CALCULATING CHURN

What is Churn?

A common revenue model for SaaS (Software as a service) companies is to charge a monthly subscription fee for access to their product. Frequently, these companies aim to continually increase the number of users paying for their product. One metric that is helpful for this goal is churn rate.
Churn rate is the percent of subscribers that have canceled within a certain period, usually a month. For a user base to grow, the churn rate must be less than the new subscriber rate for the same period.
To calculate the churn rate, we only will be considering users who are subscribed at the beginning of the month. The churn rate is the number of these users who cancel during the month divided by the total number:

$$\frac{\text{cancellations}}{\text{total subscribers}}$$

For example, suppose you were analyzing data for a monthly video streaming service called CodeFlix. At the beginning of February, CodeFlix has 1,000 customers. In February, 250 of these customers cancel. The churn rate for February would be:

```
250 / 1000 = 25% churn rate
```

1. In March, CodeFlix started with 2,000 customers. During the month, 100 of these customers canceled. What is the March churn rate as a ratio? Use a `SELECT` statement to calculate the answer. Be sure to use a decimal in your calculations to force a float answer.

```
select 100./2000;
```

2. In April, CodeFlix started with 3,000 customers. During the month, 450 of these customers canceled. The service added 500 new customers during the same period. What is the April churn rate as a ratio? On a new line, use a `SELECT` statement to calculate the answer. Be sure to use a decimal in your calculations to force a float answer.

```
select 450./3000;
```

Single Month I

Now that we've gone over what churn is, let's see how we can calculate it using SQL. In this example, we'll calculate churn for the month of December 2016.

Typically, there will be data in a `subscriptions` table available in the following format:

- `id` - the customer id
- `subscription_start` - the subscribe date
- `subscription_end` - the cancel date

When customers have a `NULL` value for their `subscription_end`, that's a good thing. It means they haven't canceled!

Remember from the previous exercise that churn rate is:

$$\frac{\text{cancellations}}{\text{total subscribers}}$$

For the numerator, we only want the portion of the customers who cancelled during December:

```
SELECT COUNT(*)
FROM subscriptions
WHERE subscription_start < '2016-12-01'
  AND (
    subscription_end
    BETWEEN '2016-12-01' AND '2016-12-31'
  );
```

For the denominator, we only want to be considering customers who were active at the beginning of December:

```
SELECT COUNT(*)
FROM subscriptions
WHERE subscription_start < '2016-12-01'
  AND (
    (subscription_end >= '2016-12-01')
    OR (subscription_end IS NULL)
  );
```

You might've noticed there are quite a few parentheses in these two queries.

When there are multiple conditions in a `WHERE` clause using `AND` and `OR`, it's the best practice to always use the parentheses to enforce the order of execution. It reduces confusion and will make the code easier to understand. The condition within the brackets/parenthesis will always be executed first.
Anyways, now that we have the users who canceled during December, and total subscribers, let's divide the two to get the churn rate.

When dividing, we need to be sure to multiply by `1.0` to cast the result as a float:

```
SELECT 1.0 *
(
  SELECT COUNT(*)
  FROM subscriptions
  WHERE subscription_start < '2016-12-01'
  AND (
    subscription_end
    BETWEEN '2016-12-01'
    AND '2016-12-31'
  )
) / (
  SELECT COUNT(*)
  FROM subscriptions
  WHERE subscription_start < '2016-12-01'
  AND (
    (subscription_end >= '2016-12-01')
    OR (subscription_end IS NULL)
  )
)
AS result;
```

Here, we have the numerator divided by the denominator, and then multiplying the answer by `1.0`. At the very end, we are renaming the final answer to `result` using `AS`.

1. We've imported 4 months of data for a company from when they began selling subscriptions. This company has a minimum commitment of 1 month, so there are no cancellations in the first month. The `subscriptions` table contains:
    - `id`
    - `subscription_start`
    - `subscription_end`

Use the methodology provided in the narrative to calculate the churn for January 2017.

```
SELECT 1.0 *
(
  SELECT COUNT(*)
  FROM subscriptions
  WHERE subscription_start < '2017-01-01'
  AND (
    subscription_end
    BETWEEN '2017-01-01'
    AND '2017-01-31'
  )
) / (
  SELECT COUNT(*)
  FROM subscriptions
  WHERE subscription_start < '2017-01-01'
  AND (
    (subscription_end >= '2017-01-01')
    OR (subscription_end IS NULL)
  )
)
AS result;
```

Single Month II

The previous method worked, but you may have noticed we selected the same group of customers twice for the same month and repeated a number of conditional statements.

