```sql
-- Q1) LAG: Her siparişin bir önceki sipariş tarihini ve amount'unu göster
SELECT
    user_id,
    order_id,
    order_date,
    amount,
    LAG(order_date) OVER (PARTITION BY user_id ORDER BY order_date, order_id) AS
prev_order_date,
    LAG(amount) OVER (PARTITION BY user_id ORDER BY order_date, order_id) AS
prev_amount

FROM orders
ORDER BY user_id, order_date, order_id;

-- Q2) Kümülatif harcama + kümülatif ortalama

SELECT
    user_id,
    order_id,
    order_date,
    amount,
    SUM(amount) OVER (PARTITION BY user_id ORDER BY order_date, order_id ROWS
BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS running_total,
    AVG(amount) OVER (PARTITION BY user_id ORDER BY order_date, order_id ROWS
BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS running_avg
FROM orders
ORDER BY user_id, order_date, order_id;


-- Q3) Siparişler arası gün farkı (LAG  + date diff)
SELECT
    user_id,
    order_id,
    order_date,
    amount,
    LAG(order_date) OVER (PARTITION BY user_id ORDER BY order_date, order_id) AS
prev_order_date,
    (order_date - LAG(order_date) OVER (PARTITION BY user_id ORDER BY order_date,
order_id)) AS days_between_orders
FROM orders
ORDER BY user_id, order_date, order_id;


-- Q4 Kullanıcı başına en pahalı sipariş (ROW_NUMBER)
WITH ranked AS (
    SELECT
        o.*,
        ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY amount DESC, order_date
DESC, order_id DESC) AS rn
    FROM orders o
)
SELECT
```

```
    user_id, order_id, order_date, amount
FROM ranked
WHERE rn = 1
ORDER BY user_id;

-- Q5) Günlük gelir + 7 günlük hareketli ortalama
WITH daily AS (
    SELECT
        order_date,
        SUM(amount) AS daily_revenue
    FROM orders
    GROUP BY order_date
)
SELECT
    order_date,
    daily_revenue,
    AVG(daily_revenue) OVER (ORDER BY order_date ROWS BETWEEN 6 PRECEDING AND
CURRENT ROW) AS revenue_7d_ma
FROM daily
ORDER BY order_date;

-- Q6) Aynı gün birden fazla sipariş veren kullanıcılar
SELECT
    user_id,
    order_date,
    COUNT(*) AS orders_count
FROM orders
GROUP BY user_id, order_date
HAVING COUNT(*) >= 2
ORDER BY order_date, user_id;

--Q7) İlk siparişten itibaren 30 gün içinde en az 2 sipariş veren kullanıcılar
WITH first_order AS(
    SELECT
        user_id,
        MIN(order_date) AS first_order_date
    FROM orders
    GROUP BY user_id
),

orders_first30 AS (
    SELECT
        o.user_id,
        fo.first_order_date,
        o.order_id,
        o.order_date
    FROM orders o
    JOIN first_order fo ON fo.user_id = o.user_id
    WHERE o.order_date < fo.first_order_date + INTERVAL '30 days'

)
SELECT
    user_id,
    first_order_date,
```

```
    COUNT(*) AS orders_in_first_30_days
FROM orders_first30
GROUP BY user_id, first_order_date
HAVING COUNT(*) >= 2
ORDER BY user_id;

-- Q8) Total spent'e göre 4 çeyrek (NTILE)
WITH user_spend AS (
    SELECT
        u.user_id,
        u.full_name,
        COALESCE(SUM(o.amount), 0) AS total_spent
    FROM users u
    LEFT JOIN orders o  ON o.user_id = u.user_id
    GROUP BY u.user_id, u.full_name
)

SELECT
    user_id,
    full_name,
    total_spent,
    NTILE(4) OVER (ORDER BY total_spent) AS spend_quartile
FROM user_spend
ORDER BY total_spent, user_id;

-- Q9) Aylık gelir + MoM büyüme (LAG)
WITH monthly AS (
    SELECT
        DATE_TRUNC('month', order_date)::date AS month,
        SUM(amount) as monthly_revenue
    FROM orders
    GROUP BY DATE_TRUNC('month', order_date)
)
SELECT
    month,
    monthly_revenue,
    LAG(monthly_revenue) OVER (ORDER BY month) AS prev_month_revenue,
    CASE
        WHEN LAG(monthly_revenue) OVER (ORDER BY month) IS NULL THEN NULL
        WHEN LAG(monthly_revenue) OVER (ORDER BY month) = 0 THEN NULL
        ELSE ROUND(
            ( (monthly_revenue - LAG(monthly_revenue) OVER (ORDER BY month)) /
LAG(monthly_revenue) OVER (ORDER BY month))*100,2)
    END as mom_growth_pct
FROM monthly
ORDER BY month;


-- Q10) Churn Label : Son sipariş 30 günden eskiyse 1, değilse 0

WITH last_order AS (
    SELECT
        u.user_id,
        u.full_name,
```

```
        MAX(o.order_date) AS last_order_date
    FROM users u
    LEFT JOIN orders o ON o.user_id = u.user_id
    GROUP BY u.user_id, u.full_name

)

SELECT
    user_id,
    full_name,
    last_order_date,
    CASE
        WHEN last_order_date IS NULL THEN 1
        WHEN last_order_date < (CURRENT_DATE - INTERVAL '30 days') THEN 1
        ELSE 0
    END AS is_churned_30d
FROM last_order
ORDER BY user_id;
```