

Calculating Churn Rates for subscriptions : 2 user segments

Gabriel Pena

```

1  SELECT *
2  FROM subscriptions
3  LIMIT 100;

```

Here we are getting familiar with the table but still limiting the query to give us 100 rows back as we are still getting a feel for the rows, columns and values.

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

89	2016-12-05	∅	30
90	2016-12-06	2017-02-25	87
91	2016-12-06	2017-03-14	87
92	2016-12-06	2017-02-22	87
93	2016-12-06	2017-02-05	87
94	2016-12-06	2017-01-28	87
95	2016-12-06	2017-02-03	87
96	2016-12-06	2017-02-20	87
97	2016-12-06	2017-03-12	87
98	2016-12-06	2017-03-05	87
99	2016-12-06	∅	30
100	2016-12-06	2017-03-11	30

Database Schema

subscriptions		2000 rows
id	INTEGER	
subscription_start	TEXT	
subscription_end	TEXT	
segment	INTEGER	

This will show us the range of months of data in the table by asking for the earliest start date recorded and latest start date recorded. We can use these months to calculate for churn because the values in the table will only be from 2016-12-01 through 2017-03-30

```
5  SELECT MIN(subscription_start),  
6         MAX(subscription_start)  
7  FROM subscriptions;
```

MIN(subscription_start)	MAX(subscription_start)
2016-12-01	2017-03-30

Database Schema

Here we are creating a temporary table of months that make up the first 3 months of 2017. We do not include December in this temporary table because the month of December does not have subscription end dates since a user can not start and end a subscription on the same month. We will need this first to begin calculating the churn rate for the two segments listed as 87 and 30.

```
9 WITH months AS
10 (SELECT
11     '2017-01-01' AS first_day,
12     '2017-01-31' AS last_day
13 UNION
14     SELECT
15     '2017-02-01' AS first_day,
16     '2017-02-28' AS last_day
17 UNION
18     SELECT
19     '2017-03-01' AS first_day,
20     '2017-03-31' AS last_day)
21
22 SELECT *
23 FROM months;
```

first_day	last_day
2017-01-01	2017-01-31
2017-02-01	2017-02-28
2017-03-01	2017-03-31

```

25 WITH months AS
26 (SELECT
27     '2017-01-01' AS first_day,
28     '2017-01-31' AS last_day
29 UNION
30     SELECT
31     '2017-02-01' AS first_day,
32     '2017-02-28' AS last_day
33 UNION
34     SELECT
35     '2017-03-01' AS first_day,
36     '2017-03-31' AS last_day),
37
38 cross_join AS (
39     SELECT *
40     FROM subscriptions
41     CROSS JOIN months)
42
43     SELECT *
44     FROM cross_join
45     LIMIT 30;

```

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
4	2016-12-01	2017-02-12	87	2017-02-01	2017-02-28
4	2016-12-01	2017-02-12	87	2017-03-01	2017-03-31
5	2016-12-01	2017-03-09	87	2017-01-01	2017-01-31
5	2016-12-01	2017-03-09	87	2017-02-01	2017-02-28
5	2016-12-01	2017-03-09	87	2017-03-01	2017-03-31
6	2016-12-01	2017-01-19	87	2017-01-01	2017-01-31
6	2016-12-01	2017-01-19	87	2017-02-01	2017-02-28
6	2016-12-01	2017-01-19	87	2017-03-01	2017-03-31
7	2016-12-01	2017-02-03	87	2017-01-01	2017-01-31
7	2016-12-01	2017-02-03	87	2017-02-01	2017-02-28
7	2016-12-01	2017-02-03	87	2017-03-01	2017-03-31
8	2016-12-01	2017-03-02	87	2017-01-01	2017-01-31
8	2016-12-01	2017-03-02	87	2017-02-01	2017-02-28
8	2016-12-01	2017-03-02	87	2017-03-01	2017-03-31
9	2016-12-01	2017-02-17	87	2017-01-01	2017-01-31
9	2016-12-01	2017-02-17	87	2017-02-01	2017-02-28
9	2016-12-01	2017-02-17	87	2017-03-01	2017-03-31
10	2016-12-01	2017-01-01	87	2017-01-01	2017-01-31
10	2016-12-01	2017-01-01	87	2017-02-01	2017-02-28
10	2016-12-01	2017-01-01	87	2017-03-01	2017-03-31

