codecademy

Calculating Churn Rates

Analyze Data with SQL

Lukas Heinzl

September 29, 2023

Table of Contents

Table of Contents

1.Inspect subscriptions table

1. Take a look at the first 100 rows of subscriptions.

How many segments do you see?

- Two segments: 87 and 30

```
SELECT *
FROM subscriptions
LIMIT 100;
```

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
11	2016-12-01	2017-01-17	87
12	2016-12-01	2017-02-07	87
13	2016-12-01	Ø	30
14	2016-12-01	2017-03-07	30
15	2016-12-01	2017-02-22	30
16	2016-12-01	Ø	30
17	2016-12-01	Ø	30
18	2016-12-02	2017-01-29	87
19	2016-12-02	2017-01-13	87
20	2016-12-02	2017-01-15	87
21	2016-12-02	2017-01-15	87
22	2016-12-02	2017-01-24	87
23	2016-12-02	2017-01-14	87

2. Determine the range of data.

2. Determine the range of months of data provided.

Range of data is from December 2016 to March 2017

What months will you be able to calculate the churn for?

- January 2017, February, 2017, March 2017
- Cancellations are not allowed in the first 31 days of subscription

MIN(subscription_start)	MAX(subscription_end)
2016-12-01	2017-03-31

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

3. Create months table.

3. Create a temporary table called months.

Temporary table: months
Months table shows the first and last day of each month during the range of months for which we are analzying the churn rates.

```
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
)
SELECT *
FROM months;
```

first_day	last_day
2017-01-01	2017-01-31
2017-02-01	2017-02-28
2017-03-01	2017-03-31

4. Create cross_join table

4. Create a temporary table called cross_join.

- From subscription and months
- Select every column
- Cross_join table combines the subscription table with the months table so viewers can see at glance all the information we are about to analyze for the status of each subscription.

id	subscription_start	subscription_end	segment	first_day	last_day
1	2016-12-01	2017-02-01	87	2017-01-01	2017-01-31
1	2016-12-01	2017-02-01	87	2017-02-01	2017-02-28
1	2016-12-01	2017-02-01	87	2017-03-01	2017-03-31
2	2016-12-01	2017-01-24	87	2017-01-01	2017-01-31
2	2016-12-01	2017-01-24	87	2017-02-01	2017-02-28
2	2016-12-01	2017-01-24	87	2017-03-01	2017-03-31
3	2016-12-01	2017-03-07	87	2017-01-01	2017-01-31
3	2016-12-01	2017-03-07	87	2017-02-01	2017-02-28
3	2016-12-01	2017-03-07	87	2017-03-01	2017-03-31
4	2016-12-01	2017-02-12	87	2017-01-01	2017-01-31
4	2016-12-01	2017-02-12	87	2017-02-01	2017-02-28
4	2016-12-01	2017-02-12	87	2017-03-01	2017-03-31
5	2016-12-01	2017-03-09	87	2017-01-01	2017-01-31
5	2016-12-01	2017-03-09	87	2017-02-01	2017-02-28
5	2016-12-01	2017-03-09	87	2017-03-01	2017-03-31
6	2016-12-01	2017-01-19	87	2017-01-01	2017-01-31
6	2016-12-01	2017-01-19	87	2017-02-01	2017-02-28
6	2016-12-01	2017-01-19	87	2017-03-01	2017-03-31
7	2016-12-01	2017-02-03	87	2017-01-01	2017-01-31
7	2016-12-01	2017-02-03	87	2017-02-01	2017-02-28
7	2016-12-01	2017-02-03	87	2017-03-01	2017-03-31
8	2016-12-01	2017-03-02	87	2017-01-01	2017-01-31
8	2016-12-01	2017-03-02	87	2017-02-01	2017-02-28

```
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
)
SELECT *
FROM cross_join;
```

5. Create status table

5. Create a temporary table called status.

- This table shows the status of active Codeflix users from both segments.
- From cross_join
- Id from cross_join
- Month aliased as first_day
- Is_active_87 from users who subscribed prior to beginning of month
- Is_active_30 from users who subscribed prior to beginning of month

id	month	is_active_87	is_active_30
1	2017-01-01	0	0
1	2017-02-01	0	0
1	2017-03-01	0	0
2	2017-01-01	0	0
2	2017-02-01	0	0
2	2017-03-01	0	0
3	2017-01-01	0	0
3	2017-02-01	0	0
3	2017-03-01	0	0

```
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 segment = 87
    AND (
      subscription_end > first_day
    OR subscription_end IS NULL
     ) THEN 1
    ELSE 0
    END AS is_active_87,
    CASE
    WHEN(subscription_start >
first_day)
    AND segment = 30
    AND (
      subscription_end > first_day
      OR subscription_end IS NULL
       ) THEN 1
    ELSE 0
    END AS is_active_30
    FROM cross_join
    )
    SELECT *
    FROM status;
```

