daily full sql

/\* =========================

DAILY KPIs — MULTI-DAY with SUMMARY ROW

Currency filter uses resolved txn currency (metadata → txn → wallet → company)

========================= \*/

WITH

/\* --- Optional inputs: empty by default, 1 row when provided --- \*/

start\_input AS (

SELECT NULL::date AS start\_date WHERE FALSE

[[ UNION ALL SELECT {{start\_date}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_date WHERE FALSE

[[ UNION ALL SELECT {{end\_date}}::date ]]

),

/\* --- Normalize window (no inputs → last 31 days to today) --- \*/

bounds\_raw AS (

SELECT

COALESCE((SELECT MAX(end\_date) FROM end\_input), CURRENT\_DATE) AS end\_date\_raw,

(SELECT MAX(start\_date) FROM start\_input) AS start\_date\_raw

),

bounds AS (

SELECT

end\_date\_raw AS end\_date,

/\* default start = end - 31 days; if user set start, clamp to <= end \*/

CASE

WHEN start\_date\_raw IS NULL THEN end\_date\_raw - INTERVAL '31 day'

WHEN start\_date\_raw > end\_date\_raw THEN end\_date\_raw -- guard: swap if inverted

ELSE start\_date\_raw

END AS start\_date

FROM bounds\_raw

),

/\* --- Daily series over the chosen window --- \*/

date\_series AS (

SELECT

d::date AS report\_date,

d AS start\_ts,

LEAST(d + INTERVAL '1 day', NOW()) AS end\_ts

FROM generate\_series(

(SELECT start\_date FROM bounds),

(SELECT end\_date FROM bounds),

INTERVAL '1 day'

) AS d

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND CONCAT(players.os, ' / ', players.browser) = {{registration\_launcher}} ]]

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* ---------- GLOBAL FTD ---------- \*/

ftd\_first AS (

SELECT

t.player\_id,

MIN(t.created\_at) AS first\_deposit\_ts

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

/\* bring players/companies (not aliased) for currency fallback + widget \*/

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.status = 'completed'

AND t.balance\_type = 'withdrawable'

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY t.player\_id

),

player\_reg AS (

SELECT

p.id AS player\_id,

p.created\_at AS registration\_ts,

p.email\_verified,

c.name AS brand\_name

FROM players p

INNER JOIN filtered\_players fp ON p.id = fp.player\_id

LEFT JOIN companies c ON p.company\_id = c.id

),

registrations AS (

SELECT

ds.report\_date,

COUNT(pr.\*) AS total\_registrations,

COUNT(CASE WHEN pr.email\_verified = TRUE THEN 1 END) AS complete\_registrations

FROM date\_series ds

LEFT JOIN player\_reg pr

ON pr.registration\_ts >= ds.start\_ts

AND pr.registration\_ts < ds.end\_ts

GROUP BY ds.report\_date

),

ftds AS (

SELECT

DATE\_TRUNC('day', ff.first\_deposit\_ts)::date AS report\_date,

pr.player\_id,

pr.registration\_ts,

ff.first\_deposit\_ts

FROM ftd\_first ff

JOIN player\_reg pr ON pr.player\_id = ff.player\_id

),

ftd\_metrics AS (

SELECT

ds.report\_date

, COUNT(DISTINCT f.player\_id) AS ftds\_count

, COUNT(\*) FILTER (WHERE DATE\_TRUNC('month', f.registration\_ts) = DATE\_TRUNC('month', f.first\_deposit\_ts)) AS new\_ftds

, COUNT(\*) FILTER (WHERE DATE\_TRUNC('month', f.registration\_ts) < DATE\_TRUNC('month', f.first\_deposit\_ts)) AS old\_ftds

, COUNT(\*) FILTER (WHERE DATE\_TRUNC('day', f.registration\_ts) = DATE\_TRUNC('day', f.first\_deposit\_ts)) AS d0\_ftds

, COUNT(\*) FILTER (WHERE DATE\_TRUNC('day', f.registration\_ts) <> DATE\_TRUNC('day', f.first\_deposit\_ts)) AS late\_ftds

FROM date\_series ds

LEFT JOIN ftds f ON f.report\_date = ds.report\_date

GROUP BY ds.report\_date

),

/\* ---------- DEPOSITS ---------- \*/

deposit\_metrics AS (

SELECT

ds.report\_date,

COUNT(DISTINCT t.player\_id) FILTER (

WHERE t.transaction\_category='deposit'

AND t.transaction\_type='credit'

AND t.status='completed'

AND t.balance\_type='withdrawable'

) AS unique\_depositors,

COUNT(\*) FILTER (

WHERE t.transaction\_category='deposit'

AND t.transaction\_type='credit'

AND t.status='completed'

AND t.balance\_type='withdrawable'

) AS deposits\_count,

COALESCE(SUM(CASE

WHEN t.transaction\_category='deposit'

AND t.transaction\_type='credit'

AND t.status='completed'

AND t.balance\_type='withdrawable'

THEN t.amount END), 0) AS deposits\_amount

FROM date\_series ds

LEFT JOIN transactions t

ON t.created\_at >= ds.start\_ts

AND t.created\_at < ds.end\_ts

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY ds.report\_date

),

/\* ---------- WITHDRAWALS ---------- \*/

withdrawal\_metrics AS (

SELECT

ds.report\_date,

COUNT(t.id) FILTER (

WHERE t.transaction\_category='withdrawal'

AND t.transaction\_type='debit'

AND t.balance\_type='withdrawable'

AND t.status='completed'

) AS withdrawals\_count,

COALESCE(SUM(ABS(t.amount)) FILTER (

WHERE t.transaction\_category='withdrawal'

AND t.transaction\_type='debit'

AND t.balance\_type='withdrawable'

AND t.status='completed'

), 0) AS withdrawals\_amount,

COALESCE(SUM(ABS(t.amount)) FILTER (

WHERE t.transaction\_category='withdrawal'

AND t.transaction\_type='debit'

AND t.balance\_type='withdrawable'

AND t.status='cancelled'

), 0) AS withdrawals\_cancelled

FROM date\_series ds

LEFT JOIN transactions t

ON t.created\_at >= ds.start\_ts

AND t.created\_at < ds.end\_ts

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY ds.report\_date

),

/\* ---------- ACTIVE PLAYERS & BETTING ---------- \*/

active\_players AS (

SELECT

ds.report\_date,

COUNT(DISTINCT CASE WHEN t.transaction\_category='game\_bet' THEN t.player\_id END) AS active\_players\_count,

COUNT(DISTINCT CASE WHEN t.transaction\_category='game\_bet' AND t.balance\_type='withdrawable' THEN t.player\_id END) AS real\_active\_players

FROM date\_series ds

LEFT JOIN transactions t

ON t.created\_at >= ds.start\_ts

AND t.created\_at < ds.end\_ts

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY ds.report\_date

),

betting\_metrics AS (

SELECT

ds.report\_date,

COALESCE(SUM(CASE

WHEN t.transaction\_type='debit'

AND t.transaction\_category='game\_bet'

AND t.balance\_type='withdrawable'

AND t.status='completed'

THEN t.amount END), 0) AS cash\_bet,

COALESCE(SUM(CASE

WHEN t.transaction\_type='credit'

AND t.transaction\_category='game\_bet'

AND t.balance\_type='withdrawable'

AND t.status='completed'

THEN t.amount END), 0) AS cash\_win,

-- promo\_bet: bets using non-withdrawable balance (kept as the correct metric)

COALESCE(SUM(CASE

WHEN t.transaction\_type='debit'

AND t.transaction\_category='bonus'

AND t.balance\_type='non-withdrawable'

AND t.status='completed'

THEN t.amount END), 0) AS promo\_bet,

-- promo\_win: ADJUSTED to be Bonus Credits + Free Spin Winnings (completed free spins),

-- i.e. include transactions where transaction\_category = 'bonus' and are credits & completed.

COALESCE(SUM(CASE

WHEN t.transaction\_type='credit'

AND t.status='completed'

AND t.balance\_type='non-withdrawable'

AND t.transaction\_category = 'bonus'

THEN t.amount END), 0) AS promo\_win

FROM date\_series ds

LEFT JOIN transactions t

ON t.created\_at >= ds.start\_ts

AND t.created\_at < ds.end\_ts

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY ds.report\_date

),

/\* ---------- BONUS CONVERTED (wagering completions only) ---------- \*/

bonus\_converted AS (

SELECT

ds.report\_date,

COALESCE(SUM(CASE

WHEN t.transaction\_type='credit'

AND t.transaction\_category='bonus\_completion'

AND t.status='completed'

AND t.balance\_type='withdrawable'

THEN t.amount END), 0) AS bonus\_converted\_amount

FROM date\_series ds

LEFT JOIN transactions t

ON t.created\_at >= ds.start\_ts

AND t.created\_at < ds.end\_ts

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY ds.report\_date

),

/\* ---------- BONUS COST (ALL bonus-origin money that became withdrawable) ---------- \*/

bonus\_cost AS (

SELECT

ds.report\_date,

COALESCE(SUM(CASE

WHEN t.transaction\_type='credit'

AND t.balance\_type='withdrawable'

AND t.status='completed'

AND t.transaction\_category='bonus\_completion'

THEN t.amount END), 0) AS total\_bonus\_cost

FROM date\_series ds

LEFT JOIN transactions t

ON t.created\_at >= ds.start\_ts

AND t.created\_at < ds.end\_ts

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY ds.report\_date

),

/\* ---------- PREPARE DAILY DATA ---------- \*/

daily\_data AS (

SELECT

0 as sort\_order, -- Add sort column

ds.report\_date::text AS "Date",

COALESCE(r.total\_registrations, 0) AS "#Registrations",

COALESCE(fm.ftds\_count, 0) AS "#FTDs",

COALESCE(fm.new\_ftds, 0) AS "#New FTDs",

ROUND(CASE WHEN COALESCE(fm.ftds\_count,0) > 0

THEN fm.new\_ftds::numeric / fm.ftds\_count \* 100 ELSE 0 END, 2) AS "%New FTDs",

COALESCE(fm.old\_ftds, 0) AS "#Old FTDs",

ROUND(CASE WHEN COALESCE(fm.ftds\_count,0) > 0

THEN fm.old\_ftds::numeric / fm.ftds\_count \* 100 ELSE 0 END, 2) AS "% Old FTDs",

COALESCE(fm.d0\_ftds, 0) AS "#D0 FTDs",

ROUND(CASE WHEN COALESCE(fm.ftds\_count,0) > 0

THEN fm.d0\_ftds::numeric / fm.ftds\_count \* 100 ELSE 0 END, 2) AS "%D0 FTDs",

COALESCE(fm.late\_ftds, 0) AS "#Late FTDs",

ROUND(CASE WHEN COALESCE(r.total\_registrations,0) > 0

THEN COALESCE(fm.ftds\_count,0)::numeric / r.total\_registrations \* 100 ELSE 0 END, 2) AS "%Conversion total reg",

ROUND(CASE WHEN COALESCE(r.complete\_registrations,0) > 0

THEN COALESCE(fm.ftds\_count,0)::numeric / r.complete\_registrations \* 100 ELSE 0 END, 2) AS "%Conversion complete reg",

COALESCE(dm.unique\_depositors, 0) AS "Unique Depositors",

COALESCE(dm.deposits\_count, 0) AS "#Deposits",

ROUND(COALESCE(dm.deposits\_amount, 0), 2) AS "Deposits Amount",

COALESCE(wm.withdrawals\_count, 0) AS "#Withdrawals",

ROUND(COALESCE(wm.withdrawals\_amount, 0), 2) AS "Withdrawals Amount",

ROUND(COALESCE(wm.withdrawals\_cancelled, 0), 2) AS "Withdrawals Amount Canceled",

ROUND(CASE WHEN COALESCE(dm.deposits\_amount,0) > 0

THEN COALESCE(wm.withdrawals\_amount,0) / dm.deposits\_amount \* 100 ELSE 0 END, 2) AS "%Withdrawals/Deposits",

ROUND(COALESCE(dm.deposits\_amount,0) - COALESCE(wm.withdrawals\_amount, 0), 2) AS "CashFlow",

COALESCE(ap.active\_players\_count, 0) AS "Active Players",

COALESCE(ap.real\_active\_players, 0) AS "Real Active Players",

ROUND(COALESCE(bet.cash\_bet, 0), 2) AS "Cash Bet",

ROUND(COALESCE(bet.cash\_win, 0), 2) AS "Cash Win",

ROUND(COALESCE(bet.promo\_bet, 0), 2) AS "Promo bet",

ROUND(COALESCE(bet.promo\_win, 0), 2) AS "Promo Win",

ROUND(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0), 2) AS "Turnover",

ROUND(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0), 2) AS "Turnover Casino",

ROUND(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)

- COALESCE(bet.cash\_win,0)

- COALESCE(bet.promo\_win,0), 2) AS "GGR",

ROUND(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)

- COALESCE(bet.cash\_win,0)

- COALESCE(bet.promo\_win,0), 2) AS "GGR Casino",

ROUND(COALESCE(bet.cash\_bet,0) - COALESCE(bet.cash\_win,0), 2) AS "Cash GGR",

ROUND(COALESCE(bet.cash\_bet,0) - COALESCE(bet.cash\_win,0), 2) AS "Cash GGR Casino",

ROUND(COALESCE(bc.bonus\_converted\_amount, 0), 2) AS "Bonus Converted",

ROUND(COALESCE(bcost.total\_bonus\_cost, 0), 2) AS "Bonus Cost",

ROUND(CASE

WHEN (COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)

- COALESCE(bet.cash\_win,0)

- COALESCE(bet.promo\_win,0)) > 0

THEN COALESCE(bcost.total\_bonus\_cost,0) /

(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)

- COALESCE(bet.cash\_win,0)

- COALESCE(bet.promo\_win,0)) \* 100

ELSE 0 END, 2) AS "Bonus Ratio (GGR)",

ROUND(CASE WHEN COALESCE(dm.deposits\_amount,0) > 0

THEN COALESCE(bcost.total\_bonus\_cost,0) / dm.deposits\_amount \* 100 ELSE 0 END, 2) AS "Bonus Ratio (Deposits)",

ROUND(CASE

WHEN (COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)) > 0

THEN (COALESCE(bet.cash\_win,0) + COALESCE(bet.promo\_win,0)) /

(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)) \* 100

ELSE 0 END, 2) AS "Payout %",

ROUND(CASE

WHEN (COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)

- COALESCE(bet.cash\_win,0)

- COALESCE(bet.promo\_win,0)) > 0

THEN (COALESCE(dm.deposits\_amount,0) - COALESCE(wm.withdrawals\_amount,0)) /

(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)

- COALESCE(bet.cash\_win,0)

- COALESCE(bet.promo\_win,0)) \* 100

ELSE 0 END, 2) AS "%CashFlow to GGR"

FROM date\_series ds

LEFT JOIN registrations r ON r.report\_date = ds.report\_date

LEFT JOIN ftd\_metrics fm ON fm.report\_date = ds.report\_date

LEFT JOIN deposit\_metrics dm ON dm.report\_date = ds.report\_date

LEFT JOIN withdrawal\_metrics wm ON wm.report\_date = ds.report\_date

LEFT JOIN active\_players ap ON ap.report\_date = ds.report\_date

LEFT JOIN betting\_metrics bet ON bet.report\_date = ds.report\_date

LEFT JOIN bonus\_converted bc ON bc.report\_date = ds.report\_date

LEFT JOIN bonus\_cost bcost ON bcost.report\_date = ds.report\_date

)

/\* ========== FINAL OUTPUT WITH TOTAL ROW ========== \*/

-- TOTAL row at top with proper unique calculations

SELECT

-1 as sort\_order, -- Total gets -1 to appear first

'TOTAL' AS "Date", -- Changed from SUMMARY to TOTAL

SUM("#Registrations") AS "#Registrations",

SUM("#FTDs") AS "#FTDs",

SUM("#New FTDs") AS "#New FTDs",

ROUND(CASE WHEN SUM("#FTDs") > 0

THEN SUM("#New FTDs")::numeric / SUM("#FTDs") \* 100 ELSE 0 END, 2) AS "%New FTDs",

