## 1. Aliasing and arithmetic

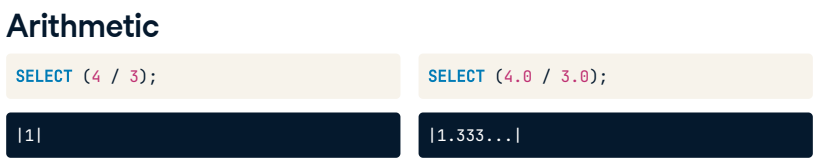
Welcome back! In this lesson, we will learn about using more arithmetic in our queries and take a closer look at aliasing with AS.

## 2. Arithmetic



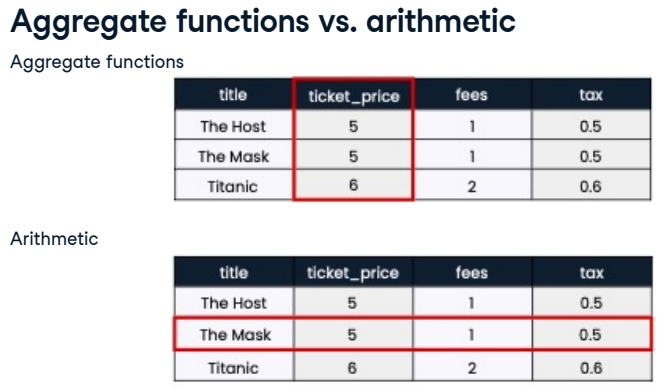
We can perform basic arithmetic with symbols like plus, minus, multiply, and divide. Using parentheses with arithmetic indicates to the processor when the calculation needs to execute. Here are some basic examples of how we can use arithmetic in SQL. We can add, subtract, multiply, and divide as follows. In these examples, the parentheses are not required as only one calculation takes place; however, they provide more clarity to the code. But, the division gives a result of one; why is that?

## 3. Arithmetic



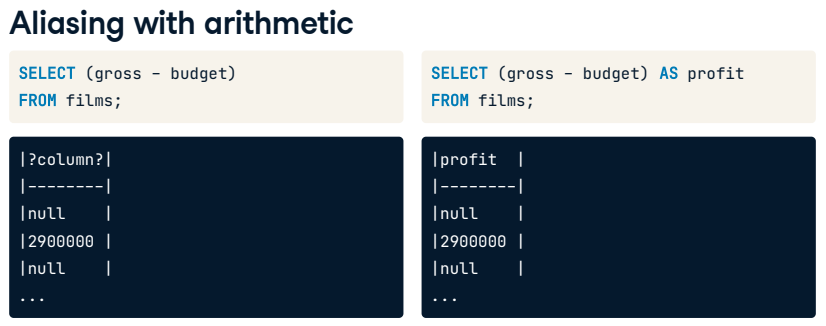
Similar to other programming languages, SQL assumes that we want to get an integer back if we divide an integer by an integer. So be careful! When dividing, we can add decimal places to our numbers if we want more precision. For example, SELECT four-point-zero divided by three-point-zero gives us the result we would expect: 1-point-3 repeating.

## 4. Aggregate functions vs. arithmetic



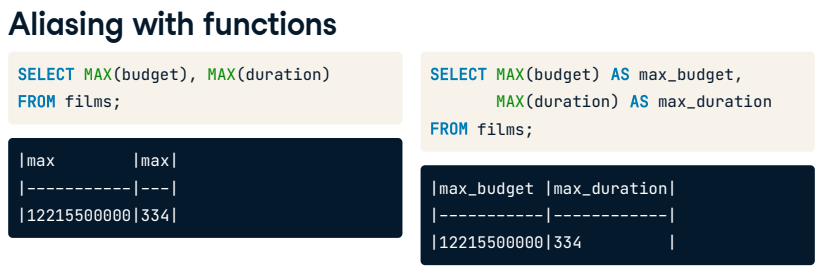
What's the difference between using aggregate functions and arithmetic? The key difference is that aggregate functions, like SUM, perform their operations on the fields vertically while arithmetic adds up the records horizontally.

