L. Elaine Dazzio BSE Chapter 10: Project #1 Payment Funnel Analysis

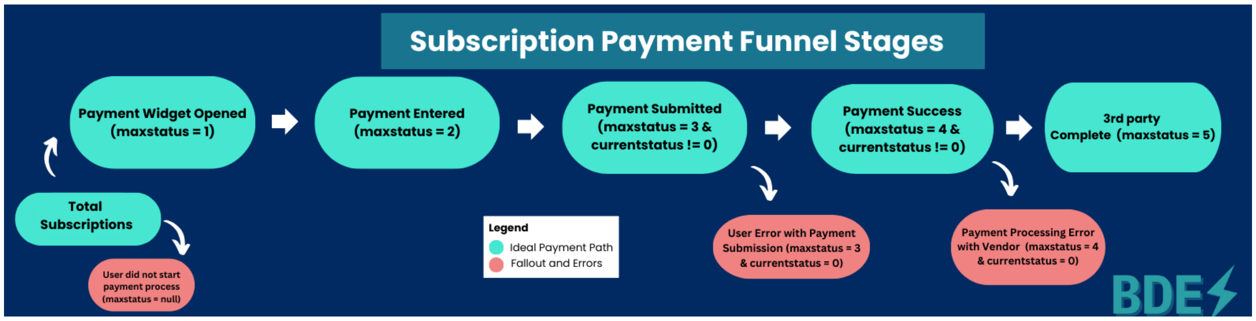
Project 1: Payment Funnel Analysis

Case Study: Payment Funnel Analysis

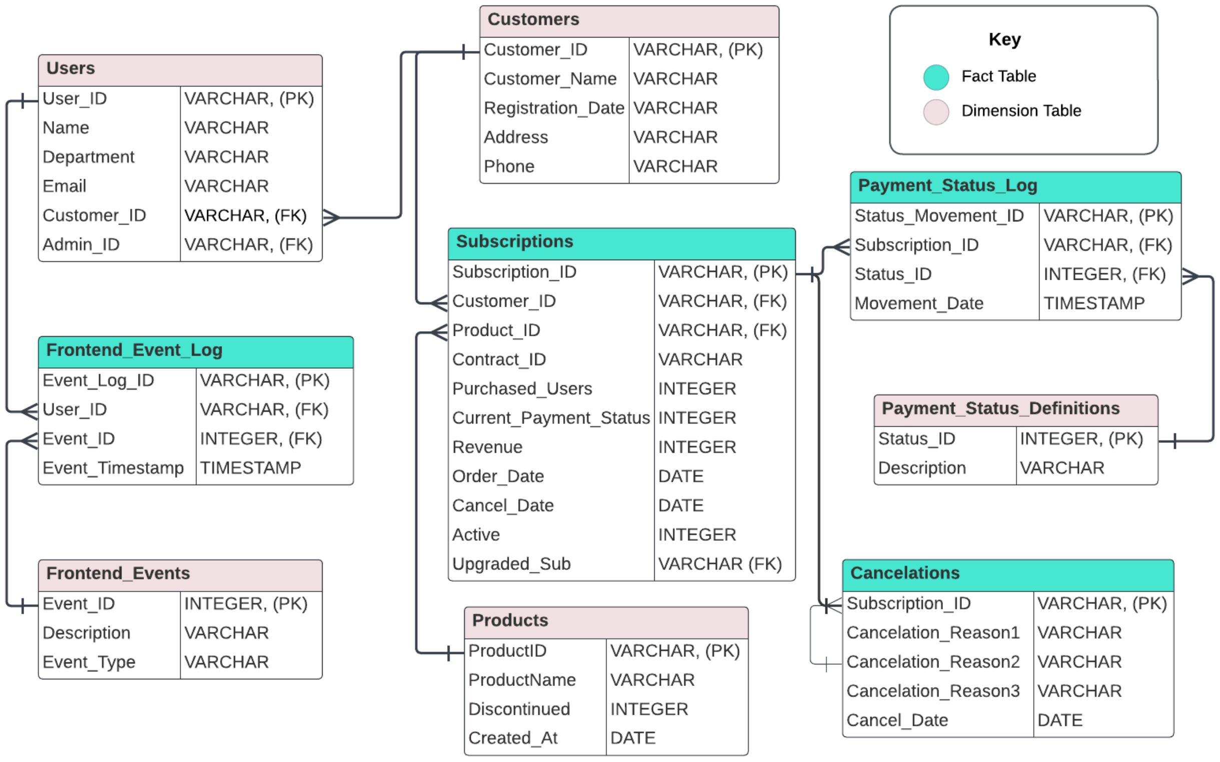
Your finance team comes to you one day asking about why there are so many unpaid subscriptions. Lately, customers have been choosing or opting into a paid subscription plan, but many are not completing the process by paying for their subscription. When customers sign up for a subscription, we consider them to officially be a customer, but they aren't considered "converted" into a paid plan until they actually pay for their subscription by completing the payment process. Because of this, the company has a less-than-desired conversion rate since many companies have started a subscription but haven't actually paid yet. This is a huge issue for the company because we have customers who are signing up for our product but aren't paying— which has resulted in a large loss in revenue.

