SQL Churn Project

Stephen Ellingson

August 6, 2021

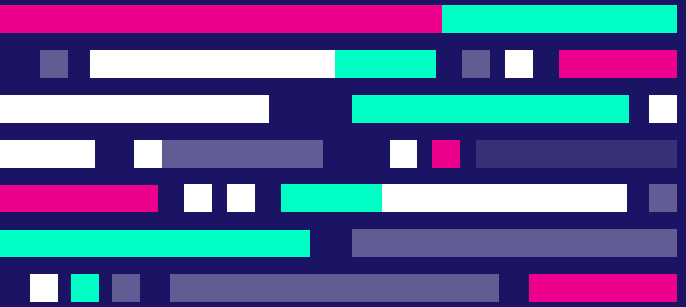
TABLE OF CONTENTS

INTRODUCTION

As part of the Data Analyst Certification Course offered through Codecademy, this SQL project tasks me with analyzing monthly subscription data to discover user churn rates for different subscriber segments.

01

INSPECTING DATA



WHAT DATA ARE WE DEALING WITH?

Before analysis, the subscription table provided must be inspected to determine what information is available. A query to inspect the first few rows of data will provide insight as to what I am working with.

CODE

```
SELECT *
FROM subscriptions
LIMIT 100;
```

RESULT

id	subscription_start	subscription_end	segment
1	2016-12-01	2017-02-01	87
2	2016-12-01	2017-01-24	87
3	2016-12-01	2017-03-07	87
4	2016-12-01	2017-02-12	87
5	2016-12-01	2017-03-09	87

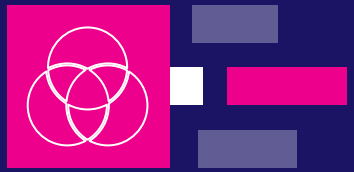
WHERE DO WE GO FROM HERE?

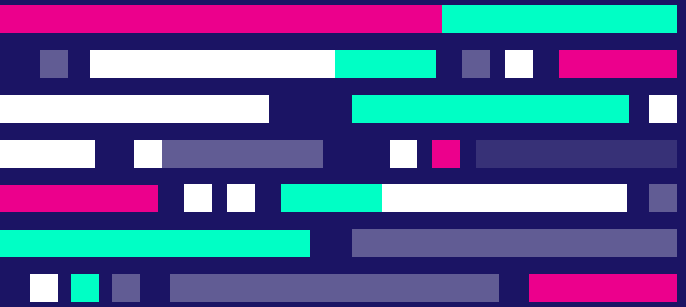
Find Date Range

The range of months should be identified in order to find monthly churn rates.

Identify Segments

From looking at the first 100 rows, there are only two segments: Segment 87 and Segment 30.





FINDING DATE RANGE

While it's clear that this subscription data covers a time period of many months, I can't currently tell how many months are present just by selecting the first few columns. To resolve this, using MIN() and MAX() will tell me how far my data spans.

CODE

```
SELECT MIN(subscription_start) AS start,
  MAX(subscription_start) AS end
FROM subscriptions;
```

RESULT

start	end
2016-12-01	2017-03-30

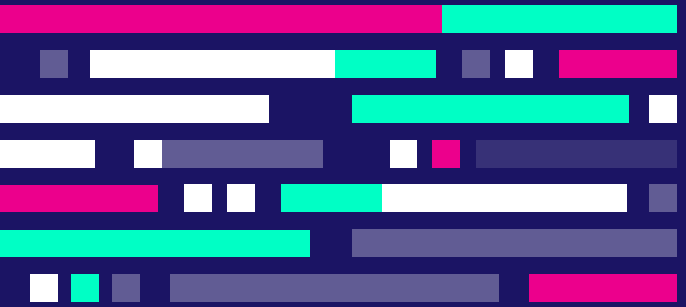
RESULT INSIGHTS

1. Date Range

We now know what range our dates fall under! We can see that data spans over a four-month period. However, we'll only be looking at the last three months as subscriptions are under a monthly basis. Also, for simplicity, none of the users here have subscribed and canceled within the same month.

02

CALCULATING CHURN RATES BY SEGMENT



HOW DO WE FIND CHURN RATES?

To find churn rates per month for the two subscriber segments, a series of temporary tables will have to be created in which I will define monthly ranges and calculate monthly user count for active and canceled subscriptions.