## 5. Aliasing with arithmetic



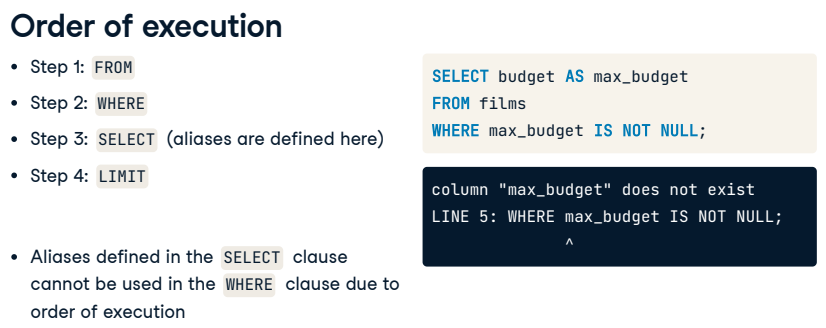
Before we move on, let's run through an arithmetic example using our database. Here we have selected the gross, how much the movie made, minus the budget, how much the movie cost, from our films table. The result is the amount of profit. Notice that the query's result doesn't give us a defined field name. We will always need to use an alias when summarizing data with aggregate functions and arithmetic.

## 6. Aliasing with functions



As we progress and learn how to manipulate our data, it will be even more important to keep our field names clear. For example, if we're using multiple MAX functions in one query, we'll have two fields named max, which isn't very useful! This is a situation when it's especially important to alias like we do here.

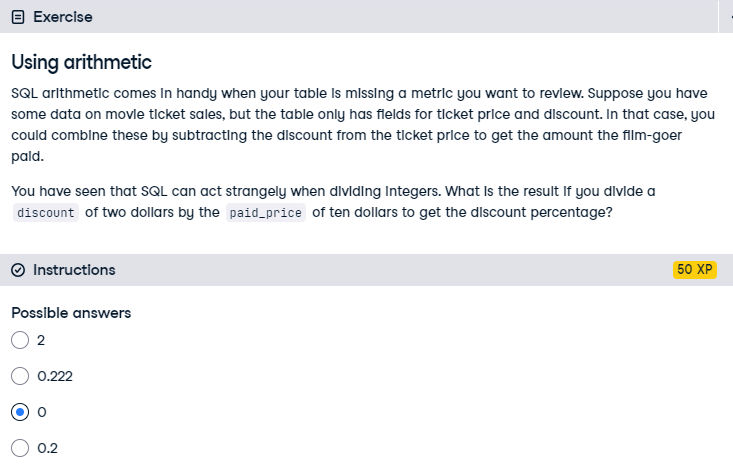
## 7. Order of execution



Let's explore how using an alias fits into the SQL execution order. Here is a reminder of the order of execution we know so far: SQL will process the FROM statement first, followed by the WHERE clause, then the SELECT statement, and finally, LIMIT. When adding an alias for a field name in the SELECT clause, we might assume we could use it later in our query with the WHERE clause. Unfortunately, that is not possible; as we can see by the order of execution, the query would not have created the alias yet, and our code would generate an error.

## 8. Let's practice!

Time for more practice.



Correct! SQL thinks we want the answer to be an integer since we are dividing two integers. 0 is the closest integer to 0.2.

## Exercise

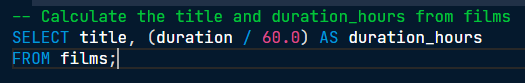
# Aliasing with functions

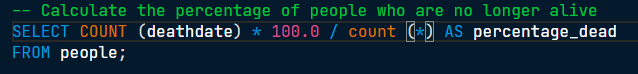
Aliasing can be a lifesaver, especially as we start to do more complex SQL queries with multiple criteria. Aliases help you keep your code clean and readable. For example, if you want to find the MAX() value of several fields without aliasing, you'll end up with the result with several columns called max and no idea which is which. You can fix this with aliasing.

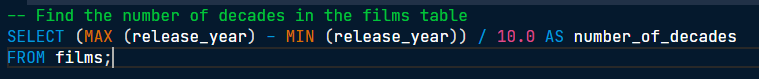
Now, it's over to you to clean up the following queries.

## Instructions

* + Select the title and duration in hours for all films and alias as duration\_hours; since the current durations are in minutes, you'll need to divide duration by 60.0.
  + Calculate the percentage of people who are no longer alive and alias the result as percentage\_dead.
  + Find how many decades (period of ten years) the films table covers by using MIN() and MAX(); alias as number\_of\_decades.







Amazing work mastering arithmetic, aggregate functions, and aliasing! Now you know that our films table covers films released over one hundred years!

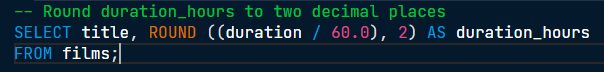
## Exercise

# Rounding results

You found some valuable insights in the previous exercise, but many of the results were inconveniently long. We forgot to round! We won't make you redo them all; however, you'll update the worst offender in this exercise.

## Instructions

* Update the query by adding ROUND() around the calculation and round to two decimal places.



That's better! Now you can clearly see how long a movie is.