

LEARN SQL FROM SCRATCH

Calculating Churn Rates

By

Jean Baptiste Aunial

TABLE OF CONTENTS

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?

1. GET FAMILIAR WITH THE COMPANY.

TAKE A LOOK AT THE FIRST 100 ROWS OF DATA IN THE SUBSCRIPTIONS TABLE.

HOW MANY DIFFERENT SEGMENTS DO YOU SEE?

THE SUBSCRIPTION OWNER BELONG TO TWO SEGMENT, 87 AND 31.

THE COMPANY HAS BEEN OPERATING FOR FOUR MONTHS, BUT WE HAVE INFORMATION TO CALCULATE THE CHURN RATE FOR THREE MONTHS.

```
1 SELECT *
2 FROM subscriptions
3 LIMIT 100;
```

Query Results			
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	Ø	30
14	2016-12-01	2017-03-07	30
15	2016-12-01	2017-02-22	30
16	2016-12-01	Ø	30
17	2016-12-01	Ø	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
25	2016-12-02	2017-02-24	87
26	2016-12-02	2017-01-18	87
27	2016-12-02	2017-01-11	87
28	2016-12-02	2017-03-30	30
29	2016-12-02	2017-02-11	30
30	2016-12-02	2017-01-20	30
31	2016-12-02	Ø	30

DETERMINE THE RANGE OF MONTHS OF DATA PROVIDED.

WHICH MONTHS WILL YOU BE ABLE TO CALCULATE CHURN FOR?

I WILL BE ABLE TO CALCULATE THE CHURN RATE FOR JANUARY, FEBRUARY AND MARCH.

Calculating Churn Rates

test.sqlite

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

Query Results

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

Database Schema

subscriptions	2000 rows
id	INTEGER

CALCULATE CHURN RATE FOR EACH SEGMENT

YOU'LL BE CALCULATING THE CHURN RATE FOR BOTH SEGMENTS (87 AND 30) OVER THE FIRST 3 MONTHS OF 2017 (YOU CAN'T CALCULATE IT FOR DECEMBER, SINCE THERE ARE NO SUBSCRIPTION END VALUES YET). TO GET STARTED, CREATE A TEMPORARY TABLE OF MONTHS.

test.sqlite

```
1
2
3 with months as
4 (SELECT
5  '2016-12-01' as first_day,
6  '2016-12-31' as last_day
7 UNION
8 SELECT
9  '2017-01-01' as first_day,
10 '2017-01-31' as last_day
11 UNION
12 SELECT
13  '2017-02-01' AS first_day,
14  '2017-02-28' AS last_day
15 UNION
16 SELECT
17  '2017-03-01' as first_day,
18  '2017-03-31' AS last_day)
19 SELECT *
20 FROM months;
```

Query Results

first_day	last_day
2016-12-01	2016-12-31
2017-01-01	2017-01-31
2017-02-01	2017-02-28
2017-03-01	2017-03-31

Database Schema

subscriptions	
id	INTEGER
subscription_start	TEXT
subscription_end	TEXT
segment	INTEGER

CREATE A TEMPORARY TABLE, CROSS_JOIN, FROM SUBSCRIPTIONS AND YOUR MONTHS. BE SURE TO SELECT EVERY COLUMN.

