Calculating Churn Rates with Codeflix

Learn SQL from Scratch

Emily Brown | 29-Aug-18

Table of Contents

1. Becoming Familiar with Codeflix

1.1 Becoming Familiar with Codeflix

- Codeflix is a recently-launched streaming video startup. Four months into launching Codeflix, management would like to review churn rates to see how the company is doing
- Codeflix's marketing team is particularly interested in how the churn compares between two segments of users, and have provided a dataset containing subscription data for users who were acquired through two distinct channels
- This dataset contains one table, `subscriptions`. Within the table, there are 4 columns:
  1. `id` - the subscription id
  2. `subscription_start` - the start date of the subscription
  3. `subscription_end` - the end date of the subscription
  4. `segment` - this identifies which segment the subscription owner belongs to
- The database schema for `subscriptions` suggests that there are 2,000 rows (fields) of data to use in the analysis

Database Schema	
subscriptions	**2000 rows**
columns	**format**
id	INTEGER
subscription_start	TEXT
subscription_end	TEXT
segment	INTEGER

1.2 Codeflix Subscription Data Date Range

Four months into launching Codeflix, management would like to review churn rates to see how the company is doing

The dates available in the subscriptions dataset that we can use to calculate churn can be found using the MIN and MAX functions of the subscription_start column

- The MIN(subscription_start) is December 1, 2016
- The MAX (subscription_start) is March 30, 2017

Note that for the purposes of churn rate calculations, Codeflix requires a minimum subscription length of 31 days, so a user can never start and end their subscription in the same month

This means that we can calculate churn rates for the months of January 2017, February 2017 and March 2017 (users starting on December 1, 2016 would only be able to cancel from January 1, 2017)

SQL Query

```
SELECT MIN(subscription_start) AS Start_of_Data,
       MAX(subscription_start) AS End_of_Data
FROM subscriptions;
```

Query Results

Start_of_Data	End_of_Data
2016-12-01	2017-03-30

1.3 Codeflix User Segments

Codeflix’s marketing team is particularly interested in how the churn compares between two segments of users, and have provided a dataset containing subscription data for users who were acquired through two distinct channels

A review of the subset of the subscriptions data (using Query #1 to the right) shows that Codeflix labels these two user segments “87” and “30”

Grouping a count of the rows of subscriptions data by segment (Query #2 to the right) **shows that there are 1000 fields of data in the “30” segment and 1000 fields in the “87” segment**

SQL Queries

```
SELECT *
FROM subscriptions
LIMIT 100;

SELECT segment, COUNT(*)
FROM subscriptions
GROUP BY segment;
```

Query Results (Query #2)

segment	COUNT(*)
30	1000
87	1000

2. Churn Rate by Month

2.1 Codeflix Churn Rate by Month

Churn rate is the percent of subscribers that have canceled within a certain period

- For our analysis, we calculate churn rate on **a monthly basis**, using the formula: **cancellations / total subscribers for each month**
- **This allows us to compare churn over time**

Based on the Codeflix's data and using our SQL query to the right, we calculate Codeflix's overall churn rates as:

- **16.2% in January 2017**
- **19.0% in February 2017**
- **27.4% in March 2017**

This means that Codeflix's churn rate has been increasing over time on a month-to-month basis

Query Results

month	churn_rate
2017-01-01	0.161687170474517
2017-02-01	0.189795918367347
2017-03-01	0.274258219727346

SQL Query

```
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,
    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,
    SUM(is_active) AS active,
    SUM(is_canceled) AS canceled
  FROM status
  GROUP BY month)
SELECT month, 1.0 * canceled / active AS churn_rate
FROM status_aggregate;
```

3. Churn Rate by User Segment

3.1 Churn Rate by User Segment

Codeflix also asked us to analyze **churn rate by user segment**

- With churn rate calculated as: **cancellations / total subscribers for each month and user segment (in this case segments “87” and “30”)**

Based on the Codeflix’s data and using our SQL query to the right, we calculate Codeflix’s churn rates by month and user segment as:

Segment 87

- January 2017: 25.2%
- February 2017: 32.0%
- March 2017: 48.6%

Segment 30

- January 2017: 7.6%
- February 2017: 7.3%
- March 2017: 11.7%

Query Results

month	churn_rate_87	churn_rate_30
2017-01-01	0.251798561151079	0.0756013745704467
2017-02-01	0.32034632034632	0.0733590733590734
2017-03-01	0.485875706214689	0.11731843575419

SQL Query

```
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,
    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 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),
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 month, 1.0 * sum_canceled_87 / sum_active_87 AS churn_rate_87, 1.0 *
sum_canceled_30 / sum_active_30 AS churn_rate_30
FROM status_aggregate;
```

