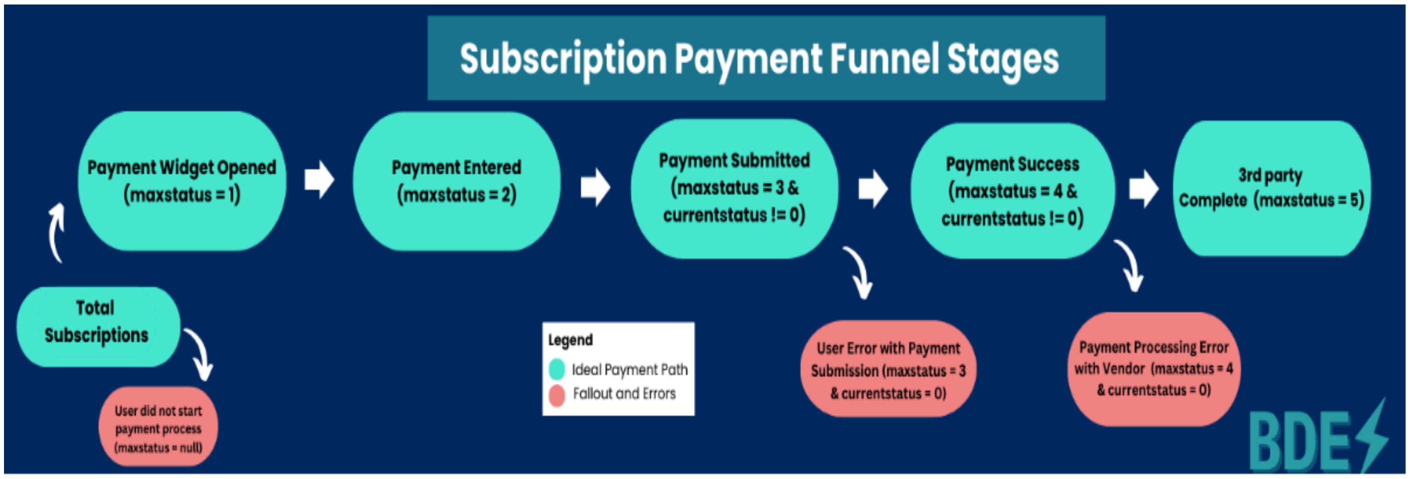
Payment Funnel Analysis Report

1. Executive Summary

This report analyzes the payment funnel to identify reasons for incomplete subscription payments. Through SQL-based data exploration, we assess user drop-off points, error frequencies, and conversion rates. Key findings indicate that user errors and payment processing failures are major bottlenecks. Recommendations include UI/UX improvements, better error handling, and enhanced follow-up strategies to improve payment completion rates.

2. Business Problem

The finance team has reported a high number of unpaid subscriptions, negatively impacting revenue. While users initiate the subscription process, many do not complete the payment. Understanding the pain points in the payment funnel is critical to improving conversion rates and minimizing revenue loss.


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

3. Skills Utilized

- **SQL Data Analysis**: CTEs, CASE, subqueries and window functions. Querying and aggregating data from payment logs.
- **Data Visualisation**
- **Data Interpretation**: Identifying user drop-off points and analyzing error frequencies.
- **Business Intelligence**: Translating data insights into actionable recommendations.
- **Problem-Solving**: Suggesting improvements to enhance payment completion rates.

