Capstone: Calculating Churn Rates

Learn SQL From Scratch

by Greg McFarland

Table Of Contents

1.1 Introduction

Codeflix, a streaming video startup, is interested in measuring their user churn rate. This presentation outlines the tasks completed to assist in answering the following questions about their churn:

1. Getting familiar with the company.

- **How many months has the company been operating?**
- **Which months do you have enough information to calculate a churn rate?**
- **What segments of users exist?**

2. What is the overall churn trend since the company started?

3. Compare the churn rates between user segments.

2. Get Familiar With the Company

2.1 Get familiar with the company - Initial Analysis

The company has provided a raw data set named 'subscriptions'. The queries and their sample data sets (below), will yield necessary information to complete the Initial Analysis.

2.1.1 - From the results of this query (LIMIT*ed* to the first 100 rows), we can determine:

- *field names (id, subscription_start, subscription_end and segment).*
- *data formats (i.e., subscription_start and subscription_end are in the YYYY-MM-DD format).*
- *data content (at least 2 segments 87 and 30)*

NOTE: Because the raw data set is larger than 100 records and the possibility that there are a greater number of segments that could fall outside of the returned 100, a 'GROUP BY segment' statement can be inserted into this query that will return all unique segment values.

```
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	30

2.1 Get familiar with the company - Initial Analysis (cont.)

2.1.2 - In order to satisfy another requirement of Initial Analysis, we must determine how many months Codeflix has been doing business. This can be accomplished with the following query that returns the listed results.

```
SELECT

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

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

The result tells us that the oldest subscription_start is 2016-01-01 *(MIN(...))* and the newest subscription_start is 2017-03-30 *(MAX(...))*. A span of almost 4 full months.

2.2 Get familiar with the company - Summary

2.2.1 - Through our Initial Analysis we have been able to answer the following questions:

- How many months has the company been operating? **(2016-12-01 to 2017-03-30) 4 months**
- Which months do you have enough information to calculate a churn rate? **January 2017, February 2017 and March 2017**
- What segments of users exist? **87 and 30**

3. What is the overall churn trend since the company started?

3.1 What is the overall churn trend since the company started?

3.1.1 - Before the 'overall churn trend' can be determined, additional data will be required. A temporary table, called *'months'*, will be created using the *'WITH temporary_name AS(...)'* command, containing the fields 'first_day' and 'last_day. The following query returns the listed results.

```
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
)
SELECT *
FROM months;
```

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

3.1 What is the overall churn trend since the company started?

3.1.2 - Using the resulting temporary table *'months'* from the previous step, we will now create a second temporary table named *'cross_join'* By adding the following command to our query we will *'CROSS JOIN'* the *'subscriptions'* table with the *'months'* table, producing the sample of results on the right.

```
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
)

SELECT *
FROM cross_join
LIMIT 100;
```

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
2	2016-12-01	2017-01-24	87	2017-03-01	2017-03-31

3.1 What is the overall churn trend since the company started?

3.1.3 - The next query in this process uses the *'cross_join'* temporary table as input to create the temporary table named *'status'*. Using *'CASE ..WHEN'* statements, if segment is *'87'* AND the date criteria is true then the field *'is_active_87'* is set to *'1'*, otherwise *'is_active_87'* is set to *'0'*. If segment is *'30'* AND the date criteria is true then the field *'is_active_30'* is set to *'1'*, otherwise *'is_active_30'* is set to *'0'*.

```
...

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 IS NULL)
        ) THEN 1
      ELSE 0
    END AS is_active_87,

    CASE
      WHEN (segment = 30)
        AND (
            (subscription_start < first_day)
             AND (subscription_end IS NULL)
        ) THEN 1
      ELSE 0
    END AS is_active_30

  FROM cross_join
)

SELECT *
FROM status
LIMIT 100;
```

id	month	is_active_87	is_active_30
1	2017-01-01	0	0
1	2017-02-01	0	0
1	2017-03-01	0	0
2	2017-01-01	0	0
2	2017-02-01	0	0
2	2017-03-01	0	0
3	2017-01-01	0	0
3	2017-02-01	0	0
3	2017-03-01	0	0

