codecademy

Codeflix Churn Rates

Learn SQL from Scratch

Jonathan Khou | 7.17.18 Cohort

Table of Contents

1. Getting familiar Codeflix

2. Determining monthly churn rate

3. Churn rate comparison

- About Codeflix
  - Codeflix is a subscription based video streaming service which currently has two user segments the marketing team is evaluating
    - Query with “GROUP BY” to see available segments: 30 and 87
  - Querying the min / max subscription start and end, respectively, tells us the earliest start of their business, and most recent transaction
    - This is true if the max “subscription end” date is greater than the max “subscription start” (most recent transactions)
    - We also see that, in this case, both segments have equal date ranges
  - Here is what we know so far:
    - Current months of operation: 12/1/2016 – 3/31/2017
    - Foreseeable months of operation: 12/1/2016 – 5/1/2017
    - Segment 30 and 87 exist and have an equal subscription range

Earliest_Start	Max_Sub_End	Max_Sub_Start	Segment
12/1/2016	3/31/2017	3/30/2017	30
12/1/2016	3/31/2017	3/30/2017	87

```sql
--1. Get familiar with the data - get data limit to 100
 SELECT *
 FROM Subscriptions
 LIMIT 100;

--2. Identify company operating months and available segments of users - get max and min timeframes
 SELECT MIN(subscription_start) as 'Earliest_Start',
        MAX(subscription_end) AS 'Max_Sub_End',
        MAX(subscription_start) AS 'Max_Sub_Start',
        --'Max_Sub_Start' is useful if it is in the same month as the 'Max_Sub_End' because the business rule requires the user to hold the subscription for +31 days at minimum
        segment
 FROM subscriptions
 GROUP BY 4
 ORDER BY 1,2 DESC;
```

- The Codeflix marketing team wants to know how the churn rates compare between their user segments
  - But first we need to understand churn rate and calculating it based on available data. Churn rate is measured as the (total number of cancellations in a given time period) / (total number of active users in that time period).

$$\text{Churn Rate} = \frac{Total\ cancellations}{Total\ active\ users}$$

  - To calculate churn rate, we need to consider the following:
    1. Cancellations in a given month are determined by the subscription end date
    2. Active users within a given month are determined by both start and end date
    3. By using the earlier SELECT * FROM subscriptions query, we see 4 fields exist
    4. We need the data table to show which users are active and canceled each month
    5. We need to be able to SUM the active and SUM the canceled users each month

```sql
--1. Get familiar with the data - get data limit to 100
SELECT *
FROM Subscriptions
LIMIT 100;
```

id	subscription_start	subscription_end	segment
1	12/1/2016	2/1/2017	87
2	12/1/2016	1/24/2017	87

- To ensure we are comparing apples to apples for marketing, let us take a look at how many users are in each segment (this tells me if the data needs to be normalized)
  - Since they have the same volume of unique users (ids), the churn rate calculations will weigh normally, so we are good to go!

```
SELECT segment,
       COUNT(DISTINCT Id) AS 'Count'
FROM subscriptions
GROUP BY 1;
```

segment	Count
30	1000
87	1000

- Next, to help calculate the churn trend, we will create the temporary table called “months”. By using a UNION statement, we can build out a table with the first and last days of each month we desire. This will be useful when we are querying on first / last day parameters (remember – active users are based on the first day of each month, and canceled on both).

```
--3.Visualize the churn trend over the first 3 month period - Start by creating a months table to reference
WITH months 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
```

first_day	last_day
1/1/2017	1/31/2017
2/1/2017	2/28/2017
3/1/2017	3/31/2017

- We can now tie each id to each reference month to help us see which months the user is active in and where the user canceled
- By using a CROSS JOIN statement, we append each month in “months” to each row in “subscriptions” (each id is tied to each month)

```
38  --4.Cross join the subscriptions and months table
39  cross_join AS (
40  SELECT *
41  FROM subscriptions
42  CROSS JOIN months),
```

id	subscription_start	subscription_end	segment	first_day	last_day
1	12/1/2016	2/1/2017	87	1/1/2017	1/31/2017
1	12/1/2016	2/1/2017	87	2/1/2017	2/28/2017
1	12/1/2016	2/1/2017	87	3/1/2017	3/31/2017
2	12/1/2016	1/24/2017	87	1/1/2017	1/31/2017
2	12/1/2016	1/24/2017	87	2/1/2017	2/28/2017
2	12/1/2016	1/24/2017	87	3/1/2017	3/31/2017