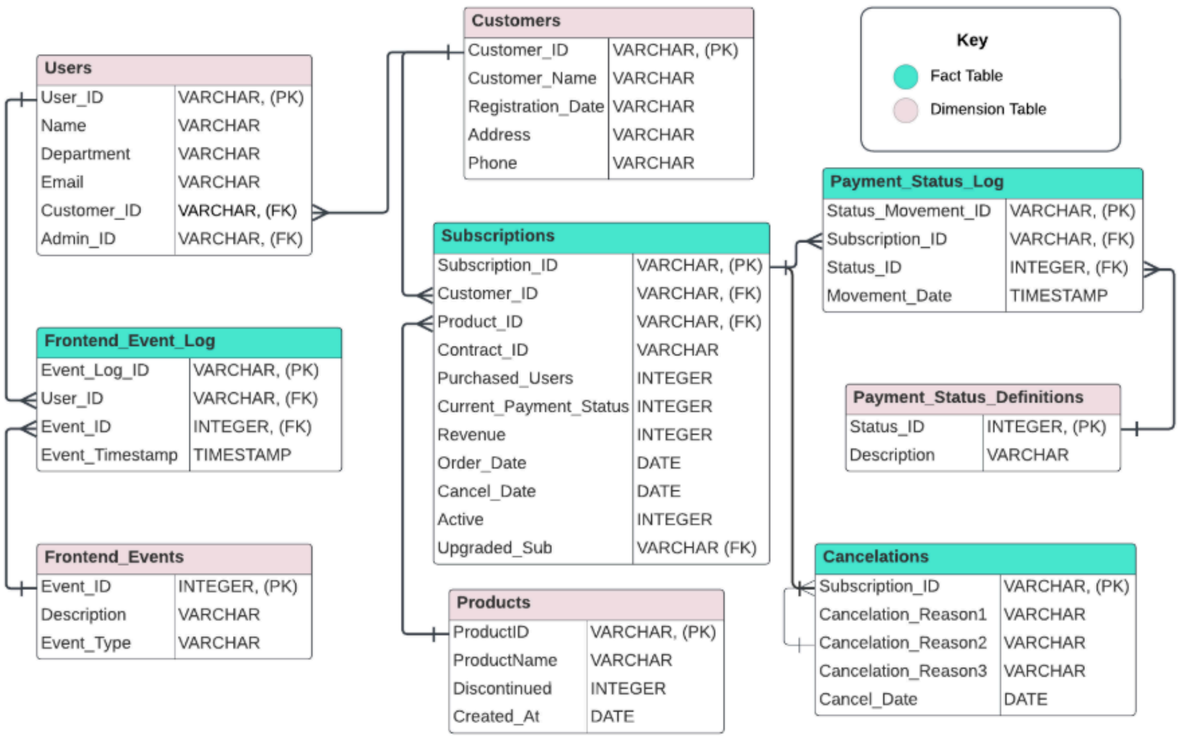
4. Data Exploration and Understanding

4.1 Dataset Overview

We analyzed two key tables:

- **payment_status_log**: Tracks different payment statuses for each subscription.
- **payment_status_definitions**: Provides definitions for each status ID in the payment process.

Main data model-



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

Products

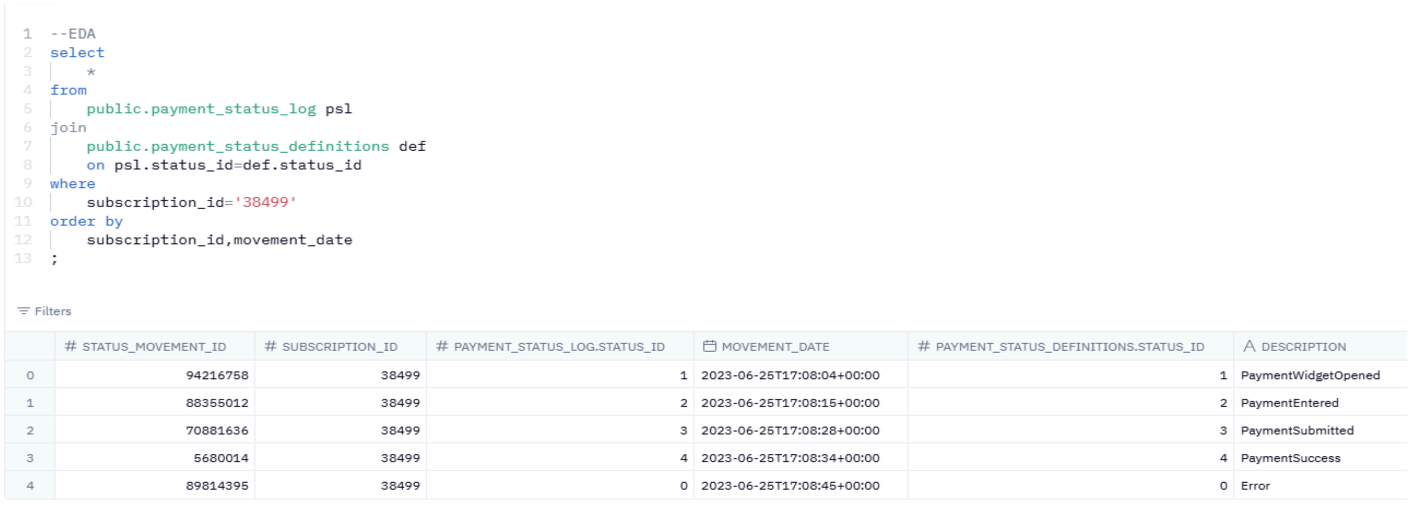
ProductID	VARCHAR, (PK)
ProductName	VARCHAR
Discontinued	INTEGER
Created_At	DATE

