codecademy

Capstone: Churn Rates

Learn SQL from Scratch

Jeremy Isler

Table of Contents

Basic Goals

1. Get familiar with Codeflix.
2. Overall churn trend since launch?
3. Compare user segments.

Specific Goals

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

1. Get familiar with the company, Codeflix

- To get familiar with the company Codeflix (more importantly the database they provided), it was necessary to determine the time frame of the data collected.
- 2 queries were run to show the databases time frame:

✓ The first and last dates subscriptions started

✓ The first and last dates subscriptions ended

- This shows the database goes from 12/01/2016 to 3/31/2017

```sql
--1. Find when the earliest and latest subscription start dates.
SELECT MIN(subscription_start) AS 'First_subscription_start',
MAX(subscription_start) AS 'Last_subscription_start'
FROM subscriptions;

--2. Find when the earliest and latest subscription end dates.
SELECT MIN(subscription_end) AS 'First_subscription_ends',
MAX(subscription_end) AS 'Last_subscription_ends'
FROM subscriptions;
```

Query Results	
First_subscription_start	Last_subscription_start
2016-12-01	2017-03-30
First_subscription_ends	Last_subscription_ends
2017-01-01	2017-03-31

Months Codeflix has been Operating

- The previous slide showed the database to be over the time frame 12/01/2016 until 03/31/2017.
- However an exact number of months of the data was also need.
- Shown to the right is the code which calculates the number of months of operating data.
- November 2016 was used in the temporary table of months to help validate the code only returns months operated with data, rather than the number of months listed in the months table

Months_Operating
4

```
WITH months AS (
  SELECT
    '2016-11-01' AS 'first_day',
    '2016-11-30' AS 'last_day'
  UNION
  SELECT
    '2016-12-01' AS 'first_day',
    '2016-12-31' AS 'last_day'
  UNION
  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 < last_day)
        AND (subscription_end > last_day OR
subscription_end IS NULL)
      THEN 1
      ELSE 0
    END AS 'user_in_month'
  FROM cross_join),
status_aggregate AS (SELECT month,
    SUM(user_in_month) AS 'users'
  FROM status
  GROUP BY month),
time AS (SELECT *
FROM status_aggregate
WHERE users > 0)
 SELECT COUNT(month) AS 'Months_Operating'
 FROM time;
```

Get familiar with the company, Codeflix – Churn Rates possible

- The equation for Churn Rate is: $\text{Churn Rate: } \frac{\text{Cancellations in Month}}{\text{Subscribers to Start Month}}$

- For a churn rate to be possible there must be cancellations during the month and there must be active users to start the month.
- In slide 3, it was shown that the first subscription started 12/01/2016. Thus the first month possible for churn rates based on active users requirements is 01/01/2017.
- To determine the months possible for churn rates, the code on the right calculates which months a churn rate is possible by having cancellations.
- Replacing lines 37 on in the code to the right with the snippet below, calculates the number of months for possible churn rates (being 3)

```
37  SELECT COUNT(Cancellations_in_Month) AS
    'Possible_Churn_Rates'
38  FROM status_aggregate
39  WHERE Cancellations_in_Month > 0;
```

Query Results
Possible_Churn_Rates
3

Query Results	
month	Cancellations_in_Month
2016-12-01	0
2017-01-01	85
2017-02-01	175
2017-03-01	316
2017-04-01	0

```
1   WITH months AS (
2     SELECT
3       '2016-12-01' AS 'first_day',
4       '2016-12-31' AS 'last_day'
5     UNION
6     SELECT
7       '2017-01-01' AS 'first_day',
8       '2017-01-31' AS 'last_day'
9     UNION
10    SELECT
11      '2017-02-01' AS 'first_day',
12      '2017-02-28' AS 'last_day'
13    UNION
14    SELECT
15      '2017-03-01' AS 'first_day',
16      '2017-03-31' AS 'last_day'
17  UNION
18  SELECT
19      '2017-04-01' AS 'first_day',
20      '2017-04-30' AS 'last_day'),
21  cross_join AS (SELECT *
22    FROM subscriptions
23    CROSS JOIN months),
24  status AS (SELECT id,
25      first_day AS 'month',
26      CASE
27        WHEN (subscription_end < last_day)
28    AND (subscription_end > first_day)
29        THEN 1
30        ELSE 0
31      END AS 'users_ending'
32    FROM cross_join),
33  status_aggregate AS (SELECT month,
34      SUM(users_ending) AS 'Cancellations_in_Month'
35    FROM status
36    GROUP BY month)
37  SELECT *
38  FROM status_aggregate;
```

