Capstone Project:

Calculating Churn Rates

Learn SQL from Scratch (Jan 1, 2019-Mar 4,2019)

Kim Youngquist

Table of Contents:

1. Get familiar with Codeflix:

A. How many months has the company been operating?

ANSWER : The first subscription started on December 1st 2016 and the last start date was March 30th, found by using the MIN and MAX aggregate functions. This gives a total of four months - December 2016, January, February and March of 2017.

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

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

B. Which months will you be able to calculate churn for?

ANSWER : Codeflix requires a minimum subscription of 31 days. We see the first subscription start date is December 1st (from above) and there are no cancellations (no subscription end dates) for December and the fact that there were no subssription start dates prior to December, then we would only be finding churn rates for January, February and March 2017. Sample results in table to right showing zero subscription end dates for the month of December and then moving on to January.

```
SELECT DISTINCT subscription_end
FROM subscriptions
ORDER BY 1;
```

subscription_end
0
2017-01-01
2017-01-06
2017-01-08

1. Get familiar with Codeflix….*continued*

C. What segments of users exist?

ANSWER: There are two segments, 30 and 87. By using the distinct function

```
SELECT DISTINCT segment
FROM subscriptions
ORDER BY 1;
```

segment
30
87

2. What is the overall churn trend since the company started?

In order to calculate the churn rate, we need to know the total cancelations for the each of the three months, and also the total subscribers (active users) at the beginning of each month. The only information we currently have is from the subscribers table that consists of fours columns (below) which contains the user ids, subscription start dates, subscription end dates and the segment for each user. However, we can create temporary tables, as part of the query, in order to find the status of each user and from there calculate the churn rate per month.

I will walk through the steps I used to find the churn rates per month:

Steps 1, 2 & 3 – Define each user's status per month; active or canceled

Step 4 – Calculating total active and total canceled users per month

Step 5 – Calculating the churn rates for each month

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

```
SELECT *
FROM subscriptions
Limit 100;
```

2. What is the overall churn trend since the company started?

...continued

STEP 1: I first created a temporary months table. I will need this to compare against the subscription start and ends dates to help define active and inactive status for each user later.

first_day	last_day
2017-01-01	2017-01-31
2017-02-01	2017-02-28
2017-03-01	2017-03-31

```
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)
SELECT *
FROM months;
```

2. What is the overall churn trend since the company started?

...continued

STEP 2: I then added to the previous query and created another temporary table called cross_join (in bold) which joins four columns from the subscriptions table and two columns from the above temporary month table. I need to tie the first and last days of month(s) to the users information (id, Subscription start and end dates) in order to assign a status for each month, which I will do in the next step. I added a select statement so I could look a the current results (small sample):

```
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)
SELECT *
FROM cross_join
LIMIT 200;
```

id	subscription_start	subscription_end	segment	first_day	last_day
1	2016-12-01	2017-02-01	87	2017_01_01	2017_01_31
1	2016-12-01	2017-02-01	87	2017_02_01	2017_02_28
1	2016-12-01	2017-02-01	87	2017_03_01	2017_03_31

2. What is the overall churn trend since the company started?

...continued

STEP 3: I then created another temporary table to define the status for each user; if they were active or if they cancelled their subscription, for each month. In order to count as active, the user needs to have a subscription start date prior to the first day of the particular 'month' and also that they did not cancel within that same month. In order to count as canceled for the month, the subscription end date needs to be between the first and last day of the month. For each status, if the statement was true, a 1 was given for that month. If the statement was false, a 0 was given. I added the subscription start and ends dates to this temporary table, and then also a SELECT statement at the end, so I could spot check the query was working correctly. Below is a sampling of results. Here we see for user id 1, they were considered ACTIVE for the month of January (subscription start date was in December) but then they were considered CANCELED in February (subscription end date in February). Since they canceled in February, notice then March is populated with 0 for both.

