codecademy

Capstone Project
Calculating Churn Results

Learn SQL from Scratch

Chester Green 2018.10.29

Tasks

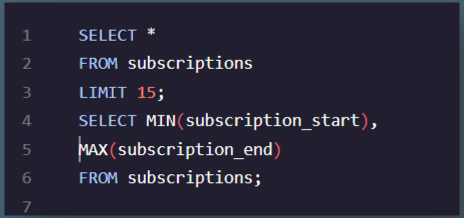
1. Get familiar with Codeflix
2. What is the overall churn rate by month?
3. Compare the churn rates between segments

1. Get familiar with Codeflix

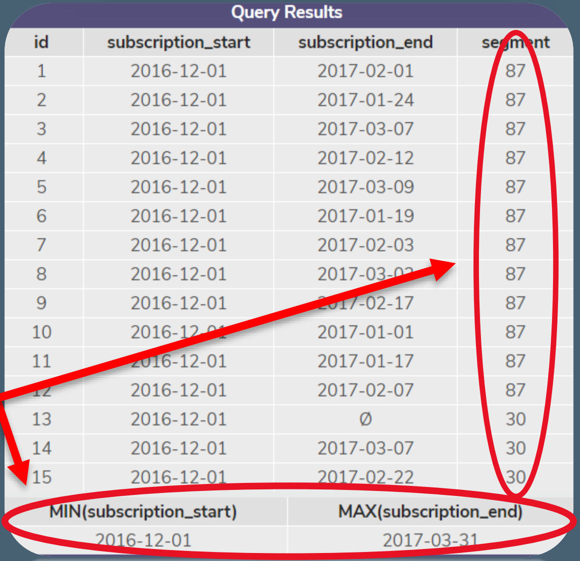
Codeflix has 3 months of data provided. The data begins 2016-12-01 and ends 2017-03-31.

Codeflix has enough information to provide the churn rate for 3 months of service. We can only see subscriptions for December through March. We need data before the 1st day of a month and for the entire month in order to calculate churn for any particular month.

Codeflix has 2 sources of subscriptions. Segment 87 and Segment 30. (15 rows were selected to display that both segments are presented)



```
SELECT *
FROM subscriptions
LIMIT 15;
SELECT MIN(subscription_start),
MAX(subscription_end)
FROM subscriptions;
```



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

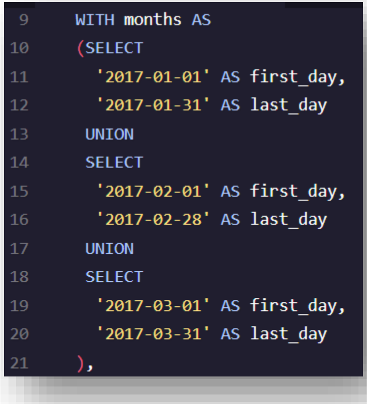
MIN(subscription_start)	MAX(subscription_end)
2016-12-01	2017-03-31

2. What is the overall churn rate by month?

To find the overall churn rate by month required many temporary tables and combining tables

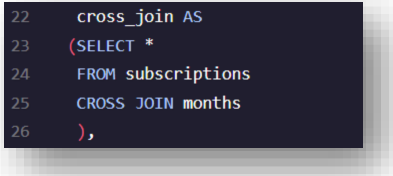
1. The first temporary table I created was called ‘months’ and defines the 1st and last day of each month
2. I then created a 2nd table ‘cross_join’ to combine the temporary table ‘months’ and the original table ‘subscriptions’
3. I then created a 3rd table ‘status’ which pulled out the most important data from the ‘cross_join’ temporary table and included several cases that specified which segments were active and canceled during each month

