01

```sql
WITH paid_orders AS (
    SELECT o.order_id, o.customer_id, o.order_date
    FROM orders o
    WHERE o.status = 'paid'
),
order_revenue AS (
    SELECT
        po.order_id,
        po.customer_id,
        po.order_date,
        SUM(oi.quantity * oi.unit_price) AS revenue
    FROM paid_orders po
    JOIN order_items oi ON oi.order_id = po.order_id
    GROUP BY 1,2,3
)

SELECT
    DATE_TRUNC('month', order_date)::date AS month,
    COUNT(DISTINCT order_id)              AS orders,
    COUNT(DISTINCT customer_id)           AS active_customers,
    ROUND(SUM(revenue), 2)                AS total_revenue,
    ROUND(AVG(revenue), 2)                AS aov
FROM order_revenue
GROUP BY 1
ORDER BY 1;
```

02

```sql
WITH paid_orders AS (
    SELECT o.order_id, o.customer_id, o.order_date
    FROM orders o
    WHERE o.status = 'paid'
),
order_revenue AS (
    SELECT
        DATE_TRUNC('month', po.order_date)::date AS month,
        po.order_id,
        SUM(oi.quantity * oi.unit_price) AS revenue
    FROM paid_orders po
    JOIN order_items oi ON oi.order_id = po.order_id
    GROUP BY 1,2
),

monthly AS (
    SELECT
        month,
```

```
        SUM(revenue) AS monthly_revenue
    FROM order_revenue
    GROUP BY 1
)

SELECT
    month,
    ROUND(monthly_revenue,2) AS monthly_revenue,
    ROUND(
        SUM(monthly_revenue) OVER (ORDER BY month),
        2
    ) AS running_revenue,
    ROUND(
        (monthly_revenue - LAG(monthly_revenue) OVER (ORDER BY month))
        / NULLIF(LAG(monthly_revenue) OVER(ORDER BY month), 0) *100,
        2
    ) AS mom_growth_pct
FROM monthly
ORDER BY month;

WITH paid_orders AS (
    SELECT o.order_id, o.order_date
    FROM orders o
    WHERE o.status = 'paid'
),

product_monthly AS (
    SELECT
        DATE_TRUNC('month', po.order_date)::date AS month,
        p.product_name,
        p.category,
        SUM(oi.quantity * oi.unit_price) AS revenue
    FROM paid_orders po
    JOIN order_items oi ON oi.order_id = po.order_id
    JOIN products p ON p.product_id = oi.product_id
    GROUP BY 1,2,3
),

ranked AS (
    SELECT
        *,
        RANK() OVER (PARTITION BY month ORDER BY revenue DESC) AS rnk
    FROM product_monthly
)

SELECT
    month, product_name, category, ROUND(revenue,2) AS revenue, rnk
FROM ranked
WHERE rnk <= 3
ORDER BY month, rnk, revenue DESC;
```

03

```sql
WITH paid_orders AS (
    SELECT o.customer_id, o.order_date
    FROM orders o
    WHERE o.status = 'paid'
),

first_paid AS (
    SELECT
        customer_id,
        MIN(DATE_TRUNC('month', order_date)) AS cohort_month
    FROM paid_orders
    GROUP BY 1
),

activity AS (
    SELECT
        po.customer_id,
        DATE_TRUNC('month', po.order_date) AS activity_month
    FROM paid_orders po
    GROUP BY 1,2
),

cohort_activity AS (
    SELECT
        fp.cohort_month,
        a.activity_month,
        (DATE_PART('year', a.activity_month) - DATE_PART('year', fp.cohort_month))
* 12
        + (DATE_PART('month', a.activity_month) - DATE_PART('month',
fp.cohort_month)) AS month_index,
        COUNT(DISTINCT a.customer_id) AS active_customers
    FROM first_paid fp
    JOIN activity a ON a.customer_id = fp.customer_id
    GROUP BY 1,2,3
),

cohort_size AS (
    SELECT
        cohort_month,
        COUNT(DISTINCT customer_id) AS cohort_size
    FROM first_paid
    GROUP BY 1
)

SELECT
    ca.cohort_month ::date,
    ca.activity_month::date,
    ca.month_index::int,
    ca.active_customers,
    cs.cohort_size,
    ROUND(ca.active_customers::numeric / NULLIF(cs.cohort_size,0),4) AS
retention_rate
```

```
FROM cohort_activity ca
JOIN cohort_size cs USING (cohort_month)
ORDER BY cohort_month, month_index;
```