Get familiar with Codeflix Segments

- As the company uses “segments” to determine how they acquired their customers.
- To have a better understanding of how to efficiently market Codeflix, the segments must be better understood.
- Shown in the top code is the code to count the number of segments in the database (being 2).
- To show the specifics of the segments, the bottom code shows which segments exist (being 87 and 30).

```
1  -- Count of Segments
2  With seg_count AS (SELECT *
3                     FROM subscriptions
4                     GROUP BY segment)
5  SELECT COUNT(*) AS '# of Segments'
6  FROM seg_count;
```

Query Results
of Segments
2

Query Results
segment
87
30

```
1  SELECT DISTINCT segment
2  FROM subscriptions;
```

2. Overall Churn Rate

- After getting familiar with the company it is important to begin to analyze the data collected.
- The most basic analysis for the entire data set is simply the churn rate of the company since inception.
- Shown to the right is the code to calculate the overall churn rate of Codeflix since it began.
- The query results below shows the overall churn rate to be 0.303.
- This means overall nearly 30% of the users who have signed up for the service is no longer a customer.

Query Results
Overall_Churn_Rate
0.303

```
WITH months AS (SELECT
    '2017-12-01' 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 < last_day)
      THEN 1
      ELSE 0
    END AS 'users',
    CASE
      WHEN subscription_end < last_day THEN 1
      ELSE 0
    END AS 'has_canceled'
  FROM cross_join),
status_aggregate AS (SELECT month,
    SUM(users) AS 'active',
    SUM(has_canceled) AS 'canceled'
  FROM status
  GROUP BY month)
SELECT
1.0 * canceled / active AS 'Overall_Churn_Rate'
FROM status_aggregate;
```

Churn Rate by Month

- To gather slightly more information it is important to analyze the data on shorter time frames.
- The code to the right calculates the churn rate for each of the 3 months possible.
- The query results below shows the churn rate to be continually increasing with month.

```
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,
  ROUND(1.0 * canceled / active,3) AS
'Churn_Rate'
FROM status_aggregate;
```

Query Results

month	Churn_Rate
2017-01-01	0.162
2017-02-01	0.19
2017-03-01	0.274

3. Compare Segment Churn Rates

- The churn rate can then be used to analyze the segments more in depth.
- The query on the right calculates the churn rate for each segment based on the month.
- This shows the segment 87 has a significantly higher churn rate for each month than for segment 30.

Query Results		
month	Churn_Rate_87	Churn_Rate_30
2017-01-01	0.252	0.076
2017-02-01	0.32	0.073
2017-03-01	0.486	0.117