1



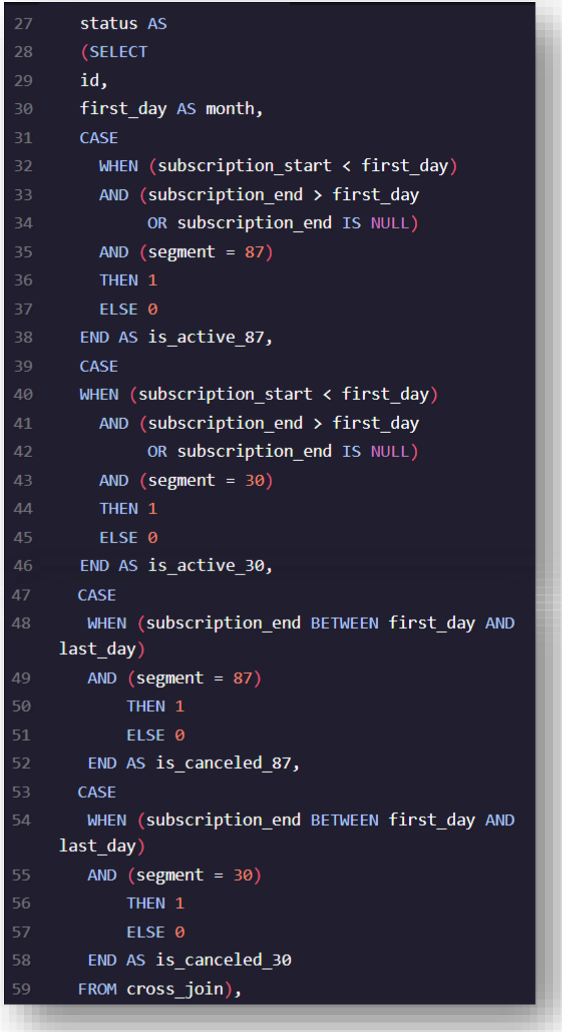
```sql
WITH months AS
(SELECT
  '2017-01-01' AS first_day,
  '2017-01-31' AS last_day
 UNION
 SELECT
  '2017-02-01' AS first_day,
  '2017-02-28' AS last_day
 UNION
 SELECT
  '2017-03-01' AS first_day,
  '2017-03-31' AS last_day
),
```

2



```sql
cross_join AS
(SELECT *
FROM subscriptions
CROSS JOIN months
),
```

3

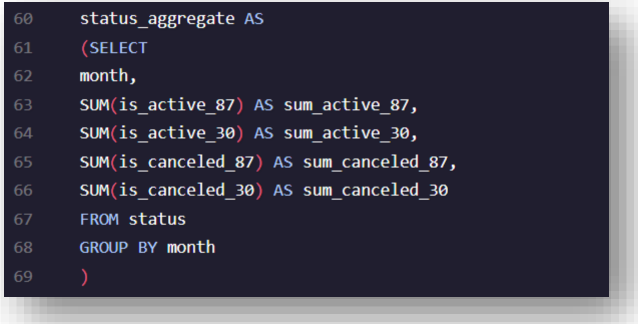


```sql
status AS
(SELECT
id,
first_day AS month,
CASE
  WHEN (subscription_start < first_day)
  AND (subscription_end > first_day
      OR subscription_end IS NULL)
  AND (segment = 87)
  THEN 1
  ELSE 0
END AS is_active_87,
CASE
WHEN (subscription_start < first_day)
  AND (subscription_end > first_day
      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),
```

2. What is the overall churn rate by month? (continued)

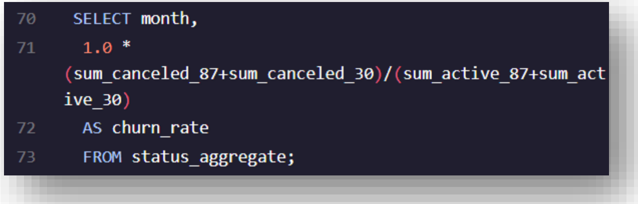
4. I then created a 4th temporary table 'status_aggregate' to add up the number of active and canceled segments for each month.
5. Finally, I selected the 3 months by row and executed an equation to calculate churn for each month.
6. The Results are displayed. The churn rate is trending UP every month, Codeflix might want to change their marketing strategy

4



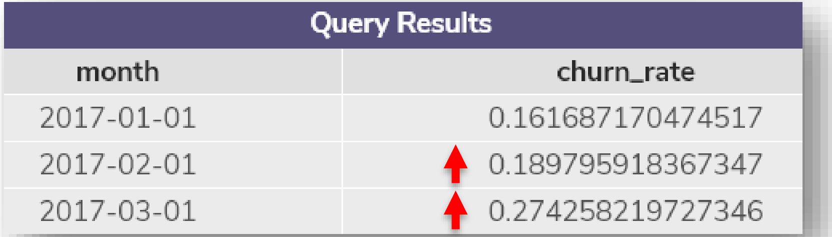
```
status_aggregate AS
(SELECT
month,
SUM(is_active_87) AS sum_active_87,
SUM(is_active_30) AS sum_active_30,
SUM(is_canceled_87) AS sum_canceled_87,
SUM(is_canceled_30) AS sum_canceled_30
FROM status
GROUP BY month
)
```

