Codeflix User Churn Rates

Final Project: Learn SQL from Scratch

Submitted by Miranda Neerhof

On March 26, 2019

Table of Contents

1. About Codeflix

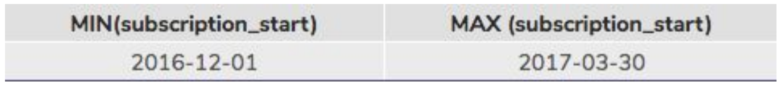
Codeflix has been operating for 4 months

We can see in the data that the first users subscribed on the Dec 1st 2016 and the newest subscribers on March 30st, 2017.

QUERY

```
SELECT MIN(subscription_start),
MAX(subscription_start)
FROM subscriptions;
```

Output



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

We can calculate churn starting in Jan, 2017

Codeflix requires user to subscribe for a minimum of 31 days. As a result, no users can unsubscribe in Dec 2016.

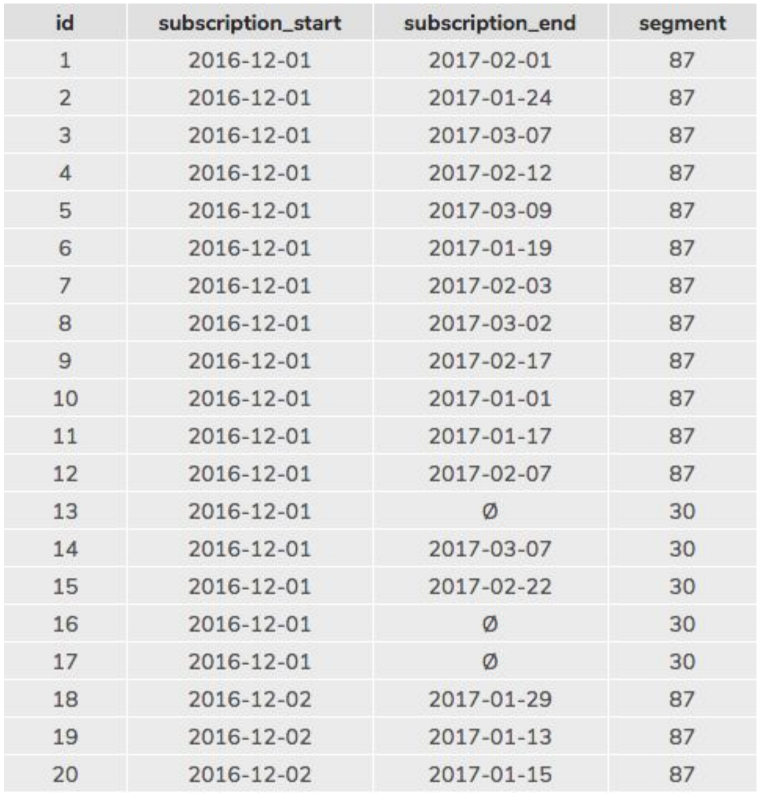
There are 2 user segments, 87 and 30

By taking a look at the first 100 Codeflix users we can notice these two separate segments.

QUERY

```
SELECT *
from subscriptions
LIMIT 100;
```

Output



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
11	2016-12-01	2017-01-17	87
12	2016-12-01	2017-02-07	87
13	2016-12-01	Ø	30
14	2016-12-01	2017-03-07	30
15	2016-12-01	2017-02-22	30
16	2016-12-01	Ø	30
17	2016-12-01	Ø	30
18	2016-12-02	2017-01-29	87
19	2016-12-02	2017-01-13	87
20	2016-12-02	2017-01-15	87

2. Overall Churn

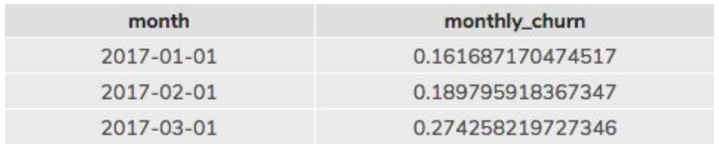
Churn is increasing since the company began

In January churn was 16.1%, in February it was 18.9% and in March it was 27.4%.

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 subscriptions.*, months.*
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 monthly_churn
FROM status_aggregate;
```

Output



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

3. Churn by Segment

Codeflix should expand segment 30

Month to month, churn rates for segment 30 are substantially lower than segment 87.

Focusing on segment 30 will produce higher returns for the company.

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 subscriptions.*, months.*
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 = 30
     THEN 1
   ELSE 0
 END as is_active_30,
  CASE
   WHEN subscription_end BETWEEN first_day AND
last_day
     AND segment = 87
     THEN 1
   ELSE 0
  END as is_canceled_87,
  CASE
    WHEN subscription_end BETWEEN first_day AND
last_day
      AND segment = 30
      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_87,
1.0 * sum_canceled_30/sum_active_30 as churn_30
FROM status_aggregate;
```

Output

month	churn_87	churn_30
2017-01-01	0.251798561151079	0.0756013745704467
2017-02-01	0.32034632034632	0.0733590733590734
2017-03-01	0.485875706214689	0.11731843575419

Thank you