SUM("#Old FTDs") AS "#Old FTDs",

ROUND(CASE WHEN SUM("#FTDs") > 0

THEN SUM("#Old FTDs")::numeric / SUM("#FTDs") \* 100 ELSE 0 END, 2) AS "% Old FTDs",

SUM("#D0 FTDs") AS "#D0 FTDs",

ROUND(CASE WHEN SUM("#FTDs") > 0

THEN SUM("#D0 FTDs")::numeric / SUM("#FTDs") \* 100 ELSE 0 END, 2) AS "%D0 FTDs",

SUM("#Late FTDs") AS "#Late FTDs",

ROUND(CASE WHEN SUM("#Registrations") > 0

THEN SUM("#FTDs")::numeric / SUM("#Registrations") \* 100 ELSE 0 END, 2) AS "%Conversion total reg",

-- FIX: Calculate complete registration conversion from totals

(SELECT ROUND(

CASE WHEN COUNT(CASE WHEN pr.email\_verified = TRUE THEN 1 END) > 0

THEN COUNT(DISTINCT CASE WHEN ff.first\_deposit\_ts IS NOT NULL THEN ff.player\_id END)::numeric /

COUNT(CASE WHEN pr.email\_verified = TRUE THEN 1 END) \* 100

ELSE 0 END, 2)

FROM player\_reg pr

LEFT JOIN ftd\_first ff ON pr.player\_id = ff.player\_id

WHERE pr.registration\_ts >= (SELECT start\_date FROM bounds)

AND pr.registration\_ts < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

) AS "%Conversion complete reg",

-- FIX: Calculate actual unique depositors for entire period

(SELECT COUNT(DISTINCT t.player\_id)

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category='deposit'

AND t.transaction\_type='credit'

AND t.status='completed'

AND t.balance\_type='withdrawable'

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

) AS "Unique Depositors",

SUM("#Deposits") AS "#Deposits",

ROUND(SUM("Deposits Amount"), 2) AS "Deposits Amount",

SUM("#Withdrawals") AS "#Withdrawals",

ROUND(SUM("Withdrawals Amount"), 2) AS "Withdrawals Amount",

ROUND(SUM("Withdrawals Amount Canceled"), 2) AS "Withdrawals Amount Canceled",

ROUND(CASE WHEN SUM("Deposits Amount") > 0

THEN SUM("Withdrawals Amount") / SUM("Deposits Amount") \* 100 ELSE 0 END, 2) AS "%Withdrawals/Deposits",

ROUND(SUM("CashFlow"), 2) AS "CashFlow",

-- FIX: Calculate actual active players for entire period

(SELECT COUNT(DISTINCT t.player\_id)

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category='game\_bet'

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

) AS "Active Players",

-- FIX: Calculate actual real active players for entire period

(SELECT COUNT(DISTINCT t.player\_id)

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category='game\_bet'

AND t.balance\_type='withdrawable'

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

) AS "Real Active Players",

ROUND(SUM("Cash Bet"), 2) AS "Cash Bet",

ROUND(SUM("Cash Win"), 2) AS "Cash Win",

ROUND(SUM("Promo bet"), 2) AS "Promo bet",

ROUND(SUM("Promo Win"), 2) AS "Promo Win",

ROUND(SUM("Turnover"), 2) AS "Turnover",

ROUND(SUM("Turnover Casino"), 2) AS "Turnover Casino",

ROUND(SUM("GGR"), 2) AS "GGR",

ROUND(SUM("GGR Casino"), 2) AS "GGR Casino",

ROUND(SUM("Cash GGR"), 2) AS "Cash GGR",

ROUND(SUM("Cash GGR Casino"), 2) AS "Cash GGR Casino",

ROUND(SUM("Bonus Converted"), 2) AS "Bonus Converted",

ROUND(SUM("Bonus Cost"), 2) AS "Bonus Cost",

ROUND(CASE WHEN SUM("GGR") > 0

THEN SUM("Bonus Cost") / SUM("GGR") \* 100 ELSE 0 END, 2) AS "Bonus Ratio (GGR)",

ROUND(CASE WHEN SUM("Deposits Amount") > 0

THEN SUM("Bonus Cost") / SUM("Deposits Amount") \* 100 ELSE 0 END, 2) AS "Bonus Ratio (Deposits)",

ROUND(CASE WHEN SUM("Turnover") > 0

THEN (SUM("Cash Win") + SUM("Promo Win")) / SUM("Turnover") \* 100 ELSE 0 END, 2) AS "Payout %",

ROUND(CASE WHEN SUM("GGR") > 0

THEN SUM("CashFlow") / SUM("GGR") \* 100 ELSE 0 END, 2) AS "%CashFlow to GGR"

FROM daily\_data

UNION ALL

-- Daily rows

SELECT \* FROM daily\_data

ORDER BY sort\_order, "Date" DESC;

monthly full sql

/\* =========================

MONTHLY KPIs — MULTI-MONTH VIEW with COMPLETE SUMMARY ROW

Each row represents a complete calendar month

Summary row properly calculates unique counts and percentages

BONUS LOGIC:

- Bonus Converted = Original bonus amounts that converted to real money

- Bonus Cost = Total cost including converted bonuses + wins generated from bonus play

========================= \*/

WITH

/\* --- Optional inputs: empty by default, 1 row when provided --- \*/

start\_input AS (

SELECT NULL::date AS start\_date WHERE FALSE

[[ UNION ALL SELECT {{start\_date}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_date WHERE FALSE

[[ UNION ALL SELECT {{end\_date}}::date ]]

),

/\* --- Normalize window (no inputs → last 12 months to current month) --- \*/

bounds\_raw AS (

SELECT

COALESCE((SELECT MAX(end\_date) FROM end\_input), CURRENT\_DATE) AS end\_date\_raw,

(SELECT MAX(start\_date) FROM start\_input) AS start\_date\_raw

),

bounds AS (

SELECT

DATE\_TRUNC('month', end\_date\_raw) + INTERVAL '1 month' - INTERVAL '1 day' AS end\_date,

/\* default start = beginning of month 12 months ago \*/

CASE

WHEN start\_date\_raw IS NULL THEN DATE\_TRUNC('month', end\_date\_raw - INTERVAL '12 months')

WHEN start\_date\_raw > end\_date\_raw THEN DATE\_TRUNC('month', end\_date\_raw)

ELSE DATE\_TRUNC('month', start\_date\_raw)

END AS start\_date

FROM bounds\_raw

),

/\* --- Monthly series over the chosen window --- \*/

month\_series AS (

SELECT

DATE\_TRUNC('month', d)::date AS report\_month,

DATE\_TRUNC('month', d) AS start\_ts,

LEAST(DATE\_TRUNC('month', d) + INTERVAL '1 month', NOW()) AS end\_ts

FROM generate\_series(

(SELECT start\_date FROM bounds),

(SELECT end\_date FROM bounds),

INTERVAL '1 month'

) AS d

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND players.os = {{registration\_launcher}} ]] -- Simplified to OS-only (matching daily report)

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* ---------- GLOBAL FTD ---------- \*/

ftd\_first AS (

SELECT

t.player\_id,

MIN(t.created\_at) AS first\_deposit\_ts

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

/\* bring players/companies (not aliased) for currency fallback + widget \*/

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.status = 'completed'

AND t.balance\_type = 'withdrawable'

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY t.player\_id

),

player\_reg AS (

SELECT

p.id AS player\_id,

p.created\_at AS registration\_ts,

p.email\_verified,

c.name AS brand\_name

FROM players p

INNER JOIN filtered\_players fp ON p.id = fp.player\_id

LEFT JOIN companies c ON p.company\_id = c.id

),

registrations AS (

SELECT

ms.report\_month,

COUNT(pr.\*) AS total\_registrations,

COUNT(CASE WHEN pr.email\_verified = TRUE THEN 1 END) AS complete\_registrations

FROM month\_series ms

LEFT JOIN player\_reg pr

ON pr.registration\_ts >= ms.start\_ts

AND pr.registration\_ts < ms.end\_ts

GROUP BY ms.report\_month

),

ftds AS (

SELECT

DATE\_TRUNC('month', ff.first\_deposit\_ts)::date AS report\_month,

pr.player\_id,

pr.registration\_ts,

ff.first\_deposit\_ts

FROM ftd\_first ff

JOIN player\_reg pr ON pr.player\_id = ff.player\_id

),

ftd\_metrics AS (

SELECT

ms.report\_month

, COUNT(DISTINCT f.player\_id) AS ftds\_count

, COUNT(\*) FILTER (WHERE DATE\_TRUNC('month', f.registration\_ts) = DATE\_TRUNC('month', f.first\_deposit\_ts)) AS new\_ftds

, COUNT(\*) FILTER (WHERE DATE\_TRUNC('month', f.registration\_ts) < DATE\_TRUNC('month', f.first\_deposit\_ts)) AS old\_ftds

, COUNT(\*) FILTER (WHERE DATE\_TRUNC('day', f.registration\_ts) = DATE\_TRUNC('day', f.first\_deposit\_ts)) AS d0\_ftds

, COUNT(\*) FILTER (WHERE DATE\_TRUNC('day', f.registration\_ts) <> DATE\_TRUNC('day', f.first\_deposit\_ts)) AS late\_ftds

FROM month\_series ms

LEFT JOIN ftds f ON f.report\_month = ms.report\_month

GROUP BY ms.report\_month

),

/\* ---------- DEPOSITS ---------- \*/

deposit\_metrics AS (

SELECT

ms.report\_month,

COUNT(DISTINCT t.player\_id) FILTER (

WHERE t.transaction\_category='deposit'

AND t.transaction\_type='credit'

AND t.status='completed'

AND t.balance\_type='withdrawable'

) AS unique\_depositors,

COUNT(\*) FILTER (

WHERE t.transaction\_category='deposit'

AND t.transaction\_type='credit'

AND t.status='completed'

AND t.balance\_type='withdrawable'

) AS deposits\_count,

COALESCE(SUM(CASE

WHEN t.transaction\_category='deposit'

AND t.transaction\_type='credit'

AND t.status='completed'

AND t.balance\_type='withdrawable'

THEN t.amount END), 0) AS deposits\_amount

FROM month\_series ms

LEFT JOIN transactions t

ON t.created\_at >= ms.start\_ts

AND t.created\_at < ms.end\_ts

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY ms.report\_month

),

/\* ---------- WITHDRAWALS ---------- \*/

withdrawal\_metrics AS (

SELECT

ms.report\_month,

COUNT(t.id) FILTER (

WHERE t.transaction\_category='withdrawal'

AND t.transaction\_type='debit'

AND t.balance\_type='withdrawable'

AND t.status='completed'

) AS withdrawals\_count,

COALESCE(SUM(ABS(t.amount)) FILTER (

WHERE t.transaction\_category='withdrawal'

AND t.transaction\_type='debit'

AND t.balance\_type='withdrawable'

AND t.status='completed'

), 0) AS withdrawals\_amount,

COALESCE(SUM(ABS(t.amount)) FILTER (

WHERE t.transaction\_category='withdrawal'

AND t.transaction\_type='debit'

AND t.balance\_type='withdrawable'

AND t.status='cancelled'

), 0) AS withdrawals\_cancelled

FROM month\_series ms

LEFT JOIN transactions t

ON t.created\_at >= ms.start\_ts

AND t.created\_at < ms.end\_ts

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY ms.report\_month

),

/\* ---------- ACTIVE PLAYERS & BETTING ---------- \*/

active\_players AS (

SELECT

ms.report\_month,

COUNT(DISTINCT CASE WHEN t.transaction\_category='game\_bet' THEN t.player\_id END) AS active\_players\_count,

COUNT(DISTINCT CASE WHEN t.transaction\_category='game\_bet' AND t.balance\_type='withdrawable' THEN t.player\_id END) AS real\_active\_players

FROM month\_series ms

LEFT JOIN transactions t

ON t.created\_at >= ms.start\_ts

AND t.created\_at < ms.end\_ts

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY ms.report\_month

),

betting\_metrics AS (

SELECT

ms.report\_month,

COALESCE(SUM(CASE

WHEN t.transaction\_type='debit'

AND t.transaction\_category='game\_bet'

AND t.balance\_type='withdrawable'

AND t.status='completed'

THEN t.amount END), 0) AS cash\_bet,

COALESCE(SUM(CASE

WHEN t.transaction\_type='credit'

AND t.transaction\_category='game\_bet' -- Matching daily: game\_bet (not game\_win)

AND t.balance\_type='withdrawable'

AND t.status='completed'

THEN t.amount END), 0) AS cash\_win,

-- promo\_bet: matching daily report exactly

COALESCE(SUM(CASE

WHEN t.transaction\_type='debit'

AND t.transaction\_category='bonus' -- Matching daily: bonus category

AND t.balance\_type='non-withdrawable'

AND t.status='completed'

THEN t.amount END), 0) AS promo\_bet,

-- promo\_win: matching daily report exactly

COALESCE(SUM(CASE

WHEN t.transaction\_type='credit'

AND t.status='completed'

AND t.balance\_type='non-withdrawable'

AND t.transaction\_category = 'bonus' -- Matching daily: bonus category

THEN t.amount END), 0) AS promo\_win

FROM month\_series ms

LEFT JOIN transactions t

ON t.created\_at >= ms.start\_ts

AND t.created\_at < ms.end\_ts

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY ms.report\_month

),

/\* ---------- BONUS CONVERTED (wagering completions only) ---------- \*/

/\* This represents the original bonus amounts that successfully converted to real money \*/

bonus\_converted AS (

SELECT

ms.report\_month,

-- For now, we use bonus\_completion as a proxy for converted bonuses

-- In an ideal world, this would track only the original bonus portion

COALESCE(SUM(CASE

WHEN t.transaction\_type='credit'

AND t.transaction\_category='bonus\_completion'

AND t.status='completed'

AND t.balance\_type='withdrawable'

THEN t.amount END), 0) AS bonus\_converted\_amount

FROM month\_series ms

LEFT JOIN transactions t

ON t.created\_at >= ms.start\_ts

AND t.created\_at < ms.end\_ts

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY ms.report\_month

),

/\* ---------- BONUS COST (Total cost = converted bonuses + wins from bonus play) ---------- \*/

/\* Since bonus\_completion includes both original bonus AND wins made with that bonus,

this is actually our total bonus cost \*/

bonus\_cost AS (

SELECT

ms.report\_month,

-- bonus\_completion represents the total amount that became withdrawable from bonus play

-- This includes both the original bonus AND any wins generated while using that bonus

COALESCE(SUM(CASE

WHEN t.transaction\_type='credit'

AND t.transaction\_category='bonus\_completion'

AND t.status='completed'

AND t.balance\_type='withdrawable'

THEN t.amount END), 0) AS total\_bonus\_cost

FROM month\_series ms

LEFT JOIN transactions t

ON t.created\_at >= ms.start\_ts

AND t.created\_at < ms.end\_ts

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY ms.report\_month

),

/\* ---------- PREPARE MONTHLY DATA ---------- \*/

monthly\_data AS (

SELECT

0 as sort\_order,

TO\_CHAR(ms.report\_month, 'YYYY-MM') AS "Month",

COALESCE(r.total\_registrations, 0) AS "#Registrations",

COALESCE(fm.ftds\_count, 0) AS "#FTDs",

COALESCE(fm.new\_ftds, 0) AS "#New FTDs",

ROUND(CASE WHEN COALESCE(fm.ftds\_count,0) > 0

THEN fm.new\_ftds::numeric / fm.ftds\_count \* 100 ELSE 0 END, 2) AS "%New FTDs",

COALESCE(fm.old\_ftds, 0) AS "#Old FTDs",

ROUND(CASE WHEN COALESCE(fm.ftds\_count,0) > 0

THEN fm.old\_ftds::numeric / fm.ftds\_count \* 100 ELSE 0 END, 2) AS "% Old FTDs",

