codecademy

Calculating Churn Rates

Learn SQL from Scratch: Capstone Project

Latara Rose

February 19, 2019

Contents

1.a Getting Familiar with Codeflix - Segments

To get familiar with the company Codeflix, I first took a look at the first 100 rows of data in the subscriptions table. The top right picture shows the first 16 rows of data in the table.
From looking at this data, it seemed there were only 2 segments of users in the subscriptions table. To be sure, I also did a count of each ID number in each segment. This verified that the only segments of users in the table were 30 and 87. The bottom right picture shows the Query results.

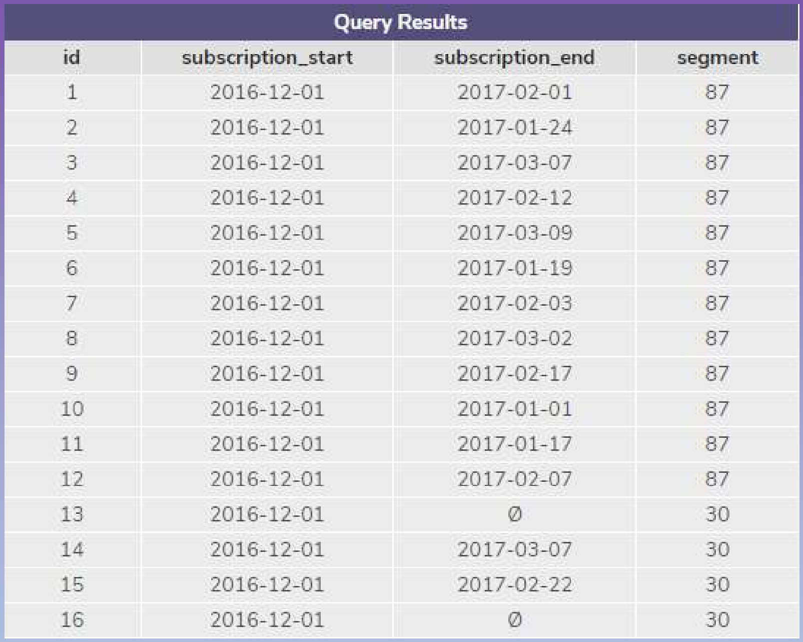
*Below are the queries used to find data

```
-- First 100 rows of data

SELECT *
 from subscriptions
 limit 100;
```

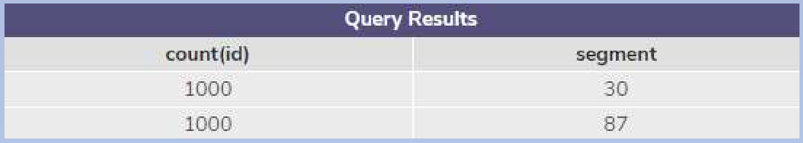
```
-- Verify segment count

SELECT count(id), segment
 from subscriptions
 group by segment;
```



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



Query Results

count(id)	segment
1000	30
1000	87

codecademy

1.b Getting Familiar with Codeflix – Months in Operation

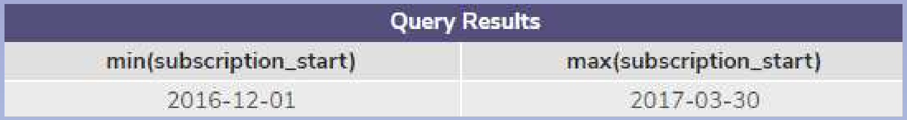
As I was becoming familiar with Codeflix, I also wanted to get the months in operation. I ran a query that gave me the earliest (min) subscription start date (subscription_start) and the latest (max) subscription start date (subscription_start).

This data showed that the company has had users subscribing from December 1, 2016 up until March 30, 2017.