Companies typically look at churn data over a period of many months. We need to modify the calculation a bit to make it easier to mold into a multi-month result. This is done by making use of `WITH` and `CASE`.

To start, use `WITH` to create the group of customers that are active going into December:

```
WITH enrollments AS
(SELECT *
FROM subscriptions
WHERE subscription_start < '2016-12-01'
AND (
  (subscription_end >= '2016-12-01')
  OR (subscription_end IS NULL)
)),
```

Let’s create another temporary table that contains an `is_canceled` status for each of these customers. This will be 1 if they cancel in December and 0 otherwise (their cancellation date is after December or `NULL`).

```
status AS
(SELECT
CASE
  WHEN (subscription_end > '2016-12-31')
    OR (subscription_end IS NULL) THEN 0
    ELSE 1
  END as is_canceled,
...
```

We could just `COUNT()` the rows to determine the number of users. However, to support the multiple month calculation, lets add a `is_active` column to the `status` temporary table. This uses the same condition we created `enrollments` with:

```
status AS
  ...
  CASE
    WHEN subscription_start < '2016-12-01'
      AND (
        (subscription_end >= '2016-12-01')
        OR (subscription_end IS NULL)
      ) THEN 1
    ELSE 0
  END as is_active
  FROM enrollments
  )
```

This tells us if someone is active at the beginning of the month.
The last step is to do the math on the `status` table to calculate the month's churn:

```
SELECT 1.0 * SUM(is_canceled) / SUM(is_active)
FROM status;
```

We make sure to multiply by `1.0` to force a float result instead of an integer.

1. Use the methodology provided in the narrative to calculate the churn for January 2017.
The `subscriptions` table contains:

- `id`
- `subscription_start`
- `subscription_end`

```
SELECT *
FROM subscriptions
WHERE subscription_start < '2017-01-01'
AND (
(subscription_end >= '2017-01-01')
OR (subscription_end IS NULL)
)
limit 5;

WITH enrollments AS
(SELECT *
FROM subscriptions
WHERE subscription_start < '2017-01-01'
AND (
(subscription_end >= '2017-01-01')
OR (subscription_end IS NULL)
)), status AS
(SELECT
CASE
WHEN (subscription_end > '2017-01-31')
OR (subscription_end IS NULL) THEN 0
ELSE 1
END as is_canceled,
CASE
WHEN subscription_start < '2017-01-01'
AND (
(subscription_end >= '2017-01-01')
OR (subscription_end IS NULL)
) THEN 1
ELSE 0
END as is_active
FROM enrollments
)
SELECT *
FROM status
limit 5;

WITH enrollments AS
(SELECT *
FROM subscriptions
WHERE subscription_start < '2017-01-01'
AND (
(subscription_end >= '2017-01-01')
OR (subscription_end IS NULL)
)), status AS
(SELECT
CASE
WHEN (subscription_end > '2017-01-31')
OR (subscription_end IS NULL) THEN 0
ELSE 1
END as is_canceled,
CASE
WHEN subscription_start < '2017-01-01'
AND (
(subscription_end >= '2017-01-01')
OR (subscription_end IS NULL)
) THEN 1
ELSE 0
END as is_active
FROM enrollments
)
SELECT 1.0 * SUM(is_canceled) / SUM(is_active) as churn
FROM status;
```

Multiple Month: Create Months Temporary Table

Our single month calculation is now in a form that we can extend to a multiple month result. But first, we need months! Some SQL table schemes will contain a prebuilt table of months. Ours doesn’t, so we’ll need to build it using `UNION`. We’ll need the first and last day of each month.
Our churn calculation uses the first day as a cutoff for subscribers and the last day as a cutoff for cancellations.
This table can be created like:

```
SELECT
  '2016-12-01' AS first_day,
  '2016-12-31' AS last_day
UNION
SELECT
  '2017-01-01' AS first_day,
  '2017-01-31' AS last_day;
```

1. We will be using the months as a temporary table (using `WITH`) in the churn calculation. Create the `months` temporary table using `WITH` and `SELECT` everything from it so that you can see the structure.We need a table for January, February, and March of 2017.

```
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;
```

Multiple Month: Cross Join Months and Users

Now that we have a table of months, we will join it to the subscriptions table. This will result in a table containing every combination of month and subscription. Ultimately, this table will be used to determine the status of each subscription in each month.

