BAMBASWAP

Customer Acquisition Rate

**Explanation of Customer Acquisition Rate (CAR)**

**Customer Acquisition Rate (CAR)** measures the rate at which a business gains new customers over a specific time period. In this case, for the **bambaswap\_combined\_jan\_aug\_2024** data, we are interested in calculating the **CAR** for each month (from February to August) by comparing the number of **new customers** acquired in a given month to the total number of customers from the **previous month**.

The formula is:

CARMonth=New Customers in Current MonthTotal Customers in Previous Month×100\text{CAR}\_{\text{Month}} = \frac{\text{New Customers in Current Month}}{\text{Total Customers in Previous Month}} \times 100CARMonth​=Total Customers in Previous MonthNew Customers in Current Month​×100

**How CAR is Calculated in the Query**

1. **Monthly Customers** (monthly\_customers CTE):
   * This part of the query identifies **distinct customers** (phone\_number) for each month, using the date\_only column to extract the month number and month name. This creates a list of unique customers and associates them with the **month** they transacted in.
   * Example: A customer who made a transaction in **January** will be grouped as a **January** customer, and the same logic applies to all other months.

sql

Copy code

SELECT DISTINCT phone\_number,

EXTRACT(MONTH FROM date\_only) AS month\_number,

TO\_CHAR(date\_only, 'Month') AS transaction\_month

1. **New Customers** (new\_customers\_per\_month CTE):
   * This part of the query uses the LAG() window function to check if a customer existed in the **previous month**. If a customer appears in the current month but not in the previous month, they are considered a **new customer**.
   * This helps identify customers who are new for each month by comparing the current month to the previous month.

sql

Copy code

LAG(mc.transaction\_month) OVER (PARTITION BY mc.phone\_number ORDER BY mc.month\_number) AS previous\_month

1. **Counting New Customers** (new\_customers\_by\_month CTE):
   * Here, the query counts how many **new customers** there are for each month by grouping them by transaction\_month and month\_number.
   * Only customers who did not appear in the previous month are counted as new customers.

sql

Copy code

COUNT(phone\_number) AS new\_customers

1. **Total Customers in Each Month** (total\_customers\_by\_month CTE):
   * This part of the query calculates the **total number of customers** (both new and returning) for each month.
   * This is done by grouping customers by month and ensuring that duplicates are removed using COUNT(DISTINCT phone\_number).
2. **Previous Month's Customers** (previous\_month\_customers CTE):
   * Using the **LAG()** function, this section calculates how many **total customers** were present in the **previous month** for each month. This is crucial for calculating the CAR, as it allows us to compare the new customers in the current month to the total customers in the previous month.

sql

Copy code

LAG(total\_customers) OVER (ORDER BY month\_number) AS previous\_month\_customers

1. **Final CAR Calculation**:
   * The main query calculates the **Customer Acquisition Rate** by dividing the number of **new customers** in the current month by the **total customers in the previous month**. The result is multiplied by 100 to get the CAR as a percentage.

sql

Copy code

(n.new\_customers::NUMERIC / p.previous\_month\_customers) \* 100 AS acquisition\_rate

**Practical Example of CAR Calculation**

Let’s say the following happens:

* **January** has **100 customers**.
* In **February**, 50 new customers are acquired (and 50 customers from January return).
* In **March**, 30 new customers are acquired.

The CAR for February would be:

CARFebruary=50100×100=50%\text{CAR}\_{\text{February}} = \frac{50}{100} \times 100 = 50\%CARFebruary​=10050​×100=50%

The CAR for March would be:

CARMarch=30150×100=20%\text{CAR}\_{\text{March}} = \frac{30}{150} \times 100 = 20\%CARMarch​=15030​×100=20%

**Summary of the CAR Query:**

1. **Calculates the number of new customers** acquired each month by comparing them to the previous month.
2. **Uses the date\_only column** for efficient date extraction.
3. **Calculates the CAR** for each month by comparing the number of new customers to the total customers in the previous month.
4. **Outputs CAR** as a percentage, helping to track how successful Bambaswap is at acquiring new customers over time.

**Key Insights from CAR:**

* **High CAR** indicates that the business is acquiring new customers at a high rate compared to the previous month.
* **Low CAR** or a decreasing CAR trend may indicate that the business is struggling to attract new customers, which could point to the need for adjustments in marketing or customer acquisition strategies.

The query and results provide a clear picture of how Bambaswap’s customer base is growing (or not) over the period from **January to August 2024**.