- Now we need a way to identify which users are active and which have canceled each month within this new “cross-join” temporary table (this will play a role in the actual calculation of churn rates)

- Using the “cross-join” table, we will create the “status” table to help us identify which users are active or canceled each month
  - CASE WHEN statements can utilize multiple criteria that, when true, will result in the first “THEN” statement value, and if false, will result in the “ELSE” statement value
  - In this “case” ☺, we chose to make each value a 1 (for true) or 0 (for false)
  - Note: we added a condition for segment but a more efficient way will be shown later

```
--5.Create a new table that references the cross_join table to identify active users per desired segment
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,

--6. Add an is_canceled_87 and is_canceled_30 column to the temporary table to identify monthly cancellations
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
),
```

id	month	is_active_87	is_active_30	is_canceled_87	is_canceled_30
1	1/1/2017	1	0	0	0
1	2/1/2017	0	0	1	0
1	3/1/2017	0	0	0	0
2	1/1/2017	1	0	1	0
2	2/1/2017	0	0	0	0
2	3/1/2017	0	0	0	0
3	1/1/2017	1	0	0	0
3	2/1/2017	1	0	0	0
3	3/1/2017	1	0	1	0
4	1/1/2017	1	0	0	0
4	2/1/2017	1	0	1	0
4	3/1/2017	0	0	0	0

- Using the “status” table, we will create the “status-aggregate” table to help us SUM the active and canceled users each month for each segment (requires 2 lines of code for each segment)

```
49  --7. Create a temporary table that sums the active users and cancellations for
    each segment, respectively
50  status_aggregate AS (
51  SELECT month,
52         SUM(is_active_87) AS sum_active_87,
53         SUM(is_active_30) AS sum_active_30,
54         SUM(is_canceled_87) AS sum_canceled_87,
55         SUM(is_canceled_30) AS sum_canceled_30
56  FROM status
57  GROUP BY month
58  )
```

month	sum_active_87	sum_active_30	sum_canceled_87	sum_canceled_30
1/1/2017	278	291	70	22
2/1/2017	462	518	148	38
3/1/2017	531	716	258	84

- From this table, we can already see that the overall Codeflix cancellations are increasing each month for each segment, but we want to know how each segment is performing