As a seasoned data analyst, you know that the finance team's concerns are valid and worth looking into, so you immediately come up with a plan to dig into this. You meet with the product manager, and she walks you through the entire payment process. First, users have to open and enter the payment portal— and you already notice that this could be a large friction point for customers. Once inside the payment portal, they have to enter their credit card information and hit submit. It's possible for users to hit an error here if they input incorrect or incomplete information. Then the data is sent to a 3rd party payment processing company where the credit card is actually processed. It's also possible for users to hit an error here if the vendor has an issue processing the card. If everything is successfully completed with the vendor, they send the success message back to us, and we're able to log the transaction as complete on our side too.

After learning more about the business side of things and what the user sees on the frontend, you have to determine if we even have data to track all of these user events. If the data doesn't exist, you may have to measure proxies, brainstorm a workaround, and propose new user events to track in order to have better data collection for the future. Luckily, after meeting with your frontend engineer and data engineer, you learn that all of the major payment portal user events are tracked in the payment_status_log. You immediately start brainstorming ways to determine how to measure the success of each subscriptions, and more importantly, where the friction points are. Once you develop some insights, you'll be able to go back to the product manager with product recommendations to reduce friction and increase successful payments. This will have a large impact on revenue and get you noticed by the leadership team.


Subscription Payment Funnel Stages
Payment Widget Opened (maxstatus = 1)
Payment Entered (maxstatus = 2)
Payment Submitted (maxstatus = 3 & currentstatus != 0)
Payment Success (maxstatus = 4 & currentstatus != 0)
3rd party Complete (maxstatus = 5)
Total Subscriptions
User did not start payment process (maxstatus = null)
Legend
Ideal Payment Path
Fallout and Errors
User Error with Payment Submission (maxstatus = 3 & currentstatus = 0)
Payment Processing Error with Vendor (maxstatus = 4 & currentstatus = 0)
BDE

Big SQL Energy Main Data Model



Key
- ● Fact Table
- ● Dimension Table

Users	
User_ID	VARCHAR, (PK)
Name	VARCHAR
Department	VARCHAR
Email	VARCHAR
Customer_ID	VARCHAR, (FK)
Admin_ID	VARCHAR, (FK)

Customers	
Customer_ID	VARCHAR, (PK)
Customer_Name	VARCHAR
Registration_Date	VARCHAR
Address	VARCHAR
Phone	VARCHAR

Payment_Status_Log	
Status_Movement_ID	VARCHAR, (PK)
Subscription_ID	VARCHAR, (FK)
Status_ID	INTEGER, (FK)
Movement_Date	TIMESTAMP

Subscriptions	
Subscription_ID	VARCHAR, (PK)
Customer_ID	VARCHAR, (FK)
Product_ID	VARCHAR, (FK)
Contract_ID	VARCHAR
Purchased_Users	INTEGER
Current_Payment_Status	INTEGER
Revenue	INTEGER
Order_Date	DATE
Cancel_Date	DATE
Active	INTEGER
Upgraded_Sub	VARCHAR (FK)

