


Full Metrics Library

 This document is a work on progress

This document contains the details on how common metrics used across the business are calculated and used.

- Company-wide handy definitions
 - User
 - Customer
 - TTV
 - AOV
 - Net Transaction Margin (NTM)
- Company-wide handy metrics
 - Metrics
 - Login rate
 - Credit success rate
 - Approval rate
 - Risk success to completion rate
 - Acceptance rate
 - Completion rate
 - Default rate
- Metrics - more detailed
 - Order status definitions:
 - Order intent
 - Order assessment
 - Credit decision success
 - Order succeeded
 - Order complete
 - Order mature
 - Metrics actual calculation
- Department-specific definitions
 - Account Management
 - Collections
 - Creative
 - Customer Care
 - Finance
 - Integration
 - Legal
 - Marketing
 - Onboarding
 - Product & Technology
 - Retail ops
 - Risk
 - Sales
 - General

Company-wide handy definitions

User

- Any person with a Scalapay account, regardless of the status and if they have made orders
- Additionally, a good user is a user that has had at least 2 orders and has never been in default

Customer

- A Scalapay user with at least 1 order
- When looking at orders details, customers are categorized as:
 - New → if it's the customer's first order
 - Repeat other → if the customer has previously bought with Scalapay but from other merchants
 - Repeat same → if the customer has already bought with Scalapay from the same merchant at some point in history

TTV

- total transaction volume through the platform

- aggregated by the invoice created at date
- *How it's calculated: sum of order amounts*

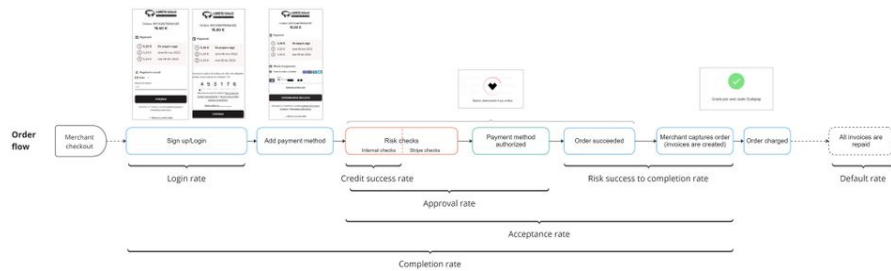
AOV [↗](#)

- average order value
- *How it's calculated: average of order amounts*

Net Transaction Margin (NTM) [↗](#)

- profit (or loss) based on unit economics of each order
- *How it's calculated: (our take rate) - (transaction costs)*

Company-wide handy metrics [↗](#)



Metrics [↗](#)

Login rate [↗](#)

users that successfully log in to Scalapay out of all users that attempt to do so within a given period of time.

Drop off reasons:

- Users do not want to enter phone number
- OTP is not sent
- OTP is not verified

How it's calculated: (users that trigger "login viewed") / (users that successfully verify the OTP)

Credit success rate [↗](#)

% of users that are pass our internal checks but still have to go through Stripe checks. It is therefore the approval rate of our internal credit decision engine, they can still be rejected by Stripe.

Drop off reasons:

- rejected by the credit decision algorithm
- rejected by one of our other internal rules: [Credit decision rejection reasons](#)

How it's calculated: (total number of orders passing our internal checks) / (number of users that generate "confirm button clicked")

Approval rate [↗](#)

% of users that pass both our internal checks and Stripe checks.

Drop off reasons:

- any reason in the credit success rate
- failed card authentication
- rejected by stripe rules Stripe Rules
 - full list of stripe rule block is available here: [Decline codes](#)

How it's calculated: (total number of orders passing our internal checks + Stripe checks) / (number of users that generate "confirm button clicked")

Risk success to completion rate [↗](#)

The rate of orders that go to completion after having successfully passed the risk checks.

Drop off reasons:

- merchant fails to capture the order

How it's calculated: (number of orders captured by the merchant) / (total number of orders passing all risk checks)

Acceptance rate [🔗](#)

% of orders that enter the risk assessment and go to completion out of all the orders that enter the risk assessment.

Drop off reasons:

- rejected by internal credit decision engine
- rejected by stripe rules
- failed card authentication
- merchant fails to capture the order

How it's calculated: $(\text{number of orders captured by the merchant}) / (\text{number of "confirm button clicked"})$

Completion rate [🔗](#)

% of completed orders since the beginning of the funnel, so out of all orders that start at the login screen.

