codecademy

Calculating Churn Rates

Analyze Data with SQL

Nilay Bayram

Data Analyst

06/01/2022

Introduction

An imaginary company called Codeflix, has been launched four months ago.

The marketing team wants to look into subscription churn rates and asks support of data analytics team. It’s early on in the business and people are excited to know how the company is doing.

The marketing department’s aim for this study is to compare churn rates between two segments of users.

Churn Rate is calculated as :

$$= \frac{Number\ of\ Cancellation\ During\ Given\ Period}{Number\ of\ Users\ at\ the\ Beginning\ of\ Given\ Period}$$

Codeflix requires a minimum subscription length of 31 days, so a user can never start and end their subscription in the same month.

The dataset provided contains one SQL table, called **subscriptions**.

The dataset is on Codecademy database and all the values are imaginary.

Analysing the Data

Within the table, there are 4 columns:

- **id** - the subscription id
- **subscription_start** - the start date of the subscription
- **subscription_end** - the end date of the subscription
- **segment** - this identifies which segment the subscription owner belongs to

There are two customer segments:

- **Segment 30**
- **Segment 87**

Churn rate can be calculated for 3 months:

January 2017, February 2017 and March 2017

December 2016 can't be taken into account because minimum subscription length of 31 days, there are no subscription_end values yet in December 2016.

```
--To see the column names

SELECT *
FROM subscriptions ;

--To see the segment names

SELECT *
FROM subscriptions
GROUP BY segment;

--To see which months I can calculate churn
rate:

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

MIN (subscription_start)	MAX(subscription_start)
2016-12-01	2017-03-30

Analysing the Data

To get started, create a temporary table of months:

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

To analyse activity or inactivity of a user for each period, subscription and months tables are joined with “Cross Join” command. New table name is **cross_join** :

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

```
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
FROM subscriptions)
SELECT *
FROM months;
_______________________________________________

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

Analysing the Data

A new temporary table called **status** is created from **cross_join** table.

This table will summarize the active users and canceled users during three periods. The code is given in the next page, the printed table is below:

id	subscription_start	subscription_end	first_day	last_day	segment	is_active_87	is_active_30	is_canceled_87	is_canceled_30
1	2016-12-01	2017-02-01	2017-01-01	2017-01-31	87	1	0	0	0
1	2016-12-01	2017-02-01	2017-02-01	2017-02-28	87	0	0	1	0
1	2016-12-01	2017-02-01	2017-03-01	2017-03-31	87	0	0	0	0
2	2016-12-01	2017-01-24	2017-01-01	2017-01-31	87	1	0	1	0
2	2016-12-01	2017-01-24	2017-02-01	2017-02-28	87	0	0	0	0
2	2016-12-01	2017-01-24	2017-03-01	2017-03-31	87	0	0	0	0
3	2016-12-01	2017-03-07	2017-01-01	2017-01-31	87	1	0	0	0
3	2016-12-01	2017-03-07	2017-02-01	2017-02-28	87	1	0	0	0
3	2016-12-01	2017-03-07	2017-03-01	2017-03-31	87	1	0	1	0

Analysing the Data

```
--status table is added to current code

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

```
status AS (
  SELECT id, subscription_start, subscription_end,
first_day,last_day, segment, 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 (segment=87) AND (subscription_end BETWEEN
first_day AND last_day) THEN 1
  ELSE 0
  END AS is_canceled_87,
  CASE
  WHEN (segment=30) AND (subscription_end BETWEEN
first_day AND last_day) THEN 1
  ELSE 0
  END AS is_canceled_30
  FROM cross_join)
  SELECT *
  FROM status;
```

Analysing the Data

A new temporary table called **status_aggregate** is created. That is a SUM of the active and cancelled subscriptions for each segment, for each month. Group By command is used to group the results by month.

The resulting 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

Finally, the churn rate has been calculated from status_aggregate table above. The SQL query at the next page for both steps.

month	Segment_87	Segment_30
2017-01-01	0.25	0.08
2017-02-01	0.32	0.07
2017-03-01	0.49	0.12

Analysing the Data

```
--status_aggregate table is added

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
FROM subscriptions),
cross_join AS (
  SELECT *
  FROM subscriptions
  CROSS JOIN months
), status AS (
  SELECT id, subscription_start,subscription_end,
first_day,last_day, segment, 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 (segment=87) AND (subscription_end BETWEEN
first_day AND last_day) THEN 1
  ELSE 0
  END AS is_canceled_87,
  CASE
  WHEN (segment=30) AND (subscription_end BETWEEN
first_day AND last_day) THEN 1
  ELSE 0
  END AS is_canceled_30
  FROM cross_join), status_aggregate AS(
    SELECT first_day AS 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 *
  FROM status_aggregate;
```

Analysing the Data

```
--Final query for churn rate calculation
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
FROM subscriptions),
cross_join AS (
  SELECT *
  FROM subscriptions
  CROSS JOIN months
), status AS (
  SELECT id, subscription_start,subscription_end,
first_day,last_day, segment, 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 (segment=87) AND (subscription_end BETWEEN
first_day AND last_day) THEN 1
  ELSE 0
  END AS is_canceled_87,
  CASE
  WHEN (segment=30) AND (subscription_end BETWEEN
first_day AND last_day) THEN 1
  ELSE 0
  END AS is_canceled_30
  FROM cross_join), status_aggregate AS(
    SELECT first_day AS 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,
ROUND(1.0*sum_canceled_87/sum_active_87,2) AS
Segment_87,
ROUND(1.0*sum_canceled_30/sum_active_30,2) AS
Segment_30
FROM  status_aggregate;
```

Result

To memorize, the formula to calculate churn rate is :

$$= \frac{Number\ of\ Cancellation\ During\ Given\ Period}{Number\ of\ Users\ at\ the\ Beginning\ of\ Given\ Period}$$

At the last query, churn rate is calculated for three months period as below.

month	Segment_87	Segment_30
2017-01-01	0.25	0.08
2017-02-01	0.32	0.07
2017-03-01	0.49	0.12

- It can be seen from the table that, **segment 30 has lower churn rate.**
- January 2017 has the lowest churn rate for Segment 87, although February 2017 has the lowest churn rate for Segment 30.
- March 2017 has the highest churn rate for both Segment 30 and segment 87. It seems like users have bad experiences recently and unsubscribed the service. The reason behind should be researched.