test.sqlite		Query Results					
		id	subscription_start	subscription_end	segment	first_day	last_day
1	<pre>with months as (SELECT '2016-12-01' as first_day, '2016-12-31' as last_day UNION 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) SELECT * FROM cross_join limit 100;</pre>	1	2016-12-01	2017-02-01	87	2016-12-01	2016-12-31
2		1	2016-12-01	2017-02-01	87	2017-01-01	2017-01-31
3		1	2016-12-01	2017-02-01	87	2017-02-01	2017-02-28
4		1	2016-12-01	2017-02-01	87	2017-03-01	2017-03-31
5		2	2016-12-01	2017-01-24	87	2016-12-01	2016-12-31
6		2	2016-12-01	2017-01-24	87	2017-01-01	2017-01-31
7		2	2016-12-01	2017-01-24	87	2017-02-01	2017-02-28
8		2	2016-12-01	2017-01-24	87	2017-03-01	2017-03-31
9		3	2016-12-01	2017-03-07	87	2016-12-01	2016-12-31
10		3	2016-12-01	2017-03-07	87	2017-01-01	2017-01-31
11		3	2016-12-01	2017-03-07	87	2017-02-01	2017-02-28
12		3	2016-12-01	2017-03-07	87	2017-03-01	2017-03-31
13		4	2016-12-01	2017-02-12	87	2016-12-01	2016-12-31
14		4	2016-12-01	2017-02-12	87	2017-01-01	2017-01-31
15		4	2016-12-01	2017-02-12	87	2017-02-01	2017-02-28
16		4	2016-12-01	2017-02-12	87	2017-03-01	2017-03-31
17		5	2016-12-01	2017-03-09	87	2016-12-01	2016-12-31
18		5	2016-12-01	2017-03-09	87	2017-01-01	2017-01-31
19		5	2016-12-01	2017-03-09	87	2017-02-01	2017-02-28
20		5	2016-12-01	2017-03-09	87	2017-03-01	2017-03-31
21		6	2016-12-01	2017-01-19	87	2016-12-01	2016-12-31
22		6	2016-12-01	2017-01-19	87	2017-01-01	2017-01-31
23		6	2016-12-01	2017-01-19	87	2017-02-01	2017-02-28
24		6	2016-12-01	2017-01-19	87	2017-03-01	2017-03-31
25		7	2016-12-01	2017-02-03	87	2016-12-01	2016-12-31
		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	

CREATE A TEMPORARY TABLE, STATUS, FROM THE CROSS_JOIN TABLE YOU CREATED.

THIS TABLE SHOULD 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

test.sqlite		Query Results			
		id	month	is_active_87	is_active_30
3	'2017-02-01' AS first_day,	1	2016-12-01	0	0
4	'2017-02-28' AS last_day	1	2017-01-01	1	0
5	UNION	1	2017-02-01	0	0
6	SELECT	1	2017-03-01	0	0
7	'2017-03-01' as first_day,	2	2016-12-01	0	0
8	'2017-03-31' AS last_day),	2	2017-01-01	1	0
9	cross_join as	2	2017-02-01	0	0
10	(SELECT *	2	2017-03-01	0	0
11	FROM subscriptions	3	2016-12-01	0	0
12	CROSS JOIN months),	3	2017-01-01	1	0
13	status as	3	2017-02-01	1	0
14	(SELECT id, first_day as month,	3	2017-03-01	1	0
15	CASE	4	2016-12-01	0	0
16	WHEN ((subscription_start < first_day)	4	2017-01-01	1	0
17	AND (subscription_end > first_day OR subscription_end IS NULL)) AND	4	2017-02-01	1	0
18	(segment = 87)	4	2017-03-01	0	0
19	THEN 1	5	2016-12-01	0	0
20	ELSE 0	5	2017-01-01	1	0
21	END AS is_active_87,	5	2017-02-01	1	0
22	CASE	5	2017-03-01	1	0
23	WHEN ((subscription_start < first_day)	6	2016-12-01	0	0
24	AND (subscription_end > first_day OR subscription_end IS NULL)) AND	6	2017-01-01	1	0
25	(segment = 30)	6	2017-02-01	0	0
26	THEN 1	6	2017-03-01	0	0
27	ELSE 0	6	2017-03-01	0	0
28	END AS is_active_30	7	2016-12-01	0	0
29	FROM cross_join)	7	2017-01-01	1	0
30	SELECT *	7	2017-02-01	1	0
	FROM status;	7	2017-03-01	0	0
		8	2016-12-01	0	0
		8	2017-01-01	1	0

ADD AN IS_CANCELED_87 AND AN IS_CANCELED_30 COLUMN TO THE STATUS TEMPORARY TABLE.

