codecademy

CodeFlix Churn Rate Analysis

prepared by Elena Filina

Table of Contents

1.1 CodeFlix introduction and data availability

CodeFlix is a streaming video startup established in December 2016.

- It has been operating for 4 months (Dec-2016 – March-2017).
- The company has a Minimum Subscription length of 31 days, i.e. a user can't start and end their subscription in the same month which is supported by the data provided by the marketing department, see Table 1.

The following prerequisites have been used for churn rate calculation:

- Three months: January – March 2017 (since no data of December 2016 cancellations was available);
- Two user segments: 30 and 87 (as requested by the Marketing department)

Table 1

Earliest_start	Latest_start	Earliest_cancellation	Latest_cancellation
2016-12-01	2017-03-30	2017-01-01	2017-03-31

```
SQL_Code:

select min(subscription_start) as
'earliest_start',max(subscription_start)
as 'latest_start', min(subscription_end)
as 'earliest_cancellation',
max(subscription_end) as
'latest_cancellation'
from subscriptions;
```

2.1 Churn Rate Calculation – by month

Data analysis for three months provided the following results:

- Calculation of overall churn rates by month demonstrated growth over the reviewed period;
- Gaining number of new subscriptions has a slower pace than subscription cancelations;

Note: Although 3 months-period is insufficient for substantial conclusions and/or actions, it demonstrates that further analysis should be performed in order to attract and retain users.

Month	Sum_active	Sum_canceled	Churn_rate
2017-01-01	569	92	0.16
2017-02-01	980	186	0.19
2017-03-01	1247	342	0.27

```
SQL-Code:
  with months as
    (select ------),
    cross_join as
    (select ------),
    status as
    (select ------
     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, sum_active, sum_canceled,
round(1.0* sum_canceled / sum_active,2) as
'churn_rate'
     from status_aggregate
     group by month;
```

3.1 Churn Rate by Segment – Calculation & Conclusions

Review of the churn rates for the 3 months-period revealed the following:

- Churn rates for segment 87 was significantly higher than for segment 30;
- There was slight decrease in subscription cancelations in Feb

for segment 30, which picked up the pace back in March. Further monitoring and analysis of the data would identify the root cause and it's sustainability;

Month	Churn_Rate_87	Churn_Rate_30
2017-01-01	0.25	0.08
2017-02-01	0.32	0.07
2017-03-01	0.48	0.12

Although the data demonstrated the excellence of segment 30 vs 87 and it's obvious to focus on segment 30, I would recommend the following:

- To perform detailed analysis of the data to identify which of the activities (campaigns, promotions, etc.) that have been implemented in segment 30 had a positive effect on attracting and retaining users;
- Evaluate if those activities could be applied to segment 87 as it has a larger portion of the user subscription pool and represent a higher risk to the financial performance;

3.2 SQL Code: Churn Rate by Segment

```
with months as
    (select -------),
    cross_join as
    (select -------),
    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_end between first_day and last_day) and (segment = 87)
     then 1 else 0
     end as 'is_canceled_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 = 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_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, round(1.0* sum_canceled_87 / sum_active_87,2) as 'churn_rate_87', round(1.0* sum_canceled_30 /
sum_active_30,2) as 'churn_rate_30'
     from status_aggregate
     group by month;
```

3.3 Verification Step

In order to verify accuracy and completeness of the SQL code used for Churn Rate Calculation by Segment, the following SQL query was run. The results are presented in the table below and agreed to the main SQL query with no exceptions.

Month	Segment	Sum_active	Sum_canceled	Churn_rate
2017-01-01	30	291	22	
2017-02-01	30	518	38	
2017-03-01	30	718	84	
2017-01-01	87	279	70	
2017-02-01	87	467	148	
2017-03-01	87	541	258	

```
with months as
    (select ------),
    cross_join as
    (select ------),
    status as
    (select id, segment, 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, segment, sum(is_active) as 'sum_active',
sum(is_canceled) as 'sum_canceled'
     from status
     group by month,segment)
     select month, segment, sum_active, sum_canceled
     from status_aggregate
     group by segment, month;
```