Frontend_Event_Log	
Event_Log_ID	VARCHAR, (PK)
User_ID	VARCHAR, (FK)
Event_ID	INTEGER, (FK)
Event_Timestamp	TIMESTAMP

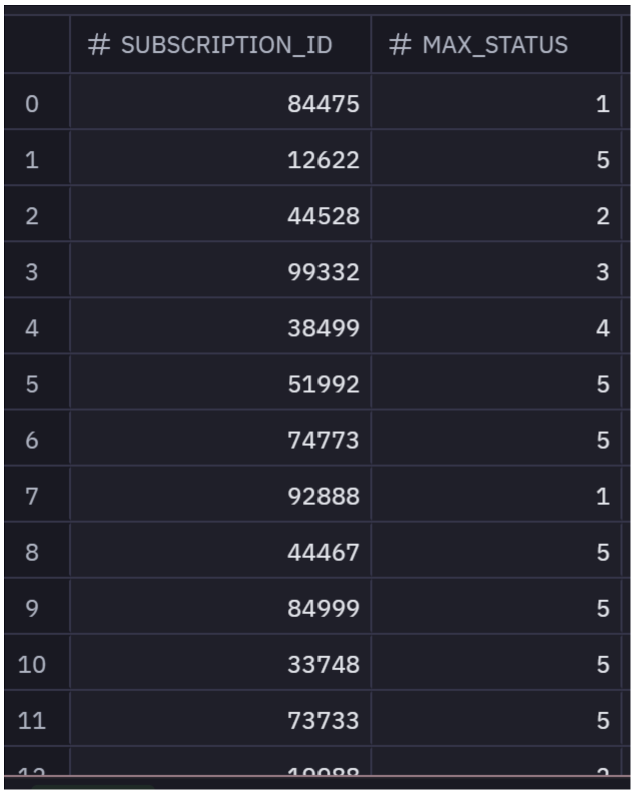
Payment_Status_Definitions	
Status_ID	INTEGER, (PK)
Description	VARCHAR

Frontend_Events	
Event_ID	INTEGER, (PK)
Description	VARCHAR
Event_Type	VARCHAR

Cancelations	
Subscription_ID	VARCHAR, (PK)
Cancelation_Reason1	VARCHAR
Cancelation_Reason2	VARCHAR
Cancelation_Reason3	VARCHAR
Cancel_Date	DATE

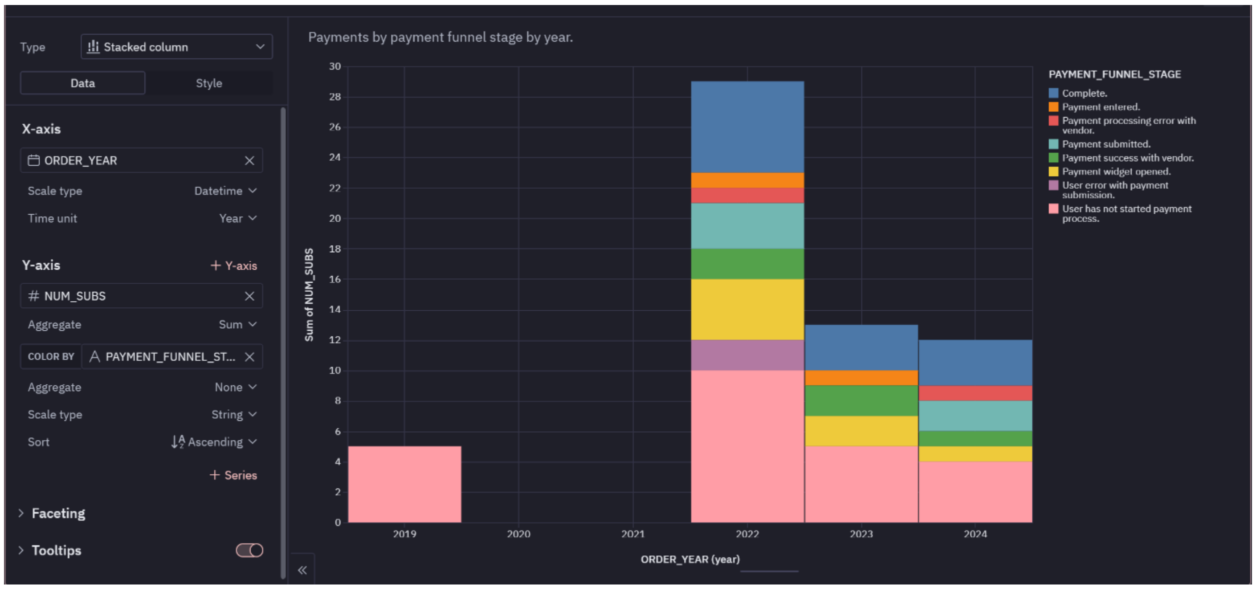
Products	
ProductID	VARCHAR, (PK)
ProductName	VARCHAR
Discontinued	INTEGER
Created_At	DATE

```
SELECT
    psl.subscription_id,
    MAX(psl.status_id) AS max_status
FROM
    public.payment_status_log psl
GROUP BY
    1
;
```