**To the right is the Query used to find the data and the Query results.

```
-- Months in Operation

select min(subscription_start),
max(subscription_start)
 from subscriptions;
```



Query Results

min(subscription_start)	max(subscription_start)
2016-12-01	2017-03-30

codecademy

1.c Getting Familiar with Codeflix – Churn Rate Data

Churn Rate = Cancellations / Total Subscribers

In order to accurately calculate churn rate for the month, there must be active users subscribed prior to the first day in that month.

example: For the month of December 2016 we would need to have users subscribed in November 2016 (those would be the total subscribers in the month)

The earliest Subscription Start date for Codeflix is 12-01-16, so the earliest month we could calculate total subscribers would be January 2017.

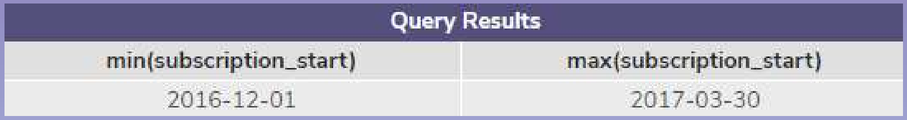
We also must have data for cancelations in each month we want to calculate churn rate. We have this data from January-March 2017.

So we are able to calculate Churn Rate for Codeflix for January 2017, February 2017, and March 2017.

**To the right are the Queries used to find the data and the Query results.

```
-- Subscription Start Data

select min(subscription_start),
max(subscription_start)
 from subscriptions;
```

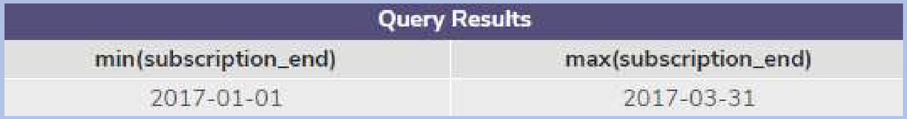


Query Results

min(subscription_start)	max(subscription_start)
2016-12-01	2017-03-30

```
-- Cancellation Data

select min(subscription_end),
max(subscription_end)
 from subscriptions;
```



Query Results

min(subscription_end)	max(subscription_end)
2017-01-01	2017-03-31

2. Overall Company Churn Rate Cont.

The Churn rate is the percentage of subscribers that have cancelled within a certain period. For Codeflix, I am using months as the periods.
Churn Rate = Cancellations / Total Subscribers

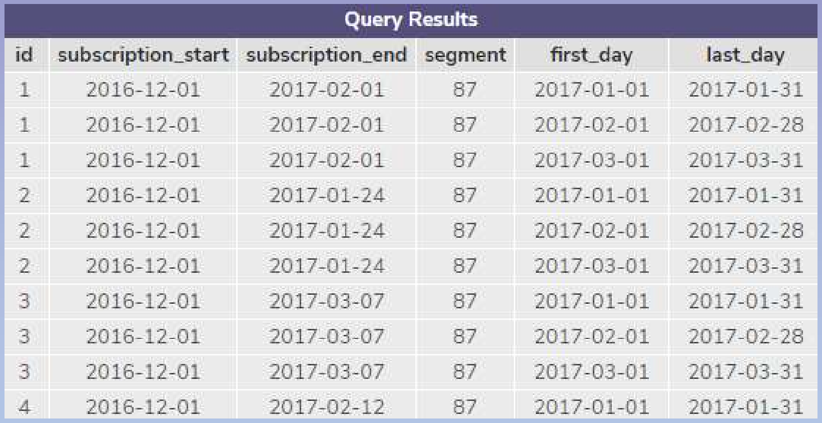
To get the overall company Churn rate trend, I have to get the Churn rate for each period (month) the company was operating. (Jan-Mar 2017)

- ❑ First, I created a temporary table called ‘months’ that breaks the data into the three months that we will be looking at (Jan, Feb, and Mar 2017)
- ❑ Second, I performed a cross join from subscriptions table and months table so we can use these columns to easily figure out which months users subscribed or cancelled their memberships.

