codecademy

Calculating Churn Rates

Analyze Data with SQL

Sergio Ventura Fiorentino Recabarren

11 september 2022

Calculating Churn Rates

Four months into launching Codeflix, management asks you to look into subscription churn rates. It's early on in the business and people are excited to know how the company is doing.

The marketing department is particularly interested in how the churn compares between two segments of users. They provide you with a dataset containing subscription data for users who were acquired through two distinct channels.

The dataset provided to you contains one SQL table, subscriptions. 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

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

1. Get familiar with the data

1.1 Take a look at the first 100 rows of data

How many different segments do you see?

In the 100 rows you can see two different user segments (87 and 30), below is a fragment of the table with the first two and the last two.

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

```
-- You can put your query here
 SELECT *
 FROM subscriptions
 LIMIT 100;
```

1.2 Determine the range of months of data provided.

Which months will you be able to calculate churn for?

The dropout rate could be calculated for 3 of 2017.

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

```
-- You can put your query here
 SELECT MIN (subscription_start),
 MAX(subscription_start)
 FROM subscriptions;
```

2. Calculate churn rate for each segment

2.1 Create a temporary table of months.

You'll be calculating the churn rate for both segments (87 and 30) over the first 3 months of 2017 (you can't calculate it for December, since there are no subscription_end values yet). To get started, create a temporary table (WITH) of months:

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

```
-- You can put your query here
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 months;
```

2.2 Create a temporary table, cross_join.

Create a temporary table, cross_join, from subscriptions and your months. Be sure to SELECT every column.

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

```
-- You can put your query here
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 3;
```

2.3 Create a temporary table, status, from the cross_join table you created.

Create a temporary table, status, from the cross_join table you created. This table should contain:

- id selected from cross_join
- month as an alias of first_day
- is_active_87 created using a CASE WHEN to find any users from segment 87 who existed prior to the beginning of the month. This is 1 if true and 0 otherwise.
- is_active_30 created using a CASE WHEN to find any users from segment 30 who existed prior to the beginning of the month. This is 1 if true and 0 otherwise.

id	month	is_active_87	is_active_30
1	2017-01-01	1	0
1	2017-02-01	0	0
1	2017-03-01	0	0
2	2017-01-01	1	0
2	2017-02-01	0	0

```
-- You can put your query here
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
FROM cross_join
) SELECT * FROM status LIMIT 5;
```

2.4 Add an is_canceled_87 and an is_canceled_30 column to the status temporary table.

Add an is_canceled_87 and an is_canceled_30 column to the status temporary table. This should be 1 if the subscription is canceled during the month and 0 otherwise.

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	1	0	1	0
2	2017-02-01	0	0	0	0
2	2017-03-01	0	0	0	0
3	2017-01-01	1	0	0	0
3	2017-02-01	1	0	0	0
3	2017-03-01	1	0	1	0
4	2017-01-01	1	0	0	0

```
-- You can put your query here
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
) SELECT * FROM status LIMIT 10;
```

2.5 Create a status aggregate temporary table.

Create a status_aggregate temporary table that is a SUM of the active and canceled subscriptions for each segment, for each month.
The resulting columns should be:
sum_active_87
sum_active_30
sum_canceled_87
sum_canceled_30

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

```
-- You can put your query here
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,
```

```
-- You can put your query here
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 * FROM status_aggregate;
```

2.6 Calculate the churn rates for the two segments over the three month period.

Which segment has a lower churn rate?

Answer: The user segment with the best churn rate is 30.

month	churn_rate_87	churn_rate_30
2017-01-01	0.251	0.075
2017-02-01	0.320	0.073
2017-03-01	0.485	0.117

```
-- You can put your query here
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,
```

```
-- You can put your query here
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;
```