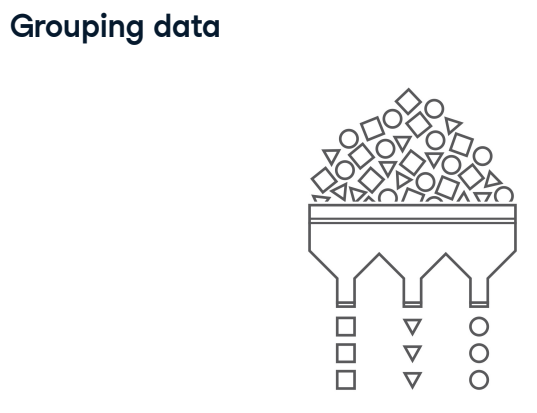
## 1. Grouping data

We've learned how to sort data. Next, we'll look at grouping our results.

## 2. Grouping data



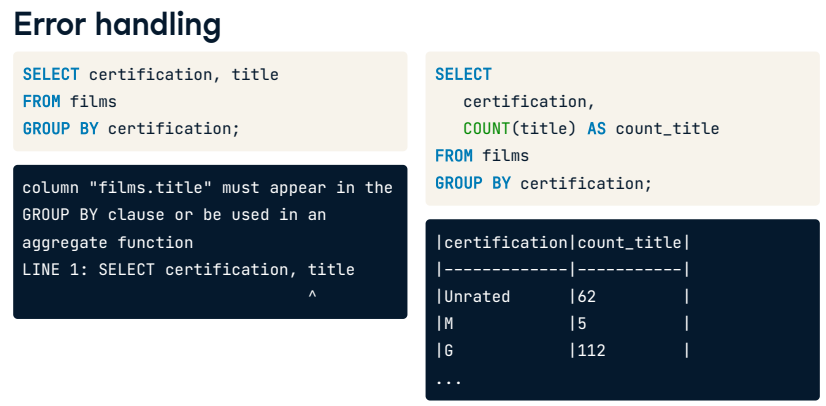
In the real world, we'll often need to summarize data for a particular group of results. For example, we might want to see the film data grouped by certification and make calculations on those groups, such as the average duration for each certification.

## 3. GROUP BY single fields

****

SQL allows us to group with the GROUP BY clause. Here it is used in a query where we have grouped by certification. GROUP BY is commonly used with aggregate functions to provide summary statistics, particularly when only grouping a single field, certification, and selecting multiple fields, certification and title. This is because the aggregate function will reduce the non-grouped field to one record only, which will need to correspond to one group.

## 4. Error handling



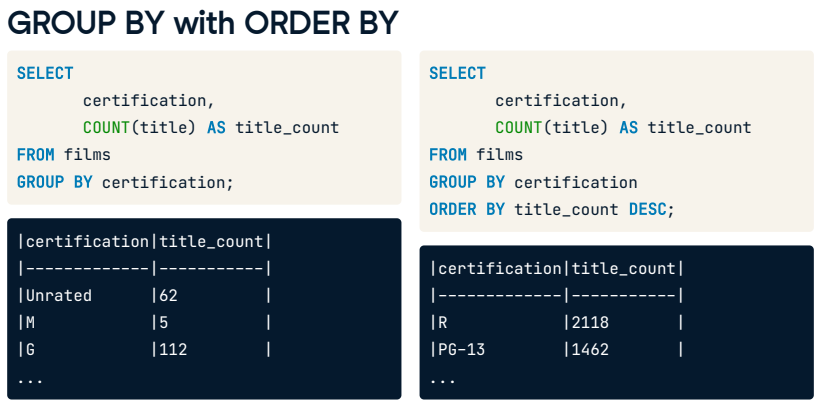
SQL will return an error if we try to SELECT a field that is not in our GROUP BY clause. We'll need to correct this by adding an aggregate function around title.

## 5. GROUP BY multiple fields



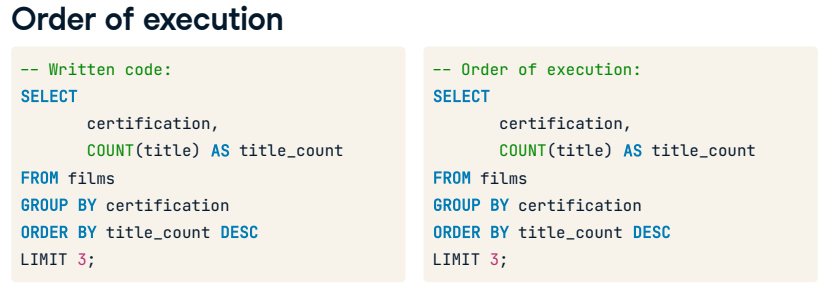
We can use GROUP BY on multiple fields similar to ORDER BY. The order in which we write the fields affects how the data is grouped. The query here selects and groups certification and language while aggregating the title. The result shows that we have five films that have missing values for both certification and language, two films that are unrated and in Japanese, two films that are rated R and in Norwegian, and so on.

