codecademy

Monthly Churn by User Segment at Codeflix

Learn SQL from Scratch

Samuel Campbell | 2018.08.09

Table of Contents

1. Get familiar with Codeflix (1 of 2)

Single-table database with four columns

Database Schema

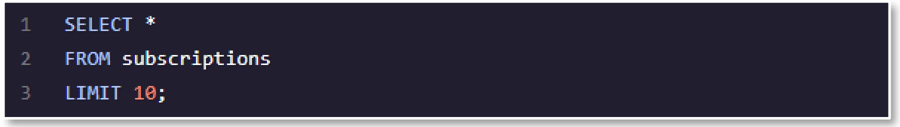
subscriptions	
id	INTEGER
subscription_start	TEXT
subscription_end	TEXT
segment	INTEGER

- "Subscriptions" table contains four columns:
  - id: unique user number assigned to each subscriber
  - subscription_start: the date each user subscribed
  - subscription_end: the date a subscriber canceled ('null' signifies currently active users)
  - segment: user assigned to one of two unique segments—either '87' or '30'

Example: First 10 Rows

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
6	2016-12-01	2017-01-19	87
7	2016-12-01	2017-02-03	87
8	2016-12-01	2017-03-02	87
9	2016-12-01	2017-02-17	87
10	2016-12-01	2017-01-01	87

SQL Code



```sql
SELECT *
FROM subscriptions
LIMIT 10;
```

1. Get familiar with Codeflix (2 of 2)

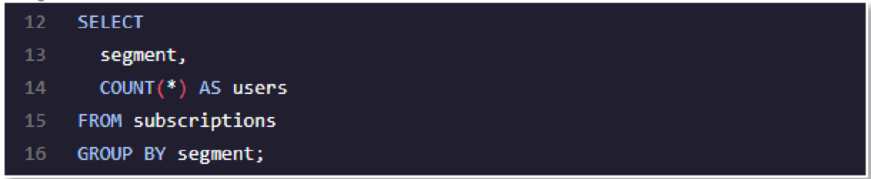
Two user segments (87, 30) and three months of churn data (Jan, Feb, Mar)

User Segments

Codeflix divides users into two segments,’30’ and ’87.’ Each segment has 1,0000 total users

segment	users
30	1000
87	1000

SQL Code



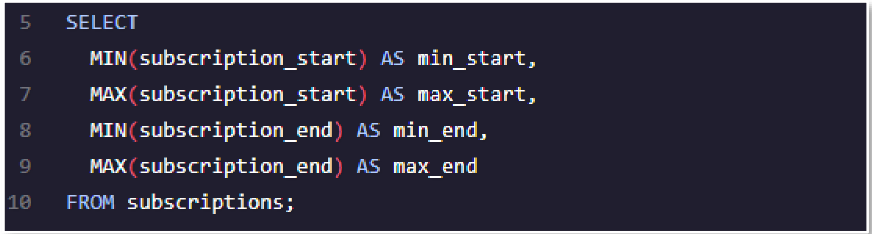
```sql
SELECT
  segment,
  COUNT(*) AS users
FROM subscriptions
GROUP BY segment;
```

Months of Churn Data

Codeflix has been operating for four months, from December 1, 2016 to March 31, 2017. This allows churn rates to be calculated for three months—January, February and March—with December serving as a baseline.

