codecademy

Codeflix’ churn rates

Learn SQL from Scratch

Table of Contents

1. Get familiar with Codeflix

1.1 How many months has the company been operating?

To calculate how many months the company has been operating, I selected the first day of of the table and the last day by using MIN and MAX aggregate functions. Of course the only months we can do calculations on are january to march of 2017 because of the minimum subscriptions length of 31 days.

first_day_operating	last_day_operating
2016-12-01	2017-03-31

```
-- Selecting the first day of operating and the last
day

SELECT MIN(subscription_start) AS
first_day_operating,
       MAX(subscription_end) AS last_day_operating
FROM subscriptions;
```

1.2 What segments of users exist?

To find out what segments of users exist i selected all columns and grouped by segment. The segments are 30 and 87

segment
30
87

```
-- Group by segment to find out what segment of users
exist

SELECT segment
FROM subscriptions
GROUP BY segment;
```

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

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

For this we needed to calculate the churn rate per month. So we had to create some temporary tables. First a month table using the first day and last day of each month. Then a table to see the amount of subscribers and cancellations per month. And use that to calculate the churn rate per month. With this data it is clearly visible the churn rate is getting higher and the overall trend is that more subscribers are cancelling over time, with one exception which we focus on in the next question.

month	churn_rate_87	churn_rate_30
2017-01-01	0.25089605734767	0.0756013745704467
2017-02-01	0.316916488222698	0.0733590733590734
2017-03-01	0.476894639556377	0.116991643454039

```
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
FROM status_aggregate;
```

3. Compare the churn rates between user segments

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

To answer this question we can use the same query results from the previous question. When you compare the results from churn_rate_87 and churn_rate_30 it clearly shows that segment 87 users has substantial higher churn rate then segment 30 users. So we can conclude that the company should be focusing on expanding the segment 30 users. Also what should be noted is that segment 30 users does not have a clear trend as segment 87 of churn rates going up. The churn rate in february is actually slightly lower than in january. March though is substantially higher than february, so the company should focus on users keeping their subscription active.

month	churn_rate_87	churn_rate_30
2017-01-01	0.25089605734767	0.0756013745704467
2017-02-01	0.316916488222698	0.0733590733590734
2017-03-01	0.476894639556377	0.116991643454039

```
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
FROM status_aggregate;
```

4. Bonus question

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

To support a large number of segments we should include a column with all segments, and than GROUP BY segment AND GROUP BY month. This way we get the churn rate per segment per month. First we change the temporary status table by getting rid of the hardcoding of the segments and including the segment column.

```
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_start < first_day) AND
        (subscription_end BETWEEN first_day AND
last_day)
        THEN 1
      ELSE 0
    END as is_canceled
  FROM cross_join
),
```

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

Than we change the status_aggregate table to include again the segment column and simplify the rest of the code because we don't need to hard code the different segments anymore. Just a SUM of all active and canceled users. And we GROUP BY month and segment.

To calculate the churn rates per segment per month we SELECT month and segment and divide sum_canceled with sum_active FROM status_aggregate. I ORDER BY segment to easily see the differences in churn rates between segments.

```
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
FROM status_aggregate
ORDER BY 2;
```