codecademy

Learn SQL from Scratch

Therese Perino

November 5, 2018

Calculating Churn Rates

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?

Step 1

“Get familiar with the data”

```
SELECT *
FROM subscriptions
LIMIT 100;
```

Columns include:
id
subscription_start
subscription_end
Segment

Looks like they have two different types of segments for acquiring subscriptions, 30 & 87.

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
6	2016-12-01	2017-01-19	87
7	2016-12-01	2017-02-03	87
8	2016-12-01	2017-03-02	87
9	2016-12-01	2017-02-17	87
10	2016-12-01	2017-01-01	87
11	2016-12-01	2017-01-17	87
12	2016-12-01	2017-02-07	87
13	2016-12-01		30
14	2016-12-01	2017-03-07	30
15	2016-12-01	2017-02-22	30
16	2016-12-01		30
17	2016-12-01		30
18	2016-12-02	2017-01-29	87
19	2016-12-02	2017-01-13	87
20	2016-12-02	2017-01-15	87
21	2016-12-02	2017-01-15	87
22	2016-12-02	2017-01-24	87
23	2016-12-02	2017-01-14	87
24	2016-12-02	2017-01-18	87

STEP 2

"Determine the range of months of data provided."

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

Subscription range is from Dec 1, 2016 to March 31, 2017. We find this data by selecting minimum, "min" subscription_start which shows the earliest start date included in table. Maximum, "max" subscription_end, shows the last end date of subscriptions included in the table

So from that we are only able to determine the churn rate for 3 months, Jan, Feb, and March. We don't include Dec 2016 because you need to run subscription for 31 days we do not want to include Dec for subscription start.

Query Results

MIN(subscription_start)	MAX(subscription_end)
2016-12-01	2017-03-31

STEP 3

"You'll be calculating the churn rate for both segments (87 and 30) over the first 3 months of 2017."

We make a temporary table, using the command "With", which you use for creating your first temporary table. Alias "month" using the command "AS", selecting first and last days of Jan, Feb and March. We use the command "union" to show begin and end dates of subscriptions of months chosen. Union allows me to stack 2 or more table on top of each other.
I put limit at 10 to keep table manageable.

```
WITH month 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
 FROM subscriptions)
 SELECT *
 FROM subscriptions
 LIMIT 10;
```

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
6	2016-12-01	2017-01-19	87
7	2016-12-01	2017-02-03	87
8	2016-12-01	2017-03-02	87
9	2016-12-01	2017-02-17	87
10	2016-12-01	2017-01-01	87

STEP 4

"Create a temporary table, cross_join, from subscriptions and your months. "

With "cross join" we combine all rows from table "subscriptions" and table "month". Results show id, subscription_start, subscription_end, segment, first_day, and last_day.

```
cross_join AS (
                    SELECT *
  FROM subscriptions
   CROSS JOIN month)
 SELECT *
 FROM cross_join
 LIMIT 10;
```

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	2016-12-01	2017-01-24	87	2017-01-01	2017-01-31
2	2016-12-01	2017-01-24	87	2017-02-01	2017-02-28
2	2016-12-01	2017-01-24	87	2017-03-01	2017-03-31
3	2016-12-01	2017-03-07	87	2017-01-01	2017-01-31
3	2016-12-01	2017-03-07	87	2017-02-01	2017-02-28
3	2016-12-01	2017-03-07	87	2017-03-01	2017-03-31
4	2016-12-01	2017-02-12	87	2017-01-01	2017-01-31