min_start	max_start	min_end	max_end
2016-12-01	2017-03-30	2017-01-01	2017-03-31

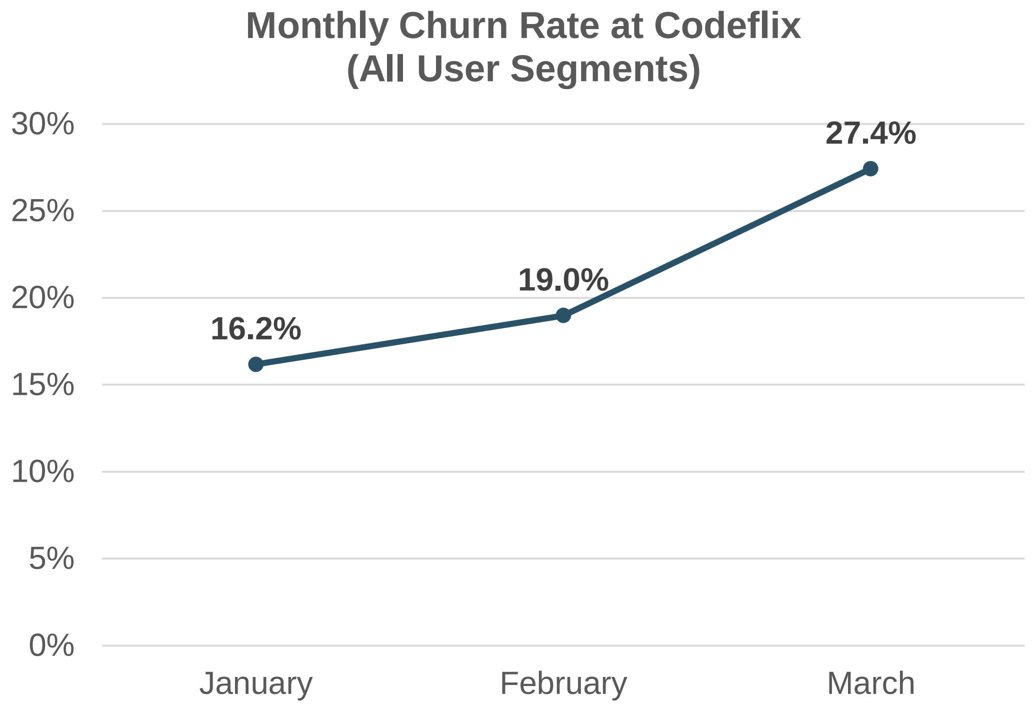
SQL Code



```sql
SELECT
  MIN(subscription_start) AS min_start,
  MAX(subscription_start) AS max_start,
  MIN(subscription_end) AS min_end,
  MAX(subscription_end) AS max_end
FROM subscriptions;
```

2. Overall churn rate by month

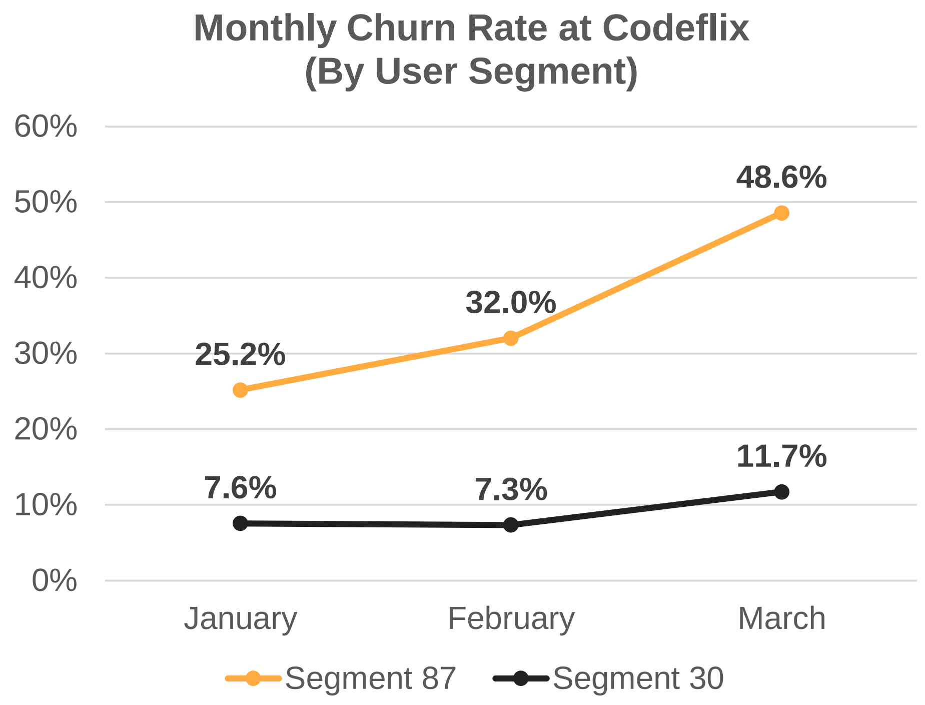
Monthly churn increased steadily from ~16% to >27% during January-March



- Monthly churn rates increased from 16.2% to 27.4% from January to March
- Increase represents an alarming rise of >11pps in only three months
- To build a sustainable business, Codeflix reverse this trend and lower user churn

