CODECADEMY
Learn SQL from Scratch
Sean Dennehy
12/29/2018

$$\text{Churn Rate} = \frac{\text{Cancellations}}{\text{Total Subscribers}}$$

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?

1. Up close and personal with Codeflix

- **Take a look at the first 100 rows of data in the subscriptions table.**

```sql
--Get a feel for the table
SELECT *
FROM subscriptions
LIMIT 100;
```

Id	subscription_start	subscription_end	segment

The dataset provided 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**

- **How many months has the company been operating?**

```sql
--Since when has Codeflix been operating
SELECT MIN(subscription_start) AS 'Operating From',
MAX(subscription_start) AS 'Operating To'
FROM subscriptions;
```

Operating From	Operating To
2016-12-01	2017-03-30

Months Operating
4

1. Up close and personal continued

- **Which months do you have enough information to calculate a churn rate?**

```sql
--For which months can we calculate churn
SELECT MIN(subscription_end) AS 'First Cancel',
       MAX(subscription_end) AS 'Last Cancel'
FROM subscriptions;
```

First Cancel	Last Cancel
2017-01-01	2017-03-31

Months for Churn Calculation
3

- **What segments of users exist?**

```sql
--What segments of users exist
SELECT DISTINCT segment
FROM subscriptions;
```

segment
87
30

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

- **Create a temporary table for the months we have information available to calculate the churn rate from**

```sql
--Create a temporary table for the months
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),
```

- **Create a temporary table, cross_join, from subscriptions and your months. Be sure to SELECT every column**

```sql
--Create temporary table, cross_join, from subscriptions and your months
cross_join AS (
  SELECT *
  FROM subscriptions
  CROSS JOIN months),
```

2. Overall churn trend continued

- Create a temporary table, status, from the cross_join table created earlier. This table should contain:
  - id – selected from cross_join
  - month – as an alias of first_day
  - is_active_87 – created using a CASE WHEN to find any users from segment 87 who existed prior to the beginning of the month. This is 1 if true and 0 otherwise.
  - is_active_30 – created using a CASE WHEN to find any users from segment 30 who existed prior to beginning of the month. This is 1 if true and 0 otherwise.

```
16  --Create temporary table, status, from the cross_join temporary table
17  --for active subscriptions for segment 87
18  status AS (
19    SELECT id,
20    first_day AS month,
21    CASE
22      WHEN (subscription_start < first_day)
23      AND (subscription_end > first_day
24      OR subscription_end IS NULL)
25      AND (segment=87)
26      THEN 1
27      ELSE 0
28    END AS is_active_87,
```

```
29  --for active subscriptions for segment 30
30    CASE
31      WHEN (subscription_start < first_day)
32      AND (subscription_end > first_day
33      OR subscription_end IS NULL)
34      AND (segment=30)
35      THEN 1
36      ELSE 0
37    END AS is_active_30,
```

2. Overall churn trend continued

- **Add an is_canceled_87 and an is_canceled_30 column to the status temporary table. This should be 1 if the subscription is canceled during the month and 0 otherwise.**

```
38  --for canceled subscriptions for segment 87
39    CASE
40      WHEN (subscription_end BETWEEN first_day AND last_day)
41      AND (segment=87)
42      THEN 1
43      ELSE 0
44    END AS is_canceled_87,
```

```
45  --for canceled subscriptions for segment 30
46    CASE
47      WHEN (subscription_end BETWEEN first_day AND last_day)
48      AND (segment=30)
49      THEN 1
50      ELSE 0
51    END AS is_canceled_30
52    FROM cross_join),
```

- **Create a status_aggregate temporary table that is a sum of the active and canceled subscriptions for each segment, for each month. The resulting columns should be:**
  - **sum_active_87**
  - **sum_active_30**
  - **sum_canceled_87**
  - **sum_canceled_30**

