codecademy

Codeflix User Churn Rates

Analyze Data with SQL

Ashisha Konnur

11 Oct 2021

Table of Contents

- Introduction (p. 3)
- Getting familiar with the data (p. 4)
- Churn rate for each segment (p. 7)
- Code adaptation for any number of segments (p. 13)

Introduction

Four months into launching Codeflix, management wants to look into subscription churn rates to know how the company is doing.

The marketing department is particularly interested in how the churn compares between two segments of users. Available, we have a dataset containing subscription data for users who were acquired through two distinct channels.

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

Getting familiar with the data

1. The 'subscriptions' table

- The table on the top right shows the 10 first records of the 'subscriptions' table with descriptors id, start date, end date end segment.
- Using the DISTINCT clause on the 'segment' descriptor, we see that there are two channels: 87 and 30.

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
6	2016-12-01	2017-01-19	87
7	2016-12-01	2017-02-03	87
8	2016-12-01	2017-03-02	87
9	2016-12-01	2017-02-17	87
10	2016-12-01	2017-01-01	87

segment
87
30

2. Date range

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

Although Codeflix has been active for 4 months, we cannot consider the first month when calculating the churn rate; the 31-day rule makes it so that there are no cancelations during the first month.
With subscriptions running from 2016-12-01 to 2017-03-30, **we will compare the segments' performance over the three-month period from January 1st , 2017 through March 31st , 2017 using their churn rates.**

Churn rate for each segment

1. Creating a temporary table, 'months'.

Our first step is to create a table with the first and last days of the months we will be calculating a churn rate over.

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

2. Creating a temporary table, ‘cross_join’

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
2	2016-12-01	2017-01-24	87	2017-01-01	2017-01-31
2	2016-12-01	2017-01-24	87	2017-02-01	2017-02-28
2	2016-12-01	2017-01-24	87	2017-03-01	2017-03-31
3	2016-12-01	2017-03-07	87	2017-01-01	2017-01-31
3	2016-12-01	2017-03-07	87	2017-02-01	2017-02-28
3	2016-12-01	2017-03-07	87	2017-03-01	2017-03-31

The next step is to attach each row from the Subscriptions table to each row of the Months table using the CROSS JOIN clause. We get the resulting table on the left.

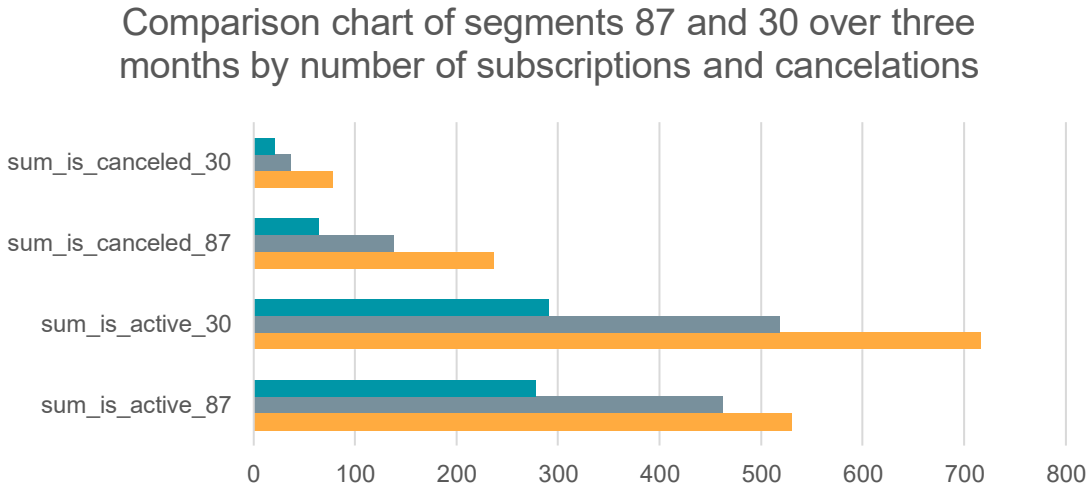
3. Creating a temporary table, ‘status’

Using our previous table, we can now introduce four new columns that will give us information on a subscriber’s status each month: 1 corresponds to yes, 0 corresponds to no. We also distinguish a user’s segment by the column within the status column. We will later see a more efficient method of grouping subscribers by segment number.

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	0	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
2	2017-03-01	0	0	0	0
3	2017-01-01	1	0	0	0
3	2017-02-01	1	0	0	0
3	2017-03-01	1	0	1	0

4. Creating and analysing temporary table, ‘status_aggregate’

The table to the bottom right groups the previous table by month and sums each row so we can get the total number of corresponding descriptor in each month. Above it, we can better see how the performance of one segment compares to the other. Both segments incur an increase in cancelations and subscriptions month after month. However, by March segment 30 has significantly less cancelations and far more subscriptions.



	sum_is_active_87	sum_is_active_30	sum_is_canceled_87	sum_is_canceled_30
1/1/2017	278	291	64	21
2/1/2017	462	518	138	37
3/1/2017	531	716	237	79

churn_rate_87	churn_rate_30
0.3454	0.0898

Which segment has a lower churn rate?

- The churn rate for segment 87 is 35% while for segment 30 it is only 9%. The computation of the chum rate quantifies what we saw on the previous slide: segment 30 performed 26% better than segment 87 over the three-month period.

Code adaptation for any number of segments

By modifying our code in the initial 'status_aggregate' table, we avoid having to include each segment number manually. This is accomplished by **removing the segment condition in WHERE clause** and **using the 'GROUP BY segment' command instead**. The results are shown below.

Temporary status_aggregate table without hardcoding the segment numbers

month	segment	SUM(is_active)	SUM(is_canceled)
2017-01-01	30	291	21
2017-02-01	30	518	37
2017-03-01	30	716	79
2017-01-01	87	278	64
2017-02-01	87	462	138
2017-03-01	87	531	237

Getting the churn rate by hardcoding the segment numbers

churn_rate_87	churn_rate_30
0.3454	0.0898

Getting the churn rate without hardcoding the segment numbers

segment	churn_rate
30	0.0898
87	0.3454

Thank you for your time and attention.