Enterprise-Scale Advanced SQL for Streaming Analytics: Churn Prediction, MAU Segmentation, and Genre Ranking with Recursive CTEs & JSON Processing

/*

Context:

Assume we are working with a Netflix-like streaming service database.

Tables:

- users(user_id, name, country, signup_date, subscription_plan, metadata JSON)
- watch_history(user_id, content_id, watch_date, watch_duration, device, region)
- content(content_id, title, genre, release_year, metadata JSON)
- payments(payment_id, user_id, amount, currency, payment_date, status)

Goal:

Build an advanced analytical query for:

1. Monthly active users (MAU) segmented by region & subscription plan.
2. Churn risk prediction (users inactive > 30 days but subscribed).
3. Top 3 genres per region using advanced window functions.
4. Incorporating JSON fields from metadata.
5. Optimizing performance with partitioning + indexing hints.

*/

```
-- 1. Recursive CTE: Generate rolling months (last 12 months)
WITH RECURSIVE month_calendar AS (
    SELECT DATE_TRUNC('month', CURRENT_DATE) - INTERVAL '11 months' AS month_start
    UNION ALL
    SELECT month_start + INTERVAL '1 month'
    FROM month_calendar
    WHERE month_start + INTERVAL '1 month' <= DATE_TRUNC('month', CURRENT_DATE)
),

-- 2. Monthly Active Users (MAU) per region & plan
active_users AS (
    SELECT
        u.country AS region,
        u.subscription_plan,
        DATE_TRUNC('month', w.watch_date) AS activity_month,
        COUNT(DISTINCT u.user_id) AS mau
    FROM users u
    JOIN watch_history w
      ON u.user_id = w.user_id
    WHERE w.watch_date >= CURRENT_DATE - INTERVAL '12 months'
    GROUP BY region, u.subscription_plan, DATE_TRUNC('month', w.watch_date)
),
```

```
-- 3. Churn Risk Detection: users subscribed but no activity for >30 days
churn_risk AS (
    SELECT
        u.user_id,
        u.country AS region,
        u.subscription_plan,
        MAX(w.watch_date) AS last_watch_date,
        CURRENT_DATE - MAX(w.watch_date) AS days_inactive
    FROM users u
    LEFT JOIN watch_history w ON u.user_id = w.user_id
    WHERE u.subscription_plan IS NOT NULL
    GROUP BY u.user_id, region, u.subscription_plan
    HAVING MAX(w.watch_date) < CURRENT_DATE - INTERVAL '30 days'
),

-- 4. Genre Popularity Ranking (Top 3 genres per region)
genre_rank AS (
    SELECT
        w.region,
        c.genre,
        COUNT(*) AS watch_count,
        RANK() OVER (PARTITION BY w.region ORDER BY COUNT(*) DESC) AS
genre_rank
    FROM watch_history w
    JOIN content c ON w.content_id = c.content_id
```

```
    WHERE w.watch_date >= CURRENT_DATE - INTERVAL '6 months'
    GROUP BY w.region, c.genre
    HAVING COUNT(*) > 100  -- rare filter: remove sparse genres
),

-- 5. SON Extraction (e.g., device type from user metadata)
device_pref AS (
    SELECT
        u.user_id,
        u.country AS region,
        JSON_EXTRACT_PATH_TEXT(u.metadata, 'preferred_device') AS preferred_device
    FROM users u
    WHERE JSON_EXTRACT_PATH_TEXT(u.metadata, 'preferred_device') IS NOT NULL
)

-- Final Combined Report
SELECT
    m.month_start AS report_month,
    a.region,
    a.subscription_plan,
    a.mau,
    COALESCE(c.days_inactive, 0) AS churn_days_inactive,
    g.genre AS top_genre,
    d.preferred_device,
```

```
    ROUND(AVG(p.amount) FILTER (WHERE p.status = 'SUCCESS'), 2) AS
avg_payment_usd

FROM month_calendar m

LEFT JOIN active_users a

    ON m.month_start = a.activity_month

LEFT JOIN churn_risk c

    ON a.region = c.region AND a.subscription_plan = c.subscription_plan

LEFT JOIN genre_rank g

    ON a.region = g.region AND g.genre_rank <= 3

LEFT JOIN device_pref d

    ON a.region = d.region

LEFT JOIN payments p

    ON a.subscription_plan IS NOT NULL AND p.user_id = d.user_id

GROUP BY m.month_start, a.region, a.subscription_plan, a.mau, c.days_inactive, g.genre,
d.preferred_device

ORDER BY m.month_start, a.region, a.subscription_plan, a.mau DESC;
```