



Capstone: Churn Rates

Learn SQL from Scratch

Wesley Yang | 2018.07.10



Table of Contents

- Get familiar with Codeflix
- What is the overall churn rate by month?
- Compare the churn rates between segments

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 Codeflix

Get familiar with the company

How many months has the company been operating?

- Selecting the MIN subscription start and the MAX subscription end date gives you the full time range for the dataset.
- By looking at the maximum and minimum dates from above, you can determine that there are 4 months (December 2016, January, February and March 2017).

What segments of users exist?

- Using DISTINCT on the segment and grouping by those segments will give the individual groupings.
- There are two segments (30 and 87)

```
1  ---How long has Codeflix been operating?---
2  SELECT MIN(subscription_start) 'Beginning date',
3         MAX(subscription_end) 'End date'
4  FROM subscriptions;
5
6  ---How many total Codeflix Users?---
7  SELECT COUNT(DISTINCT ID) AS 'Codeflix Users'
8  FROM subscriptions;
9
10 ---How many segments are there?---
11 Select DISTINCT segment,
12        COUNT (DISTINCT id) AS 'Count'
13 FROM subscriptions
14 GROUP BY segment;
```

Beginning date	End date
2016-12-01	2017-03-31
Codeflix Users	
2000	
segment	Count
30	1000
87	1000

2. What is the overall churn rate by month?

What is the overall churn trend since the company started?

- Three tables were created by using the common table expression at the (months, status and status_aggregate) at the beginning so that the final query is readable.
 - months was created for the calendar date ranges
 - status was used to determine the user's status by month and to cross join the months table to each user
 - status_aggregate is needed to sum up the active/canceled users per month
- The query was used to calculate the churn rate by month.
- There was no churn during December due to the 31 day minimum subscription length.
- The overall churn increased as time progressed.

month	churn_rate
2016-12-01	N/A
2017-01-01	16.17%
2017-02-01	18.98%
2017-03-01	27.43%

```
1 WITH months AS (  
2     SELECT '2016-12-01' AS First_Day,  
3           '2017-12-31' AS Last_Day  
4  
5     UNION  
6     SELECT '2017-01-01' AS First_Day,  
7           '2017-01-31' AS Last_Day  
8  
9     UNION  
10    SELECT '2017-02-01' AS First_Day,  
11          '2017-02-28' AS Last_Day  
12  
13    UNION  
14    SELECT '2017-03-01' AS First_Day,  
15          '2017-03-31' AS Last_Day),  
16  
17 -----  
18 status AS (  
19     SELECT id,  
20           first_day AS month,  
21           CASE WHEN subscription_start < first_day  
22                 AND (subscription_end > first_day  
23                     OR subscription_end IS NULL) THEN 1  
24                 ELSE 0 END AS is_active,  
25  
26           CASE WHEN subscription_end BETWEEN first_day  
27                 AND last_day THEN 1 else 0 END  
28                 AS is_canceled  
29     FROM subscriptions  
30     CROSS JOIN months),  
31  
32 -----  
33 status_aggregate AS (  
34     SELECT month,  
35           SUM(is_active) AS sum_active,  
36           SUM(is_canceled) AS sum_canceled  
37     FROM status  
38     GROUP BY month)  
39  
40 -----  
41 SELECT month,  
42       ((1.0*sum_canceled)/sum_active) AS churn_rate  
43 FROM status_aggregate  
44 GROUP BY month;
```

3. Compare the churn rates between segments

```
1 WITH months AS
2   (SELECT
3
4     '2017-01-01' AS First_Day,
5     '2017-01-31' AS Last_Day
6
7   UNION
8   SELECT
9
10    '2017-02-01' AS First_Day,
11    '2017-02-28' AS Last_Day
12
13   UNION
14   SELECT
15
16    '2017-03-01' AS First_Day,
17    '2017-03-31' AS Last_Day),
18
19 -----
20 status AS (
21   SELECT id,
22
23     first_day AS month,
24
25     CASE WHEN segment = 87
26           AND subscription_start < first_day
27           AND (subscription_end > first_day
28               OR subscription_end IS NULL) THEN 1
29           ELSE 0 END AS is_active_87,
30
31     CASE WHEN segment = 30
32           AND subscription_start < first_day
33           AND (subscription_end > first_day
34               OR subscription_end IS NULL) THEN 1
35           ELSE 0 END AS is_active_30,
36
37     CASE WHEN (segment = 87)
38           AND subscription_end BETWEEN First_Day
39           AND Last_day THEN 1 else 0 END
40     AS is_canceled_87,
41
42     CASE WHEN (segment = 30)
43           AND subscription_end BETWEEN First_Day
44           AND Last_day THEN 1 else 0 END
45     AS is_canceled_30
46
47   FROM subscriptions
48   CROSS JOIN months),
49
50 -----
51 status_aggregate AS (
52   SELECT month,
53
54     SUM(is_active_87) AS sum_active_87,
55     SUM(is_active_30) AS sum_active_30,
56     SUM(is_canceled_87) AS sum_canceled_87,
57     SUM(is_canceled_30) AS sum_canceled_30
58   FROM status
59   GROUP BY month)
60
61 -----
62 SELECT month,
63
64     ((1.0*sum_canceled_30)/sum_active_30) AS churn_rate_30,
65     ((1.0*sum_canceled_87)/sum_active_87) AS churn_rate_87
66 FROM status_aggregate;
```

month	churn_rate_30	churn_rate_87
2017-01-01	7.56%	25.18%
2017-02-01	7.34%	32.03%
2017-03-01	11.73%	48.59%

Compare the churn rates between user segments.

