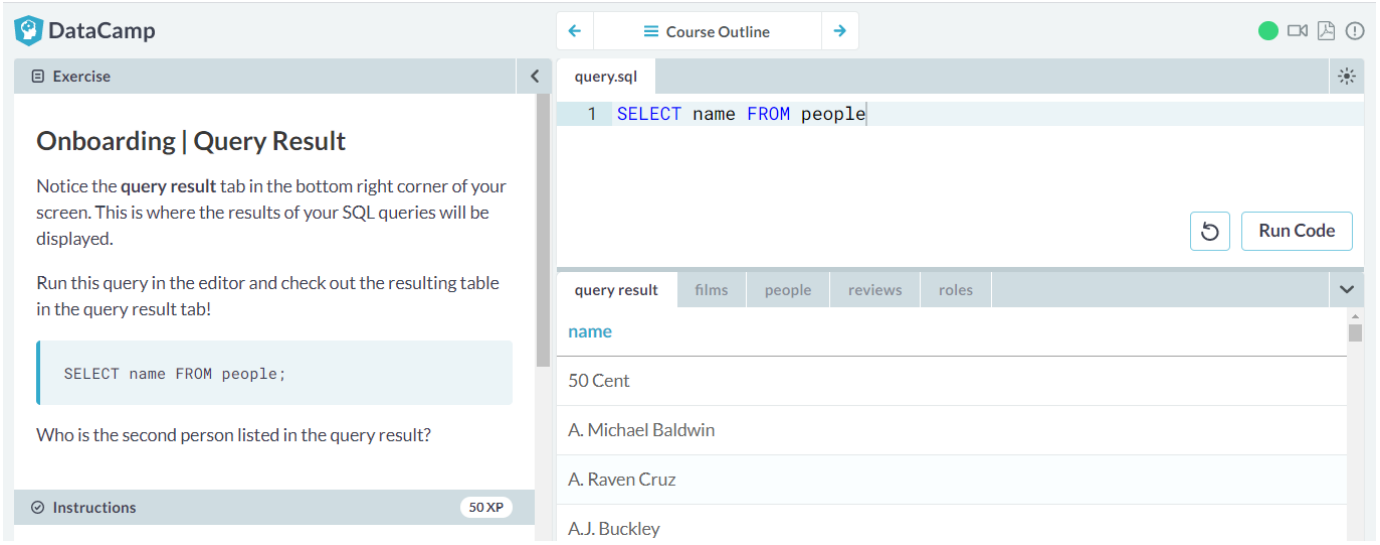
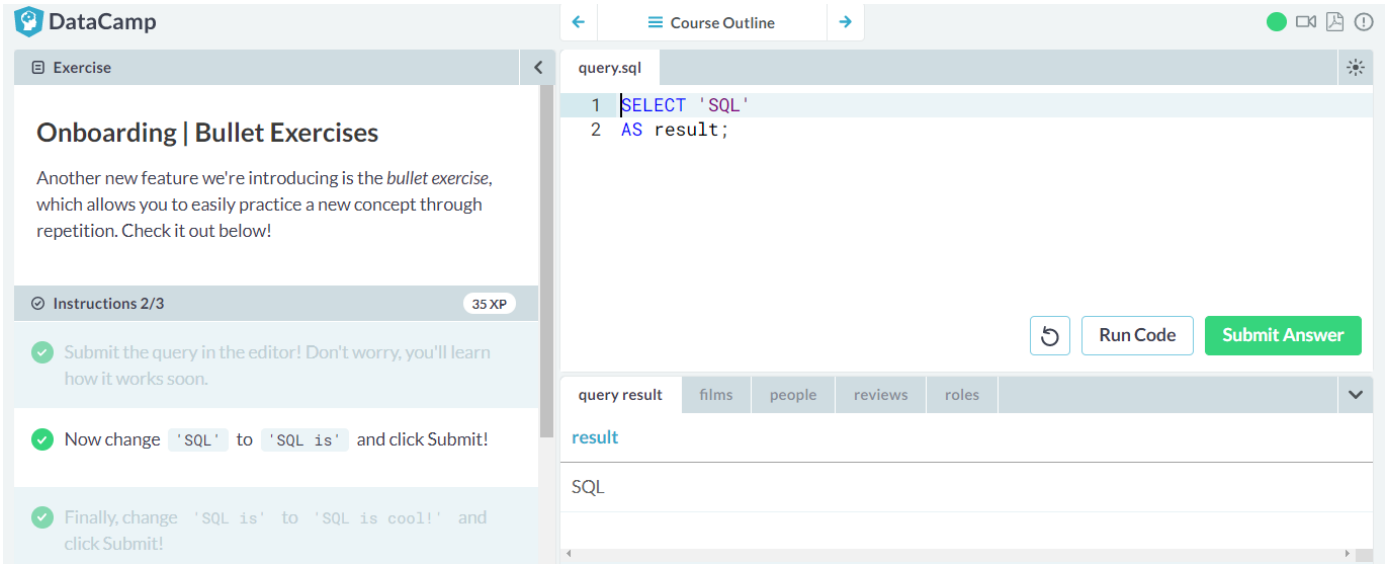
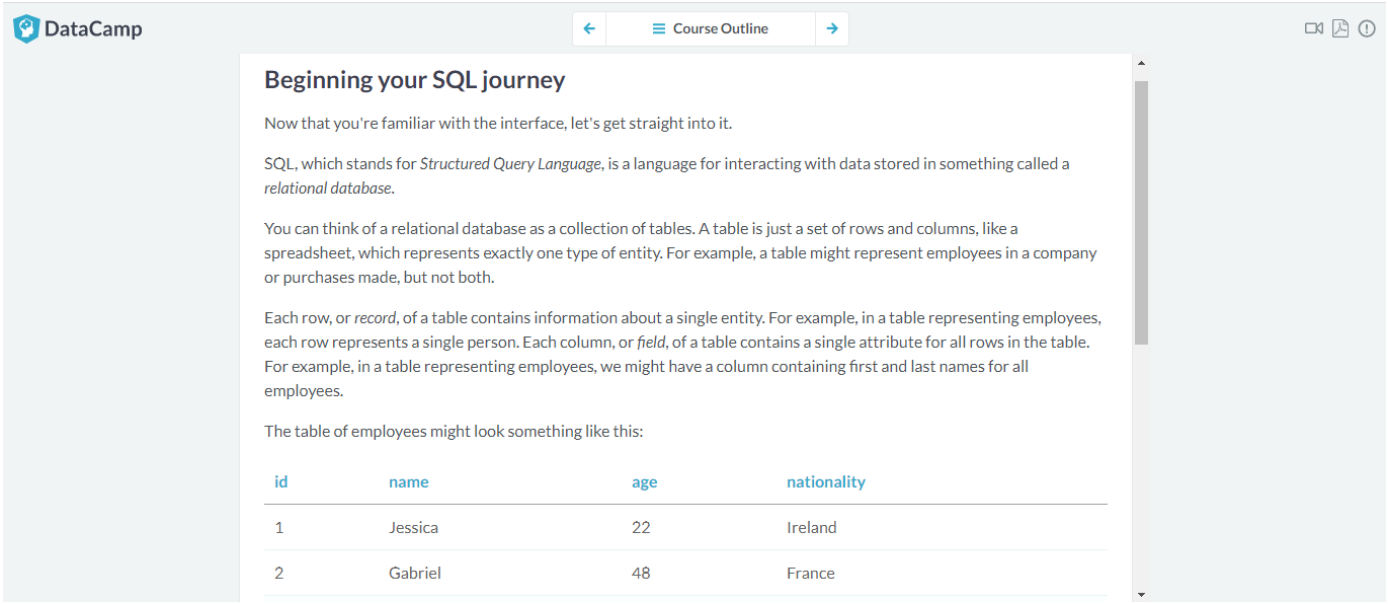
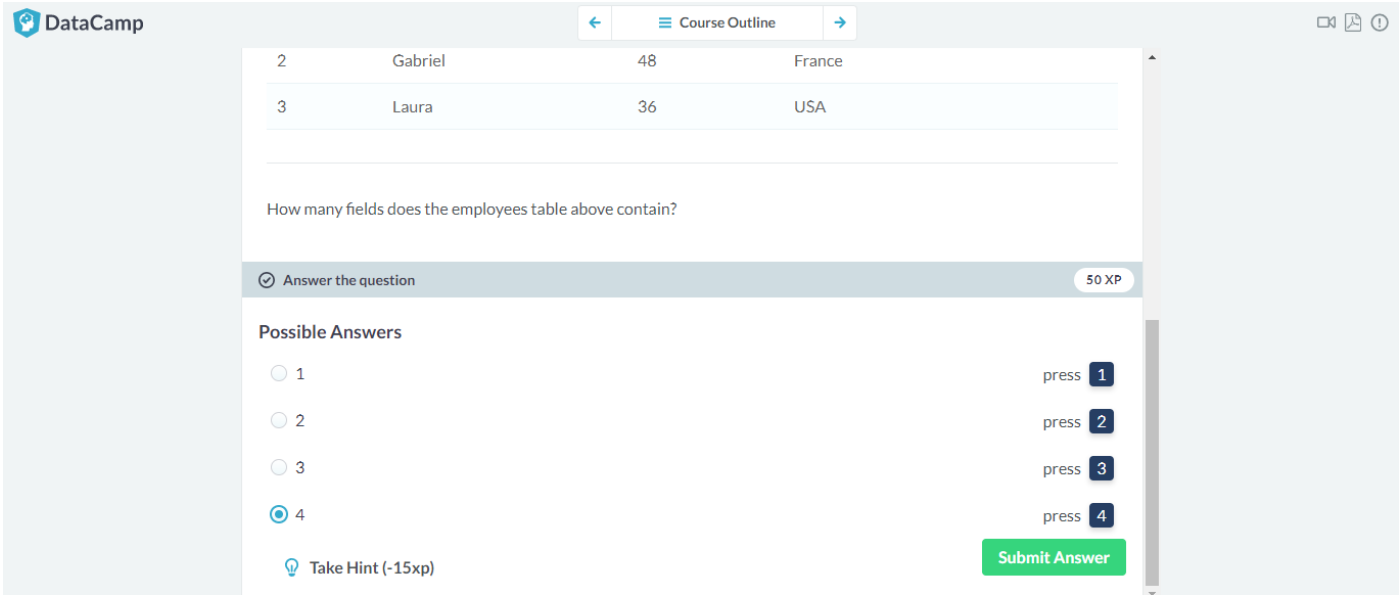
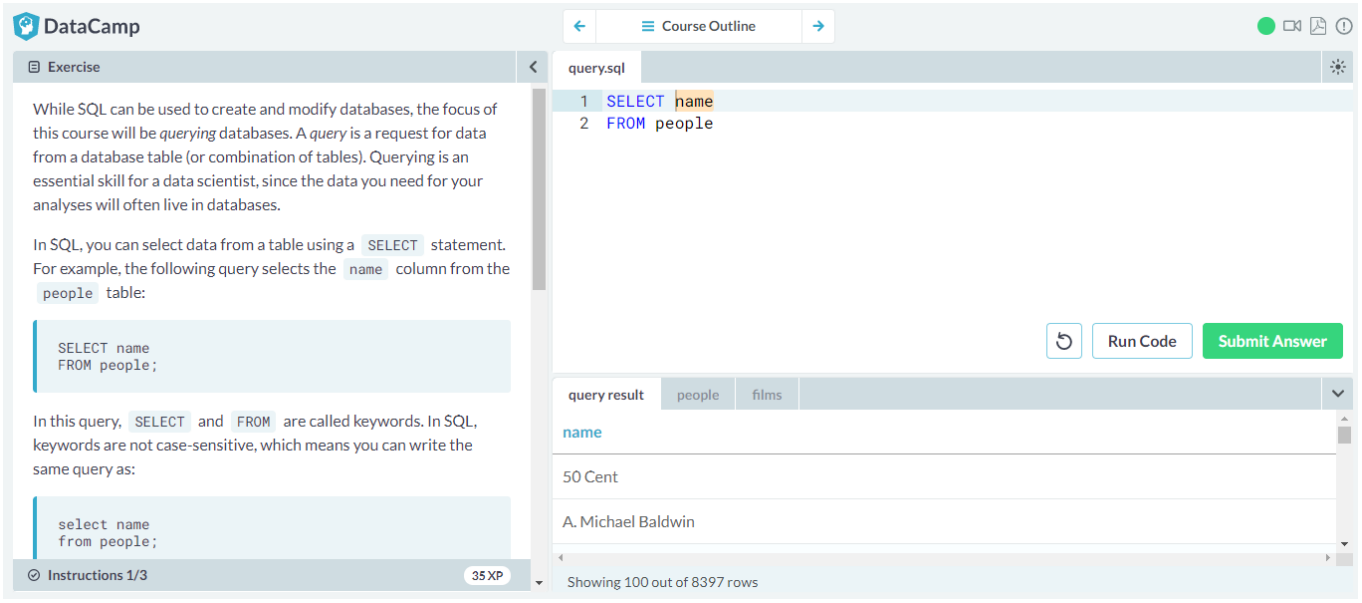
INTRODUCTION TO SQL

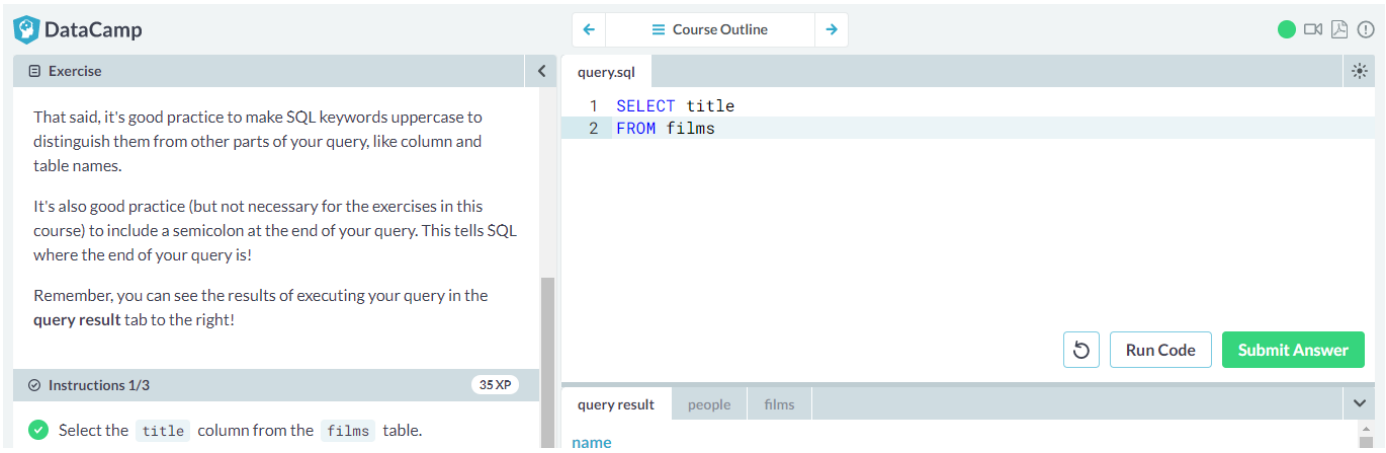


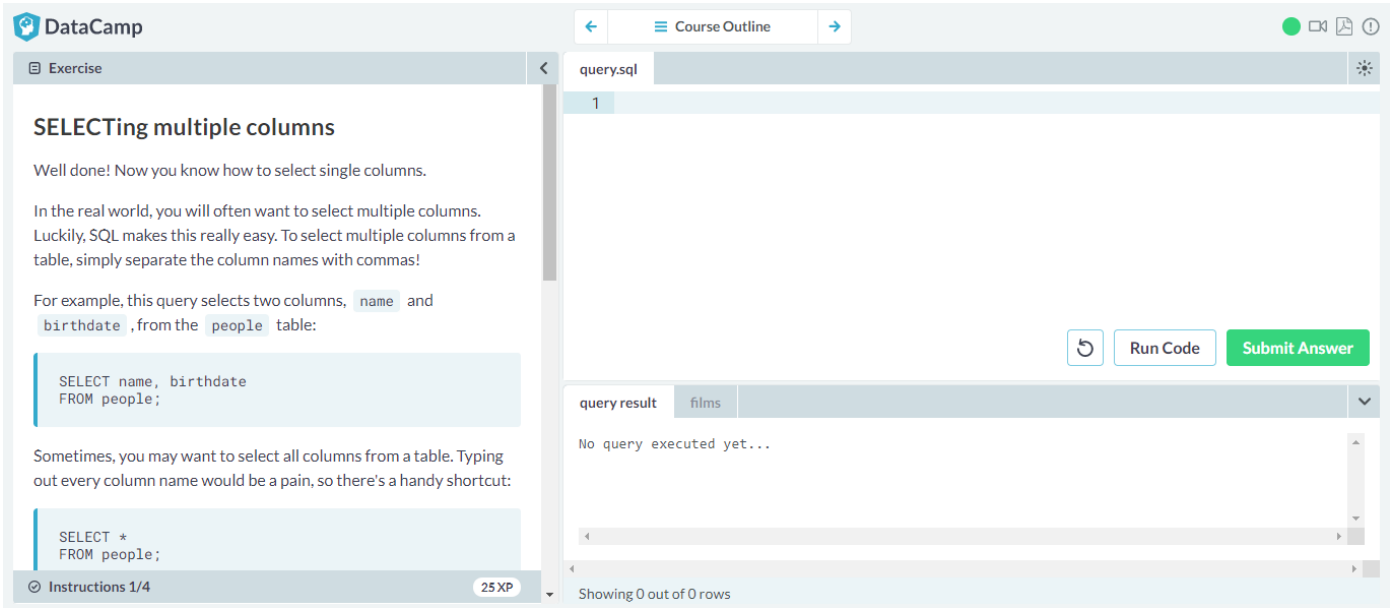


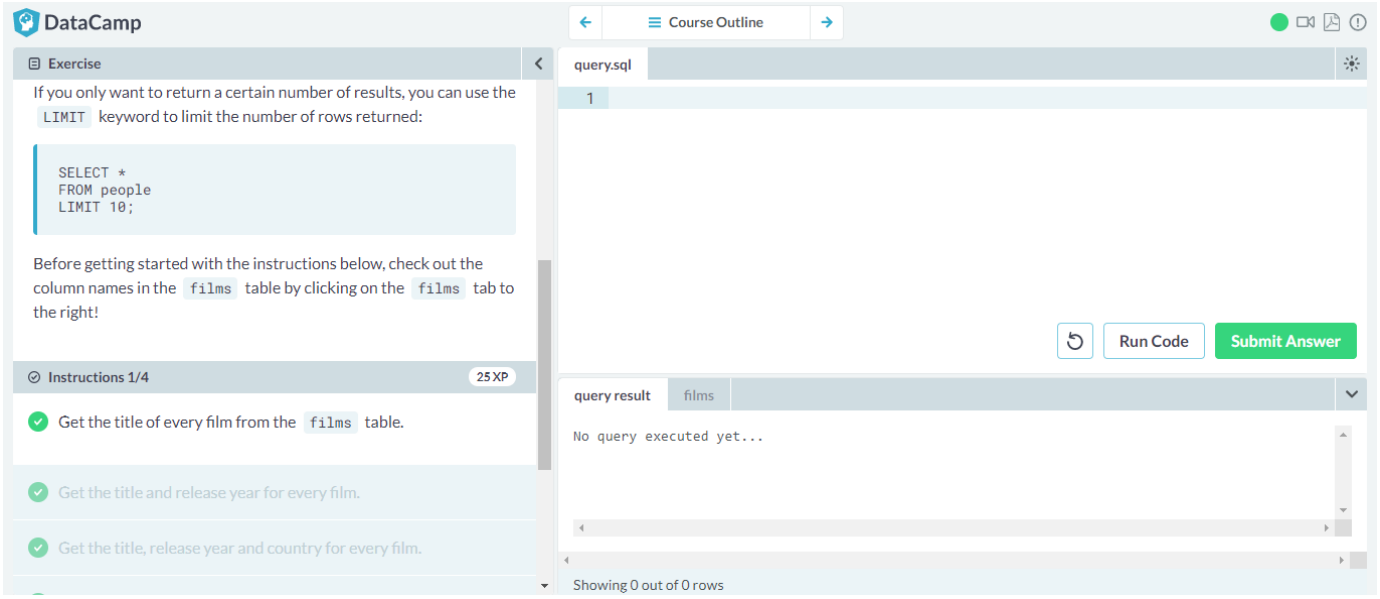

DataCamp
Course Outline
Beginning your SQL journey
Now that you're familiar with the interface, let's get straight into it.
SQL, which stands for *Structured Query Language*, is a language for interacting with data stored in something called a *relational database*.
You can think of a relational database as a collection of tables. A table is just a set of rows and columns, like a spreadsheet, which represents exactly one type of entity. For example, a table might represent employees in a company or purchases made, but not both.
Each row, or *record*, of a table contains information about a single entity. For example, in a table representing employees, each row represents a single person. Each column, or *field*, of a table contains a single attribute for all rows in the table. For example, in a table representing employees, we might have a column containing first and last names for all employees.
The table of employees might look something like this:
id name age nationality
1 Jessica 22 Ireland
2 Gabriel 48 France

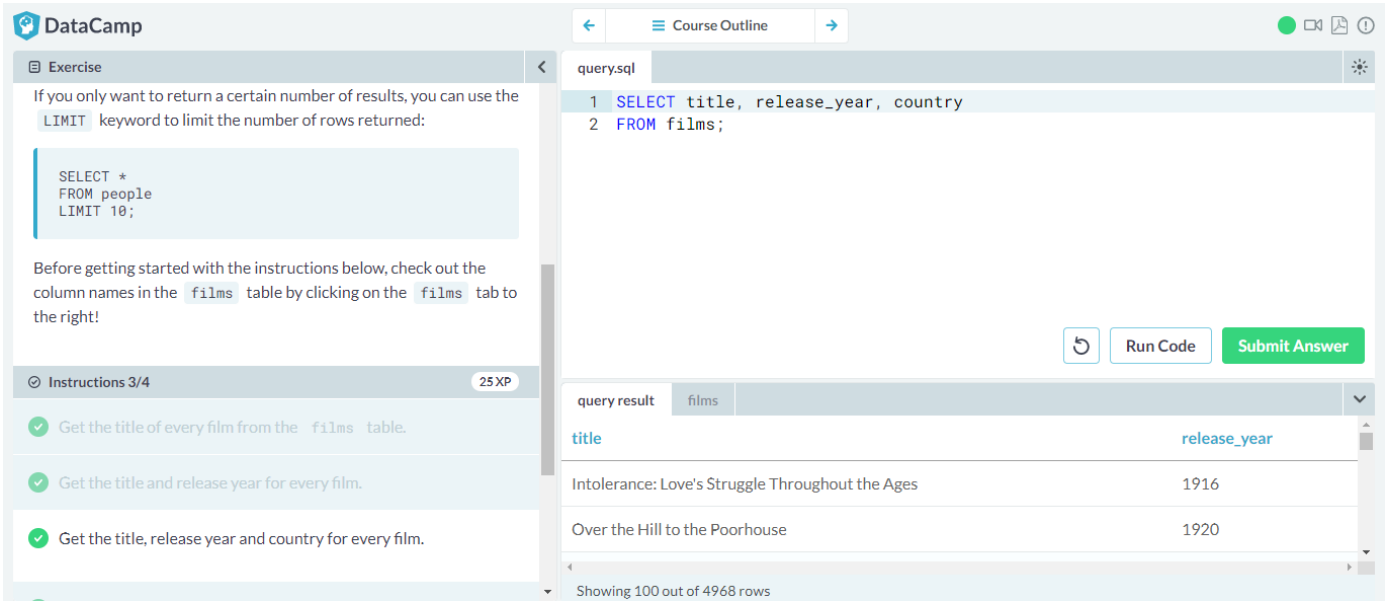

DataCamp
Course Outline
2 Gabriel 48 France
3 Laura 36 USA
How many fields does the employees table above contain?
Answer the question
50 XP
Possible Answers
1 press 1
2 press 2
3 press 3
4 press 4
Take Hint (-15xp)
Submit Answer

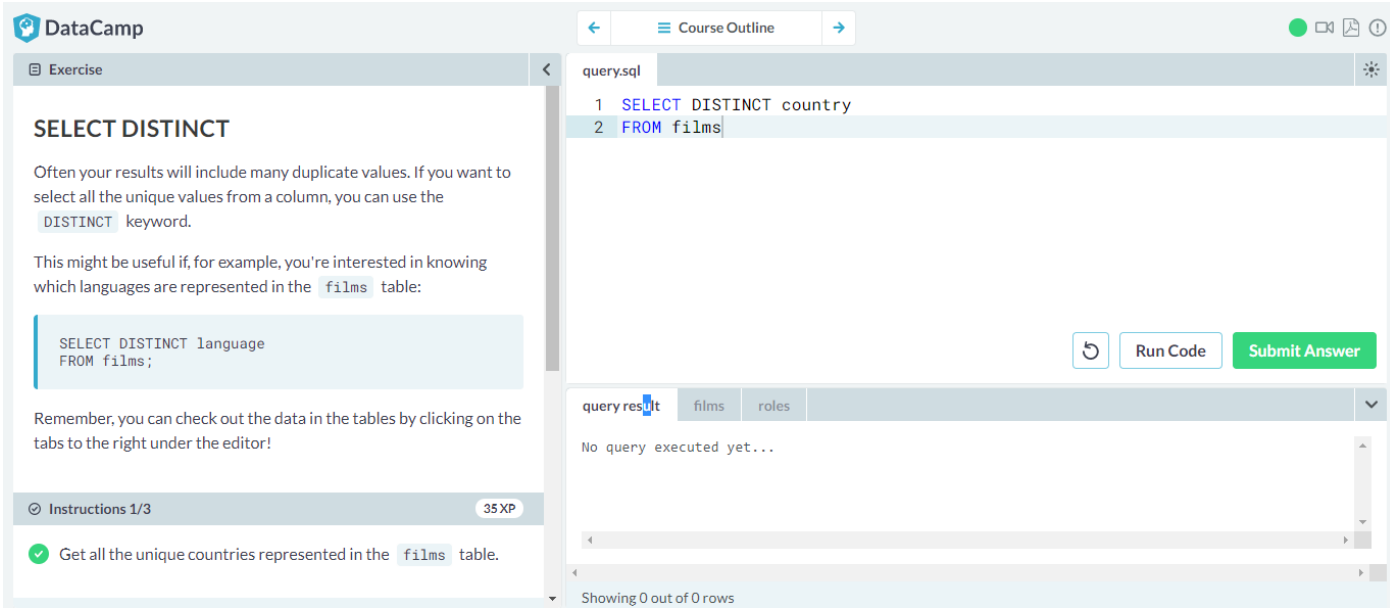

DataCamp
Course Outline
Exercise
While SQL can be used to create and modify databases, the focus of this course will be querying databases. A query is a request for data from a database table (or combination of tables). Querying is an essential skill for a data scientist, since the data you need for your analyses will often live in databases.
In SQL, you can select data from a table using a SELECT statement. For example, the following query selects the name column from the people table:
SELECT name
FROM people;
In this query, SELECT and FROM are called keywords. In SQL, keywords are not case-sensitive, which means you can write the same query as:
select name
from people;
Instructions 1/3
35 XP
query.sql
1 SELECT name
2 FROM people
Run Code
Submit Answer
query result
people
films
name
50 Cent
A. Michael Baldwin
Showing 100 out of 8397 rows