2. Bambaswap Customer Conversion Rate

**Here's how you can modify the query to calculate the conversion rate:**

sql

Copy code

WITH transacted\_customers AS (

SELECT

month,

month\_number,

COUNT(DISTINCT phone\_number) AS transacted\_count

FROM

bambaswap\_combined\_jan\_aug\_2024

WHERE

state = 'complete'

GROUP BY

month, month\_number

),

total\_leads AS (

SELECT

month,

month\_number,

COUNT(DISTINCT phone\_number) AS total\_leads\_count

FROM

bambaswap\_combined\_jan\_aug\_2024

GROUP BY

month, month\_number

)

SELECT

t.month,

t.month\_number,

t.transacted\_count,

l.total\_leads\_count,

ROUND(

(t.transacted\_count::decimal / l.total\_leads\_count) \* 100, 2

) AS conversion\_rate\_percentage

FROM

transacted\_customers t

JOIN

total\_leads l ON t.month = l.month AND t.month\_number = l.month\_number

ORDER BY

t.month\_number;

**Explanation:**

1. **transacted\_customers CTE**:
   * This counts the number of **distinct customers who transacted** (where state = 'complete') in each month (month\_number).
2. **total\_leads CTE**:
   * This counts the total number of **distinct leads** (unique phone numbers) for each month, regardless of their state.
3. **Main Query**:
   * It joins the transacted\_customers and total\_leads CTEs on the month and month\_number columns.
   * The **conversion rate** is calculated as the number of transacted customers divided by the total leads, multiplied by 100.
   * The **ROUND function** is used to round the conversion rate to two decimal places.

**Output:**

* **month**: Name of the month (January, February, etc.).
* **month\_number**: The numeric representation of the month.
* **transacted\_count**: Number of customers who transacted (state = 'complete').
* **total\_leads\_count**: Total unique leads (distinct phone numbers) in that month.
* **conversion\_rate\_percentage**: The conversion rate as a percentage.

4.

**Bambaswap Monthly Recurring Rate**

**Explanation for the MRR and ARPU Query:**

The query you provided calculates two key metrics for each month:

1. **ARPU (Average Revenue Per User)**: This is the total revenue divided by the number of **loyal customers** (customers who made 3 or more transactions in that month).
2. **MRR (Monthly Recurring Revenue)**: This is calculated by multiplying ARPU by the number of loyal customers, providing the monthly revenue attributable to loyal, repeat customers.

**Breakdown of the Query:**

**1. arpu CTE (Common Table Expression)**

* This part calculates the **total revenue** (bs\_revenue) for each month by summing up the bs\_revenue for all transactions:

sql

Copy code

SELECT

month,

month\_number,

SUM(bs\_revenue) AS total\_revenue

FROM

bambaswap\_combined\_jan\_aug\_2024

GROUP BY

month, month\_number

* **Result**: The total revenue for each month is stored in the arpu CTE, with columns month, month\_number, and total\_revenue.

**2. filtered\_numbers CTE**

* This part identifies **loyal customers**, defined as customers who made 3 or more transactions in a given month. It counts distinct phone\_numbers that meet this condition:

sql

Copy code

SELECT

month,

month\_number,

COUNT(DISTINCT phone\_number) AS loyal\_customers

FROM (

SELECT

phone\_number,

month,

month\_number

FROM bambaswap\_combined\_jan\_aug\_2024

GROUP BY

phone\_number, month, month\_number

HAVING COUNT(\*) >= 3 -- Loyal customers appear 3 or more times in the month

) AS loyal\_customers\_per\_month

GROUP BY

month, month\_number

* **Result**: The number of loyal customers (those who transacted 3 or more times in the month) is calculated for each month.

**3. arpu\_calculation CTE**

* This part calculates the **ARPU** (Average Revenue Per User) for each month by dividing the **total revenue** by the number of **loyal customers**:

sql

Copy code

SELECT

arpu.month,

arpu.month\_number,

ROUND(arpu.total\_revenue / filtered\_numbers.loyal\_customers, 2) AS arpu

FROM

arpu

JOIN

filtered\_numbers ON arpu.month = filtered\_numbers.month

* **Result**: The ARPU for each month is calculated, rounding to two decimal places.

**4. Final MRR Calculation**

* The final part calculates the **Monthly Recurring Revenue (MRR)** by multiplying the ARPU by the number of loyal customers:

sql

Copy code

SELECT

arpu\_calculation.month,

arpu\_calculation.month\_number,

arpu\_calculation.arpu,

ROUND (arpu\_calculation.arpu \* filtered\_numbers.loyal\_customers, 2) AS mrr

FROM

arpu\_calculation

JOIN

filtered\_numbers ON arpu\_calculation.month = filtered\_numbers.month

ORDER BY

arpu\_calculation.month\_number;

* **MRR** is calculated as ARPU \* loyal\_customers for each month. The result is rounded to two decimal places.