COALESCE(fm.d0\_ftds, 0) AS "#D0 FTDs",

ROUND(CASE WHEN COALESCE(fm.ftds\_count,0) > 0

THEN fm.d0\_ftds::numeric / fm.ftds\_count \* 100 ELSE 0 END, 2) AS "%D0 FTDs",

COALESCE(fm.late\_ftds, 0) AS "#Late FTDs",

ROUND(CASE WHEN COALESCE(r.total\_registrations,0) > 0

THEN COALESCE(fm.ftds\_count,0)::numeric / r.total\_registrations \* 100 ELSE 0 END, 2) AS "%Conversion total reg",

ROUND(CASE WHEN COALESCE(r.complete\_registrations,0) > 0

THEN COALESCE(fm.ftds\_count,0)::numeric / r.complete\_registrations \* 100 ELSE 0 END, 2) AS "%Conversion complete reg",

COALESCE(dm.unique\_depositors, 0) AS "Unique Depositors",

COALESCE(dm.deposits\_count, 0) AS "#Deposits",

ROUND(COALESCE(dm.deposits\_amount, 0), 2) AS "Deposits Amount",

COALESCE(wm.withdrawals\_count, 0) AS "#Withdrawals",

ROUND(COALESCE(wm.withdrawals\_amount, 0), 2) AS "Withdrawals Amount",

ROUND(COALESCE(wm.withdrawals\_cancelled, 0), 2) AS "Withdrawals Amount Canceled",

ROUND(CASE WHEN COALESCE(dm.deposits\_amount,0) > 0

THEN COALESCE(wm.withdrawals\_amount,0) / dm.deposits\_amount \* 100 ELSE 0 END, 2) AS "%Withdrawals/Deposits",

ROUND(COALESCE(dm.deposits\_amount,0) - COALESCE(wm.withdrawals\_amount, 0), 2) AS "CashFlow",

COALESCE(ap.active\_players\_count, 0) AS "Active Players",

COALESCE(ap.real\_active\_players, 0) AS "Real Active Players",

ROUND(COALESCE(bet.cash\_bet, 0), 2) AS "Cash Bet",

ROUND(COALESCE(bet.cash\_win, 0), 2) AS "Cash Win",

ROUND(COALESCE(bet.promo\_bet, 0), 2) AS "Promo bet",

ROUND(COALESCE(bet.promo\_win, 0), 2) AS "Promo Win",

ROUND(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0), 2) AS "Turnover",

ROUND(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0), 2) AS "Turnover Casino",

ROUND(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)

- COALESCE(bet.cash\_win,0)

- COALESCE(bet.promo\_win,0), 2) AS "GGR",

ROUND(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)

- COALESCE(bet.cash\_win,0)

- COALESCE(bet.promo\_win,0), 2) AS "GGR Casino",

ROUND(COALESCE(bet.cash\_bet,0) - COALESCE(bet.cash\_win,0), 2) AS "Cash GGR",

ROUND(COALESCE(bet.cash\_bet,0) - COALESCE(bet.cash\_win,0), 2) AS "Cash GGR Casino",

-- Bonus Converted: Original bonus amounts that converted (currently using bonus\_completion as proxy)

ROUND(COALESCE(bc.bonus\_converted\_amount, 0), 2) AS "Bonus Converted",

-- Bonus Cost: Total cost including converted bonuses + wins (bonus\_completion includes both)

ROUND(COALESCE(bcost.total\_bonus\_cost, 0), 2) AS "Bonus Cost",

ROUND(CASE

WHEN (COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)

- COALESCE(bet.cash\_win,0)

- COALESCE(bet.promo\_win,0)) > 0

THEN COALESCE(bcost.total\_bonus\_cost,0) /

(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)

- COALESCE(bet.cash\_win,0)

- COALESCE(bet.promo\_win,0)) \* 100

ELSE 0 END, 2) AS "Bonus Ratio (GGR)",

ROUND(CASE WHEN COALESCE(dm.deposits\_amount,0) > 0

THEN COALESCE(bcost.total\_bonus\_cost,0) / dm.deposits\_amount \* 100 ELSE 0 END, 2) AS "Bonus Ratio (Deposits)",

ROUND(CASE

WHEN (COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)) > 0

THEN (COALESCE(bet.cash\_win,0) + COALESCE(bet.promo\_win,0)) /

(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)) \* 100

ELSE 0 END, 2) AS "Payout %",

ROUND(CASE

WHEN (COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)

- COALESCE(bet.cash\_win,0)

- COALESCE(bet.promo\_win,0)) > 0

THEN (COALESCE(dm.deposits\_amount,0) - COALESCE(wm.withdrawals\_amount,0)) /

(COALESCE(bet.cash\_bet,0) + COALESCE(bet.promo\_bet,0)

- COALESCE(bet.cash\_win,0)

- COALESCE(bet.promo\_win,0)) \* 100

ELSE 0 END, 2) AS "%CashFlow to GGR"

FROM month\_series ms

LEFT JOIN registrations r ON r.report\_month = ms.report\_month

LEFT JOIN ftd\_metrics fm ON fm.report\_month = ms.report\_month

LEFT JOIN deposit\_metrics dm ON dm.report\_month = ms.report\_month

LEFT JOIN withdrawal\_metrics wm ON wm.report\_month = ms.report\_month

LEFT JOIN active\_players ap ON ap.report\_month = ms.report\_month

LEFT JOIN betting\_metrics bet ON bet.report\_month = ms.report\_month

LEFT JOIN bonus\_converted bc ON bc.report\_month = ms.report\_month

LEFT JOIN bonus\_cost bcost ON bcost.report\_month = ms.report\_month

)

/\* ========== FINAL OUTPUT WITH COMPLETE SUMMARY ROW ========== \*/

-- Summary row with proper unique count calculations

SELECT

-1 as sort\_order,

'TOTAL' AS "Month",

SUM("#Registrations") AS "#Registrations",

SUM("#FTDs") AS "#FTDs",

SUM("#New FTDs") AS "#New FTDs",

ROUND(CASE WHEN SUM("#FTDs") > 0

THEN SUM("#New FTDs")::numeric / SUM("#FTDs") \* 100 ELSE 0 END, 2) AS "%New FTDs",

SUM("#Old FTDs") AS "#Old FTDs",

ROUND(CASE WHEN SUM("#FTDs") > 0

THEN SUM("#Old FTDs")::numeric / SUM("#FTDs") \* 100 ELSE 0 END, 2) AS "% Old FTDs",

SUM("#D0 FTDs") AS "#D0 FTDs",

ROUND(CASE WHEN SUM("#FTDs") > 0

THEN SUM("#D0 FTDs")::numeric / SUM("#FTDs") \* 100 ELSE 0 END, 2) AS "%D0 FTDs",

SUM("#Late FTDs") AS "#Late FTDs",

ROUND(CASE WHEN SUM("#Registrations") > 0

THEN SUM("#FTDs")::numeric / SUM("#Registrations") \* 100 ELSE 0 END, 2) AS "%Conversion total reg",

-- FIX 1: Calculate complete registration conversion from totals

(SELECT ROUND(

CASE WHEN COUNT(CASE WHEN pr.email\_verified = TRUE THEN 1 END) > 0

THEN COUNT(DISTINCT CASE WHEN ff.first\_deposit\_ts IS NOT NULL THEN ff.player\_id END)::numeric /

COUNT(CASE WHEN pr.email\_verified = TRUE THEN 1 END) \* 100

ELSE 0 END, 2)

FROM player\_reg pr

LEFT JOIN ftd\_first ff ON pr.player\_id = ff.player\_id

WHERE pr.registration\_ts >= (SELECT start\_date FROM bounds)

AND pr.registration\_ts < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

) AS "%Conversion complete reg",

-- FIX 2: Calculate actual unique depositors for entire period

(SELECT COUNT(DISTINCT t.player\_id)

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category='deposit'

AND t.transaction\_type='credit'

AND t.status='completed'

AND t.balance\_type='withdrawable'

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

) AS "Unique Depositors",

SUM("#Deposits") AS "#Deposits",

ROUND(SUM("Deposits Amount"), 2) AS "Deposits Amount",

SUM("#Withdrawals") AS "#Withdrawals",

ROUND(SUM("Withdrawals Amount"), 2) AS "Withdrawals Amount",

ROUND(SUM("Withdrawals Amount Canceled"), 2) AS "Withdrawals Amount Canceled",

ROUND(CASE WHEN SUM("Deposits Amount") > 0

THEN SUM("Withdrawals Amount") / SUM("Deposits Amount") \* 100 ELSE 0 END, 2) AS "%Withdrawals/Deposits",

ROUND(SUM("CashFlow"), 2) AS "CashFlow",

-- FIX 3: Calculate actual active players for entire period

(SELECT COUNT(DISTINCT t.player\_id)

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category='game\_bet'

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

) AS "Active Players",

-- FIX 4: Calculate actual real active players for entire period

(SELECT COUNT(DISTINCT t.player\_id)

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category='game\_bet'

AND t.balance\_type='withdrawable'

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

) AS "Real Active Players",

ROUND(SUM("Cash Bet"), 2) AS "Cash Bet",

ROUND(SUM("Cash Win"), 2) AS "Cash Win",

ROUND(SUM("Promo bet"), 2) AS "Promo bet",

ROUND(SUM("Promo Win"), 2) AS "Promo Win",

ROUND(SUM("Turnover"), 2) AS "Turnover",

ROUND(SUM("Turnover Casino"), 2) AS "Turnover Casino",

ROUND(SUM("GGR"), 2) AS "GGR",

ROUND(SUM("GGR Casino"), 2) AS "GGR Casino",

ROUND(SUM("Cash GGR"), 2) AS "Cash GGR",

ROUND(SUM("Cash GGR Casino"), 2) AS "Cash GGR Casino",

ROUND(SUM("Bonus Converted"), 2) AS "Bonus Converted",

ROUND(SUM("Bonus Cost"), 2) AS "Bonus Cost",

ROUND(CASE WHEN SUM("GGR") > 0

THEN SUM("Bonus Cost") / SUM("GGR") \* 100 ELSE 0 END, 2) AS "Bonus Ratio (GGR)",

ROUND(CASE WHEN SUM("Deposits Amount") > 0

THEN SUM("Bonus Cost") / SUM("Deposits Amount") \* 100 ELSE 0 END, 2) AS "Bonus Ratio (Deposits)",

ROUND(CASE WHEN SUM("Turnover") > 0

THEN (SUM("Cash Win") + SUM("Promo Win")) / SUM("Turnover") \* 100 ELSE 0 END, 2) AS "Payout %",

ROUND(CASE WHEN SUM("GGR") > 0

THEN SUM("CashFlow") / SUM("GGR") \* 100 ELSE 0 END, 2) AS "%CashFlow to GGR"

FROM monthly\_data

UNION ALL

-- Monthly rows

SELECT \* FROM monthly\_data

ORDER BY sort\_order, "Month" DESC;

CASH PLAYERS COHORT

/\* ============================================

CASH PLAYERS COHORT - ALIGNED WITH DAILY/MONTHLY FILTERS

Shows retention of players making cash bets over time

============================================ \*/

WITH

/\* --- Optional date inputs for cohort window --- \*/

start\_input AS (

SELECT NULL::date AS start\_date WHERE FALSE

[[ UNION ALL SELECT {{start\_date}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_date WHERE FALSE

[[ UNION ALL SELECT {{end\_date}}::date ]]

),

/\* --- Normalize cohort window (default: last 12 months to current month) --- \*/

bounds AS (

SELECT

COALESCE((SELECT start\_date FROM start\_input),

DATE\_TRUNC('month', CURRENT\_DATE - INTERVAL '12 months')) AS start\_date,

COALESCE((SELECT end\_date FROM end\_input),

DATE\_TRUNC('month', CURRENT\_DATE)) AS end\_date

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND CONCAT(players.os, ' / ', players.browser) = {{registration\_launcher}} ]]

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* Step 1: Identify first cash bets with currency filter \*/

first\_cash\_bets AS (

SELECT

t.player\_id,

DATE\_TRUNC('month', MIN(t.created\_at)) as first\_cash\_bet\_month,

MIN(t.created\_at) as first\_cash\_bet\_date

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'game\_bet'

AND t.transaction\_type = 'debit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

-- Apply cohort date bounds

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

-- Currency filter using same resolution as daily/monthly

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY t.player\_id

),

/\* Step 2: Calculate cohort sizes \*/

cohort\_sizes AS (

SELECT

first\_cash\_bet\_month as cohort\_month,

COUNT(DISTINCT player\_id) as cohort\_size

FROM first\_cash\_bets

GROUP BY first\_cash\_bet\_month

),

/\* Step 3: Track activity for each cohort \*/

cohort\_activity AS (

SELECT

fcb.first\_cash\_bet\_month as cohort\_month,

DATE\_TRUNC('month', t.created\_at) as activity\_month,

COUNT(DISTINCT t.player\_id) as active\_players

FROM first\_cash\_bets fcb

INNER JOIN transactions t ON fcb.player\_id = t.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'game\_bet'

AND t.transaction\_type = 'debit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

AND t.created\_at >= fcb.first\_cash\_bet\_date

-- Apply same currency filter

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY fcb.first\_cash\_bet\_month, DATE\_TRUNC('month', t.created\_at)

),

/\* Step 4: Calculate retention \*/

cohort\_retention AS (

SELECT

ca.cohort\_month,

ca.activity\_month,

ca.active\_players,

cs.cohort\_size,

EXTRACT(YEAR FROM AGE(ca.activity\_month, ca.cohort\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(ca.activity\_month, ca.cohort\_month)) as months\_since\_first\_bet

FROM cohort\_activity ca

INNER JOIN cohort\_sizes cs ON ca.cohort\_month = cs.cohort\_month

WHERE EXTRACT(YEAR FROM AGE(ca.activity\_month, ca.cohort\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(ca.activity\_month, ca.cohort\_month)) <= 12

)

/\* Step 5: Pivot for final output \*/

SELECT

TO\_CHAR(cohort\_month, 'Month YYYY') as "FIRST CASH BET MONTH",

MAX(CASE WHEN months\_since\_first\_bet = 0 THEN active\_players END) as "Month 0",

MAX(CASE WHEN months\_since\_first\_bet = 1 THEN active\_players END) as "Month 1",

MAX(CASE WHEN months\_since\_first\_bet = 2 THEN active\_players END) as "Month 2",

MAX(CASE WHEN months\_since\_first\_bet = 3 THEN active\_players END) as "Month 3",

MAX(CASE WHEN months\_since\_first\_bet = 4 THEN active\_players END) as "Month 4",

MAX(CASE WHEN months\_since\_first\_bet = 5 THEN active\_players END) as "Month 5",

MAX(CASE WHEN months\_since\_first\_bet = 6 THEN active\_players END) as "Month 6",

MAX(CASE WHEN months\_since\_first\_bet = 7 THEN active\_players END) as "Month 7",

MAX(CASE WHEN months\_since\_first\_bet = 8 THEN active\_players END) as "Month 8",

MAX(CASE WHEN months\_since\_first\_bet = 9 THEN active\_players END) as "Month 9",

MAX(CASE WHEN months\_since\_first\_bet = 10 THEN active\_players END) as "Month 10",

MAX(CASE WHEN months\_since\_first\_bet = 11 THEN active\_players END) as "Month 11",

MAX(CASE WHEN months\_since\_first\_bet = 12 THEN active\_players END) as "Month 12"

FROM cohort\_retention

GROUP BY cohort\_month

ORDER BY cohort\_month;

CASH PLAYERS COHORT (%)

/\* ============================================

CASH PLAYERS COHORT (%) - NUMERIC VERSION FOR CONDITIONAL FORMATTING

Shows retention percentages as numbers (Metabase will format as %)

============================================ \*/

WITH

/\* --- Optional date inputs for cohort window --- \*/

start\_input AS (

SELECT NULL::date AS start\_date WHERE FALSE

[[ UNION ALL SELECT {{start\_date}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_date WHERE FALSE