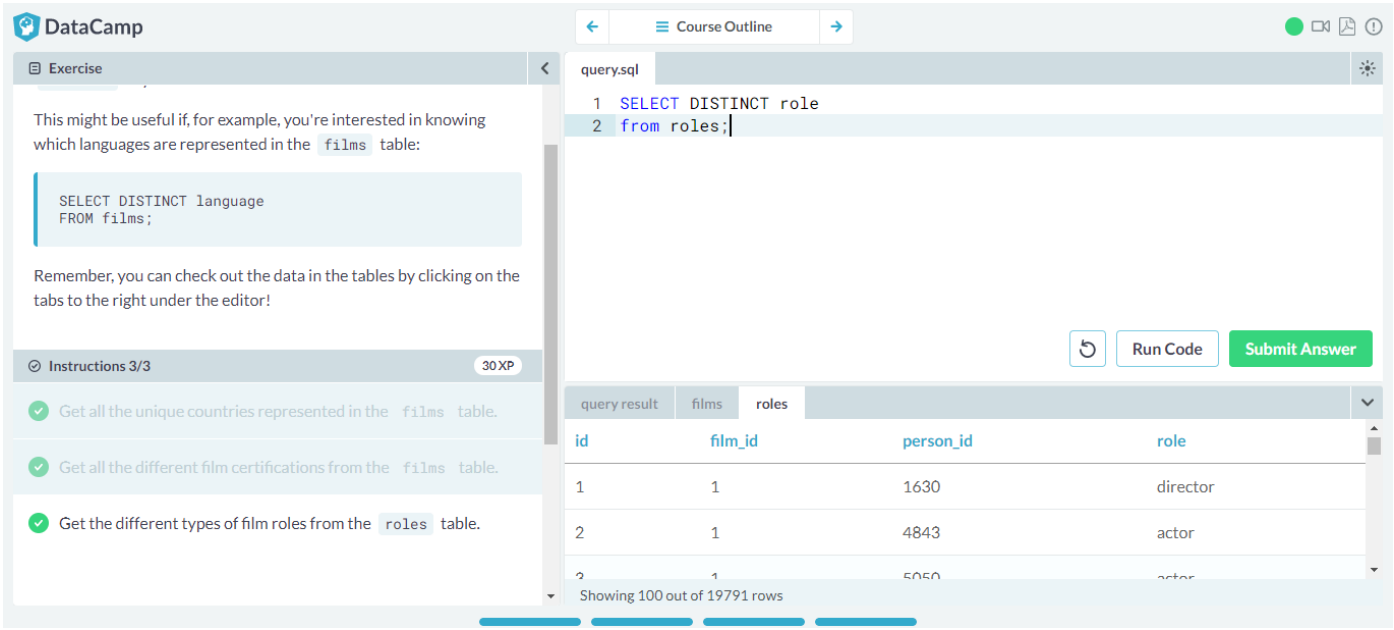

DataCamp
Course Outline
Exercise
That said, it's good practice to make SQL keywords uppercase to distinguish them from other parts of your query, like column and table names.
It's also good practice (but not necessary for the exercises in this course) to include a semicolon at the end of your query. This tells SQL where the end of your query is!
Remember, you can see the results of executing your query in the query result tab to the right!
Instructions 1/3
35 XP
Select the title column from the films table.
query.sql
1 SELECT title
2 FROM films
Run Code
Submit Answer
query result
people
films
name

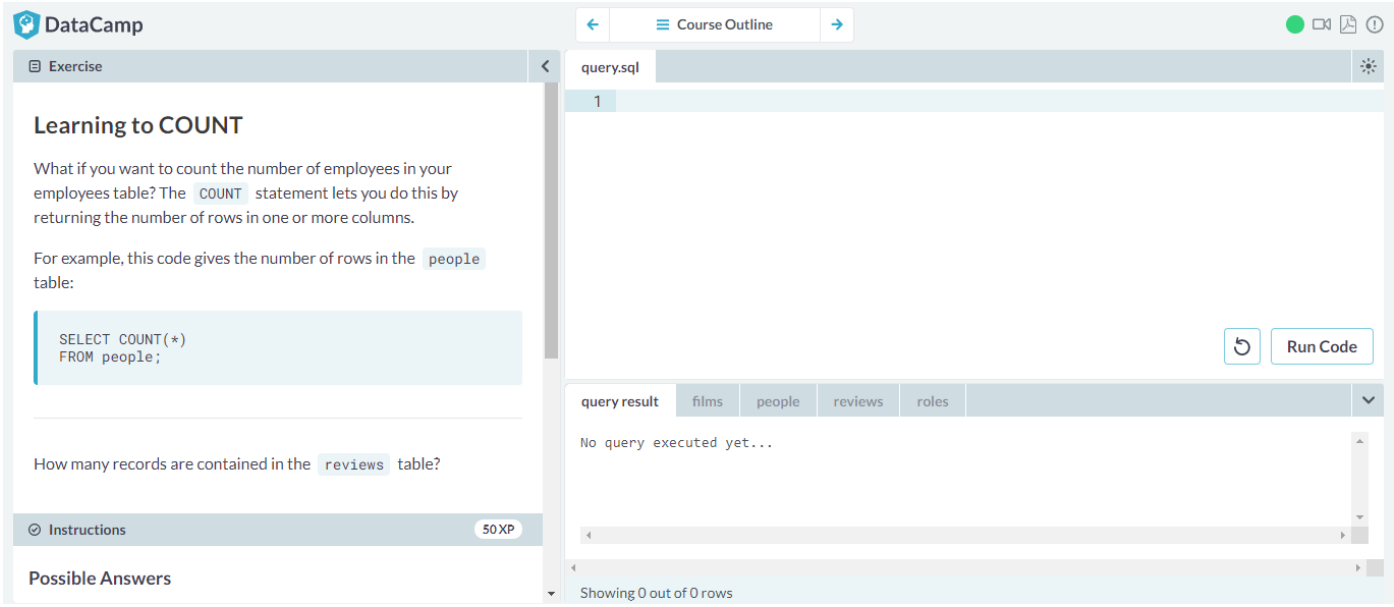

DataCamp
Course Outline
Exercise
SELECTing multiple columns
Well done! Now you know how to select single columns.
In the real world, you will often want to select multiple columns. Luckily, SQL makes this really easy. To select multiple columns from a table, simply separate the column names with commas!
For example, this query selects two columns, name and birthdate, from the people table:
SELECT name, birthdate
FROM people;
Sometimes, you may want to select all columns from a table. Typing out every column name would be a pain, so there's a handy shortcut:
SELECT *
FROM people;
Instructions 1/4
25 XP
query.sql
1
Run Code
Submit Answer
query result
films
No query executed yet...
Showing 0 out of 0 rows

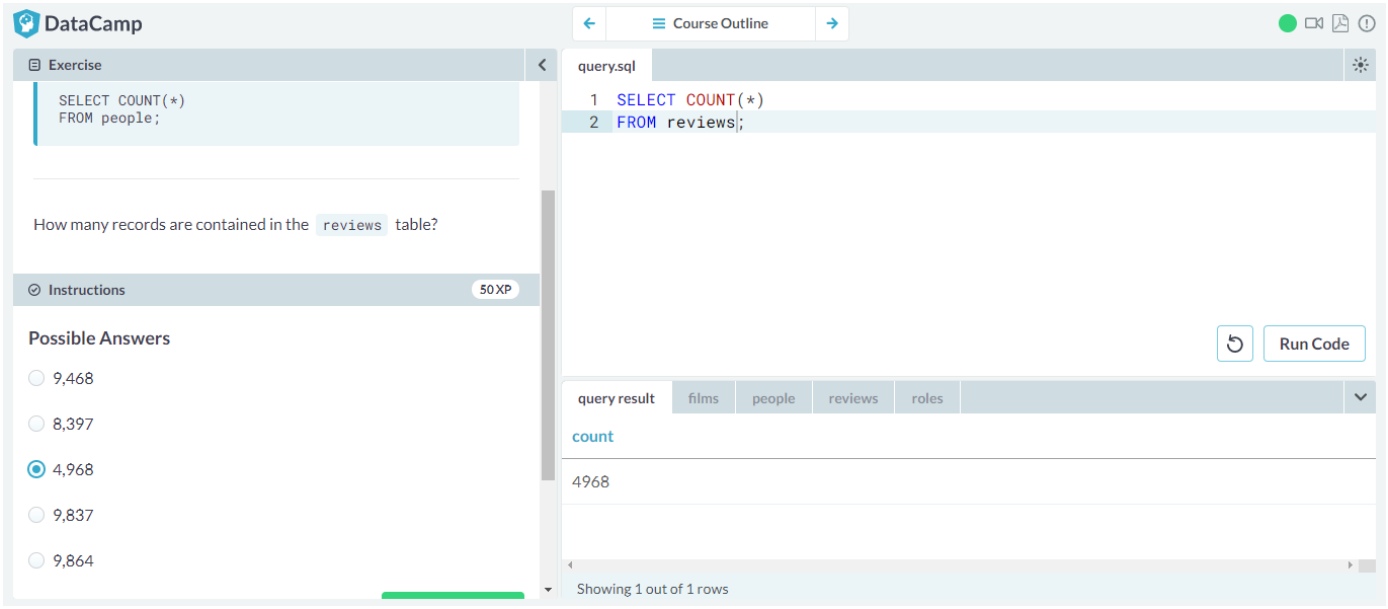

DataCamp
Course Outline
Exercise
If you only want to return a certain number of results, you can use the LIMIT keyword to limit the number of rows returned:
SELECT *
FROM people
LIMIT 10;
Before getting started with the instructions below, check out the column names in the films table by clicking on the films tab to the right!
Instructions 1/4
25 XP
Get the title of every film from the films table.
Get the title and release year for every film.
Get the title, release year and country for every film.
query.sql
1
Run Code
Submit Answer
query result
films
No query executed yet...
Showing 0 out of 0 rows

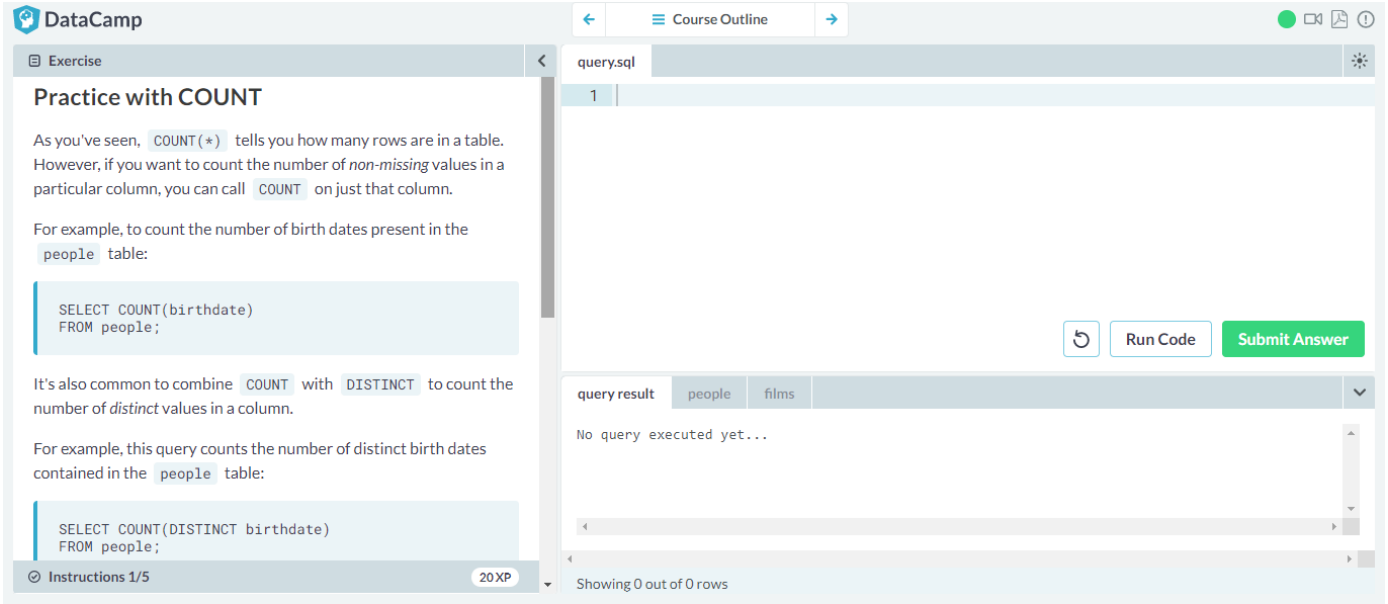

DataCamp
Course Outline
Exercise
If you only want to return a certain number of results, you can use the LIMIT keyword to limit the number of rows returned:
SELECT *
FROM people
LIMIT 10;
Before getting started with the instructions below, check out the column names in the films table by clicking on the films tab to the right!
Instructions 3/4
25 XP
Get the title of every film from the films table.
Get the title and release year for every film.
Get the title, release year and country for every film.
query.sql
1 SELECT title, release_year, country
2 FROM films;
Run Code
Submit Answer
query result
films
title
release_year
Intolerance: Love's Struggle Throughout the Ages
1916
Over the Hill to the Poorhouse
1920
Showing 100 out of 4968 rows

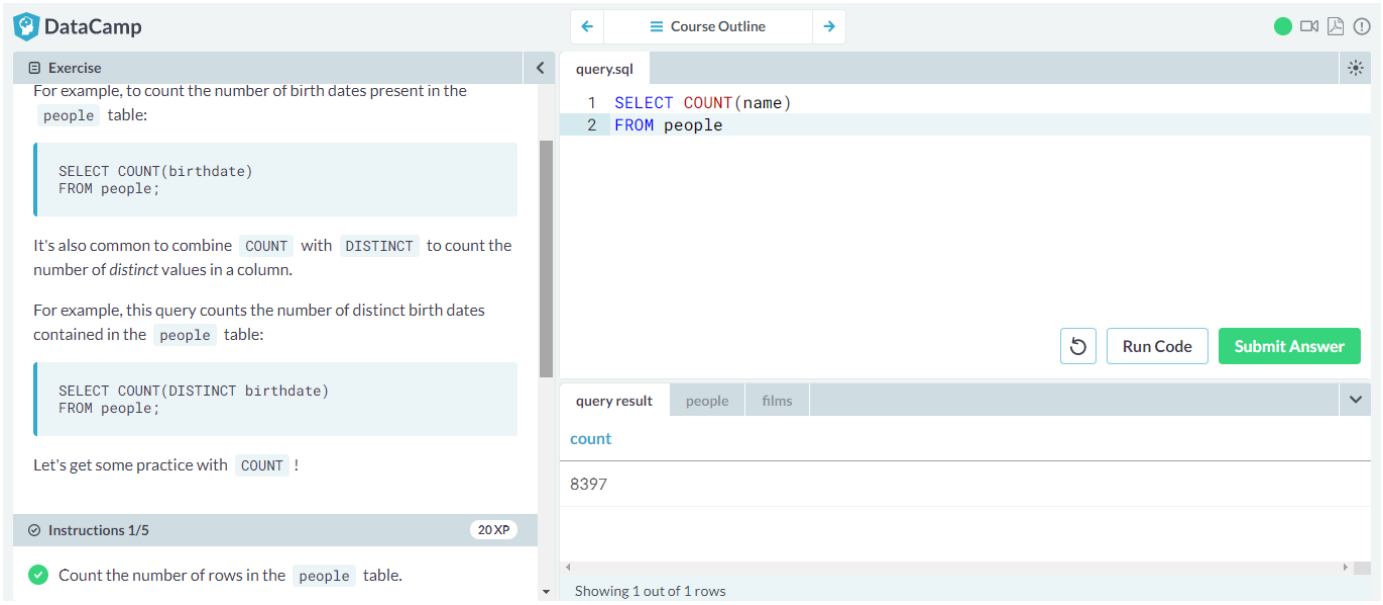

DataCamp
Course Outline
Exercise
SELECT DISTINCT
Often your results will include many duplicate values. If you want to select all the unique values from a column, you can use the DISTINCT keyword.
This might be useful if, for example, you're interested in knowing which languages are represented in the films table:
SELECT DISTINCT language
FROM films;
Remember, you can check out the data in the tables by clicking on the tabs to the right under the editor!
Instructions 1/3
35 XP
Get all the unique countries represented in the films table.
query.sql
1 SELECT DISTINCT country
2 FROM films
Run Code
Submit Answer
query result
films
roles
No query executed yet...
Showing 0 out of 0 rows

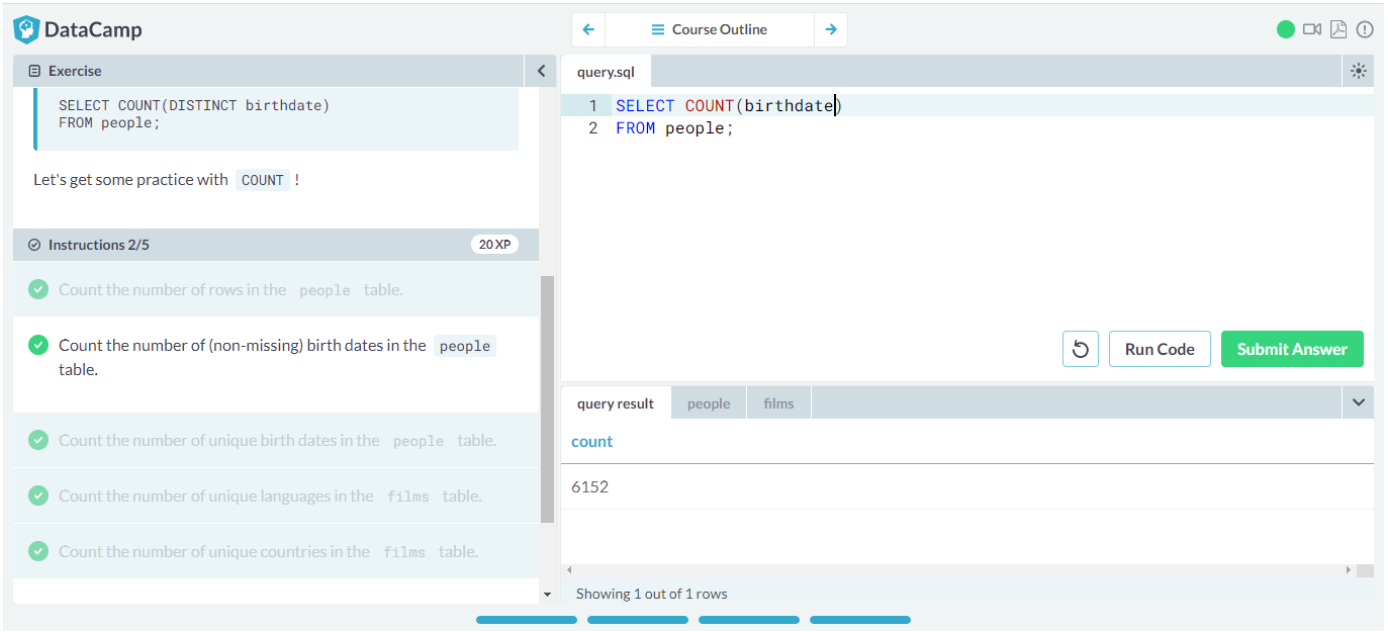

DataCamp
Course Outline
Exercise
This might be useful if, for example, you're interested in knowing which languages are represented in the films table:
SELECT DISTINCT language
FROM films;
Remember, you can check out the data in the tables by clicking on the tabs to the right under the editor!
Instructions 3/3
30 XP
Get all the unique countries represented in the films table.
Get all the different film certifications from the films table.
Get the different types of film roles from the roles table.
query.sql
1 SELECT DISTINCT role
2 from roles;
Run Code
Submit Answer
query result
films
roles
id
film_id
person_id
role
1
1
1630
director
2
1
4843
actor
Showing 100 out of 19791 rows

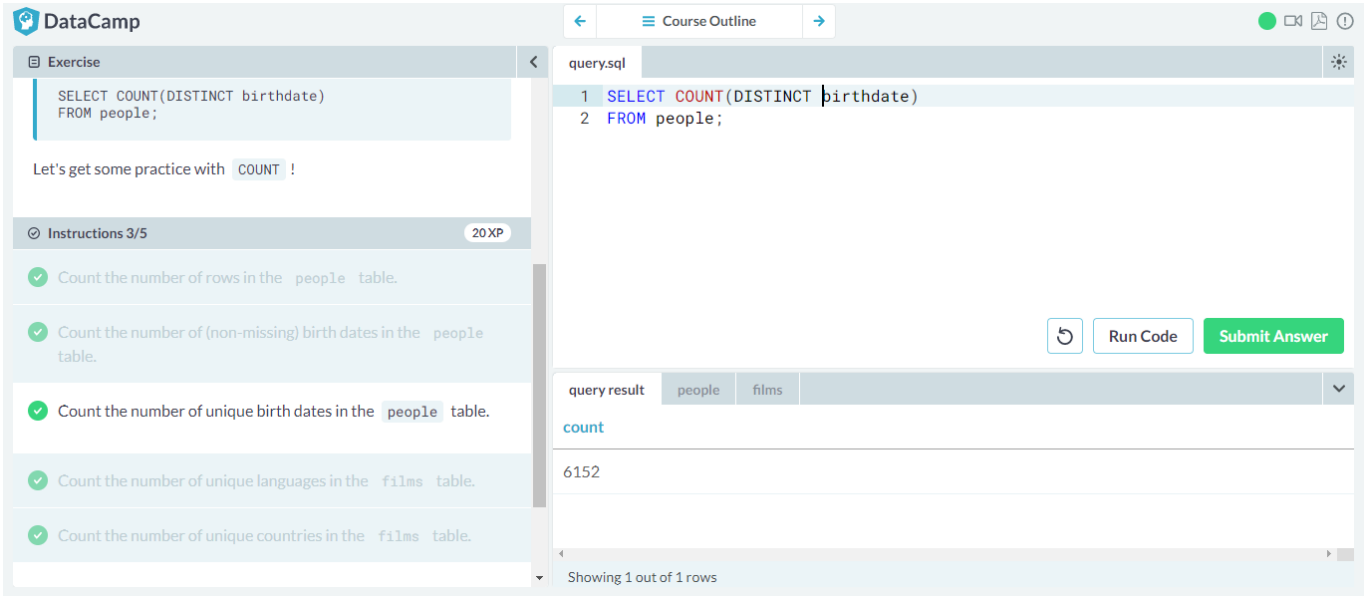

DataCamp
Course Outline
Exercise
Learning to COUNT
What if you want to count the number of employees in your employees table? The COUNT statement lets you do this by returning the number of rows in one or more columns.
For example, this code gives the number of rows in the people table:
SELECT COUNT(*)
FROM people;
How many records are contained in the reviews table?
Instructions
50 XP
Possible Answers
query.sql
1
Run Code
query result
films
people
reviews
roles
No query executed yet...
Showing 0 out of 0 rows

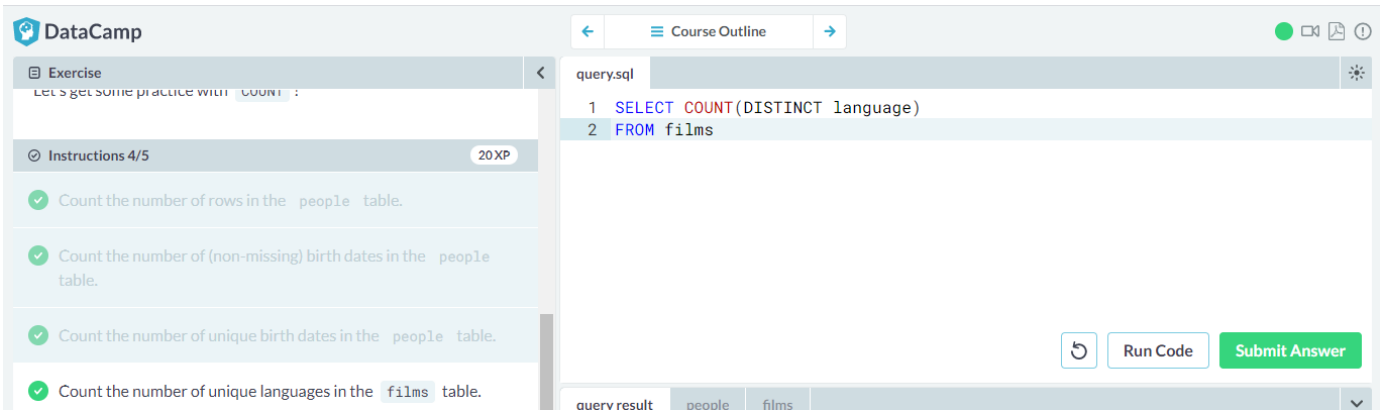

DataCamp
Course Outline
Exercise
SELECT COUNT(*)
FROM people;
How many records are contained in the reviews table?
Instructions
50 XP
Possible Answers
9,468
8,397
4,968
9,837
9,864
query.sql
1 SELECT COUNT(*)
2 FROM reviews;
Run Code
query result
films
people
reviews
roles
count
4968
Showing 1 out of 1 rows

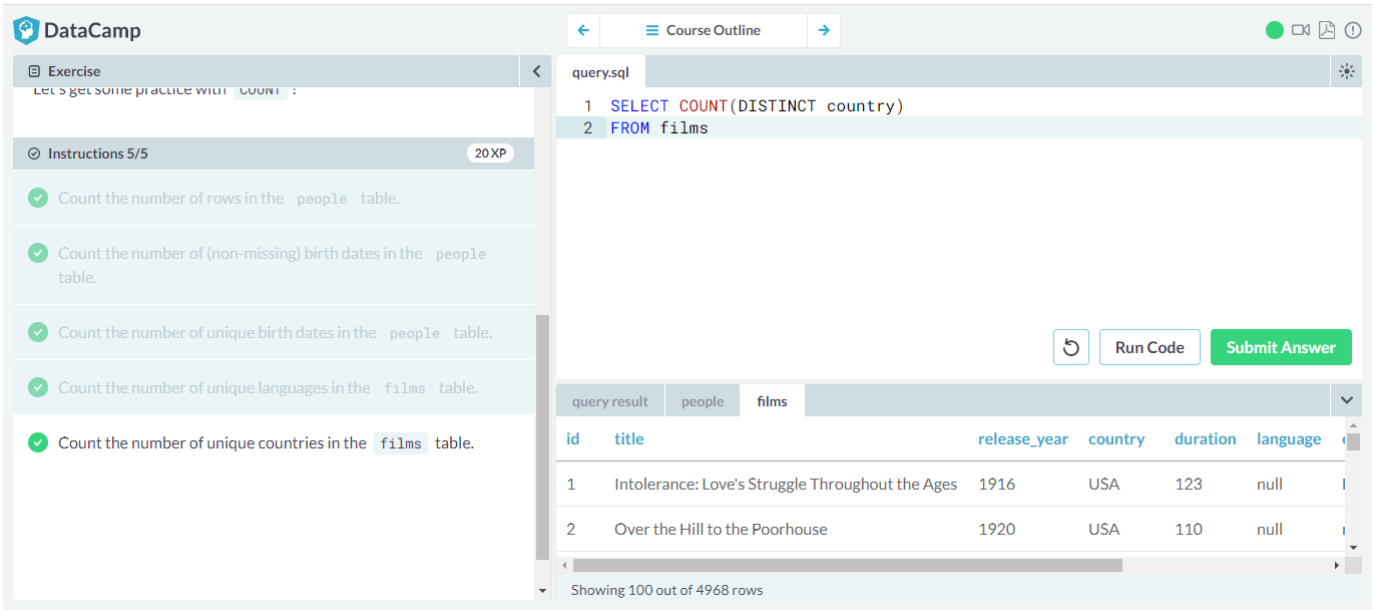

DataCamp
Course Outline
Exercise
Practice with COUNT
As you've seen, COUNT(*) tells you how many rows are in a table. However, if you want to count the number of non-missing values in a particular column, you can call COUNT on just that column.
For example, to count the number of birth dates present in the people table:
SELECT COUNT(birthdate)
FROM people;
It's also common to combine COUNT with DISTINCT to count the number of distinct values in a column.
For example, this query counts the number of distinct birth dates contained in the people table:
SELECT COUNT(DISTINCT birthdate)
FROM people;
Instructions 1/5
20 XP
query.sql
1
Run Code
Submit Answer
query result
people
films
No query executed yet...
Showing 0 out of 0 rows