THIS SHOULD BE 1 IF THE SUBSCRIPTION IS CANCELED DURING THE MONTH AND 0 OTHERWISE

test.sqlite

↕ ↗

```
20 status as
21 (SELECT id, first_day as month,
22 CASE
23 WHEN ((subscription_start < first_day)
24 AND (subscription_end > first_day OR subscription_end IS NULL)) AND
25 (segment = 87)
26 THEN 1
27 ELSE 0
28 END AS is_active_87,
29 CASE
30 WHEN ((subscription_start < first_day)
31 AND (subscription_end > first_day OR subscription_end IS NULL)) AND
32 (segment = 30)
33 THEN 1
34 ELSE 0
35 END AS is_active_30,
36 CASE
37 WHEN((subscription_end BETWEEN first_day AND last_day) AND segment =
38 87) THEN 1
39 ELSE 0
40 END AS is_cancel_87,
41 CASE
42 WHEN((subscription_end BETWEEN first_day AND last_day) AND segment =
43 30) THEN 1
44 ELSE 0
45 END AS is_cancel_30
46 FROM cross_join)
47 SELECT *
48 FROM status;
```

Query Results

id	month	is_active_87	is_active_30	is_cancel_87	is_cancel_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

CREATE A STATUS AGGREGATE TEMPORARY TABLE THAT IS A SUM OF THE ACTIVE AND CANCELED SUBSCRIPTIONS FOR EACH SEGMENT, FOR EACH MONTH.

THE RESULTING COLUMNS SHOULD BE:

SUM_ACTIVE_87

SUM_ACTIVE_30

SUM_CANCELED_87

SUM_CANCELED_30

```
test.sqlite
--
25     THEN 1
26     ELSE 0
27     END AS is_active_87,
28     CASE
29     WHEN ((subscription_start < first_day)
30     AND (subscription_end > first_day OR subscription_end IS NULL)) AND
31     (segment = 30)
32     THEN 1
33     ELSE 0
34     END AS is_active_30,
35     CASE
36     WHEN((subscription_end BETWEEN first_day AND last_day) AND segment =
37     87) THEN 1
38     ELSE 0
39     END AS is_canceled_87,
40     CASE
41     WHEN((subscription_end BETWEEN first_day AND last_day) AND segment =
42     30) THEN 1
43     ELSE 0
44     END AS is_canceled_30
45     FROM cross_join),
46     status_aggregate as
47     (SELECT month, SUM(is_active_87) as Sum_active_87, SUM(is_active_30)
48     as Sum_active_30, SUM(is_canceled_87) as Sum_canceled_87,
49     SUM(is_canceled_30) as Sum_canceled_30
50     FROM status
51     GROUP by month)
52     SELECT *
```

Query Results				
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

Database Schema	
subscriptions	
2000 rows	
id	INTEGER
subscription_start	TEXT
subscription_end	TEXT
segment	INTEGER

CALCULATE THE CHURN RATES FOR THE TWO SEGMENTS OVER THE THREE MONTH PERIOD.

WHICH SEGMENT HAS A LOWER CHURN RATE?
THE 31 SEGMENT HAS A LOWER CHURN RATE

test.sqlite

```
25      THEN 1
26      ELSE 0
27      END AS is_active_87,
28      CASE
29      WHEN ((subscription_start < first_day)
30      AND (subscription_end > first_day OR subscription_end IS NULL)) AND
31      (segment = 30)
32      THEN 1
33      ELSE 0
34      END AS is_active_30,
35      CASE
36      WHEN((subscription_end BETWEEN first_day AND last_day) AND segment =
37      87) THEN 1
38      ELSE 0
39      END AS is_canceled_87,
40      CASE
41      WHEN((subscription_end BETWEEN first_day AND last_day) AND segment =
42      30) THEN 1
43      ELSE 0
44      END AS is_canceled_30
45      FROM cross_join),
46      status_aggregate AS
47      (SELECT month, SUM(is_active_87) AS Sum_active_87, SUM(is_active_30)
48      AS Sum_active_30, SUM(is_canceled_87) AS Sum_canceled_87,
49      SUM(is_canceled_30) AS Sum_canceled_30
50      FROM status
51      GROUP BY month)
52      SELECT month,
53      1.0 * sum_canceled_87 / sum_active_87 AS churn_rate_87, 1.0 *
54      sum_canceled_30 / sum_active_30 AS churn_rate_30
55      FROM status_aggregate;
```

Query Results

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

Database Schema

subscriptions		2000 rows
id	INTEGER	
subscription_start	TEXT	
subscription_end	TEXT	
segment	INTEGER	