SQL CODE
WRITE-UP

PAYMENT FUNNEL ANALYSIS

JAN 2024

Case, Left Join & Nested CTE

LinkedIn Coding Challenge

Intermediate

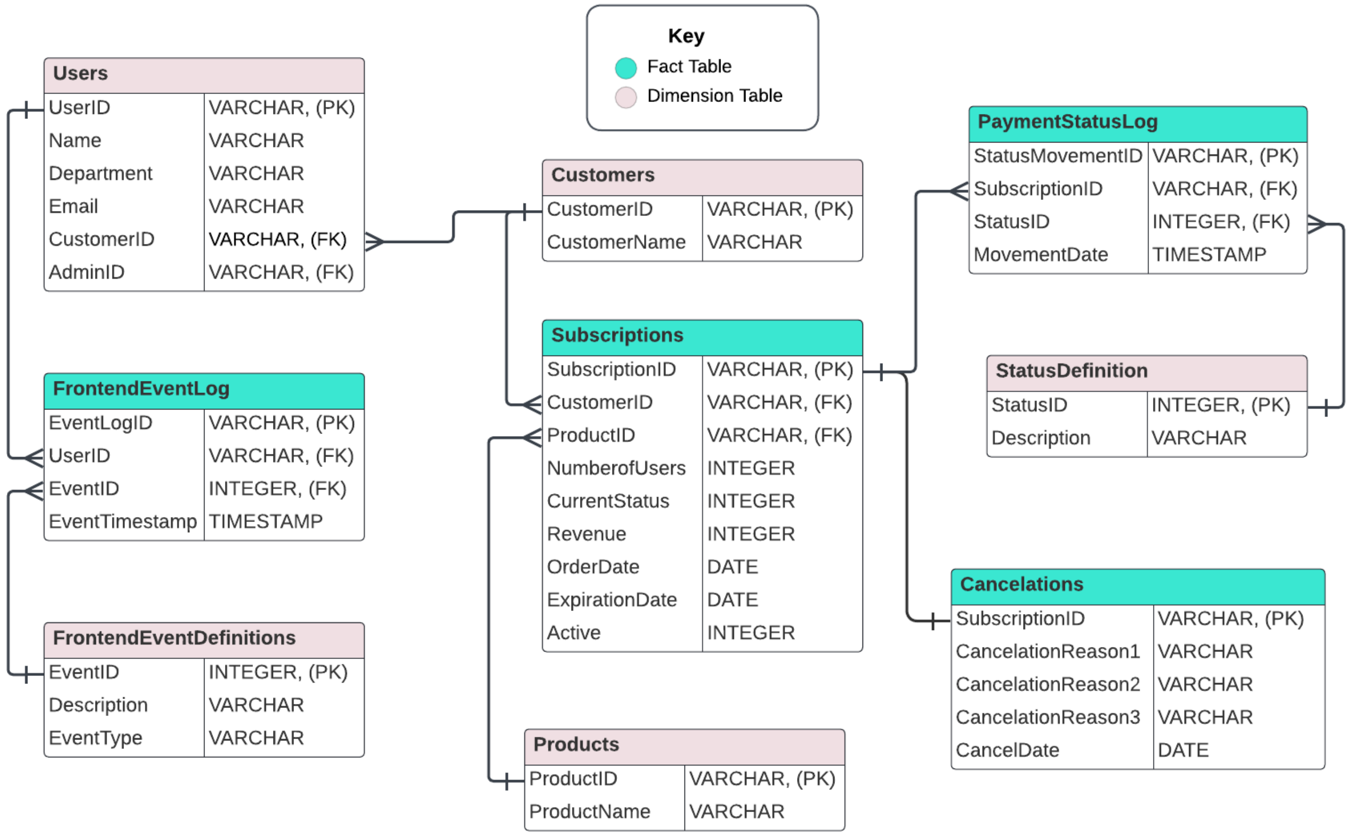
A product manager requested a payment funnel analysis to a) understand what the furthest point in the payment process users are getting, b) find out where users are falling out of the process and c) gain full visibility into each possible stage of the payment process from the user’s POV.

Count the number of subscriptions in each payment funnel stage

INPUT FORMAT

The main source tables are **PAYMENTSTATUSLOG** and **SUBSCRIPTIONS.** When queried using ***SELECT *,*** the **STATUSDEFINITION** table shows each stage of the payment funnel and its corresponding description while the diagram further below represents the logic needed for the query.

