**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.

A diagram of a payment

Description automatically generated

A diagram of a data model

Description automatically generated

SELECT

    psl.subscription\_id,

    MAX(psl.status\_id) AS max\_status

FROM

    public.payment\_status\_log psl

GROUP BY

    1

;

A screenshot of a cellphone

Description automatically generated

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

;

A graph with different colored squares

Description automatically generated

A graph on a black background

Description automatically generated

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,

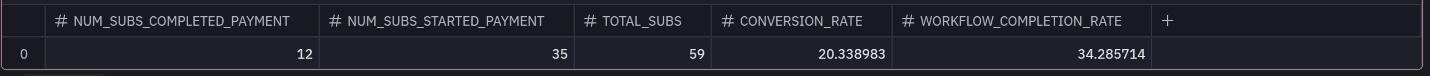
    num\_subs\_completed\_payment \* 100/ total\_subs AS conversion\_rate,

    num\_subs\_completed\_payment \* 100/ num\_subs\_started\_payment as workflow\_completion\_rate

FROM

    payment\_funnel\_stages

;



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

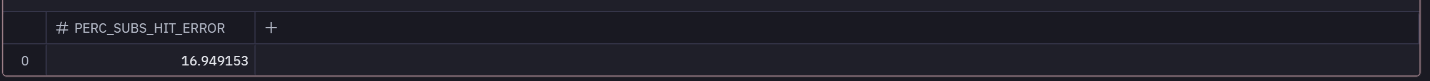
    public.subscriptions subs

LEFT JOIN

    error\_subs err

    ON subs.subscription\_id = err.subscription\_id

;



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

;

A screen shot of a graph

Description automatically generated

-- 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

;

A screenshot of a computer

Description automatically generated

A graph with different colored squares

Description automatically generated