[[ UNION ALL SELECT {{end\_date}}::date ]]

),

/\* --- Normalize cohort window (default: last 12 months to current month) --- \*/

bounds AS (

SELECT

COALESCE((SELECT start\_date FROM start\_input),

DATE\_TRUNC('month', CURRENT\_DATE - INTERVAL '12 months')) AS start\_date,

COALESCE((SELECT end\_date FROM end\_input),

DATE\_TRUNC('month', CURRENT\_DATE)) AS end\_date

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND CONCAT(players.os, ' / ', players.browser) = {{registration\_launcher}} ]]

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* Step 1: Identify first cash bets with currency filter \*/

first\_cash\_bets AS (

SELECT

t.player\_id,

DATE\_TRUNC('month', MIN(t.created\_at)) as first\_cash\_bet\_month,

MIN(t.created\_at) as first\_cash\_bet\_date

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'game\_bet'

AND t.transaction\_type = 'debit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

-- Apply cohort date bounds

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

-- Currency filter using same resolution as daily/monthly

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY t.player\_id

),

/\* Step 2: Calculate cohort sizes \*/

cohort\_sizes AS (

SELECT

first\_cash\_bet\_month as cohort\_month,

COUNT(DISTINCT player\_id) as cohort\_size

FROM first\_cash\_bets

GROUP BY first\_cash\_bet\_month

),

/\* Step 3: Track activity for each cohort \*/

cohort\_activity AS (

SELECT

fcb.first\_cash\_bet\_month as cohort\_month,

DATE\_TRUNC('month', t.created\_at) as activity\_month,

COUNT(DISTINCT t.player\_id) as active\_players

FROM first\_cash\_bets fcb

INNER JOIN transactions t ON fcb.player\_id = t.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'game\_bet'

AND t.transaction\_type = 'debit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

AND t.created\_at >= fcb.first\_cash\_bet\_date

-- Apply same currency filter

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY fcb.first\_cash\_bet\_month, DATE\_TRUNC('month', t.created\_at)

),

/\* Step 4: Calculate retention \*/

cohort\_retention AS (

SELECT

ca.cohort\_month,

ca.activity\_month,

ca.active\_players,

cs.cohort\_size,

EXTRACT(YEAR FROM AGE(ca.activity\_month, ca.cohort\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(ca.activity\_month, ca.cohort\_month)) as months\_since\_first\_bet

FROM cohort\_activity ca

INNER JOIN cohort\_sizes cs ON ca.cohort\_month = cs.cohort\_month

WHERE EXTRACT(YEAR FROM AGE(ca.activity\_month, ca.cohort\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(ca.activity\_month, ca.cohort\_month)) <= 12

)

/\* Step 5: Pivot showing NUMERIC PERCENTAGES (no % symbol for conditional formatting) \*/

SELECT

TO\_CHAR(cohort\_month, 'Month YYYY') as "FIRST CASH BET MONTH",

100::numeric as "Month 0", -- Always 100 for Month 0, as numeric type

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 1 THEN active\_players::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 1",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 2 THEN active\_players::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 2",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 3 THEN active\_players::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 3",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 4 THEN active\_players::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 4",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 5 THEN active\_players::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 5",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 6 THEN active\_players::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 6",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 7 THEN active\_players::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 7",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 8 THEN active\_players::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 8",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 9 THEN active\_players::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 9",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 10 THEN active\_players::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 10",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 11 THEN active\_players::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 11",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 12 THEN active\_players::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 12"

FROM cohort\_retention

GROUP BY cohort\_month, cohort\_size

ORDER BY cohort\_month;

CASH BET AMOUNT COHORT

/\* ============================================

CASH BET AMOUNTS COHORT - ALIGNED WITH DAILY/MONTHLY FILTERS

Shows total bet amounts by cohort over time

============================================ \*/

WITH

/\* --- Optional date inputs for cohort window --- \*/

start\_input AS (

SELECT NULL::date AS start\_date WHERE FALSE

[[ UNION ALL SELECT {{start\_date}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_date WHERE FALSE

[[ UNION ALL SELECT {{end\_date}}::date ]]

),

/\* --- Normalize cohort window (default: last 12 months to current month) --- \*/

bounds AS (

SELECT

COALESCE((SELECT start\_date FROM start\_input),

DATE\_TRUNC('month', CURRENT\_DATE - INTERVAL '12 months')) AS start\_date,

COALESCE((SELECT end\_date FROM end\_input),

DATE\_TRUNC('month', CURRENT\_DATE)) AS end\_date

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND CONCAT(players.os, ' / ', players.browser) = {{registration\_launcher}} ]]

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* Step 1: Identify first cash bets with currency filter \*/

first\_cash\_bets AS (

SELECT

t.player\_id,

DATE\_TRUNC('month', MIN(t.created\_at)) as first\_cash\_bet\_month,

MIN(t.created\_at) as first\_cash\_bet\_date

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'game\_bet'

AND t.transaction\_type = 'debit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

-- Apply cohort date bounds

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

-- Currency filter using same resolution as daily/monthly

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY t.player\_id

),

/\* Step 2: Calculate cohort sizes \*/

cohort\_sizes AS (

SELECT

first\_cash\_bet\_month as cohort\_month,

COUNT(DISTINCT player\_id) as cohort\_size

FROM first\_cash\_bets

GROUP BY first\_cash\_bet\_month

),

/\* Step 3: Calculate TOTAL BET AMOUNTS for each cohort across months \*/

cohort\_bet\_amounts AS (

SELECT

fcb.first\_cash\_bet\_month as cohort\_month,

DATE\_TRUNC('month', t.created\_at) as activity\_month,

SUM(ABS(t.amount)) as total\_amount\_wagered, -- Using ABS for debits

COUNT(t.id) as total\_bets,

COUNT(DISTINCT t.player\_id) as active\_players

FROM first\_cash\_bets fcb

INNER JOIN transactions t ON fcb.player\_id = t.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'game\_bet'

AND t.transaction\_type = 'debit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

AND t.created\_at >= fcb.first\_cash\_bet\_date

-- Apply same currency filter

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY fcb.first\_cash\_bet\_month, DATE\_TRUNC('month', t.created\_at)

),

/\* Step 4: Calculate months since first bet \*/

cohort\_retention AS (

SELECT

cba.cohort\_month,

cba.activity\_month,

cba.total\_amount\_wagered,

cba.total\_bets,

cba.active\_players,

cs.cohort\_size,

EXTRACT(YEAR FROM AGE(cba.activity\_month, cba.cohort\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(cba.activity\_month, cba.cohort\_month)) as months\_since\_first\_bet

FROM cohort\_bet\_amounts cba

INNER JOIN cohort\_sizes cs ON cba.cohort\_month = cs.cohort\_month

WHERE EXTRACT(YEAR FROM AGE(cba.activity\_month, cba.cohort\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(cba.activity\_month, cba.cohort\_month)) <= 12

)

/\* Step 5: Pivot showing NUMERIC AMOUNTS ONLY \*/

SELECT

TO\_CHAR(cohort\_month, 'Month YYYY') as "FIRST CASH BET MONTH",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 0 THEN total\_amount\_wagered END), 2) as "Month 0",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 1 THEN total\_amount\_wagered END), 2) as "Month 1",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 2 THEN total\_amount\_wagered END), 2) as "Month 2",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 3 THEN total\_amount\_wagered END), 2) as "Month 3",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 4 THEN total\_amount\_wagered END), 2) as "Month 4",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 5 THEN total\_amount\_wagered END), 2) as "Month 5",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 6 THEN total\_amount\_wagered END), 2) as "Month 6",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 7 THEN total\_amount\_wagered END), 2) as "Month 7",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 8 THEN total\_amount\_wagered END), 2) as "Month 8",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 9 THEN total\_amount\_wagered END), 2) as "Month 9",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 10 THEN total\_amount\_wagered END), 2) as "Month 10",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 11 THEN total\_amount\_wagered END), 2) as "Month 11",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 12 THEN total\_amount\_wagered END), 2) as "Month 12"

FROM cohort\_retention

GROUP BY cohort\_month

ORDER BY cohort\_month;

CASH BET AMOUNTS COHORT (%)

/\* ============================================

CASH BET AMOUNTS COHORT (%) - NUMERIC VERSION FOR CONDITIONAL FORMATTING

Shows bet amounts as percentage of Month 0 (as numbers)

============================================ \*/

WITH

/\* --- Optional date inputs for cohort window --- \*/

start\_input AS (

SELECT NULL::date AS start\_date WHERE FALSE

[[ UNION ALL SELECT {{start\_date}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_date WHERE FALSE

[[ UNION ALL SELECT {{end\_date}}::date ]]

),

/\* --- Normalize cohort window (default: last 12 months to current month) --- \*/

bounds AS (

SELECT

COALESCE((SELECT start\_date FROM start\_input),

DATE\_TRUNC('month', CURRENT\_DATE - INTERVAL '12 months')) AS start\_date,

COALESCE((SELECT end\_date FROM end\_input),

DATE\_TRUNC('month', CURRENT\_DATE)) AS end\_date

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND CONCAT(players.os, ' / ', players.browser) = {{registration\_launcher}} ]]

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* Step 1: Identify first cash bets with currency filter \*/

first\_cash\_bets AS (

SELECT

t.player\_id,

DATE\_TRUNC('month', MIN(t.created\_at)) as first\_cash\_bet\_month,

MIN(t.created\_at) as first\_cash\_bet\_date

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'game\_bet'

AND t.transaction\_type = 'debit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

-- Apply cohort date bounds

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

-- Currency filter using same resolution as daily/monthly

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY t.player\_id

),

/\* Step 2: Calculate TOTAL BET AMOUNTS for each cohort across months \*/

cohort\_bet\_amounts AS (

SELECT

fcb.first\_cash\_bet\_month as cohort\_month,

DATE\_TRUNC('month', t.created\_at) as activity\_month,

SUM(ABS(t.amount)) as total\_amount\_wagered,

EXTRACT(YEAR FROM AGE(DATE\_TRUNC('month', t.created\_at), fcb.first\_cash\_bet\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(DATE\_TRUNC('month', t.created\_at), fcb.first\_cash\_bet\_month)) as months\_since\_first\_bet

FROM first\_cash\_bets fcb

INNER JOIN transactions t ON fcb.player\_id = t.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'game\_bet'

AND t.transaction\_type = 'debit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

AND t.created\_at >= fcb.first\_cash\_bet\_date

-- Apply same currency filter

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY fcb.first\_cash\_bet\_month, DATE\_TRUNC('month', t.created\_at)

),

/\* Step 3: Get Month 0 amounts for baseline \*/

month\_0\_amounts AS (

SELECT

cohort\_month,

total\_amount\_wagered as month\_0\_amount

FROM cohort\_bet\_amounts

WHERE months\_since\_first\_bet = 0

)

/\* Step 4: Pivot showing NUMERIC PERCENTAGES (no % symbol) \*/

SELECT

TO\_CHAR(cba.cohort\_month, 'Month YYYY') as "FIRST CASH BET MONTH",

100::numeric as "Month 0", -- Always 100 for Month 0

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 1 THEN total\_amount\_wagered / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 1",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 2 THEN total\_amount\_wagered / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 2",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 3 THEN total\_amount\_wagered / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 3",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 4 THEN total\_amount\_wagered / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 4",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 5 THEN total\_amount\_wagered / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 5",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 6 THEN total\_amount\_wagered / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 6",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 7 THEN total\_amount\_wagered / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 7",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 8 THEN total\_amount\_wagered / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 8",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 9 THEN total\_amount\_wagered / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 9",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 10 THEN total\_amount\_wagered / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 10",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 11 THEN total\_amount\_wagered / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 11",

ROUND(MAX(CASE WHEN months\_since\_first\_bet = 12 THEN total\_amount\_wagered / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 12"

FROM cohort\_bet\_amounts cba

JOIN month\_0\_amounts m0 ON cba.cohort\_month = m0.cohort\_month

WHERE months\_since\_first\_bet <= 12

GROUP BY cba.cohort\_month

ORDER BY cba.cohort\_month;

DEPOSITORS COHORT

/\* ============================================

DEPOSITORS COHORT - ALIGNED WITH DAILY/MONTHLY FILTERS

Shows retention of depositors over time

============================================ \*/

WITH

/\* --- Optional date inputs for cohort window --- \*/

start\_input AS (

SELECT NULL::date AS start\_date WHERE FALSE

[[ UNION ALL SELECT {{start\_date}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_date WHERE FALSE

[[ UNION ALL SELECT {{end\_date}}::date ]]

),

/\* --- Normalize cohort window (default: last 12 months to current month) --- \*/

bounds AS (

SELECT

COALESCE((SELECT start\_date FROM start\_input),

DATE\_TRUNC('month', CURRENT\_DATE - INTERVAL '12 months')) AS start\_date,

COALESCE((SELECT end\_date FROM end\_input),

DATE\_TRUNC('month', CURRENT\_DATE)) AS end\_date

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND CONCAT(players.os, ' / ', players.browser) = {{registration\_launcher}} ]]

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* Step 1: Identify first deposits with currency filter \*/

first\_deposits AS (

SELECT

t.player\_id,

DATE\_TRUNC('month', MIN(t.created\_at)) as first\_deposit\_month,

MIN(t.created\_at) as first\_deposit\_date

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

-- Apply cohort date bounds

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

-- Currency filter using same resolution as daily/monthly

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY t.player\_id

),

/\* Step 2: Calculate cohort sizes \*/

cohort\_sizes AS (

SELECT

first\_deposit\_month as cohort\_month,

COUNT(DISTINCT player\_id) as cohort\_size

FROM first\_deposits

GROUP BY first\_deposit\_month

),

/\* Step 3: Track deposit activity for each cohort \*/

cohort\_activity AS (

SELECT

fd.first\_deposit\_month as cohort\_month,

DATE\_TRUNC('month', t.created\_at) as activity\_month,

COUNT(DISTINCT t.player\_id) as active\_depositors

FROM first\_deposits fd

INNER JOIN transactions t ON fd.player\_id = t.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

AND t.created\_at >= fd.first\_deposit\_date

-- Apply same currency filter

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY fd.first\_deposit\_month, DATE\_TRUNC('month', t.created\_at)

),

/\* Step 4: Calculate retention \*/

cohort\_retention AS (

SELECT

ca.cohort\_month,

ca.activity\_month,

ca.active\_depositors,

cs.cohort\_size,

EXTRACT(YEAR FROM AGE(ca.activity\_month, ca.cohort\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(ca.activity\_month, ca.cohort\_month)) as months\_since\_first\_deposit

FROM cohort\_activity ca

INNER JOIN cohort\_sizes cs ON ca.cohort\_month = cs.cohort\_month

WHERE EXTRACT(YEAR FROM AGE(ca.activity\_month, ca.cohort\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(ca.activity\_month, ca.cohort\_month)) <= 12

)

/\* Step 5: Pivot for final output \*/

SELECT

TO\_CHAR(cohort\_month, 'Month YYYY') as "FIRST DEPOSIT MONTH",

MAX(CASE WHEN months\_since\_first\_deposit = 0 THEN active\_depositors END) as "Month 0",

MAX(CASE WHEN months\_since\_first\_deposit = 1 THEN active\_depositors END) as "Month 1",

MAX(CASE WHEN months\_since\_first\_deposit = 2 THEN active\_depositors END) as "Month 2",

MAX(CASE WHEN months\_since\_first\_deposit = 3 THEN active\_depositors END) as "Month 3",

MAX(CASE WHEN months\_since\_first\_deposit = 4 THEN active\_depositors END) as "Month 4",

MAX(CASE WHEN months\_since\_first\_deposit = 5 THEN active\_depositors END) as "Month 5",

MAX(CASE WHEN months\_since\_first\_deposit = 6 THEN active\_depositors END) as "Month 6",

MAX(CASE WHEN months\_since\_first\_deposit = 7 THEN active\_depositors END) as "Month 7",

MAX(CASE WHEN months\_since\_first\_deposit = 8 THEN active\_depositors END) as "Month 8",