**To the right are the Queries used to find the data and the Query results.

```
-- Temporary months table and cross join with
subscriptions table

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
),
cross_join AS
( select * from subscriptions
 cross join months
) select * from cross_join limit 10;
```



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
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

codecademy

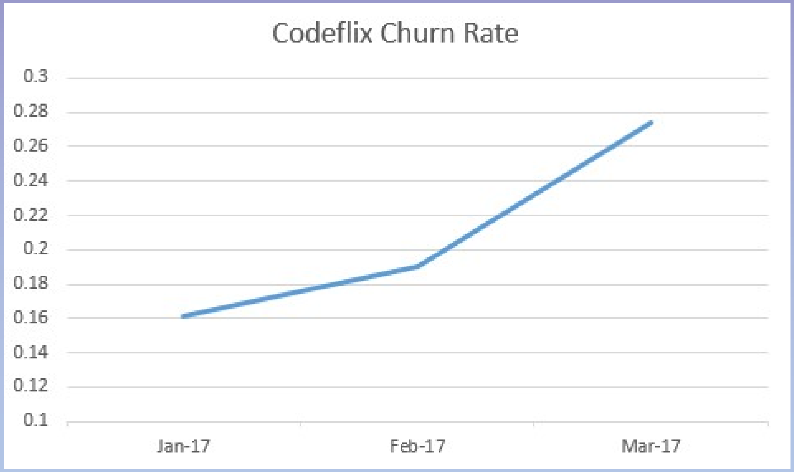
2. Overall Company Churn Rate Cont.

- ❑ Next, I created another temporary table, status, from the cross_join table. This step gives me columns of when a user ID was active for each month and in which month they cancelled, if any.
- ❑ The last temporary table I created was the status_aggregate table. This table summed up all of the active users for each month and all of the cancelled users for each month.
- ❑ Using these tables, I was able to calculate the church rate by taking the sum of the cancelled users by month divided by the sum of the total active users by month.

- ❑ From the data I created a line graph that shows the Codeflix's Churn rate is trending unfavorably from Jan-Mar 2017.
    - ❑ A higher churn rate means more subscribers are cancelling that month.

**On the next page are the Queries used to find the data and the Query results.

Month	Churn Rate
2017-01-01	0.161687170474517
2017-02-01	0.189795918367347
2017-03-01	0.274258219727346

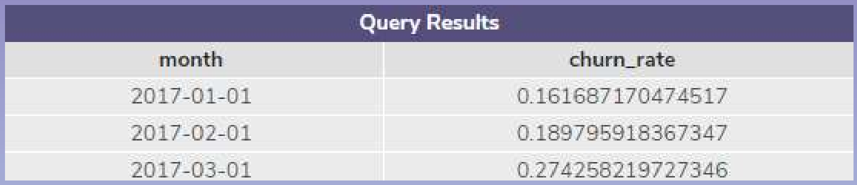


2. Overall Company Churn Rate Cont.

```
-- Full query to calculate company churn rate
for each month

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
),
cross_join AS
( select * from subscriptions
 cross join months
),
status AS
(select
 id,
 first_day AS month, segment,
 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
 from cross_join
), status_aggregate AS
(select month,
 sum(is_active) AS sum_active,
 sum(is_canceled) AS sum_canceled
 from status
 group by month
) select month,
1.0 * sum_canceled/sum_active AS churn_rate
from status_aggregate;
```



Query Results

month	churn_rate
2017-01-01	0.161687170474517
2017-02-01	0.189795918367347
2017-03-01	0.274258219727346

