Project Assignment: User Churn using SQL

Codeflix, a streaming video startup, is interested in measuring their user churn rate. In this project, you'll be helping them answer these questions about their churn:

1. Get 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.

•Which segment of users should the company focus on expanding?

Four months into launching Codeflix, management asks you to look into subscription churn rates. It's early on in the business and people are excited to know how the company is doing.

The marketing department is particularly interested in how the churn compares between two segments of users. They provide you with a dataset containing subscription data for users who were acquired through two distinct channels.

The dataset provided to you contains one SQL table, subscriptions. Within the table, there are 4 columns:

- id - the subscription id
- subscription_start - the start date of the subscription
- subscription_end - the end date of the subscription
- segment - this identifies which segment the subscription owner belongs to

Codeflix requires a minimum subscription length of 31 days, so a user can never start and end their subscription in the same month.

codecademy

User Churn

Analyze Data with SQL
Delphine Vincent
16/07/2024

Table of Contents

1. Inspect data

1.1 Inspect subscriptions table

```
/* Task 1 - Inspect survey table */
SELECT * FROM subscriptions LIMIT 100;
```

There are 2 segments: 87 and 30.

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
30	2016-12-02	2017-01-20	30
31	2016-12-02		30

1.2 Date range

```
/* Task 2 - Determine the range of months */
SELECT MIN(subscription_start) AS date_min,
MAX(subscription_start) AS date_max
FROM subscriptions;
```

Dates range from 2016-12-01 to 2017-03-30.
4 months are covered.

date_min	date_max
2016-12-01	2017-03-30

2. Churn rates

2.1 Create Month table

```
/* Task 3 - create a temporary table of
months */
WITH months as
(SELECT
  '2016-12-01' as first_day,
  '2016-12-31' as last_day
UNION
SELECT
  '2017-01-01' as first_day,
  '2017-01-31' as last_day
UNION
SELECT
  '2017-02-01' as first_day,
  '2017-02-23' as last_day
UNION
SELECT
  '2017-03-01' as first_day,
  '2017-03-30' as last_day
)
SELECT *
FROM months;
```

first_day	last_day
2016-12-01	2016-12-31
2017-01-01	2017-01-31
2017-02-01	2017-02-23
2017-03-01	2017-03-30

2.2 Create a cross join table

```
/* Task 4 - create a temporary
cross_join, from subscriptions
and months */
-- COPY PASTE CODES FROM 2.1
cross_join AS
(SELECT *
FROM subscriptions
CROSS JOIN months)
SELECT *
FROM cross_join
LIMIT 8;
```

id	subscription_ start	subscription_ end	segment	first_day	last_day
1	2016-12-01	2017-02-01	87	2016-12-01	2016-12-31
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	2016-12-01	2016-12-31
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

2.3 Create a Status table

```
/* Task 5-6 - create a temporary table, status, from the cross_join table */
-- COPY PASTE CODES FROM 2.1-2.2
cross_join AS (
  SELECT subscriptions.*, months.first_day, months.last_day
  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 > 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
)
SELECT * FROM status
LIMIT 10;
```

2.4 Create a Status_Aggregate table

```
/* Task 7 -  Create a status_aggregate temporary table that is a SUM of the active and
canceled subscriptions for each segment, for each month */
-- COPY PASTE CODES FROM 2.1-2.3
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
  GROUP BY month
)
SELECT * FROM status_aggregate;
```

sum_active_87	sum_active_30	sum_canceled_87	sum_canceled_30
0	0	0	0
278	291	70	22
462	518	148	38
531	716	258	84

2.5 Compute churn rates

```
/* Task 8 - Calculate the churn rates for the two segments over the
three month period */
-- COPY PASTE CODES FROM 2.1-2.4
SELECT
  month,
  churn_rate_87,
  churn_rate_30,
  ROUND(churn_rate_87 / churn_rate_30, 1) AS churn_rate_ratio
FROM (
  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
) AS calculated_rates;
```

month	churn_rate_87	churn_rate_30	churn_rate_ratio
2016-12-01			
2017-01-01	0.25	0.08	3.1
2017-02-01	0.32	0.07	4.6
2017-03-01	0.49	0.12	4.1

Segment 30 has a churn rate that is 3.1-4.6 lower than segment 87.
(full code available in file 'SQL_User_Churn_dlf2024.sql').

Codeflix should investigate how those segments differ to try and mitigate future churn rates.