Cancelations

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

An initial exploration query was performed to track the payment journey for a sample subscription:

This provided insights into the sequence of payment status changes for individual subscriptions.



```
--EDA
select
    *
from
    public.payment_status_log psl
join
    public.payment_status_definitions def
    on psl.status_id=def.status_id
where
    subscription_id='38499'
order by
    subscription_id,movement_date
;
```

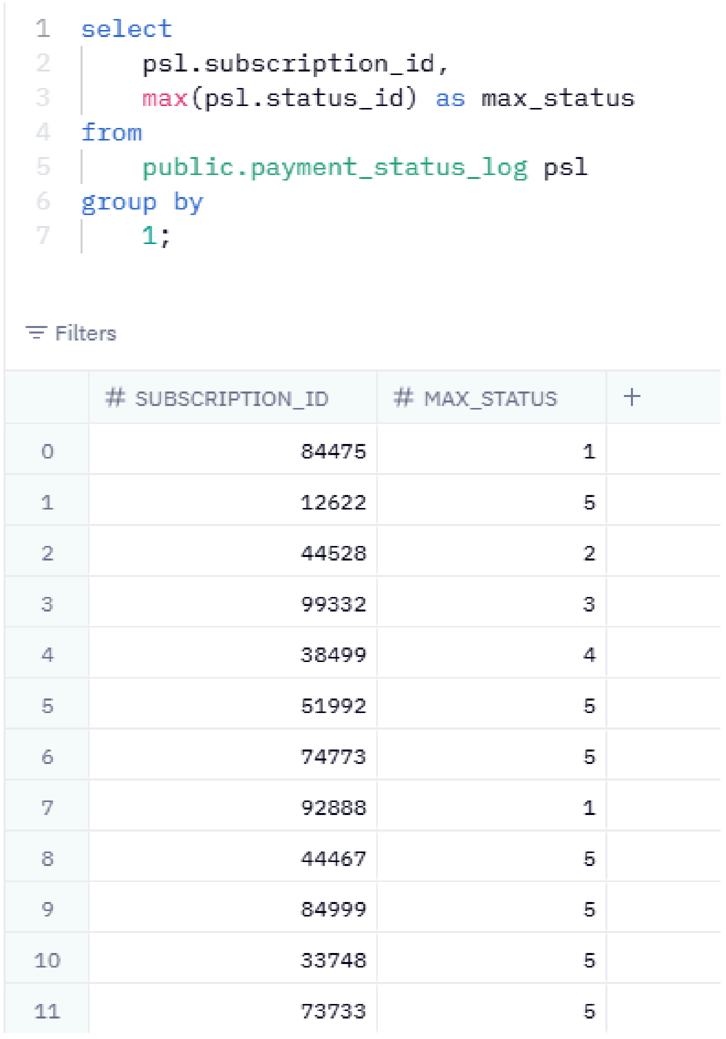
Filters

	# STATUS_MOVEMENT_ID	# SUBSCRIPTION_ID	# PAYMENT_STATUS_LOG.STATUS_ID	MOVEMENT_DATE	# PAYMENT_STATUS_DEFINITIONS.STATUS_ID	DESCRIPTION
0	94216758	38499	1	2023-06-25T17:08:04+00:00	1	PaymentWidgetOpened
1	88355012	38499	2	2023-06-25T17:08:15+00:00	2	PaymentEntered
2	70881636	38499	3	2023-06-25T17:08:28+00:00	3	PaymentSubmitted
3	5680014	38499	4	2023-06-25T17:08:34+00:00	4	PaymentSuccess
4	89814395	38499	0	2023-06-25T17:08:45+00:00	0	Error