04

```
WITH params AS (
    SELECT DATE '2025-07-01' AS as_of_date
),

paid_orders AS (
    SELECT o.order_id, o.customer_id, o.order_date
    FROM orders o
    WHERE o.status = 'paid'
),

customer_revenue AS (
    SELECT
        po.customer_id,
        po.order_id,
        po.order_date,
        SUM(oi.quantity * oi.unit_price) AS revenue
    FROM paid_orders po
    JOIN order_items oi ON oi.order_id = po.order_id
    GROUP BY 1,2,3
),

rfm_raw AS (
    SELECT
        cr.customer_id,
        (SELECT as_of_date FROM params) - MAX(cr.order_date) AS recency_days,
        COUNT(DISTINCT cr.order_id) AS frequency,
        ROUND(SUM(cr.revenue), 2) AS monetary
    FROM customer_revenue cr
    GROUP BY 1
),

scored AS (
    SELECT
        *,
        NTILE(5) OVER (ORDER BY recency_days ASC) AS r_score,
        NTILE(5) OVER (ORDER BY frequency DESC) AS f_score,
        NTILE(5) OVER (ORDER BY monetary DESC) AS m_score
    FROM rfm_raw

),
segmented AS (
    SELECT
        *,
        (r_score + f_score + m_score) AS rfm_total,
        CASE
```

```
            WHEN (r_score >= 4 AND f_score >= 4 AND m_score >= 4) THEN 'Champions'
            WHEN (f_score >= 4 AND m_score >=4) THEN 'Loyal Big Spenders'
            WHEN (r_score >= 4 AND f_score >= 3) THEN 'Loyal'
            WHEN (r_score <= 2 AND f_score >= 3) THEN 'At Risk'
            WHEN (r_score <= 2 AND f_score <= 2) THEN 'Hibernating'
            ELSE 'Potential'
        END AS segment
    FROM scored
)
SELECT
    customer_id,
    recency_days,
    frequency,
    monetary,
    r_score, f_score,m_score,
    rfm_total,
    segment
FROM segmented
ORDER BY rfm_total DESC, monetary DESC;
```

05

```
WITH paid_orders AS (
    SELECT customer_id, COUNT(*) AS order_cnt
    FROM orders
    WHERE status = 'paid'
    GROUP BY 1

)
SELECT
    ROUND(AVG(CASE WHEN order_cnt >= 2 THEN 1 ELSE 0 END)::numeric,4) AS
repeat_purchase_rate,
    COUNT(*) AS total_customers
FROM paid_orders;

WITH paid_orders AS (
    SELECT order_id
    FROM orders
    WHERE status = 'paid'
),
rev AS (
    SELECT
        p.category,
        SUM(oi.quantity * oi.unit_price) AS revenue
    FROM paid_orders po
    JOIN order_items oi ON oi.order_id = po.order_id
    JOIN products p ON p.product_id = oi.product_id
    GROUP BY 1
),
tot AS (
    SELECT SUM(revenue) AS total_revenue FROM rev
```

```
)
SELECT
    r.category,
    ROUND(r.revenue,2) AS revenue,
    ROUND(r.revenue / NULLIF(t.total_revenue,0) * 100, 2) AS revenue_share_pct
FROM rev r
CROSS JOIN tot t
ORDER BY revenue DESC;

WITH paid AS (
    SELECT customer_id, order_date
    FROM orders
    WHERE status = 'paid'
),
seq AS (
    SELECT
        customer_id,
        order_date,
        LEAD(order_date) OVER (PARTITION BY customer_id ORDER BY order_date) AS
next_order_date
    FROM paid
)

SELECT
    customer_id,
    order_date,
    next_order_date,
    (next_order_date - order_date) AS gap_days
FROM seq
WHERE next_order_date IS NOT NULL
ORDER BY gap_days DESC
LIMIT 10;
```