This syntax will join the results for the temporary 'months' table with all the values in the original table 'subscriptions' using a CROSS JOIN command. We have selected that all values from both tables are joined with one another. Due to the amount of values the tables contain we have limited the query to only return 30 rows, with no specific order of values.

```

47 WITH months AS
48 (SELECT
49   '2017-01-01' AS first_day,
50   '2017-01-31' AS last_day
51 UNION
52  SELECT
53   '2017-02-01' AS first_day,
54   '2017-02-28' AS last_day
55 UNION
56  SELECT
57   '2017-03-01' AS first_day,
58   '2017-03-31' AS last_day),
59
60 cross_join AS (
61  SELECT *
62  FROM subscriptions
63  CROSS JOIN months),
64
65 status AS(
66  SELECT id,
67         first_day AS month,
68         CASE WHEN segment = 87
69              AND (subscription_start < first_day)
70              AND (subscription_end > first_day
71                  OR subscription_end IS NULL)
72              THEN 1
73              ELSE 0
74         END as is_active_87,
75         CASE WHEN segment = 30
76              AND(subscription_start < first_day)
77              AND (subscription_end > first_day
78                  OR subscription_end IS NULL)
79              THEN 1
80              ELSE 0
81         END as is_active_30
82  FROM cross_join)
83
84  SELECT *
85  FROM status
86  LIMIT 50;

```

Here we have created another temporary table, 'status', that will return a query with all the users from segment 87 and segment 30 who existed prior to the begging of the month. This info will be taken from the previous table we created 'cross-join'. We use a CASE statement for segment 87 marking all users with a 1 who started before the first day of a month and have an end date or no end date after the first day of a month. We do the same for segment 30.

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
8	2017-01-01	1	0
8	2017-02-01	1	0
8	2017-03-01	1	0
9	2017-01-01	1	0
9	2017-02-01	1	0
9	2017-03-01	0	0
10	2017-01-01	0	0
10	2017-02-01	0	0
10	2017-03-01	0	0
11	2017-01-01	1	0
11	2017-02-01	0	0
11	2017-03-01	0	0
12	2017-01-01	1	0
12	2017-02-01	1	0
12	2017-03-01	0	0
13	2017-01-01	0	1
13	2017-02-01	0	1
13	2017-03-01	0	1
14	2017-01-01	0	1
14	2017-02-01	0	1
14	2017-03-01	0	1
15	2017-01-01	0	1
15	2017-02-01	0	1
15	2017-03-01	0	0
16	2017-01-01	0	1
16	2017-02-01	0	1
16	2017-03-01	0	1
17	2017-01-01	0	1
17	2017-02-01	0	1

Using the previous syntax we will be adding two more CASE statement columns that return a query containing the users who canceled during a month. To find this we will be using the BETWEEN command to mark all users with a 1 that canceled between the first and last day of each month for segment 87 and the same thing for segment 30.

```

88 WITH months AS
89   (SELECT
90     '2017-01-01' AS first_day,
91     '2017-01-31' AS last_day
92   UNION
93     SELECT
94     '2017-02-01' AS first_day,
95     '2017-02-28' AS last_day
96   UNION
97     SELECT
98     '2017-03-01' AS first_day,
99     '2017-03-31' AS last_day),
100
101 cross_join AS (
102   SELECT *
103   FROM subscriptions
104   CROSS JOIN months),
105

```

```

106   status AS(
107   SELECT id,
108         first_day AS month,
109         CASE WHEN segment = 87
110           AND (subscription_start < first_day)
111           AND (subscription_end > first_day
112             OR subscription_end IS NULL)
113         THEN 1
114         ELSE 0
115       END as is_active_87,
116         CASE WHEN segment = 30
117           AND(subscription_start < first_day)
118           AND (subscription_end > first_day
119             OR subscription_end IS NULL)
120         THEN 1
121         ELSE 0
122       END as is_active_30,
123         CASE WHEN segment = 87
124           AND subscription_end BETWEEN first_day AND
125           last_day THEN 1
126         ELSE 0
127       END as is_canceled_87,
128         CASE WHEN segment = 30
129           AND subscription_end BETWEEN first_day AND
130           last_day THEN 1
131         ELSE 0
132       END as is_canceled_30
133   FROM cross_join)
134   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	1	0
1	2017-03-01	0	0	0	0
2	2017-01-01	1	0	1	0
2	2017-02-01	0	0	0	0
2	2017-03-01	0	0	0	0
3	2017-01-01	1	0	0	0
3	2017-02-01	1	0	0	0
3	2017-03-01	1	0	1	0
4	2017-01-01	1	0	0	0
4	2017-02-01	1	0	1	0
4	2017-03-01	0	0	0	0
5	2017-01-01	1	0	0	0
5	2017-02-01	1	0	0	0
5	2017-03-01	1	0	1	0
6	2017-01-01	1	0	1	0
6	2017-02-01	0	0	0	0
6	2017-03-01	0	0	0	0
7	2017-01-01	1	0	0	0
7	2017-02-01	1	0	1	0
7	2017-03-01	0	0	0	0
8	2017-01-01	1	0	0	0
8	2017-02-01	1	0	0	0
8	2017-03-01	1	0	1	0
9	2017-01-01	1	0	0	0
9	2017-02-01	1	0	1	0
9	2017-03-01	0	0	0	0
10	2017-01-01	0	0	1	0
10	2017-02-01	0	0	0	0
10	2017-03-01	0	0	0	0
11	2017-01-01	1	0	1	0
11	2017-02-01	0	0	0	0
11	2017-03-01	0	0	0	0
12	2017-01-01	1	0	0	0
12	2017-02-01	1	0	1	0
12	2017-03-01	0	0	0	0
13	2017-01-01	0	1	0	0
13	2017-02-01	0	1	0	0
13	2017-03-01	0	1	0	0
14	2017-01-01	0	1	0	0
14	2017-02-01	0	1	0	0
14	2017-03-01	0	1	0	1
15	2017-01-01	0	1	0	0
15	2017-02-01	0	1	0	1
15	2017-03-01	0	0	0	0
16	2017-01-01	0	1	0	0
16	2017-02-01	0	1	0	0
16	2017-03-01	0	1	0	0
17	2017-01-01	0	1	0	0
17	2017-02-01	0	1	0	0


