codecademy

CODEFLIX CHURN RATES

Learn SQL from Scratch

Meredith Clement 09/2018

Contents

Get familiar with Codeflix.

- How many months has the company been operating?
- For which months is there enough information to calculate a churn rate?
- What segments of users exist?

Find the overall churn trend since the company started.

Compare the churn rates between user segments.

- Which segment of users should the company focus on expanding?

Modify the code to support a large number of segments.

codecademy

CODEFLIX, AN OVERVIEW

LEARN SQL FROM SCRATCH

CLEMENT SEP.2018

CODEFLIX AS OF APRIL 1, 2017

Codeflix has been operating for four months, since December 1, 2016.

Since a minimum subscription length of 31 days is required, churn rate can be calculated on the last three months: January, February, and March 2017.

There are currently two different segments of subscribers – 30 and 87 – which were acquired through two separate channels.

Earliest Start	Latest End
2016-12-01	2017-03-31

Segments
30
87

```
SELECT
  MIN(subscription_start) AS 'Earliest Start',
  MAX(subscription_end) AS 'Latest End'
FROM subscriptions;


SELECT DISTINCT segment AS Segments
FROM subscriptions
ORDER BY segment;
```

codecademy

Churn vs Growth

CHURN VS GROWTH AS OF APRIL 1, 2017

TRENDS TO DATE

Since the company began, the total churn rate has been increasing over time. Codeflix is still very new to the market, and this is not a positive trajectory.

While churn increased only slightly from January to February, March saw a sharp incline. At the same time, per-month customer acquisition declined by nearly 19% between December 2016 and March 2017.

New customer starts held steady from February to March, so while the hope is that it has reached its floor, that cannot be assumed. Even if this stabilizes or improves, it still costs more to gain new customers than to retain existing ones.

Consider providing an incentive for customers to subscribe for a longer term than monthly, thus allowing for fewer cancellation opportunities each year. Tests could be conducted between offers for 3-, 6-, and 12-month terms.

month	total_churn
2017-01-01	0.16140350877193
2017-02-01	0.188832487309645
2017-03-01	0.27164416203336

month	total_starts
2016-12-01	570
2017-01-01	507
2017-02-01	460
2017-03-01	463

CHURN VS GROWTH AS OF APRIL 1, 2017

SEGMENT COMPARISON

Whereas churn has increased in both segments, the majority of it is attributable to cancellations of segment 87 subscribers. Customer starts have some slight variations between the two segments, but they even out over time.

Note the total March starts were bolstered by the increase in segment 87 starts; without that March bump, the total starts would have continued to fall. Coupled with the rising churn, this is a disturbing trend.

Where should the next efforts be targeted:

How are subscribers in segment 30 different from those in segment 87?

month	churn_87	churn_30	total_churn
2017-01-01	0.2509	0.0756	0.1614
2017-02-01	0.3169	0.0734	0.1888
2017-03-01	0.4769	0.117	0.2716

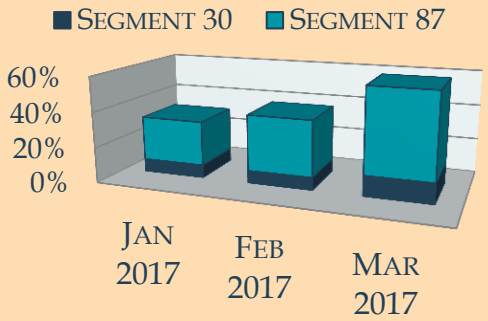
month	start_87	start_30	total_starts
2016-12-01	279	291	570
2017-01-01	258	249	507
2017-02-01	222	238	460
2017-03-01	241	222	463

CHURN VS GROWTH AS OF APRIL 1, 2017

RECOMMENDATIONS

Since churn in segment 87 subscribers is three to four times that of segment 30, the latter should be expanded if feasible.

CHURN RATES



ACTIVE SUBSCRIBERS

ON FIRST OF EACH MONTH

Because of the lower churn, segment 30 shows steady growth even with fewer new starts each month.

Ideally, the decline of new starts in segment 30 would be curtailed, and the churn stabilized or improved.

Consider conducting satisfaction surveys of customers in both segments. There may be simple changes that could be implemented to reverse the churn rate in segment 87 subscribers, rather than discontinuing that channel altogether. Survey responses may also indicate optimal directions for new marketing.

codecademy

Future Needs

FUTURE NEEDS FOR CALCULATING CHURN

```
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
),
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
),
```

continued…

In future, additional segments are expected. And one or both of the current segments may fall away. The code for calculating churn can be modified to remove the segment numbers, but this method has some weaknesses over time:

What if there were 30 segments over the course of a full year, or 100? What does the results table look like in three years, or in 15? The temporary 'months' table will be extremely long, and it requires manual updates for each month.

```
…

status_aggregate AS (
  SELECT
    month,
    segment,
    SUM(is_active) AS sum_active,
    SUM(is_canceled) AS sum_canceled
  FROM status
  GROUP BY month,segment
)
SELECT
  month AS Month,
  segment AS Segment,
  ROUND(1.0* sum_canceled / sum_active,4) AS Churn
FROM status_aggregate
GROUP BY month,segment;
```

Month	Segment	Churn
2017-01-01	30	0.0756
2017-01-01	87	0.2509
2017-02-01	30	0.0734
2017-02-01	87	0.3169
2017-03-01	30	0.117
2017-03-01	87	0.4769

FUTURE NEEDS FOR CALCULATING CHURN

LONG-TERM SOLUTION

The code to generate results like in the table below will work in perpetuity with no updates needed, and also adds functionality. As presented here, it can be used to:

- determine an annual business cycle.
- make year-over-year comparisons to evaluate effects of promotions or other events.
- convert the query to a temporary table to make further queries against it or limit what data is displayed for different purposes/audiences.

Year	Segment	Jan Churn %	Feb Churn %	Mar Churn %	Apr Churn %	May Churn %	Jun Churn %	Jul Churn %	Aug Churn %	Sep Churn %	Oct Churn %	Nov Churn %	Dec Churn %
2017	30	7.5601	7.3359	11.6992									
2017	87	25.0896	31.6916	47.6895									
2017	*segment X*												
2018	*30*												

As the dataset increases in size, the server may have difficulty creating multiple large temporary tables and querying them. One or more could be converted into permanent tables.