer_id, order_id, total_amount
FROM ranked_orders
WHERE decile = 10;
```

37. Retrieve the Longest Gap Between Orders for Each Customer (Meta)

```
WITH cte AS (
    SELECT customer_id, order_date,
           LAG(order_date) OVER (PARTITION BY customer_id ORDER BY order_date) AS
prev_order_date
    FROM Orders
)
SELECT customer_id,
       MAX(DATE_DIFF(DAY, prev_order_date, order_date)) AS max_gap_days
FROM cte
WHERE prev_order_date IS NOT NULL
GROUP BY customer_id;
```

38. Identify Customers with Revenue Below the 10th Percentile (Google)

```
WITH cte AS (
    SELECT customer_id, SUM(total_amount) AS total_revenue
    FROM Orders
    GROUP BY customer_id
)
SELECT customer_id, total_revenue
FROM cte
WHERE total_revenue < (
    SELECT PERCENTILE_CONT(0.1) WITHIN GROUP (ORDER BY total_revenue) FROM cte
);
```