$$Codeflix\ Churn\ Rate_{Jan} = \frac{22+70}{(278+291)} = 16\%$$

$$Codeflix\ Churn\ Rate_{Feb} = \frac{38+148}{(462+518)} = 19\%$$

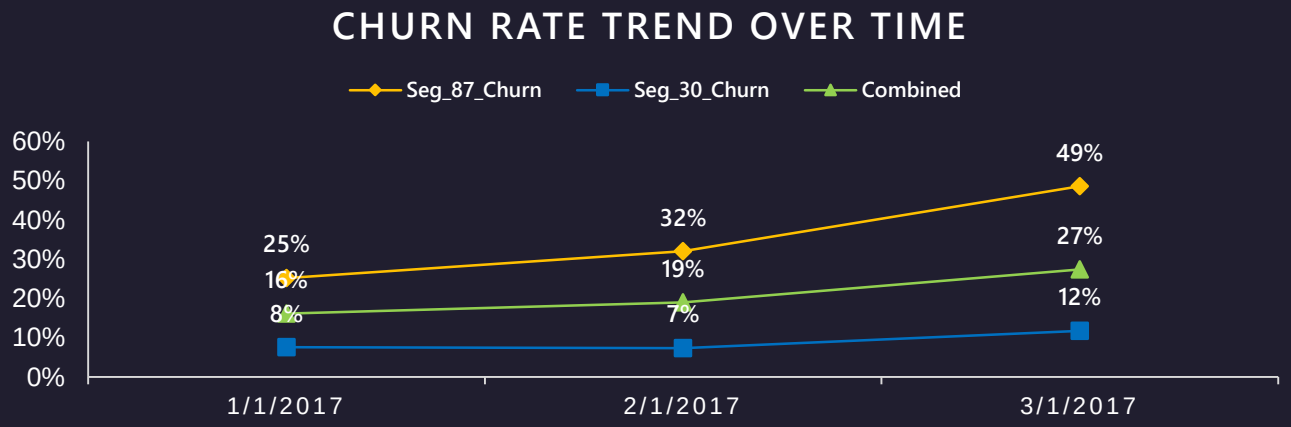
$$Codeflix\ Churn\ Rate_{Feb} = \frac{84+258}{(531+716)} = 27\%$$

A quick calculation shows that overall, churn rate is increasing

- We can then calculate monthly churn rate by dividing the total canceled users by the total active users for each month
  - We see that segment 30 has a much better churn rate trend than segment 87 (37% difference between the max)
  - Segment 87 has an upward trend, with its worst churn rate in April
  - Segment 30 has an upward trend (but dips in Feb), with its worst churn rate in April
  - Although segment 30 is performing better overall, April is Codeflix's worst month

```
59  --8. Calculate monthly churn rate and overall churn rate
60  --The below statement calculates churn rates monthly
61  SELECT month,
62        ((status_aggregate.sum_canceled_87*1.0) /
    ((status_aggregate.sum_active_87*1.0)) AS 'Seg_87_Churn',
63        ((status_aggregate.sum_canceled_30*1.0) /
    (status_aggregate.sum_active_30*1.0)) AS 'Seg_30_Churn'
64  FROM status_aggregate;
```

month	Seg_87_Churn	Seg_30_Churn
1/1/2017	25%	8%
2/1/2017	32%	7%
3/1/2017	49%	12%



- Now that we know the monthly churn rate, let's find out how the two segments compare overall
  - By using a SUM statement with the "status-aggregate" table, we effectively add all the cancellations for each month (combining them)
  - We can also use a SUM statement to add all the active users for each month – however, by doing this some users will be double counted (if active over more than 1 month) – which may be fine to see direction, but the magnitude of churn rate may be off (fix – use unique id counts)

```
66  --The below statement calculates churn rates overall (by segment) - uncomment to use
67  SELECT ((SUM(status_aggregate.sum_canceled_87)*1.0) /
    (SUM(status_aggregate.sum_active_87)*1.0)) AS 'Overall Churn 87',
68         ((SUM(status_aggregate.sum_canceled_30)*1.0) /
    (SUM(status_aggregate.sum_active_30)*1.0)) AS 'Overall Churn 30'
69  FROM status_aggregate;
```

Overall Churn 87	Overall Churn 30
37%	9%

  - Here is what we learned:
    - Overall churn rate for segment 87 = 37%
    - Overall churn rate for segment 30 = 9%
    - Monthly churn rate for segment 87 trends upwards to a max of 49%, while segment 30's max is 12%
    - April 2017 is a bad month for Codeflix – both segments are at their highest churn rate here

- SQL analyst's recommendation:
  - Codeflix should focus on expanding segment 30 – it shows the lowest churn rate trends and even showed a slight improvement in churn rate from January to February (whereas segment 87 only increased)
  - Codeflix should investigate February 2017's video stream offerings because April 2017 was its worst performing month
    - Due to the business rule of each subscription lasting 31 days at minimum, the company would not see the impact of bad video offerings until at least the next month
  - In addition to expanding segment 30, Codeflix should redirect some of its segment 87 resources towards segment 30 or reinvest in new potential user segments (consider dropping segment 87 altogether) otherwise risk losing customer base

Thank you!