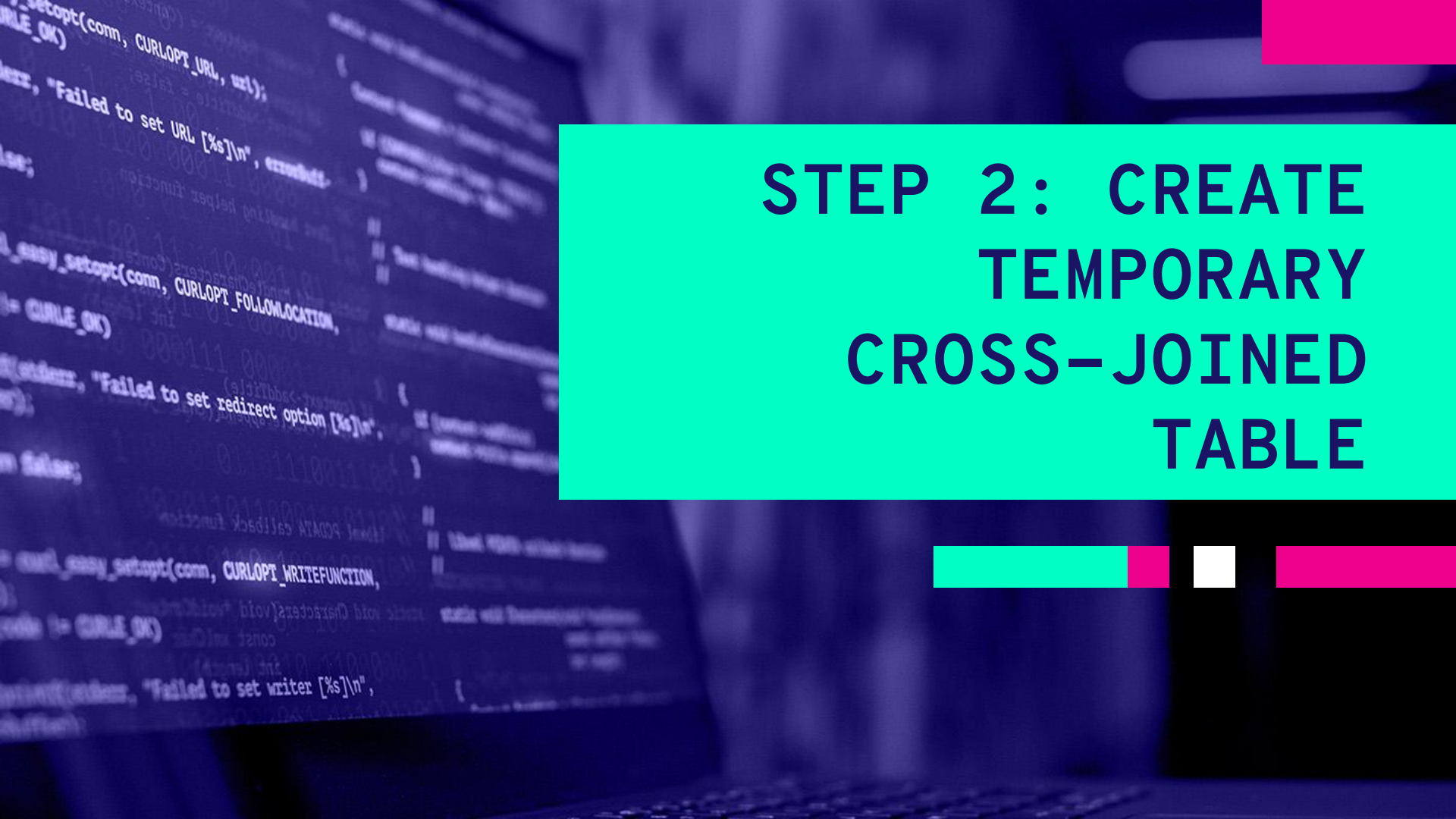

STEP 1: DEFINE
MONTH RANGES

CODE

```
WITH months AS (
  SELECT
  '2017-01-01' AS first_day,
  '2017-01-31' AS last_day
  UNION
  SELECT
  '2017-02-01' AS first_day,
  '2017-02-28' AS last_day
  UNION
  SELECT
  '2017-03-01' AS first_day,
  '2017-03-31' AS last_day
) SELECT * FROM months;
```

