Churn rates

Data analysis with SQL

Ekaterina Chernyakova

22/05/2020

Preview

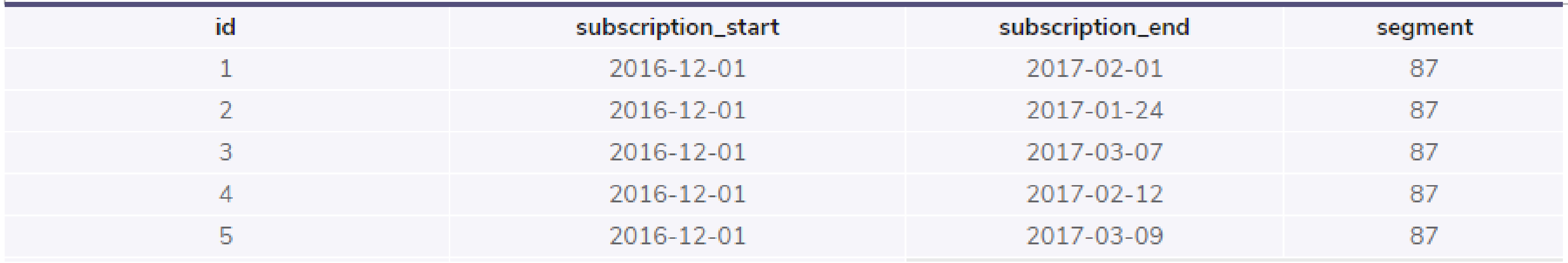
- Four months into launching Codeflix, management asks me to look into subscription churn rates. It’s early on in the business and people are excited to know how the company is doing.
- The marketing department is particularly interested in how the churn compares between two segments of users. They provide me with a dataset containing subscription data for users who were acquired through two distinct channels.
- Codeflix requires a minimum subscription length of 31 days, so a user can never start and end their subscription in the same month.

Table of Contents

1.1 Inspecting the data

The dataset that was provided to me contains one SQL table called subscriptions. Within the table, there are 4 columns:

- `id` - the subscription id
- `subscription start` - the start date of the subscription
- `subscription end` - the end date of the subscription
- `segment` - this identifies which segment the subscription owner belongs to

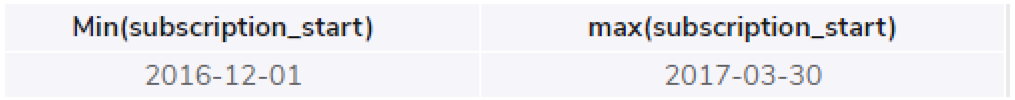


id	subscription_start	subscription_end	segment
1	2016-12-01	2017-02-01	87
2	2016-12-01	2017-01-24	87
3	2016-12-01	2017-03-07	87
4	2016-12-01	2017-02-12	87
5	2016-12-01	2017-03-09	87

1.2 Inspecting the data

Determining the range of months of data provided and which months I will be able to calculate churn for.

I'll be calculating the churn rate for both segments (87 and 30) over the first 3 months of 2017 (I can't calculate it for December, since there are no `subscription_end` values yet)



Min(subscription_start)	max(subscription_start)
2016-12-01	2017-03-30

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

2.1 Calculating the churn rates

1. Creating a temporary table months containing the information on the first and last day of each month.
2. Creating a temporary table, `cross_join`, from subscriptions and months.

```
with months as(
 select
   '2017-01-01' as first_day,
   '2017-01-31' as last_day
    from subscriptions
  UNION
   select
   '2017-02-01' as first_day,
   '2017-02-31' as last_day
    from subscriptions
  Union
   select
   '2017-03-01' as first_day,
   '2017-03-31' as last_day
   from subscriptions ),

   cross_join as (
    select * from subscriptions
    cross join months),
```

2.2 Calculating the churn rates

Creating a temporary table, status, from the `cross_join` table I created. This table will contain:

•`id` selected from `cross_join`

•`month` as an alias of `first_day`

•`is_active_87` created using a `CASE WHEN` to find any users from segment 87 who existed prior to the beginning of the month. This is 1 if true and 0 otherwise.

•`is_active_30` created using a `CASE WHEN` to find any users from segment 30 who existed prior to the beginning of the month. This is 1 if true and 0 otherwise.

```
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),
```

2.3 Calculating the churn rates

Creating a `status_aggregate` temporary table that is a SUM of the active and canceled subscriptions for each segment, for each month.

The resulting columns will be:

- `sum_active_87`
- `sum_active_30`
- `sum_canceled_87`
- `sum_canceled_30`

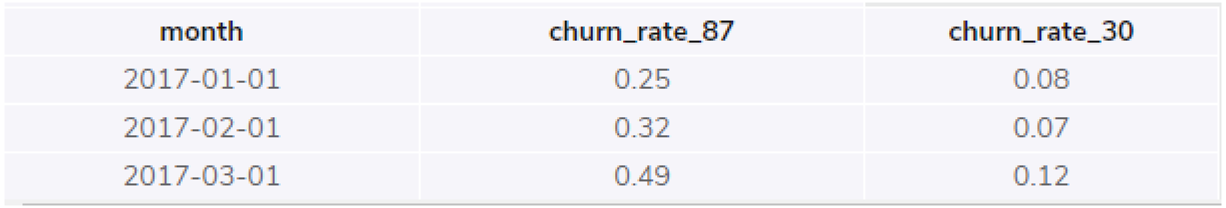
```
status_aggregate as (
  select month,
     sum(is_active_87) as 'sum_active_87',
     sum(is_active_30) as 'sum_active_30',
     sum(is_canceled_87) as 'sum_canceled_87',
     sum(is_canceled_30) as 'sum_canceled_30'
   from status
   Group by 1)
```

3 Churn rates results

Calculating the churn rates for the two segments over the three month period.

As we can see, segment 30 has a lower churn rate for each month.

```
SELECT month,
    round((1.0 *sum_canceled_87 / sum_active_87), 2)
AS 'churn_rate_87',
    round((1.0 *sum_canceled_30 / sum_active_30), 2)
AS 'churn_rate_30’
FROM status_aggregate;
```



month	churn_rate_87	churn_rate_30
2017-01-01	0.25	0.08
2017-02-01	0.32	0.07
2017-03-01	0.49	0.12