codecademy

Codeflix Presentation

Calculating Churn Rates

A SQL Capstone by Dustin Spence

Table of Contents

1. Get familiar with Codeflix
2. What is the overall churn trend since Codeflix started?
3. Compare the churn rates between user segments

Get Familiar with Codeflix

1.1 How many months has the company been operating?

Codeflix has been in operation since December 1, 2016.

- From the present day this means that the company has been operating for nearly 24 months, or two years.
- This can be determined by running the following SQL Query:

  SELECT MIN(subscription_start) AS operation_start
  FROM subscriptions;
- The Query Result is listed below

operation_start
2016-12-01

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

Codeflix requires a minimum subscription of 31 days. As such we must discount the first month of operation when considering which months have enough information to calculate a churn rate. The information available gives us sufficient data to calculate a churn rate from January, 1 2017 – March 31, 2017

- This can be determined by running the following SQL Query:

```
SELECT MIN(subscription_start) AS First_Sub_Start,
 MAX(subscription_start) AS Last_Sub_Start,
 MIN(subscription_end) AS First_Sub_End,
 MAX(subscription_end) AS Last_Sub_End
FROM subscriptions;
```

- The Query Result is listed below

First_Sub_Start	Last_Sub_Start	First_Sub_End	Last_Sub_End
2016-12-01	2017-03-30	2017-01-01	2017-03-31

1.3 What segments of users exist?

There are two segments of users: 87 and 30

- This can be determined by running the following SQL Query:

```
SELECT DISTINCT segment
FROM subscriptions;
```

- The Query Result is listed below

segment
87
30

What is the overall churn trend since Codeflix started?

2.1.1 What is the overall churn trend since Codeflix started?

There has been an increasing overall churn trend since the company started. The churn rate has increased every month for which we have data. The churn rate was 16% in January, 19% in February and 27% in March.

- This can be determined by running the SQL Query on next slide (2.1.2)
- The Query Result is listed below

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

2.1.2 What is the overall churn trend since Codeflix started?

```
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)
   THEN 1 ELSE 0
  END AS is_active,
  CASE
   WHEN subscription_end BETWEEN first_day AND last_day
   THEN 1 ELSE 0
  END AS is_canceled
 FROM cross_join),
status_aggregate AS (
 SELECT month,
  SUM(is_active) AS sum_active,
  SUM(is_canceled) AS sum_canceled
 FROM status
 GROUP BY month)
SELECT month,
 1.0*sum_canceled/sum_active AS overall_churn
FROM status_aggregate;
```

Compare the churn rates between user segments

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

The company should expand user segment 30 as this segment has a much smaller overall churn rate of 9%, compared with segment 87's 37% churn rate.

- Overall church rate can be determined by running the SQL Query on next slide (3.1.2)
- The Query Result is listed below

overall_churn_87	overall_churn_30
0.374508261211644	0.0944262295081967

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

```
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 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 = '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_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 = '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 SUM(is_active_87) AS sum_active_87,
  SUM(is_canceled_87) AS sum_canceled_87,
  SUM(is_active_30) AS sum_active_30,
  SUM(is_canceled_30) AS sum_canceled_30
 FROM status)
SELECT 1.0*sum_canceled_87/sum_active_87 AS
overall_churn_87,
 1.0*sum_canceled_30/sum_active_30 AS overall_churn_30
FROM status_aggregate;
```

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

Additionally segment 30's churn fluctuation from month to month has stayed relatively stable – seeing no significant change between January and February, and a 4% church increase between February and March. This stands in contrast to segment 87 – with a churn increase of 7% from January to February, and 16% churn increase from February to March.

- The monthly church rate can be determined by running the SQL Query on next slide (3.1.4)
- The Query Result is listed below

month	overall_churn_87	overall_churn_30
2017-01-01	0.251798561151079	0.0756013745704467
2017-02-01	0.320346320346632	0.0733590733590734
2017-03-01	0.485875706214689	0.11731843575419

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

```
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 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 = '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_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 = '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 month,
  SUM(is_active_87) AS sum_active_87,
  SUM(is_canceled_87) AS sum_canceled_87,
  SUM(is_active_30) AS sum_active_30,
  SUM(is_canceled_30) AS sum_canceled_30
 FROM status
 GROUP BY month)
SELECT month,
 1.0*sum_canceled_87/sum_active_87 AS overall_churn_87,
 1.0*sum_canceled_30/sum_active_30 AS overall_churn_30
FROM status_aggregate;
```