```
136 WITH months AS
137 (SELECT
138     '2017-01-01' AS first_day,
139     '2017-01-31' AS last_day
140 UNION
141     SELECT
142     '2017-02-01' AS first_day,
143     '2017-02-28' AS last_day
144 UNION
145     SELECT
146     '2017-03-01' AS first_day,
147     '2017-03-31' AS last_day),
148
149 cross_join AS (
150     SELECT *
151     FROM subscriptions
152     CROSS JOIN months),
153
154 status AS(
155     SELECT id,
156         first_day AS month,
157         CASE WHEN segment = 87
158             AND (subscription_start < first_day)
159             AND (subscription_end > first_day
160                 OR subscription_end IS NULL)
161             THEN 1
162             ELSE 0
163         END as is_active_87,
164         CASE WHEN segment = 30
165             AND(subscription_start < first_day)
166             AND (subscription_end > first_day
167                 OR subscription_end IS NULL)
168             THEN 1
169             ELSE 0
170         END as is_active_30,
171         CASE WHEN segment = 87
172             AND subscription_end BETWEEN first_day AND
173             last_day THEN 1
174             ELSE 0
175         END as is_canceled_87,
176         CASE WHEN segment = 30
177             AND subscription_end BETWEEN first_day AND
178             last_day THEN 1
179             ELSE 0
180         END as is_canceled_30
181     FROM cross_join),
```

```
181 status_aggregate AS (
182     SELECT month,
183         SUM (is_active_87) AS sum_active_87,
184         SUM (is_active_30) AS sum_active_30,
185         SUM (is_canceled_87) AS sum_canceled_87,
186         SUM (is_canceled_30) AS sum_canceled_30
187     FROM status
188     GROUP BY month)
189
190 SELECT *
191 FROM status_aggregate;
```

month	sum_active_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

We will now be aggregating the columns we created in the ‘status’ table by creating a new temporary table named ‘status_aggregate’ where we will find the sum of users who are active and canceled for both segments 87 and 30. To find this we will be using the SUM function for each column that was created in the ‘status’ table.


```

136 WITH months AS
137 (SELECT
138     '2017-01-01' AS first_day,
139     '2017-01-31' AS last_day
140 UNION
141     SELECT
142     '2017-02-01' AS first_day,
143     '2017-02-28' AS last_day
144 UNION
145     SELECT
146     '2017-03-01' AS first_day,
147     '2017-03-31' AS last_day),
148
149 cross_join AS (
150     SELECT *
151     FROM subscriptions
152     CROSS JOIN months),
153
154 status AS(
155     SELECT id,
156         first_day AS month,
157         CASE WHEN segment = 87
158             AND (subscription_start < first_day)
159             AND (subscription_end > first_day
160                 OR subscription_end IS NULL)
161             THEN 1
162             ELSE 0
163         END as is_active_87,
164         CASE WHEN segment = 30
165             AND(subscription_start < first_day)
166             AND (subscription_end > first_day
167                 OR subscription_end IS NULL)
168             THEN 1
169             ELSE 0
170         END as is_active_30,
171         CASE WHEN segment = 87
172             AND subscription_end BETWEEN first_day AND
173             last_day THEN 1
174             ELSE 0
175         END as is_canceled_87,
176         CASE WHEN segment = 30
177             AND subscription_end BETWEEN first_day AND
178             last_day THEN 1
179             ELSE 0
180         END as is_canceled_30
181     FROM cross_join),

```

```

238 status_aggregate AS (
239     SELECT month,
240         SUM (is_active_87) AS sum_active_87,
241         SUM (is_active_30) AS sum_active_30,
242         SUM (is_canceled_87) AS sum_canceled_87,
243         SUM (is_canceled_30) AS sum_canceled_30
244     FROM status
245     GROUP BY month)
246
247 SELECT month,
248     1.0 * sum_canceled_87 / sum_active_87 as
249     churn_rate_87,
250     1.0 * sum_canceled_30 / sum_active_30 as
251     churn_rate_30
252 FROM status_aggregate
253 ORDER BY month ASC;

```

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

Now we can calculate the churn rate for each segment for the 3 months that we are looking in to. To find churn we will need to divide the canceled users from the active users, and we will do this for each segment. Multiplying by '1.0' is so the values returned are float values instead of whole numbers.