

Google StratOps Advanced SQL Patterns & Query Templates

Production-Grade Reference for Shared-Doc Whiteboard Rehearsal & Live Interview Problem Solving

15 Tested Drills

1. Safe Division & Ratio Guardrails Core Pattern

```
CASE
  WHEN SUM(impressions) > 0
  THEN ROUND(100.0 * SUM(clicks) / SUM(impressions), 2)
  ELSE 0
END AS ctr_pct
```

2. Anti-Join (Unactivated Accounts) Core Pattern

```
SELECT a.advertiser_id, a.name
FROM advertisers a
LEFT JOIN campaigns c ON a.advertiser_id = c.advertiser_id
WHERE c.campaign_id IS NULL;
```

3. Top-N Per Group via Window Functions Core Pattern

```
WITH ranked AS (
  SELECT vertical, campaign_id, spend,
         ROW_NUMBER() OVER (
           PARTITION BY vertical ORDER BY spend DESC
         ) AS rnk
  FROM campaign_spends
)
SELECT * FROM ranked WHERE rnk <= 2;
```

4. Month-over-Month Velocity via CASE Core Pattern

```
SELECT vertical,
       SUM(CASE WHEN strftime('%m', date)='07' THEN spend END) AS jul,
       SUM(CASE WHEN strftime('%m', date)='08' THEN spend END) AS aug,
       ROUND(100.0*(aug - jul)/jul, 2) AS mom_growth_pct
FROM daily_spend GROUP BY 1;
```

5. 7-Day Rolling Moving Average Core Pattern

```
SELECT date, daily_spend,
       ROUND(AVG(daily_spend) OVER (
         ORDER BY date
         ROWS BETWEEN 6 PRECEDING AND CURRENT ROW
       ), 2) AS rolling_7d_avg
FROM daily_totals;
```

6. Cohort Retention Matrix Offset Core Pattern

```
SELECT cohort_month,
       COUNT(DISTINCT a_id) AS total_cohort,
       COUNT(DISTINCT CASE WHEN offset = 0 THEN a_id END) AS m0,
       COUNT(DISTINCT CASE WHEN offset = 1 THEN a_id END) AS m1
FROM cohort_activity GROUP BY 1;
```

7. Non-Equi Date-Range Attribution Core Pattern

```
SELECT s.seller_id, SUM(ds.spend_usd)
FROM sellers s
JOIN assignments a ON s.seller_id = a.seller_id
JOIN daily_spend ds ON a.advertiser_id = ds.advertiser_id
AND ds.date >= a.start_date
AND (a.end_date IS NULL OR ds.date <= a.end_date)
GROUP BY 1;
```

Interview Pro-Tip:
Google interviewers often test SQL in a shared Google Doc without execution. Always vocalize your query architecture aloud: 1. State assumptions, 2. Name CTEs clearly, 3. Guard against NULL division, 4. Handle date boundaries cleanly.