5. Payment Funnel Analysis

5.1 Identifying Subscription Statuses

A query was run to determine the maximum status reached by each subscription:



```
select
    psl.subscription_id,
    max(psl.status_id) as max_status
from
    public.payment_status_log psl
group by
    1;
```

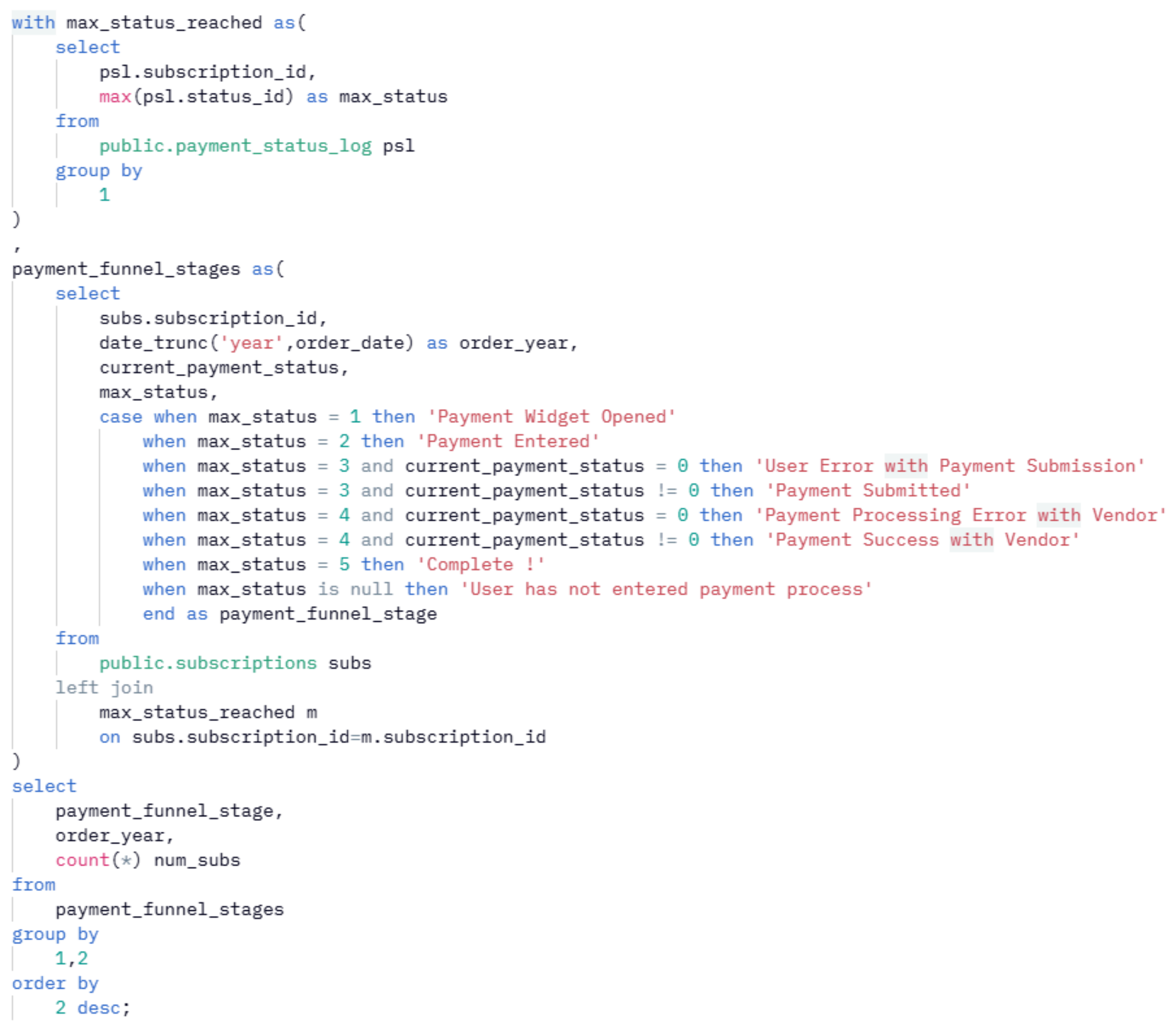
Filters

	# 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	

This helped categorize subscriptions based on how far they progressed in the payment funnel.