MAX(CASE WHEN months\_since\_first\_deposit = 9 THEN active\_depositors END) as "Month 9",

MAX(CASE WHEN months\_since\_first\_deposit = 10 THEN active\_depositors END) as "Month 10",

MAX(CASE WHEN months\_since\_first\_deposit = 11 THEN active\_depositors END) as "Month 11",

MAX(CASE WHEN months\_since\_first\_deposit = 12 THEN active\_depositors END) as "Month 12"

FROM cohort\_retention

GROUP BY cohort\_month

ORDER BY cohort\_month;

DEPOSITORS COHORT (%)

/\* ============================================

DEPOSITORS COHORT (%) - NUMERIC VERSION FOR CONDITIONAL FORMATTING

Shows retention percentages as numbers (Metabase will format as %)

============================================ \*/

WITH

/\* --- Optional date inputs for cohort window --- \*/

start\_input AS (

SELECT NULL::date AS start\_date WHERE FALSE

[[ UNION ALL SELECT {{start\_date}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_date WHERE FALSE

[[ UNION ALL SELECT {{end\_date}}::date ]]

),

/\* --- Normalize cohort window (default: last 12 months to current month) --- \*/

bounds AS (

SELECT

COALESCE((SELECT start\_date FROM start\_input),

DATE\_TRUNC('month', CURRENT\_DATE - INTERVAL '12 months')) AS start\_date,

COALESCE((SELECT end\_date FROM end\_input),

DATE\_TRUNC('month', CURRENT\_DATE)) AS end\_date

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND CONCAT(players.os, ' / ', players.browser) = {{registration\_launcher}} ]]

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* Step 1: Identify first deposits with currency filter \*/

first\_deposits AS (

SELECT

t.player\_id,

DATE\_TRUNC('month', MIN(t.created\_at)) as first\_deposit\_month,

MIN(t.created\_at) as first\_deposit\_date

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

-- Apply cohort date bounds

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

-- Currency filter using same resolution as daily/monthly

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY t.player\_id

),

/\* Step 2: Calculate cohort sizes \*/

cohort\_sizes AS (

SELECT

first\_deposit\_month as cohort\_month,

COUNT(DISTINCT player\_id) as cohort\_size

FROM first\_deposits

GROUP BY first\_deposit\_month

),

/\* Step 3: Track deposit activity for each cohort \*/

cohort\_activity AS (

SELECT

fd.first\_deposit\_month as cohort\_month,

DATE\_TRUNC('month', t.created\_at) as activity\_month,

COUNT(DISTINCT t.player\_id) as active\_depositors

FROM first\_deposits fd

INNER JOIN transactions t ON fd.player\_id = t.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

AND t.created\_at >= fd.first\_deposit\_date

-- Apply same currency filter

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY fd.first\_deposit\_month, DATE\_TRUNC('month', t.created\_at)

),

/\* Step 4: Calculate retention \*/

cohort\_retention AS (

SELECT

ca.cohort\_month,

ca.activity\_month,

ca.active\_depositors,

cs.cohort\_size,

EXTRACT(YEAR FROM AGE(ca.activity\_month, ca.cohort\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(ca.activity\_month, ca.cohort\_month)) as months\_since\_first\_deposit

FROM cohort\_activity ca

INNER JOIN cohort\_sizes cs ON ca.cohort\_month = cs.cohort\_month

WHERE EXTRACT(YEAR FROM AGE(ca.activity\_month, ca.cohort\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(ca.activity\_month, ca.cohort\_month)) <= 12

)

/\* Step 5: Pivot showing NUMERIC PERCENTAGES (no % symbol) \*/

SELECT

TO\_CHAR(cohort\_month, 'Month YYYY') as "FIRST DEPOSIT MONTH",

100::numeric as "Month 0", -- Always 100 for Month 0

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 1 THEN active\_depositors::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 1",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 2 THEN active\_depositors::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 2",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 3 THEN active\_depositors::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 3",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 4 THEN active\_depositors::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 4",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 5 THEN active\_depositors::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 5",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 6 THEN active\_depositors::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 6",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 7 THEN active\_depositors::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 7",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 8 THEN active\_depositors::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 8",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 9 THEN active\_depositors::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 9",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 10 THEN active\_depositors::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 10",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 11 THEN active\_depositors::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 11",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 12 THEN active\_depositors::numeric / NULLIF(cohort\_size, 0) \* 100 END), 1) as "Month 12"

FROM cohort\_retention

GROUP BY cohort\_month, cohort\_size

ORDER BY cohort\_month;

DEPOSIT AMOUNTS COHORT

/\* ============================================

DEPOSIT AMOUNTS COHORT - ALIGNED WITH DAILY/MONTHLY FILTERS

Shows total deposit amounts by cohort over time

============================================ \*/

WITH

/\* --- Optional date inputs for cohort window --- \*/

start\_input AS (

SELECT NULL::date AS start\_date WHERE FALSE

[[ UNION ALL SELECT {{start\_date}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_date WHERE FALSE

[[ UNION ALL SELECT {{end\_date}}::date ]]

),

/\* --- Normalize cohort window (default: last 12 months to current month) --- \*/

bounds AS (

SELECT

COALESCE((SELECT start\_date FROM start\_input),

DATE\_TRUNC('month', CURRENT\_DATE - INTERVAL '12 months')) AS start\_date,

COALESCE((SELECT end\_date FROM end\_input),

DATE\_TRUNC('month', CURRENT\_DATE)) AS end\_date

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND CONCAT(players.os, ' / ', players.browser) = {{registration\_launcher}} ]]

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* Step 1: Identify first deposits with currency filter \*/

first\_deposits AS (

SELECT

t.player\_id,

DATE\_TRUNC('month', MIN(t.created\_at)) as first\_deposit\_month,

MIN(t.created\_at) as first\_deposit\_date

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

-- Apply cohort date bounds

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

-- Currency filter using same resolution as daily/monthly

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY t.player\_id

),

/\* Step 2: Calculate cohort sizes (for reference) \*/

cohort\_sizes AS (

SELECT

first\_deposit\_month as cohort\_month,

COUNT(DISTINCT player\_id) as cohort\_size

FROM first\_deposits

GROUP BY first\_deposit\_month

),

/\* Step 3: Calculate TOTAL DEPOSIT AMOUNTS for each cohort across months \*/

cohort\_deposit\_amounts AS (

SELECT

fd.first\_deposit\_month as cohort\_month,

DATE\_TRUNC('month', t.created\_at) as activity\_month,

SUM(t.amount) as total\_deposit\_amount,

COUNT(t.id) as total\_deposits,

COUNT(DISTINCT t.player\_id) as unique\_depositors

FROM first\_deposits fd

INNER JOIN transactions t ON fd.player\_id = t.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

AND t.created\_at >= fd.first\_deposit\_date

-- Apply same currency filter

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY fd.first\_deposit\_month, DATE\_TRUNC('month', t.created\_at)

),

/\* Step 4: Calculate months since first deposit \*/

cohort\_retention AS (

SELECT

cda.cohort\_month,

cda.activity\_month,

cda.total\_deposit\_amount,

cda.total\_deposits,

cda.unique\_depositors,

cs.cohort\_size,

EXTRACT(YEAR FROM AGE(cda.activity\_month, cda.cohort\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(cda.activity\_month, cda.cohort\_month)) as months\_since\_first\_deposit

FROM cohort\_deposit\_amounts cda

INNER JOIN cohort\_sizes cs ON cda.cohort\_month = cs.cohort\_month

WHERE EXTRACT(YEAR FROM AGE(cda.activity\_month, cda.cohort\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(cda.activity\_month, cda.cohort\_month)) <= 12

)

/\* Step 5: Pivot showing NUMERIC AMOUNTS ONLY \*/

SELECT

TO\_CHAR(cohort\_month, 'Month YYYY') as "FIRST DEPOSIT MONTH",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 0 THEN total\_deposit\_amount END), 2) as "Month 0",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 1 THEN total\_deposit\_amount END), 2) as "Month 1",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 2 THEN total\_deposit\_amount END), 2) as "Month 2",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 3 THEN total\_deposit\_amount END), 2) as "Month 3",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 4 THEN total\_deposit\_amount END), 2) as "Month 4",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 5 THEN total\_deposit\_amount END), 2) as "Month 5",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 6 THEN total\_deposit\_amount END), 2) as "Month 6",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 7 THEN total\_deposit\_amount END), 2) as "Month 7",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 8 THEN total\_deposit\_amount END), 2) as "Month 8",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 9 THEN total\_deposit\_amount END), 2) as "Month 9",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 10 THEN total\_deposit\_amount END), 2) as "Month 10",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 11 THEN total\_deposit\_amount END), 2) as "Month 11",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 12 THEN total\_deposit\_amount END), 2) as "Month 12"

FROM cohort\_retention

GROUP BY cohort\_month

ORDER BY cohort\_month;

DEPOSIT AMOUNTS COHORT (%)

/\* ============================================

DEPOSIT AMOUNTS COHORT (%) - NUMERIC VERSION FOR CONDITIONAL FORMATTING

Shows deposit amounts as percentage of Month 0 (as numbers)

============================================ \*/

WITH

/\* --- Optional date inputs for cohort window --- \*/

start\_input AS (

SELECT NULL::date AS start\_date WHERE FALSE

[[ UNION ALL SELECT {{start\_date}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_date WHERE FALSE

[[ UNION ALL SELECT {{end\_date}}::date ]]

),

/\* --- Normalize cohort window (default: last 12 months to current month) --- \*/

bounds AS (

SELECT

COALESCE((SELECT start\_date FROM start\_input),

DATE\_TRUNC('month', CURRENT\_DATE - INTERVAL '12 months')) AS start\_date,

COALESCE((SELECT end\_date FROM end\_input),

DATE\_TRUNC('month', CURRENT\_DATE)) AS end\_date

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND CONCAT(players.os, ' / ', players.browser) = {{registration\_launcher}} ]]

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* Step 1: Identify first deposits with currency filter \*/

first\_deposits AS (

SELECT

t.player\_id,

DATE\_TRUNC('month', MIN(t.created\_at)) as first\_deposit\_month,

MIN(t.created\_at) as first\_deposit\_date

FROM transactions t

INNER JOIN filtered\_players fp ON t.player\_id = fp.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

-- Apply cohort date bounds

AND t.created\_at >= (SELECT start\_date FROM bounds)

AND t.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

-- Currency filter using same resolution as daily/monthly

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY t.player\_id

),

/\* Step 2: Calculate TOTAL DEPOSIT AMOUNTS for each cohort across months \*/

cohort\_deposit\_amounts AS (

SELECT

fd.first\_deposit\_month as cohort\_month,

DATE\_TRUNC('month', t.created\_at) as activity\_month,

SUM(t.amount) as total\_deposit\_amount,

EXTRACT(YEAR FROM AGE(DATE\_TRUNC('month', t.created\_at), fd.first\_deposit\_month)) \* 12 +

EXTRACT(MONTH FROM AGE(DATE\_TRUNC('month', t.created\_at), fd.first\_deposit\_month)) as months\_since\_first\_deposit

FROM first\_deposits fd

INNER JOIN transactions t ON fd.player\_id = t.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

AND t.created\_at >= fd.first\_deposit\_date

-- Apply same currency filter

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY fd.first\_deposit\_month, DATE\_TRUNC('month', t.created\_at)

),

/\* Step 3: Get Month 0 amounts for baseline \*/

month\_0\_amounts AS (

SELECT

cohort\_month,

total\_deposit\_amount as month\_0\_amount

FROM cohort\_deposit\_amounts

WHERE months\_since\_first\_deposit = 0

)

/\* Step 4: Pivot showing NUMERIC PERCENTAGES (no % symbol) \*/

SELECT

TO\_CHAR(cda.cohort\_month, 'Month YYYY') as "FIRST DEPOSIT MONTH",

100::numeric as "Month 0", -- Always 100 for Month 0

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 1 THEN total\_deposit\_amount / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 1",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 2 THEN total\_deposit\_amount / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 2",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 3 THEN total\_deposit\_amount / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 3",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 4 THEN total\_deposit\_amount / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 4",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 5 THEN total\_deposit\_amount / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 5",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 6 THEN total\_deposit\_amount / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 6",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 7 THEN total\_deposit\_amount / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 7",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 8 THEN total\_deposit\_amount / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 8",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 9 THEN total\_deposit\_amount / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 9",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 10 THEN total\_deposit\_amount / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 10",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 11 THEN total\_deposit\_amount / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 11",

ROUND(MAX(CASE WHEN months\_since\_first\_deposit = 12 THEN total\_deposit\_amount / NULLIF(m0.month\_0\_amount, 0) \* 100 END), 1) as "Month 12"

FROM cohort\_deposit\_amounts cda

JOIN month\_0\_amounts m0 ON cda.cohort\_month = m0.cohort\_month

WHERE months\_since\_first\_deposit <= 12

GROUP BY cda.cohort\_month

ORDER BY cda.cohort\_month;

NEW DEPOSITORS COHORT

/\* ============================================

NEW DEPOSITORS COHORT - DYNAMIC DATE RANGE

Shows unique count of new depositors reaching each deposit milestone

One row per month within user-selected date range

New Depositor = first\_ever\_deposit in that specific month

FILTERS: Brand, Country, Traffic Source, Affiliate, Registration Launcher, Test Account, Currency

PARAMETERS: start\_month, end\_month (DATE), currency\_filter, brand, country, traffic\_source, affiliate\_id, affiliate\_name, registration\_launcher, is\_test\_account

============================================ \*/

WITH

/\* --- Month range inputs (user-selectable) --- \*/

start\_input AS (

SELECT NULL::date AS start\_month WHERE FALSE

[[ UNION ALL SELECT {{start\_month}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_month WHERE FALSE

[[ UNION ALL SELECT {{end\_month}}::date ]]

),

/\* --- Normalize analysis window (defaults to last 12 months if not specified) --- \*/

analysis\_period AS (

SELECT

COALESCE(

(SELECT start\_month FROM start\_input),

(SELECT DATE\_TRUNC('month', MAX(created\_at) - INTERVAL '11 months')::date FROM transactions)

) AS month\_start,

COALESCE(

(SELECT end\_month FROM end\_input),

(SELECT DATE\_TRUNC('month', MAX(created\_at))::date FROM transactions)

) AS month\_end

),

/\* --- Get all months in the analysis period --- \*/

available\_months AS (

SELECT DATE\_TRUNC('month', d)::date AS month\_start

FROM GENERATE\_SERIES(

(SELECT month\_start FROM analysis\_period),

(SELECT month\_end FROM analysis\_period),

INTERVAL '1 month'

) AS d

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND players.os = {{registration\_launcher}} ]] -- Simplified to OS-only (matching daily report)

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* Step 1: Calculate lifetime deposit count and first deposit date for ALL filtered players \*/

player\_lifetime\_deposits AS (

SELECT

transactions.player\_id,

COUNT(\*) as lifetime\_deposit\_count,

MIN(transactions.created\_at) as first\_deposit\_date,

MAX(transactions.created\_at) as last\_deposit\_date

FROM transactions

INNER JOIN filtered\_players ON transactions.player\_id = filtered\_players.player\_id

JOIN players ON players.id = transactions.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE transactions.transaction\_category = 'deposit'

AND transactions.transaction\_type = 'credit'

AND transactions.balance\_type = 'withdrawable'

AND transactions.status = 'completed'

-- Currency filter using standard hierarchy

[[ AND UPPER(COALESCE(

transactions.metadata->>'currency',

transactions.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY transactions.player\_id

),

/\* Step 2: For each month in analysis period, identify NEW DEPOSITORS (first deposit in that month) \*/

monthly\_cohorts AS (

SELECT

DATE\_TRUNC('month', pld.first\_deposit\_date)::date AS cohort\_month,

pld.player\_id,

pld.lifetime\_deposit\_count

FROM player\_lifetime\_deposits pld

INNER JOIN available\_months am ON DATE\_TRUNC('month', pld.first\_deposit\_date)::date = am.month\_start