**Key Concepts:**

1. **Total Revenue**: The sum of bs\_revenue for all transactions in a month.
2. **Loyal Customers**: Customers who made 3 or more transactions in a given month.
3. **ARPU (Average Revenue Per User)**: Total revenue divided by the number of loyal customers.
4. **MRR (Monthly Recurring Revenue)**: ARPU multiplied by the number of loyal customers, representing the recurring revenue generated from loyal customers in a month.

**Query Purpose:**

* **ARPU** helps you understand the average revenue generated per loyal customer.
* **MRR** represents the monthly revenue from recurring transactions (loyal customers), which is important for understanding the health of a business with recurring or repeat customers.

This query is useful for tracking the financial performance of loyal, recurring customers over time.

**Bambaswap Monthly Retention Rate**

**Explanation of the SQL Query**

1. **Step 1: Customer\_transactions (Counting transactions)**
   * This **subquery** groups the data by phone\_number and month, calculating how many transactions each customer made per month.
   * COUNT(\*) counts the transactions for each customer within a specific month.
   * The result is a list of phone\_number, month, and transaction\_count for each customer.
2. **Step 2: qualified\_customers (Filtering customers with more than 3 transactions)**
   * This subquery filters the customers from customer\_transactions to include only those who made more than 3 transactions per month.
   * It returns the phone\_number, month, and month\_number for customers who qualify.
3. **Step 3: retained\_customers (Finding customers retained across consecutive months)**
   * This subquery identifies customers who made more than 3 transactions in **consecutive months**.
   * It performs a **self-join** on qualified\_customers, linking customers in one month (q1) to the same customers in the next month (q2) by ensuring that their month\_number values are consecutive (q1.month\_number = q2.month\_number - 1).
   * The result is a list of phone\_number, the current month (current\_month), and the next month (next\_month), for those who made more than 3 transactions in both months.
4. **Final Step: Calculating Retention Rate**
   * This final query calculates the **retention rate** for each month by:
     + Counting how many customers from the current month (q1) made more than 3 transactions in the **next month** (q2), which are considered retained customers.
     + Calculating the retention rate as the percentage of retained customers divided by the total number of customers in that month (q1).
   * It groups by month and orders by month\_number to ensure a chronological order in the results.

**What to Look for in the Results:**

1. **current\_month**: This column shows the month for which the retention rate is being calculated.
2. **retained\_customers**: This column shows the number of customers who made more than 3 transactions in the current month and were retained (i.e., also made more than 3 transactions) in the next month.
3. **total\_customers**: This shows the total number of customers who made more than 3 transactions in the current month.
4. **retention\_rate**: This is the key result. It shows the percentage of customers from one month who continued making more than 3 transactions in the next month.
   * A **high retention rate** (closer to 100%) means that most customers who made more than 3 transactions in the current month also did so in the next month.
   * A **low retention rate** suggests that fewer customers remained active in consecutive months.

**Key Insights:**

* **Trend Analysis**: Track the retention rate over several months to identify whether customer engagement is improving or declining.
* **Identify Drop-off Points**: Look for months with a significant drop in the retention rate to understand when and why customers are disengaging.
* **Customer Loyalty**: A consistently high retention rate indicates strong customer loyalty among those who perform multiple transactions.

You can use these results to assess the performance of customer retention efforts and decide if any changes need to be made to boost customer engagement. Let me know if you have any other questions or need help interpreting your specific results!

5.

**Bambaswap Monthly Churn Rate**

The **monthly churn rate** measures the percentage of customers who stop engaging (or do not return) after a given month. It's the inverse of the retention rate, focusing on the customers who are not retained.

**Formula for Monthly Churn Rate:**

Churn Rate=100−Retention Rate\text{Churn Rate} = 100 - \text{Retention Rate}Churn Rate=100−Retention Rate

Alternatively, you can directly calculate the churn rate using:

Churn Rate=(Customers who did not return in the next monthTotal customers at the start of the month)×100\text{Churn Rate} = \left( \frac{\text{Customers who did not return in the next month}}{\text{Total customers at the start of the month}} \right) \times 100Churn Rate=(Total customers at the start of the monthCustomers who did not return in the next month​)×100

**In your case (for customers with more than 3 transactions):**

1. **Customers who churned**: These are the customers who made more than 3 transactions in the current month but did not make more than 3 transactions in the next month.
2. **Total customers**: The total number of customers who made more than 3 transactions in the current month.