	# SUBSCRIPTION_ID	# MAX_STATUS
0	84475	1
1	12622	5
2	44528	2
3	99332	3
4	38499	4
5	51992	5
6	74773	5
7	92888	1
8	44467	5
9	84999	5
10	33748	5
11	73733	5

```
WITH max_status_reached AS (
    SELECT
    psl.subscription_id,
    MAX(psl.status_id) AS max_status
FROM
    public.payment_status_log psl
GROUP BY
    1
)
,
payment_funnel_stages AS(
SELECT
    subs.subscription_id,
    DATE_TRUNC('year', order_date) AS order_year,
    current_payment_status,
    max_status,
    CASE WHEN max_status = 1 THEN 'Payment widget opened.'
        WHEN max_status = 2 THEN 'Payment entered.'
        WHEN max_status = 3 AND current_payment_status = 0 THEN 'User error with payment
submission.'
        WHEN max_status = 3 AND current_payment_status != 0 THEN 'Payment submitted.'
        WHEN max_status = 4 AND current_payment_status = 0 THEN 'Payment processing error with
vendor.'
        WHEN max_status = 4 AND current_payment_status != 0 THEN 'Payment success with vendor.'
        WHEN max_status = 5 AND current_payment_status != 0 THEN 'Complete.'
        WHEN max_status IS NULL THEN 'User has not started payment process.'
        END AS payment_funnel_stage
FROM
    public.subscriptions subs
LEFT JOIN
    max_status_reached m
    on subs.subscription_id = m.subscription_id
)
SELECT
    payment_funnel_stage,
    order_year,
    COUNT(*) AS num_subs
FROM
    payment_funnel_stages
GROUP BY
    1, 2
ORDER BY
    2 DESC
;
```



Type Stacked column

Data Style

X-axis

ORDER_YEAR

Scale type Datetime

Time unit Year

Y-axis + Y-axis

NUM_SUBS

Aggregate Sum

COLOR BY PAYMENT_FUNNEL_ST...

