## 1. Filtering grouped data

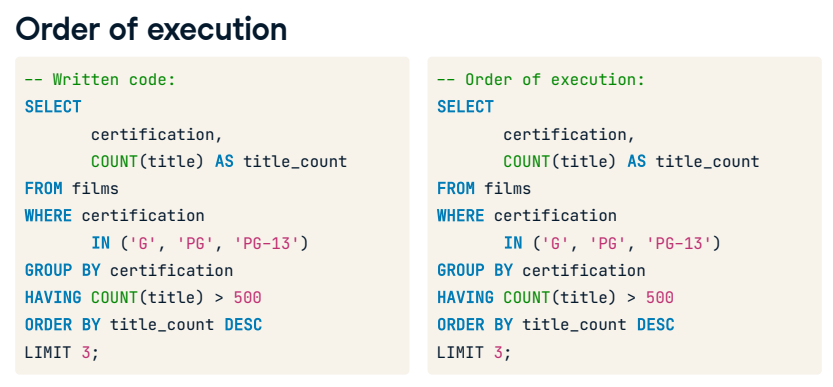
That was excellent work. We've combined sorting and grouping; next, we will combine filtering with grouping.

## 2. HAVING

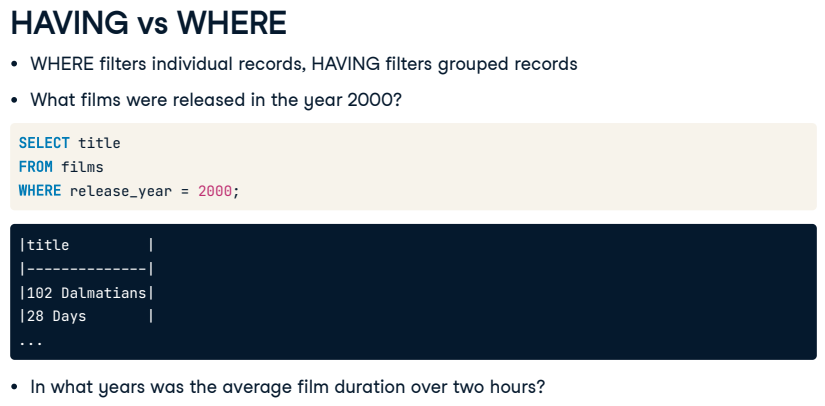


In SQL, we can't filter aggregate functions with WHERE clauses. For example, this query attempting to filter the title count is invalid. That means that if we want to filter based on the result of an aggregate function, we need another way. Groups have their own special filtering word: HAVING. For example, this query shows only those years in which more than ten films were released.

## 3. Order of execution

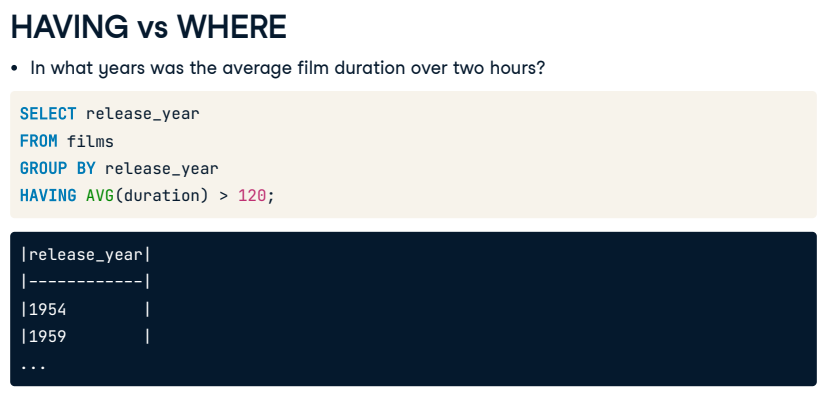
The reason why groups have their own keyword for filtering comes down to the order of execution. We've written a query using many of the keywords we have covered here. This is their written order, starting with SELECT, FROM films, WHERE the certification is G, PG, or PG-13, GROUP BY certification, HAVING the title count be greater than 500, ORDER BY title count, and LIMIT to three. In contrast, the order of execution is: FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY, and LIMIT. By reviewing this order, we can see WHERE is executed before GROUP BY and before any aggregation occurs. This order is also why we cannot use the alias with HAVING, but we can with ORDER BY.

## 4. HAVING vs WHERE



WHERE filters individual records while HAVING filters grouped records. We'll walk through two business questions here to show how to translate them into the correct filter. The first question is "What films were released in the year 2000?". This question does not indicate any sort of grouping. It asks to see only the titles from a specific year and can therefore be written as SELECT title, FROM films, WHERE release year equals 2000. The second question is, "In what years was the average film duration over two hours?". Straight away, we can see this question has a few more layers. Let's break down the question and query into smaller, easier-to-understand steps.