Bonus slides

- Bonus opportunity – to calculate churn rate for an unknown or large number of segments (scalability), we can do the following (easier on lines of coding too)

1. Add a statement for December in the “months” temporary table

```
97  ------------------------------------------------------------------------------
98
99  --9. BONUS: Below contains modified code for a large number of segments
    (scalability)
100
101 --3.Visualize the churn trend over the first 3 month period - Start by creating a
    months table to reference. Note: Dec first/last day helps with overall churn
102 WITH months AS (
103 SELECT '2016-12-01' AS first_day,
104        '2016-12-31' AS last_day
105 UNION
106 SELECT '2017-01-01' AS first_day,
107        '2017-01-31' AS last_day
108 UNION
109 SELECT '2017-02-01' AS first_day,
110        '2017-02-28' AS last_day
111 UNION
112 SELECT '2017-03-01' AS first_day,
113        '2017-03-31' AS last_day
114 ),
```

first_day	last_day
12/1/2016	12/31/2016
1/1/2017	1/31/2017
2/1/2017	2/28/2017
3/1/2017	3/31/2017

2. In “status”, add another CASE WHEN statement to add a “new-user” field that marks each new subscription (new subscription = subscription start date BETWEEN first-day of month and last-day of month

id	segment	month	is_active	is_canceled	new_user
1	87	12/1/2016	0	0	1
1	87	1/1/2017	1	0	0

3. In the “status-aggregate” table, add a GROUP BY clause so that all available segments will aggregate

segment	month	new_user_count	active	cancellations
30	12/1/2016	291	0	0
30	1/1/2017	249	291	22
30	2/1/2017	238	518	38
30	3/1/2017	222	716	84
87	12/1/2016	279	0	0
87	1/1/2017	258	278	70
87	2/1/2017	222	462	148
87	3/1/2017	241	531	258

```
--9. Modified to include additional case statement (for new users per month) and grouping clause to simplify coding
status AS (
SELECT id, segment, first_day AS month,
CASE WHEN (subscription_start < first_day)
  AND (subscription_end > first_day OR subscription_end IS NULL)
  THEN 1
  ELSE 0
  END AS is_active,
CASE WHEN (subscription_end BETWEEN first_day AND last_day)
  THEN 1
  ELSE 0
END AS is_canceled,
CASE WHEN (subscription_start BETWEEN first_day and last_day)
  THEN 1
  ELSE 0
END AS new_user
FROM cross_join
),
status_aggregate AS (
SELECT segment,
       month,
       SUM(new_user) as new_user_count,
       SUM(is_active) AS active,
       SUM(is_canceled) AS cancellations
FROM status
GROUP BY 1, 2
)
```

4. Monthly churn rate calculation remains the same, but the data will be combined in one column instead of 2

segment	month	Churn Rate
30	12/1/2016	
30	1/1/2017	8%
30	2/1/2017	7%
30	3/1/2017	12%
87	12/1/2016	
87	1/1/2017	25%
87	2/1/2017	32%
87	3/1/2017	49%

```
147  --8. Calculate monthly churn rate and overall churn rate (modified)
148  --The below statement calculates churn rates monthly
149  SELECT segment,
150         month,
151         ((status_aggregate.cancellations*1.0) / (status_aggregate.active*1.0)) AS
     'Churn Rate'
152  FROM status_aggregate;
153
154  --The below statement calculates churn rates overall (by segment) - uncomment to
     use
155  SELECT segment,
156         ((SUM(status_aggregate.cancellations)*1.0) /
     (SUM(status_aggregate.new_user_count)*1.0)) AS 'Overall Churn'
157  FROM status_aggregate
158  GROUP BY 1;
```

5. The overall churn rate calculation then becomes total cancellations from status-aggregate divided by the total unique users from status-aggregate (this gives the total unique users for the period) – this avoids double counting users

segment	Overall Churn	Previous-calculation
30	14%	9%
87	48%	37%

Same direction, but if churn rate is calculated against unique users, then it is higher