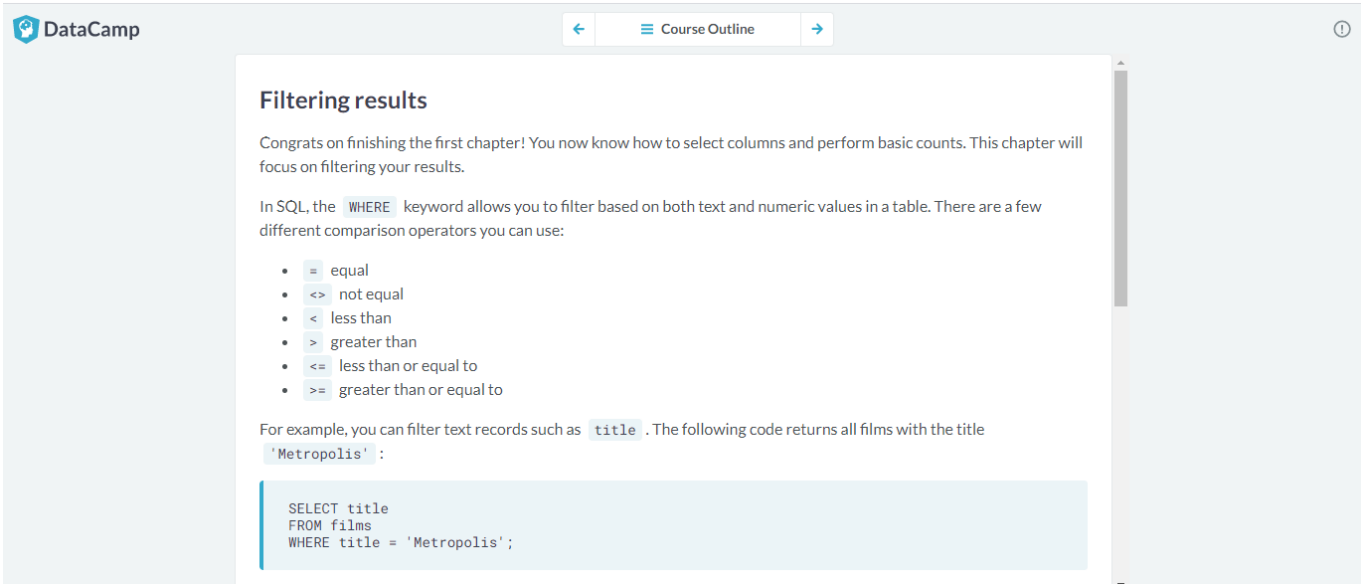

DataCamp
Course Outline
Exercise
For example, to count the number of birth dates present in the people table:
SELECT COUNT(birthdate)
FROM people;
It's also common to combine COUNT with DISTINCT to count the number of distinct values in a column.
For example, this query counts the number of distinct birth dates contained in the people table:
SELECT COUNT(DISTINCT birthdate)
FROM people;
Let's get some practice with COUNT !
Instructions 1/5
20 XP
Count the number of rows in the people table.
query.sql
1 SELECT COUNT(name)
2 FROM people
Run Code
Submit Answer
query result
people
films
count
8397
Showing 1 out of 1 rows


DataCamp
Course Outline
Exercise
SELECT COUNT(DISTINCT birthdate)
FROM people;
Let's get some practice with COUNT !
Instructions 2/5
20 XP
Count the number of rows in the people table.
Count the number of (non-missing) birth dates in the people table.
Count the number of unique birth dates in the people table.
Count the number of unique languages in the films table.
Count the number of unique countries in the films table.
query.sql
1 SELECT COUNT(birthdate)
2 FROM people;
Run Code
Submit Answer
query result
people
films
count
6152
Showing 1 out of 1 rows

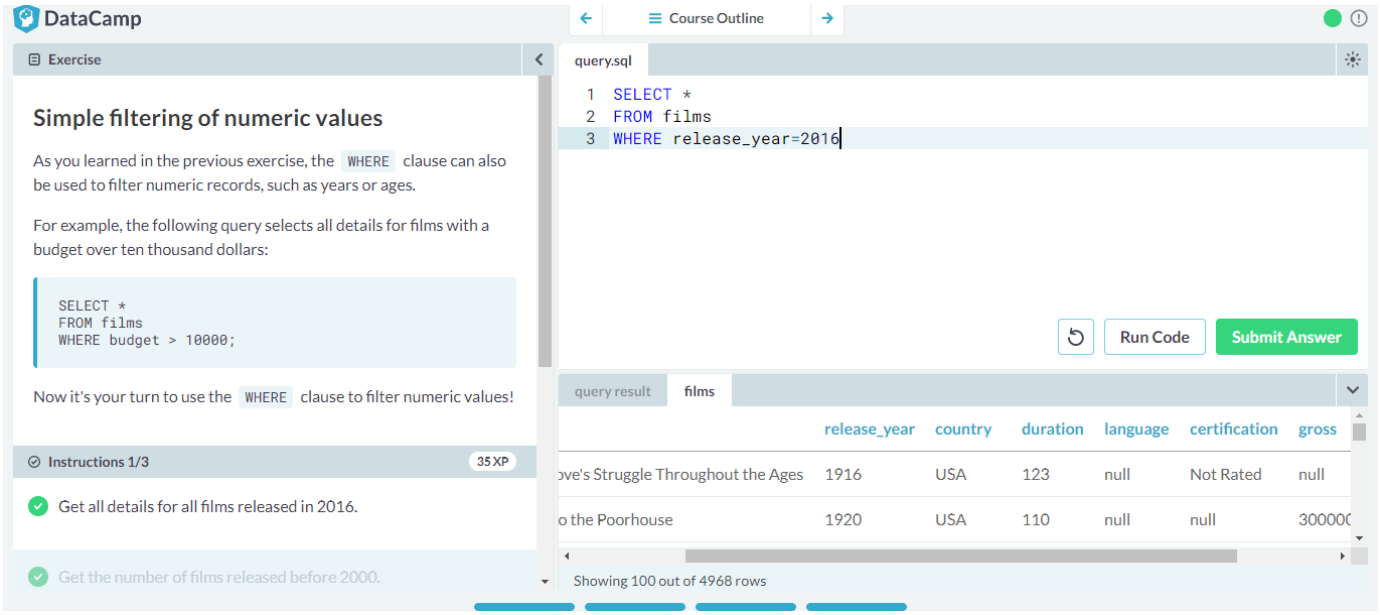

DataCamp
Course Outline
Exercise
SELECT COUNT(DISTINCT birthdate)
FROM people;
Let's get some practice with COUNT !
Instructions 3/5
20 XP
Count the number of rows in the people table.
Count the number of (non-missing) birth dates in the people table.
Count the number of unique birth dates in the people table.
Count the number of unique languages in the films table.
Count the number of unique countries in the films table.
query.sql
1 SELECT COUNT(DISTINCT birthdate)
2 FROM people;
Run Code
Submit Answer
query result
people
films
count
6152
Showing 1 out of 1 rows

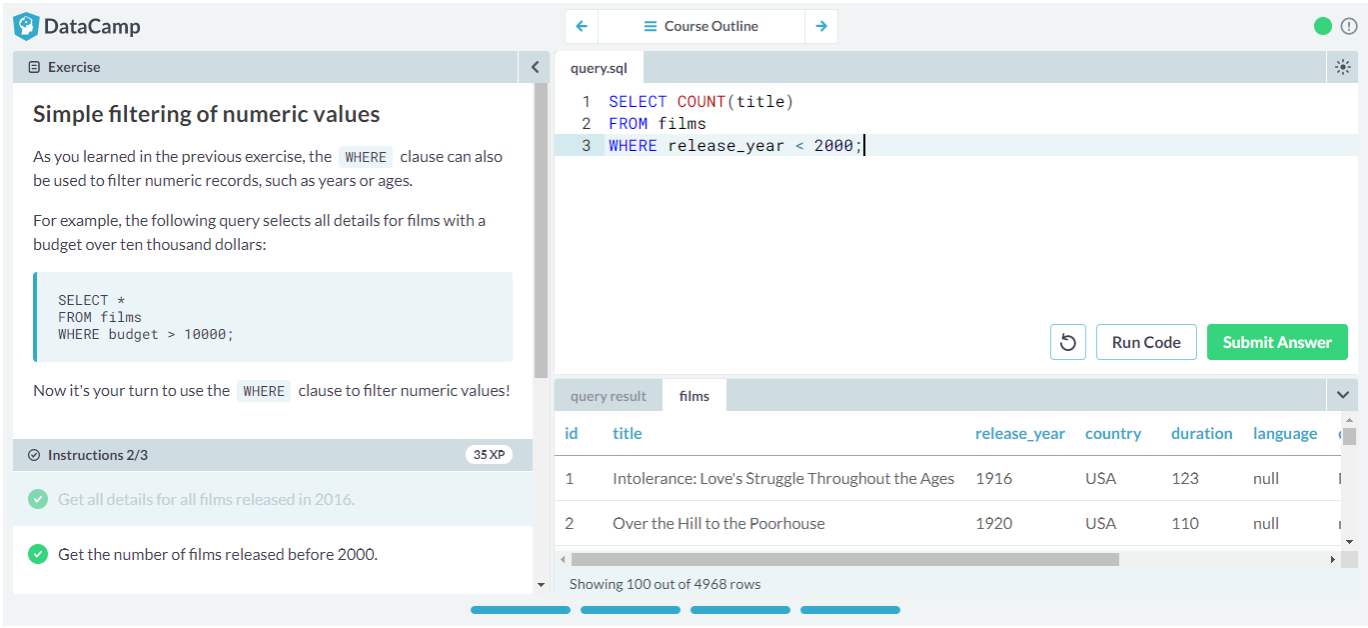

DataCamp
Course Outline
Exercise
Instructions 4/5
20 XP
Count the number of rows in the people table.
Count the number of (non-missing) birth dates in the people table.
Count the number of unique birth dates in the people table.
Count the number of unique languages in the films table.
query.sql
1 SELECT COUNT(DISTINCT language)
2 FROM films
Run Code
Submit Answer
query result
people
films

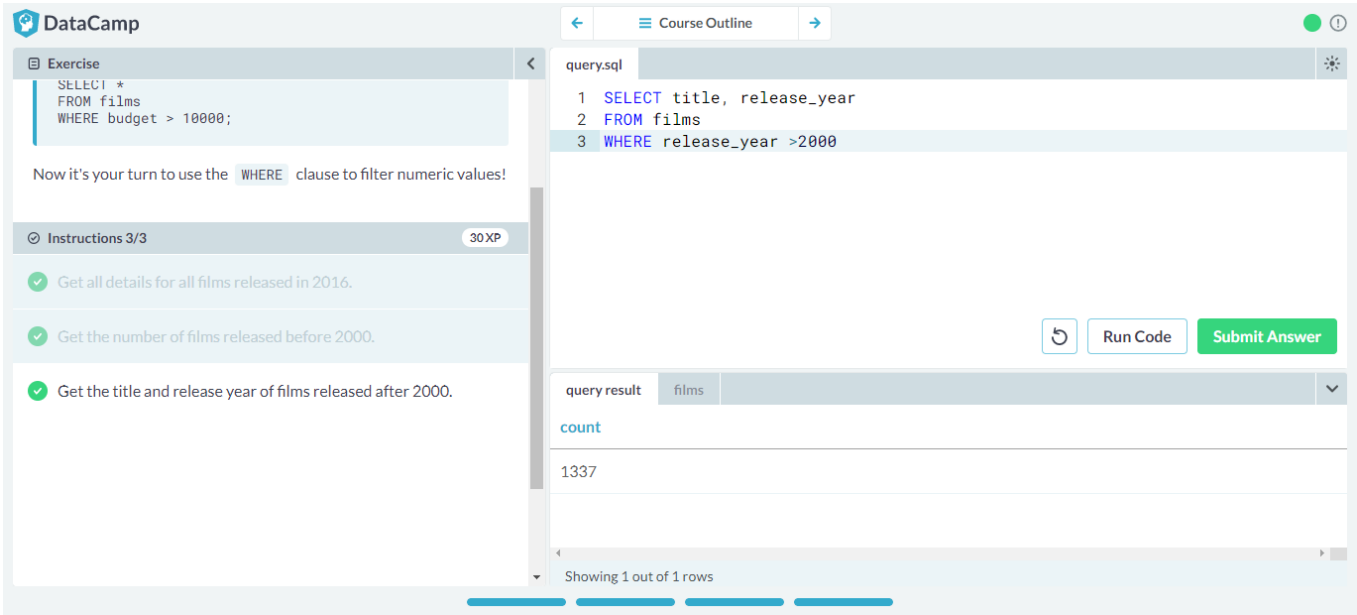

DataCamp
Course Outline
Exercise
Instructions 5/5
20 XP
Count the number of rows in the people table.
Count the number of (non-missing) birth dates in the people table.
Count the number of unique birth dates in the people table.
Count the number of unique languages in the films table.
Count the number of unique countries in the films table.
query.sql
1 SELECT COUNT(DISTINCT country)
2 FROM films
Run Code
Submit Answer
query result
people
films
id
title
release_year
country
duration
language
1
Intolerance: Love's Struggle Throughout the Ages
1916
USA
123
null
2
Over the Hill to the Poorhouse
1920
USA
110
null
Showing 100 out of 4968 rows

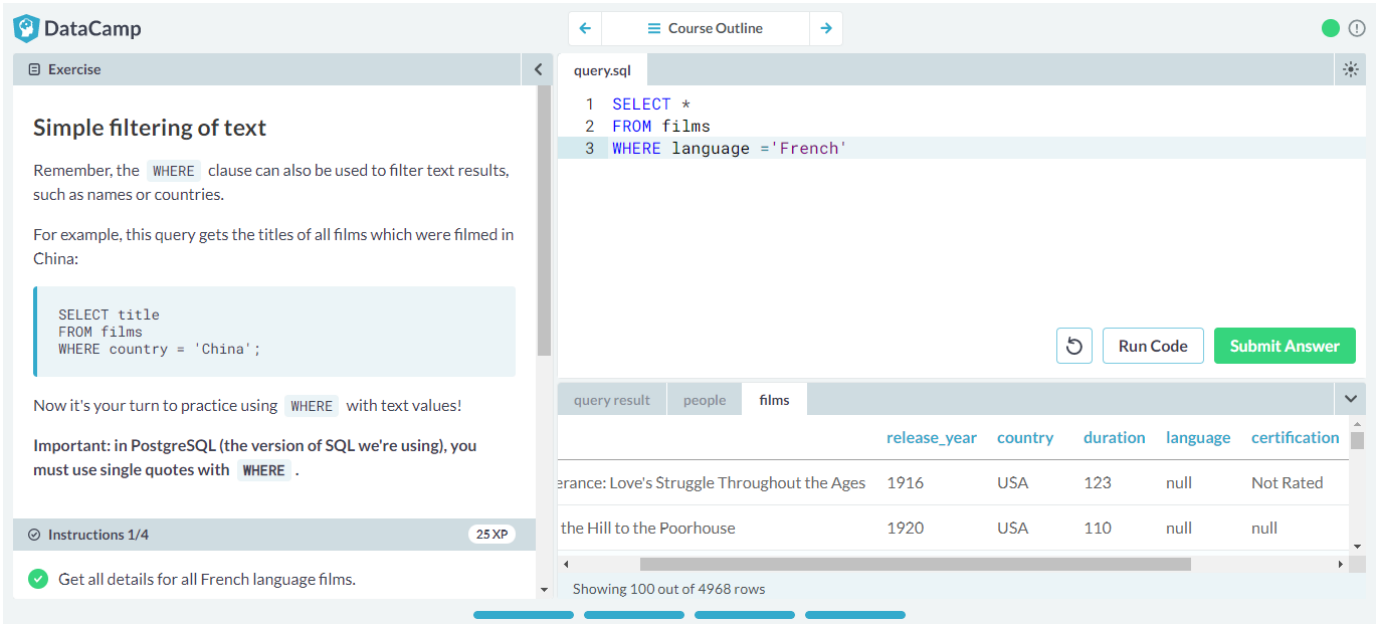

DataCamp
Course Outline
Filtering results
Congrats on finishing the first chapter! You now know how to select columns and perform basic counts. This chapter will focus on filtering your results.
In SQL, the WHERE keyword allows you to filter based on both text and numeric values in a table. There are a few different comparison operators you can use:
= equal
<> not equal
< less than
> greater than
<= less than or equal to
>= greater than or equal to
For example, you can filter text records such as title . The following code returns all films with the title 'Metropolis' :
SELECT title
FROM films
WHERE title = 'Metropolis';

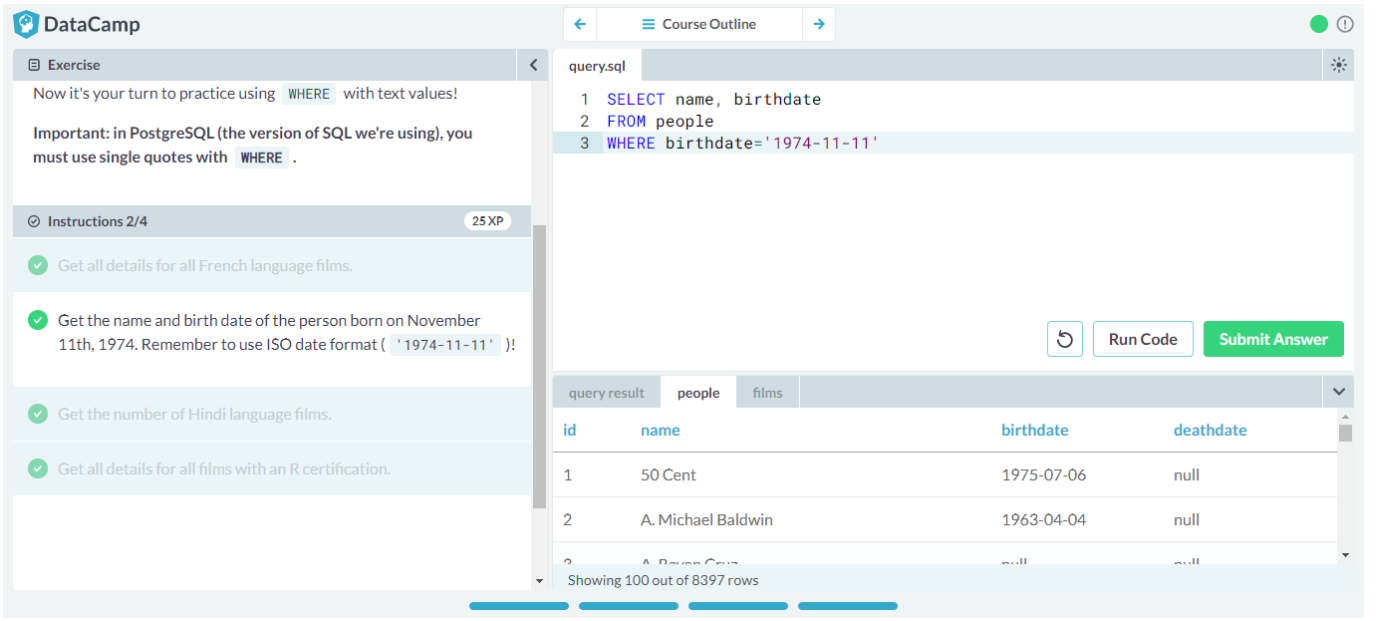

DataCamp
Course Outline
Notice that the WHERE clause always comes after the FROM statement!
Note that in this course we will use <> and not != for the not equal operator, as per the SQL standard.
What does the following query return?
SELECT title
FROM films
WHERE release_year > 2000;
Answer the question
50 XP
Possible Answers
Films released before the year 2000
press 1
Films released after the year 2000
press 2
Films released after the year 2001
press 3

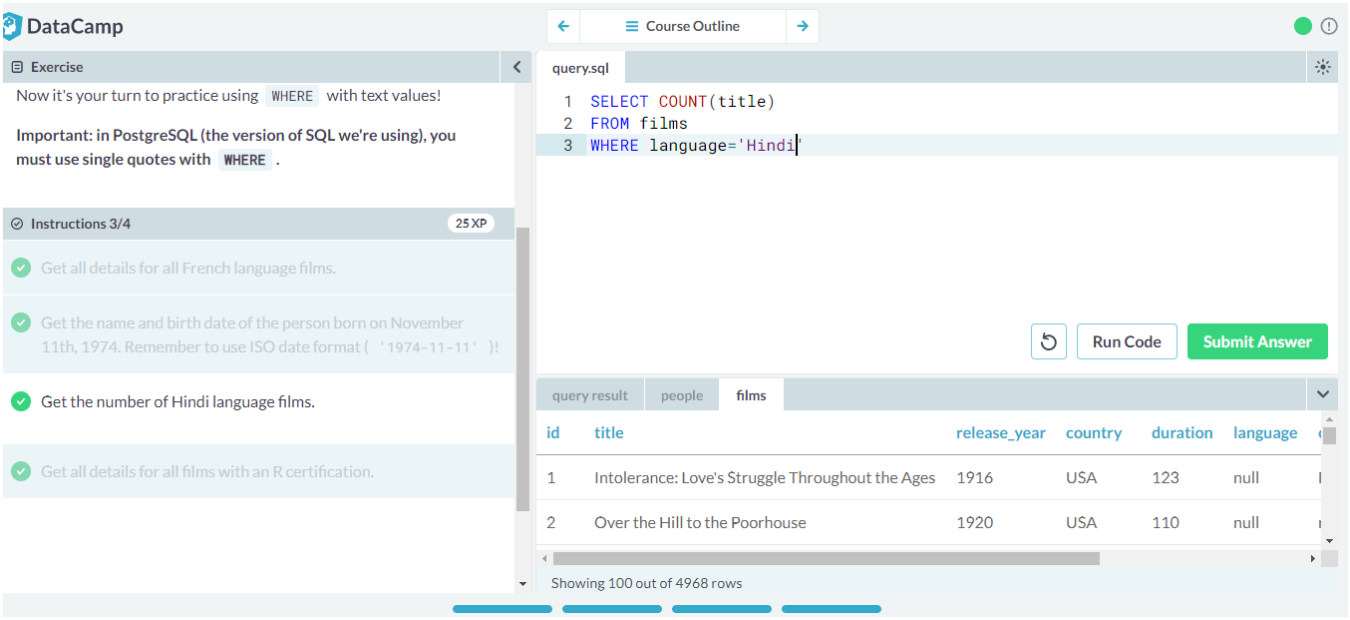

DataCamp
Course Outline
Exercise
Simple filtering of numeric values
As you learned in the previous exercise, the WHERE clause can also be used to filter numeric records, such as years or ages.
For example, the following query selects all details for films with a budget over ten thousand dollars:
SELECT *
FROM films
WHERE budget > 10000;
Now it's your turn to use the WHERE clause to filter numeric values!
Instructions 1/3
35 XP
Get all details for all films released in 2016.
Get the number of films released before 2000.
query.sql
SELECT *
FROM films
WHERE release_year=2016
Run Code
Submit Answer
query result
films
release_year
country
duration
language
certification
gross
ove's Struggle Throughout the Ages
1916
USA
123
null
Not Rated
null
o the Poorhouse
1920
USA
110
null
null
Showing 100 out of 4968 rows

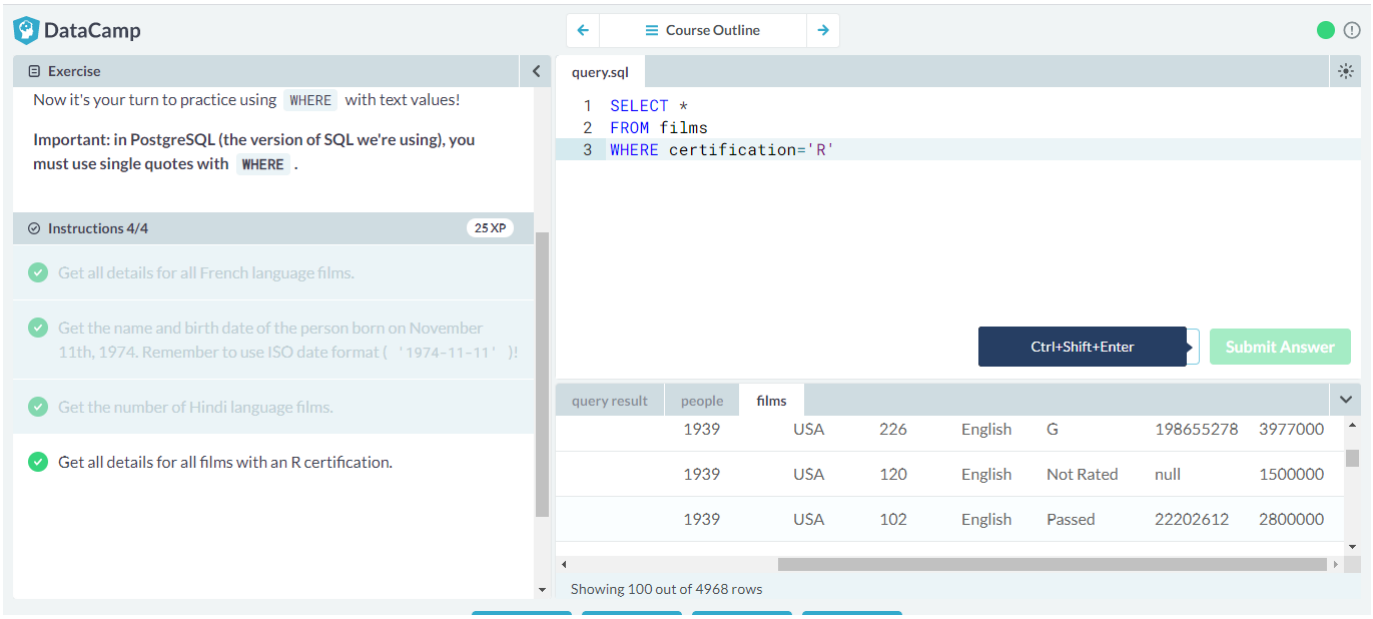

DataCamp
Course Outline
Exercise
Simple filtering of numeric values
As you learned in the previous exercise, the WHERE clause can also be used to filter numeric records, such as years or ages.
For example, the following query selects all details for films with a budget over ten thousand dollars:
SELECT *
FROM films
WHERE budget > 10000;
Now it's your turn to use the WHERE clause to filter numeric values!
Instructions 2/3
35 XP
Get all details for all films released in 2016.
Get the number of films released before 2000.
query.sql
1 SELECT COUNT(title)
2 FROM films
3 WHERE release_year < 2000;
Run Code
Submit Answer
query result
films
id
title
release_year
country
duration
language
1
Intolerance: Love's Struggle Throughout the Ages
1916
USA
123
null
2
Over the Hill to the Poorhouse
1920
USA
110
null
Showing 100 out of 4968 rows