## 5. HAVING vs WHERE



This question requires us to return information about years, so we select the release year from the films table. Next, it asks for the average film duration, which tells us we need to place AVG(duration) somewhere. Since we do not need to provide any additional information around the duration on its own, it is unlikely we need to perform the aggregation within the SELECT clause, so we'll try the HAVING clause instead. The last part of the question indicates we need to filter on the duration. Since we can't filter aggregates with WHERE, this supports our theory about using HAVING! Finally, we need to add a GROUP BY into our query since we have selected a column that has not been aggregated. Recall the aggregate function will convert the duration values into one average value. Going back to the start of our question, we're interested in knowing the average duration per year, so we group it by release year. And there we have it!

## 6. Let's practice!

Let's improve our confidence with a final round of practice.

## Exercise

# Filter with HAVING

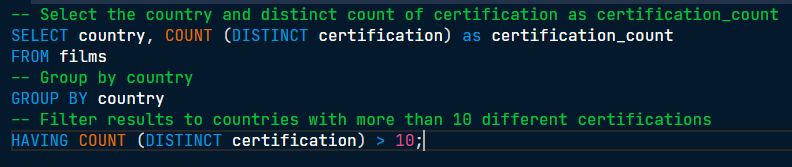
Your final keyword is HAVING. It works similarly to WHERE in that it is a filtering clause, with the difference that HAVING filters grouped data.

Filtering grouped data can be especially handy when working with a large dataset. When working with thousands or even millions of rows, HAVING will allow you to filter for just the group of data you want, such as films over two hours in length!

Practice using HAVING to find out which countries (or country) have the most varied film certifications.

## Instructions

* Select country from the films table, and get the distinct count of certification aliased as certification\_count.
* Group the results by country.
* Filter the unique count of certifications to only results greater than 10.



Great job! The answer is USA with 12 different certifications.

## Exercise

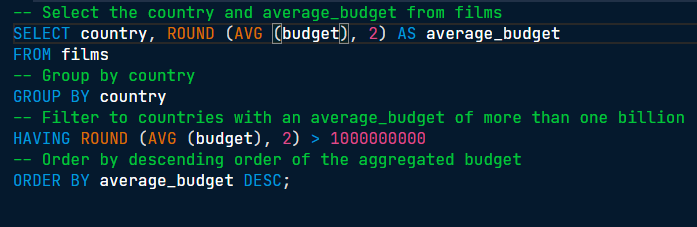
# HAVING and sorting

Filtering and sorting go hand in hand and gives you greater interpretability by ordering our results.

Let's see this magic at work by writing a query showing what countries have the highest average film budgets.

## Instructions

* Select the country and the average budget as average\_budget, rounded to two decimal, from films.
* Group the results by country.
* Filter the results to countries with an average budget of more than one billion (1000000000).
* Sort by descending order of the average\_budget.



You did it! South Korea and Hungary seem to have pricey films... or do they? Actually, these budgets are pretty standard for their local currency.

## Exercise

# All together now

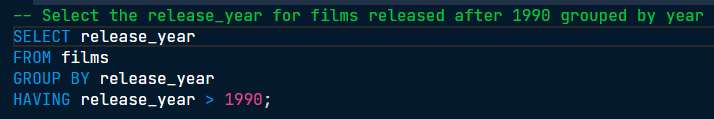
It's time to use much of what you've learned in one query! This is good preparation for using SQL in the real world where you'll often be asked to write more complex queries since some of the basic queries can be answered by playing around in spreadsheet applications.

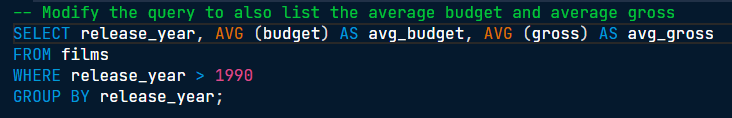
In this exercise, you'll write a query that returns the average budget and gross earnings for films each year after 1990 if the average budget is greater than 60 million.

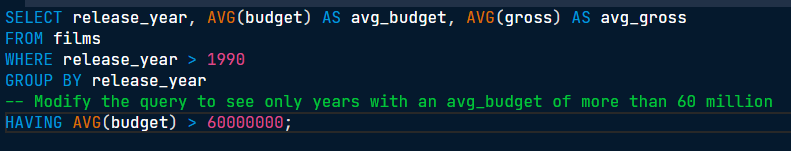
This will be a big query, but you can handle it!

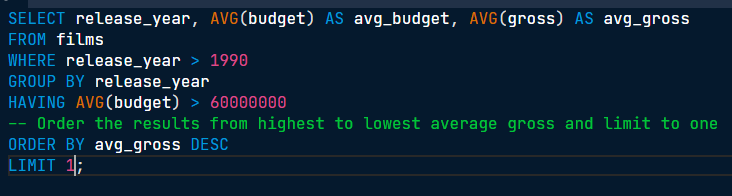
## Instructions

* + Select the release\_year for each film in the films table, filter for records released after 1990, and group by release\_year.
  + Modify the query to include the average budget aliased as avg\_budget and average gross aliased as avg\_gross for the results we have so far.
  + Modify the query once more so that only years with an average budget of greater than 60 million are included.
  + Finally, order the results from the highest average gross and limit to one.









Superb work! SQL queries can get rather long, but breaking them down into individual clauses makes them easier to write.