),

/\* Step 3: Calculate bucket distribution for each month \*/

monthly\_bucket\_counts AS (

SELECT

cohort\_month,

-- Individual buckets 1-10

COUNT(CASE WHEN lifetime\_deposit\_count >= 1 THEN 1 END) as bucket\_1,

COUNT(CASE WHEN lifetime\_deposit\_count >= 2 THEN 1 END) as bucket\_2,

COUNT(CASE WHEN lifetime\_deposit\_count >= 3 THEN 1 END) as bucket\_3,

COUNT(CASE WHEN lifetime\_deposit\_count >= 4 THEN 1 END) as bucket\_4,

COUNT(CASE WHEN lifetime\_deposit\_count >= 5 THEN 1 END) as bucket\_5,

COUNT(CASE WHEN lifetime\_deposit\_count >= 6 THEN 1 END) as bucket\_6,

COUNT(CASE WHEN lifetime\_deposit\_count >= 7 THEN 1 END) as bucket\_7,

COUNT(CASE WHEN lifetime\_deposit\_count >= 8 THEN 1 END) as bucket\_8,

COUNT(CASE WHEN lifetime\_deposit\_count >= 9 THEN 1 END) as bucket\_9,

COUNT(CASE WHEN lifetime\_deposit\_count >= 10 THEN 1 END) as bucket\_10,

-- Grouped buckets

COUNT(CASE WHEN lifetime\_deposit\_count >= 11 AND lifetime\_deposit\_count <= 15 THEN 1 END) as bucket\_11\_15,

COUNT(CASE WHEN lifetime\_deposit\_count >= 16 AND lifetime\_deposit\_count <= 20 THEN 1 END) as bucket\_16\_20,

COUNT(CASE WHEN lifetime\_deposit\_count > 20 THEN 1 END) as bucket\_over\_20,

-- Total cohort size

COUNT(\*) as total\_cohort

FROM monthly\_cohorts

GROUP BY cohort\_month

)

/\* Final Output - One row per month within selected date range \*/

SELECT

TO\_CHAR(cohort\_month, 'FMMonth YYYY') as "FTD\_Month",

bucket\_1 as "1 time",

bucket\_2 as "2 times",

bucket\_3 as "3 times",

bucket\_4 as "4 times",

bucket\_5 as "5 times",

bucket\_6 as "6 times",

bucket\_7 as "7 times",

bucket\_8 as "8 times",

bucket\_9 as "9 times",

bucket\_10 as "10 times",

bucket\_11\_15 as "11-15 times",

bucket\_16\_20 as "16-20 times",

bucket\_over\_20 as ">20 times",

total\_cohort as "Total\_FTD"

FROM monthly\_bucket\_counts

ORDER BY cohort\_month DESC;

NEW DEPOSITORS COHORT - PERCENTAGE

/\* ============================================

NEW DEPOSITORS COHORT - PERCENTAGE DISTRIBUTION

Shows percentage of new depositors reaching each deposit milestone

One row per month within user-selected date range

New Depositor = first\_ever\_deposit in that specific month

FILTERS: Brand, Country, Traffic Source, Affiliate, Registration Launcher, Test Account, Currency

PARAMETERS: start\_month, end\_month (DATE), currency\_filter, brand, country, traffic\_source, affiliate\_id, affiliate\_name, registration\_launcher, is\_test\_account

============================================ \*/

WITH

/\* --- Month range inputs (user-selectable) --- \*/

start\_input AS (

SELECT NULL::date AS start\_month WHERE FALSE

[[ UNION ALL SELECT {{start\_month}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_month WHERE FALSE

[[ UNION ALL SELECT {{end\_month}}::date ]]

),

/\* --- Normalize analysis window (defaults to last 12 months if not specified) --- \*/

analysis\_period AS (

SELECT

COALESCE(

(SELECT start\_month FROM start\_input),

(SELECT DATE\_TRUNC('month', MAX(created\_at) - INTERVAL '11 months')::date FROM transactions)

) AS month\_start,

COALESCE(

(SELECT end\_month FROM end\_input),

(SELECT DATE\_TRUNC('month', MAX(created\_at))::date FROM transactions)

) AS month\_end

),

/\* --- Get all months in the analysis period --- \*/

available\_months AS (

SELECT DATE\_TRUNC('month', d)::date AS month\_start

FROM GENERATE\_SERIES(

(SELECT month\_start FROM analysis\_period),

(SELECT month\_end FROM analysis\_period),

INTERVAL '1 month'

) AS d

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND players.os = {{registration\_launcher}} ]] -- Simplified to OS-only (matching daily report)

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* Step 1: Calculate lifetime deposit count and first deposit date for ALL filtered players \*/

player\_lifetime\_deposits AS (

SELECT

transactions.player\_id,

COUNT(\*) as lifetime\_deposit\_count,

MIN(transactions.created\_at) as first\_deposit\_date,

MAX(transactions.created\_at) as last\_deposit\_date

FROM transactions

INNER JOIN filtered\_players ON transactions.player\_id = filtered\_players.player\_id

JOIN players ON players.id = transactions.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE transactions.transaction\_category = 'deposit'

AND transactions.transaction\_type = 'credit'

AND transactions.balance\_type = 'withdrawable'

AND transactions.status = 'completed'

-- Currency filter using standard hierarchy

[[ AND UPPER(COALESCE(

transactions.metadata->>'currency',

transactions.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY transactions.player\_id

),

/\* Step 2: For each month in analysis period, identify NEW DEPOSITORS (first deposit in that month) \*/

monthly\_cohorts AS (

SELECT

DATE\_TRUNC('month', pld.first\_deposit\_date)::date AS cohort\_month,

pld.player\_id,

pld.lifetime\_deposit\_count

FROM player\_lifetime\_deposits pld

INNER JOIN available\_months am ON DATE\_TRUNC('month', pld.first\_deposit\_date)::date = am.month\_start

),

/\* Step 3: Calculate bucket counts and percentages for each month \*/

monthly\_bucket\_counts AS (

SELECT

cohort\_month,

-- Absolute counts

COUNT(\*) as total\_cohort,

COUNT(CASE WHEN lifetime\_deposit\_count >= 1 THEN 1 END) as bucket\_1\_count,

COUNT(CASE WHEN lifetime\_deposit\_count >= 2 THEN 1 END) as bucket\_2\_count,

COUNT(CASE WHEN lifetime\_deposit\_count >= 3 THEN 1 END) as bucket\_3\_count,

COUNT(CASE WHEN lifetime\_deposit\_count >= 4 THEN 1 END) as bucket\_4\_count,

COUNT(CASE WHEN lifetime\_deposit\_count >= 5 THEN 1 END) as bucket\_5\_count,

COUNT(CASE WHEN lifetime\_deposit\_count >= 6 THEN 1 END) as bucket\_6\_count,

COUNT(CASE WHEN lifetime\_deposit\_count >= 7 THEN 1 END) as bucket\_7\_count,

COUNT(CASE WHEN lifetime\_deposit\_count >= 8 THEN 1 END) as bucket\_8\_count,

COUNT(CASE WHEN lifetime\_deposit\_count >= 9 THEN 1 END) as bucket\_9\_count,

COUNT(CASE WHEN lifetime\_deposit\_count >= 10 THEN 1 END) as bucket\_10\_count,

COUNT(CASE WHEN lifetime\_deposit\_count >= 11 AND lifetime\_deposit\_count <= 15 THEN 1 END) as bucket\_11\_15\_count,

COUNT(CASE WHEN lifetime\_deposit\_count >= 16 AND lifetime\_deposit\_count <= 20 THEN 1 END) as bucket\_16\_20\_count,

COUNT(CASE WHEN lifetime\_deposit\_count > 20 THEN 1 END) as bucket\_over\_20\_count

FROM monthly\_cohorts

GROUP BY cohort\_month

)

/\* Final Output - One row per month with percentages \*/

SELECT

TO\_CHAR(cohort\_month, 'FMMonth YYYY') as "FTD\_Month",

ROUND(bucket\_1\_count::numeric / NULLIF(total\_cohort, 0) \* 100, 1) as "1 time",

ROUND(bucket\_2\_count::numeric / NULLIF(total\_cohort, 0) \* 100, 1) as "2 times",

ROUND(bucket\_3\_count::numeric / NULLIF(total\_cohort, 0) \* 100, 1) as "3 times",

ROUND(bucket\_4\_count::numeric / NULLIF(total\_cohort, 0) \* 100, 1) as "4 times",

ROUND(bucket\_5\_count::numeric / NULLIF(total\_cohort, 0) \* 100, 1) as "5 times",

ROUND(bucket\_6\_count::numeric / NULLIF(total\_cohort, 0) \* 100, 1) as "6 times",

ROUND(bucket\_7\_count::numeric / NULLIF(total\_cohort, 0) \* 100, 1) as "7 times",

ROUND(bucket\_8\_count::numeric / NULLIF(total\_cohort, 0) \* 100, 1) as "8 times",

ROUND(bucket\_9\_count::numeric / NULLIF(total\_cohort, 0) \* 100, 1) as "9 times",

ROUND(bucket\_10\_count::numeric / NULLIF(total\_cohort, 0) \* 100, 1) as "10 times",

ROUND(bucket\_11\_15\_count::numeric / NULLIF(total\_cohort, 0) \* 100, 1) as "11-15 times",

ROUND(bucket\_16\_20\_count::numeric / NULLIF(total\_cohort, 0) \* 100, 1) as "16-20 times",

ROUND(bucket\_over\_20\_count::numeric / NULLIF(total\_cohort, 0) \* 100, 1) as ">20 times",

total\_cohort as "Total\_FTD"

FROM monthly\_bucket\_counts

ORDER BY cohort\_month DESC;

EXISTING DEPOSITORS COHORT

/\* ============================================

EXISTING DEPOSITORS COHORT - PRODUCTION QUERY

Full dataset aggregation by months with complete filter suite

Shows unique count of existing depositors reaching deposit milestones within each month

One row per month across entire available data

Existing Depositor = first\_ever\_deposit BEFORE that month

Deposit count = number of deposits made DURING each month

FILTERS: Brand, Country, Traffic Source, Affiliate, Registration Launcher, Test Account, Currency

DATE RANGE: Optional start\_date and end\_date (defaults to all available data)

PARAMETERS: start\_date, end\_date, currency\_filter, brand, country, traffic\_source, affiliate\_id, affiliate\_name, registration\_launcher, is\_test\_account

============================================ \*/

WITH

/\* --- Optional inputs: empty by default, 1 row when provided --- \*/

start\_input AS (

SELECT NULL::date AS start\_date WHERE FALSE

[[ UNION ALL SELECT {{start\_date}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_date WHERE FALSE

[[ UNION ALL SELECT {{end\_date}}::date ]]

),

/\* --- Normalize window (no inputs → all available data) --- \*/

bounds\_raw AS (

SELECT

COALESCE((SELECT MAX(end\_date) FROM end\_input), CURRENT\_DATE) AS end\_date\_raw,

(SELECT MAX(start\_date) FROM start\_input) AS start\_date\_raw

),

bounds AS (

SELECT

end\_date\_raw AS end\_date,

/\* default start = beginning of all data; if user set start, use it \*/

CASE

WHEN start\_date\_raw IS NULL THEN (SELECT MIN(DATE\_TRUNC('month', created\_at))::date FROM transactions WHERE transaction\_category = 'deposit')

WHEN start\_date\_raw > end\_date\_raw THEN end\_date\_raw

ELSE start\_date\_raw

END AS start\_date

FROM bounds\_raw

),

/\* --- Step 1: Get all months in analysis period --- \*/

all\_months AS (

SELECT DISTINCT DATE\_TRUNC('month', d)::date AS month\_start

FROM generate\_series(

(SELECT start\_date FROM bounds),

(SELECT end\_date FROM bounds),

INTERVAL '1 month'

) AS d

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND CONCAT(players.os, ' / ', players.browser) = {{registration\_launcher}} ]]

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* Step 2: Identify first deposit date for all filtered players --- \*/

player\_first\_deposit AS (

SELECT

transactions.player\_id,

MIN(transactions.created\_at) as first\_deposit\_date

FROM transactions

INNER JOIN filtered\_players fp ON transactions.player\_id = fp.player\_id

JOIN players ON players.id = transactions.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE transactions.transaction\_category = 'deposit'

AND transactions.transaction\_type = 'credit'

AND transactions.balance\_type = 'withdrawable'

AND transactions.status = 'completed'

AND transactions.created\_at >= (SELECT start\_date FROM bounds)

AND transactions.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

-- Currency filter using standard hierarchy

[[ AND UPPER(COALESCE(

transactions.metadata->>'currency',

transactions.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY transactions.player\_id

),

/\* --- Step 3: Filter to EXISTING DEPOSITORS for each month --- \*/

/\* Existing = first deposit occurred BEFORE the month being analyzed \*/

existing\_depositors\_by\_month AS (

SELECT

am.month\_start,

pfd.player\_id,

pfd.first\_deposit\_date

FROM all\_months am

CROSS JOIN player\_first\_deposit pfd

INNER JOIN filtered\_players fp ON pfd.player\_id = fp.player\_id

WHERE pfd.first\_deposit\_date < am.month\_start

),

/\* --- Step 4: Count deposits per existing depositor per month with currency filter --- \*/

monthly\_deposit\_counts AS (

SELECT

edbm.month\_start,

edbm.player\_id,

COUNT(\*) as deposits\_in\_month

FROM existing\_depositors\_by\_month edbm

INNER JOIN transactions t ON edbm.player\_id = t.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

AND DATE\_TRUNC('month', t.created\_at)::date = edbm.month\_start

-- Currency filter using standard hierarchy

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY edbm.month\_start, edbm.player\_id

),

/\* --- Step 5: Calculate bucket distribution for each month --- \*/

monthly\_bucket\_counts AS (

SELECT

month\_start,

-- Individual buckets 1-10

COUNT(CASE WHEN deposits\_in\_month >= 1 THEN 1 END) as bucket\_1,

COUNT(CASE WHEN deposits\_in\_month >= 2 THEN 1 END) as bucket\_2,

COUNT(CASE WHEN deposits\_in\_month >= 3 THEN 1 END) as bucket\_3,

COUNT(CASE WHEN deposits\_in\_month >= 4 THEN 1 END) as bucket\_4,

COUNT(CASE WHEN deposits\_in\_month >= 5 THEN 1 END) as bucket\_5,

COUNT(CASE WHEN deposits\_in\_month >= 6 THEN 1 END) as bucket\_6,

COUNT(CASE WHEN deposits\_in\_month >= 7 THEN 1 END) as bucket\_7,

COUNT(CASE WHEN deposits\_in\_month >= 8 THEN 1 END) as bucket\_8,

COUNT(CASE WHEN deposits\_in\_month >= 9 THEN 1 END) as bucket\_9,

COUNT(CASE WHEN deposits\_in\_month >= 10 THEN 1 END) as bucket\_10,

-- Grouped buckets

COUNT(CASE WHEN deposits\_in\_month >= 11 AND deposits\_in\_month <= 15 THEN 1 END) as bucket\_11\_15,

COUNT(CASE WHEN deposits\_in\_month >= 16 AND deposits\_in\_month <= 20 THEN 1 END) as bucket\_16\_20,

COUNT(CASE WHEN deposits\_in\_month > 20 THEN 1 END) as bucket\_over\_20,

-- Total active existing depositors

COUNT(\*) as total\_active

FROM monthly\_deposit\_counts

GROUP BY month\_start

)

/\* --- Final Output: One row per month across all available data --- \*/

SELECT

TO\_CHAR(month\_start, 'FMMonth YYYY') as "Month",

month\_start as "month\_sort\_key",

bucket\_1 as "1 deposit",

bucket\_2 as "2 deposits",

bucket\_3 as "3 deposits",

bucket\_4 as "4 deposits",

bucket\_5 as "5 deposits",

bucket\_6 as "6 deposits",

bucket\_7 as "7 deposits",

bucket\_8 as "8 deposits",

bucket\_9 as "9 deposits",

bucket\_10 as "10 deposits",

bucket\_11\_15 as "11-15 deposits",

bucket\_16\_20 as "16-20 deposits",

bucket\_over\_20 as ">20 deposits",

total\_active as "Total Active Existing Depositors"

