Analysis of Codeflix's User Churn

Analyzing Churn Rates with SQL

by Marie Gomez

Data provided by codecademy

Table of Contents

1. Overview

Project Overview

The purpose of this project is to analyze user subscription churn rates at Codeflix. Throughout the process, I also illustrate the process of transforming the original table into a new one suited for calculating churn rates. Last, I conclude with my findings and recommendations.

Company Background:
Codeflix is a (fictional) streaming video startup that launched four months ago. The marketing department is interested in how the churn compares between two segments of users that were acquired through two different channels.

Codeflix requires that users be subscribe for a length of 31 days.

Dataset provided in this project:

1 SQL table called subscriptions:

- Contains 2000 records
- 4 columns:
  - id - subscription id
  - subscription_start
  - subscription_end
  - segment

2. Inspect the data

Inspect the subscription table

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
5	2016-12-01	2017-03-09	87
6	2016-12-01	2017-01-19	87
7	2016-12-01	2017-02-03	87
8	2016-12-01	2017-03-02	87
9	2016-12-01	2017-02-17	87
10	2016-12-01	2017-01-01	87

segment	COUNT(segment)	
30	1000	
87	1000	
segment	**COUNT(segment)**	**subscription_start**
30	5	2016-12-01
30	7	2016-12-02
30	5	2016-12-03
30	12	2016-12-04
30	9	2016-12-05
30	11	2016-12-06
30	11	2016-12-07
30	8	2016-12-08
30	16	2016-12-09
30	11	2016-12-10

There are two segments of users that indicate the two channels that users were acquired.

Both segment 30 & 87 have 1,000 users each.

Grouping by segment, subscription_start showed that both segment 30 and 87's subscription_start dates were from 2016-12-01 through 2017-03-30.

Although fours months is collected, for churn calculation December 2016 cannot be used since there are no subscription_values for that month.

```sql
SELECT *
FROM subscriptions
LIMIT 10;

SELECT segment, COUNT(segment)
FROM subscriptions
GROUP BY segment;

SELECT segment, COUNT(segment), subscription_start
FROM subscriptions
GROUP BY segment, subscription_start
LIMIT 10;
```

3. Churn Trend

3.1 Churn Trend - Format the table

To calculate churn, first the subscription table needs to be formatted into a table fitted for churn rates.

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

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

The first temp table created is ‘months’

```
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;
```


Next step is to create another temp table called ‘cross_join’ that cross joins ‘months’ to ‘subscriptions’

```
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
  )
```

3.2 Churn Trend - Format the table

Continue to add another temporary table called ‘status’ to the ‘cross_join’ table created.

```
/*partial SQL code*/
WITH months AS (
 .....
 ),
 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 = 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)
```

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

At this point, the table has has been formatted with binary values 0 and 1s.

With binary values in place, the dataset is ready to be aggregated to calculate churn rates per segment group.

3.3 Churn Trend - Calculate Churn Rates

sum_active_87	sum_canceled_87	sum_active_30	sum_canceled_30	Churn_%_87	Churn_%_30
1271	476	1525	144	37.0	9.0

Continue to add another temporary table called ‘status_aggregate’ to the ‘status’ table previously created.
The ‘status _aggregate’ sums up the active and cancelled subscriptions and includes churn rate percentages.

Findings:
The overall churn trend since the company started shows segment 87 with a higher churn rate at 37% while segment 30 had only 9% churn rate.

```
/*partial SQL code*/
WITH months AS (
  ...
  ),
  status AS (
    SELECT ...
  ),
  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,
      ROUND(100*CAST(SUM(is_canceled_87) AS REAL)/SUM(is_active_87)) AS 'Churn_%_87',
      ROUND(100*CAST(SUM(is_canceled_30) AS REAL)/SUM(is_active_30)) AS 'Churn_%_30'
    FROM status
  )
  SELECT *
  FROM status_aggregate;
```

4. Compare Churn Trend

4. Compare Churn Trend

Year-Month	sum_active_87	sum_canceled_87	sum_active_30	sum_canceled_30	Churn_%_87	Churn_%_30
2017-01	278	70	291	22	25.0	8.0
2017-02	462	148	518	38	32.0	7.0
2017-03	531	258	716	84	49.0	12.0

Findings:
The churn trend per segment per month since the company started shows segment 87 with churn rates that increased over the three months.

While segment 30 had a significantly smaller churn rate per month than segment 87 did.

```sql
/*partial SQL code*/
status_aggregate AS (
  SELECT
    STRFTIME('%Y-%m', month) AS 'Year-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,
    ROUND(100*CAST(SUM(is_canceled_87) AS REAL)/SUM(is_active_87)) AS 'Churn_%_87',
    ROUND(100*CAST(SUM(is_canceled_30) AS REAL)/SUM(is_active_30)) AS 'Churn_%_30'
  FROM status
  GROUP BY month
)
SELECT *
FROM status_aggregate;
```

5. Conclusion

Conclusion

Which segment of users should the company focus on expanding?

Ideally, Codeflix should work to expand both segments.

In this case, Segment 30 with a smaller churn rate should be focused on more because these users show less turnover, more customer loyalty, a higher lifetime value, and thus keep a stream of healthy revenue going for the company.

It is likely that the marketing efforts were done well for segment 30 and it is recommended that similar efforts be applied to segment 87 before its churn rate worsens in the next months that follow.

It is also recommended that marketing efforts done in segment 87 be applied to new future users.