1. The workspace contains the `months` temporary table from the previous exercise. Create a `cross_join` temporary table that is a `CROSS JOIN` of `subscriptions` and `months`. We've added:

```
SELECT *
FROM cross_join
LIMIT 100;
```

at the bottom of this exercise so you can visualize the temporary table you create.
It should `SELECT` all the columns from the temporary table.

```
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
LIMIT 100;
```

Multiple Month: Determine Active Status

We now have a cross joined table that looks something like:

id	subscription_start	subscription_end	month
1	2016-12-03	2017-02-15	2016-12-01
1	2016-12-03	2017-02-15	2017-01-01
1	2016-12-03	2017-02-15	2017-02-01
1	2016-12-03	2017-02-15	2017-03-01

If you remember our single month example, our ultimate calculation will make use of the `status` temporary table. The first column of this table was used in the denominator of our churn calculation:

- `is_active`: if the subscription started before the given month and has not been canceled before the start of the given month

For the example above, this column would look like:

month	is_active
2016-12-01	0
2017-01-01	1
2017-02-01	1
2017-03-01	0

1. Add a `status` temporary table. This table should have the following columns:

- `id` - selected from the `cross_join` table
- `month` - this is an alias of `first_day` from the `cross_join` table. We're using the first day of the month to represent which month this data is for.
- `is_active` - 0 or 1, derive this column using a `CASE WHEN` statement

The `is_active` column should be 1 if the `subscription_start` is before the month's `first_day` and if the `subscription_end` is either after the month's `first_day` or is `NULL`. We've added:

```
SELECT *
FROM status
LIMIT 100;
```

at the bottom of this exercise so you can visualize the temporary table you create.

```
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 (
      subscription_end > first_day
      OR subscription_end IS NULL
    ) THEN 1
  ELSE 0
END as is_active
FROM cross_join)
SELECT *
FROM status
LIMIT 100;
```

Multiple Month: Determine Cancellation Status

For our calculation, we'll need one more column on the `status` temporary table: `is_canceled`
This column will be `1` only during the month that the user cancels. From the last exercise, the sample user had a `subscription_start` on `2016-12-03` and their `subscription_end` was on `2017-02-15`. Their complete status table should look like:

month	is_active	is_canceled
2016-12-01	0	0
2017-01-01	1	0
2017-02-01	1	1
2017-03-01	0	0

In our examples, our company has a minimum subscription duration of one month. This means that the `subscription_start` always falls before the beginning of the month that contains their `subscription_end`. Outside of our examples, this is not always the case, and you may need to account for customers canceling within the same month that they subscribe.

1. Add an `is_canceled` column to the `status` temporary table. Ensure that it is equal to `1` in months containing the `subscription_end` and `0` otherwise. Derive this column using a `CASE WHEN` statement. You can use the `BETWEEN` function to check if a date falls between two others. We've added:

```
SELECT *
FROM status
LIMIT 100;
```

```
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 (
      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)
SELECT *
FROM status
LIMIT 100;
```

Multiple Month: Sum Active and Canceled Users

Now that we have an active and canceled status for each subscription for each month, we can aggregate them. We will `GROUP BY` month and create a `SUM()` of the two columns from the `status` table, `is_active` and `is_canceled`. This provides a list of months, with their corresponding number of active users at the beginning of the month and the number of those users who cancel during the month.

1. Add a `status_aggregate` temporary table. This table should have the following columns:

- `month` - selected from the `status` table
- `active` - the `SUM()` of active users for this month
- `canceled` - the `SUM()` of canceled users for this month

We've added:

```
SELECT *
FROM 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 (
      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(is_active) as active,
  SUM(is_canceled) as canceled
FROM status
GROUP BY month)
SELECT *
FROM status_aggregate;
```

Multiple Month: Churn Rate Calculation

Now comes the moment we've been waiting for - the actual churn rate.
We use the number of canceled and active subscriptions to calculate churn for each month: `churn_rate = canceled / active`

1. Add a `SELECT` statement to calculate the churn rate. The result should contain two columns:

- `month` - selected from `status_aggregate`
- `churn_rate` -calculated from `status_aggregate.canceled` and `status_aggregate.active`.

```
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 (
      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(is_active) as active,
  SUM(is_canceled) as canceled
FROM status
GROUP BY month)
SELECT
  month,
  1.0 * canceled/active AS churn_rate
FROM status_aggregate;
```

Calculating Churn Review

You can now calculate a churn rate over time for a company:

month	churn_rate
2017-01-01	0.127
2017-02-01	0.125
2017-03-01	0.237

It looks like our sample company did something in March that doubled the churn rate. They might want to look into that...

In this lesson you learned:

- The churn rate is a percent of subscribers at the beginning of a period that cancel within that period. "Monthly churn" is a typical metric and what was used in the examples.
- How to calculate this metric using SQL for a single month. This used `COUNT()` and conditions to determine the number of subscribers that were active and how many canceled.
- A more complex method to track the subscriber churn rate over many months.