Aggregate None

Scale type String

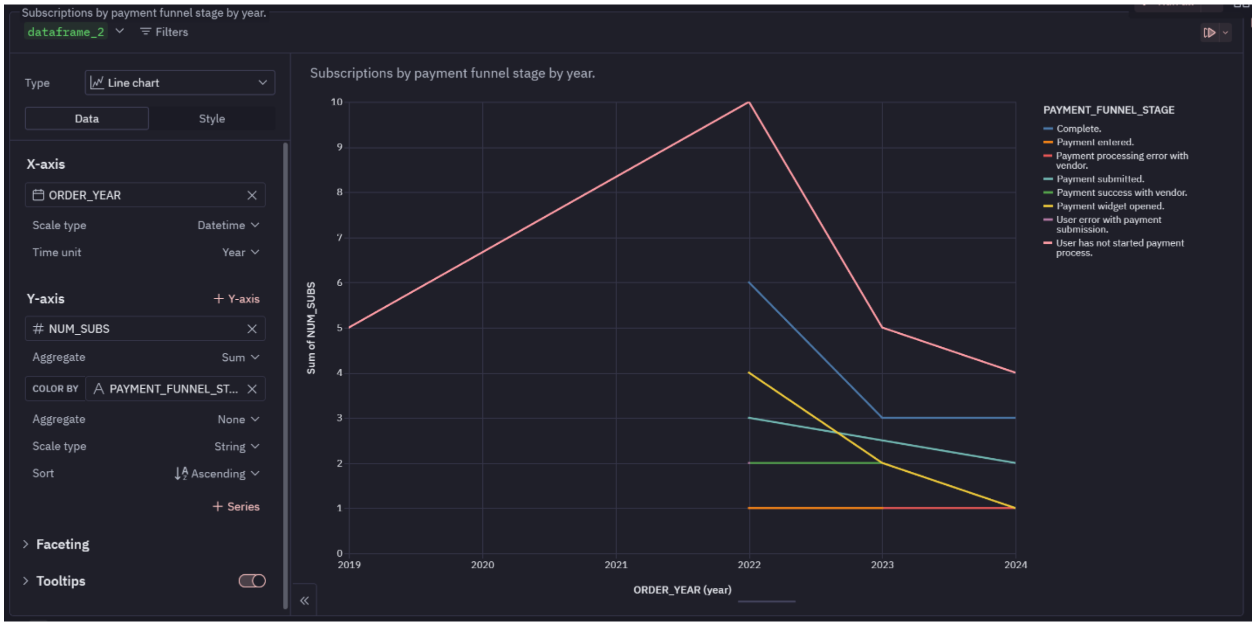
Sort Ascending

+ Series

Faceting

Tooltips

Payments by payment funnel stage by year.
Sum of NUM_SUBS
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
2019
2020
2021
2022
2023
2024
ORDER_YEAR (year)
PAYMENT_FUNNEL_STAGE
Complete.
Payment entered.
Payment processing error with vendor.
Payment submitted.
Payment success with vendor.
Payment widget opened.
User error with payment submission.
User has not started payment process.



Subscriptions by payment funnel stage by year.

dataframe_2 Filters

Type Line chart

Data Style

X-axis

ORDER_YEAR

Scale type Datetime

Time unit Year

Y-axis + Y-axis

NUM_SUBS

Aggregate Sum

COLOR BY PAYMENT_FUNNEL_ST...

Aggregate None

Scale type String

Sort Ascending

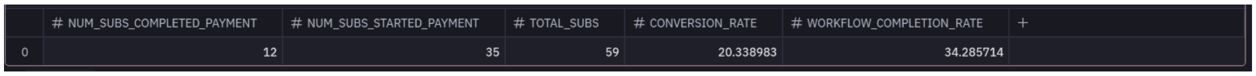
+ Series

Faceting

Tooltips

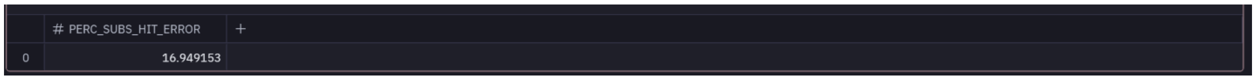
Subscriptions by payment funnel stage by year.
Sum of NUM_SUBS
0
1
2
3
4
5
6
7
8
9
10
2019
2020
2021
2022
2023
2024
ORDER_YEAR (year)
PAYMENT_FUNNEL_STAGE
Complete.
Payment entered.
Payment processing error with vendor.
Payment submitted.
Payment success with vendor.
Payment widget opened.
User error with payment submission.
User has not started payment process.

```
WITH max_status_reached AS (
    SELECT
    psl.subscription_id,
    MAX(psl.status_id) AS max_status
FROM
    public.payment_status_log psl
GROUP BY
    1
)
,
payment_funnel_stages AS(
SELECT
    subs.subscription_id,
    DATE_TRUNC('year', order_date) AS order_year,
    current_payment_status,
    max_status,
    CASE WHEN max_status = 5 THEN 1 ELSE 0 END AS completed_payment,
    CASE WHEN max_status IS NOT NULL THEN 1 ELSE 0 END AS started_payment
FROM
    public.subscriptions subs
LEFT JOIN
    max_status_reached m
    on subs.subscription_id = m.subscription_id
)
SELECT
    SUM(completed_payment) AS num_subs_completed_payment,
    SUM(started_payment) AS num_subs_started_payment,
    COUNT(*) AS total_subs,
    num_subs_completed_payment * 100/ total_subs AS conversion_rate,
    num_subs_completed_payment * 100/ num_subs_started_payment as workflow_completion_rate
FROM
    payment_funnel_stages
;
```



