codecademy

Calculating Churn Rates

Learn SQL from Scratch

Table of Contents

1. Get familiar with Codeflix

1.1 How many months has the company been operating? Which months do you have enough information to calculate a churn rate?

By using a MIN and MAX query on *subscription_start* we can see that the first month of data is December 2016 and the latest end of March 2017. The company has been operating for 4 months.

Codeflix requires a minimum subscription length of 31 days, so a user can never start and end their subscription in the same month. This means we need to look at the 3 first months of 2017 to calculate the churn rate. This is backed by data that confirm there are no *subscription_end* data in December (first value in 2017-01-01)

MIN(subscription_start)	MAX(subscription_start)
2016-12-01	2017-03-30

MIN(subscription_end)	MAX(subscription_end)
2017-01-01	2017-03-31

```
SELECT
MIN(subscription_start),
MAX(subscription_start)
FROM subscriptions;


SELECT
MIN(subscription_end),
MAX(subscription_end)
FROM subscriptions;
```

1.2 What segments of users exist?

By doing a query and only selecting the distinct values from the *segment* column we can tell that there are two segment of users, 30 and 87.

segment
87
30

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

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

2.1 What is the overall churn trend since the company started?

To calculate the overall churn rate by month I first create a temporary table called *months*. This table is then cross joined with subscriptions as *cross_join*. This table is then used to determine active and canceled status in *status*. The results in status is then aggregated and grouped per month. To find the percentage I divide sum canceled on sum active. The end results give the overall churn rate as 16,7% for January, 20% for February and 27,4% for March.

The trend is that the churn rate is increasing every month since the start of the company.

month	churn_rate
2017-01-01	0.161687170474517
2017-02-01	0.189795918367347
2017-03-01	0.274258219727346

```
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,
  SUM (status.is_active) AS sum_active,
  SUM (status.is_canceled) AS sum_canceled
FROM status
GROUP BY 1)
SELECT
 month,
 (1.0 * sum_canceled / sum_active) as 'churn_rate'
FROM status_aggregate;
```

3. Compare the churn rates between user segments

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

To compare the churn rates of the user segments I built on the query in 2.1 by adding segment column and including it into the grouping. I wanted to avoid hardcoding the segment values into the code as it is not good coding practice.

As can be seen of the results, Codeflix should focus on expanding the segment with value 30 as it has much lower churn rates than segment 87. For segment 30 the trend since the start is that the churn rate is more or less stable, starting at 7,5%, before going down to 7,3% and ending on 11,7% for March while for segment 87 the trend is that the churn rate increases every month from 25,2% to 48,6%.

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

```
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 (status.is_active) AS sum_active,
  SUM (status.is_canceled) AS sum_canceled
FROM status
GROUP BY 1,2
)
SELECT
  month, segment,
  (1.0 * sum_canceled / sum_active) as 'churn_rate'
FROM status_aggregate;
```