codecademy

Churn Rate Capstone

Learn SQL from Scratch

By Suzie McVaney

Table of Contents

1. Get Familiar with Codeflix

1.1 The subscriptions table

The dataset provided to me contained one SQL table, **subscriptions**. Within the table, there were 4 columns:

1. id - the subscription id
2. subscription_start - the start date of the subscription
3. subscription_end - the end date of the subscription
4. segment - this identifies which segment the subscription owner belongs to.

My first step was simply to take a look at the data in the subscriptions table. I did this with a simple SELECT statement limiting the query to 100 rows of data (note: I only included the 1st two rows and the last two rows of data on this slide). From this query, I was able to determine that there are 2 segments of users within the subscriptions table for Codeflix.

```
SELECT *
FROM subscriptions
LIMIT 100;
```

id	subscription_start	subscription_end	segment
1	2016-12-01	2017-02-01	87
2	2016-12-01	2017-01-24	87
99	2016-12-06		30
100	2016-12-06	2017-03-11	30

1.2 Range of Months for Churn Rate

Codeflix has been operating for 4 months. *I will be able to calculate churn for January 2017, February 2017 and March 2017. I can not calculate churn for December 2016 because Codeflix requires 31 day subscription.*

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

MAX(subscription_start)
2017-03-30

MIN(subscription_start)
2016-12-01

2. Churn rates for segments

2.1 Create temporary tables

To produce the churns rates by month for each segment, I created several temporary tables using joins, case statements, and other SQL statements. The SQL for these tables was submitted in a txt file.

Months (for the 3 months we could calculate churn rates)
Cross_join (subscriptions and months)
Status (active and cancelled status for each subscription)
Status_aggregate (sum of active and cancelled subscriptions status)

A sample of the status _aggregate table is shown below

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

2.2 Calculate Churn Rates

Finally, I calculated the churn rate by month per segment. Clearly, segment 30 is the more successful segment for Codeflix.

```
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;
```

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. Bonus

3.1 Bonus – No hardcoding of segments

Bonus: How would I modify this code to support a large number of segments? This would be much more efficient code to allow Codeflix to grow more segments! I changed the code to include a GROUP By for segment when calculating the active and cancelled statuses. To the right is a segment of the SQL I changed.

segment	month	churn_rate
30	2017-01-01	0.0756013745704467
30	2017-02-01	0.0733590733590734
30	2017-03-01	0.11731843575419
87	2017-01-01	0.251798561151079
87	2017-02-01	0.32034632034632
87	2017-03-01	0.485875706214689

```
(SELECT
    segment,
    month,
    SUM(is_active) as sum_active,
    SUM(is_canceled) as sum_canceled
FROM status
GROUP BY month, segment)
SELECT
    segment,
    month,
  1.0 * sum_canceled/sum_active AS
churn_rate
FROM status_aggregate
ORDER BY segment;
```