codecademy

Codeflix Subscription Churn Analysis

Learn SQL from Scratch Capstone

By: Nick Humann

Agenda

1. Codeflix Data Overview
2. Monthly Churn Rates
3. Comparison of Churn Rates by Segment

1. Codeflix Data Overview

1.1 Subscription Data Overview

Representative Output of *subscriptions* Database

Id (Integer)	subscription_start (Text)	subscription_end (Text)	segment (Integer)
1	2016-12-01	2017-02-01	87
15	2016-12-01	2017-02-22	30
30	2016-12-02	2017-01-20	30
45	2016-12-04	2017-02-02	87

Two segments represented – 87 and 30

Database Structure

Codeflix's subscription data is structured in four columns:

1) id (Integer) – the subscription id
2) subscription_start (Text) – the start date of the subscription
3) subscription_end (Text) – the end date of the subscription
4) segment (Integer) – this identifies which segment the subscription owner belongs to

Query Used

```
SELECT *
FROM subscriptions
LIMIT 100;
```

1.2 Subscription Data Overview

Churn Rate can be calculated for first three months of 2017

We can calculate churn rates for **January, February, and March of 2017** because we have subscription_start and subscription_end data for those months.

We do not have subscription_end data for December 2016, therefore we cannot calculate a churn rate for that month

Query Used

```
SELECT MIN(subscription_start),
       MAX(subscription_start),
       MIN(subscription_end),
       MAX(subscription_end)
FROM subscriptions;
```

MIN(subscription_start)	MAX(subscription_start)	MIN(subscription_end)	MAX(subscription_end)
2016-12-01	2017-03-30	2017-01-01	2017-03-31

2. Codeflix
Monthly Churn Rates

2.1 Codeflix is Experiencing Increasing Churn

- Over the three months of available data, Codeflix's churn rate has increased from 16.2% to 27.4% (January 2017 to March 2017)

month	combined_churn
January	16.2%
Februray	19.0%
March	27.4%

Query Used

```
SELECT month,
       1.0 * (sum_canceled_87 + sum_canceled_30) / (sum_active_87+sum_active_30) AS
          combined,
       1.0 * sum_canceled_30 / sum_active_30 AS churn_rate_30
FROM status_aggregate;
```

See Appendix for supporting code

3. Codeflix
Churn Rate by Segment

3.1 However, Codeflix Churn is Significantly Lower in Customer Segment 30

- Segment 30's monthly churn rate, while increasing, is significantly lower than Segment 87's
- **Results indicate that management should focus on replicated Segment 30's successes with all customers**

month	churn_rate_87	churn_rate_30
January	25.2%	7.6%
February	32.0%	7.3%
March	48.6%	11.7%

Query Used

```
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;
```

See Appendix for supporting code

3. APPENDIX: SUPPORTING CODE

Full Code Used (1 of 3)

```
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),
```

Full Code Used (2 of 3)

```
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),
```

Full Code Used (3 of 3)

```
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;
```