codecademy

Capstone: Churn Rates

Learn SQL from Scratch

Wesley Yang | 2018.07.10

Table of Contents

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

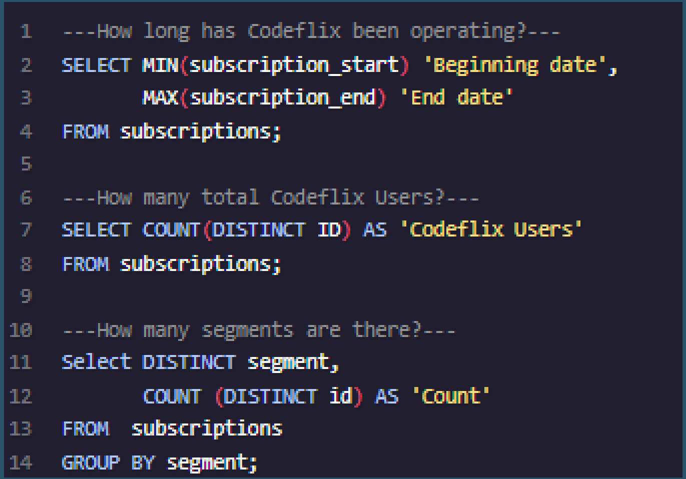
Get familiar with the company

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)



```sql
---How long has Codeflix been operating?---
SELECT MIN(subscription_start) 'Beginning date',
       MAX(subscription_end) 'End date'
FROM subscriptions;

---How many total Codeflix Users?---
SELECT COUNT(DISTINCT ID) AS 'Codeflix Users'
FROM subscriptions;

---How many segments are there?---
Select DISTINCT segment,
       COUNT (DISTINCT id) AS 'Count'
FROM  subscriptions
GROUP BY segment;
```

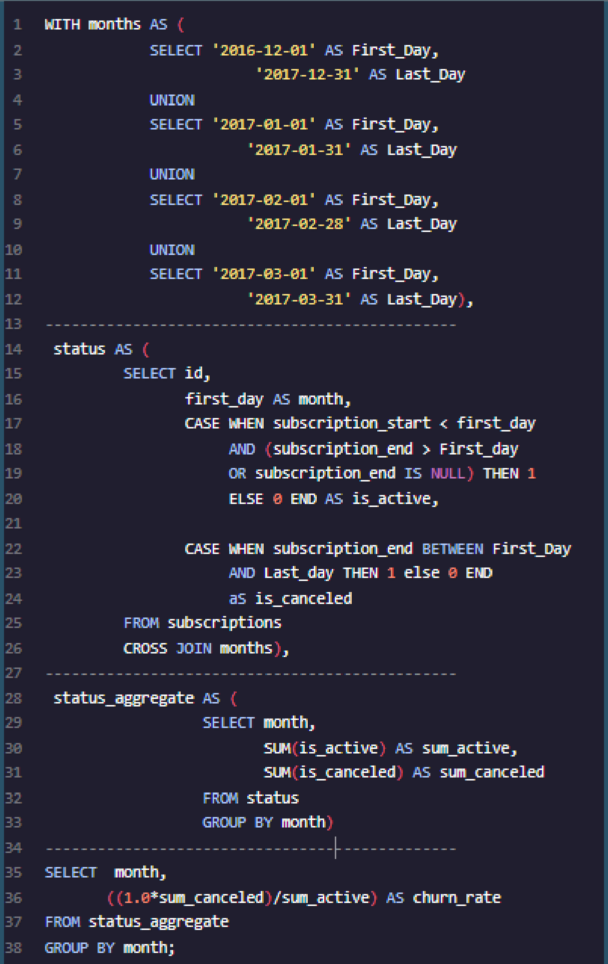
Begining date	End date
2016-12-01	2017-03-31
Codeflix Users	
2000	
segment	**Count**
30	1000
87	1000

2. What is the overall churn rate by month?

What is the overall churn trend since the company started?