3. Segmented Churn Rate Cont.

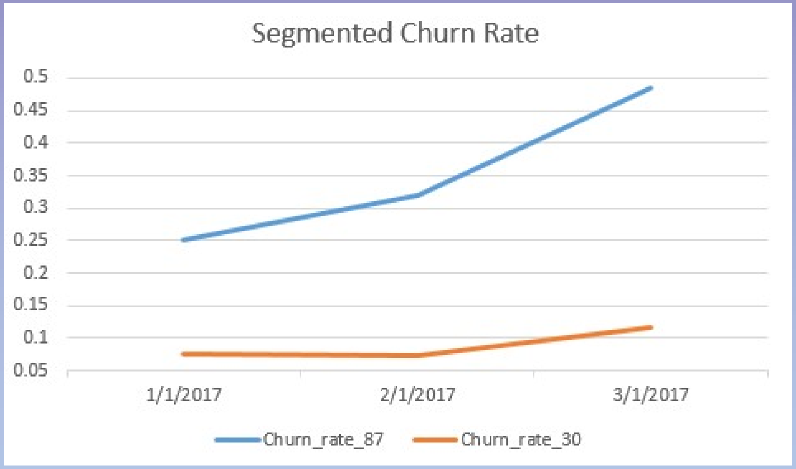
For the segmented churn rate I completed the beginning the same as calculating company churn rate up until creating the status table.
 I added more cases into the status table to give me columns for count of active users in each segment (30 and 87). I also created columns that game me the count of each user that cancelled in each month, in each segment.

I then used the same methodology with the status_aggregate table as before, but instead of only taking the same of active and cancelled users by month I got the sum active and cancelled users by segment for each month.

I then used that information to calculate the Churn rate for each segment by month.

**On the next page are the Queries used to find the data and the 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

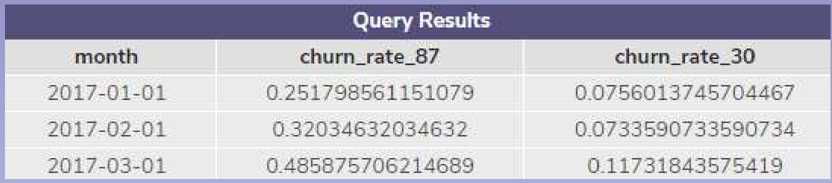


codecademy

3. Segmented Churn Rate Cont.

```
-- Full query to calculate company churn rate for each
month and each segment

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
),
cross_join AS
( select * from subscriptions
 cross join months
),
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,
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
), status_aggregate AS
(select month,
 sum(is_active_87) AS sum_active_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 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;
```


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

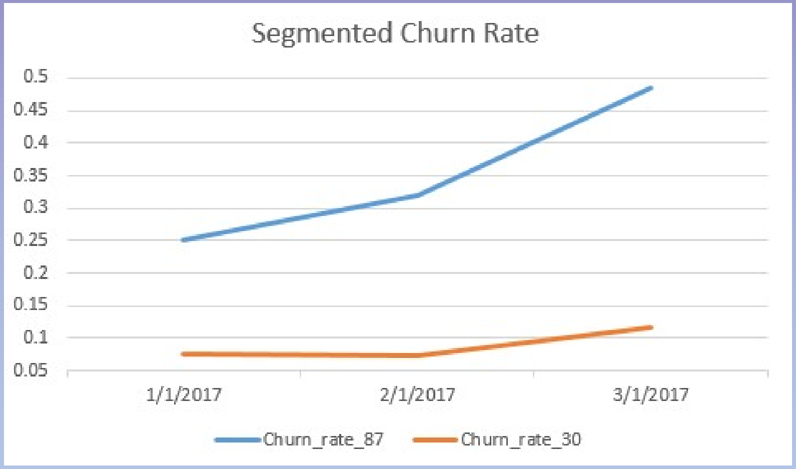
3. Segmented Churn Rate Cont.

I can clearly see from the data collected that the Churn rate for users in segment 30 is much lower than users in segment 87. This means that the users coming from segment 30 are staying subscribed for a longer period than users coming from Segment 87.

My advice to Codeflix is to focus on expanding the company with segment 30 since it is the much more successful segment.

The other option would be to dig deeper into why segment 30 is so much more successful than segment 87. Is it the wording in the adds? The place being marketed? The audience? Use this information to change segment 87 to try to get more long term subscribers to Codeflix.

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



codecademy

Thank you

Latara Rose
February 19, 2019