1. Get familiar with the company

1.1 How many months have Codeflix been operating?

Method
To know the range of the churn rate, I must first know how long the company has been operating. Given the data available, the first subscription start is my earliest data point. Now the latest data point can be either subscription start or end, since we don't know if new customers start each day. To check the latest data point, I query both the latest subscription start and end. As we can see, Codeflix has been operating from December 2016 to March 2017. We can't calculate churn for december, since there is a one month minimum subscription.

column1	column2	column3
min(subscription_start)	**max(subscription_start)**	**max(subscription_end)**
2016-12-01	2017-03-30	2017-03-31

```
SELECT
min(subscription_start),
max(subscription_start),
max(subscription_end)
FROM subscriptions;
```

1.2 What kind of segments exist?

From the query, we can see that the segments “87” and “30” exist.

segment
87
30

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

2. Overall churn trend

2.1 Date range

First we need to create a temporary table with the months that we want to calculate churn for.

```
 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
),
```

2.2 Date range cross join

In a new temporary table, we cross join our subscribers with the months table. This gives us a temporary table with a row for each subscriber times each month.

```
cross_join AS
(SELECT *
FROM subscriptions
CROSS JOIN months
),
```

2.3 “Marking” active and canceled subscribers

- With our new new cross joined table, we can create yet another temporary table. With status, we make a table with boolean columns that we can later use to count the total amount of active and canceled subscribers.
- Using Case-statements, we can set up rules for “marking” a subscriber with “1” if he/she is active during a month, and the same goes for being canceled.

```
status AS (
  SELECT
       id,
  first_day as month,
  CASE
       WHEN (subscription_start < first_day) AND
(subscription_end IS NULL OR subscription_end >
last_day) THEN 1 ELSE 0
       END AS 'is_active_total',
  CASE
       WHEN (subscription_end BETWEEN first_day AND
last_day) THEN 1 ELSE 0
       END AS 'is_canceled_total',
  ),
```

2.4 Adding it up

- Now it’s time to use aggregate functions. Using the SUM-function, it is possible to add up the total number of rows, where the new “active” and “canceled”-columns have been marked with 1, i.e. counting the total number of active and canceled customers.
- It’s important to group by month in order to view the get the results for each month.

```
status_aggregate AS(
  SELECT
    month,
    SUM(is_active_total) AS sum_active_total,
    SUM(is_canceled_total) AS sum_canceled_total,
    FROM status
    GROUP BY month
  )
```

2.5 Calculation

- In order to calculate churn, we need to divide the cancellations with the total amount of active subscribers (going into the month).
- In order to cast the number as a float, we need to multiply our calculation with 1.0.
- As you can see, the total churn rate has risen dramatically over the months.

Month	churn_rate_total
2017-01-01	0.192468619246862
2017-02-01	0.23279098873592
2017-03-01	0.372955288985823

```
SELECT
month,
1.0*sum_canceled_total/sum_active_total as
churn_rate_total
FROM status_aggregate;
```

3. Segmented churn trend

3.1 Calculating churn by segments

- We now want to compare the churn rates for both segments. In order to do so we add new CASE statements to the “Status” table as show to the right.
- Adding the “segment = “ lets us distinguish between the two segments.

```
status AS (
  SELECT
       id,
  first_day as month,
  CASE
       WHEN (subscription_start < first_day) AND
(subscription_end IS NULL OR subscription_end >
last_day) THEN 1 ELSE 0
       END AS 'is_active_total',
  CASE
       WHEN segment = 87 AND (subscription_start <
first_day) AND (subscription_end IS NULL OR
subscription_end > last_day) THEN 1 ELSE 0
       END AS 'is_active_87',
  CASE
       WHEN segment = 30 AND (subscription_start <
first_day) AND (subscription_end IS NULL OR
subscription_end > last_day) THEN 1 ELSE 0
       END AS 'is_active_30',
  CASE
       WHEN (subscription_end BETWEEN first_day AND
last_day) THEN 1 ELSE 0
       END AS 'is_canceled_total',
  CASE
       WHEN segment = 30 AND (subscription_end
BETWEEN first_day AND last_day) THEN 1 ELSE 0
       END AS 'is_canceled_30',
  CASE
       WHEN segment = 87 AND (subscription_end
BETWEEN first_day AND last_day) THEN 1 ELSE 0
       END as 'is_canceled_87'
FROM cross_join
```

3.2 Calculating churn by segments

- Next, we follow the same process as when we calculated total churn, but we aggregate both segments.
- Lastly, we do the calculations for each segment, again multiplying by 1.0 to cast as float.
- Segment 87 has seen a 65% increased churn rate, where segment 30 has a 35% increase. Therefore, Codeflix should focus on retaining segment 87.

month	churn_rate_total	churn_rate_30	churn_rate_87
2017-01-01	0.192468619246862	0.260223048327138	0.105263157894737
2017-02-01	0.23279098873592	0.308333333333333	0.119122257053292
2017-03-01	0.372955288985823	0.406940063091483	0.296819787985866

```
status_aggregate AS(
  SELECT
    month,
    SUM(is_active_total) AS sum_active_total,
    SUM(is_active_87) AS sum_active_87,
    SUM(is_active_30) AS sum_active_30,
    SUM(is_canceled_total) AS sum_canceled_total,
    SUM(is_canceled_30) AS sum_canceled_87,
    SUM(is_canceled_87) AS sum_canceled_30
    FROM status
    GROUP BY month
  )

SELECT
month,
1.0*sum_canceled_total/sum_active_total as
churn_rate_total,
1.0*sum_canceled_30/sum_active_30 as churn_rate_30,
1.0*sum_canceled_87/sum_active_87 as churn_rate_87
FROM status_aggregate;
```