**Steps to Calculate Monthly Churn Rate:**

1. Modify the existing SQL query to calculate the number of customers who did not return (churned) in the next month.
2. Use the churn rate formula based on those who did not return.

Here’s how you can adjust the SQL query to calculate the **monthly churn rate**:

sql

Copy code

WITH customer\_transactions AS (

-- Count the number of transactions per phone\_number per month

SELECT

phone\_number,

month,

COUNT(\*) AS transaction\_count

FROM

bambaswap\_combined\_jan\_aug\_2024

GROUP BY

phone\_number, month

),

qualified\_customers AS (

-- Filter for customers with more than 3 transactions per month

SELECT

phone\_number,

month,

month\_number

FROM

customer\_transactions

WHERE

transaction\_count > 3

),

retained\_customers AS (

-- Join the customers with consecutive months to find retained customers

SELECT

q1.phone\_number,

q1.month AS current\_month,

q2.month AS next\_month

FROM

qualified\_customers q1

INNER JOIN

qualified\_customers q2

ON

q1.phone\_number = q2.phone\_number

AND q1.month\_number = q2.month\_number - 1 -- Check for consecutive months

),

churned\_customers AS (

-- Find customers who churned (did not return in the next month)

SELECT

q1.phone\_number,

q1.month AS current\_month

FROM

qualified\_customers q1

LEFT JOIN

retained\_customers q2

ON

q1.phone\_number = q2.phone\_number

AND q1.month = q2.current\_month

WHERE

q2.phone\_number IS NULL -- Only include customers who did not return

)

-- Final calculation of the churn rate

SELECT

q1.month AS current\_month,

COUNT(DISTINCT q1.phone\_number) AS total\_customers,

COUNT(DISTINCT c.phone\_number) AS churned\_customers,

(COUNT(DISTINCT c.phone\_number) \* 100.0) / COUNT(DISTINCT q1.phone\_number) AS churn\_rate

FROM

qualified\_customers q1

LEFT JOIN

churned\_customers c

ON

q1.phone\_number = c.phone\_number

AND q1.month = c.current\_month

GROUP BY

q1.month

ORDER BY

q1.month\_number;

**Explanation of Changes:**

1. **churned\_customers**:
   * This subquery identifies customers who made more than 3 transactions in the current month but did not make more than 3 transactions in the next month.
   * A **LEFT JOIN** with retained\_customers is used to find those who do not appear in consecutive months (i.e., they churned).
   * The condition q2.phone\_number IS NULL ensures that only customers who did not return in the next month are included.
2. **Final Step: Calculating the Churn Rate**:
   * The final query calculates the churn rate by:
     + Counting the total number of customers in each month (q1).
     + Counting the number of churned customers (c).
     + Dividing the churned customers by the total number of customers to get the churn rate percentage.

**What to Look for in the Results:**

* **current\_month**: The month for which the churn rate is calculated.
* **total\_customers**: The total number of customers who made more than 3 transactions in that month.
* **churned\_customers**: The number of customers who did not make more than 3 transactions in the following month.
* **churn\_rate**: The churn rate for that month (percentage of customers who did not return).

**Key Insights:**

* **High churn rate**: Indicates that a significant number of customers did not return in the next month. This suggests an issue with customer engagement or satisfaction.
* **Low churn rate**: Means most customers are returning, indicating good retention and engagement.

By analyzing churn rate trends across months, you can assess where and when customer drop-offs are happening and potentially investigate the causes to take action.

6.

**Bambaswap Customer Lifetime Value (CLV) Explanation**

**Customer Lifetime Value (CLV)** is a key metric in business that represents the **total revenue** a business can expect to earn from a single customer over the entire time that the customer interacts with the business. It helps businesses understand the long-term value of their customers and assess how much they should spend to acquire and retain customers.

In the context of your **bambaswap\_consolidated\_jan\_may\_2024** data, this query calculates **CLV** for each month by considering the **average revenue per user (ARPU)** and the **churn rate**.

