Codeflix Churn Rates

Available segments

First, let's determine which segments Codeflix has. Available segments are easily calculated using `DISTINCT` operator:

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

segment
87
30

Available months

To get the months which we've got data for, let's run a `DISTINCT` query on `date` transformed to the first day of the month. `Date` modifiers such as `'start of month'` allow to transform given date into a new date.

```
SELECT DISTINCT date(subscription_start,'start of month') as "date"
  FROM subscriptions
  UNION
  SELECT DISTINCT date(subscription_end,'start of month')
  FROM subscriptions
  WHERE subscription_end IS NOT NULL;
```

date
2016-12-01
2017-01-01
2017-02-01
2017-03-01

Looks like there's no missing months and we can reuse that query to populate a temporary `months` table which will come very handy in calculating churn.

Auxiliary months table

Let's create a temporary table of months available for aggregation in the form of a months span from first day to last day. We'll do that by populating two temporary tables with first and last days of the months from `subscriptions` using built-in `date` function and then joining them on the same month value retrieved using `strftime`.

```
WITH month_starts AS (
  SELECT DISTINCT date(subscription_start,'start of month') AS first_day
  FROM subscriptions
  UNION
  SELECT DISTINCT date(subscription_end,'start of month')
  FROM subscriptions
  WHERE subscription_end IS NOT NULL
),
month_ends AS (
  SELECT DISTINCT date(subscription_start,'start of month', '+1 month', '-1 day') AS last_day
  FROM subscriptions
  UNION
  SELECT DISTINCT date(subscription_end,'start of month', '+1 month', '-1 day')
  FROM subscriptions
  WHERE subscription_end IS NOT NULL
),
months AS (
  SELECT first_day, last_day
    FROM month_starts
    JOIN month_ends
    ON strftime("%m", first_day) = strftime("%m", last_day)
)
SELECT * FROM months;
```

first_day	last_day
2016-12-01	2016-12-31
2017-01-01	2017-01-31
2017-02-01	2017-02-28
2017-03-01	2017-03-31

Churn trends calculation

Churn is the a quotient of cancellations by number of active users for a given period of time.

To calculate churn trends we need to join available month periods on subscriptions, then designate whether user had subscription or not for each given month using available and cancelled statuses. Then by summation of statuses with grouping by desired periods we can get the numbers necessary for churn calculation.

Let's begin by cross-joining the `subscriptions` and `months` tables and creating a temporary table out of the results.

```
WITH month_starts AS (
  ...
),
month_ends AS (
  ...
),
months AS (
  ...
),
cross_on_months AS (
  select *
  FROM subscriptions
  CROSS JOIN months
)
SELECT *
FROM cross_on_months;
```

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
3	2016-12-01	2017-03-07	87	2016-12-01	2016-12-31
...	...	...	...	...	...

Churn trends calculation (cont)

Let's then create a temporary table that would hold activity and cancellation status values for each of the users for each of the months and call it `active_stats`. Values are calculated as follows:

- `active` - if the month contains a day which falls between `subscription_start` and `subscription_end` dates or if the end date is null, meaning it's not cancelled;
- `cancelled` - if end date falls within a given month's `first_day` and `last_day`.

```
WITH ...
months AS (
  ...
),
cross_on_months AS (
  ...
),
active_stats AS (
  SELECT first_day,
    CASE
      WHEN first_day BETWEEN subscription_start AND subscription_end OR
        (subscription_end BETWEEN first_day AND last_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_cancelled
  FROM cross_on_months)
SELECT *
FROM active_stats;
```

id	first_day	is_active	is_cancelled
1	2016-12-01	1	0
1	2017-01-01	1	0
1	2017-02-01	1	1
1	2017-03-01	0	0
2	2016-12-01	1	0
2	2017-01-01	1	1
2	2017-02-01	0	0
2	2017-03-01	0	0
3	2016-12-01	1	0
...	...	...	...

Virtual `id` column was put here to illustrate the resulting relationships between users, months and statuses.

Churn trends calculation (cont)

Now we can add summation and churn calculation via a temporary table `status_aggregate`

```
WITH ...
active_stats AS (
  …
),
status_aggregate AS (
  SELECT strftime("%Y-%m", first_day) AS mon,
    SUM(is_active) AS sum_active,
    SUM(is_cancelled) AS sum_cancelled
  FROM active_stats
  GROUP BY mon
)
SELECT *
FROM status_aggregate;
```

mon	sum_active	sum_cancelled
2016-12	1394	0
2017-01	1747	92
2017-02	1853	186
2017-03	1722	342

And then calculate churn according to definition.

```
WITH ...
status_aggregate AS (
  ...
)
SELECT mon,
  ROUND((1.0 * sum_cancelled) / sum_active, 2) AS churn
FROM status_aggregate;
```

mon	churn
2016-12	0.0
2017-01	0.05
2017-02	0.1
2017-03	0.2

Since the start of the Company, during the first 4 months of its existence, the overall churn is increasing. Churn for December, 2016 is 0 because no one cancelled their subscription.

Extending and segmenting

Now we can apply grouping to `active_stats` using `GROUP BY` filter by `segment` and `month` to aggregate SUMs of statuses. Now we can filter that table with `WHERE` clause by desired `segment` or `month`.

```
WITH <same as previous query>
active_stats AS (
  SELECT segment,
    first_day,
    CASE
      WHEN first_day BETWEEN subscription_start AND subscription_end OR
        (subscription_end BETWEEN first_day AND last_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_cancelled
  FROM cross_on_months),
status_aggregate AS (
  SELECT segment,
    strftime("%Y-%m", first_day) AS mon,
    SUM(is_active) AS sum_active,
    SUM(is_cancelled) AS sum_cancelled
  FROM active_stats
  GROUP BY segment, mon
)
SELECT segment,
  mon,
  ROUND((1.0 * sum_cancelled) / sum_active, 2) AS churn
FROM status_aggregate;
```

segment	mon	churn
30	2016-12	0.0
30	2017-01	0.02
30	2017-02	0.04
30	2017-03	0.09
87	2016-12	0.0
87	2017-01	0.09
87	2017-02	0.17
87	2017-03	0.33

Churn trends

The table on the previous page shows that churn increases with time in both segments. Churn for December, 2016 is 0 because no one cancelled their subscription that month. Churn in segment 30 is smaller and increases slower than churn in segment 87, meaning, segment 30 is more promising from the point of user retention and the Company should focus on it.