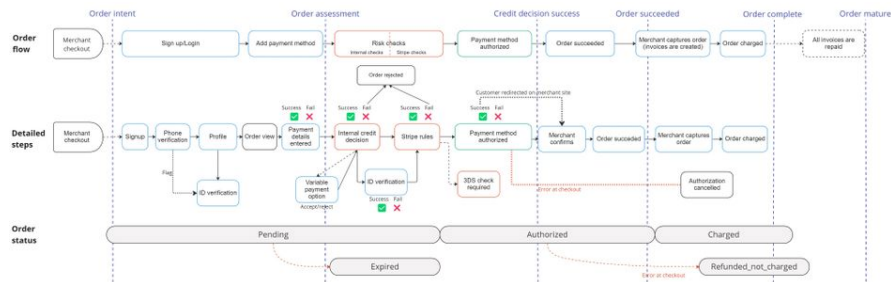
How it's calculated: $(\text{number of orders captured by the merchant}) / (\text{number of orders created})$

Default rate [🔗](#)

% of TTV not repaid after 12 days of delay since the original due date

- How it's calculated: $(\text{total transaction amount of invoices in the status default}) / (\text{total amount of TTV})$

Metrics - more detailed [🔗](#)



Order status definitions: [🔗](#)

Order intent [🔗](#)

- The customer chooses Scalapay at the checkout and an order is created with the orders API call
- It occurs at the time of order creation
- On the front end, the user sees the invoices amounts and due dates but has not confirmed the order yet

Order assessment [🔗](#)

- The customer clicks 'confirm order' button and the order enters the risk assessment
- It occurs at the time of order creation

Credit decision success [🔗](#)

- the order has passed through the risk decisioning process to determine the risk of the order and has 'approved' it
- occurs at the time of order creation

Order succeeded [🔗](#)

- The order has passed through all risk checks and is completed on our side
- the merchant still has to capture the order and the customer still has to be charged

Order complete [🔗](#)

- the merchant captures the order and the customer is charged
- occurs when the merchant captures the order (for delayed capture merchants this can be up to 7 days after the order is placed)

Order mature [🔗](#)

- The order has been fully repaid or refunded or is in default

Metrics actual calculation [🔗](#)

Login rate = $(\text{number of users that view the login page}) / (\text{number of users that successfully verify the OTP})$

SQL code snippet

```
1 -- order base & from the checkout
2 with login_viewed as(
3     select event_time,
4            order_id as order_login_viewed
5     from analytics.common_events_partitioned
6     where event_name_lowercase = 'login viewed'
7 ),
8 otp_verified as (
9     select event_time,
10            order_id as order_otp_verified
11     from analytics.common_events_partitioned
12     where event_name_lowercase = 'otp verified'
13 )
14 select login_viewed.event_time,
15        login_viewed.order_login_viewed,
16        otp_verified.order_otp_verified
17 from login_viewed
18 left join otp_verified on login_viewed.order_login_viewed = otp_verified.order_otp_verified
```

We still rely mainly on Amplitude for this metric

Credit success rate = (total number of our credit decision success) / (number of "confirm button clicked" [order assessment])

SQL code snippet

```
1 -- order based
2 select count(distinct case when foa.is_approved = 1 then foa.id
3            when foa.is_declined = 1 and dr.is_primaryapp_decline = 0 then foa.id end)*1.0 / count(distinct foa.id)
4 from warehouse.fact_order_assessment foa
5 left join warehouse.dim_decline_reason dr
6 on foa.decline_reason_id = dr.id
7
8 -- deduping by users email
9 select count(distinct case when foa.is_approved = 1 then foa.inferred_order_id
10            when foa.is_declined = 1 and dr.is_primaryapp_decline = 0 then foa.inferred_order_id end)*1.0 / count(distinct foa.inferred_order_id)
11 from warehouse.fact_order_assessment foa
12 left join warehouse.dim_decline_reason dr
13 on foa.decline_reason_id = dr.id
14
15 -- by ttv
16 select sum(distinct case when foa.is_approved = 1 then foa.order_amount
17            when foa.is_declined = 1 and dr.is_primaryapp_decline = 0 then foa.order_amount end)*1.0 / sum(distinct foa.order_amount)
18 from warehouse.fact_order_assessment foa
19 left join warehouse.dim_decline_reason dr
20 on foa.decline_reason_id = dr.id
```

Approval rate = (total number of credit decision success [order assessment where is_approved = 1]) / (number of "confirm button clicked" [order assessment])

SQL code snippet

```
1 -- order based
2 select count(distinct case when foa.is_approved = 1 then foa.id end)*1.0 / count(distinct foa.id)
3 from warehouse.fact_order_assessment foa
4
5 -- deduping by users email
6 select count(distinct case when foa.is_approved = 1 then foa.inferred_order_id end)*1.0 / count(distinct foa.inferred_order_id)
7 from warehouse.fact_order_assessment foa
8
9 -- by ttv
10 select sum(distinct case when foa.is_approved = 1 then foa.order_amount end)*1.0 / sum(distinct foa.order_amount)
11 from warehouse.fact_order_assessment foa
```

Risk success to completion rate = (number of orders captured by the merchant [order complete]) / (number of credit decision success [order assessment where is_approved = 1])

