codecademy

Codeflix – Churn Rates

Learn SQL from Scratch

Table of Contents

1. Get familiar with the company

- The company has been active from December 2016 up till March of 2017. (See query 1.1)

first_customer	last_activity
2016-12-01	2017-03-30

- The months that can be analysed for churn ratios would be the first 3 months of 2017 due to no subscriptions have been ended in December 2016 (See query 1.2)

first_end
2017-01-01

- There are 2 segments used in this data set, 87 and 30, with a 50/50 split between them of 1000 each. (See query 1.3)

segment	count(*)
30	1000
87	1000

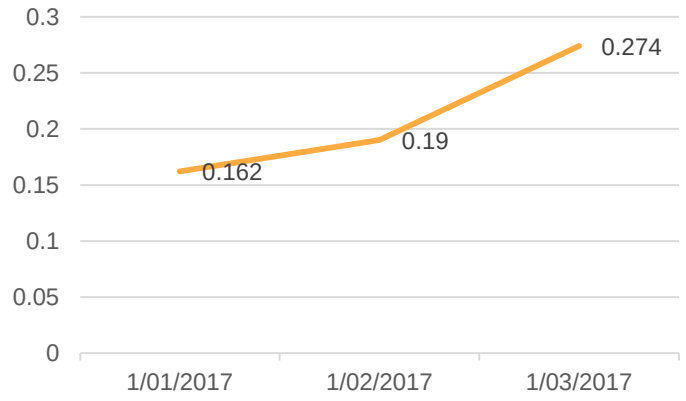
```
-- query 1.1
SELECT min(subscription_start) AS first_customer,
       max(subscription_start) AS last_activity
  FROM subscriptions;
```

```
-- query 1.2
SELECT min(subscription_end) AS first_end  FROM
subscriptions;
```

```
-- query 1.3
SELECT distinct segment,
       count(*)
  FROM subscriptions
GROUP BY segment;
```

2. Over all Churn trend

- The trend for the company so far has the churn rate increasing each month

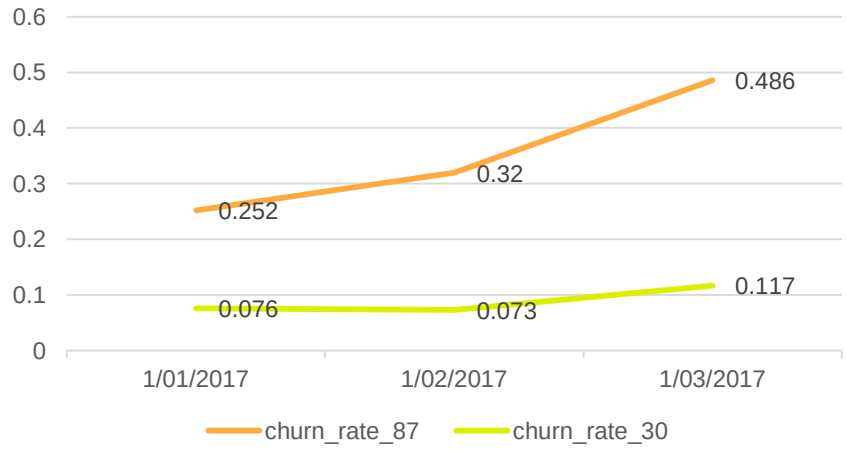



```
WITH months AS(
     SELECT '2017-01-01' AS first_day,
            '2017-01-31' AS last_day
     UNION
     SELECT '2017-02-01' AS first_day,
            '2017-02-28' AS last_day
     UNION
     SELECT '2017-03-01' AS first_day,
            '2017-03-31' AS last_day),
cross_join AS(
     SELECT *
     FROM subscriptions
     CROSS JOIN months),
status AS(
     SELECT id,
            first_day AS month,
            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 cross_join),
status_aggregate AS(
     SELECT month,
            segment,
            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, 1)  AS"churn_rate%"
   FROM status_aggregate;
```

3. Compare the churn rates between user segments

- Although both segments have an increasing churn rate, segment 87 is the most worrying trend. The cause of this increase for segment 87 would be the first area I recommend was investigated.




```
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 ASmonth,
         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_rate_87",
       1.0 * sum_canceled_30 / sum_active_30  AS "churn_rate_30"
  FROM status_aggregate ;
```

4. Extra Segments

- If extra segments were added to the data. I would modify the query as shown. This would allow for any number of segments with out modifying the query.

month	segment	churn_rate
2017-01-01	30	0.075601
2017-01-01	87	0.251799
2017-02-01	30	0.073359
2017-02-01	87	0.320346
2017-03-01	30	0.117318
2017-03-01	87	0.485876

```
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,
             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 cross_join),
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,
       round(1.0 * sum_canceled / sum_active, 6)  AS "churn_rate"
FROM status_aggregate;
```