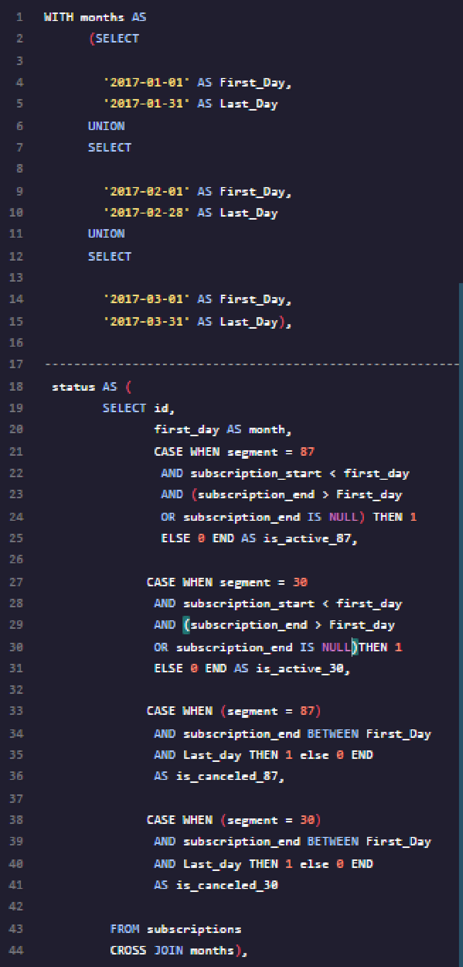
- Three tables were created by using the common table expression at the (months, status and status_ aggregate) at the beginning so that the final query is readable.
  - -months was created for the calendar date ranges
  - -status was used to determine the user’s status by month and to cross join the 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%

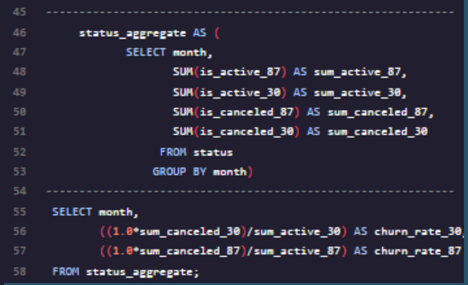


```sql
WITH months AS (
            SELECT '2016-12-01' AS First_Day,
                        '2017-12-31' AS Last_Day
            UNION
            SELECT '2017-01-01' AS First_Day,
                       '2017-01-31' AS Last_Day
            UNION
            SELECT '2017-02-01' AS First_Day,
                       '2017-02-28' AS Last_Day
            UNION
            SELECT '2017-03-01' AS First_Day,
                       '2017-03-31' AS Last_Day),
-----------------------------------------------
 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 subscriptions
         CROSS JOIN months),
-----------------------------------------------
 status_aggregate AS (
                  SELECT month,
                         SUM(is_active) AS sum_active,
                         SUM(is_canceled) AS sum_canceled
                  FROM status
                  GROUP BY month)
-----------------------------------------------
SELECT  month,
       ((1.0*sum_canceled)/sum_active) AS churn_rate
FROM status_aggregate
GROUP BY month;
```

3. Compare the churn rates between segments



```sql
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),

 ------------------------------------------------------------
 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 subscriptions
           CROSS JOIN months),
```