id	Subscription_start	Subscription_end	month	is_active	is_canceled
1	2016-12-01	2017-02-01	2017-01-01	1	0
1	2016-12-01	2017-02-01	2017-02-01	0	1
1	2016-12-01	2017-02-01	2017-03-01	0	0

```
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,
       Subscription_start,
       Subscription_end,
       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)
SELECT *
FROM STATUS
LIMIT 200;
```

2. What is the overall churn trend since the company started?

...continued

STEP 4: Now that we have the status of each user for each month, I can find the total active and canceled users for each month. Adding on to the query, I created another temporary table called status_aggregate where I summed all the active users and all the ones that canceled during each month. I added a select statement so I could see the result (below).

month	sum_active	sum_canceled
2017-01-01	569	92
2017-02-01	980	186
2017-03-01	1247	342

```
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 'sum_active',
        SUM(is_canceled) AS
'sum_canceled'
   FROM status
   GROUP BY month)
SELECT *
FROM status_aggregate;
```

2. What is the overall churn trend since the company started?

...continued

STEP 5: All that is left is to calculate the churn rate for each month, which is :

$$\frac{\text{Total Canceled}}{\text{Total Active}}$$

Here we can see, since the company has started, the churn rate has increased each month; from 16% in January to 27% of users canceling in March, an 11% increase. Let's take a look at the churn rate per segment to see which segment is performing better, resulting in a lower churn rate.

month	churn_rate
2017-01-01	0.16
2017-02-01	0.19
2017-03-01	0.27

```
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 'sum_active',
       SUM(is_canceled) AS
'sum_canceled'
  FROM status
  GROUP BY month)
SELECT month,
     round
(1.0*sum_canceled/sum_active,2) AS
churn_rate
FROM status_aggregate;
```

3. Compare the churn rates between user segments

Finding active and canceled status per user, per segment, per month :

In order to compare the churn rates between user segments, I went back to the query I used in STEP 3, and created two more case statements, one for defining active and one for cancelations , and then added another argument for each case statement if the user was in segment 87 or 30.

I added a select statement to look at the results again. Now I have an active and cancelation status for both segments, 87 and 30, for each user, per month.

id	Subscription_start	Subscription_end	month	is_active_87	is_active_30	is_canceled_87	is_canceled_30
1	2016-12-01	2017-02-01	2017-01-01	1	0	0	0
1	2016-12-01	2017-02-01	2017-02-01	0	0	1	0
1	2016-12-01	2017-02-01	2017-03-01	0	0	0	0

```
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,
       Subscription_start,
       Subscription_end,
       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)
SELECT *
FROM STATUS
LIMIT 200;
```

3. Compare the churn rates between user segments...*continued*

Finding <u>total active</u> and <u>total canceled</u> status <u>per segment</u>, <u>per month</u> :
I now need to sum the status for each segment, similar to what I did in step 4.

I added a select statement to look at the results (1st table below).

I also went back to the results from Step 4 (2nd table), for the overall totals, and did a check that now that I split into segments the two add up to the overall total for each active and canceled status (3rd table).

month	sum_active_87	sum_active_30	sum_canceled_87	sum_canceled_30
2017-01-01	278	291	70	22
2017-02-01	462	518	148	38
2017-03-01	531	716	258	84

month	sum_active	sum_canceled
2017-01-01	569	92
2017-02-01	980	186
2017-03-01	1247	342

sum_active_87	sum_active_30	Total	sum_canceled_87	sum_canceled_30	Total
278	**291**	**569**	**70**	**22**	**92**
462	518	**980**	148	38	**186**
531	716	**1247**	258	84	**342**

```
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 = 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 *
FROM status_aggregate;
```

3. Compare the churn rates between user segments...*continued*

Churn rates <u>per segment</u>, <u>per month</u>:

Breaking the churn rates out by segment, we see segment 87 has a much higher churn rate each month than segment 30. Also, it's interesting to see segment 30's rate actually decreases in February. By March, segment 87 has close to a 49% churn rate while segment 30 is just under a 12% churn rate; that is about a 23% increase for segment 87 and only a 4% increase for segment 30 from January.

I would recommend the company continue with their strategy for segment 30 and discontinue their strategy for segment 87.

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 = 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,
   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;
```