**Explanation of Key Concepts in the Query**

1. **Average Revenue Per User (ARPU)**:
   * ARPU is a measure of the **average revenue generated per customer** in a given month. It helps to understand how much, on average, each customer contributes in terms of revenue.
   * **Formula**: ARPU=Total RevenueTotal Number of Customers\text{ARPU} = \frac{\text{Total Revenue}}{\text{Total Number of Customers}}ARPU=Total Number of CustomersTotal Revenue​
   * In your query, ARPU is calculated for each month in the monthly\_revenue CTE:

sql

Copy code

ROUND (SUM(bs\_revenue) / COUNT(DISTINCT phone\_number), 2) AS arpu

1. **Churn Rate**:
   * **Churn Rate** is the percentage of customers who stop doing business with a company over a given period. It helps assess how many customers leave or “churn” each month.
   * **Formula**: Churn Rate=Customers LostTotal Customers at the Start of the Period×100\text{Churn Rate} = \frac{\text{Customers Lost}}{\text{Total Customers at the Start of the Period}} \times 100Churn Rate=Total Customers at the Start of the PeriodCustomers Lost​×100
   * In your query, churn rate is calculated by comparing the number of customers in a given month (customers\_current\_month) to those who remain in the next month (customers\_next\_month):

sql

Copy code

ROUND((COUNT(DISTINCT cm.phone\_number) - COUNT(DISTINCT nm.phone\_number)) \* 100.0 / COUNT(DISTINCT cm.phone\_number), 2) AS churn\_rate

1. **Customer Lifetime Value (CLV)**:
   * CLV estimates the total revenue a customer will generate over the time they remain a customer, considering the **ARPU** and the **churn rate**.
   * **Formula**: CLV=ARPUChurn Rate\text{CLV} = \frac{\text{ARPU}}{\text{Churn Rate}}CLV=Churn RateARPU​
   * A **higher CLV** means customers are more valuable and tend to stay longer, generating more revenue over their lifetime.
   * The formula in your query checks if the churn rate is greater than 0 to avoid division by zero:

sql

Copy code

CASE

WHEN cr.churn\_rate > 0 THEN ROUND(mr.arpu / (cr.churn\_rate / 100), 2)

ELSE NULL

END AS clv

**Step-by-Step Breakdown of the Query**

1. **customers\_current\_month CTE**:
   * This step collects all the customers who made at least one transaction each month (grouped by month).
   * It ensures that the distinct customers and their associated month numbers are available for the calculations.

sql

Copy code

SELECT phone\_number, month\_number, month\_3\_letter

FROM bambaswap\_consolidated\_jan\_may\_2024

GROUP BY phone\_number, month\_number, month\_3\_letter

1. **customers\_next\_month CTE**:
   * This step looks at customers who made a transaction in the next month. It helps determine which customers have **churned** by checking whether they are missing in the next month’s data.

sql

Copy code

SELECT phone\_number, month\_number AS next\_month\_number

FROM bambaswap\_consolidated\_jan\_may\_2024

GROUP BY phone\_number, month\_number

1. **monthly\_revenue CTE**:
   * Calculates the total revenue and ARPU for each month, giving an idea of how much revenue each customer generates on average.

sql

Copy code

SELECT month\_number, month\_3\_letter, SUM(bs\_revenue) AS total\_revenue,

COUNT(DISTINCT phone\_number) AS total\_customers,

ROUND(SUM(bs\_revenue) / COUNT(DISTINCT phone\_number), 2) AS arpu

FROM bambaswap\_consolidated\_jan\_may\_2024

GROUP BY month\_number, month\_3\_letter

1. **churn\_rate CTE**:
   * Calculates the churn rate by checking how many customers were lost between the current month and the next month. It’s crucial for understanding how frequently customers leave.

sql

Copy code

ROUND((COUNT(DISTINCT cm.phone\_number) - COUNT(DISTINCT nm.phone\_number)) \* 100.0 / COUNT(DISTINCT cm.phone\_number), 2) AS churn\_rate

1. **Final CLV Calculation**:
   * Joins the **ARPU** and **churn rate** to calculate the **Customer Lifetime Value**. It uses the formula **CLV = ARPU / Churn Rate** and rounds the result to 2 decimal places.

sql

Copy code

CASE

WHEN cr.churn\_rate > 0 THEN ROUND(mr.arpu / (cr.churn\_rate / 100), 2)

ELSE NULL

END AS clv

**What CLV Tells Us About Customers in Bambaswap:**

* **High CLV**: Indicates that the customers generate significant revenue and tend to remain customers for a long time. This suggests that customer retention strategies are effective.
* **Low CLV**: Suggests that customers either generate lower revenue or have a high churn rate, meaning they do not stay with the company for long periods.

By calculating CLV, **Bambaswap** can:

* **Adjust acquisition spending**: Knowing the CLV helps Bambaswap optimize its marketing spend to acquire customers without exceeding the expected lifetime value.
* **Focus on retention**: If churn is high and CLV is low, focusing on improving customer retention strategies can help improve profitability over time.
* **Understand customer behavior**: CLV gives insights into how much value a customer provides over time, allowing Bambaswap to tailor offers and communications based on this value.

Let me know if you'd like to adjust the query further or dive deeper into specific parts of the CLV calculation!