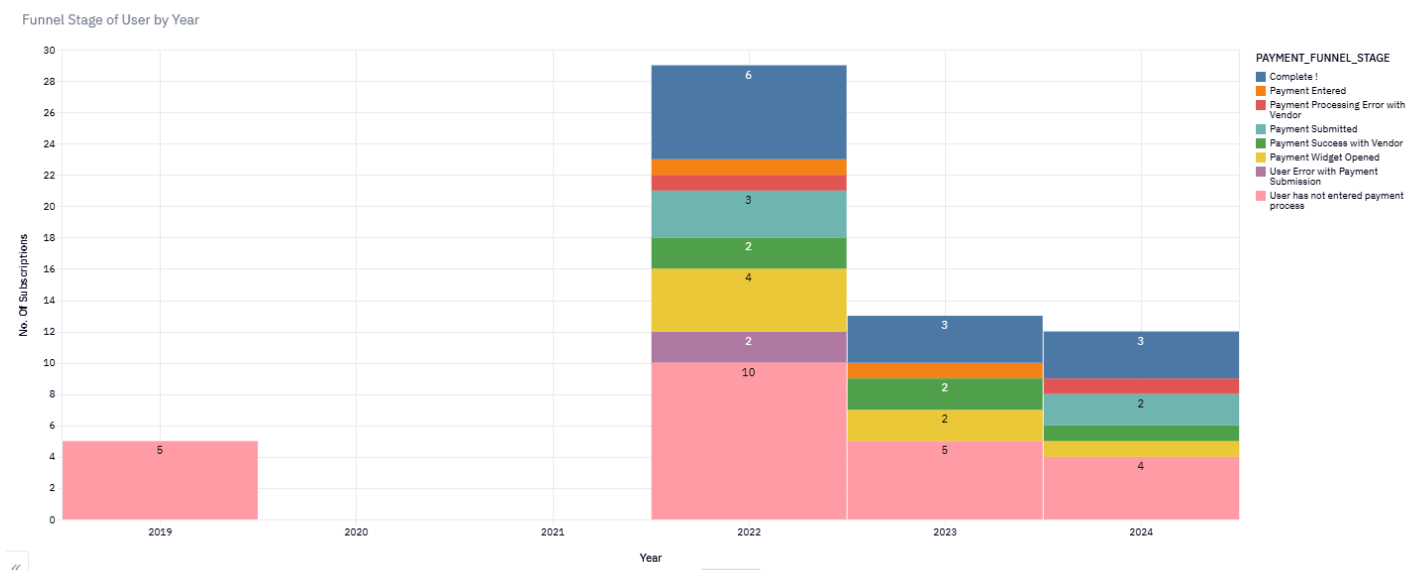
5.2 Breakdown of Funnel Stages

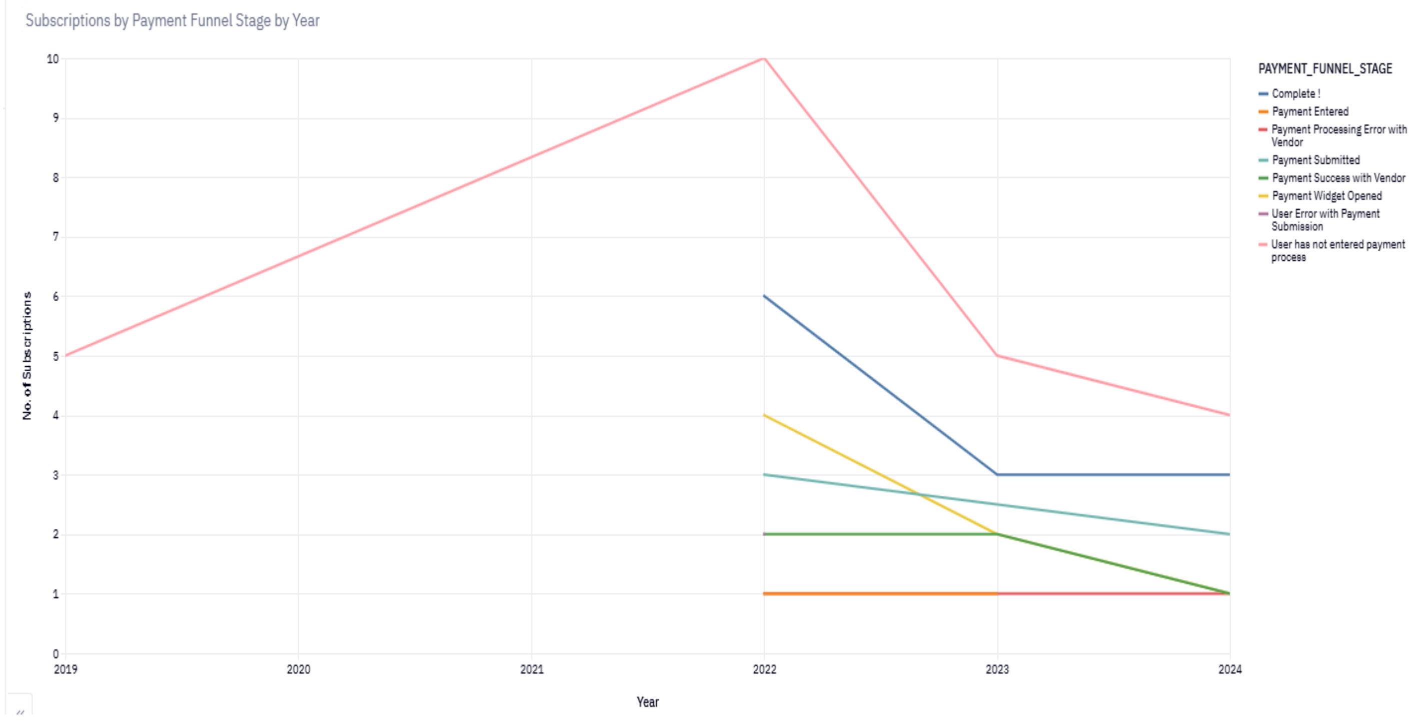
To track subscriptions through different stages of the payment funnel, a query was constructed:



```sql
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 then 'Complete !'
            when max_status is null then 'User has not entered 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(*) num_subs
from
    payment_funnel_stages
group by
    1,2
order by
    2 desc;
```

