Payment funnel analysis with multiple CTEs

Business problem:

The product manager has requested a payment funnel analysis from the analytics team; she wants to understand what the furthest point in the payment process users are getting to and where users are falling out of the process. She wants to have full visibility into each possible stage of the payment process from the user's point of view.

Here's the payment process a user goes through when signing up for a subscription:

1. The user opens the widget to initiate payment process.
2. The user types in credit card information.
3. The user clicks the submit button to complete their part of the payment process.
4. The product sends the data to the third-party payment processing company.
5. The payment company completes the transaction and reports back with "complete."

This process is converted into `statusid`s using the following mapping in the `statusdefinition` table below:

```
select * from statusdefinition;
```

```
---------------------------------
| STATUSID | DESCRIPTION         |
---------------------------------
| 0        | Error               |
| 1        | PaymentWidgetOpened |
| 2        | PaymentEntered      |
| 3        | PaymentSubmitted    |
| 4        | PaymentSuccess      |
| 5        | Complete            |
---------------------------------
```

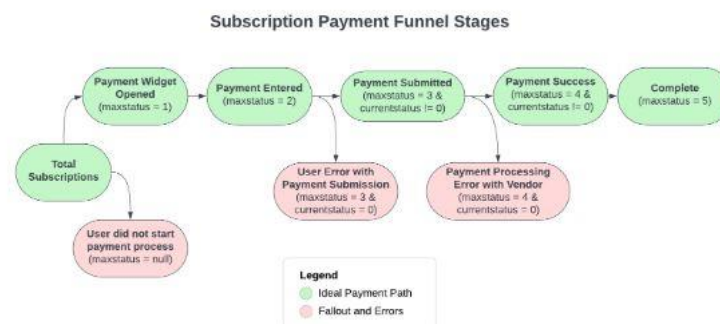To determine a subscription's `paymentfunnelstage`, we want to consider its **max statusid** because this will show us the **furthest** point in the workflow that they successfully reached—regardless of whether they:

- completed the process,
- hit an error and started the process over,
- or hit an error, gave up, and left the workflow.

In addition to the max status reached, we also want to consider if the subscription is **currently** stuck in an error using the `currentstatus` column from the `subscriptions` table.

If a user reaches `statusid = 3` but submits an incorrect card number, the transaction will be stopped and they will see an error message. The user will then need to restart the process and use the correct information. This is considered a **user error**. If a user reaches `statusid = 4` and submits correct payment information, the data is sent to a third-party payment company. If they are unable to process the data and complete the transaction due to an error on their end, it will produce an error message to the user. This is considered a **vendor error**.
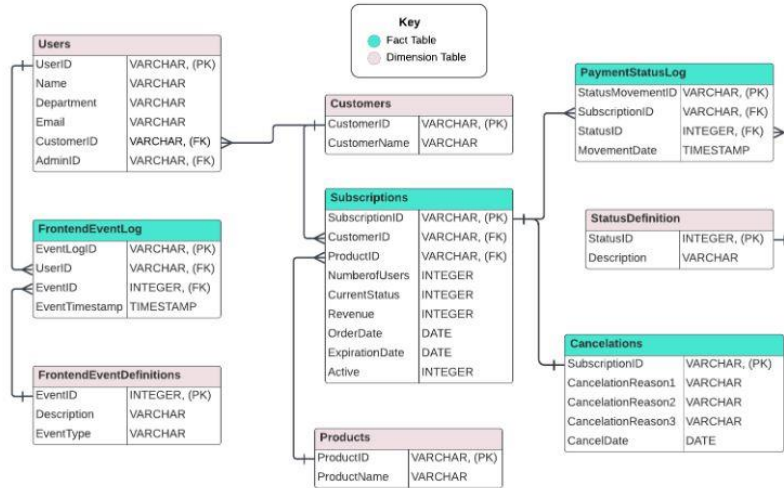
We want to differentiate between these two different user fallout reasons in our funnel analysis, because they point to different business actions to fix them. Here's a visualization of the logic needed:

Subscription Payment Funnel Stages

Task:

Count the number of subscriptions in each `paymentfunnelstage` as outlined in the code that I've given you by incorporating the the `maxstatus` reached and `currentstatus` per subscription. Use the `paymentstatuslog` and `subscriptions` tables.

Main Data Model

Key
- Fact Table
- Dimension Table

Users
UserID	VARCHAR, (PK)
Name	VARCHAR
Department	VARCHAR
Email	VARCHAR
CustomerID	VARCHAR, (FK)
AdminID	VARCHAR, (FK)

Customers
| CustomerID | VARCHAR, (PK) |
| CustomerName | VARCHAR |

PaymentStatusLog
StatusMovementID	VARCHAR, (PK)
SubscriptionID	VARCHAR, (FK)
StatusID	INTEGER, (FK)
MovementDate	TIMESTAMP

Subscriptions
SubscriptionID	VARCHAR, (PK)
CustomerID	VARCHAR, (FK)
ProductID	VARCHAR, (FK)
NumberofUsers	INTEGER
CurrentStatus	INTEGER
Revenue	INTEGER
OrderDate	DATE
ExpirationDate	DATE
Active	INTEGER

StatusDefinition
| StatusID | INTEGER, (PK) |
| Description | VARCHAR |

FrontendEventLog
EventLogID	VARCHAR, (PK)
UserID	VARCHAR, (FK)
EventID	INTEGER, (FK)
EventTimestamp	TIMESTAMP

Cancelations
SubscriptionID	VARCHAR, (PK)
CancelationReason1	VARCHAR
CancelationReason2	VARCHAR
CancelationReason3	VARCHAR
CancelDate	DATE

FrontendEventDefinitions
EventID	INTEGER, (PK)
Description	VARCHAR
EventType	VARCHAR

Products
| ProductID | VARCHAR, (PK) |
| ProductName | VARCHAR |

```sql
-- SQL code by John Uzoma
with subtable1 as (
    select
        subscriptionid,
        max(statusid) as maxstatus
    from paymentstatuslog
    group by subscriptionid
),

subtable2 as (
    select
        A.subscriptionid,
        case when B.maxstatus = 1 then 'PaymentWidgetOpened'
        when B.maxstatus = 2 then 'PaymentEntered'
        when B.maxstatus = 3 and A.currentstatus = 0 then 'User Error with Payment Submission'
        when B.maxstatus = 3 and A.currentstatus != 0 then 'Payment Submitted'
        when B.maxstatus = 4 and A.currentstatus = 0 then 'Payment Processing Error with Vendor'
        when B.maxstatus = 4 and A.currentstatus != 0 then 'Payment Success'
        when B.maxstatus = 5 then 'Complete'
        when B.maxstatus is null then 'User did not start payment process'
        end as paymentfunnelstage
    from subscriptions A
    LEFT JOIN subtable1 B
    on A.subscriptionid = B.subscriptionid
)

select
    paymentfunnelstage,
    count(subscriptionid) as SUBSCRIPTIONS
from subtable2
group by paymentfunnelstage
```

Result

PAYMENTFUNNELSTAGE	SUBSCRIPTIONS
Complete	12
Payment Processing Error with Vendor	1
Payment Submitted	1
Payment Success	1
PaymentEntered	2
PaymentWidgetOpened	7
User Error with Payment Submission	1
User did not start payment process	3