## 6. GROUP BY with ORDER BY



We can combine GROUP BY with ORDER BY to group our results, make a calculation, and then order our results. For example, we can clean up one of our previous queries by sorting the results by the title count in descending order. Here is that query without ORDER BY, and this is the same query with ordering added. ORDER BY is always written after GROUP BY, and notice that we can refer back to the alias within the query. That is because of the order of execution. It looks like movies rated R are most common in our database.

## 7. Order of execution



GROUP BY fits into our order after FROM and before all other clauses. Our updated queries will begin with FROM, followed by grouping, selecting the data and creating the alias, sorting the results, and limiting them to the desired number.

## 8. Let's practice!

In the following exercises, we'll examine our film database to find out about release year, review, and budget patterns. Let's practice!

## Exercise

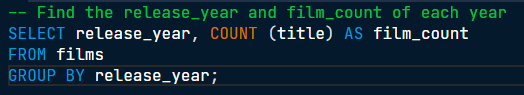
# GROUP BY single fields

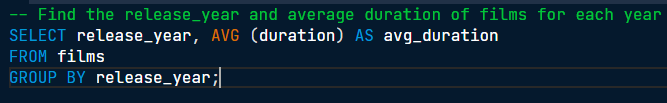
GROUP BY is a SQL keyword that allows you to group and summarize results with the additional use of aggregate functions. For example, films can be grouped by the certification and language before counting the film titles in each group. This allows you to see how many films had a particular certification and language grouping.

In the following steps, you'll summarize other groups of films to learn more about the films in your database.

## Instructions

* + Select the release\_year and count of films released in each year aliased as film\_count.
  + Select the release\_year and average duration aliased as avg\_duration of all films, grouped by release\_year.





Great job grouping! Using GROUP BY with a time or date field such as release\_year can help us identify trends such as a period of time where movies were really short!

## Exercise

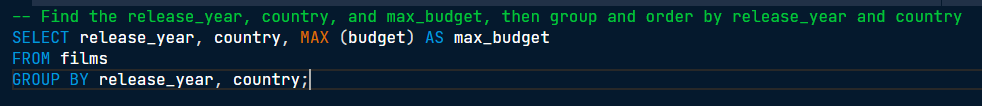
# GROUP BY multiple fields

GROUP BY becomes more powerful when used across multiple fields or combined with ORDER BY and LIMIT.

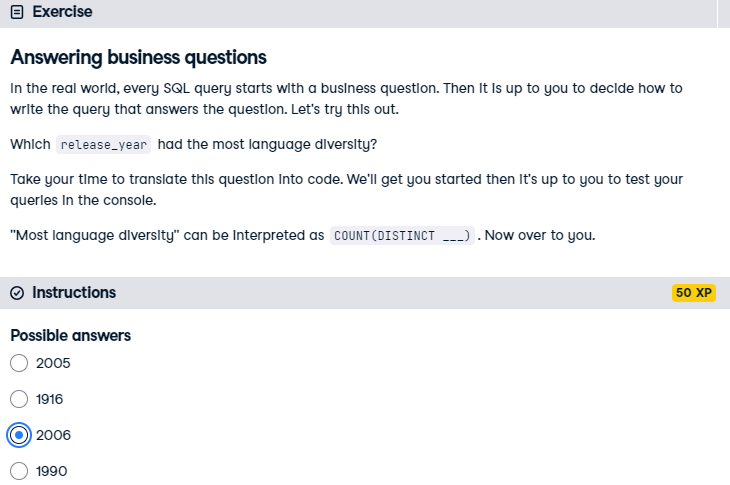
Perhaps you're interested in learning about budget changes throughout the years in individual countries. You'll use grouping in this exercise to look at the maximum budget for each country in each year there is data available.

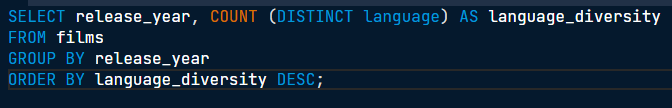
## Instructions

* Select the release\_year, country, and the maximum budget aliased as max\_budget for each year and each country; sort your results by release\_year and country.



Well done! You can see how building on your SQL queries helps you gain more insights and detect trends in the data, such as how film budgets may change throughout the years.





Well done! The year 2006 had 16 distinct languages, that's more than any other year.