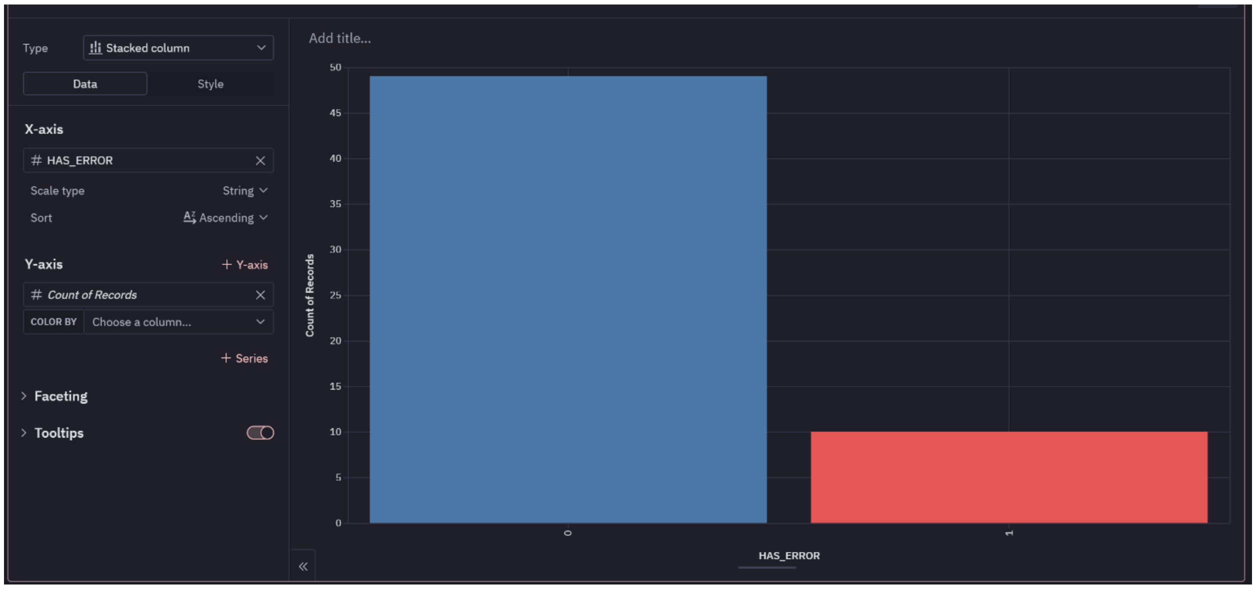
	# NUM_SUBS_COMPLETED_PAYMENT	# NUM_SUBS_STARTED_PAYMENT	# TOTAL_SUBS	# CONVERSION_RATE	# WORKFLOW_COMPLETION_RATE
0	12	35	59	20.338983	34.285714

```
WITH error_subs AS(
    SELECT
        DISTINCT subscription_id
    FROM
        public.payment_status_log
    WHERE
        status_id = 0
)
SELECT
    COUNT(err.subscription_id) * 100 / COUNT(subs.subscription_id) AS perc_subs_hit_error
FROM
    public.subscriptions subs
LEFT JOIN
    error_subs err
    ON subs.subscription_id = err.subscription_id
;
```