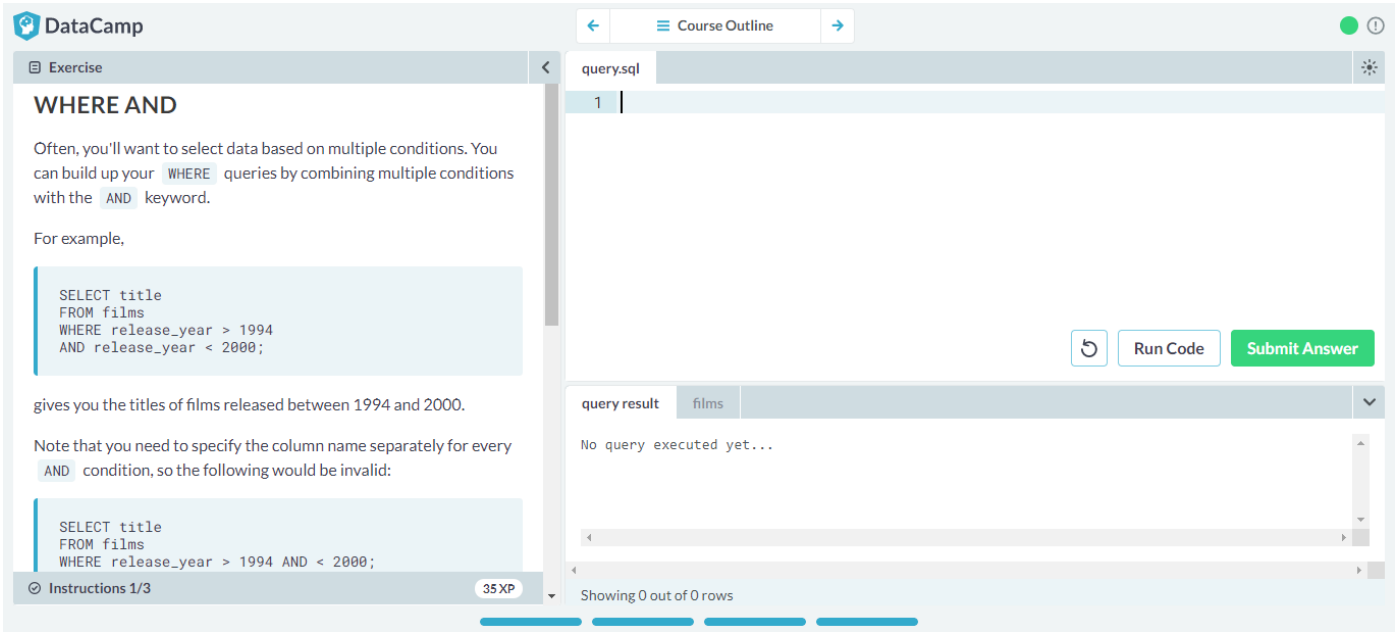

DataCamp
Course Outline
Exercise
SELECT *
FROM films
WHERE budget > 10000;
Now it's your turn to use the WHERE clause to filter numeric values!
Instructions 3/3
30 XP
Get all details for all films released in 2016.
Get the number of films released before 2000.
Get the title and release year of films released after 2000.
query.sql
1 SELECT title, release_year
2 FROM films
3 WHERE release_year >2000
Run Code
Submit Answer
query result
films
count
1337
Showing 1 out of 1 rows

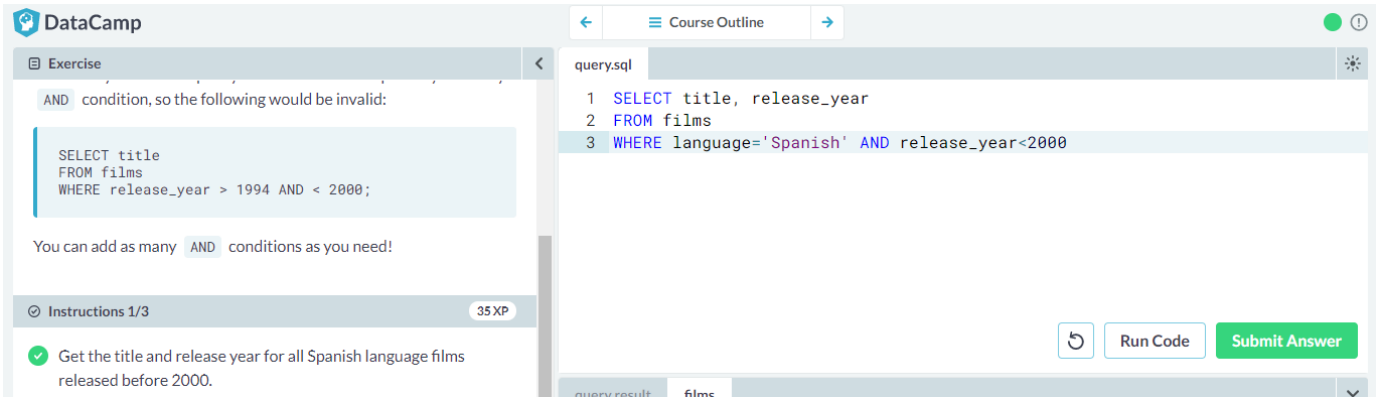

DataCamp
Course Outline
Exercise
Simple filtering of text
Remember, the WHERE clause can also be used to filter text results, such as names or countries.
For example, this query gets the titles of all films which were filmed in China:
SELECT title
FROM films
WHERE country = 'China';
Now it's your turn to practice using WHERE with text values!
Important: in PostgreSQL (the version of SQL we're using), you must use single quotes with WHERE .
Instructions 1/4
25 XP
Get all details for all French language films.
query.sql
1 SELECT *
2 FROM films
3 WHERE language ='French'
Run Code
Submit Answer
query result
people
films
release_year
country
duration
language
certification
erance: Love's Struggle Throughout the Ages
1916
USA
123
null
Not Rated
the Hill to the Poorhouse
1920
USA
110
null
null
Showing 100 out of 4968 rows

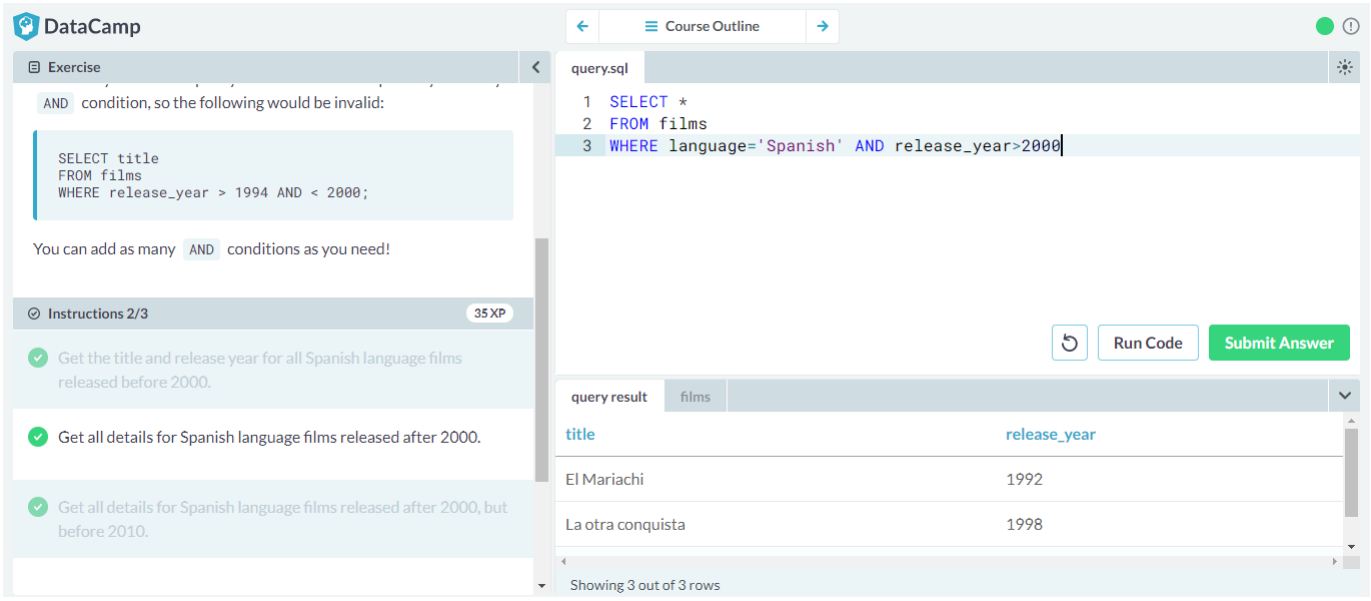

DataCamp
Course Outline
Exercise
Now it's your turn to practice using WHERE with text values!
Important: in PostgreSQL (the version of SQL we're using), you must use single quotes with WHERE .
Instructions 2/4
25 XP
Get all details for all French language films.
Get the name and birth date of the person born on November 11th, 1974. Remember to use ISO date format ('1974-11-11')!
Get the number of Hindi language films.
Get all details for all films with an R certification.
query.sql
1 SELECT name, birthdate
2 FROM people
3 WHERE birthdate='1974-11-11'
Run Code
Submit Answer
query result
people
films
id
name
birthdate
deathdate
1
50 Cent
1975-07-06
null
2
A. Michael Baldwin
1963-04-04
null
Showing 100 out of 8397 rows

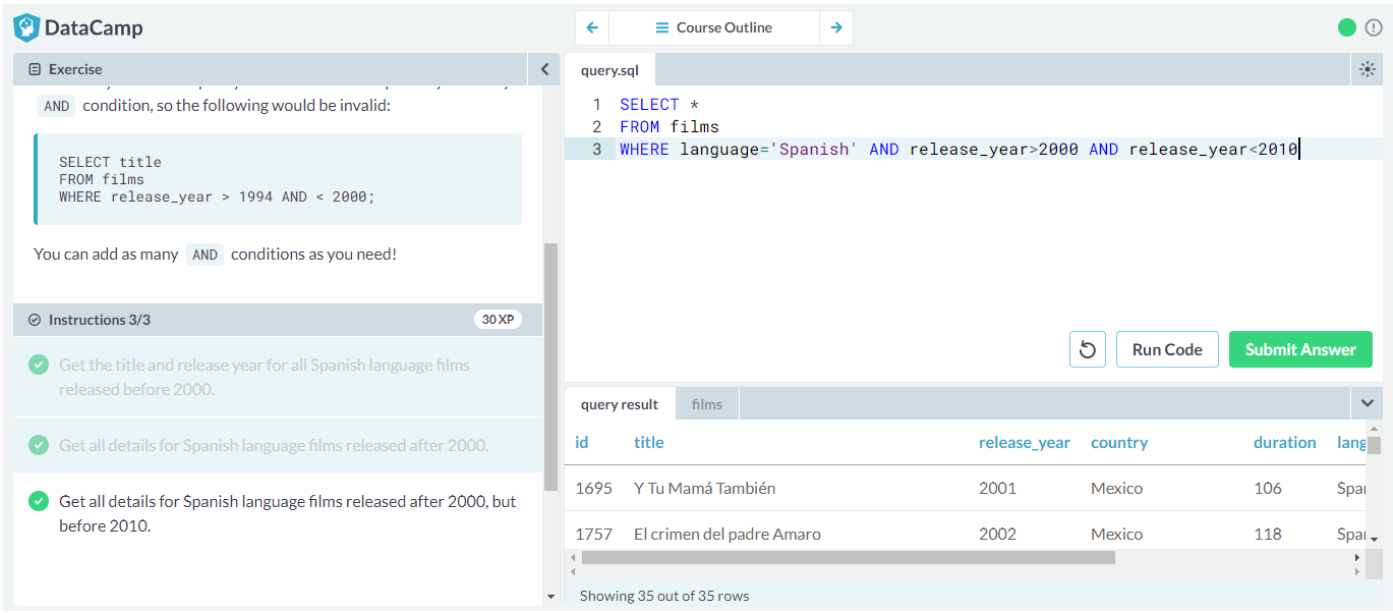

DataCamp
Course Outline
Exercise
Now it's your turn to practice using WHERE with text values!
Important: in PostgreSQL (the version of SQL we're using), you must use single quotes with WHERE .
Instructions 3/4
25 XP
Get all details for all French language films.
Get the name and birth date of the person born on November 11th, 1974. Remember to use ISO date format ('1974-11-11')!
Get the number of Hindi language films.
Get all details for all films with an R certification.
query.sql
1 SELECT COUNT(title)
2 FROM films
3 WHERE language='Hindi'
Run Code
Submit Answer
query result
people
films
id
title
release_year
country
duration
language
1
Intolerance: Love's Struggle Throughout the Ages
1916
USA
123
null
2
Over the Hill to the Poorhouse
1920
USA
110
null
Showing 100 out of 4968 rows


DataCamp
Course Outline
Exercise
Now it's your turn to practice using WHERE with text values!
Important: in PostgreSQL (the version of SQL we're using), you must use single quotes with WHERE .
Instructions 4/4
25 XP
Get all details for all French language films.
Get the name and birth date of the person born on November 11th, 1974. Remember to use ISO date format ('1974-11-11')!
Get the number of Hindi language films.
Get all details for all films with an R certification.
query.sql
1 SELECT *
2 FROM films
3 WHERE certification='R'
Ctrl+Shift+Enter
Submit Answer
query result
people
films
1939 USA 226 English G 198655278 3977000
1939 USA 120 English Not Rated null 1500000
1939 USA 102 English Passed 22202612 2800000
Showing 100 out of 4968 rows

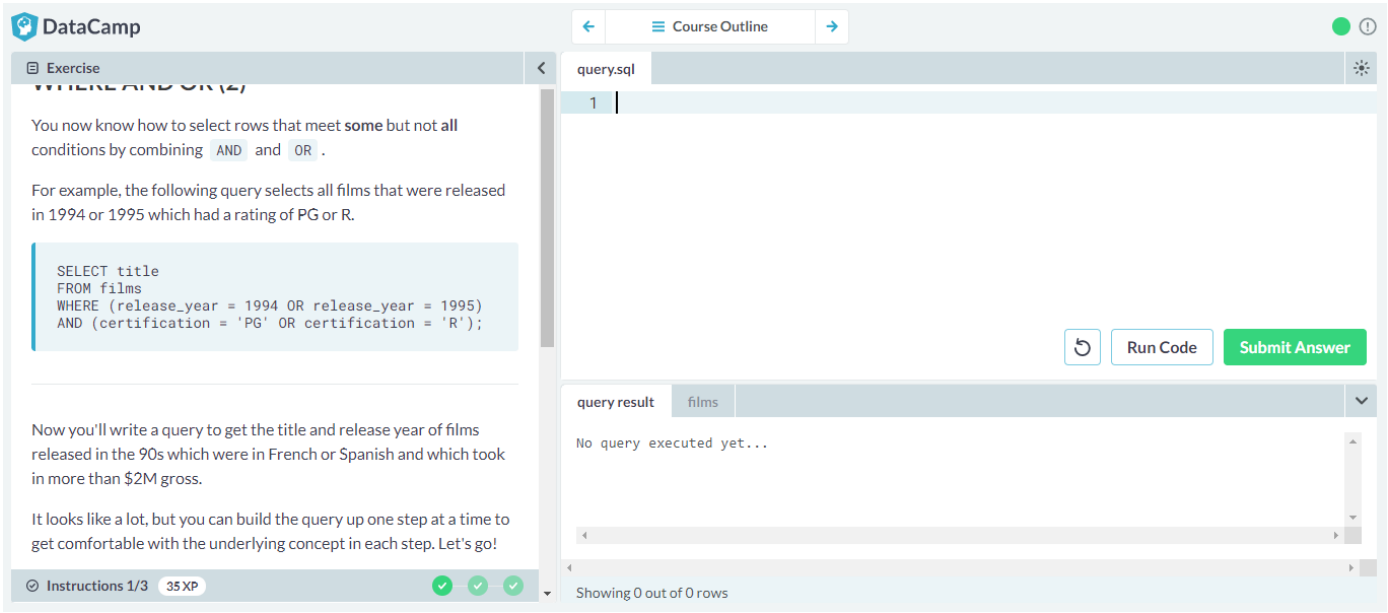

DataCamp
Course Outline
Exercise
WHERE AND
Often, you'll want to select data based on multiple conditions. You can build up your WHERE queries by combining multiple conditions with the AND keyword.
For example,
SELECT title
FROM films
WHERE release_year > 1994
AND release_year < 2000;
gives you the titles of films released between 1994 and 2000.
Note that you need to specify the column name separately for every AND condition, so the following would be invalid:
SELECT title
FROM films
WHERE release_year > 1994 AND < 2000;
Instructions 1/3
35 XP
query.sql
1
Run Code
Submit Answer
query result
films
No query executed yet...
Showing 0 out of 0 rows

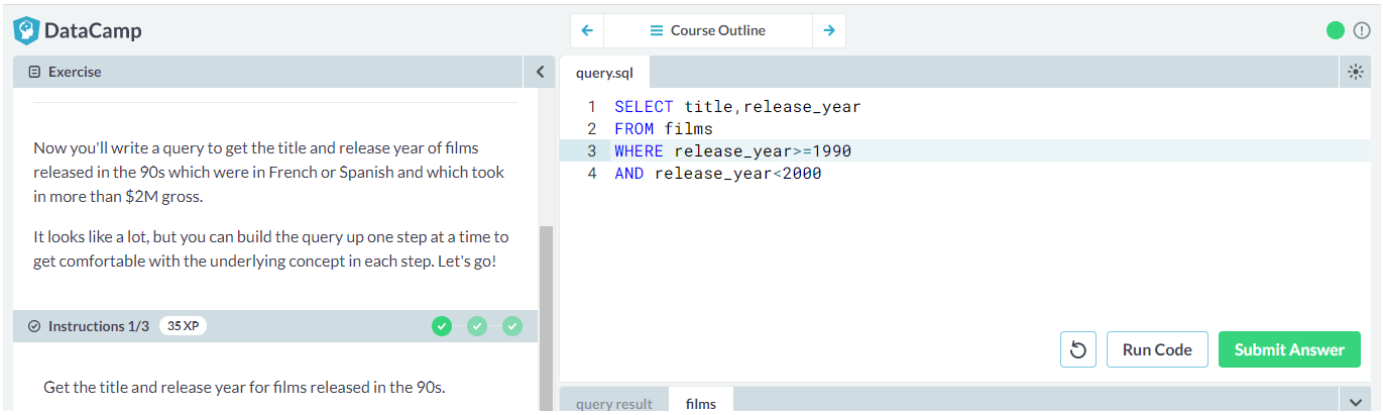

DataCamp
Course Outline
Exercise
AND condition, so the following would be invalid:
SELECT title
FROM films
WHERE release_year > 1994 AND < 2000;
You can add as many AND conditions as you need!
Instructions 1/3
35 XP
Get the title and release year for all Spanish language films released before 2000.
query.sql
1 SELECT title, release_year
2 FROM films
3 WHERE language='Spanish' AND release_year<2000
Run Code
Submit Answer
films

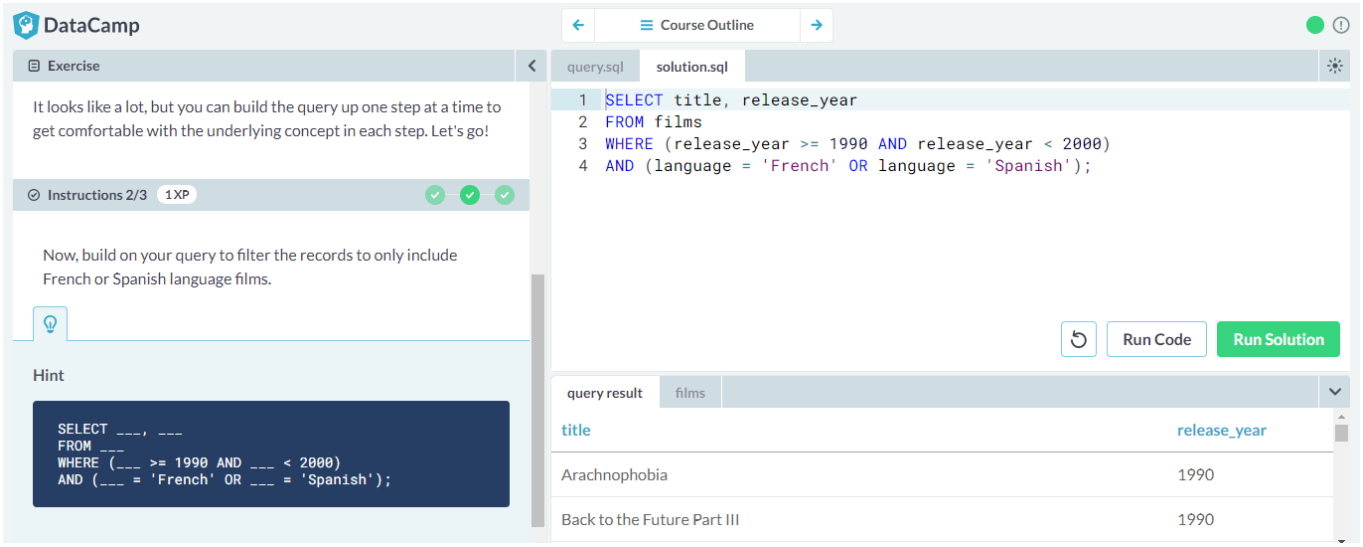

DataCamp
Course Outline
Exercise
AND condition, so the following would be invalid:
SELECT title
FROM films
WHERE release_year > 1994 AND < 2000;
You can add as many AND conditions as you need!
Instructions 2/3
35 XP
Get the title and release year for all Spanish language films released before 2000.
Get all details for Spanish language films released after 2000.
Get all details for Spanish language films released after 2000, but before 2010.
query.sql
1 SELECT *
2 FROM films
3 WHERE language='Spanish' AND release_year>2000
Run Code
Submit Answer
query result
films
title
release_year
El Mariachi
1992
La otra conquista
1998
Showing 3 out of 3 rows

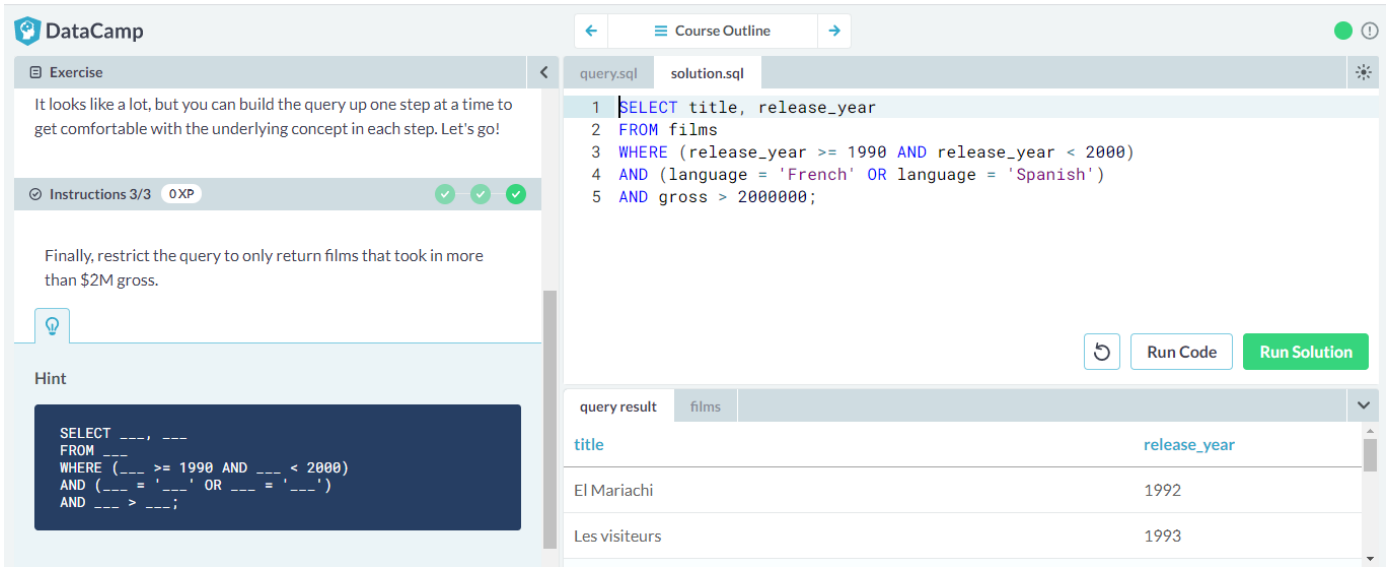

DataCamp
Course Outline
Exercise
AND condition, so the following would be invalid:
SELECT title
FROM films
WHERE release_year > 1994 AND < 2000;
You can add as many AND conditions as you need!
Instructions 3/3
30 XP
Get the title and release year for all Spanish language films released before 2000.
Get all details for Spanish language films released after 2000.
Get all details for Spanish language films released after 2000, but before 2010.
query.sql
1 SELECT *
2 FROM films
3 WHERE language='Spanish' AND release_year>2000 AND release_year<2010
Run Code
Submit Answer
query result
films
id
title
release_year
country
duration
1695
Y Tu Mamá También
2001
Mexico
106
1757
El crimen del padre Amaro
2002
Mexico
118
Showing 35 out of 35 rows

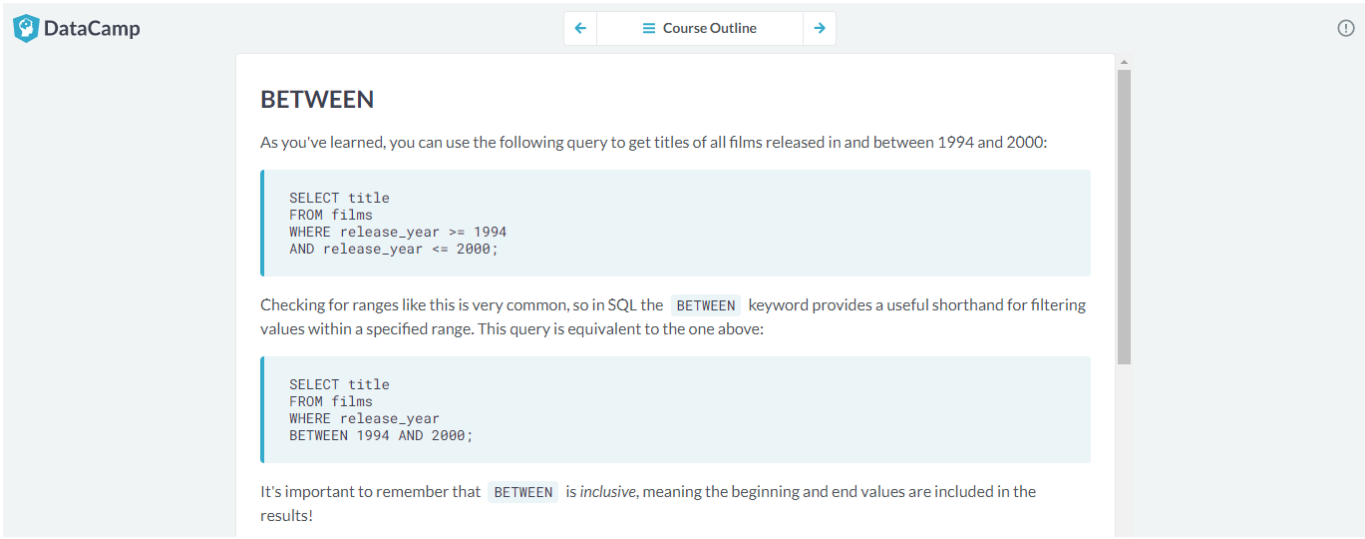

DataCamp
Course Outline
WHERE AND OR
What if you want to select rows based on multiple conditions where some but not all of the conditions need to be met? For this, SQL has the OR operator.
For example, the following returns all films released in either 1994 or 2000:
SELECT title
FROM films
WHERE release_year = 1994
OR release_year = 2000;
Note that you need to specify the column for every OR condition, so the following is invalid:
SELECT title
FROM films
WHERE release_year = 1994 OR 2000;
When combining AND and OR , be sure to enclose the individual clauses in parentheses, like so:
SELECT title
FROM films
WHERE (release_year = 1994 OR release_year = 1995)
AND (certification = 'PG' OR certification = 'R');