FROM monthly\_bucket\_counts

WHERE total\_active > 0 -- Exclude empty months

ORDER BY month\_start DESC;

EXISTING DEPOSITORS COHORT %

/\* ============================================

EXISTING DEPOSITORS COHORT (%) - PRODUCTION QUERY

Full dataset aggregation by months with complete filter suite

Shows percentage distribution of existing depositors across deposit frequency buckets

One row per month across entire available data

Existing Depositor = first\_ever\_deposit BEFORE that month

Percentage = (Depositors in bucket / Total existing depositors) \* 100

FILTERS: Brand, Country, Traffic Source, Affiliate, Registration Launcher, Test Account, Currency

DATE RANGE: Optional start\_date and end\_date (defaults to all available data)

PARAMETERS: start\_date, end\_date, currency\_filter, brand, country, traffic\_source, affiliate\_id, affiliate\_name, registration\_launcher, is\_test\_account

============================================ \*/

WITH

/\* --- Optional inputs: empty by default, 1 row when provided --- \*/

start\_input AS (

SELECT NULL::date AS start\_date WHERE FALSE

[[ UNION ALL SELECT {{start\_date}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_date WHERE FALSE

[[ UNION ALL SELECT {{end\_date}}::date ]]

),

/\* --- Normalize window (no inputs → all available data) --- \*/

bounds\_raw AS (

SELECT

COALESCE((SELECT MAX(end\_date) FROM end\_input), CURRENT\_DATE) AS end\_date\_raw,

(SELECT MAX(start\_date) FROM start\_input) AS start\_date\_raw

),

bounds AS (

SELECT

end\_date\_raw AS end\_date,

/\* default start = beginning of all data; if user set start, use it \*/

CASE

WHEN start\_date\_raw IS NULL THEN (SELECT MIN(DATE\_TRUNC('month', created\_at))::date FROM transactions WHERE transaction\_category = 'deposit')

WHEN start\_date\_raw > end\_date\_raw THEN end\_date\_raw

ELSE start\_date\_raw

END AS start\_date

FROM bounds\_raw

),

/\* --- Step 1: Get all months in analysis period --- \*/

all\_months AS (

SELECT DISTINCT DATE\_TRUNC('month', d)::date AS month\_start

FROM generate\_series(

(SELECT start\_date FROM bounds),

(SELECT end\_date FROM bounds),

INTERVAL '1 month'

) AS d

),

/\* ---------- FILTERED PLAYERS (NO ALIASES for Field Filters) ---------- \*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND {{brand}} ]] -- Field Filter → Companies.name

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

-- Text/Category variables (optional)

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

[[ AND {{affiliate\_id}} ]]

[[ AND {{affiliate\_name}} ]]

[[ AND CONCAT(players.os, ' / ', players.browser) = {{registration\_launcher}} ]]

-- TEST ACCOUNT as a Field Filter → Players.is\_test\_account (boolean)

[[ AND {{is\_test\_account}} ]]

),

/\* Step 2: Identify first deposit date for all filtered players --- \*/

player\_first\_deposit AS (

SELECT

transactions.player\_id,

MIN(transactions.created\_at) as first\_deposit\_date

FROM transactions

INNER JOIN filtered\_players fp ON transactions.player\_id = fp.player\_id

JOIN players ON players.id = transactions.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE transactions.transaction\_category = 'deposit'

AND transactions.transaction\_type = 'credit'

AND transactions.balance\_type = 'withdrawable'

AND transactions.status = 'completed'

AND transactions.created\_at >= (SELECT start\_date FROM bounds)

AND transactions.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

-- Currency filter using standard hierarchy

[[ AND UPPER(COALESCE(

transactions.metadata->>'currency',

transactions.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY transactions.player\_id

),

/\* --- Step 3: Filter to EXISTING DEPOSITORS for each month --- \*/

/\* Existing = first deposit occurred BEFORE the month being analyzed \*/

existing\_depositors\_by\_month AS (

SELECT

am.month\_start,

pfd.player\_id,

pfd.first\_deposit\_date

FROM all\_months am

CROSS JOIN player\_first\_deposit pfd

INNER JOIN filtered\_players fp ON pfd.player\_id = fp.player\_id

WHERE pfd.first\_deposit\_date < am.month\_start

),

/\* --- Step 4: Count deposits per existing depositor per month with currency filter --- \*/

monthly\_deposit\_counts AS (

SELECT

edbm.month\_start,

edbm.player\_id,

COUNT(\*) as deposits\_in\_month

FROM existing\_depositors\_by\_month edbm

INNER JOIN transactions t ON edbm.player\_id = t.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

AND DATE\_TRUNC('month', t.created\_at)::date = edbm.month\_start

-- Currency filter using standard hierarchy

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY edbm.month\_start, edbm.player\_id

),

/\* --- Step 5: Calculate bucket counts and total for each month --- \*/

monthly\_bucket\_counts AS (

SELECT

month\_start,

-- Individual buckets 1-10

COUNT(CASE WHEN deposits\_in\_month >= 1 THEN 1 END) as bucket\_1,

COUNT(CASE WHEN deposits\_in\_month >= 2 THEN 1 END) as bucket\_2,

COUNT(CASE WHEN deposits\_in\_month >= 3 THEN 1 END) as bucket\_3,

COUNT(CASE WHEN deposits\_in\_month >= 4 THEN 1 END) as bucket\_4,

COUNT(CASE WHEN deposits\_in\_month >= 5 THEN 1 END) as bucket\_5,

COUNT(CASE WHEN deposits\_in\_month >= 6 THEN 1 END) as bucket\_6,

COUNT(CASE WHEN deposits\_in\_month >= 7 THEN 1 END) as bucket\_7,

COUNT(CASE WHEN deposits\_in\_month >= 8 THEN 1 END) as bucket\_8,

COUNT(CASE WHEN deposits\_in\_month >= 9 THEN 1 END) as bucket\_9,

COUNT(CASE WHEN deposits\_in\_month >= 10 THEN 1 END) as bucket\_10,

-- Grouped buckets

COUNT(CASE WHEN deposits\_in\_month >= 11 AND deposits\_in\_month <= 15 THEN 1 END) as bucket\_11\_15,

COUNT(CASE WHEN deposits\_in\_month >= 16 AND deposits\_in\_month <= 20 THEN 1 END) as bucket\_16\_20,

COUNT(CASE WHEN deposits\_in\_month > 20 THEN 1 END) as bucket\_over\_20,

-- Total active existing depositors

COUNT(\*) as total\_active

FROM monthly\_deposit\_counts

GROUP BY month\_start

)

/\* --- Final Output: One row per month with percentages --- \*/

SELECT

TO\_CHAR(month\_start, 'FMMonth YYYY') as "Month",

month\_start as "month\_sort\_key",

ROUND(bucket\_1::numeric / NULLIF(total\_active, 0) \* 100, 1) as "1 deposit %",

ROUND(bucket\_2::numeric / NULLIF(total\_active, 0) \* 100, 1) as "2 deposits %",

ROUND(bucket\_3::numeric / NULLIF(total\_active, 0) \* 100, 1) as "3 deposits %",

ROUND(bucket\_4::numeric / NULLIF(total\_active, 0) \* 100, 1) as "4 deposits %",

ROUND(bucket\_5::numeric / NULLIF(total\_active, 0) \* 100, 1) as "5 deposits %",

ROUND(bucket\_6::numeric / NULLIF(total\_active, 0) \* 100, 1) as "6 deposits %",

ROUND(bucket\_7::numeric / NULLIF(total\_active, 0) \* 100, 1) as "7 deposits %",

ROUND(bucket\_8::numeric / NULLIF(total\_active, 0) \* 100, 1) as "8 deposits %",

ROUND(bucket\_9::numeric / NULLIF(total\_active, 0) \* 100, 1) as "9 deposits %",

ROUND(bucket\_10::numeric / NULLIF(total\_active, 0) \* 100, 1) as "10 deposits %",

ROUND(bucket\_11\_15::numeric / NULLIF(total\_active, 0) \* 100, 1) as "11-15 deposits %",

ROUND(bucket\_16\_20::numeric / NULLIF(total\_active, 0) \* 100, 1) as "16-20 deposits %",

ROUND(bucket\_over\_20::numeric / NULLIF(total\_active, 0) \* 100, 1) as ">20 deposits %",

total\_active as "Total Active Existing Depositors"

FROM monthly\_bucket\_counts

WHERE total\_active > 0 -- Exclude empty months

ORDER BY month\_start DESC;

COHORT LTV LIFETIME REPORT

/\*\*

=============================================================================

COHORT LTV LIFETIME REPORT - PRODUCTION SQL

Gaming/Casino Platform Analytics

=============================================================================

PURPOSE:

Aggregate lifetime metrics for each player cohort (by registration month)

showing: Registrations, FTDs, Conversion Rate, LTV, Deposits, Withdrawals,

GGR, NGR, Bonus Cost, and Revenue

DATA MODEL:

- One row per registration\_month (across all global filters)

- Metrics aggregated from registration date through TODAY

- All registrations included (including non-FTDs)

- Bonus metrics included in GGR calculation

- Proper currency resolution cascade applied

OUTPUT COLUMNS:

month\_year | YYYY-MM format registration month

REG | Total registrations in cohort

FTD | First-time depositors in cohort

conversion\_rate | (FTD / REG) \* 100 %

ltv | Deposit per FTD (LTV proxy)

deposit | Total deposits from cohort

wd | Total withdrawals from cohort

ggr | Gross Gaming Revenue (Cash Bet + Promo Bet - Cash Win - Promo Win)

ngr | Net Gaming Revenue (same as GGR in this setup)

bonus\_cost | Total bonus completion amounts (costs)

revenue | NGR - bonus\_cost (final revenue)

FILTERS APPLIED (all optional via Metabase widgets):

{{brand}} - Company filter

{{country}} - Player country mapping

{{traffic\_source}} - Organic vs Affiliate

{{affiliate\_id}} - Specific affiliate

{{affiliate\_name}} - Affiliate name search

{{registration\_launcher}} - OS / Browser combination

{{is\_test\_account}} - Exclude/include test accounts

{{currency\_filter}} - Currency code (EUR, USD, etc)

{{start\_date}} - Registration period start

{{end\_date}} - Registration period end

=============================================================================

\*/

WITH

/\*\*

---------------------------------------------------------------------------

STEP 1: DATE BOUNDS - Determine reporting window

---------------------------------------------------------------------------

Default: Last 24 months of registrations

Allows user override via {{start\_date}} and {{end\_date}} filters

Why this structure:

- Two separate input CTEs allow independent filter application

- bounds CTE provides clean references throughout query

- Default to last 24 months for cohort depth

\*/

start\_input AS (

SELECT NULL::date AS start\_date WHERE FALSE

[[ UNION ALL SELECT {{start\_date}}::date ]]

),

end\_input AS (

SELECT NULL::date AS end\_date WHERE FALSE

[[ UNION ALL SELECT {{end\_date}}::date ]]

),

bounds AS (

SELECT

COALESCE(

(SELECT MAX(end\_date) FROM end\_input),

CURRENT\_DATE

) AS end\_date,

COALESCE(

(SELECT MAX(start\_date) FROM start\_input),

CURRENT\_DATE - INTERVAL '24 months'

) AS start\_date

),

/\*\*

---------------------------------------------------------------------------

STEP 2: FILTERED PLAYERS - Apply all global filters as gatekeeper

---------------------------------------------------------------------------

This CTE is the SINGLE SOURCE OF TRUTH for which players are included.

Every metric calculation MUST inner join to this to inherit all filters.

Filters applied:

1. Brand filter (companies.name via {{brand}} widget)

2. Country filter (players.country with case mapping)

3. Traffic source (organic = no affiliate, affiliate = has affiliate\_id)

4. Specific affiliate ID or affiliate name

5. Registration launcher (OS / Browser combination)

6. Test account flag (boolean exclude)

Why INNER JOIN pattern:

- Ensures every metric only counts filtered players

- Single change to this CTE ripples through entire report

- Consistent with daily/monthly report architecture

Why NO ALIASES on company fields:

- Allows Metabase field filters to work automatically

- If you created an alias like "brand\_name", field filter breaks

\*/

filtered\_players AS (

SELECT DISTINCT players.id AS player\_id

FROM players

LEFT JOIN companies ON companies.id = players.company\_id

WHERE 1=1

/\* Brand filter - Companies.name field filter in Metabase \*/

[[ AND {{brand}} ]]

/\* Country filter with case mapping - full EU country coverage \*/

[[ AND players.country = CASE {{country}}

WHEN 'Romania' THEN 'RO'

WHEN 'France' THEN 'FR'

WHEN 'Germany' THEN 'DE'

WHEN 'Cyprus' THEN 'CY'

WHEN 'Poland' THEN 'PL'

WHEN 'Spain' THEN 'ES'

WHEN 'Italy' THEN 'IT'

WHEN 'Canada' THEN 'CA'

WHEN 'Australia' THEN 'AU'

WHEN 'United Kingdom' THEN 'GB'

WHEN 'Finland' THEN 'FI'

WHEN 'Albania' THEN 'AL'

WHEN 'Austria' THEN 'AT'

WHEN 'Belgium' THEN 'BE'

WHEN 'Brazil' THEN 'BR'

WHEN 'Bulgaria' THEN 'BG'

WHEN 'Georgia' THEN 'GE'

WHEN 'Greece' THEN 'GR'

WHEN 'Hungary' THEN 'HU'

WHEN 'India' THEN 'IN'

WHEN 'Netherlands' THEN 'NL'

WHEN 'Portugal' THEN 'PT'

WHEN 'Singapore' THEN 'SG'

WHEN 'Turkey' THEN 'TR'

WHEN 'United Arab Emirates' THEN 'AE'

WHEN 'Afghanistan' THEN 'AF'

WHEN 'Armenia' THEN 'AM'

WHEN 'Denmark' THEN 'DK'

WHEN 'Algeria' THEN 'DZ'

WHEN 'Andorra' THEN 'AD'

END ]]

/\* Traffic source filter - Organic vs Affiliate \*/

[[ AND CASE

WHEN {{traffic\_source}} = 'Organic' THEN players.affiliate\_id IS NULL

WHEN {{traffic\_source}} = 'Affiliate' THEN players.affiliate\_id IS NOT NULL

ELSE TRUE

END ]]

/\* Specific affiliate ID filter \*/

[[ AND {{affiliate\_id}} ]]

/\* Affiliate name search/filter \*/

[[ AND {{affiliate\_name}} ]]

/\* Device filter - OS and Browser combination \*/

[[ AND CONCAT(players.os, ' / ', players.browser) = {{registration\_launcher}} ]]

/\* Test account filter - boolean \*/

[[ AND {{is\_test\_account}} ]]

),

/\*\*

---------------------------------------------------------------------------

STEP 3: PLAYER COHORTS - Register all filtered players with their cohort

---------------------------------------------------------------------------

This is our TRUTH TABLE for player counts by registration month.

Key decisions:

- Uses actual registration date (p.created\_at) not bucket date

- Truncates to month for cohort grouping (later queries will group by this)

- Stores full timestamp for potential future use

- Only includes players in bounds window

- INNER JOIN to filtered\_players ensures all global filters apply

Why separate CTE:

- Other CTEs join to this to get cohort membership

- Keeps cohort logic in one place

- Easier to debug cohort membership issues

\*/

player\_cohorts AS (

SELECT

p.id AS player\_id,

p.created\_at AS registration\_ts,

DATE\_TRUNC('month', p.created\_at)::date AS registration\_month,

p.company\_id

FROM players p

INNER JOIN filtered\_players fp ON p.id = fp.player\_id

WHERE p.created\_at >= (SELECT start\_date FROM bounds)

AND p.created\_at < (SELECT end\_date FROM bounds) + INTERVAL '1 day'

),

/\*\*

---------------------------------------------------------------------------

STEP 4: FTD DATA - Identify first deposits per player

---------------------------------------------------------------------------

This CTE finds each player's FIRST successful deposit.