RESULT

first_day	last_day
2017-01-01	2017-01-31
2017-02-01	2017-02-28
2017-03-01	2017-03-31


STEP 2: CREATE TEMPORARY CROSS-JOINED TABLE

CODE

```
...
cross_join AS (
  SELECT *
  FROM subscriptions
  CROSS JOIN months
) SELECT * FROM cross_join LIMIT 10;
```

RESULT

id	subscription_start	subscription_end	segment	first_day	last_day
1	2016-12-01	2017-02-01	87	2017-01-01	2017-01-31
2	2016-12-01	2017-01-24	87	2017-02-01	2017-02-28
3	2016-12-01	2017-03-07	87	2017-03-01	2017-03-31
4	2016-12-01	2017-02-12	87	2017-01-01	2017-01-31
5	2016-12-01	2017-03-09	87	2017-02-01	2017-02-28


STEP 3: DEFINE
ACTIVE AND
CANCELED STATUS

CODE

```
...
status AS (
  SELECT id,
  first_day AS month,
  CASE
    WHEN(segment = 87)
    AND (subscription_start < first_day)
    AND (
      subscription_end > first_day
      OR subscription_end IS NULL
        ) THEN 1
    ELSE 0
  END AS is_active_87,
  CASE
    WHEN(segment = 30)
    AND (subscription_start < first_day)
    AND (
      subscription_end > first_day
      OR subscription_end IS NULL
        ) THEN 1
    ELSE 0
  END AS is_active_30,
  CASE
    WHEN (subscription_end BETWEEN first_day AND last_day)
    AND (segment = 87) THEN 1
    ELSE 0
  END AS is_canceled_87,
  CASE
    WHEN (subscription_end BETWEEN first_day AND last_day)
    AND (segment = 30) THEN 1
    ELSE 0
  END AS is_canceled_30
FROM cross_join
) SELECT * FROM status LIMIT 5;
```

RESULT

id	month	is_active_87	is_active_30	is_canceled_87	is_canceled_30
1	2017-01-01	1	0	0	0
1	2017-02-01	0	0	1	0
1	2017-03-01	0	0	0	0
2	2017-01-01	1	0	1	0
2	2017-02-01	0	0	0	0


STEP 4: AGGREGATE
STATUS COUNTS

CODE

```
...
status_aggregate AS (
SELECT
  month,
  SUM(is_active_87) AS sum_active_87,
  SUM(is_active_30) AS sum_active_30,
  SUM(is_canceled_87) AS sum_canceled_87,
  SUM(is_canceled_30) AS sum_canceled_30
FROM status
GROUP BY month
) SELECT * FROM status_aggregate;
```

RESULT

month	sum_active_87	sum_active_30	sum_canceled_87	sum_canceled_30
2017-01-01	278	291	70	22
2017-02-01	462	518	148	38
2017-03-01	531	716	258	84


FINAL STEP: FIND CHURN RATE BY SEGMENT

CODE

```
...
SELECT
  month,
  ROUND((1.0 * sum_canceled_87 /
sum_active_87), 5) AS churn_rate_87,
  ROUND((1.0 * sum_canceled_30 /
sum_active_30), 5) AS churn_rate_30
FROM status_aggregate;
```

RESULT

month	churn_rate_87	churn_rate_30
2017-01-01	0.2518	0.0756
2017-02-01	0.32035	0.07336
2017-03-01	0.48588	0.11732

CHURN RATE INSIGHTS

JAN	FEB	MAR
25%	32%	49%

Segment 80 Churn Rates

Segment 80 has significantly higher churn rates than Segment 30 for all three months.

JAN	FEB	MAR
8%	7%	12%

Segment 30 Churn Rates

Segment 30's churn rates per month stay below 10%, save for March. Performance is much better than Segment 80.

CONCLUSION

Over the course of this project, we took a look at subscription data over time, identified subscriber segments, and found monthly churn rates per month for those segments. **The result was a clear indication that Segment 80 has an unsatisfactory churn rate compared to Segment 30 for every month in our time frame.**

My recommendation would be for this company to identify the differences between subscribers' experiences in both segments and implement changes to minimize churn rates in future months.

THANKS !

I appreciate any questions or feedback offered, so feel free to reach out to me at:
stephen.ellingson1@gmail.com