5



```
SELECT month,
 1.0 *
(sum_canceled_87+sum_canceled_30)/(sum_active_87+sum_active_30)
 AS churn_rate
 FROM status_aggregate;
```

6



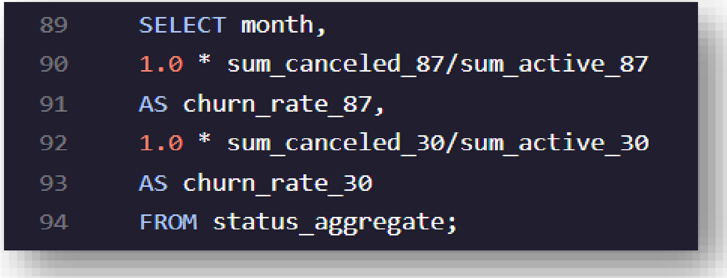
Query Results

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

3. Compare the churn rates between segments

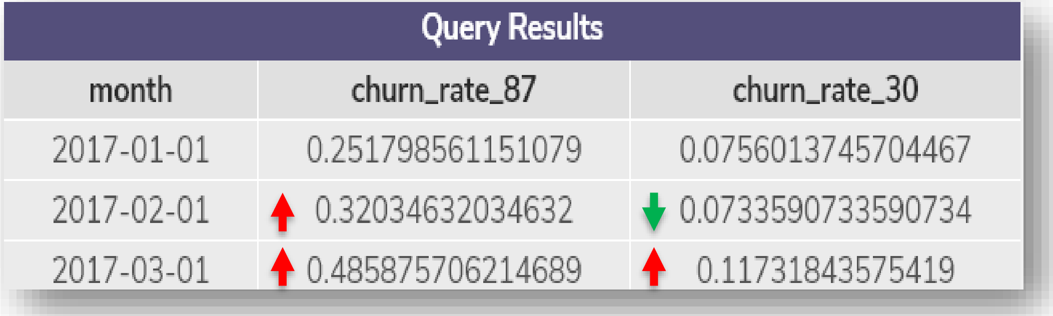
1. To compare the churn rates between segments I had to modify my selection to include the 3 different months and divide the findings into the 2 different segments.
2. The results are displayed. Segment 87 appears to go up each month. Segment 30 has a churn rate that is quite a bit lower. Segment 30 has a churn rate that has both gone up and down since the 1st month calculation. Segment 30 has been a better asset than Segment 87 so far. Codeflix should definitely focus more on Segment 30.

1



```
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;
```

2

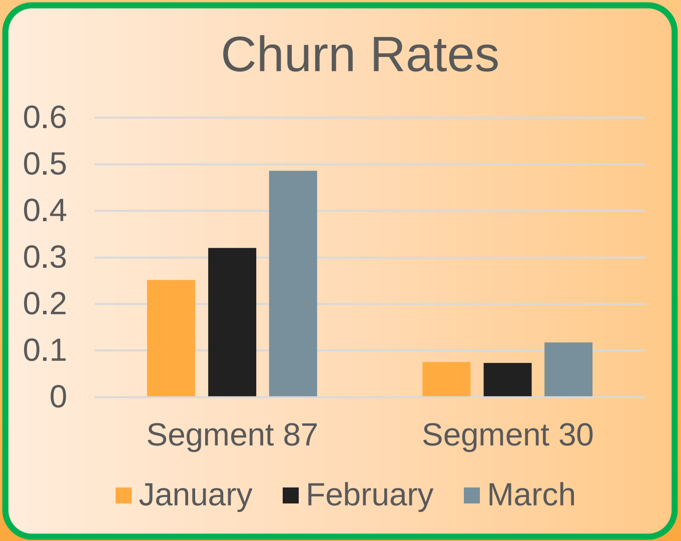


Query Results

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

Conclusion

- Codeflix is a new company with a lot of possibilities. The data they have shared has been very valuable in finding trends.
- Segment 87 and Segment 30 have been very valuable in bringing new business to Codeflix. The marketing department should look into expanding to other Segments as well. Of the 2 segments they are currently utilizing Segment 30 is definitely retaining more subscriptions than Segment 87
- If Codeflix ever wanted to add more Segments I revised my code to adjust for multiple segments appearing. The code is including in the text code.sql.


Churn Rates
Segment 87
Segment 30
January
February
March