



# Codeflix Churn Rates

Learn SQL from Scratch

# Table of Contents

1. Get familiar with Codeflix
2. Calculating Churn Rate
3. Overall Churn Rate
4. Churn Rate by Segment
5. Conclusions
6. Bonus

# 1. Get familiar with Codeflix

```
1 SELECT MIN(subscription_start) AS 'earliest
   subscription',
2 MAX(subscription_start) AS 'latest subscription'
3 FROM subscriptions;
4
5 --Data provided: December 2016, and January,
   February, March of 2017
6
7 SELECT
8 MIN(subscription_end) AS 'earliest cancel',
9 MAX(subscription_end) AS 'latest cancel'
10 FROM subscriptions;
11
12 --However December churn rates can't be provided,
   since the earliest cancel can only start in
   January.
```

*How many months has the company been operating?*

- 4 months (from December to March)

*Which months do you have enough information to calculate a churn rate?*

- From January to March

Query Results	
earliest subscription	latest subscription
2016-12-01	2017-03-30
earliest cancel	latest cancel
2017-01-01	2017-03-31

# 1. Get familiar with Codeflix

*What segments of users exist?*

- Two segments: 30 & 87
- To confirm, I wrote another bit of code

```
1 SELECT *
2 FROM subscriptions
3 LIMIT 100;
4
5 --looks like two segments, 87 and 30.
```

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
...			
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	Ø	30
100	2016-12-06	2017-03-11	30

```
6
7 --To confirm it, I ran the below query:
8
9 SELECT segment
10 FROM subscriptions
11 GROUP BY segment;
```

Alternative code:

```
SELECT DISTINCT segment
FROM subscriptions;
```

segment
30
87

But 87 will show above 30 in that case

## 2. Calculate Churn Rate

Create table of months, then a table of cross joins:

Query Results					
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

...

33	2016-12-02	Ø	30	2017-01-01	2017-01-31
33	2016-12-02	Ø	30	2017-02-01	2017-02-28
33	2016-12-02	Ø	30	2017-03-01	2017-03-31
34	2016-12-02	2017-02-06	30	2017-01-01	2017-01-31

```
1  WITH months AS (  
2      SELECT '2017-01-01' AS first_day,  
3             '2017-01-31' AS last_day  
4  
5      UNION  
6      SELECT '2017-02-01' AS first_day,  
7             '2017-02-28' AS last_day  
8  
9      UNION  
10     SELECT '2017-03-01' AS first_day,  
11            '2017-03-31' AS last_day  
12 )  
13 cross_join AS (  
14     SELECT *  
15     FROM subscriptions  
16     CROSS JOIN months  
17 )  
18  
19 SELECT *  
20 FROM cross_join  
21 LIMIT 100;
```

## 2. Calculate Churn Rate

Then, editing from the previous code from line 17, create table of status, starting with active status first

```
13 cross_join AS (  
14     SELECT *  
15     FROM subscriptions  
16     CROSS JOIN months  
17 ),  
18 status AS (  
19     SELECT id,  
20     first_day AS month,
```

```
21  
22     --is_active_87  
23     CASE  
24         WHEN (subscription_start < first_day)  
25         AND (  
26             subscription_end > first_day OR  
27             subscription_end IS NULL  
28         )  
29         AND (segment = 87) THEN 1  
30         ELSE 0  
31     END AS is_active_87,  
32  
33     --is_active_30  
34     CASE  
35         WHEN (subscription_start < first_day)  
36         AND (  
37             subscription_end > first_day OR  
38             subscription_end IS NULL  
39         )  
40         AND (segment = 30) THEN 1
```

```
41     ELSE 0  
42     END AS is_active_30,  
43  
44     FROM cross_join  
45 )  
46  
47 SELECT *  
48 FROM status  
49 LIMIT 100;
```

Query Results			
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
33	2017-02-01	0	1
33	2017-03-01	0	1
34	2017-01-01	0	1