	# PERC_SUBS_HIT_ERROR
0	16.949153

```
WITH error_subs AS(
    SELECT
        DISTINCT subscription_id
    FROM
        public.payment_status_log
    WHERE
        status_id = 0
)
SELECT
    subs.subscription_id,
    CASE
        WHEN err.subscription_id IS NOT NULL THEN 1
        ELSE 0
        END AS has_error
FROM
    public.subscriptions subs
LEFT JOIN
    error_subs err
    ON subs.subscription_id = err.subscription_id
;
```

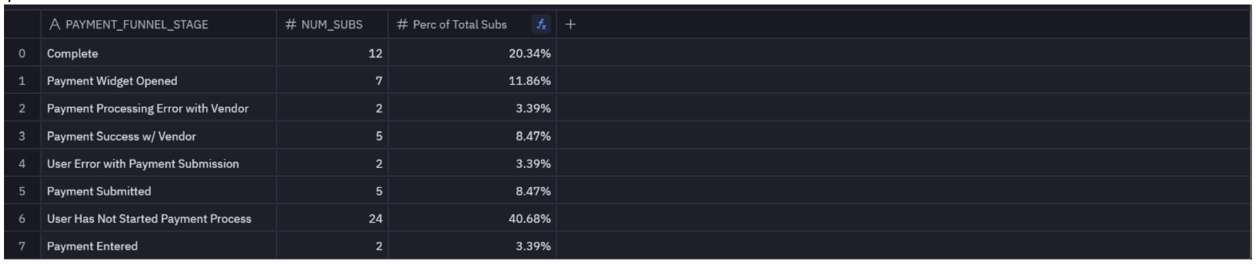

Type: Stacked column

X-axis: HAS_ERROR
Scale type: String
Sort: Ascending

Y-axis: Count of Records
COLOR BY: Choose a column...

```
-- Payment funnel stages tracking that uses a window function to calculate the current status
-- instead of the current_payment_status column in the subscriptions table

WITH max_status_reached AS (
SELECT
    subscription_id,
    MAX(status_id) AS max_status
FROM
    public.payment_status_log
GROUP BY
    1
)
,
subs_current_status AS(
SELECT
    subscription_id,
    status_id AS current_status,
    movement_date,
    ROW_NUMBER() OVER(PARTITION BY subscription_id ORDER BY movement_date DESC) AS
most_recent_status
FROM
    payment_status_log
QUALIFY
    most_recent_status = 1
)
,
payment_funnel_stages AS(
SELECT
    subs.subscription_id,
    DATE_TRUNC('year', order_date) AS order_year,
    current_status,
    max_status,
    CASE WHEN max_status = 1 THEN 'Payment Widget Opened'
        WHEN max_status = 2 THEN 'Payment Entered'
        WHEN max_status = 3 AND current_payment_status = 0 THEN 'User Error with Payment
Submission'
        WHEN max_status = 3 AND current_payment_status != 0 THEN 'Payment Submitted'
        WHEN max_status = 4 AND current_payment_status = 0 THEN 'Payment Processing Error with
Vendor'
        WHEN max_status = 4 AND current_payment_status != 0 THEN 'Payment Success w/ Vendor'
        WHEN max_status = 5 THEN 'Complete'
        WHEN max_status IS NULL THEN 'User Has Not Started Payment Process'
        END AS payment_funnel_stage
FROM
    subscriptions subs
LEFT JOIN
    max_status_reached m
    ON subs.subscription_id = m.subscription_id
LEFT JOIN
    subs_current_status curr
    ON subs.subscription_id = curr.subscription_id
)
SELECT
    payment_funnel_stage,
    COUNT(*) AS num_subs
FROM
    payment_funnel_stages
GROUP BY
    1
;
```



