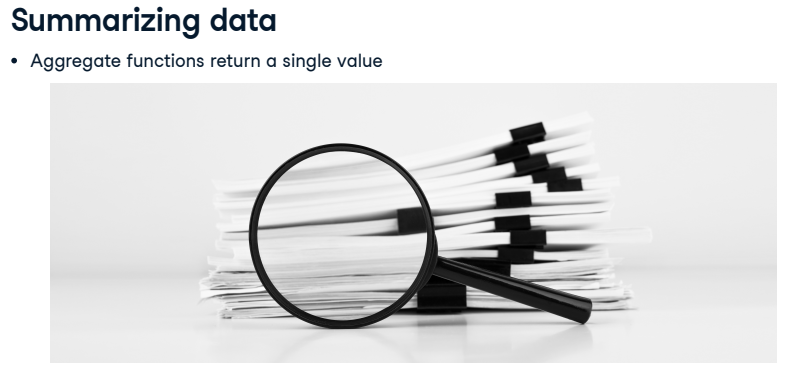
## 1. Summarizing data

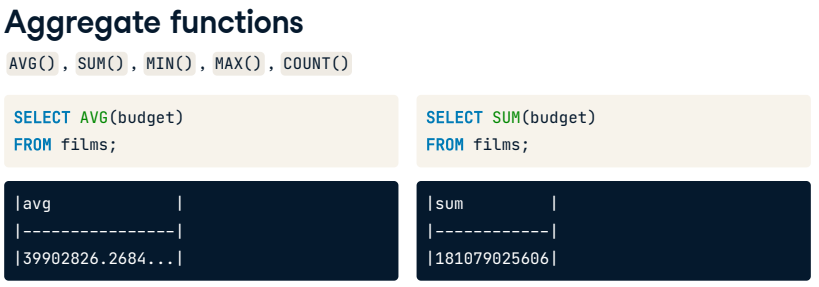
Fantastic job getting this far! Let's learn about summarizing data with aggregate functions.

## 2. Summarizing data



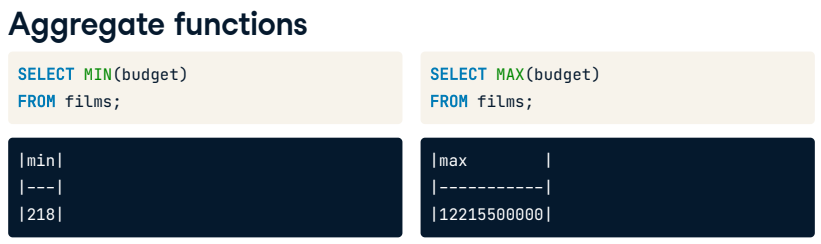
When analyzing data, we often want to understand the dataset as a whole in addition to looking at individual records. One way to do this is to summarize the data using SQL's aggregate functions. An aggregate function performs a calculation on several values and returns a single value.

## 3. Aggregate functions



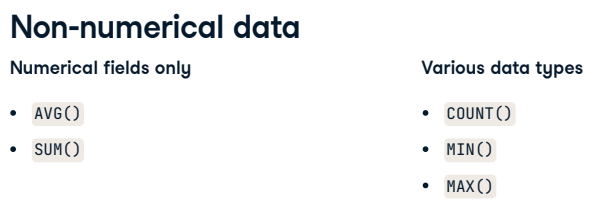
We already know one aggregate function, COUNT()! We'll now learn four new aggregate functions, allowing us to find the average, sum, minimum, and maximum of a specified field. Let's look at some examples of how they work. These aggregate functions come after SELECT, exactly like COUNT(). This first query gives us the average value from the budget field of the films table. That's an average of over 39 million per film in the films table. The SUM() function returns the result of adding the values in the budget field. Here, the total budget of all the films is over 181 billion!

## 4. Aggregate functions



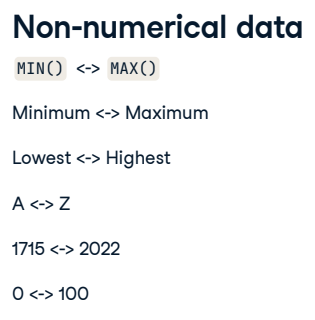
The MIN() function returns the lowest budget, and the MAX() function returns the highest budget. The 2006 South Korean movie "The Host" had a budget of over 12 billion. That sounds huge, but our data is in multiple currencies, so a true comparison would require currency exchange rates as well. Note that we operate on the field (or column) with all of these aggregate functions, not the records (or rows).