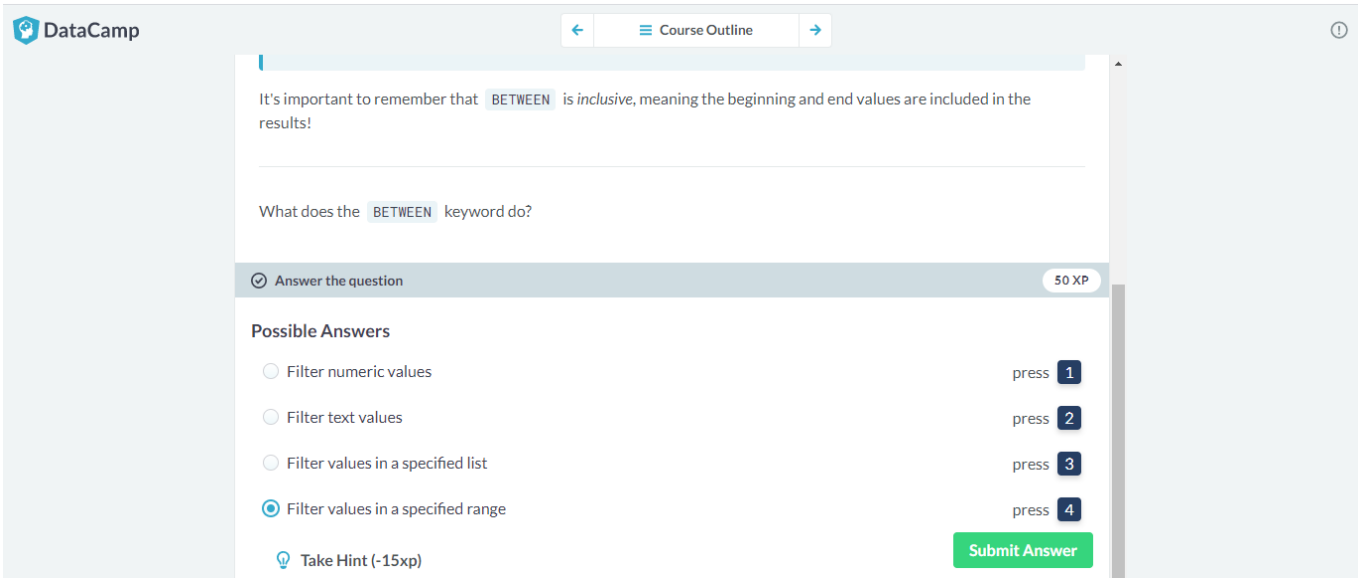

DataCamp
Course Outline
Exercise
You now know how to select rows that meet some but not all conditions by combining AND and OR .
For example, the following query selects all films that were released in 1994 or 1995 which had a rating of PG or R.
SELECT title
FROM films
WHERE (release_year = 1994 OR release_year = 1995)
AND (certification = 'PG' OR certification = 'R');
Now you'll write a query to get the title and release year of films released in the 90s which were in French or Spanish and which took in more than $2M gross.
It looks like a lot, but you can build the query up one step at a time to get comfortable with the underlying concept in each step. Let's go!
Instructions 1/3
35 XP
query.sql
1
Run Code
Submit Answer
query result
films
No query executed yet...
Showing 0 out of 0 rows

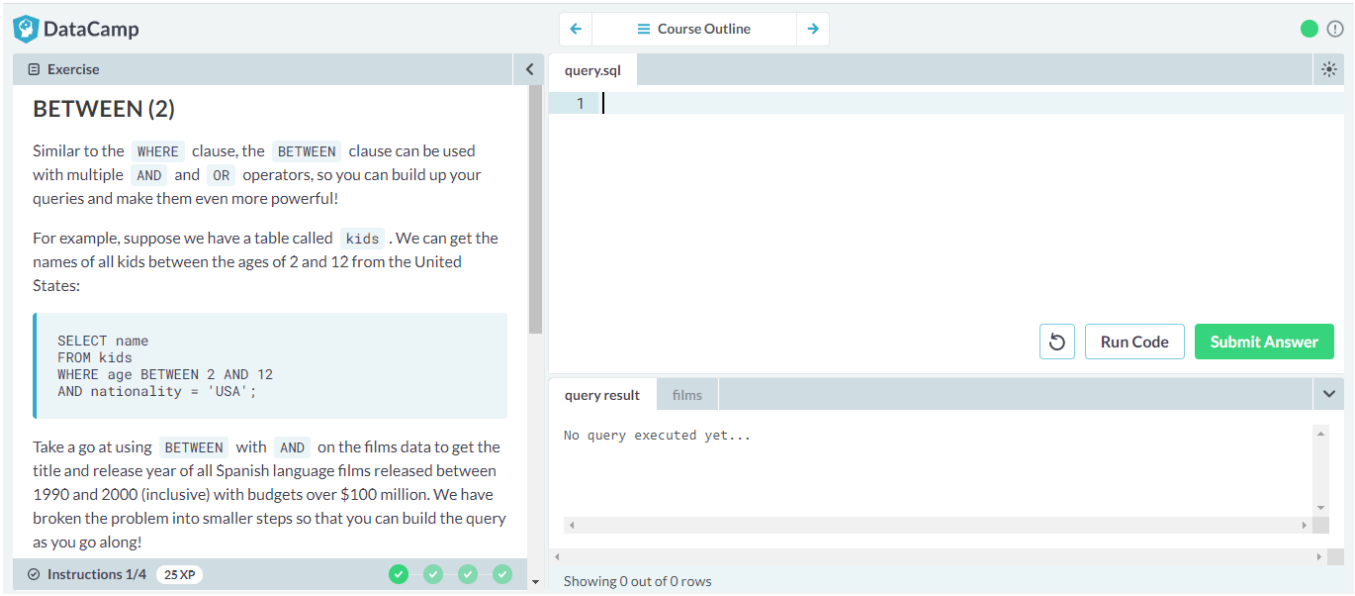

DataCamp
Course Outline
Exercise
Now you'll write a query to get the title and release year of films released in the 90s which were in French or Spanish and which took in more than $2M gross.
It looks like a lot, but you can build the query up one step at a time to get comfortable with the underlying concept in each step. Let's go!
Instructions 1/3
35 XP
Get the title and release year for films released in the 90s.
query.sql
SELECT title,release_year
FROM films
WHERE release_year>=1990
AND release_year<2000
Run Code
Submit Answer
query result
films

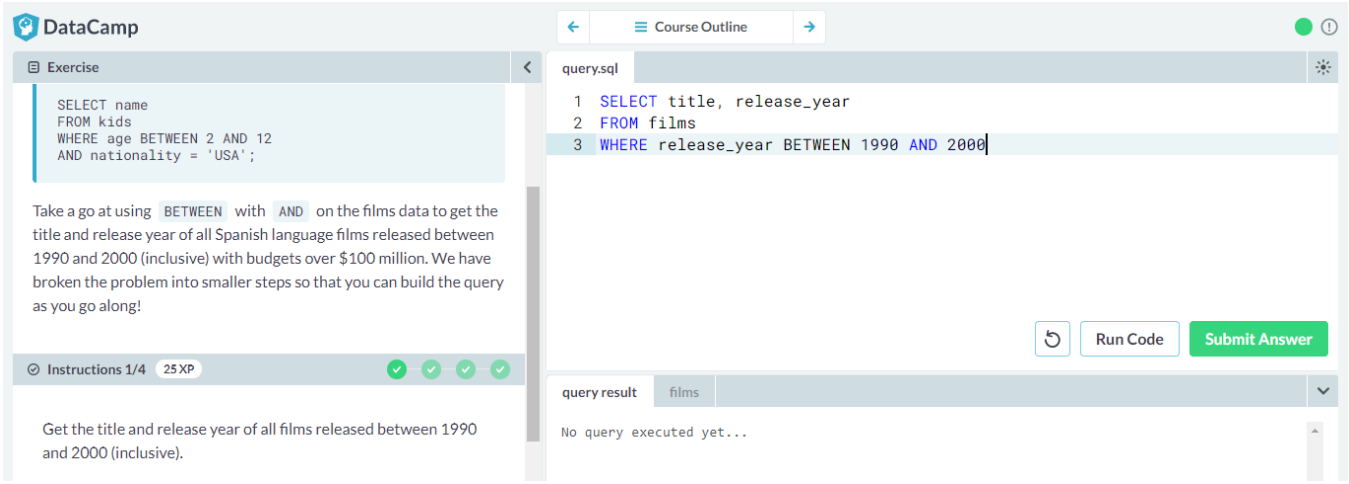

DataCamp
Course Outline
Exercise
It looks like a lot, but you can build the query up one step at a time to get comfortable with the underlying concept in each step. Let's go!
Instructions 2/3
1 XP
Now, build on your query to filter the records to only include French or Spanish language films.
Hint
SELECT ___, ___
FROM ___
WHERE (___ >= 1990 AND ___ < 2000)
AND (___ = 'French' OR ___ = 'Spanish');
query.sql
solution.sql
SELECT title, release_year
FROM films
WHERE (release_year >= 1990 AND release_year < 2000)
AND (language = 'French' OR language = 'Spanish');
Run Code
Run Solution
query result
films
title
release_year
Arachnophobia
1990
Back to the Future Part III
1990

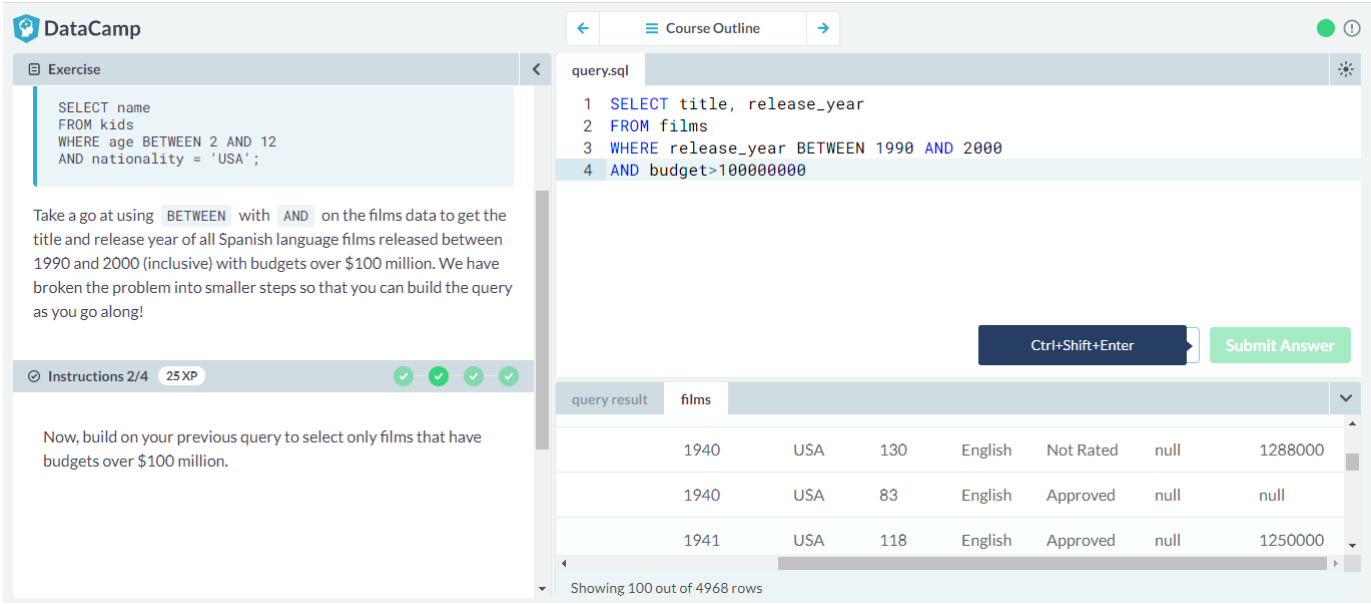

DataCamp
Course Outline
Exercise
It looks like a lot, but you can build the query up one step at a time to get comfortable with the underlying concept in each step. Let's go!
Instructions 3/3
0 XP
Finally, restrict the query to only return films that took in more than $2M gross.
Hint
SELECT ___, ___
FROM ___
WHERE (___ >= 1990 AND ___ < 2000)
AND (___ = '___' OR ___ = '___')
AND ___ > ___;
query.sql
solution.sql
SELECT title, release_year
FROM films
WHERE (release_year >= 1990 AND release_year < 2000)
AND (language = 'French' OR language = 'Spanish')
AND gross > 2000000;
Run Code
Run Solution
query result
films
title
release_year
El Mariachi
1992
Les visiteurs
1993

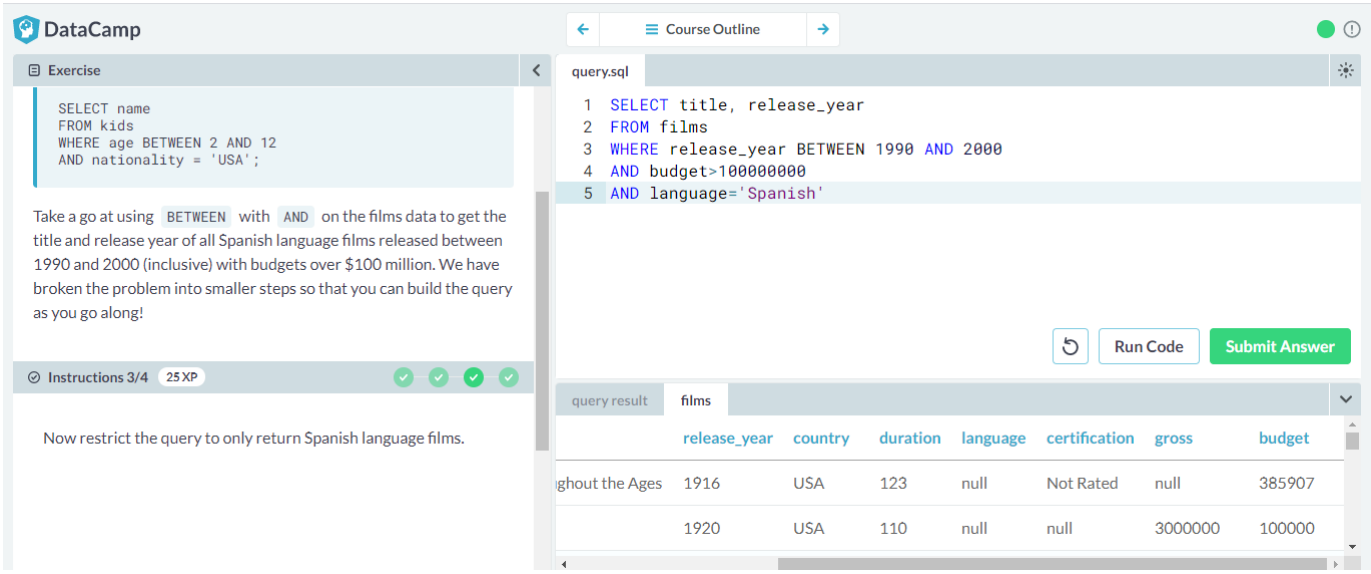

DataCamp
Course Outline
BETWEEN
As you've learned, you can use the following query to get titles of all films released in and between 1994 and 2000:
SELECT title
FROM films
WHERE release_year >= 1994
AND release_year <= 2000;
Checking for ranges like this is very common, so in SQL the BETWEEN keyword provides a useful shorthand for filtering values within a specified range. This query is equivalent to the one above:
SELECT title
FROM films
WHERE release_year
BETWEEN 1994 AND 2000;
It's important to remember that BETWEEN is inclusive, meaning the beginning and end values are included in the results!

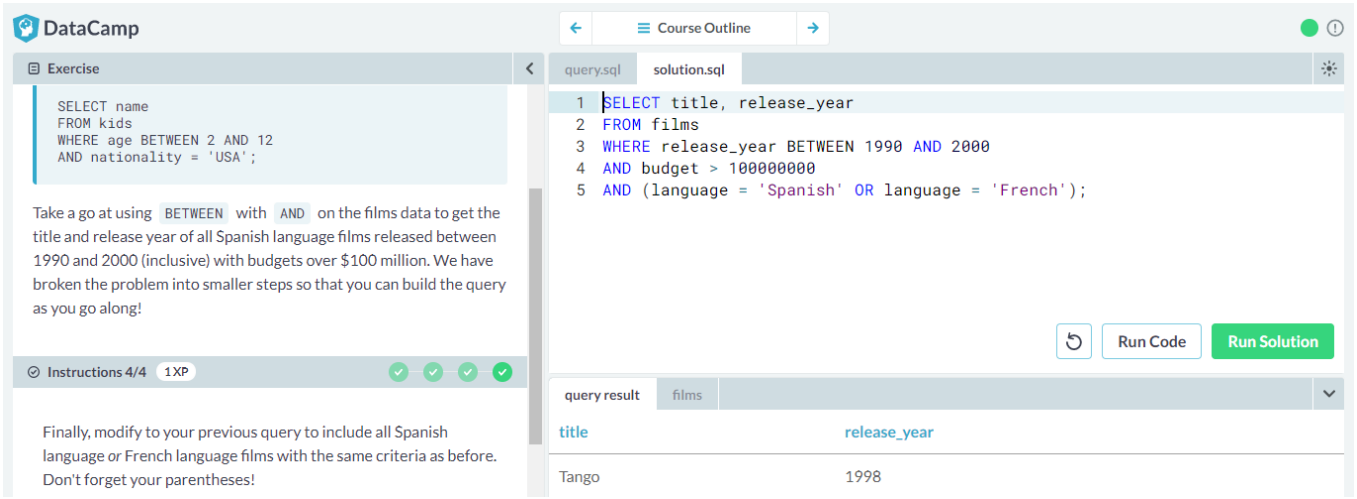

DataCamp
Course Outline
It's important to remember that BETWEEN is inclusive, meaning the beginning and end values are included in the results!
What does the BETWEEN keyword do?
Answer the question
50 XP
Possible Answers
Filter numeric values
press 1
Filter text values
press 2
Filter values in a specified list
press 3
Filter values in a specified range
press 4
Take Hint (-15xp)
Submit Answer

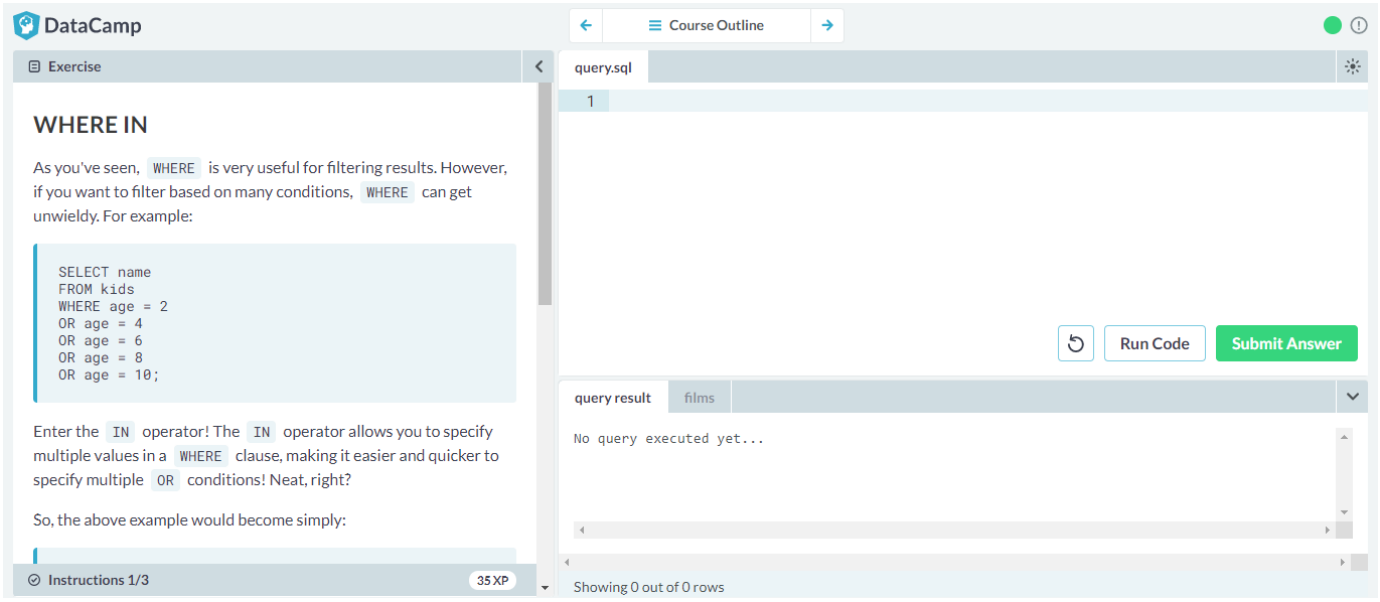

DataCamp
Course Outline
Exercise
BETWEEN (2)
Similar to the WHERE clause, the BETWEEN clause can be used with multiple AND and OR operators, so you can build up your queries and make them even more powerful!
For example, suppose we have a table called kids . We can get the names of all kids between the ages of 2 and 12 from the United States:
SELECT name
FROM kids
WHERE age BETWEEN 2 AND 12
AND nationality = 'USA';
Take a go at using BETWEEN with AND on the films data to get the title and release year of all Spanish language films released between 1990 and 2000 (inclusive) with budgets over $100 million. We have broken the problem into smaller steps so that you can build the query as you go along!
Instructions 1/4
25 XP
query.sql
1
Run Code
Submit Answer
query result
films
No query executed yet...
Showing 0 out of 0 rows

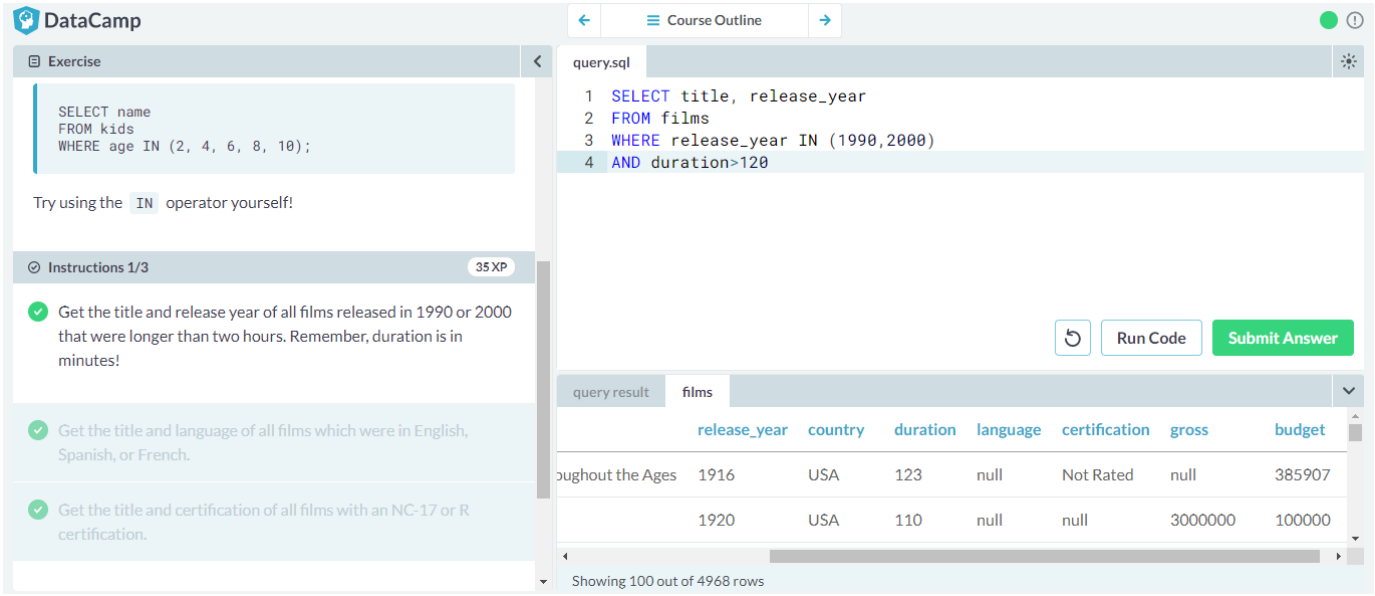

DataCamp
Course Outline
Exercise
SELECT name
FROM kids
WHERE age BETWEEN 2 AND 12
AND nationality = 'USA';
Take a go at using BETWEEN with AND on the films data to get the title and release year of all Spanish language films released between 1990 and 2000 (inclusive) with budgets over $100 million. We have broken the problem into smaller steps so that you can build the query as you go along!
Instructions 1/4
25 XP
Get the title and release year of all films released between 1990 and 2000 (inclusive).
query.sql
1 SELECT title, release_year
2 FROM films
3 WHERE release_year BETWEEN 1990 AND 2000
Run Code
Submit Answer
query result
films
No query executed yet...