SQL code snippet

```
1 -- order based
2 select count(distinct foc.id)*1.0 / count(distinct case when foa.is_approved = 1 then foa.id end)
3 from warehouse.fact_order_assessment foa
4 left join warehouse.fact_order_complete foc on foa.id = foc.id
5
6 -- deduping by users email
7 select count(distinct foc.inferred_order_id)*1.0 / count(distinct case when foa.is_approved = 1 then foa.inferred_order_id end)
8 from warehouse.fact_order_assessment foa
9 left join warehouse.fact_order_complete foc on foa.id = foc.id
10
```

```

11 -- by ttv
12 select sum(foc.order_amount)*1.0 / sum(case when foa.is_approved = 1 then foa.order_amount end)
13 from warehouse.fact_order_assessment foa
14 left join warehouse.fact_order_complete foc on foa.id = foc.id

```

Acceptance rate = (number of orders captured by the merchant [order complete]) / (number of "confirm button clicked" [order assessment])

SQL code snippet

```

1 -- order based
2 select count(distinct foc.id)*1.0 / count(distinct foa.id)
3 from warehouse.fact_order_assessment foa
4 left join warehouse.fact_order_complete foc on foa.id = foc.id
5
6 -- deduping by users email
7 select count(distinct foc.inferred_order_id)*1.0 / count(distinct foa.inferred_order_id)
8 from warehouse.fact_order_assessment foa
9 left join warehouse.fact_order_complete foc on foa.id = foc.id
10
11 -- by ttv
12 select sum(foc.order_amount)*1.0 / sum(foa.order_amount)
13 from warehouse.fact_order_assessment foa
14 left join warehouse.fact_order_complete foc on foa.id = foc.id

```

Completion rate = (number of orders captured by the merchant [order complete]) / (number of orders created [order intent])


SQL code snippet

```

1 -- order based
2 select count(distinct foc.id)*1.0 / count(distinct foi.id)
3 from warehouse.fact_order_intent foi
4 left join warehouse.fact_order_complete foc on foi.id = foc.id
5
6 -- deduping by users email
7 select count(distinct foc.inferred_order_id)*1.0 / count(distinct foi.inferred_order_id)
8 from warehouse.fact_order_intent foi
9 left join warehouse.fact_order_complete foc on foi.id = foc.id
10
11 -- by ttv
12 select sum(foc.order_amount)*1.0 / sum(foi.order_amount)
13 from warehouse.fact_order_intent foi
14 left join warehouse.fact_order_complete foc on foi.id = foc.id

```

Department-specific definitions

 This part is intended to be filled with teams' feedbacks

In case you would like a metric to be added in this document, please raise a ticket in [this board](#) specifying the team that uses it, the metric name and the metric definition.

Account Management [↗](#)

Collections [↗](#)

Creative [↗](#)

Customer Care [↗](#)

Finance [↗](#)

Integration [↗](#)

Legal [↗](#)

Marketing [↗](#)

Onboarding [↗](#)

Product & Technology [↗](#)

Product conversion = % of users showing interest in proceeding with an order. It covers events from `Order viewed` to `Confirm button clicked`.

Merchant referral = users clicking on a merchant (`Merchant clicked`).

Referral conversion = % of users clicking on a merchant from the homepage. It covers events from `Homepage viewed` to `Merchant clicked`.

Homepage CR = % of users completing an order after clicking on a merchant from the homepage. It covers events from `Homepage viewed` to `Merchant clicked` to `Order succeeded`.

Shop directory CR = % of users completing an order clicking on a merchant from the shop directory. It covers events from `Shop directory viewed` to `Merchant clicked` to `Order succeeded`.

Retail ops [↗](#)

Risk [↗](#)

Rolling X day default rate = total transaction amount of invoices in the status default divided by the total amount of TTV only looking at the last X number of days

Default rate pre collections = Total transaction amount of invoices at any time were in the status default divided by the total amount of TTV

Sales [↗](#)

General [↗](#)

Items

- total number of things in the carts, extracted from the order details provided by the merchants
- eg) a cart with 4x quantity of the same SKU has 4 items and 1 SKU

invoice pending default

- the due date of the invoice has passed and is still in a pending status

invoice paid late

- the invoice was paid (status = 'charged') after the due date

invoice default

- the invoice status is in the status 'default'
- occurs 12 days after the due date

invoice contractual default

- the invoice is still in default 60 days after the due date

invoice collections win

- the invoice was once in 'default' (finalDefault = 1 in the db) but the debt was recovered by the collections team and the status is now 'charged'

order mature default

- any of the three invoices went into a default status

order mature collections win

- all invoices successfully charged after 1 or more went into default and were recovered by collections

Country aggregations

- when aggregating by a country do so by the user's country, found in the users table in the db

Find all status definitions here: [📖 Status descriptions of key Scalapay data models](#)