	PAYMENT_FUNNEL_STAGE	NUM_SUBS	Perc of Total Subs
0	Complete	12	20.34%
1	Payment Widget Opened	7	11.86%
2	Payment Processing Error with Vendor	2	3.39%
3	Payment Success w/ Vendor	5	8.47%
4	User Error with Payment Submission	2	3.39%
5	Payment Submitted	5	8.47%
6	User Has Not Started Payment Process	24	40.68%
7	Payment Entered	2	3.39%

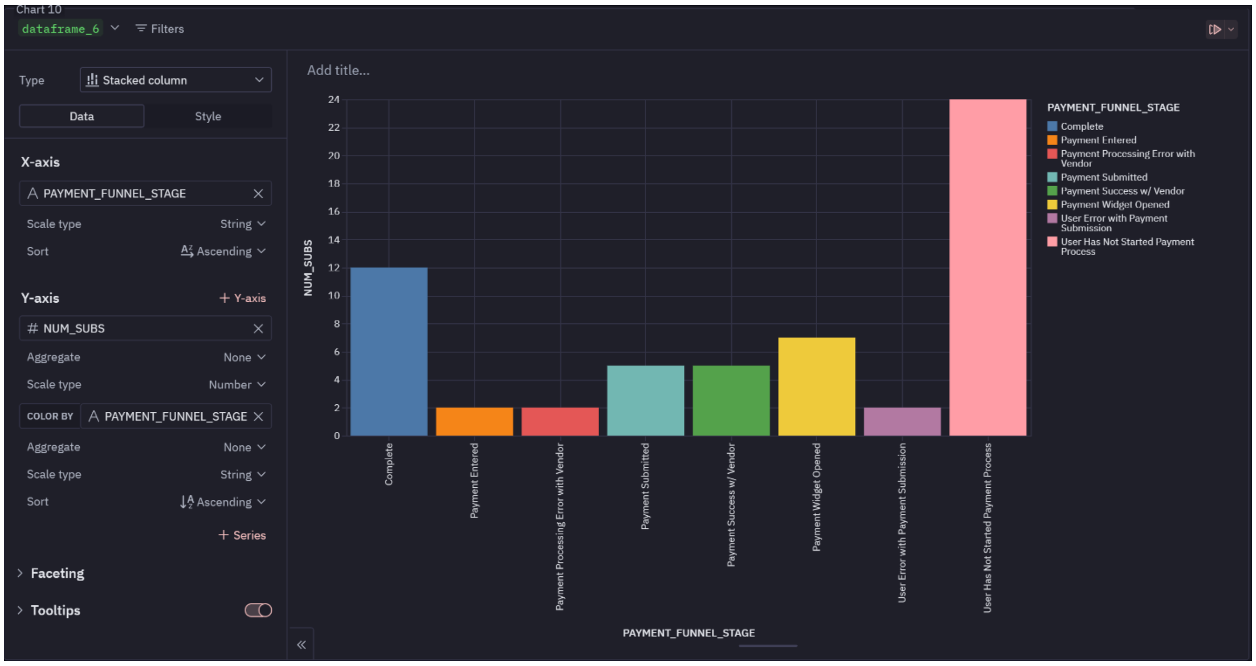


Chart 10
dataframe_6 Filters

Type: Stacked column
Data | Style

X-axis
PAYMENT_FUNNEL_STAGE
Scale type: String
Sort: Ascending

Y-axis + Y-axis
NUM_SUBS
Aggregate: None
Scale type: Number
COLOR BY: PAYMENT_FUNNEL_STAGE
Aggregate: None
Scale type: String
Sort: Ascending
+ Series

Faceting
Tooltips

Add title...
NUM_SUBS
0
2
4
6
8
10
12
14
16
18
20
22
24
Complete
Payment Entered
Payment Processing Error with Vendor
Payment Submitted
Payment Success w/ Vendor
Payment Widget Opened
User Error with Payment Submission
User Has Not Started Payment Process
PAYMENT_FUNNEL_STAGE
PAYMENT_FUNNEL_STAGE
Complete
Payment Entered
Payment Processing Error with Vendor
Payment Submitted
Payment Success w/ Vendor
Payment Widget Opened
User Error with Payment Submission
User Has Not Started Payment Process