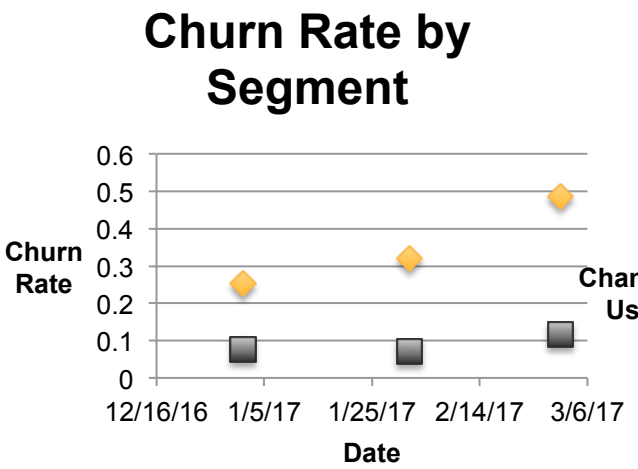
```
WITH months AS (SELECT
    '2017-01-01' AS 'first_day',
    '2017-01-31' AS 'last_day'
  UNION
  SELECT
    '2017-02-01' AS 'first_day',
    '2017-02-28' AS 'last_day'
  UNION
  SELECT
    '2017-03-01' AS 'first_day',
    '2017-03-31' AS 'last_day'),
cross_join AS (SELECT *
               FROM subscriptions
               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)
           AND segment = '87' THEN 1
           ELSE 0
           END AS 'is_active_87', CASE
           WHEN (subscription_start < first_day)
           AND (subscription_end > first_day OR
                subscription_end IS NULL)
           AND segment IS '30' 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, ROUND(1.0 *
sum_canceled_87 / sum_active_87,3) AS
'Churn_Rate_87', ROUND(1.0 * sum_canceled_30 /
sum_active_30,3) AS 'Churn_Rate_30'
             FROM status_aggregate;
```

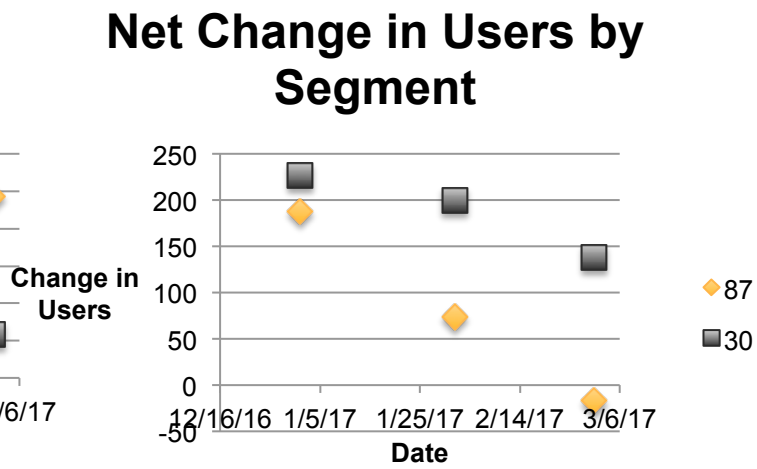
Segment to Focus on

Query Results		
month	Net Change in 87	Net Change in 30
2017-01-01	188	227
2017-02-01	74	200
2017-03-01	-17	138

- To the right is the query to calculate the net change in users for each segment for each month (with the table above).
- This net calculation shows that while both segments have seen decreases in each month of growing active users, *segment 87 actually net lost users in the latest month*.
- Charts below show in both churn rate and net change in user by segment segment 30 has outperformed segment 87.
- To maximize the potential for growth, **Codeflix should focus on segment 30 to expand** as it has shown the best retention of customers.

```
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 BETWEEN
first_day AND last_day) AND segment = '87' THEN 1
      ELSE 0
    END AS 'added_87', CASE
           WHEN (subscription_start BETWEEN
first_day AND last_day) AND segment IS '30' THEN
1
           ELSE 0
           END AS 'added_30', CASE
      WHEN segment = '87' AND (subscription_end
BETWEEN first_day AND last_day) THEN 1
      ELSE 0
    END AS 'lost_87', CASE
      WHEN segment = '30' AND (subscription_end
BETWEEN first_day AND last_day) THEN 1
      ELSE 0
    END AS 'lost_30'
           FROM cross_join
          ),
          status_aggregate AS (SELECT month,
SUM(added_87) AS 'Added_by_87', SUM(added_30) AS
'Added_by_30', SUM(lost_87) AS 'Canceled_by_87',
SUM(lost_30) AS 'Canceled_by_30'
                               FROM status
                               GROUP BY month)
               SELECT month, 1*(Added_by_87-
Canceled_by_87) AS 'Net Change in 87', 1*
(Added_by_30-Canceled_by_30) AS 'Net Change in
30'
               FROM status_aggregate;
```





Thank You!