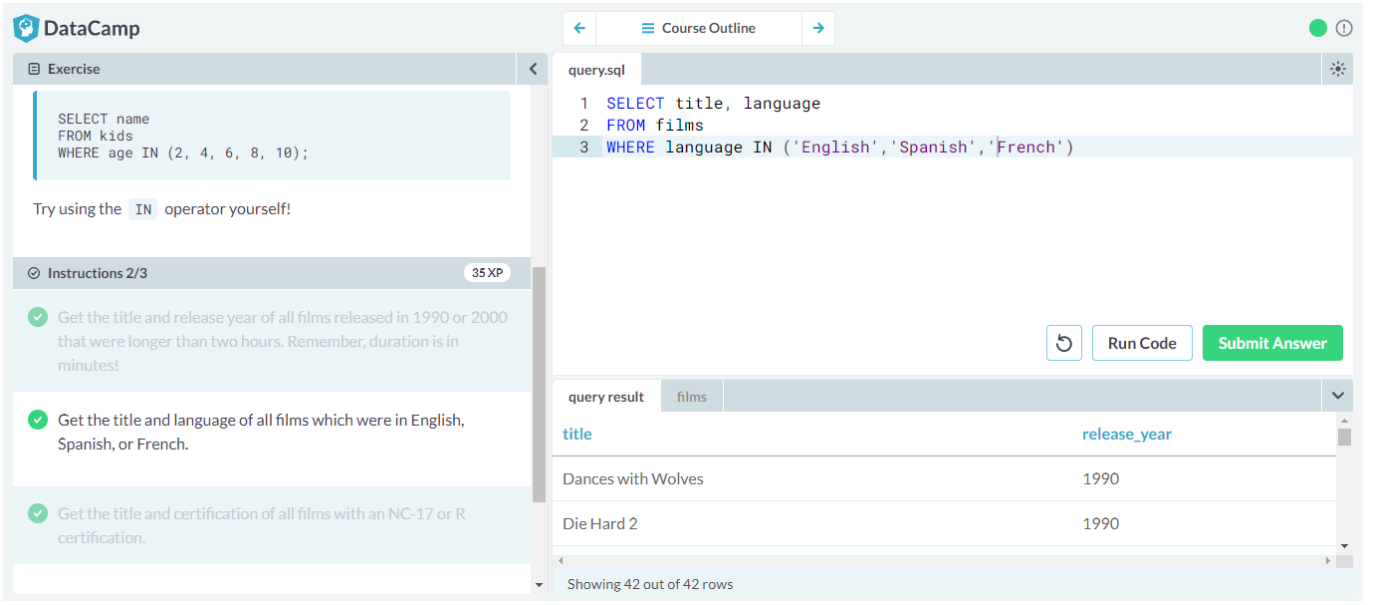

DataCamp
Course Outline
Exercise
SELECT name
FROM kids
WHERE age BETWEEN 2 AND 12
AND nationality = 'USA';
Take a go at using BETWEEN with AND on the films data to get the title and release year of all Spanish language films released between 1990 and 2000 (inclusive) with budgets over $100 million. We have broken the problem into smaller steps so that you can build the query as you go along!
Instructions 2/4
25 XP
Now, build on your previous query to select only films that have budgets over $100 million.
query.sql
1 SELECT title, release_year
2 FROM films
3 WHERE release_year BETWEEN 1990 AND 2000
4 AND budget>100000000
Ctrl+Shift+Enter
Submit Answer
query result
films
1940 USA 130 English Not Rated null 1288000
1940 USA 83 English Approved null null
1941 USA 118 English Approved null 1250000
Showing 100 out of 4968 rows

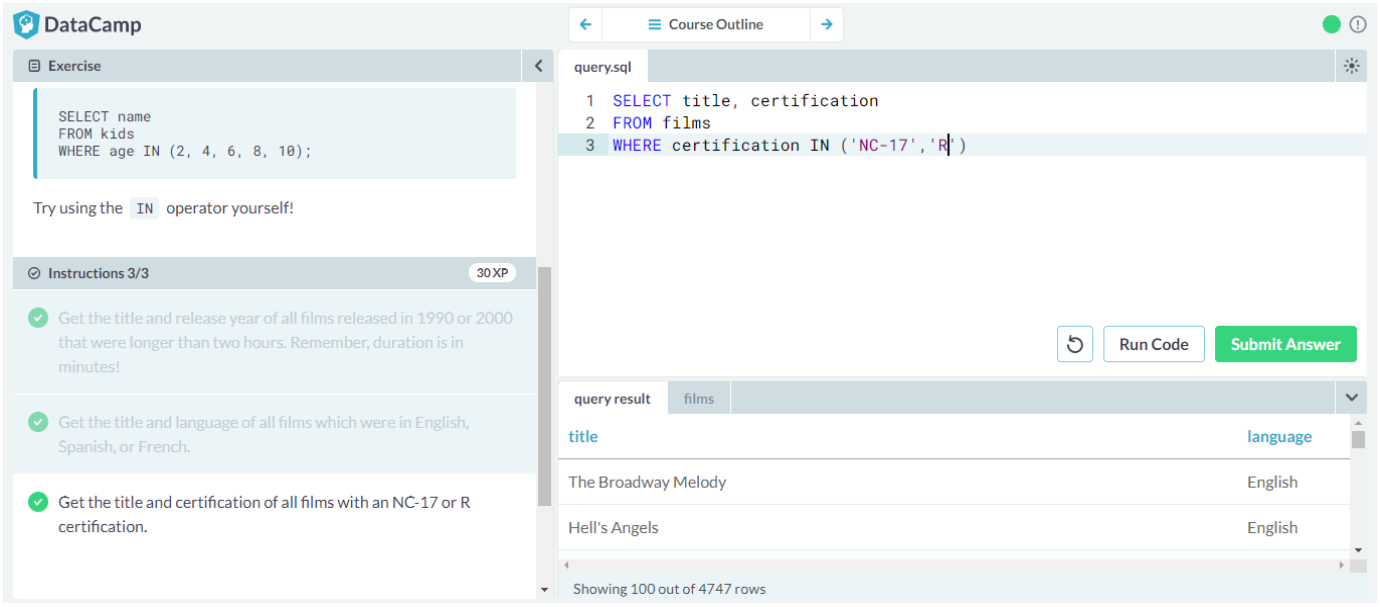

DataCamp
Course Outline
Exercise
SELECT name
FROM kids
WHERE age BETWEEN 2 AND 12
AND nationality = 'USA';
Take a go at using BETWEEN with AND on the films data to get the title and release year of all Spanish language films released between 1990 and 2000 (inclusive) with budgets over $100 million. We have broken the problem into smaller steps so that you can build the query as you go along!
Instructions 3/4
25 XP
Now restrict the query to only return Spanish language films.
query.sql
1 SELECT title, release_year
2 FROM films
3 WHERE release_year BETWEEN 1990 AND 2000
4 AND budget>100000000
5 AND language='Spanish'
Run Code
Submit Answer
query result
films
release_year country duration language certification gross budget
ghout the Ages 1916 USA 123 null Not Rated null 385907
1920 USA 110 null null 3000000 100000

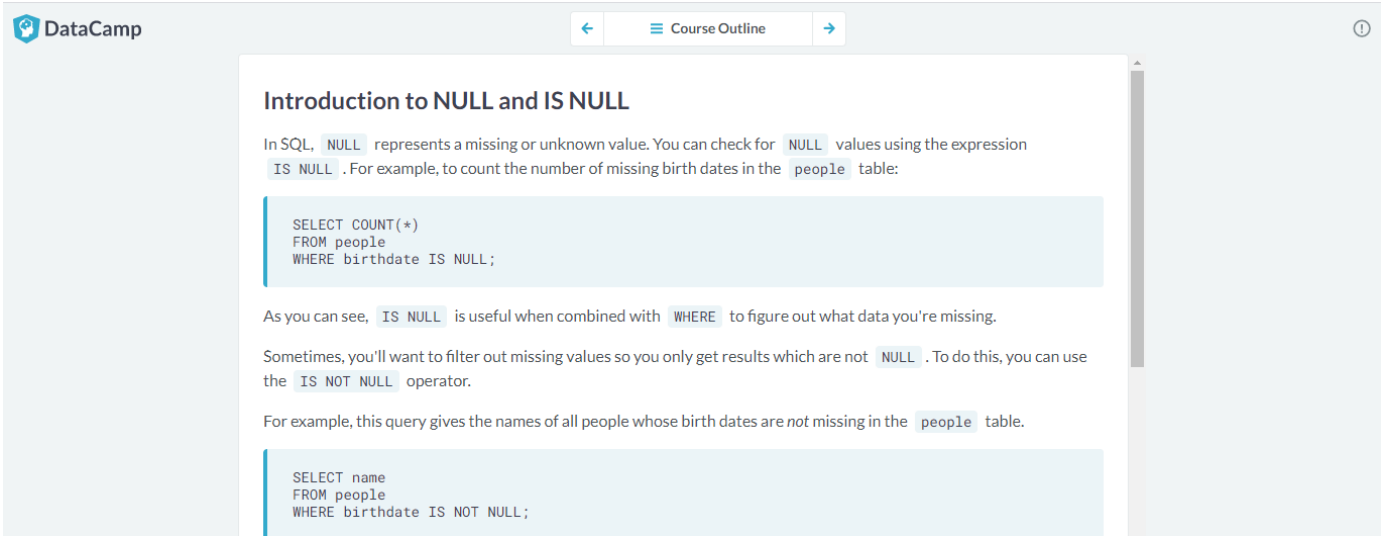

DataCamp
Course Outline
Exercise
SELECT name
FROM kids
WHERE age BETWEEN 2 AND 12
AND nationality = 'USA';
Take a go at using BETWEEN with AND on the films data to get the title and release year of all Spanish language films released between 1990 and 2000 (inclusive) with budgets over $100 million. We have broken the problem into smaller steps so that you can build the query as you go along!
Instructions 4/4
1XP
Finally, modify to your previous query to include all Spanish language or French language films with the same criteria as before. Don't forget your parentheses!
query.sql
solution.sql
1 SELECT title, release_year
2 FROM films
3 WHERE release_year BETWEEN 1990 AND 2000
4 AND budget > 100000000
5 AND (language = 'Spanish' OR language = 'French');
Run Code
Run Solution
query result
films
title
release_year
Tango
1998

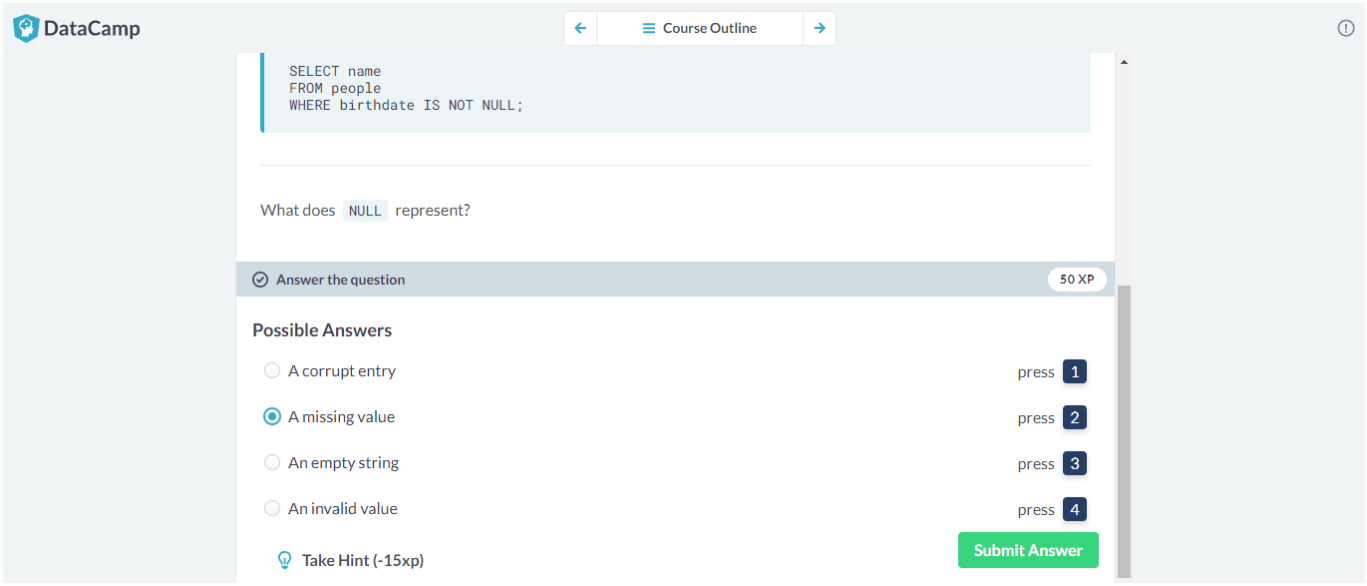

DataCamp
Course Outline
Exercise
WHERE IN
As you've seen, WHERE is very useful for filtering results. However, if you want to filter based on many conditions, WHERE can get unwieldy. For example:
SELECT name
FROM kids
WHERE age = 2
OR age = 4
OR age = 6
OR age = 8
OR age = 10;
Enter the IN operator! The IN operator allows you to specify multiple values in a WHERE clause, making it easier and quicker to specify multiple OR conditions! Neat, right?
So, the above example would become simply:
Instructions 1/3
35 XP
query.sql
1
Run Code
Submit Answer
query result
films
No query executed yet...
Showing 0 out of 0 rows

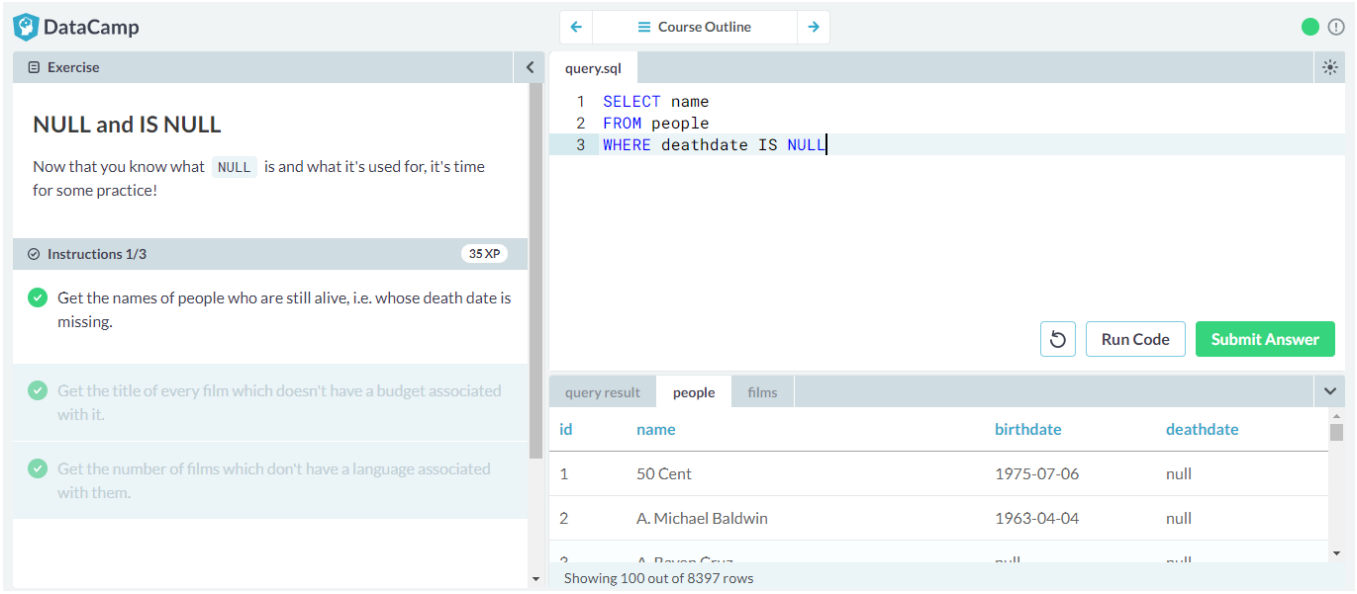

DataCamp
Course Outline
Exercise
SELECT name
FROM kids
WHERE age IN (2, 4, 6, 8, 10);
Try using the IN operator yourself!
Instructions 1/3
35 XP
Get the title and release year of all films released in 1990 or 2000 that were longer than two hours. Remember, duration is in minutes!
Get the title and language of all films which were in English, Spanish, or French.
Get the title and certification of all films with an NC-17 or R certification.
query.sql
1 SELECT title, release_year
2 FROM films
3 WHERE release_year IN (1990,2000)
4 AND duration>120
Run Code
Submit Answer
query result
films
release_year country duration language certification gross budget
oughout the Ages 1916 USA 123 null Not Rated null 385907
1920 USA 110 null null 3000000 100000
Showing 100 out of 4968 rows

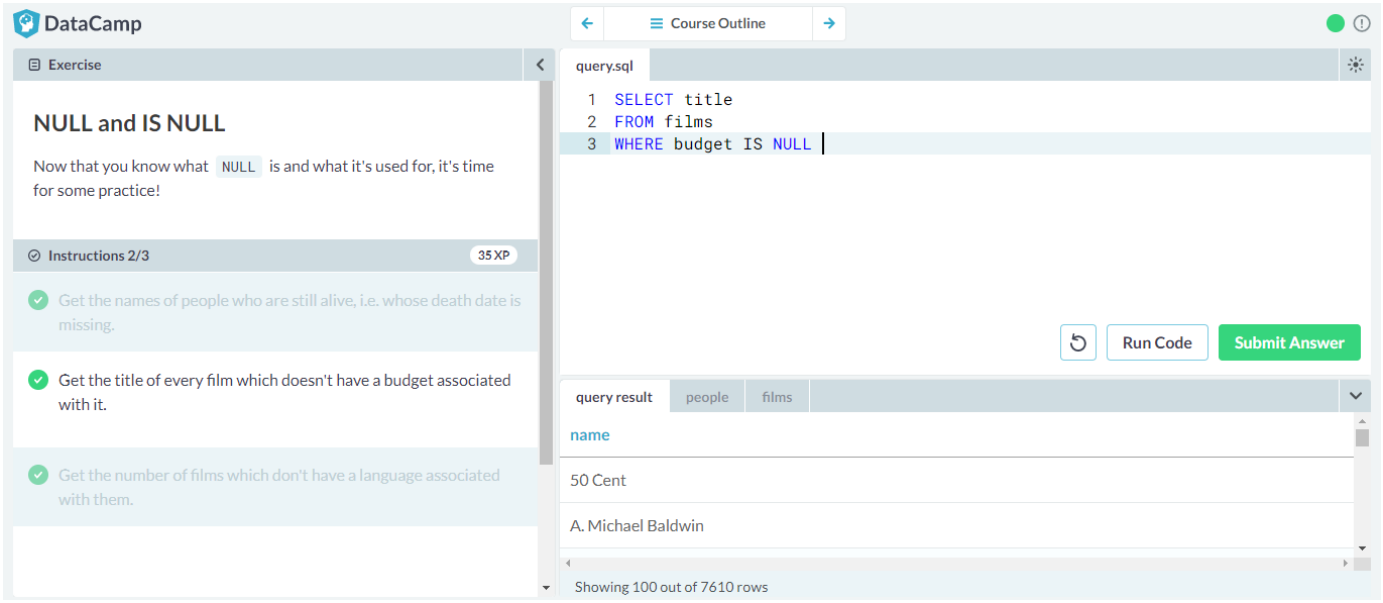

DataCamp
Course Outline
Exercise
SELECT name
FROM kids
WHERE age IN (2, 4, 6, 8, 10);
Try using the IN operator yourself!
Instructions 2/3
35 XP
Get the title and release year of all films released in 1990 or 2000 that were longer than two hours. Remember, duration is in minutes!
Get the title and language of all films which were in English, Spanish, or French.
Get the title and certification of all films with an NC-17 or R certification.
query.sql
1 SELECT title, language
2 FROM films
3 WHERE language IN ('English','Spanish','French')
Run Code
Submit Answer
query result
films
title release_year
Dances with Wolves 1990
Die Hard 2 1990
Showing 42 out of 42 rows

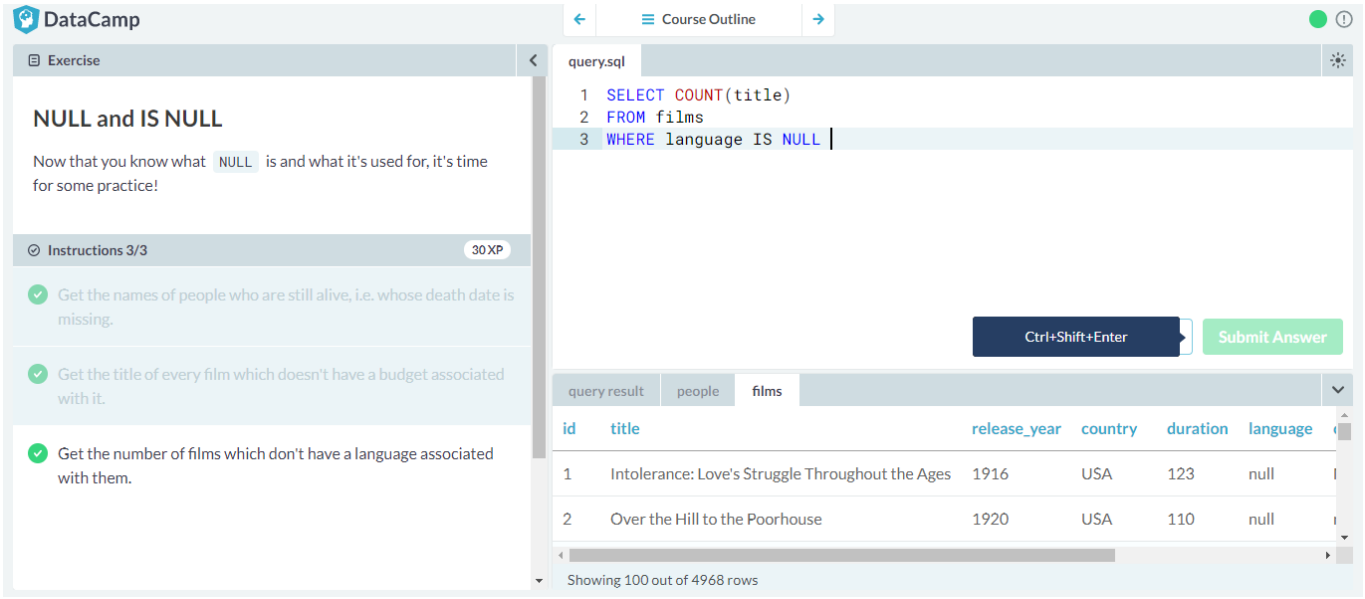

DataCamp
Course Outline
Exercise
SELECT name
FROM kids
WHERE age IN (2, 4, 6, 8, 10);
Try using the IN operator yourself!
Instructions 3/3
30 XP
Get the title and release year of all films released in 1990 or 2000 that were longer than two hours. Remember, duration is in minutes!
Get the title and language of all films which were in English, Spanish, or French.
Get the title and certification of all films with an NC-17 or R certification.
query.sql
1 SELECT title, certification
2 FROM films
3 WHERE certification IN ('NC-17','R')
Run Code
Submit Answer
query result
films
title
language
The Broadway Melody
English
Hell's Angels
English
Showing 100 out of 4747 rows

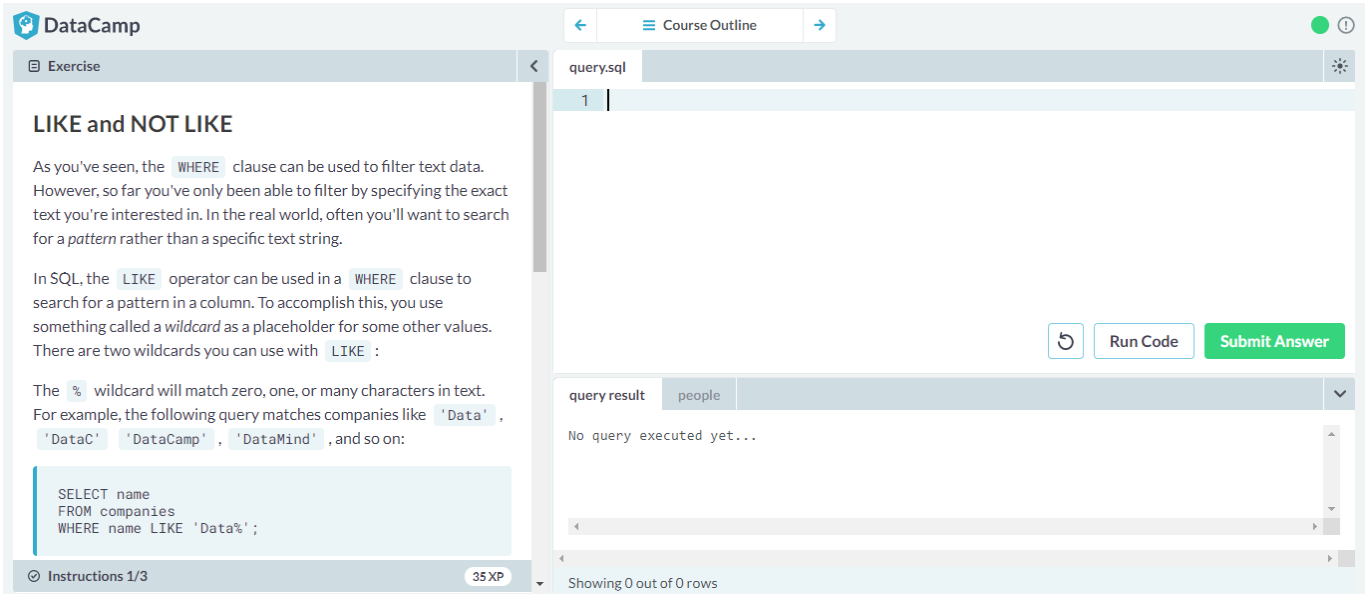

DataCamp
Course Outline
Introduction to NULL and IS NULL
In SQL, NULL represents a missing or unknown value. You can check for NULL values using the expression IS NULL. For example, to count the number of missing birth dates in the people table:
SELECT COUNT(*)
FROM people
WHERE birthdate IS NULL;
As you can see, IS NULL is useful when combined with WHERE to figure out what data you're missing.
Sometimes, you'll want to filter out missing values so you only get results which are not NULL. To do this, you can use the IS NOT NULL operator.
For example, this query gives the names of all people whose birth dates are not missing in the people table.
SELECT name
FROM people
WHERE birthdate IS NOT NULL;

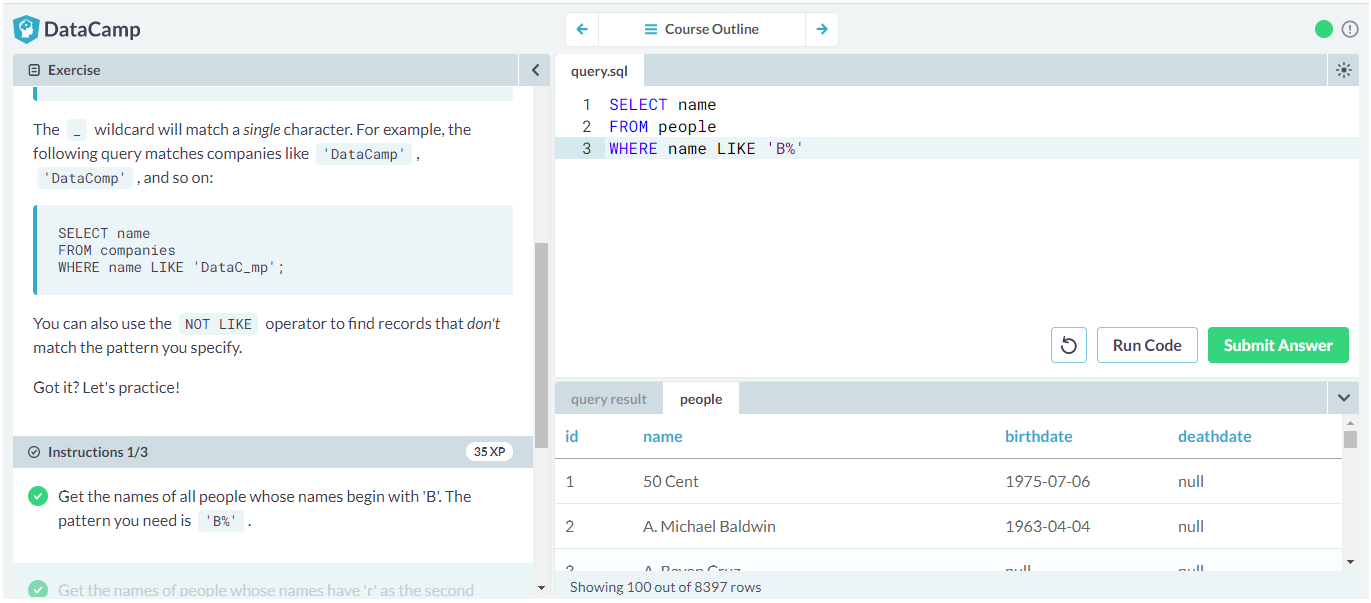

DataCamp
Course Outline
SELECT name
FROM people
WHERE birthdate IS NOT NULL;
What does NULL represent?
Answer the question
50 XP
Possible Answers
A corrupt entry
press 1
A missing value
press 2
An empty string
press 3
An invalid value
press 4
Take Hint (-15xp)
Submit Answer

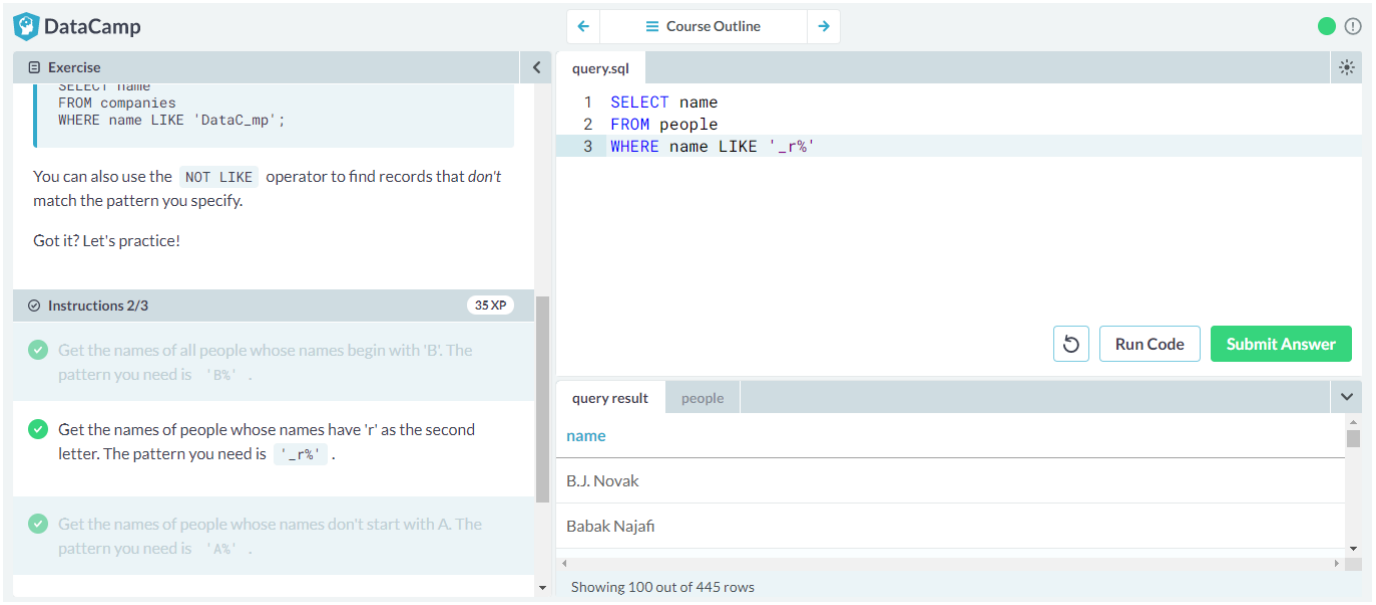

DataCamp
Course Outline
Exercise
NULL and IS NULL
Now that you know what NULL is and what it's used for, it's time for some practice!
Instructions 1/3
35 XP
Get the names of people who are still alive, i.e. whose death date is missing.
Get the title of every film which doesn't have a budget associated with it.
Get the number of films which don't have a language associated with them.
query.sql
1 SELECT name
2 FROM people
3 WHERE deathdate IS NULL
Run Code
Submit Answer
query result
people
films
id
name
birthdate
deathdate
1
50 Cent
1975-07-06
null
2
A. Michael Baldwin
1963-04-04
null
Showing 100 out of 8397 rows

