codecademy

Calculating Churn Rates

Learn SQL from Scratch

By Shannon Spaulding 05.2018

Table of Contents

Codeflix Introduction

1.1 How many months has the company been operating?

According to the transactions from the subscriptions table, Codeflix was accepting subscriptions starting December 01, 2016, and logging subscription cancellations as late as March 31, 2017; **a total of 4 months of recorded activity**. There are also 1380 subscriptions without end dates, which would make it seem the company might still be in operation.

A.

first_start_date	last_end_date
2016-12-01	2017-03-31

B.

count_active_subscriptions
1380

```
-- A. Range of Start and End dates for subscriptions logged as transaction dates.

SELECT MIN(subscription_start) AS first_start_date,
                    MAX(subscription_end) AS last_end_date
FROM subscriptions;

-- B. Count of records that do not have an End Date (Active Subscriptions)

SELECT count(*) As count_active_subscriptions
FROM subscriptions
WHERE subscription_end IS NULL;
```

1.2 Which months do you have enough information to calculate a churn rate?

Churn rate is defined as the number of new subscriptions versus the number of cancellations.
Codeflix's policy allows for 3 months worth of data available for churn rate calculation, because the minimum subscription length, 31 days, would eliminate the month of December within the calculation.

A.

first_start_date	last_end_date
2016-12-01	2017-03-31

B.

count_active_subscriptions
1380

```
-- A. Range of Start and End dates for subscriptions logged as transaction dates.

SELECT MIN(subscription_start) AS first_start_date,
                    MAX(subscription_end) AS last_end_date
FROM subscriptions;

-- B. Count of records that do not have an End Date (Active Subscriptions)

SELECT count(*) As count_active_subscriptions
FROM subscriptions
WHERE subscription_end IS NULL;
```

1.3 What segments of users exist?

There are two unique segment values in the subscription table; ‘87’ and ‘30’.

A.

segment	count_user_segment
87	1000
30	1000

```
-- A. List of distince values for segment

SELECT DISTINCT segment, count(*) AS count_user_segment
FROM subscriptions
GROUP BY segment;
```

Monthly Churn Rate

2.1 What is the overall churn trend since the company started?

The overall churn trend for the company since the company started has been between 16% and 27 % with an upward trend.

SEE SLIDE 10 for churn rates by segment.

A.

month	churn_rate_ALL
2017-01-01	0.161687170474517
2017-02-01	0.189795918367347
2017-03-01	0.274258219727346

```
-- A. The overall monthly company churn rate
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 segment = '87'
                                                AND (first_day >
subscription_start)
                                                AND (first_day <
subscription_end
```

Churn Rate Comparison

3.1 Which segment of users should the company focus on expanding?

Codeflix should focus on expanding users assigned to segment 30, because this group is less likely to cancel within the first 3 months of the subscription. The average 3 months churn rate for segment 30 users is roughly 9%, while churn rate for segment 87 is 35%.

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

```
-- The monthly churn rate for segment ‘87’ and ‘30

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 segment = '87'
                                                            AND (first_day >
subscription_start)
                                                            AND (first_day <
```

Bonus

4.1 How would you modify this code to support a large number of segments?

By removing the hard coding of the segment within the fields and group by the segment, along with the month, the output of the churn rate will aways include every possible churn_rate by month by segment; allowing scalability.

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

```
-- Calculation of churn rate by month by segment; scalable to added
segments

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,
                    segment,
                    first_day AS month,
                    CASE
                                        WHEN  (first_day >
subscription_start)
```