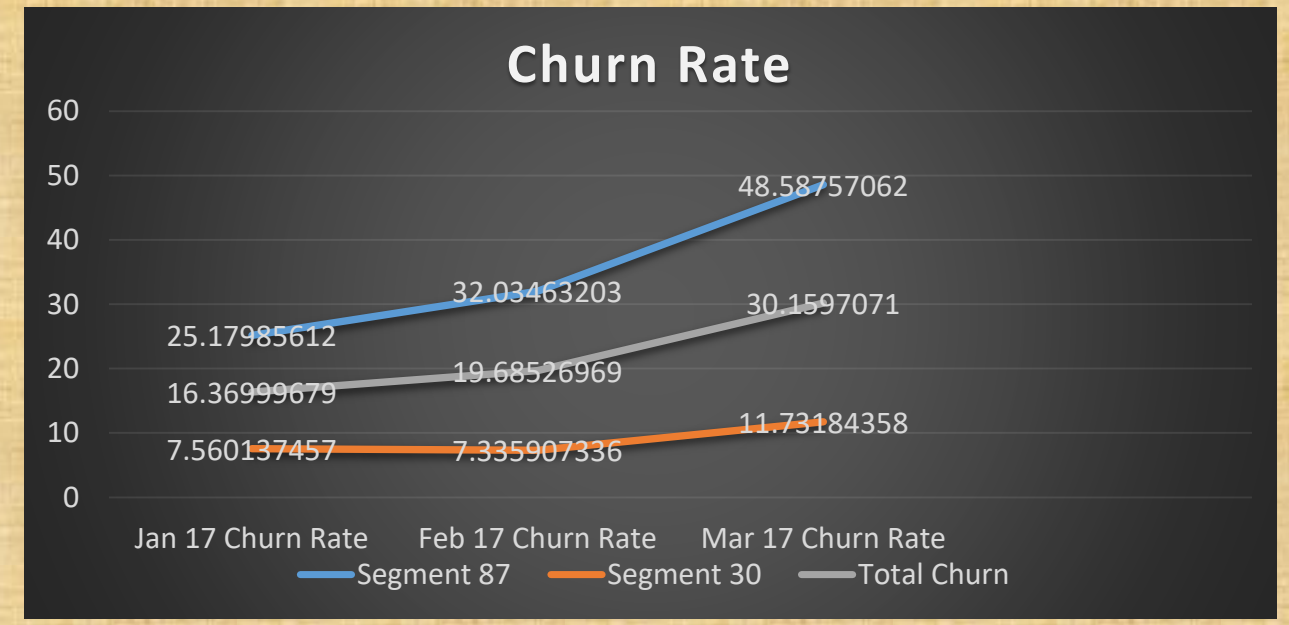
```
53  --Create temporary table, status_aggregate, to sum up active
54  --subscriptions for each segment for each month
55  status_aggregate AS (
56    SELECT month,
57           SUM(is_active_87) AS sum_active_87,
58           SUM(is_active_30) AS sum_active_30,
59           SUM(is_canceled_87) AS sum_canceled_87,
60           SUM(is_canceled_30) AS sum_canceled_30
61    FROM status
62    GROUP BY month)
```

2. Overall churn trend continued

- **Calculate the churn rates for the two segments over the three month period.**

```
63  --Calculate churn rate for each segment
64  SELECT month,
65         1.0 * sum_canceled_87/sum_active_87 AS churn_rate_87,
66         1.0 * sum_canceled_30/sum_active_30 AS churn_rate_30
67  FROM status_aggregate;
```

month	churn_rate_87	churn_rate_30
2017-01-01	0.251798561151079	0.0756013745704467
2017-02-01	0.32034632034632	0.0733590733590734
2017-03-01	0.485875706214689	0.11731843575419

