codecademy

Churn Rate for CodeFlix

By Teresa Silves

Table of Contents

CodeFlix

CodeFlix Subscription Table

The subscription table has four columns. The first column is the id column. This column is unique and identifies the customer. The second column subscription_start is the date this customer started their subscription. The third column is the date the customer ended their subscription if it is no longer active. The final column is the segment the customer belongs to. Below is the query I used to find this information.

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

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

How Many Different Segments are there in the Subscriptions Table?

By running the query below, I was able to determine that there were 2 distinct segments present in the subscription table. The results were segments 30 & 87.

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

segment
87
30

How Many Different Segments are there in the Subscriptions Table?

By running the query below, I was able to determine that there were 4 months of data recorded in the CodeFlix database. Since the first cancelation took place during the second month data was recorded, the churn rate can only be calculated for the months of January, February and March

```
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

Query Used to Calculate Churn Rate

Range of Months

The first step in determining the churn rate for subscriptions is to create a temporary table called months. The months table contains three rows. One row for each month we will need a churn rate for. Each row contains the first day of that month and the last day of that month.

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

```
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
```

Join Subscription Table and Months Table

The second step in calculating a churn rate is to join the subscription table to the months table using a CROSS JOIN command. This will create a new much larger table that will triple in size. Each row in the subscription table expands into three rows. Each of these rows contains the first and last day of each month the churn rate is being calculated.

```
cross_join AS (
  SELECT *
  FROM subscriptions
  CROSS JOIN months)
```

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
2	2016-12-01	2017-01-24	87	2017-01-01	2017-01-31
2	2016-12-01	2017-01-24	87	2017-02-01	2017-02-28

Find Active Subscribers Results

The next step is to create a new table that uses a CASE statement to assign a value of 1 for each month a subscriber is active in the is_active column for the segment the subscriber is in. If the subscriber is in segment 87 and is active for the given month the is_active_87 column will have a value of 1. The same thing CASE statement is used for subscribers in segment 30, but the is_active_30 column will have a value of 1 instead.

```
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,
```

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

Find Canceled Subscribers Results

If a subscriber cancels their subscription the is_canceled column will be 1 for the month they cancel. If the subscribers is in the 87 segment the is_canceled_87 column will have a value of one. If the subscriber is in segment 30, the is_canceled_30 segment will have a value of 1. This calculation is also done using the CASE statement.
A temporary table is created called status that holds the data calculated using this query.

```
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
```

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

Find Active and Canceled Subscribers Complete Query

```
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
    LIMIT 10;
```

Subscription Table

Before the final query to calculate the churn rate is done, the is_active and is_canceled columns are added to find the total number of subscribers active in a month and the total number of subscribers who cancel each month.

The results from this query are below. This new table is called status_aggregate.

```
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 1;
```

month	sum_active_87	sum_active_30	sum_canceled_87	sum_canceled_30
2017-01-01	278	291	70	22
2017-02-01	462	518	148	38
2017-03-01	531	716	258	84

Churn Rate by Month for each Segment

The final query uses the status_aggregate table to calculate the churn rate for each month. This calculation divides the total number of active subscribers during the given month by the total number of subscribers who cancel in that month. Segment 30 has the lowest churn rate for all three months. The final month of March has the highest churn rate for both segments.

```
SELECT month,
  1.0 * sum_canceled_87/sum_active_87 AS segment_87_churn_rate,
  1.0 * sum_canceled_30/sum_active_30 AS segment_30_churn_rate
FROM status_aggregate;
```

month	segment_87_churn_rate	segment_30_churn_rate
2017-01-01	0.251798561151079	0.0756013745704467
2017-02-01	0.32034632034632	0.0733590733590734
2017-03-01	0.485875706214689	0.11731843575419

Bonus Query

Changes to Query to Include Multiple Segments

The first change I would make in my query would be to the status table. I would first add segment as the third column. I would then change the is_active columns and is_canceled columns to one is_active and one is_canceled column. Below are the results from that query. I compared these results to my previous query and they were the same.

id	month	segment	is_active	is_canceled
1	2017-01-01	87	1	0
1	2017-02-01	87	0	1
1	2017-03-01	87	0	0
2	2017-01-01	87	1	1
2	2017-02-01	87	0	0
2	2017-03-01	87	0	0

```
status As (
  SELECT
        id,
        first_day AS month,
        segment
        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
  )
```

Changes to Query to Include Multiple Segments

The second change I would make in my query would be to the status_aggregate table. I would first add segment as the second column. I would then change the sum_active columns and sum_canceled columns to one sum_active and one sum_canceled column. The last change I made to the status_aggregate query was to add column 2(segment) to the Group By clause. Below are the results from that query. I compared these results to my previous query and they were the same.

month	segment	sum_active	sum_canceled
2017-01-01	30	291	22
2017-01-01	87	278	70
2017-02-01	30	518	38
2017-02-01	87	462	148
2017-03-01	30	716	84
2017-03-01	87	531	258

```
status_aggregate AS (
 SELECT month,
      segment,
      SUM(is_active) AS sum_active,
      SUM(is_canceled) AS sum_canceled
      FROM status
GROUP BY 1, 2)
```

Final Query Grouped by Month then Segment

The final query to calculate the churn rate is below and the results are on the right. The results are now grouped not only by month, but by segment as well. With this grouping the number of segments can increase without changing the query.

```
SELECT month,
  segment,
  1.0 * sum_canceled/sum_active AS churn_rate
FROM status_aggregate;
```

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	0.117318435754189
2017-03-01	87	0.485875706214689

Conclusion

The segment with the highest churn rate is segment 87. I would recommend that CodeFlix concentrate on finding ways to lower the churn rate in this segment. Both segments saw an increase in churn rate during the month of March. This is concerning and would be an area worth investigation.