codecademy

Capstone: Churn Rate

Learn SQL from Scratch

Luxi Hong

2018/08/07

Table of Contents

Getting Familiar with Codeflix

1.1 Getting familiar with the data table

By taking a look at the first 100 rows of subscriptions, here're some of the key findings:

- There are 4 columns in the table. They are: id, subscription_start, subscription_end, and segment
- There are only two segments: 87 and 30.
- Time range is from 2016-12-01 to 2017-03-31. In total there are 3 months which I can calculate churn rate for because the user cannot start and end the subscription within the same month.
- Below is the first 4 rows of subscriptions.

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

```
SELECT *
FROM subscriptions
LIMIT 100;

SELECT DISTINCT segment
FROM subscriptions;

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

Churn Rate per Month

2.1 Creating month table

In order to calculate churn rate, I first need a month table

- I used the first day and last day of each month to mark the beginning and end of each month, and then use union function to stag the rows together
- Because there are only 3 months available for churn rate, so I didn't include December 2016 as the company required a mandatory 31 days subscription after the subscription start
- months table would look like this:

first_day	last_day
2017-01-01	2017-01-31
2017-02-01	2017-02-28
2017-03-01	2017-03-31

```
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
```

2.2 Creating cross_join table

In order to calculate the active and cancelation numbers of each month in each segment,

- By cross joining months table with subscriptions table, each row in subscriptions table is multiplied with every row in months table
- cross_join table would look like this:

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

```
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)
SELECT *
   FROM subscriptions
   CROSS JOIN months
```

2.3 Creating status table

In the status table which is created based on cross_join table:

- Active subscribers for the month are IDs whose subscription starting date is before first day of the month, while subscription ending date is after the first day of the month or is null.
- Canceled subscribers for the month are IDs whose subscription ending date is between the first day and the last day of the month.
- In the status table, there are columns as shown below. For example, subscriber ID=1 is active in segment 87, and is active in Jan 2017, and canceled in Feb 2017.

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
2	2017-01-01	0	0	1	0

```
WITH
months AS
  (…),
cross_join AS
  (…),
  SELECT id, first_day AS month,
  CASE
    WHEN segment = 87 AND subscription_start < first_day
    AND (subscription_end > first_day
                OR subscription_end IS NULL) THEN 1
                ELSE 0
  END AS is_active_87,
  CASE
    WHEN segment = 30 AND subscription_start < first_day
    AND (subscription_end > first_day
                OR subscription_end IS NULL) 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;
```

2.4 Calculating churn rate by aggregating result in status table

- Churn rate = total number of canceled subscription / total number of active subscriptions
- By using SUM and GROUP BY function, I can calculate the total number of canceled subscriptions and total number of active subscriptions for each month and each segment.
- The result 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

```
WITH
months AS
  (…),
cross_join AS
  (…),
status 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;
```

2.4 Calculating churn rate

- Churn rate = total number of canceled subscription / total number of active subscriptions
- The result is shown below.

month	churn_rate_87	churn_rate_30
2017-01-01	0.251798561151079	0.0756013745704467
2017-02-01	0.320346320346632	0.0733590733590734
2017-03-01	0.485875706214689	0.11731843575419

```
WITH
months AS
  (…),
cross_join AS
  (…),
status AS
  (…),
status_aggregate AS
  (…)
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;
```

Comparing Churn Rate by Segment

3.1 Comparing Churn Rate by Segment

By looking at the result, we have the following conclusions:

- As indicated below, the churn rate for segment 87 is higher than segment 30 for all three months in 2017.
- March has the higher churn rate for both segments.
- The churn rate for segment 87 is climbing up steadily, while churn rate for segment 30 remains stable and low for the first two months of 2017 and jumps up significantly in March.

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

3.2 What if we have a large number of segments?

If we have a large number of segments, comparing churn rate for three months seems easier comparatively. So the main modification in the code would include:

- In the status table, hard code months instead of segment
- Calculating the churn rate by summing the total active subscriptions and total canceled substitutions in each month
- Group the result by segment instead of month.
- The result would look like this below:

segment	churn_rate_Jan	churn_rate_Feb	churn_rate_Mar
30	0.0756013745704467	0.0733590733590734	0.11731843575419
87	0.251798561151079	0.32034632034632	0.485875706214689

codecademy

Thank you!

Learn SQL from Scratch

Luxi Hong

2018/08/07