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

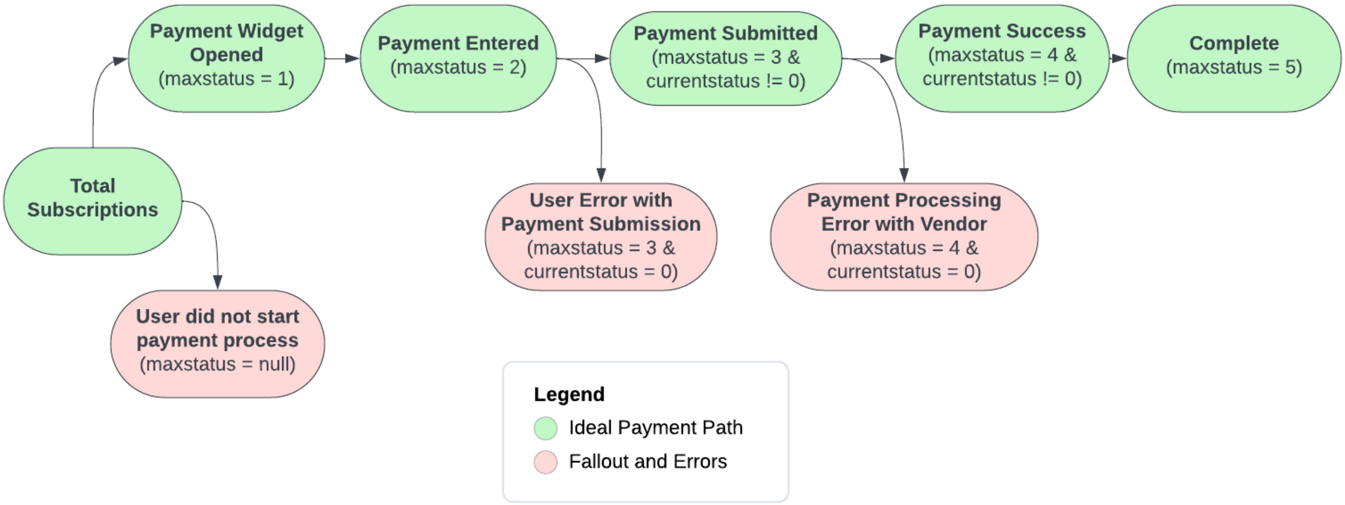
FrontendEventDefinitions	
EventID	INTEGER, (PK)
Description	VARCHAR
EventType	VARCHAR

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

Products	
ProductID	VARCHAR, (PK)
ProductName	VARCHAR

```
----------------------------------

| STATUSID | DESCRIPTION          |

----------------------------------

| 0        | Error                |

| 1        | PaymentWidgetOpened  |

| 2        | PaymentEntered       |

| 3        | PaymentSubmitted     |

| 4        | PaymentSuccess       |

| 5        | Complete             |

----------------------------------
```

Subscription Payment Funnel Stages


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

CODE SOLUTION

```
WITH funn_sheet AS


(WITH funneling AS
(SELECT
stat.SUBSCRIPTIONID,
max(STATUSID) AS maxstatus
FROM
paymentstatuslog stat
GROUP BY stat.SUBSCRIPTIONID)


SELECT
sub.SUBSCRIPTIONID,
CASE WHEN maxstatus = 1 THEN 'PaymentWidgetOpened'
        WHEN maxstatus = 2 THEN 'PaymentEntered'
        WHEN maxstatus = 3 AND currentstatus = 0 THEN 'User Error with Payment Submission'
        WHEN maxstatus = 3 AND currentstatus != 0 THEN 'Payment Submitted'
        WHEN maxstatus = 4 AND currentstatus = 0 THEN 'Payment Processing Error with Vendor'
        WHEN maxstatus = 4 AND currentstatus != 0 THEN 'Payment Success'
        WHEN maxstatus = 5 THEN 'Complete'
        WHEN maxstatus IS NULL THEN 'User did not start payment process'
        END AS paymentfunnelstage
FROM
subscriptions sub
LEFT JOIN funneling
ON funneling.SUBSCRIPTIONID = sub.SUBSCRIPTIONID
GROUP BY sub.SUBSCRIPTIONID)
```

```
SELECT
PAYMENTFUNNELSTAGE,
COUNT(SUBSCRIPTIONID) AS SUBSCRIPTIONS
FROM funn_sheet
GROUP BY PAYMENTFUNNELSTAGE
```

SOLUTION PROCESS

- Nested CTE function: This initial CTE labeled **FUNNELING** calculates the furthest point in a user's payment journey using MAX and GROUP BY. The second column is renamed as ***maxstatus*** in order to match the logic diagram above
- Case function: This function serves to define each ***maxstatus*** value in non-technical terms for relevant stakeholders, namely the product manager in this case, by returning the corresponding funnel stage in the logic diagram. For each stage with more than one condition, **AND** is used while **!=** signifies the operation "not equal to"
- Left Join function: The CASE function draws on ***maxstatus*** and ***currentstatus*** from the **FUNNELING** CTE and the **SUBSCRIPTIONS** table respectively. Because ***maxstatus*** contains null values, a LEFT JOIN is used as opposed to a standard JOIN
- Outer CTE function: This CTE labeled **FUNN_SHEET** returns the ***paymentfunnelstage*** for each subscription
- Select function: Selects ***paymentfunnelstage*** for each users and COUNTS each instance. and GROUPS BY ***paymentfunnelstage*** to determine how many users fall into each category. Alternative to the code above is COUNT (*)

OUTPUT

```
--------------------------------------------------------
| 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             |
--------------------------------------------------------
```