SQL Hard Interview Questions

PRAGYA RATHI

1. 7-Day User Retention Rate

Task: Calculate the percentage of users active 7 days after their signup.

```
SELECT
  signup_week,
  COUNT(DISTINCT a.user_id) * 100.0 / COUNT(DISTINCT s.user_id) AS retention_rate
FROM signups s
LEFT JOIN activity a
  ON s.user_id = a.user_id
  AND a.event_date = s.signup_date + INTERVAL '7 days'
GROUP BY signup_week;
```

---

2. Top 5 Products per Category (with Ties)

Task: Rank products within categories by sales, including ties.

```
WITH ranked_products AS (
  SELECT
    category,
    product_id,
    sales,
    DENSE_RANK() OVER (PARTITION BY category ORDER BY sales DESC) AS rank
  FROM products
)
SELECT category, product_id, sales
FROM ranked_products
WHERE rank <= 5;
```

---

3. Month-over-Month Revenue Growth

Task: Compute monthly revenue growth percentage, handling missing months.

```
WITH monthly_revenue AS (
  SELECT
    DATE_TRUNC('month', order_date) AS month,
    SUM(revenue) AS revenue
  FROM orders
  GROUP BY 1
)
SELECT
  month,
```

```
  (revenue - LAG(revenue) OVER (ORDER BY month)) * 100.0 / LAG(revenue) OVER (ORDER
BY month) AS growth
FROM monthly_revenue;
```

---

4. Consecutive Purchases/Logins

Task: Find users with 3+ consecutive days of activity.

```
WITH user_dates AS (
  SELECT
    user_id,
    event_date,
    LAG(event_date, 2) OVER (PARTITION BY user_id ORDER BY event_date) AS prev_date
  FROM activity
)
SELECT DISTINCT user_id
FROM user_dates
WHERE event_date - prev_date = 2;
```

---

5. Funnel Conversion Rates

Task: Calculate drop-offs between stages (e.g., visit → signup → purchase).

```
WITH stages AS (
  SELECT
    visit.user_id,
    signup.user_id IS NOT NULL AS signed_up,
    purchase.user_id IS NOT NULL AS purchased
  FROM visits
  LEFT JOIN signups ON visit.user_id = signup.user_id
  LEFT JOIN purchases ON visit.user_id = purchase.user_id
)
SELECT
  COUNT(*) AS visits,
  SUM(signed_up) AS signups,
  SUM(purchased) AS purchases
FROM stages;
```

---

6. Median Session Duration

Task: Compute median session duration without built-in functions.

```
WITH ordered_sessions AS (
```

```
  SELECT
    duration,
    ROW_NUMBER() OVER (ORDER BY duration) AS rn,
    COUNT(*) OVER () AS total
  FROM sessions
)
SELECT AVG(duration) AS median
FROM ordered_sessions
WHERE rn BETWEEN total/2 AND total/2 + 1;
```

---

7. Cohort Weekly Retention

Task: Track weekly retention for 8 weeks post-signup.

```
SELECT
  signup_week,
  COUNT(DISTINCT user_id) AS cohort_size,
  COUNT(DISTINCT CASE WHEN week_diff = 1 THEN user_id END) AS week_1,
  COUNT(DISTINCT CASE WHEN week_diff = 2 THEN user_id END) AS week_2,
  ...
FROM (
  SELECT
    s.user_id,
    DATE_TRUNC('week', s.signup_date) AS signup_week,
    DATE_TRUNC('week', a.event_date) - DATE_TRUNC('week', s.signup_date) AS week_diff
  FROM signups s
  LEFT JOIN activity a ON s.user_id = a.user_id
)
GROUP BY 1;
```

---

8. 30-Day Moving Average of Sales

Task: Calculate a rolling average of daily sales.

```
SELECT
  date,
  AVG(sales) OVER (ORDER BY date RANGE BETWEEN INTERVAL '29 days' PRECEDING
AND CURRENT ROW) AS moving_avg
FROM daily_sales;
```

---

9. Reciprocal User Follows

Task: Identify mutual follows (A → B and B → A).

```
SELECT DISTINCT
  f1.user_id AS user_a,
  f2.user_id AS user_b
FROM follows f1
JOIN follows f2
  ON f1.user_id = f2.target_id
  AND f1.target_id = f2.user_id
  AND f1.user_id < f2.user_id;
```

---

10. Net Promoter Score (NPS)

Task: Compute NPS from survey scores (0-10).

```
WITH nps_categories AS (
  SELECT
    CASE
      WHEN score >= 9 THEN 'promoter'
      WHEN score <= 6 THEN 'detractor'
      ELSE 'passive'
    END AS category
  FROM survey
)
SELECT
  (COUNT(*) FILTER (WHERE category = 'promoter') -
   COUNT(*) FILTER (WHERE category = 'detractor')) * 100.0 / COUNT(*) AS nps
FROM nps_categories;
```