3. Monthly churn rate by segment

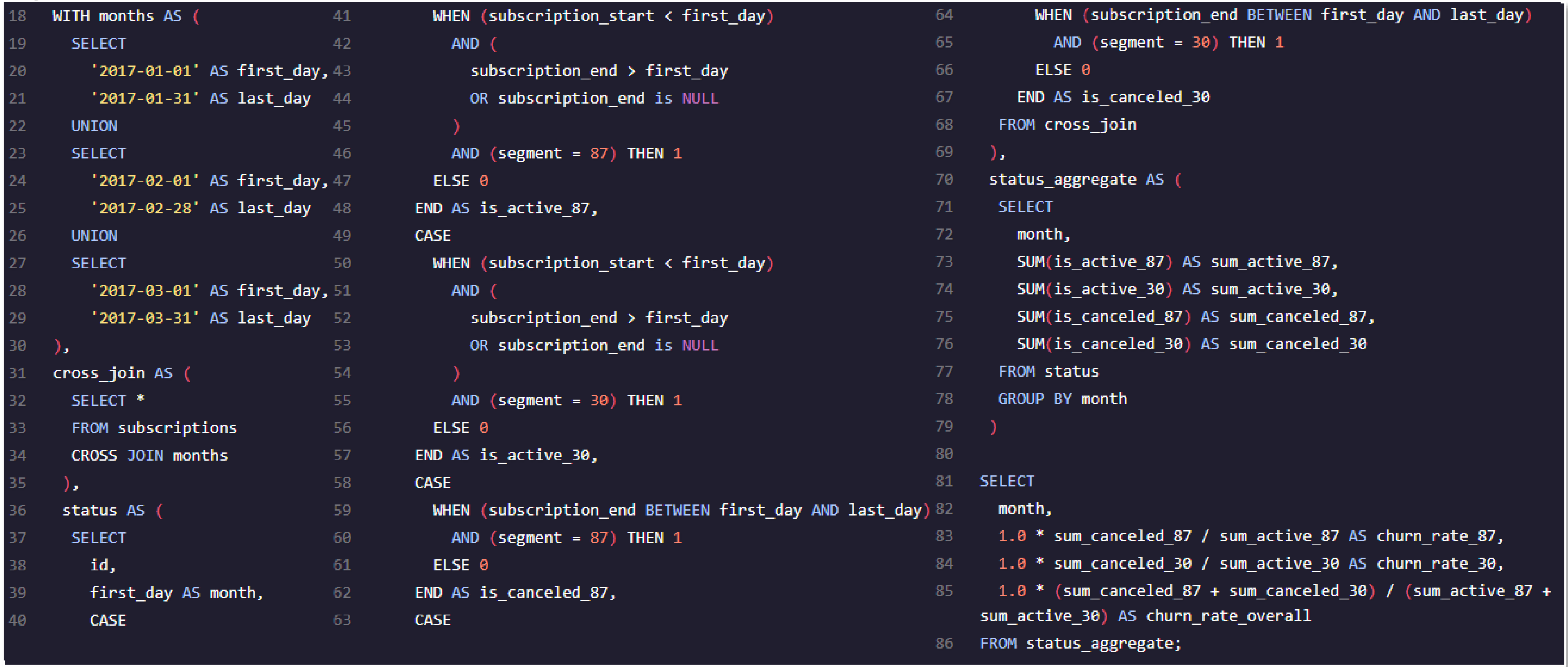
Churn rate is highest for segment 87 and increasingly rapidly each month



- Monthly churn is significantly higher for segment 87 than for segment 30
  - In fact, the churn rate is 3.3X to 4.4X higher for segment 87
- Further, churn rates grew much more rapidly from January to March for segment 87
  - Churn for segment 87 increased by >23pps versus only ~4pps for segment 30
- Moving forward, Codeflix should:
  - Target segment 30 for new user acquisition activities
  - Investigate the root cause of high and growing churn rates for section 87
  - Monitor churn for segment 30 to determine if March uptick is a blip or beginning of a trend

2 and 3. SQL code for churn rate calculations

SQL Code



```sql
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
 ),
 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
 ),
 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,
  1.0 * (sum_canceled_87 + sum_canceled_30) / (sum_active_87 + sum_active_30) AS churn_rate_overall
FROM status_aggregate;
```