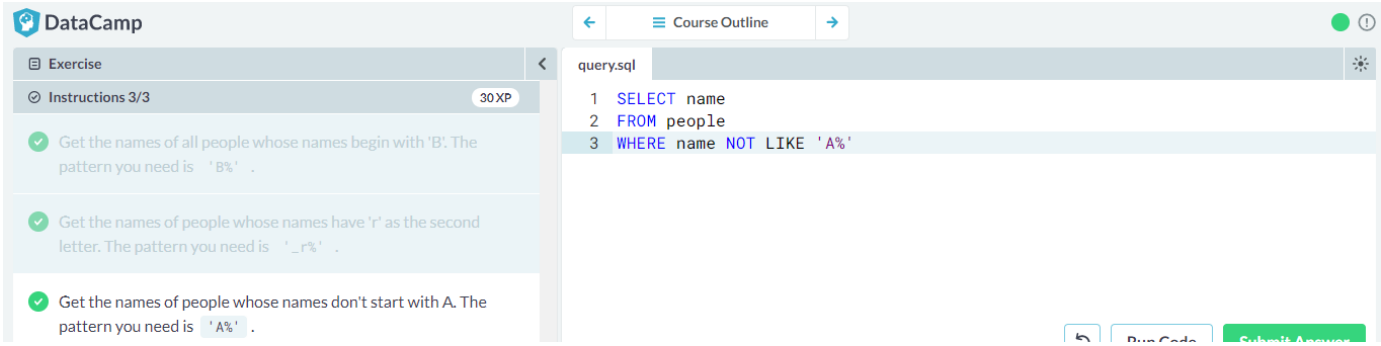

DataCamp
Course Outline
Exercise
NULL and IS NULL
Now that you know what NULL is and what it's used for, it's time for some practice!
Instructions 2/3
35 XP
Get the names of people who are still alive, i.e. whose death date is missing.
Get the title of every film which doesn't have a budget associated with it.
Get the number of films which don't have a language associated with them.
query.sql
1 SELECT title
2 FROM films
3 WHERE budget IS NULL
Run Code
Submit Answer
query result
people
films
name
50 Cent
A. Michael Baldwin
Showing 100 out of 7610 rows

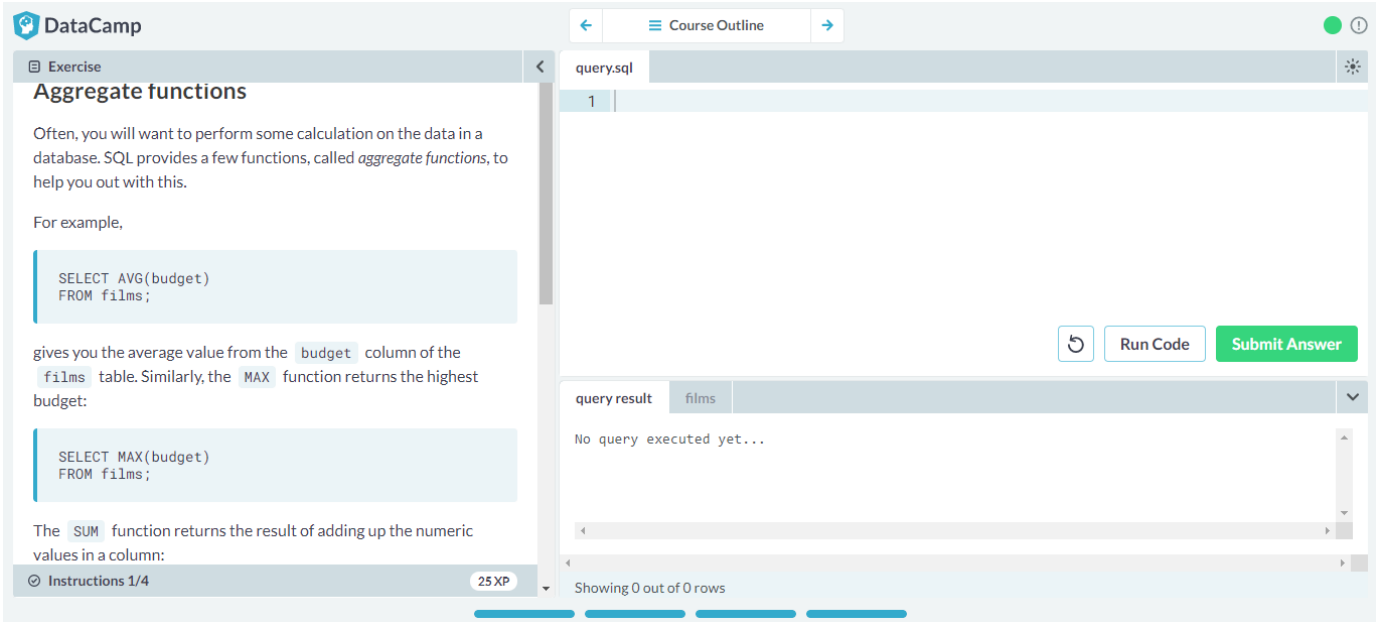

DataCamp
Course Outline
Exercise
NULL and IS NULL
Now that you know what NULL is and what it's used for, it's time for some practice!
Instructions 3/3
30 XP
Get the names of people who are still alive, i.e. whose death date is missing.
Get the title of every film which doesn't have a budget associated with it.
Get the number of films which don't have a language associated with them.
query.sql
1 SELECT COUNT(title)
2 FROM films
3 WHERE language IS NULL
Ctrl+Shift+Enter
Submit Answer
query result
people
films
id
title
release_year
country
duration
language
1
Intolerance: Love's Struggle Throughout the Ages
1916
USA
123
null
2
Over the Hill to the Poorhouse
1920
USA
110
null
Showing 100 out of 4968 rows

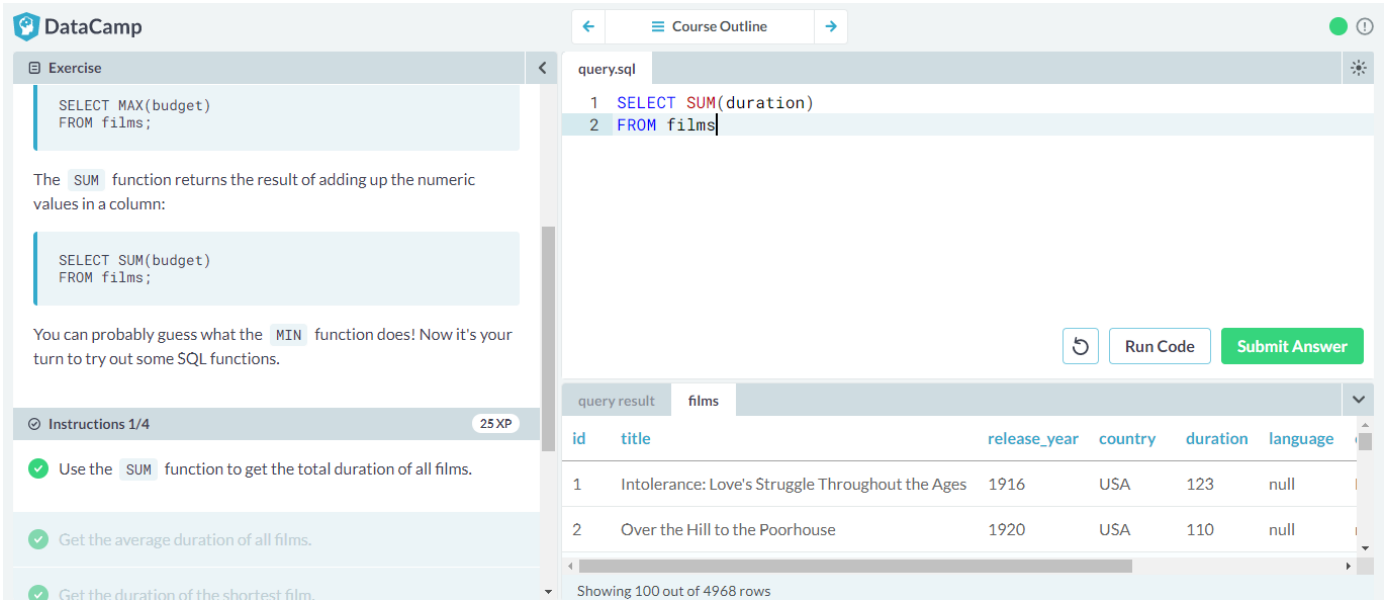

DataCamp
Course Outline
Exercise
LIKE and NOT LIKE
As you've seen, the WHERE clause can be used to filter text data. However, so far you've only been able to filter by specifying the exact text you're interested in. In the real world, often you'll want to search for a pattern rather than a specific text string.
In SQL, the LIKE operator can be used in a WHERE clause to search for a pattern in a column. To accomplish this, you use something called a wildcard as a placeholder for some other values. There are two wildcards you can use with LIKE :
The % wildcard will match zero, one, or many characters in text. For example, the following query matches companies like 'Data' , 'DataC' 'DataCamp' , 'DataMind' , and so on:
SELECT name
FROM companies
WHERE name LIKE 'Data%';
Instructions 1/3
35 XP
query.sql
1
Run Code
Submit Answer
query result
people
No query executed yet...
Showing 0 out of 0 rows

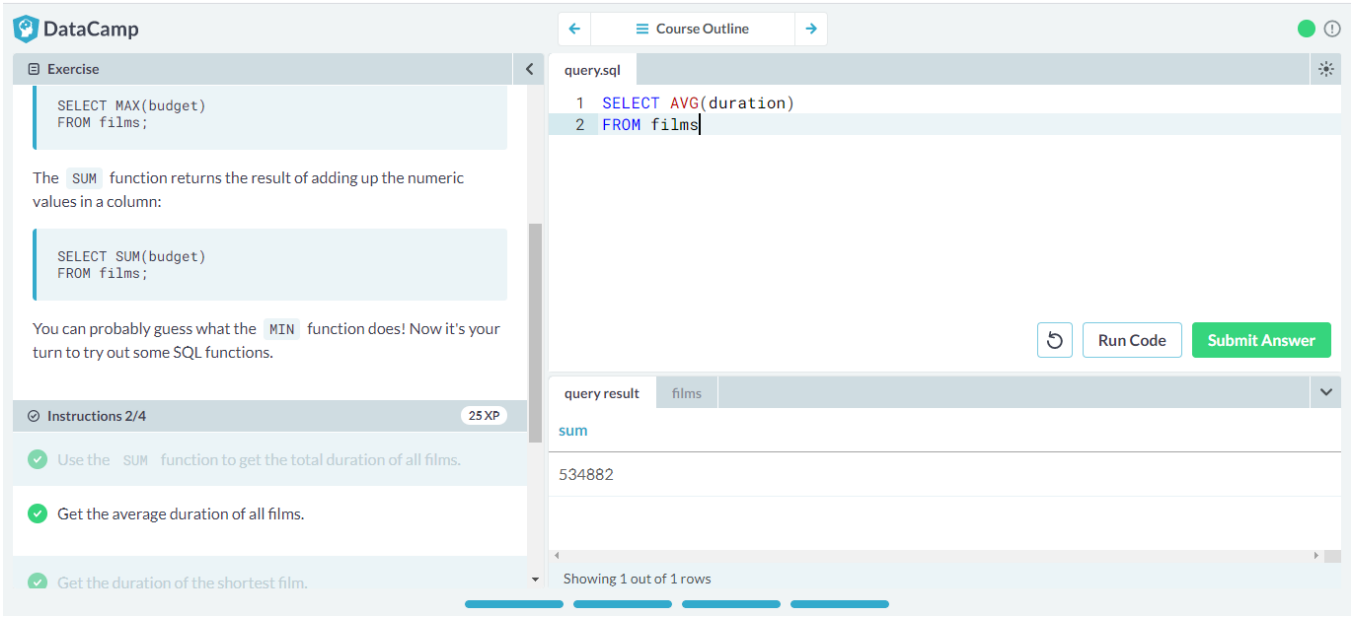

DataCamp
Course Outline
Exercise
The _ wildcard will match a single character. For example, the following query matches companies like 'DataCamp' , 'DataComp' , and so on:
SELECT name
FROM companies
WHERE name LIKE 'DataC_mp';
You can also use the NOT LIKE operator to find records that don't match the pattern you specify.
Got it? Let's practice!
Instructions 1/3
35 XP
Get the names of all people whose names begin with 'B'. The pattern you need is 'B%' .
Get the names of people whose names have 'r' as the second
query.sql
1 SELECT name
2 FROM people
3 WHERE name LIKE 'B%'
Run Code
Submit Answer
query result
people
id
name
birthdate
deathdate
1
50 Cent
1975-07-06
null
2
A. Michael Baldwin
1963-04-04
null
Showing 100 out of 8397 rows

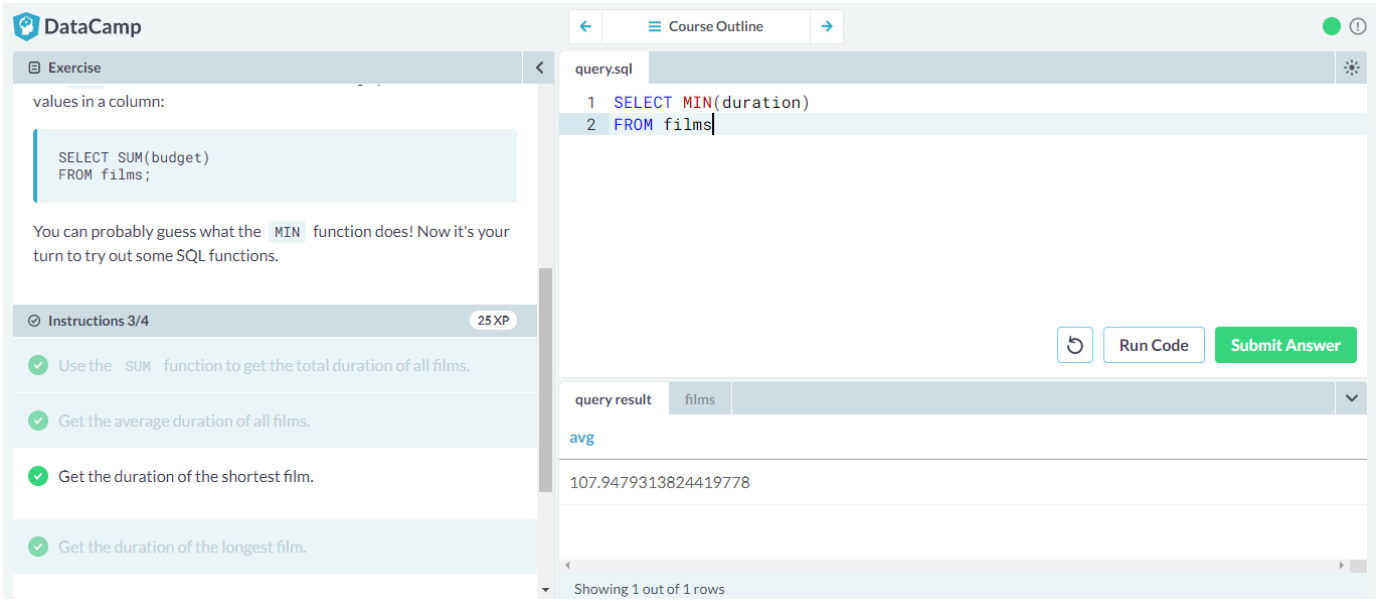

DataCamp
Course Outline
Exercise
FROM companies
WHERE name LIKE 'DataC_mp';
You can also use the NOT LIKE operator to find records that don't match the pattern you specify.
Got it? Let's practice!
Instructions 2/3
35 XP
Get the names of all people whose names begin with 'B'. The pattern you need is 'B%'.
Get the names of people whose names have 'r' as the second letter. The pattern you need is '_r%'.
Get the names of people whose names don't start with A. The pattern you need is 'A%'.
query.sql
1 SELECT name
2 FROM people
3 WHERE name LIKE '_r%'
Run Code
Submit Answer
query result
people
name
B.J. Novak
Babak Najafi
Showing 100 out of 445 rows

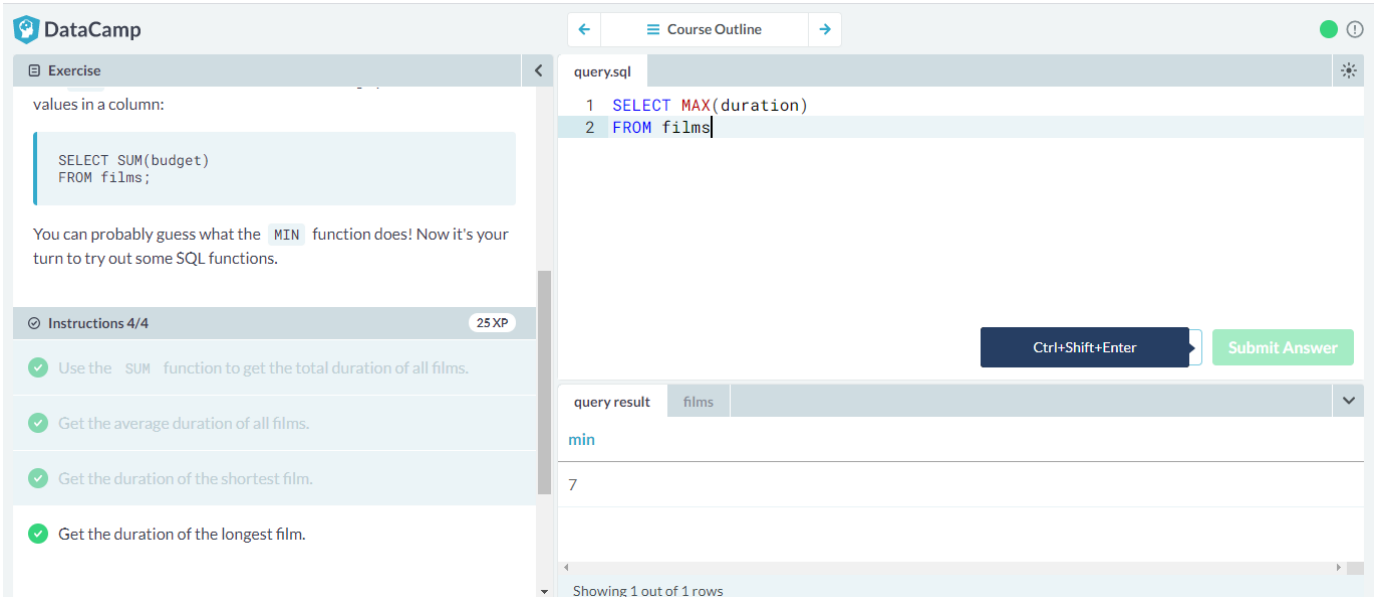

DataCamp
Course Outline
Exercise
Instructions 3/3
30 XP
Get the names of all people whose names begin with 'B'. The pattern you need is 'B%'.
Get the names of people whose names have 'r' as the second letter. The pattern you need is '_r%'.
Get the names of people whose names don't start with A. The pattern you need is 'A%'.
query.sql
1 SELECT name
2 FROM people
3 WHERE name NOT LIKE 'A%'

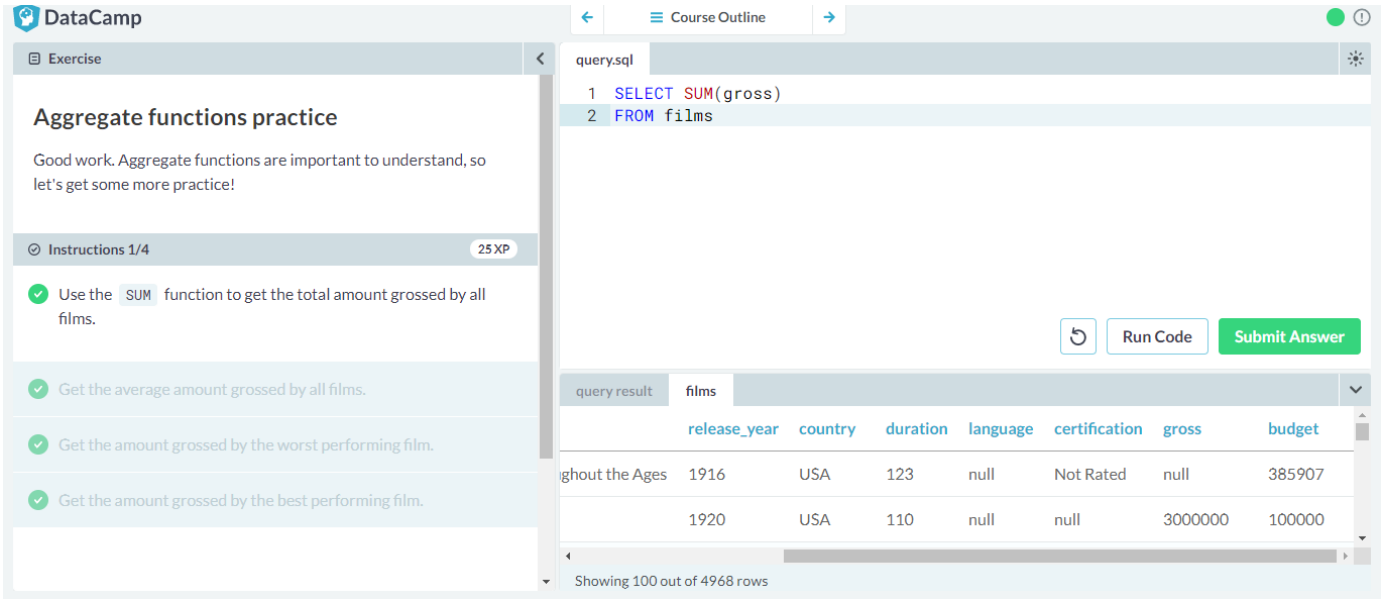

DataCamp
Course Outline
Exercise
Aggregate functions
Often, you will want to perform some calculation on the data in a database. SQL provides a few functions, called aggregate functions, to help you out with this.
For example,
SELECT AVG(budget)
FROM films;
gives you the average value from the budget column of the films table. Similarly, the MAX function returns the highest budget:
SELECT MAX(budget)
FROM films;
The SUM function returns the result of adding up the numeric values in a column:
Instructions 1/4
25 XP
query.sql
1
Run Code
Submit Answer
query result
films
No query executed yet...
Showing 0 out of 0 rows

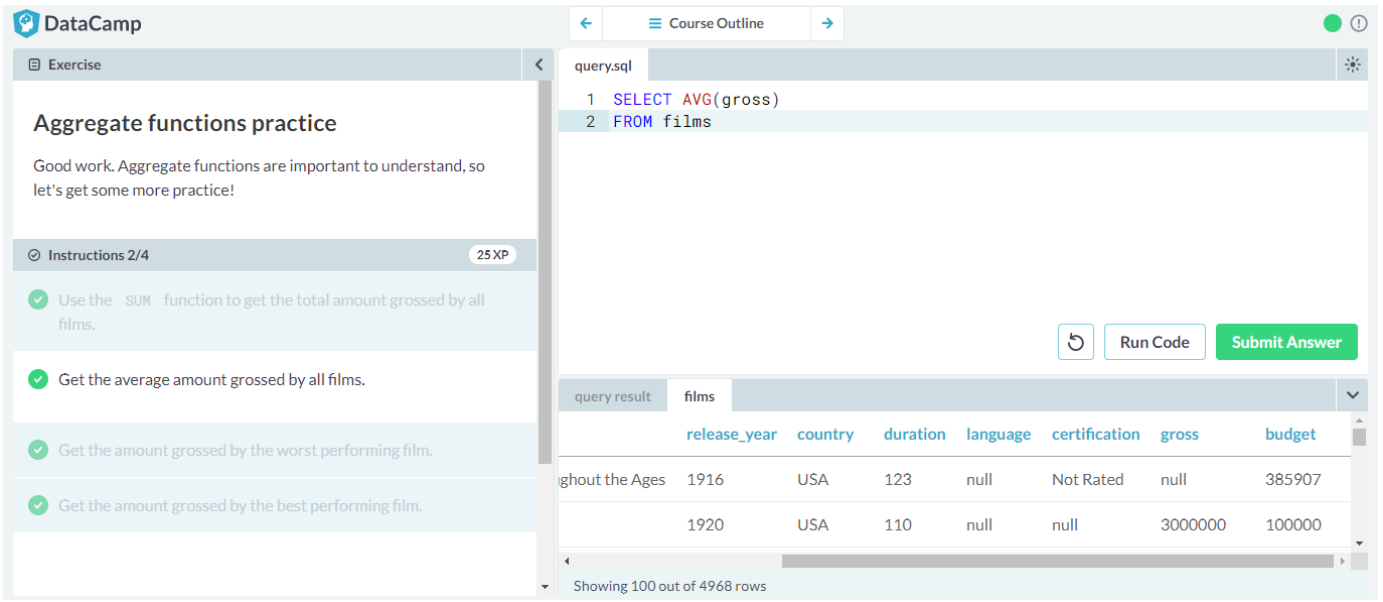

DataCamp
Course Outline
Exercise
SELECT MAX(budget)
FROM films;
The SUM function returns the result of adding up the numeric values in a column:
SELECT SUM(budget)
FROM films;
You can probably guess what the MIN function does! Now it's your turn to try out some SQL functions.
Instructions 1/4
25 XP
Use the SUM function to get the total duration of all films.
Get the average duration of all films.
Get the duration of the shortest film.
query.sql
1 SELECT SUM(duration)
2 FROM films
Run Code
Submit Answer
query result
films
id
title
release_year
country
duration
language
1
Intolerance: Love's Struggle Throughout the Ages
1916
USA
123
null
2
Over the Hill to the Poorhouse
1920
USA
110
null
Showing 100 out of 4968 rows

