codecademy

Calculating Codeflix Churn Rates

Analyze Data with SQL

Brian Harris

July 4, 2023

Table of Contents

1. Abstract

This project was designed by Codecademy to use SQL knowledge to calculate the Monthly User Churn rate for the fictional company Codeflix.

2. Introduction

Codeflix, a streaming video startup, is interested in measuring their user churn rate.

- Four months into launching Codeflix, management asks you to look into subscription churn rates. It’s early on in the business and people are excited to know how the company is doing.
- The marketing department is particularly interested in how the churn compares between two segments of users. They provide you with a dataset containing subscription data for users who were acquired through two distinct channels.

Key Questions

1. How many months has the company been operating? Which months do you have enough information to calculate a churn rate?
2. What segments of users exist?
3. What is the overall churn trend since the company started? Churn rate = cancelled users / active users
4. Compare the churn rates between user segments. Which segment of users should the company focus on expanding?

3. Methods

Basic Information

The dataset provided to you contains one SQL table, `subscriptions`. Within the table, there are 4 columns:

- `id` - the subscription id
- `subscription_start` - the start date of the subscription
- `subscription_end` - the end date of the subscription
- `segment` - this identifies which segment the subscription owner belongs to

Codeflix requires a minimum subscription length of 31 days, so a user can never start and end their subscription in the same month.

3.1 Get Familiar with the Codeflix data

Take a look at the first 100 rows of data in the subscriptions table. How many different segments do you see?

Two (2) segments:

- Segment 30
- Segment 87

id	subscription_start	subscription_end	segment
11	2016-12-01	2017-01-17	87
12	2016-12-01	2017-02-07	87
13	2016-12-01		30
14	2016-12-01	2017-03-07	30

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

3.2 Get Familiar with the Codeflix data

Determine the range of months of data provided. Which months will you be able to calculate churn for?

Data has a start date range of 2016-12-01 to 2016-03-30. I will be able to calculate churn only for January, February, and March. December doesn't have any active users since a user is defined as active if they signed up before the first of the month. There are also no cancelled users in December since Codeflix requires a minimum 31 day subscription.

MIN(subscription_start)	MAX(subscription_start)
2016-12-01	2017-03-30

```
SELECT MIN(subscription_start),
MAX(subscription_start)
 FROM subscriptions;
```

3.3 Calculate Churn for Each Segment: Create a Months Table

Create a temporary table of months in order to make a comparison in the next few steps.

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

```
 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;
```

3.4 Calculate Churn for Each Segment: Cross Join subscriptions to months

Generate a table with all possible combinations to set up a comparison that determines what month start dates and end dates fall into.

id	subscription_start	subscription_end	segment	first_day	last_day
1	2016-12-01	2017-02-01	87	2017-01-01	2017-01-31
1	2016-12-01	2017-02-01	87	2017-02-01	2017-02-28
1	2016-12-01	2017-02-01	87	2017-03-01	2017-03-31

```
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
),
cross_join AS (
  SELECT *
  FROM subscriptions
  CROSS JOIN months
)
SELECT *
FROM cross_join
LIMIT 50;
```

3.5 Calculate Churn for Each Segment: Create status table with active columns

Generate a table that shows whether or not the user was active in that particular month. Add columns to count indicate which segment the user is in: segment 37 or segment 80.

id	month	is_active_87	is_active_30
1	2017-01-01	1	0
1	2017-02-01	0	0
1	2017-03-01	0	0

```
…,
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
)
SELECT *
FROM status
LIMIT 10;
```

3.6 Calculate Churn for Each Segment: Create status table with active + canceled columns

Add columns that indicate if the user in that segment canceled that month.

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

```
…,
CASE
   WHEN (segment = 87) AND (subscription_end BETWEEN
first_day AND last_day) THEN 1
   ELSE 0
   END AS is_canceled_87,
  CASE
   WHEN (segment = 30) AND (subscription_end BETWEEN
first_day AND last_day) THEN 1
   ELSE 0
   END AS is_canceled_30
FROM cross_join
)
SELECT *
FROM status
LIMIT 10;
```

3.7 Calculate Churn for Each Segment: Add a table to count the active users and cancellations in each segment

Add a table that groups by month and sums the number of active and canceled users in each month and each segment.

```
…,
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
LIMIT 10;
```

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

3.8 Calculate Churn for Each Segment: Modify SELECT statement

Modify the SELECT statement to indicate the month and to calculate the churn in each segment.

```
…,
SELECT
  month,
  1.0 * sum_canceled_87 / sum_active_87 AS churn_87,
  1.0 * sum_canceled_30 / sum_active_30 AS churn_30
FROM status_aggregate;
```

month	churn_87	churn_30
2017-01-01	0.251798561151079	0.0756013745704467
2017-02-01	0.32034632034632	0.0733590733590734
2017-03-01	0.485875706214689	0.11731843575419

3.9 BONUS

How would you modify this code to support a large number of segments?
Remove the segment hard coding from the SELECT statement in the status table. GROUP BY month AND segment in the status_aggregate table.

month	segment	churn
2017-01-01	30	0.0756013745704467
2017-01-01	87	0.251798561151079
2017-02-01	30	0.0733590733590734
2017-02-01	87	0.32034632034632
2017-03-01	30	0.11731843575419
2017-03-01	87	0.485875706214689

```
…,
status AS (
SELECT id,
  first_day AS month,
  segment,
  CASE
    WHEN (subscription_start < first_day) AND
(subscription_end > first_day OR subscription_end IS
NULL) THEN 1
    ELSE 0
  END AS is_active,
  CASE
   WHEN (subscription_end BETWEEN first_day AND
last_day) THEN 1
   ELSE 0
   END AS is_canceled
FROM cross_join
),
status_aggregate AS (
SELECT
  month,
  segment,
  SUM(is_active) as sum_active,
  SUM(is_canceled) as sum_canceled
FROM status
GROUP BY month, segment
)
```

4. Results & Discussion

4.1 Results and Discussion

Codeflix has been open for 4 months (Dec 1, 2016 - March 31, 2017). There is enough data to analyze churn for 3 of those months (January, February, March). Marketers have divided customers into 2 segments: Segment 87 and Segment 30. The overall churn rate for segment 87 has been steadily increasing whereas the churn rate for segment 30 has remained relatively flat. Segment 30 has a much lower churn rate (~75%) than Segment 87.

Therefore, Codeflix should focus on expanding the customers in Segment 30.

month	Segment 87 Churn %	Segment 30 Churn %
January	25.2%	7.6%
February	32.0%	7.3%
March	48.9%	11.7%

5. Acknowledgements

All Data and worflow derived from “User Churn” project in the Codecademy course “Analyze Data with SQL”.