Output Visualised-

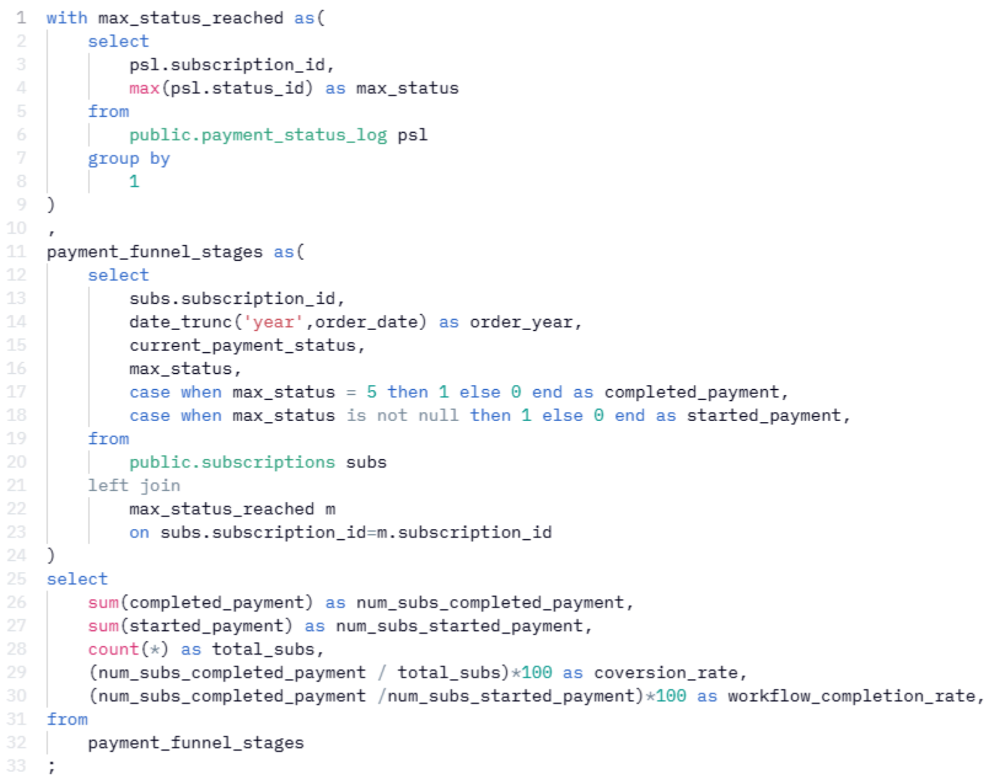




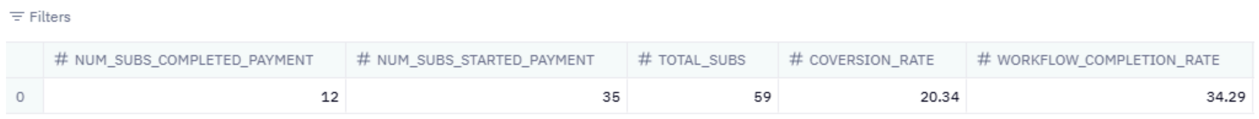
5.3 Conversion Rate and Workflow Completion Rate

We calculated:

- **Conversion Rate** = (Number of completed payments / Total subscriptions) * 100
- **Workflow Completion Rate** = (Number of completed payments / Number of users who started payment) * 100



```sql
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 / total_subs)*100 as coversion_rate,
    (num_subs_completed_payment /num_subs_started_payment)*100 as workflow_completion_rate,
from
    payment_funnel_stages
;
```