3.1 What is the overall churn trend since the company started?

3.1.4 - Two additional *'CASE ..WHEN'* statements are added to determine if the subscription is cancelled. If segment is *'87'* AND the date criteria is true then the field *'is_canceled_87'* is set to *'1'*, otherwise, *'is_canceled_87'* is set to *'0'*.
If segment is *'30'* AND the date criteria is true then the field *'is_canceled_30'* is set to *'1'*, otherwise *'is_active_30'* is set to *'0'*.

```
(...)

status AS (
  SELECT id, first_day AS month,
    CASE
      WHEN (segment = 87)
        AND (
            (subscription_start < first_day)
           AND (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_end BETWEEN first_day
           AND last_day)
           ) THEN 1
     ELSE 0
     END AS is_canceled_30
  FROM cross_join
)

SELECT *
FROM status
LIMIT 100;
```

id	month	is_active_87	is_canceled_87	is_active_30	is_canceled_30
1	2017-01-01	0	0	0	0
1	2017-02-01	0	1	0	0
1	2017-03-01	0	0	0	0
2	2017-01-01	0	1	0	0
2	2017-02-01	0	0	0	0
2	2017-03-01	0	0	0	0
3	2017-01-01	0	0	0	0
3	2017-02-01	0	0	0	0
3	2017-03-01	0	1	0	0

3.1 What is the overall churn trend since the company started?

3.1.5 - The following query creates a temporary table *'status_aggregate',* that contains the totals of active and canceled subscriptions by segment.

```
(...)

status_aggregate AS (
  SELECT 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
)
SELECT *
FROM status_aggregate;
```

sum_active_87	sum_active_30	sum_canceled_87	sum_canceled_30
395	1247	476	144

3.1.6 - Finally, we can calculate the overall churn trend over the three month period by adding the following statements to the query:

```
(...)

SELECT (1.0 * (sum_canceled_87 + sum_canceled_30)) /
  (1.0 * (sum_active_87 + sum_active_30)) AS churn_rate
FROM status_aggregate;

SELECT *
FROM status_aggregate;
```

churn_rate
0.377588306942753

3.1 Conclusion

3.1.7 - Overall, the churn trend is 0.377588306942753.

4. Compare the Churn Rates Between Segments.

4.1 - Compare the Churn Rates Between Segments.

4.1.1 - To calculate the churn rates for each segment for the same three month period, the statements in 3.1.6 can be replaced with following statements, yielding the listed results.

```
(...)

SELECT 1.0 * sum_canceled_87 / sum_active_87,
  1.0 * sum_canceled_30 / sum_active_30
FROM status_aggregate;

SELECT *
FROM status_aggregate;
```

1.0 * sum_canceled_87 / sum_active_87	1.0 * sum_canceled_30 / sum_active_30
1.20506329113924	0.115477145148356

4.1 - Conclusion

4.1.2 - From this data set, we can determine that segment 87 has a much higher churn rate than segment 30. More subscribers are canceling segment 87 than are joining over this three month period. While segment 30 shows a lower churn rate in subscribers.

5. Bonus Question.

5.1 - Bonus Question.

5.1.1 - *'How would you modify this code to support a large number of segments?'*. By selecting *'month, segment'* and modifying the *'CASE WHEN (...)'* statements when we create the temporary table *'status'*, as shown in the code snippet below, we are able to capture the subscription data at a less detailed level and then break it down further in the subsequent queries (also shown) producing 1 of several of the sample possible listed results. This query is *'GROUP(ed) BY month, segment.*

```
(...)

status AS (
  SELECT id, first_day AS month, segment,
    CASE
      WHEN (subscription_start < first_day)
        AND (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
)

SELECT month, segment,
       sum(is_active),
       sum(is_canceled),
       1.0 * sum(is_canceled) / SUM(is_active) AS
                    churn_rate
FROM status
GROUP BY month, segment;
```

month	segment	sum(is_active)	sum(is_canceled)	churn_rate
2017-01-01	30	202	22	0.108910891089109
2017-01-01	87	9	70	7.77777777777778
2017-02-01	30	411	38	0.0924574209245742
2017-02-01	87	103	148	1.4368932038835
2017-03-01	30	634	84	0.132492113564669
2017-03-01	87	283	258	0.911660777385159

THE END