4. Conclusions

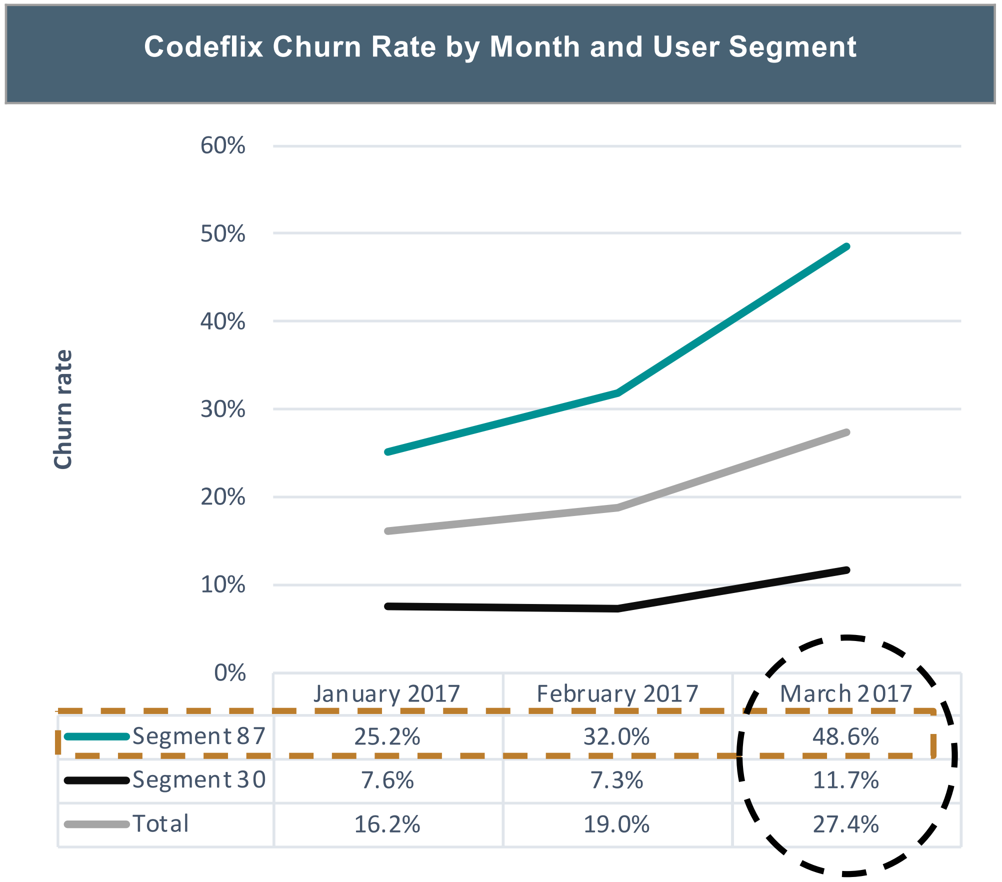
4.1 Conclusions

A review of Codeflix’s churn rates by month and user segment show a clear trend of **increasing total churn rate driven primarily by the significant churn rate increase in Segment 87**

While the churn rate of Segment 30 does increase in March 2017 (11.7%) compared to January and February 2017 (c.7.5% p.m.), this increase is not nearly as significant compared to the increase in churn rate of Segment 87 (25.2% in January 2017 rising to 48.6% in March 2017)

We would therefore recommend that **Codeflix should focus on expanding customers in Segment 30**

Additionally, we would recommend that Codeflix spend some time talking to former customers in Segment 87 to understand why they have stopped using the product (as well as former customers in Segment 30). There may be important lessons to learn from them



	January 2017	February 2017	March 2017
Segment 87	25.2%	32.0%	48.6%
Segment 30	7.6%	7.3%	11.7%
Total	16.2%	19.0%	27.4%

5. Bonus Question

5.1 Bonus Question

How would you modify the SQL query to support a larger number of segments?

- While the SQL query used in Section 3.1 is fine for two segments, if Codeflix were to add a larger number of segments to their business then this query could be come unwieldy
- To support a larger number of segments, I would modify the ”Total Churn” query from Section 2.1 to include the “segment” column through the various temporary tables, and would include “segment” in the GROUP BY clause of the status_aggregate temporary table
- I would make the churn rate calculation a temporary table and include the “segment” in the SELECT clause of the temporary table churn_rate
- In the final part of the query used to return the outputted table, I would use “SUM(CASE ...)” language for each segment to create output columns for each segment (this can easily be expanded to include more segments)

Query Results

month	churn_rate_87	churn_rate_30
2017-01-01	0.251798561151079	0.0756013745704467
2017-02-01	0.32034632034632	0.0733590733590734
2017-03-01	0.485875706214689	0.11731843575419

SQL Query

```
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),
status_aggregate AS (SELECT month, segment, SUM(is_active) AS active,
    SUM(is_canceled) AS canceled
  FROM status
  GROUP BY month, segment),
churn_rate AS (SELECT month, segment, 1.0 * canceled / active AS
churn_rate
FROM status_aggregate)
SELECT month,
    SUM(CASE WHEN segment = 87
             THEN churn_rate
             ELSE 0
    END) AS churn_rate_87,
    SUM(CASE WHEN segment = 30
             THEN churn_rate
             ELSE 0
    END) AS churn_rate_30
FROM churn_rate
GROUP BY month;
```