- The with statement was used to create three tables
 - months is used for the date ranges
 - status was used to select user id and to determine active/canceled status for the two segments.
 - The is_active case statements used a hard coded segment number with date logic to determine if the user account was active during the specific month.
- status_aggregate is needed to sum up the active/canceled users by month and segment.
- The query was used to calculate churn rate and grouped by segment and month.
- Which segment of users should the company focus on expanding?
 - user segment 30 had the highest retention out of the two segments.

3. Compare the churn rates between segments

```
1 WITH months AS
2 (SELECT
3
4     '2017-01-01' AS First_Day,
5     '2017-01-31' AS Last_Day
6
7 UNION
8
9     '2017-02-01' AS First_Day,
10    '2017-02-28' AS Last_Day
11
12 UNION
13
14    '2017-03-01' AS First_Day,
15    '2017-03-31' AS Last_Day),
16
17 -----
18 status AS (
19     SELECT id,
20            first_day AS month,
21            CASE WHEN segment = 87
22                 AND subscription_start < first_day
23                 AND (subscription_end > first_day
24                     OR subscription_end IS NULL) THEN 1
25                 ELSE 0 END AS is_active_87,
26
27            CASE WHEN segment = 30
28                 AND subscription_start < first_day
29                 AND (subscription_end > first_day
30                     OR subscription_end IS NULL) THEN 1
31                 ELSE 0 END AS is_active_30,
32
33            CASE WHEN (segment = 87)
34                 AND subscription_end BETWEEN first_day
35                 AND last_day THEN 1 else 0 END
36                 AS is_canceled_87,
37
38            CASE WHEN (segment = 30)
39                 AND subscription_end BETWEEN first_day
40                 AND last_day THEN 1 else 0 END
41                 AS is_canceled_30
42
43 FROM subscriptions
44 CROSS JOIN months),
45
46 status_aggregate AS (
47     SELECT month,
48            SUM(is_active_87) AS sum_active_87,
49            SUM(is_active_30) AS sum_active_30,
50            SUM(is_canceled_87) AS sum_canceled_87,
51            SUM(is_canceled_30) AS sum_canceled_30
52
53 FROM status
54 GROUP BY month)
55
56 SELECT month,
57        ((1.0*sum_canceled_30)/sum_active_30) AS churn_rate_30,
58        ((1.0*sum_canceled_87)/sum_active_87) AS churn_rate_87
59 FROM status_aggregate;
```

Compare the churn rates between user segments.

- The with statement was used to create three tables
 - months is used for the date ranges
 - status was used to select 2017-01-01 and to determine the active/canceled status for the two segments.
 - The is_active case statements used a hard coded segment number with date logic to determine if the user account was active during the specific month.
 - status_aggregate is needed to sum up the active/canceled users by month and segment.
- The query was used to calculate churn rate and grouped by segment and month.
- Which segment of users should the company focus on expanding?
 - user segment 30 had the highest retention out of the two segments.

month	segment	churn_rate
2016-12-01	30	
2017-01-01	30	7.56%
2017-02-01	30	7.34%
2017-03-01	30	11.73%
2016-12-01	87	
2017-01-01	87	25.18%
2017-02-01	87	32.03%
2017-03-01	87	48.59%

Bonus

How would you modify this code to support a large number of segments?

- The with statement creates three tables (similar to the previous query)
 - months creates the date ranges
 - status – the user segment is added for the next step
 - status_aggregate helps aggregate the individual user data by month and by segment.
- By adding segment into GROUP BY, this statement can now accept any number of segments without redundancies.

month	segment	churn_rate
2016-12-01	30	N/A
2017-01-01	30	7.56%
2017-02-01	30	7.34%
2017-03-01	30	11.73%
2016-12-01	87	N/A
2017-01-01	87	25.18%
2017-02-01	87	32.03%
2017-03-01	87	48.59%

```
1 WITH months AS
2   (SELECT '2016-12-01' AS First_Day,
3     '2017-12-31' AS Last_Day
4   UNION
5   SELECT
6     '2017-01-01' AS First_Day,
7     '2017-01-31' AS Last_Day
8   UNION
9   SELECT
10    '2017-02-01' AS First_Day,
11    '2017-02-28' AS Last_Day
12  UNION
13  SELECT
14    '2017-03-01' AS First_Day,
15    '2017-03-31' AS Last_Day),
16
17 status AS (
18   SELECT id,
19     first_day AS month,
20     segment,
21     CASE WHEN subscription_start < first_day
22       AND (subscription_end > first_day
23         OR subscription_end IS NULL) THEN 1
24       ELSE 0 END AS is_active,
25
26     CASE WHEN subscription_end BETWEEN first_day
27       AND last_day THEN 1 else 0 END
28     AS is_canceled
29   FROM subscriptions
30   CROSS JOIN months),
31
32 status_aggregate AS (
33   SELECT month,
34     segment,
35     SUM(is_active) AS sum_active,
36     SUM(is_canceled) AS sum_canceled
37   FROM status
38   GROUP BY month, segment)
39
40 SELECT month,
41   segment,
42   ((1.0*sum_canceled)/sum_active) churn_rate
43 FROM status_aggregate
44 GROUP BY segment, month;
```