codecademy

CAPSTONE: CODEFLIX CHURN RATES

Learn SQL From Scratch

Christian Monzon

August 2018

TABLE OF CONTENTS

This subscriber retention analysis will provide an overview of the churn rate for each month and segment of available subscription data and will provide recommendations on how to reprioritize segment growth strategies

Questions answered by this analysis:

1. Dataset overview
   - 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. Overall company churn rate

3. Churn rates by user segments.
   - Which segment of users should the company focus on expanding?

DATASET OVERVIEW

1. How many months has the company been operating?
   a) Based on the dataset, we can infer that the company has been operating for at least four months, starting December 1st 2016. A basic query selecting the minimum and maximum subscription start date gives us the workable range of subscription data.
2. Which months do you have enough information to calculate a churn rate?
   a) The minimum Codeflix subscription length is 31 days, which limits the analysis to observe only three of the four months of data since there can't be any cancellations in the first month (December).
3. What segments of users exist?
   a) There are two distinct user segments: 87 and 30

Subscriptions Table Schema		
Column Name	Column Description	Data Type
id	Subscription id	Integer
subscription_start	Start date of subscription	Date
subscription_end	End date of subscription	Date
segment	Subscription segment	Integer

```
--Identifying range of subscription data
SELECT min(subscription_start) AS Min,
       max(subscription_start) AS Max
FROM subscriptions;

--Identifying distinct user segments
SELECT DISTINCT segment AS Distinct_User_Segments
FROM subscriptions;
```

Query Results	
Min	Max
2016-12-01	2017-03-30
Distinct_User_Segments	
87	
30	

OVERALL COMPANY CHURN RATE – CONT'D

c) The monthly churn needed to be calculated to understand the company churn trend. To isolate the monthly churn, a temporary "months" table was created.

```
--Temporary "months" table
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
),
```

Note that December was excluded since subscribers cannot cancel within 31 days of starting their subscription. Cancellations dates are required to calculate a churn rate (cancellations / total subs)

d) With the "months" table created, a cross join was performed to create another temporary table("cross_join") containing all of the possible combinations between months and subscriptions. "cross_join" served as the table upon which the subscription statuses could be calculated.

```
--Temp cross join table
---of months and subscriptions tables
cross_join AS
(SELECT *
FROM subscriptions
CROSS JOIN months),
```

OVERALL COMPANY CHURN RATE – CONT'D

e) Active and cancelled subscribers were identified by comparing the subscription_start and first_day columns of the temporary cross_join table.

```
--Temp table active and cancelled subscribers
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
),
```

f) Finally, the total number of active and canceled subscribers were summed and the churn rate for each month was calculated.

```
--SUM active and canceled subscribers
status_aggregate AS (
  SELECT
    month,
    SUM(is_active) AS active,
    SUM(is_canceled) AS canceled
  FROM status
  GROUP BY month
)
SELECT
  month,
  100* (1.0 * canceled / active) AS churn_rate
FROM status_aggregate;
```

OVERALL COMPANY CHURN RATE – CONT'D

The company churn rates for each month are as follows:

month	churn_rate
January 2017	16.17%
February 2017	18.98%
March 2017	27.423%

Codeflix's churn rates for the company have increased since the launch and drastically increased in March. Since January, there has been ~70% increase in the churn rate. It is recommended that the business reevaluate any significant business decisions that were made between February and March that may have influenced the dramatic change in churn rate.

CHURN RATES BY USER SEGMENTS

As identified earlier in the analysis, there are two distinct user segments identified by the segment column: 87 and 30. Although the overall company churn rate was found to be increasing, evaluating the user groups by segments proved to be more insightful. By using similar query structure as reviewed in the previous section and isolating for the distinct user segments, it was found that user segment 87 experienced significantly higher churn rates than user segment 30.

The average churn rate was not only higher for user segment 87, but also experienced a steady increase from January to March. User segment 30, however, experienced a slight drop in churn rate during the month of February.

When comparing the number of active and canceled subscribers for segment by month, user segment 87 is simply adding fewer subscribers each month and experiencing higher cancelations, therefore experiencing a higher churn rate than user segment 30. It is recommended that Codeflix prioritize growth and retention strategies for user segment 87 to decrease future churn rates. It is also recommended that any growth and retention strategies currently employed by user segment 30 should be immediately applied to user group 87 to ideally follow similar growth trends.

Month	churn_rate_87	churn_rate_30	Active87	Active30	Canceled87	Canceled30
January 2017	25.18%	7.56%	278	291	70	22
February 2017	32.03%	7.34%	462	518	148	38
March 2017	48.56%	11.73%	531	716	258	84
Average	35.26%	8.88%	424	508	159	48

BONUS

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

You would create a temporary table that retrieves all of the unique segment values and then call the temporary table in the calculations of the subscriber status.