STEP 5

Create a temporary table, status.

```
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)
SELECT *
FROM cross_join
limit 50;
```

id	month	is_active_87	is_active_30
1	2017-01-01	1	0
1	2017-02-01	0	0
1	2017-03-01	0	0
2	2017-01-01	1	0
2	2017-02-01	0	0
2	2017-03-01	0	0
3	2017-01-01	1	0
3	2017-02-01	1	0
3	2017-03-01	1	0
4	2017-01-01	1	0
4	2017-02-01	1	0
4	2017-03-01	0	0
5	2017-01-01	1	0
5	2017-02-01	1	0
5	2017-03-01	1	0
6	2017-01-01	1	0
6	2017-02-01	0	0
6	2017-03-01	0	0
7	2017-01-01	1	0
7	2017-02-01	1	0
7	2017-03-01	0	0

As you see it breaks down subscriptions into id, then first 3 months and separates into active 87 and 30 segments. You can see we don't include the command "WITH" to create this temporary table because this is not the first temporary table we have created. We also added 2 additional columns "is_active_87" and "is_active_30" to this table using the "CASE" command. The case command allows use several conditions to organize our data. In this "CASE" we need subscriptions that are still active "AND" in segments 87 or 30.

Step 6

Add an “is_canceled_87” and “is_canceled_30” row to temporary “status” table

```
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 50;
```

id	month	is_active_87	is_active_30	is_canceled_87	is_canceled_30
1	2017-01-01	1	0	0	0
1	2017-02-01	0	0	0	0
1	2017-03-01	0	0	1	1
2	2017-01-01	1	0	0	0
2	2017-02-01	0	0	1	1
2	2017-03-01	0	0	1	1
3	2017-01-01	1	0	0	0
3	2017-02-01	1	0	0	0
3	2017-03-01	1	0	0	0
4	2017-01-01	1	0	0	0
4	2017-02-01	1	0	0	0
4	2017-03-01	0	0	1	1
5	2017-01-01	1	0	0	0
5	2017-02-01	1	0	0	0
5	2017-03-01	1	0	0	0
6	2017-01-01	1	0	0	0
6	2017-02-01	0	0	1	1
6	2017-03-01	0	0	1	1
7	2017-01-01	1	0	0	0

We have added the is_canceled_87 and is_canceled_30 columns to our temporary table “status” which helps us see which type of subscriptions are still active or canceled in each month and segment. To do this we add a comma at the end of the last case and continue to add these two case commands to add the additional columns to our table.

Step 7

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

```
status_aggregate AS
( SELECT
 month,
 sum(is_active_87) AS sum_actve_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;
```

month	sum_actve_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

This table shows the total active and canceled per month of the 87 and 30 segments. By using the command “sum” we add the individual columns we created in our temporary table “status” and renamed the totals sum columns as sum _active_&& or sum_canceled_&&. We also wanted to group the results by month, by adding “Group BY command you can see is show sum for each in a monthly format. So just by a quick look at this table you can see that in the month of January we have 278 active with 70 canceled in segment 80 yet in segment 30 we have 291 active with only 22 canceled. This shows a quick overview of each segment in a particular month.

Step 8

Calculate the churn rates for the two segments over the three month period.
Which segment has a lower churn rate?

```
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;
```

month	churn_rate_87	churn_rate_30
2017-01-01	0.251798561151079	0.0756013745704467
2017-02-01	0.32034632034632	0.0733590733590734
2017-03-01	0.485875706214689	0.11731843575419

When finding the churn rate you want to divide cancellation by total subscriptions and then multiply by 1.0 to get proper float. As you see from command we took sum_canceled_87 divided by sum_active_87 then multiplied by 1.0, doing the same with segment 30. We also selected month so we could get an accurate month by month churn rate.
It looks like in Jan. segment 30 with a .0756 churn rate compared to segment 80's .25179 churn rate. Each month show that segment 30 is more successful than segment 80. Whatever segment 30 is doing for advertising or exposure is much better than segment 80

Step 9

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

It tells you in the hint to avoid hard coding and in the video it tells you about adding a segment column and group by month and segment. I could act like I didn’t watch that or look at the hint but won’t. I was thinking on a larger scale you could look at this and block it off in quarters. You would find the information from a quarter would be more helpful for this type business than monthly.