## 5. Non-numerical data



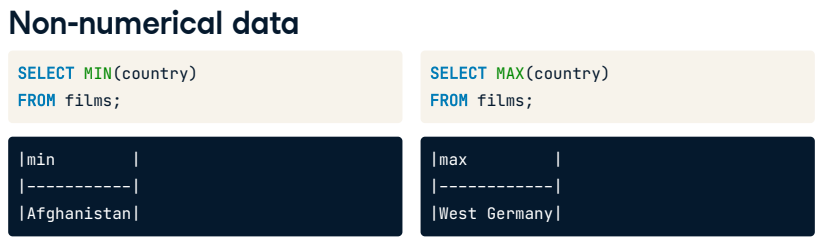
Although these functions appear to be mathematical, we can use several of them with both numerical and non-numerical fields. Average and sum are the two aggregate functions we can only use on numerical fields since they require arithmetic. We can use count, minimum, and maximum with non-numerical fields as we already saw with COUNT() in previous lessons.

## 6. Non-numerical data



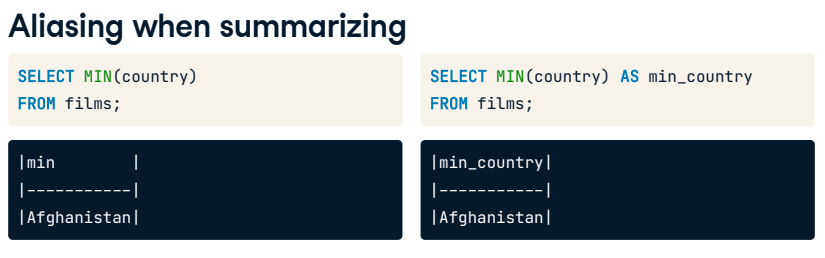
COUNT() can provide a total of any non-missing, or not null, records in a field regardless of their type. Similarly, minimum and maximum will give the record that is figuratively the lowest or highest. Lowest can mean the letter A when dealing with strings or the earliest date when dealing with dates. And, of course, with numbers, it is the highest or the lowest number.

## 7. Non-numerical data



Here are some examples of using these functions with non-numerical fields. We can select the minimum and maximum country from the films database and see that our minimum country, or country that would come first in the alphabet, is Afghanistan. In contrast, West Germany is the maximum country that would come last in the alphabet. Our database does not contain any films made in Zambia or Zimbabwe, but it does have at least one film made back when Germany was two different countries!

## 8. Aliasing when summarizing



Notice how all query results have automatically updated the field name to the function. In this case, min. It's best practice to use an alias when summarizing data so that our results are clear to anyone reading our code.

## 9. Let's practice!

That's it! Let's get on to practice!

##### Exercise

#### Aggregate functions and data types

Aggregate functions are another valuable tool for the SQL programmer. They are used extensively across businesses to calculate important metrics, such as the average cost of making a film.

You know five different aggregate functions:

-AVG()

-SUM()

-MIN()

-MAX()

-COUNT()

Test your knowledge of what data types they are compatible with.

##### Instructions

* Classify the function based on what data type it is compatible with.



Congratulations! You've got aggregate functions under control.

## Exercise

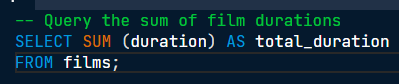
# Practice with aggregate functions

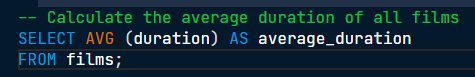
Now let's try extracting summary information from a table using these new aggregate functions. Summarizing is helpful in real life when extracting top-line details from your dataset. Perhaps you'd like to know how old the oldest film in the films table is, what the most expensive film is, or how many films you have listed.

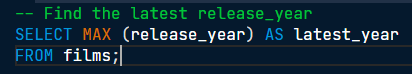
Now it's your turn to get more insights about the films table!

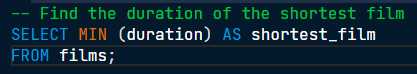
## Instructions

* + Use the SUM() function to calculate the total duration of all films and alias with total\_duration.
  + Calculate the average duration of all films and alias with average\_duration.
  + Find the most recent release\_year in the films table, aliasing as latest\_year.
  + Find the duration of the shortest film and use the alias shortest\_film.









Well done! You'll find yourself using aggregate functions over and over again to get a quick grasp of the data in a SQL database.