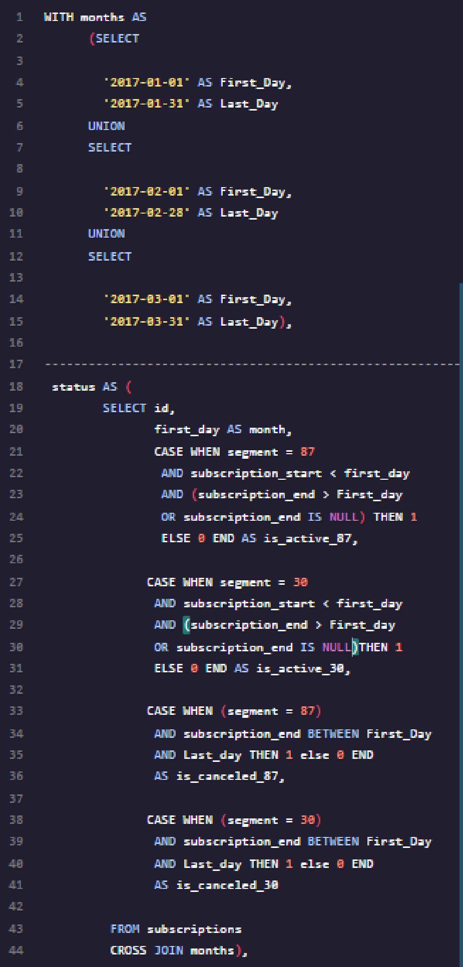
```sql
 ------------------------------------------------------------
     status_aggregate AS (
             SELECT month,
                    SUM(is_active_87) AS sum_active_87,
                    SUM(is_active_30) AS sum_active_30,
                    SUM(is_canceled_87) AS sum_canceled_87,
                    SUM(is_canceled_30) AS sum_canceled_30
                 FROM status
                GROUP BY month)
 ------------------------------------------------------------
 SELECT month,
       ((1.0*sum_canceled_30)/sum_active_30) AS churn_rate_30,
       ((1.0*sum_canceled_87)/sum_active_87) AS churn_rate_87
 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%

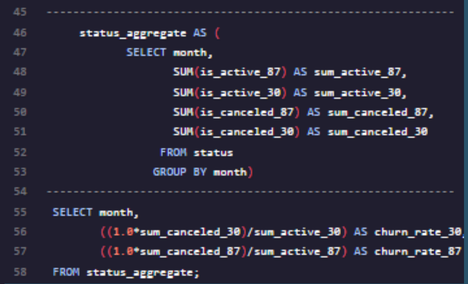
Compare the churn rates between user segments.

- 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



```sql
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),

 -----------------------------------------------------

 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 subscriptions
         CROSS JOIN months),
```



```sql
-----------------------------------------------------------
    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_30)/sum_active_30) AS churn_rate_30,
        ((1.0*sum_canceled_87)/sum_active_87) AS churn_rate_87
 FROM status_aggregate;
```

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%

Compare the churn rates between user segments.

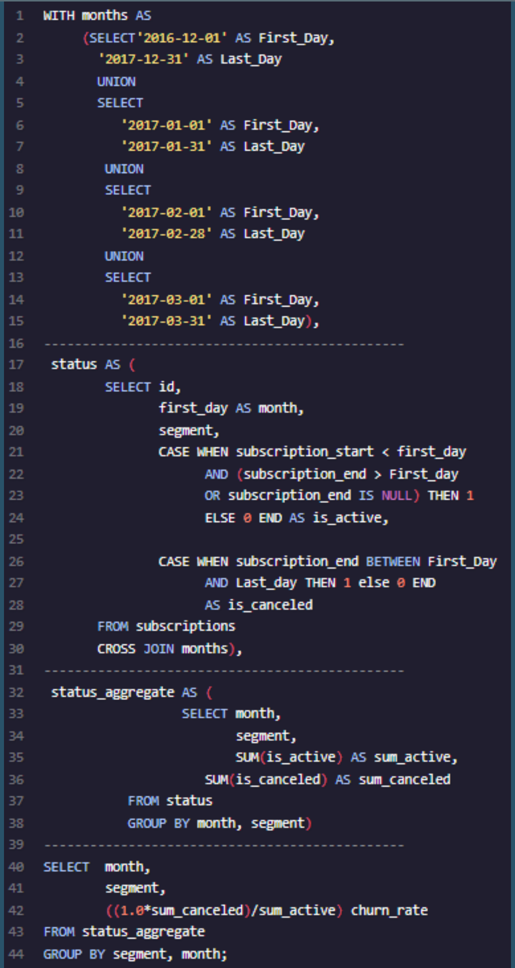
- The with statement was used to create three tables
  - -months is used for the date ranges
  - -status was used to select id and to determine active/canceled status for the two segments.
    - -The is_active case statements used a hard coded segment number with date logic to determine if the user account 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.

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%



```sql
WITH months AS
     (SELECT'2016-12-01' AS First_Day,
       '2017-12-31' 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),
----------------------------------------------
 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 subscriptions
       CROSS JOIN months),
----------------------------------------------
 status_aggregate AS (
                  SELECT month,
                         segment,
                         SUM(is_active) AS sum_active,
                     SUM(is_canceled) AS sum_canceled
           FROM status
           GROUP BY month, segment)
----------------------------------------------
SELECT  month,
        segment,
        ((1.0*sum_canceled)/sum_active) churn_rate
FROM status_aggregate
GROUP BY segment, month;
```