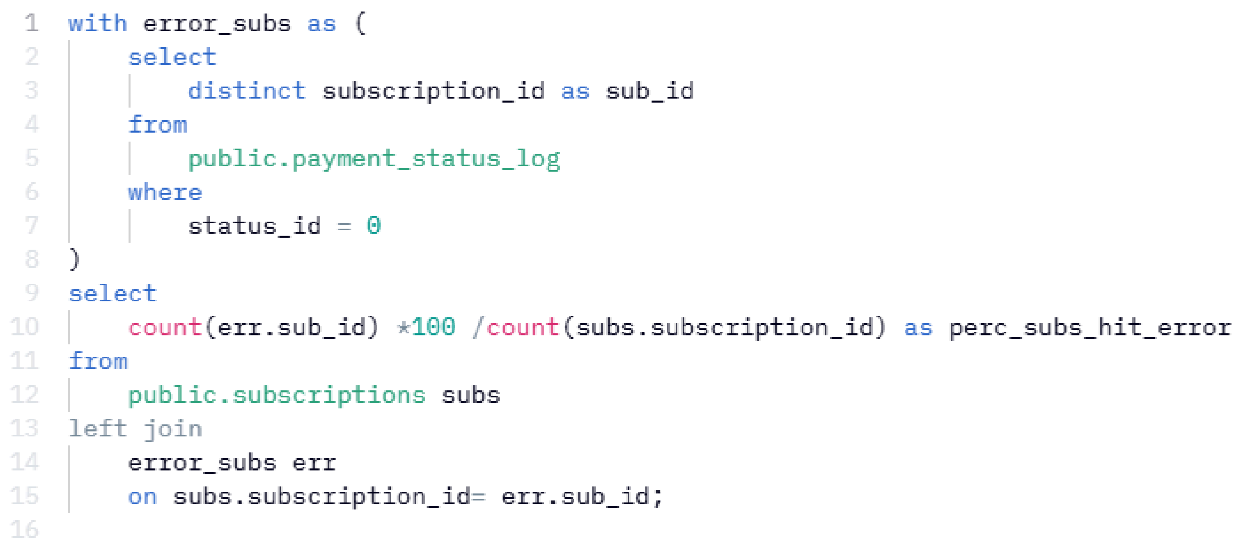


Filters

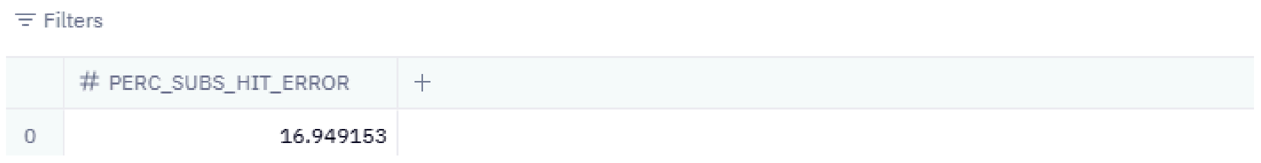
	# NUM_SUBS_COMPLETED_PAYMENT	# NUM_SUBS_STARTED_PAYMENT	# TOTAL_SUBS	# COVERSION_RATE	# WORKFLOW_COMPLETION_RATE
0	12	35	59	20.34	34.29

5.4 Error Frequency Analysis

To determine the proportion of subscriptions encountering errors:

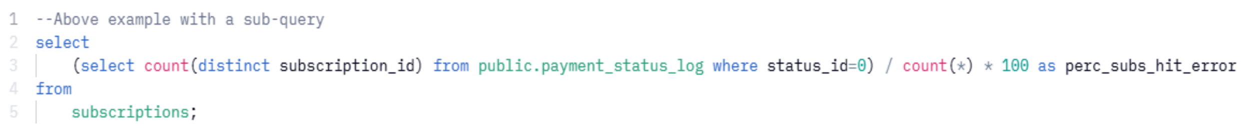


```sql
with error_subs as (
    select
        distinct subscription_id as sub_id
    from
        public.payment_status_log
    where
        status_id = 0
)
select
    count(err.sub_id) *100 /count(subs.subscription_id) as perc_subs_hit_error
from
    public.subscriptions subs
left join
    error_subs err
    on subs.subscription_id= err.sub_id;
```

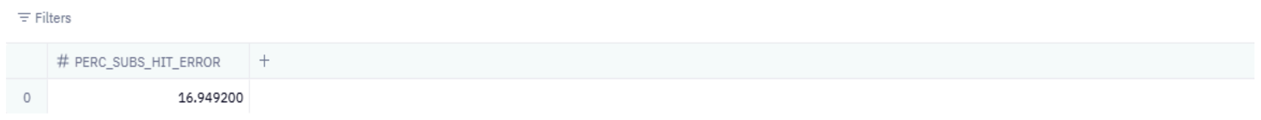


Filters

	# PERC_SUBS_HIT_ERROR
0	16.949153



```sql
--Above example with a sub-query
select
    (select count(distinct subscription_id) from public.payment_status_log where status_id=0) / count(*) * 100 as perc_subs_hit_error
from
    subscriptions;
```



Filters

	# PERC_SUBS_HIT_ERROR
0	16.949200

6. Key Findings

- A significant percentage of users start the payment process but do not complete it.
- The most common friction points include:
  - Users failing to enter payment details correctly (user errors).
  - Payment processing failures from the third-party vendor.
- The **error rate among users is 17% and conversion rate is 20%**, with a breakdown of user errors vs. vendor errors.
- The conversion rate and workflow completion rate metrics highlight the need for process optimization.

7. Recommendations

- **Improve UI/UX for the payment portal**:
  - Add real-time validation for credit card fields to prevent user errors.
  - Implement clear error messages and guidance for users encountering errors.
- **Optimize third-party payment processing**:
  - Investigate vendor-side failures and explore alternative payment processors.
  - Implement retry mechanisms for failed transactions.
- **User Follow-up Strategies**:
  - Send automated reminders to users who started payment but did not complete it.
  - Offer alternative payment methods to users facing issues.

8. Conclusion

By analyzing the payment funnel, we identified key bottlenecks in the subscription payment process. Addressing these friction points can significantly improve conversion rates, reduce revenue loss, and enhance user experience. The next steps involve collaborating with the product and engineering teams to implement these recommendations and measure the impact over time.