6. Track cancellations in segments

6. Add columns to track cancellation in the segments.

- Now we are tracking both active and cancelled users in each segment.
- Add is_canceled_87
- Add is_canceled_30
- Status temporary table
- 1 if subscription is cancelled during the month
- 0 otherwise

id	month	is_active_87	is_active_30	is_canceled_87	is_canceled_30
1	2017-01-01	0	0	0	0
1	2017-02-01	0	0	1	0
1	2017-03-01	0	0	0	0
2	2017-01-01	0	0	1	0
2	2017-02-01	0	0	0	0
2	2017-03-01	0	0	0	0
3	2017-01-01	0	0	0	0
3	2017-02-01	0	0	0	0
3	2017-03-01	0	0	1	0
4	2017-01-01	0	0	0	0
4	2017-02-01	0	0	1	0
4	2017-03-01	0	0	0	0

```
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 segment = 87
    AND (
      subscription_end > first_day
    OR subscription_end IS NULL
) THEN 1
    ELSE 0
    END AS is_active_87,
  CASE
    WHEN(subscription_start >
first_day)
    AND segment = 30
    AND (
      subscription_end > first_day
      OR subscription_end IS NULL
       ) 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
    )
    SELECT *
    FROM status;
```

7. Create status_aggregate table

7. Create a temporary table called status_aggregate.

```
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 segment = 87
    AND (
      subscription_end > first_day
    OR subscription_end IS NULL
     ) THEN 1
ELSE 0
    END AS is_active_87,
  CASE
    WHEN(subscription_start >
first_day)
    AND segment = 30
    AND (
      subscription_end > first_day
      OR subscription_end IS NULL
       ) 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 1)
SELECT *
FROM status_aggregate;
```

7. Create a temporary table called status_aggregate

- Status_aggregate adds up the statuses of active and cancelled users for each segment to provide an overall total for each month.
- Codeflix should check why so many users are cancelling over time in segment 87. They can survey the users to analyze pain points (such as system errors) and try to fix them. They can also offer more incentives to continue the subscription.
- Sum of active and cancelled subscriptions for each segment, for each month
- Column names: sum_active_87, sum_active_30, sum_canceled_87, sum_canceled_30
- See previous slide for code of creating status_aggregate temporary table.

month	sum_active_87	sum_active_30	sum_canceled_87	sum_canceled_30
2017-01-01	715	703	70	22
2017-02-01	459	450	148	38
2017-03-01	229	213	258	84

8. Calculate the churn rates

8.1 Calculate the churn rates for the two segments over the three month time period.

```
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 segment = 87
    AND (
      subscription_end > first_day
    OR subscription_end IS NULL
     ) THEN 1
```

```
ELSE 0
    END AS is_active_87,
  CASE
    WHEN(subscription_start >
first_day)
    AND segment = 30
    AND (
      subscription_end > first_day
      OR subscription_end IS NULL
       ) 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 1)
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;
```

8.1 Calculate the churn rates for the two segments over the three month time period.

- See previous slide for code of creating status_aggregate temporary table and calculating churn rates.
- Segment 87 had a particularly high churn rate in March.

month	churn_rate_87	churn_rate_30
2017-01-01	0.0979020979020979	0.0312944523470839
2017-02-01	0.32244008714597	0.0844444444444444
2017-03-01	1.12663755458515	0.394366197183099

8.2 Which segment has a lower churn rate?

```
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 segment = 87
    AND (
      subscription_end > first_day
    OR subscription_end IS NULL
     ) THEN 1
ELSE 0
    END AS is_active_87,
  CASE
    WHEN(subscription_start >
first_day)
    AND segment = 30
    AND (
      subscription_end > first_day
      OR subscription_end IS NULL
       ) 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 1),
churn_rate_summary AS (
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)
SELECT
SUM(churn_rate_87) AS
churn_rate_87_total,
   SUM(churn_rate_30) AS
churn_rate_30_total
FROM churn_rate_summary;
```

8.2 Which segment has a lower churn rate?

- See previous slide for code of creating status_aggregate temporary table and calculating churn rates.
- Segment 30 has a lower churn rate.
- Codeflix should try to analyze why segment 30 has a lower churn rate by surveying customer's satisfaction and try to see if they can retain more customers in segment 87 by raising customer satisfaction by fixing pain points and providing incentives to continue subscription.

churn_rate_87_total	churn_rate_30_total
1.54697973963322	0.510105093974627