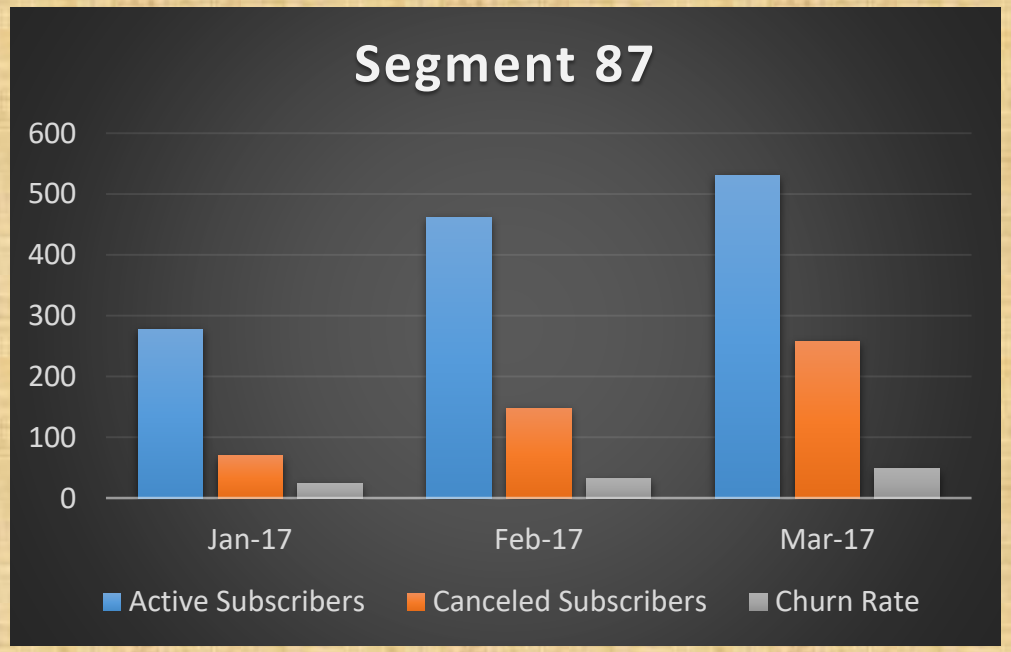


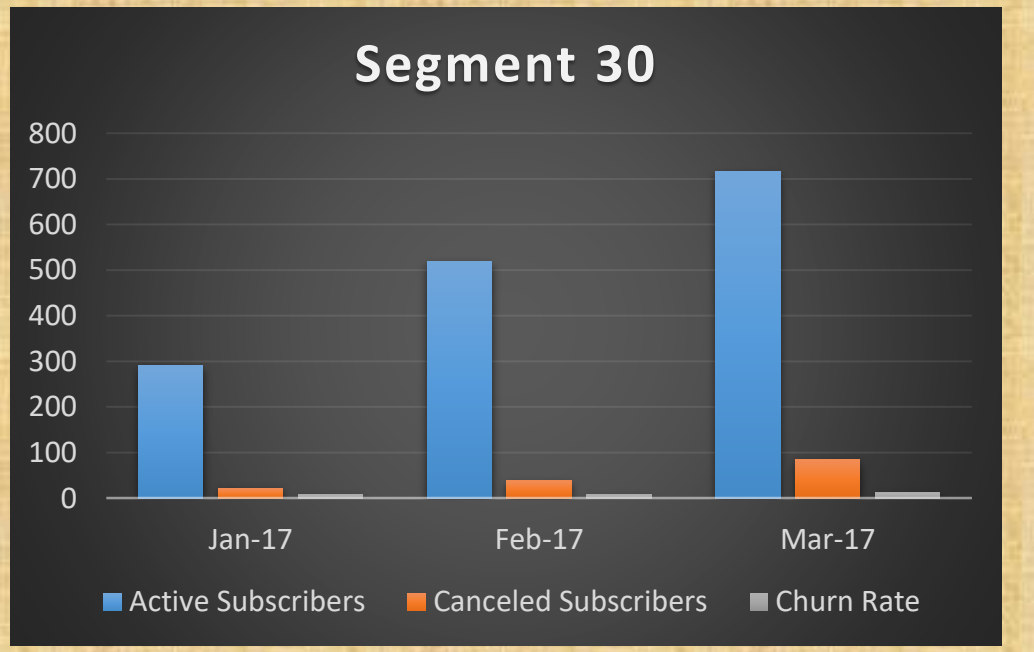
The overall churn rate since the company started is 20.8580436%

3. Compare the churn rates between user segments.

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

Segment 30 has a significantly better churn rate than the segment 87.





Bonus

- **How would you modify this code to support a large number of segments?**

Don't hard code for segments

```
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,
         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),
status_aggregate AS (
  SELECT month,
         segment,
         SUM(is_active) AS sum_active,
         SUM(is_canceled) AS sum_canceled
  FROM status
  GROUP BY month, segment)
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	0.11731843575419
2017-03-01	87	0.485875706214689