Why separate CTE:

- Used later to count unique FTDs per cohort

- Can reuse for other metrics (deposit frequency, etc)

- Clear separation of concerns

Key logic:

- MIN(t.created\_at) gets first deposit chronologically

- Filters: category=deposit, type=credit, status=completed, balance=withdrawable

- Currency filter applied here too (must match global currency context)

- INNER JOIN to player\_cohorts ensures only included players have FTD data

Why INNER JOIN instead of LEFT:

- If player has no valid deposit, they won't appear

- Later LEFT JOIN in aggregation allows counting as non-FTD

- This separation is intentional and correct

Currency resolution cascade (FROM YOUR SCHEMA):

1. t.metadata->>'currency' - Most specific (transaction override)

2. t.cash\_currency - Transaction default

3. players.wallet\_currency - Player account default

4. companies.currency - Company/brand default

This ensures proper currency filtering across transaction types.

\*/

ftd\_data AS (

SELECT

t.player\_id,

MIN(t.created\_at) AS first\_deposit\_ts

FROM transactions t

INNER JOIN player\_cohorts pc ON t.player\_id = pc.player\_id

JOIN players ON players.id = t.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.status = 'completed'

AND t.balance\_type = 'withdrawable'

/\* Currency filter - cascade through transaction hierarchy \*/

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY t.player\_id

),

/\*\*

---------------------------------------------------------------------------

STEP 5: DEPOSIT & WITHDRAWAL METRICS - Aggregate cash flow per cohort

---------------------------------------------------------------------------

Sums deposits and withdrawals grouped by registration cohort.

Why GROUP BY registration\_month:

- We want one row per cohort month

- All players in same registration\_month group together

- Results in one row per registration month (our desired granularity)

LEFT JOIN pattern:

- Some months may have no deposits (e.g., zero-deposit month)

- LEFT JOIN ensures all cohort months appear (even if zero deposits)

- COALESCE in final SELECT handles NULL → 0 conversion

Filters:

- deposit: category=deposit, type=credit, status=completed, balance=withdrawable

- withdrawal: category=withdrawal, type=debit, status=completed, balance=withdrawable

- Currency filter applied

- INNER JOIN to player\_cohorts ensures only filtered players

Why ABS() on withdrawals:

- Withdrawals stored as negative amounts in some systems

- ABS ensures we sum positive values

- May not be needed in your schema, but safe to include

\*/

deposit\_withdrawal\_metrics AS (

SELECT

pc.registration\_month,

COALESCE(SUM(CASE

WHEN t.transaction\_category = 'deposit'

AND t.transaction\_type = 'credit'

AND t.status = 'completed'

AND t.balance\_type = 'withdrawable'

THEN t.amount

END), 0) AS total\_deposits,

COALESCE(SUM(CASE

WHEN t.transaction\_category = 'withdrawal'

AND t.transaction\_type = 'debit'

AND t.status = 'completed'

AND t.balance\_type = 'withdrawable'

THEN ABS(t.amount)

END), 0) AS total\_withdrawals

FROM player\_cohorts pc

LEFT JOIN transactions t ON pc.player\_id = t.player\_id

JOIN players ON players.id = pc.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY pc.registration\_month

),

/\*\*

---------------------------------------------------------------------------

STEP 6: GGR/NGR METRICS - Aggregate betting metrics per cohort

---------------------------------------------------------------------------

Calculates Gross Gaming Revenue components by cohort.

GGR Formula (Gaming Industry Standard):

GGR = (Cash Bet + Promo Bet) - (Cash Win + Promo Win)

Where:

- Cash Bet: game\_bet category, debit, withdrawable balance, completed

- Cash Win: game\_bet category, credit, withdrawable balance, completed

- Promo Bet: bonus category, debit, non-withdrawable balance, completed

- Promo Win: bonus category, credit, non-withdrawable balance, completed

Why separate Cash and Promo:

- Bonus play tracked separately per gaming platform conventions

- Allows analysis of bonus impact vs cash play

- Both contribute to GGR (total revenue)

Why 'non-withdrawable' for bonus:

- Bonus balance can't be withdrawn directly

- Must be wagered/converted to cash

- Separate lifecycle from cash balance

Key insight:

NGR in this report = GGR (your platform doesn't deduct taxes in transaction layer)

- Some platforms calculate: NGR = GGR - Taxes - Fees

- Your taxes/fees calculated downstream in revenue calc

- So GGR = NGR for transaction aggregation purposes

\*/

ggr\_ngr\_metrics AS (

SELECT

pc.registration\_month,

COALESCE(SUM(CASE

WHEN t.transaction\_type = 'debit'

AND t.transaction\_category = 'game\_bet'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

THEN ABS(t.amount)

END), 0) AS cash\_bet,

COALESCE(SUM(CASE

WHEN t.transaction\_type = 'credit'

AND t.transaction\_category = 'game\_bet'

AND t.balance\_type = 'withdrawable'

AND t.status = 'completed'

THEN t.amount

END), 0) AS cash\_win,

COALESCE(SUM(CASE

WHEN t.transaction\_type = 'debit'

AND t.transaction\_category = 'bonus'

AND t.balance\_type = 'non-withdrawable'

AND t.status = 'completed'

THEN ABS(t.amount)

END), 0) AS promo\_bet,

COALESCE(SUM(CASE

WHEN t.transaction\_type = 'credit'

AND t.transaction\_category = 'bonus'

AND t.balance\_type = 'non-withdrawable'

AND t.status = 'completed'

THEN t.amount

END), 0) AS promo\_win

FROM player\_cohorts pc

LEFT JOIN transactions t ON pc.player\_id = t.player\_id

JOIN players ON players.id = pc.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY pc.registration\_month

),

/\*\*

---------------------------------------------------------------------------

STEP 7: BONUS COST METRICS - Aggregate bonus costs per cohort

---------------------------------------------------------------------------

Sums all bonus completion amounts (bonuses that converted to real money).

Bonus Cost Definition:

- All bonus\_completion transactions that became withdrawable

- Represents the COST to platform of bonus promotions

- Includes original bonus amount + wins generated from bonus play

- Could also track separately if needed

Filter logic:

- category = 'bonus\_completion' (not 'bonus' which is bonus play activity)

- type = 'credit' (money going INTO withdrawable balance)

- status = 'completed' (only finalized bonuses)

- balance\_type = 'withdrawable' (bonus converted to cash)

Why separate CTE:

- Bonus costs might be calculated differently in future

- Easy to modify bonus rules in one place

- Other reports might need bonus metrics

- Clear intent in code

\*/

bonus\_cost\_metrics AS (

SELECT

pc.registration\_month,

COALESCE(SUM(CASE

WHEN t.transaction\_type = 'credit'

AND t.transaction\_category = 'bonus\_completion'

AND t.status = 'completed'

AND t.balance\_type = 'withdrawable'

THEN t.amount

END), 0) AS total\_bonus\_cost

FROM player\_cohorts pc

LEFT JOIN transactions t ON pc.player\_id = t.player\_id

JOIN players ON players.id = pc.player\_id

JOIN companies ON companies.id = players.company\_id

WHERE 1=1

[[ AND UPPER(COALESCE(

t.metadata->>'currency',

t.cash\_currency,

players.wallet\_currency,

companies.currency

)) IN ({{currency\_filter}}) ]]

GROUP BY pc.registration\_month

),

/\*\*

---------------------------------------------------------------------------

STEP 8: COHORT LIFETIME METRICS - Combine all aggregations

---------------------------------------------------------------------------

This is the MAIN AGGREGATION that brings all metrics together.

Join pattern:

- Start with player\_cohorts (LEFT JOIN) to ensure all cohorts appear

- FTD data (LEFT JOIN) - not all players have deposits

- All metric CTEs (LEFT JOIN) - some cohorts may have zero activity

Why LEFT JOINs:

- Non-FTD players should appear as 0, not excluded

- Cohorts with zero activity should still show 0, not be missing

- This preserves complete cohort visibility

GROUP BY logic:

- Group by registration\_month to aggregate all players in that cohort

- COUNT(DISTINCT player\_id) for total registrations

- COUNT(DISTINCT ftd.player\_id) for FTDs (LEFT JOIN handles nulls)

- Other columns come from metric CTEs (already grouped by registration\_month)

Why COUNT(DISTINCT ...):

- Ensures each player counted once even if multiple transactions

- LEFT JOINs could cause duplicate counts without DISTINCT

- DISTINCT ftd.player\_id counts non-null FTD records only

\*/

cohort\_lifetime\_metrics AS (

SELECT

pc.registration\_month,

COUNT(DISTINCT pc.player\_id) AS total\_registrations,

COUNT(DISTINCT ftd.player\_id) AS ftd\_count,

COALESCE(dwm.total\_deposits, 0) AS total\_deposits,

COALESCE(dwm.total\_withdrawals, 0) AS total\_withdrawals,

COALESCE(ggr.cash\_bet, 0) AS cash\_bet,

COALESCE(ggr.cash\_win, 0) AS cash\_win,

COALESCE(ggr.promo\_bet, 0) AS promo\_bet,

COALESCE(ggr.promo\_win, 0) AS promo\_win,

COALESCE(bcm.total\_bonus\_cost, 0) AS total\_bonus\_cost

FROM player\_cohorts pc

LEFT JOIN ftd\_data ftd ON pc.player\_id = ftd.player\_id

LEFT JOIN deposit\_withdrawal\_metrics dwm ON pc.registration\_month = dwm.registration\_month

LEFT JOIN ggr\_ngr\_metrics ggr ON pc.registration\_month = ggr.registration\_month

LEFT JOIN bonus\_cost\_metrics bcm ON pc.registration\_month = bcm.registration\_month

GROUP BY pc.registration\_month, dwm.total\_deposits, dwm.total\_withdrawals,

ggr.cash\_bet, ggr.cash\_win, ggr.promo\_bet, ggr.promo\_win, bcm.total\_bonus\_cost

),

/\*\*

---------------------------------------------------------------------------

STEP 9: FINAL REPORT DATA - Calculate KPIs and format output

---------------------------------------------------------------------------

This CTE calculates all derived metrics and formats them for display.

Column calculations:

1. month\_year = TO\_CHAR(registration\_month, 'YYYY-MM')

Purpose: Human-readable month for display

Example: '2024-08' for August 2024

2. REG = total\_registrations

Purpose: Total players registered in this month

Range: [0, infinity)

3. FTD = ftd\_count

Purpose: Players who made first deposit

Range: [0, REG]

Note: FTD ≤ REG always

4. conversion\_rate = (FTD / REG) \* 100

Purpose: % of registrations that became depositors

Formula: ROUND(FTD::numeric / NULLIF(REG, 0) \* 100, 2)

Range: [0, 100]

Note: NULLIF prevents division by zero, returns 0 if REG=0

5. ltv = total\_deposits / FTD

Purpose: Average deposit per first-time depositor

Formula: ROUND(total\_deposits::numeric / NULLIF(FTD, 0), 2)

Range: [0, infinity)

Note: This is "deposits per FTD" which proxies LTV at deposit stage

Interpretation: Higher = better initial deposits

6. deposit = total\_deposits

Purpose: Total deposits from cohort

Formula: ROUND(total\_deposits, 2)

Range: [0, infinity)

Currency: € (euros in your reports)

7. wd = total\_withdrawals

Purpose: Total withdrawals from cohort

Formula: ROUND(total\_withdrawals, 2)

Range: [0, infinity)

Currency: €

8. ggr = cash\_bet + promo\_bet - cash\_win - promo\_win

Purpose: Gross Gaming Revenue (platform profit before costs)

Formula: ROUND(cash\_bet + promo\_bet - cash\_win - promo\_win, 2)

Range: Can be negative if payouts exceed bets

Currency: €

9. ngr = ggr (in your platform setup)

Purpose: Net Gaming Revenue (same as GGR in transaction layer)

Formula: ROUND(cash\_bet + promo\_bet - cash\_win - promo\_win, 2)

Range: Same as GGR

Note: Taxes/fees deducted later in revenue calculation

10. bonus\_cost = total\_bonus\_cost

Purpose: Total cost of bonus conversions

Formula: ROUND(total\_bonus\_cost, 2)

Range: [0, infinity)

Currency: €

11. revenue = NGR - bonus\_cost

Purpose: Final net revenue after bonus costs

Formula: ROUND((ggr - total\_bonus\_cost), 2)

Range: Can be negative

Currency: €

Interpretation: Revenue platform keeps after bonuses

sort\_order = 0:

- Used for ordering (0 = regular rows, -1 = total row)

- Allows TOTAL to appear first via ORDER BY sort\_order, month\_year DESC

\*/

final\_report\_data AS (

SELECT

0 AS sort\_order,

TO\_CHAR(registration\_month, 'YYYY-MM') AS month\_year,

total\_registrations AS REG,

ftd\_count AS FTD,

ROUND(CASE

WHEN total\_registrations > 0

THEN ftd\_count::numeric / total\_registrations \* 100

ELSE 0

END, 2) AS conversion\_rate,

ROUND(CASE

WHEN ftd\_count > 0

THEN total\_deposits::numeric / ftd\_count

ELSE 0

END, 2) AS ltv,

ROUND(total\_deposits, 2) AS deposit,

ROUND(total\_withdrawals, 2) AS wd,

ROUND(cash\_bet + promo\_bet - cash\_win - promo\_win, 2) AS ggr,

ROUND(cash\_bet + promo\_bet - cash\_win - promo\_win, 2) AS ngr,

ROUND(total\_bonus\_cost, 2) AS bonus\_cost,

ROUND((cash\_bet + promo\_bet - cash\_win - promo\_win) - total\_bonus\_cost, 2) AS revenue

FROM cohort\_lifetime\_metrics

)

/\*\*

---------------------------------------------------------------------------

FINAL OUTPUT - Combine TOTAL row with individual month rows

---------------------------------------------------------------------------

The TOTAL row aggregates all months together, showing platform-wide metrics.

TOTAL row calculation:

- sort\_order = -1 (ensures it appears first via ORDER BY)

- month\_year = 'TOTAL' (label)

- REG = SUM(all registrations) across all months

- FTD = SUM(all FTDs) across all months

- conversion\_rate = SUM(FTDs) / SUM(REGs) \* 100 (overall conversion)

- ltv = SUM(deposits) / SUM(FTDs) (average deposit per FTD overall)

- deposit/wd/ggr/ngr/bonus\_cost/revenue = SUM of each

Why separate TOTAL from monthly rows:

- Some DBs don't handle WITH ROLLUP well

- Explicit UNION ALL is more portable

- Easier to understand calculation logic

- Can disable TOTAL if needed (just remove this UNION)

ORDER BY sort\_order, month\_year DESC:

- sort\_order ASC puts TOTAL (-1) first

- month\_year DESC puts newest months first (after TOTAL)

- Result: [TOTAL] [2024-09] [2024-08] [2024-07] ...

\*/

-- TOTAL ROW (aggregated across all months)

SELECT

-1 AS sort\_order,

'TOTAL' AS month\_year,

SUM(REG) AS REG,

SUM(FTD) AS FTD,

ROUND(SUM(FTD)::numeric / NULLIF(SUM(REG), 0) \* 100, 2) AS conversion\_rate,

ROUND(SUM(deposit)::numeric / NULLIF(SUM(FTD), 0), 2) AS ltv,

ROUND(SUM(deposit), 2) AS deposit,

ROUND(SUM(wd), 2) AS wd,

ROUND(SUM(ggr), 2) AS ggr,

ROUND(SUM(ngr), 2) AS ngr,

ROUND(SUM(bonus\_cost), 2) AS bonus\_cost,

ROUND(SUM(revenue), 2) AS revenue

FROM final\_report\_data

UNION ALL

-- INDIVIDUAL MONTH ROWS

SELECT \* FROM final\_report\_data

ORDER BY sort\_order, month\_year DESC;