## 2. Calculate Churn Rate

Adding in cancellations:

```
42     END AS is_active_30,  
43  
44     --is_canceled_87  
45     CASE  
46     WHEN (subscription_end BETWEEN first_day AND last_day)  
47     AND (segment = 87) THEN 1  
48     ELSE 0  
49     END AS is_canceled_87,  
50  
51     --is_canceled_30  
52     CASE  
53     WHEN (subscription_end BETWEEN first_day AND last_day)  
54     AND (segment = 30) THEN 1  
55     ELSE 0  
56     END AS is_canceled_30  
57  
58     FROM cross_join  
59 )
```

## 2. Calculate Churn Rate

Check results:

```
60
61 SELECT *
62 FROM status
63 LIMIT 100;
```

Query Results					
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
...					
33	2017-01-01	0	1	0	0
33	2017-02-01	0	1	0	0
33	2017-03-01	0	1	0	0
34	2017-01-01	0	1	0	0



## 2. Calculate Churn Rate

From slide 7, create 'status\_aggregate' table:

```
58 FROM cross_join
59 ),
60 status_aggregate AS (
61     SELECT month,
62         SUM(is_active_87) AS sum_active_87,
63         SUM(is_active_30) AS sum_active_30,
64         SUM(is_canceled_87) AS sum_canceled_87,
65         SUM(is_canceled_30) AS sum_canceled_30
66     FROM status
67     GROUP BY 1
68 )
69
70 SELECT *
71 FROM status_aggregate;
```

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

### 3. Overall Churn Rate

Continued from slide 9, a new table was created to add active and cancel rates together:

*What is the overall churn trend since the company started?*

```
68 ),
69 overall_churn_rate AS (
70     SELECT month,
71         sum_active_87 + sum_active_30 AS active,
72         sum_canceled_87 + sum_canceled_87 AS cancel
73     FROM status_aggregate
74     GROUP BY 1
75 )
76
77 SELECT month,
78     ROUND(CAST(cancel AS FLOAT)/active*100, 2) AS churn_rate
79 FROM overall_churn_rate;
```

Query Results	
month	churn_rate
2017-01-01	24.6
2017-02-01	30.2
2017-03-01	41.38

Churn rates appear to be increasing drastically

## 4. Churn Rate by Segment

Finally, get churn rates by segment (continued from slide 9):

```
68 )
69
70 SELECT month,
71 ROUND(CAST (sum_canceled_87 AS FLOAT) / sum_active_87*100, 2) AS churn_rate_87,
72 ROUND (CAST (sum_canceled_30 AS FLOAT) / sum_active_30*100, 2) AS churn_rate_30
73 FROM status_aggregate
74 GROUP BY 1;
```

Query Results		
month	churn_rate_87	churn_rate_30
2017-01-01	25.18	7.56
2017-02-01	32.03	7.34
2017-03-01	48.59	11.73

## 4. Churn Rate by Segment

- Segment 87 seems to have much higher churn rates than segment 30 to start with, and also an overall higher increase in churn rate

From slide 11:

Query Results		
month	churn_rate_87	churn_rate_30
2017-01-01	25.18	7.56
2017-02-01	32.03	7.34
2017-03-01	48.59	11.73

## 5. Conclusions

*Which segment of users should the company focus on expanding?*

- Segment 30 – the churn rate is much lower
- A higher proportion of users subscribe for longer
- Comparing a user in segment 87 to one in segment 30, more revenue can be gained from users in segment 30, assuming the same price is charged.

## 5. Conclusions

- Segment 30 is also larger, implying that it is already earning more revenue than segment 87 (assuming same price)

Note:

- Sometimes, when total users sampled are small, churn rates may be very big even if only one user unsubscribes
- In those cases, churn rate would not be useful to determine how long a user subscribes
- However, for segment 87, as the user base large enough each month, churn rates are useful for us

From slide 9:

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

## 6. Bonus

- Codeflix should also find out why churn rate is increasing in general. While segment 30 is growing, churn rate is also increasing; this means that the growth may not be sustainable.
- A survey can be carried out on segment 87 to find out why Codeflix is losing users there, if Codeflix wishes to recapture the segment
- Something appears to have happened in March, leading to a higher than usual proportion of users from both segments unsubscribing. Codeflix may wish to find out what caused it.
- To scale the code for more segments, I'd use loops to avoid hard-coding the segment numbers.