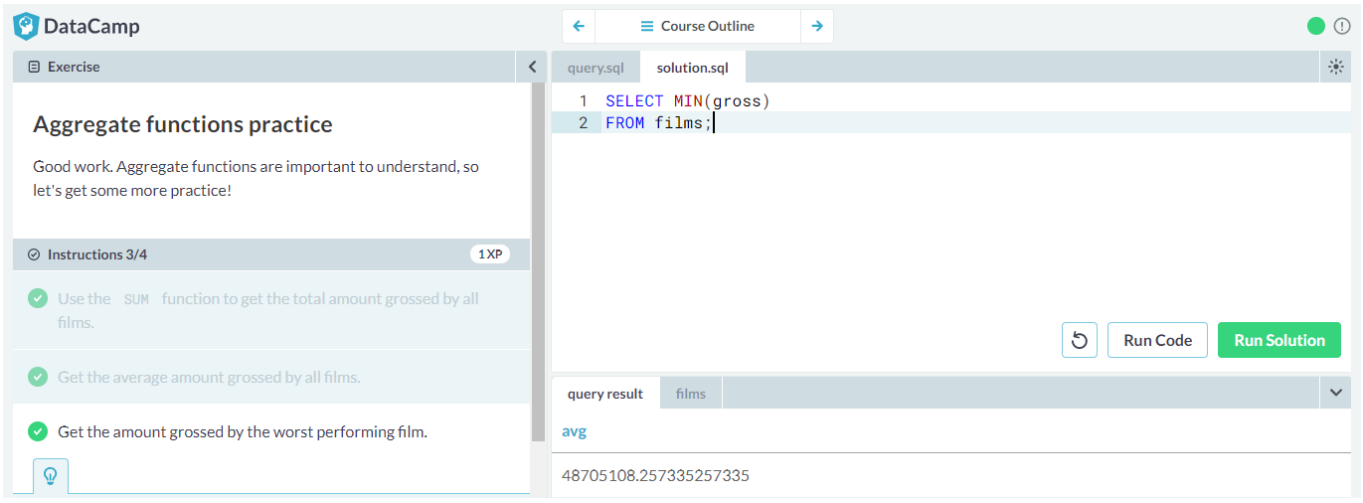

DataCamp
Course Outline
Exercise
SELECT MAX(budget)
FROM films;
The SUM function returns the result of adding up the numeric values in a column:
SELECT SUM(budget)
FROM films;
You can probably guess what the MIN function does! Now it's your turn to try out some SQL functions.
Instructions 2/4
25 XP
Use the SUM function to get the total duration of all films.
Get the average duration of all films.
Get the duration of the shortest film.
query.sql
1 SELECT AVG(duration)
2 FROM films
Run Code
Submit Answer
query result
films
sum
534882
Showing 1 out of 1 rows

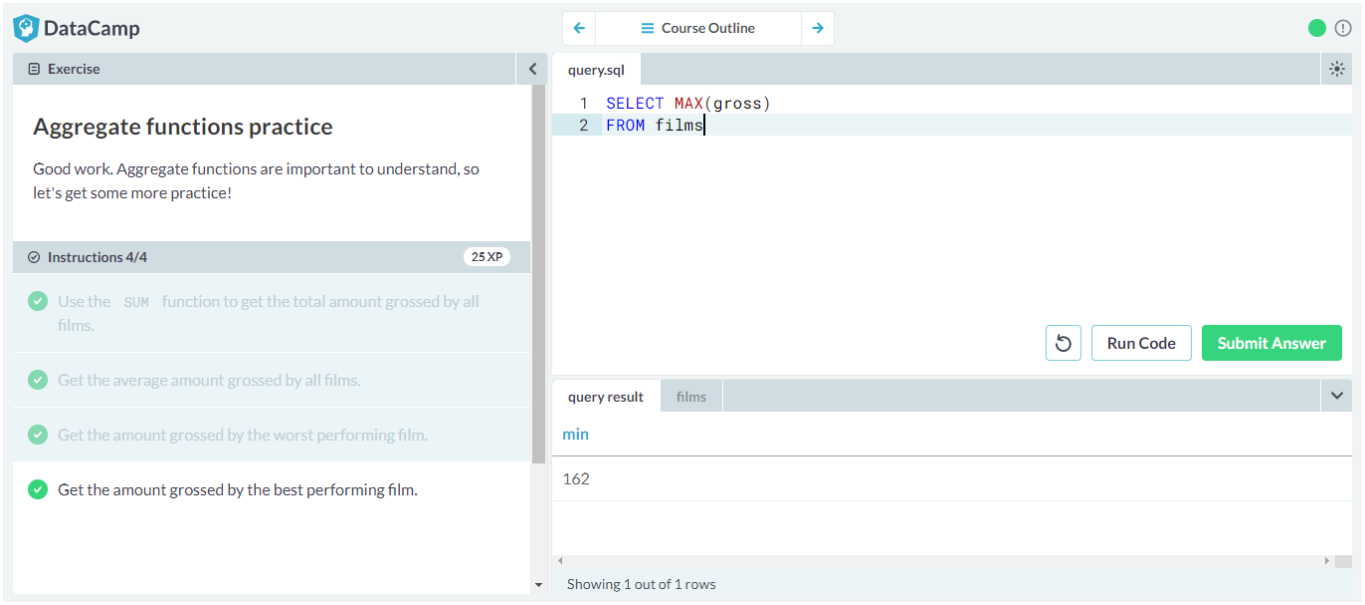

DataCamp
Course Outline
Exercise
values in a column:
SELECT SUM(budget)
FROM films;
You can probably guess what the MIN function does! Now it's your turn to try out some SQL functions.
Instructions 3/4
25 XP
Use the SUM function to get the total duration of all films.
Get the average duration of all films.
Get the duration of the shortest film.
Get the duration of the longest film.
query.sql
1 SELECT MIN(duration)
2 FROM films
Run Code
Submit Answer
query result
films
avg
107.9479313824419778
Showing 1 out of 1 rows

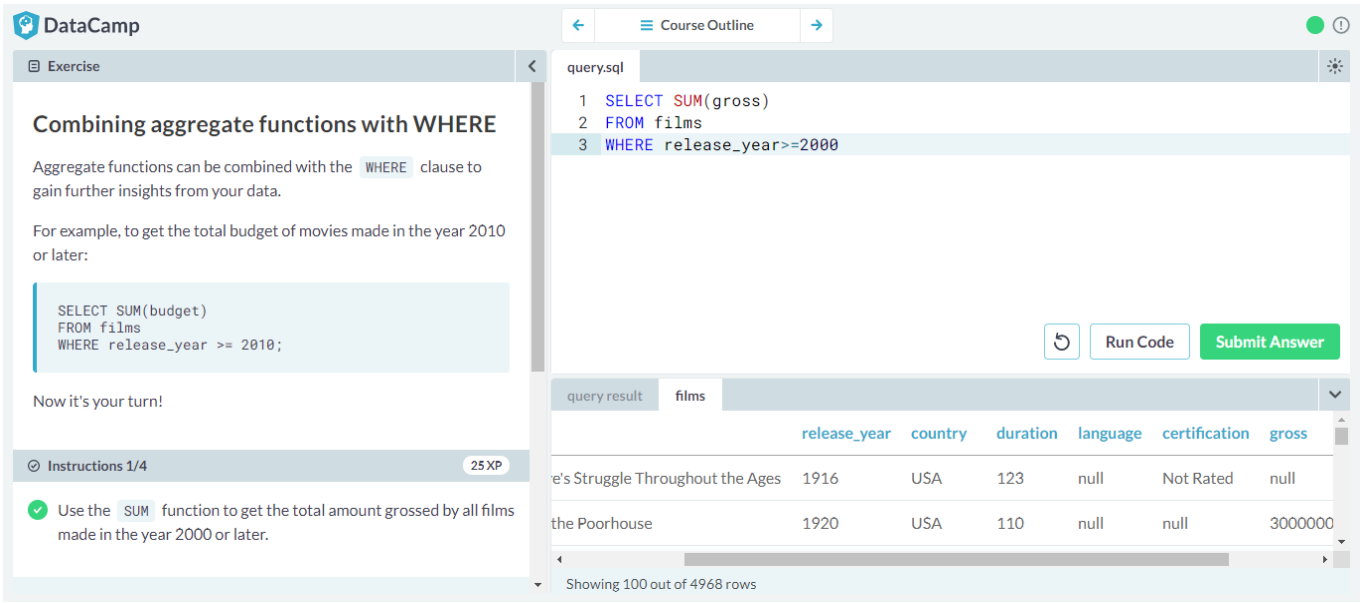

DataCamp
Course Outline
Exercise
values in a column:
SELECT SUM(budget)
FROM films;
You can probably guess what the MIN function does! Now it's your turn to try out some SQL functions.
Instructions 4/4
25 XP
Use the SUM function to get the total duration of all films.
Get the average duration of all films.
Get the duration of the shortest film.
Get the duration of the longest film.
query.sql
1 SELECT MAX(duration)
2 FROM films
Ctrl+Shift+Enter
Submit Answer
query result
films
min
7
Showing 1 out of 1 rows

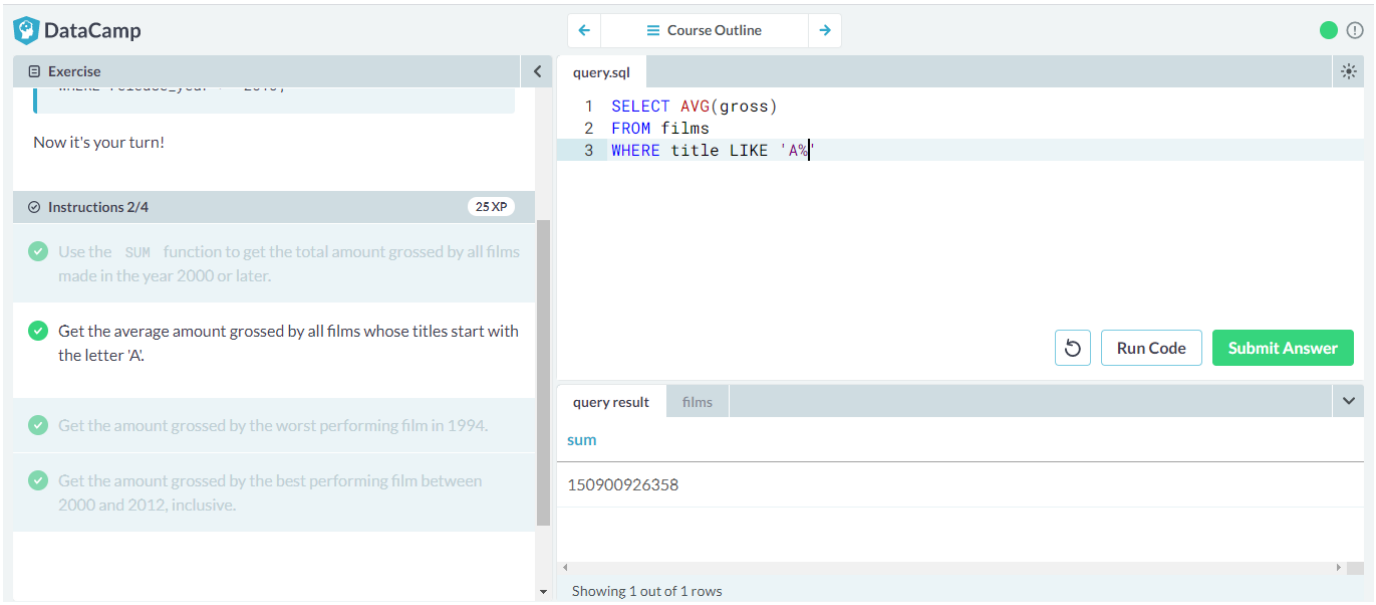

DataCamp
Course Outline
Exercise
Aggregate functions practice
Good work. Aggregate functions are important to understand, so let's get some more practice!
Instructions 1/4
25 XP
Use the SUM function to get the total amount grossed by all films.
Get the average amount grossed by all films.
Get the amount grossed by the worst performing film.
Get the amount grossed by the best performing film.
query.sql
1 SELECT SUM(gross)
2 FROM films
Run Code
Submit Answer
query result
films
release_year country duration language certification gross budget
ghout the Ages 1916 USA 123 null Not Rated null 385907
1920 USA 110 null null 3000000 100000
Showing 100 out of 4968 rows

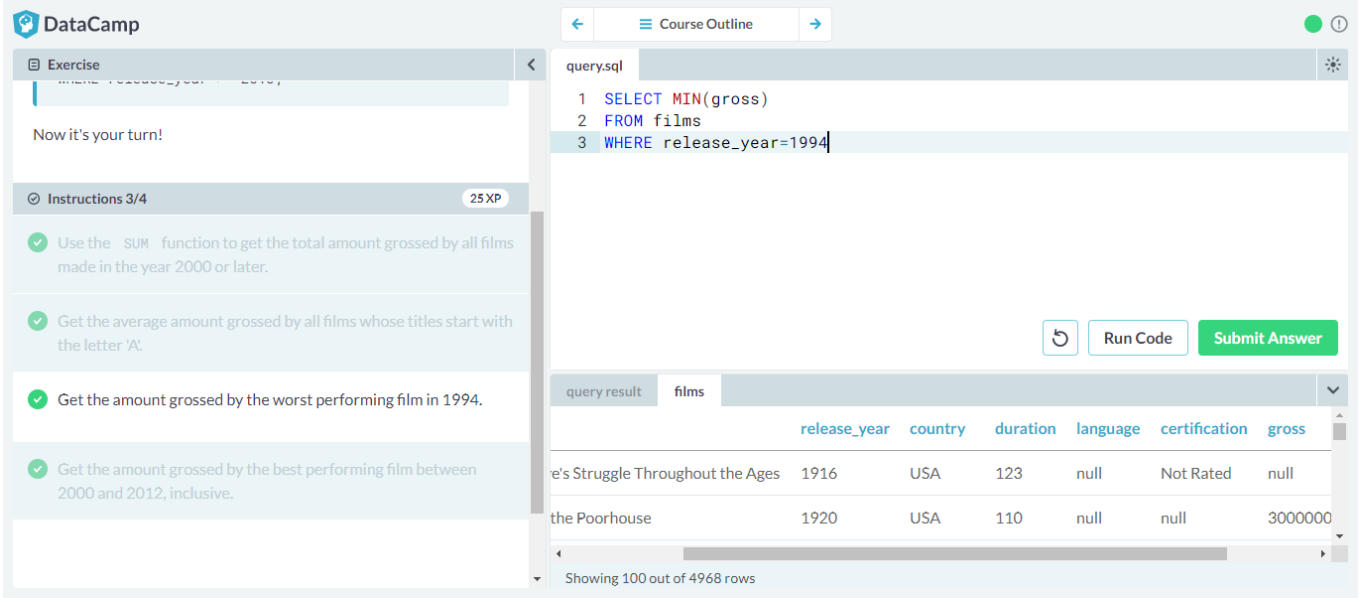

DataCamp
Course Outline
Exercise
Aggregate functions practice
Good work. Aggregate functions are important to understand, so let's get some more practice!
Instructions 2/4
25 XP
Use the SUM function to get the total amount grossed by all films.
Get the average amount grossed by all films.
Get the amount grossed by the worst performing film.
Get the amount grossed by the best performing film.
query.sql
1 SELECT AVG(gross)
2 FROM films
Run Code
Submit Answer
query result
films
release_year country duration language certification gross budget
ghout the Ages 1916 USA 123 null Not Rated null 385907
1920 USA 110 null null 3000000 100000
Showing 100 out of 4968 rows

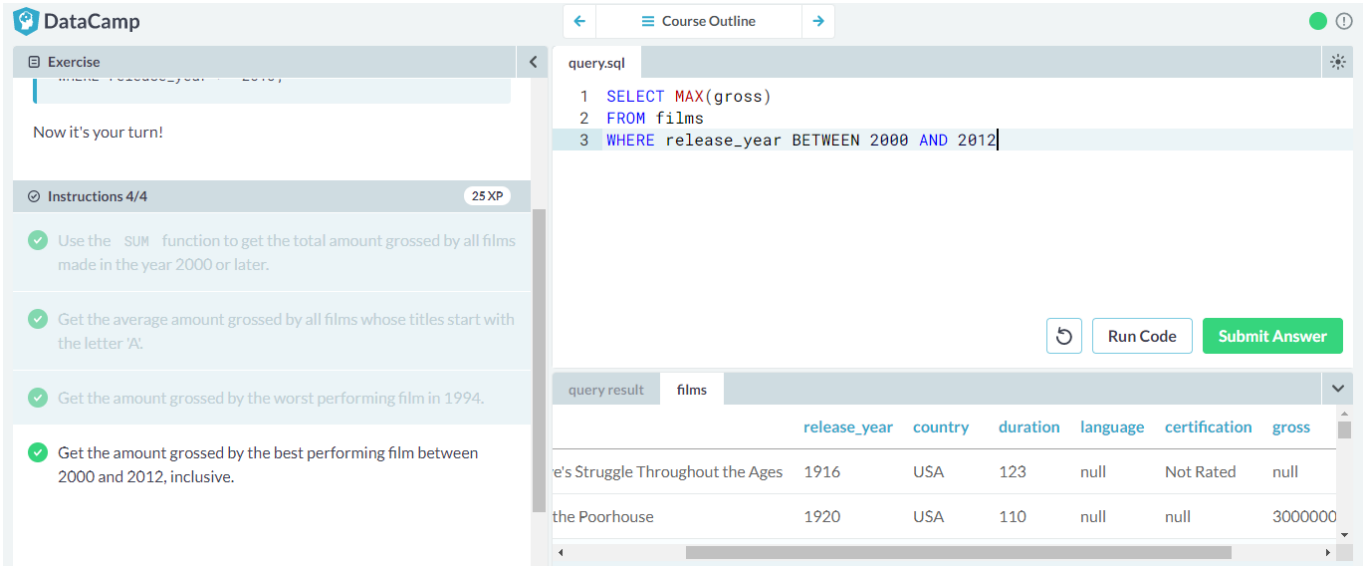

DataCamp
Course Outline
Exercise
Aggregate functions practice
Good work. Aggregate functions are important to understand, so let's get some more practice!
Instructions 3/4
1 XP
Use the SUM function to get the total amount grossed by all films.
Get the average amount grossed by all films.
Get the amount grossed by the worst performing film.
query.sql
solution.sql
1 SELECT MIN(gross)
2 FROM films;
Run Code
Run Solution
query result
films
avg
48705108.257335257335

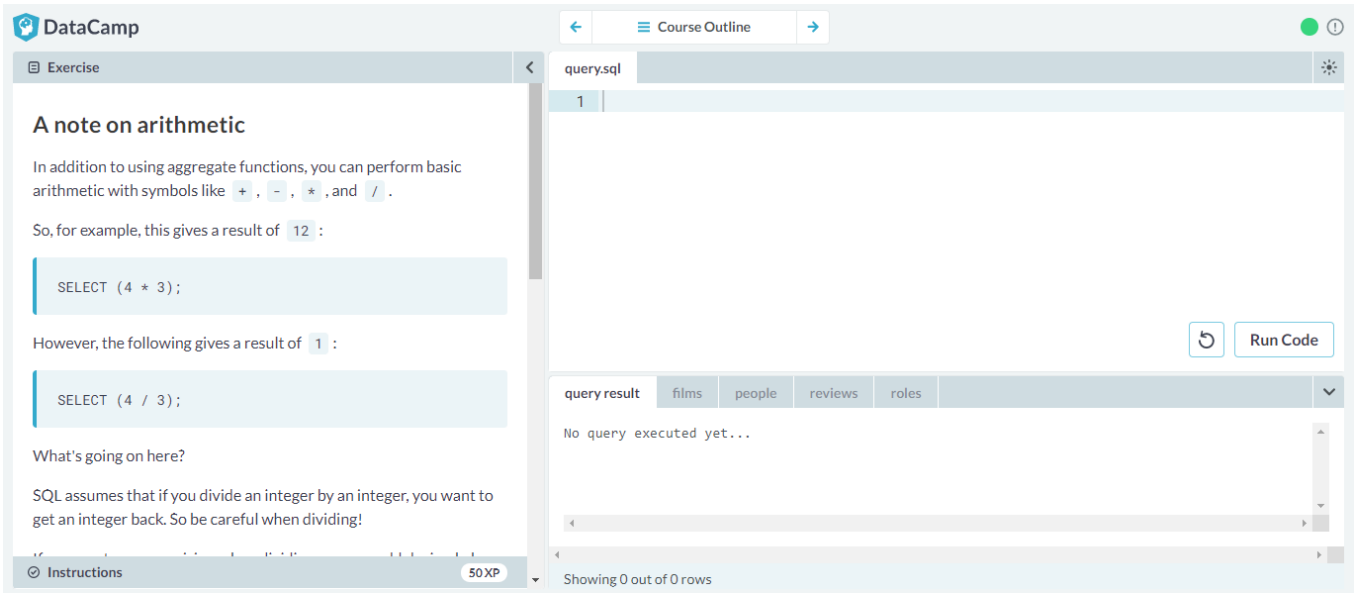

DataCamp
Course Outline
Exercise
Aggregate functions practice
Good work. Aggregate functions are important to understand, so let's get some more practice!
Instructions 4/4
25 XP
Use the SUM function to get the total amount grossed by all films.
Get the average amount grossed by all films.
Get the amount grossed by the worst performing film.
Get the amount grossed by the best performing film.
query.sql
1 SELECT MAX(gross)
2 FROM films
Run Code
Submit Answer
query result
films
min
162
Showing 1 out of 1 rows

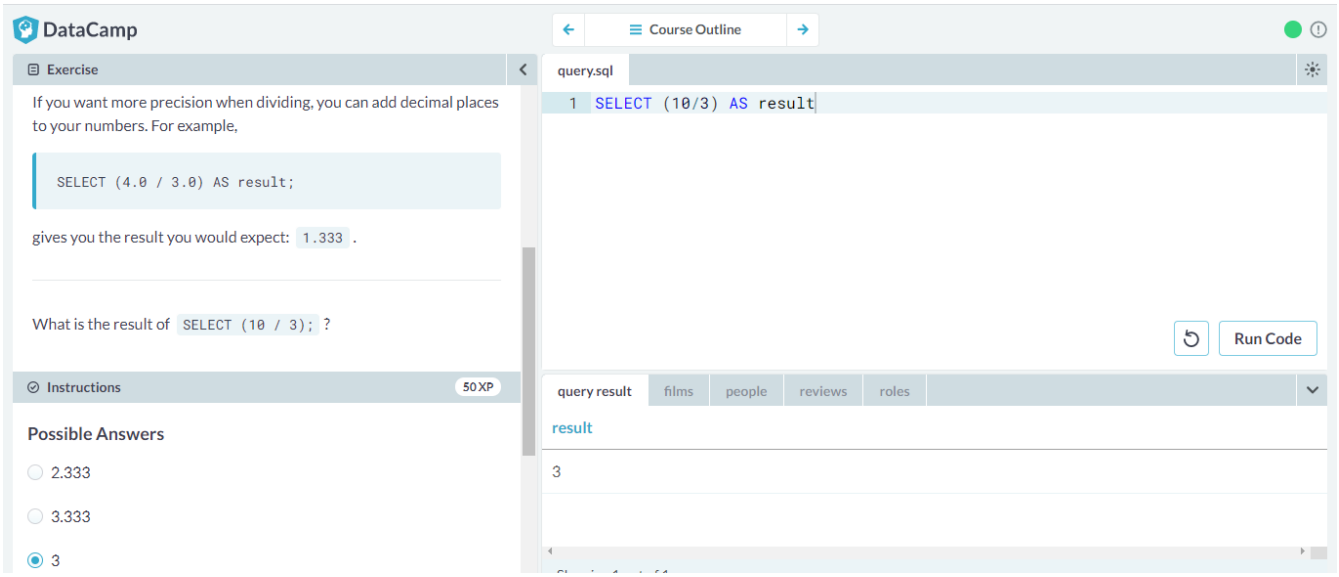

DataCamp
Course Outline
Exercise
Combining aggregate functions with WHERE
Aggregate functions can be combined with the WHERE clause to gain further insights from your data.
For example, to get the total budget of movies made in the year 2010 or later:
SELECT SUM(budget)
FROM films
WHERE release_year >= 2010;
Now it's your turn!
Instructions 1/4
25 XP
Use the SUM function to get the total amount grossed by all films made in the year 2000 or later.
query.sql
1 SELECT SUM(gross)
2 FROM films
3 WHERE release_year>=2000
Run Code
Submit Answer
query result
films
release_year country duration language certification gross
e's Struggle Throughout the Ages 1916 USA 123 null Not Rated null
the Poorhouse 1920 USA 110 null null 3000000
Showing 100 out of 4968 rows


DataCamp
Course Outline
Exercise
Now it's your turn!
Instructions 2/4
25 XP
Use the SUM function to get the total amount grossed by all films made in the year 2000 or later.
Get the average amount grossed by all films whose titles start with the letter 'A'.
Get the amount grossed by the worst performing film in 1994.
Get the amount grossed by the best performing film between 2000 and 2012, inclusive.
query.sql
1 SELECT AVG(gross)
2 FROM films
3 WHERE title LIKE 'A%'
Run Code
Submit Answer
query result
films
sum
150900926358
Showing 1 out of 1 rows

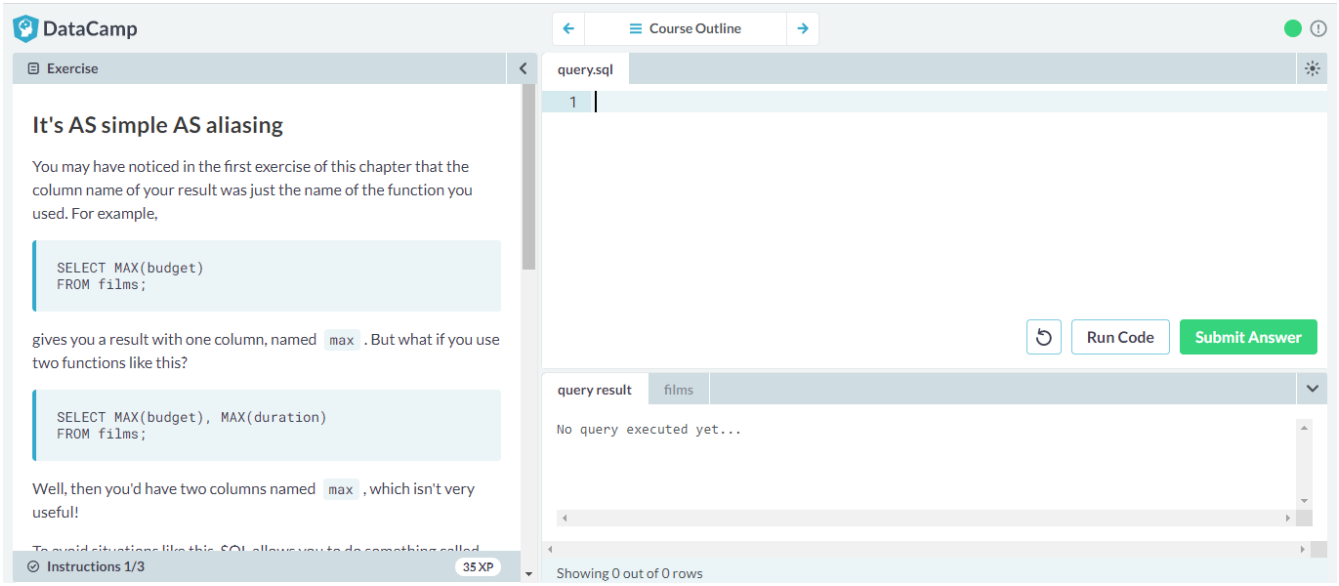

DataCamp
Course Outline
Exercise
Now it's your turn!
Instructions 3/4
25 XP
Use the SUM function to get the total amount grossed by all films made in the year 2000 or later.
Get the average amount grossed by all films whose titles start with the letter 'A'.
Get the amount grossed by the worst performing film in 1994.
Get the amount grossed by the best performing film between 2000 and 2012, inclusive.
query.sql
1 SELECT MIN(gross)
2 FROM films
3 WHERE release_year=1994
Run Code
Submit Answer
query result
films
release_year country duration language certification gross
e's Struggle Throughout the Ages 1916 USA 123 null Not Rated null
the Poorhouse 1920 USA 110 null null 3000000
Showing 100 out of 4968 rows

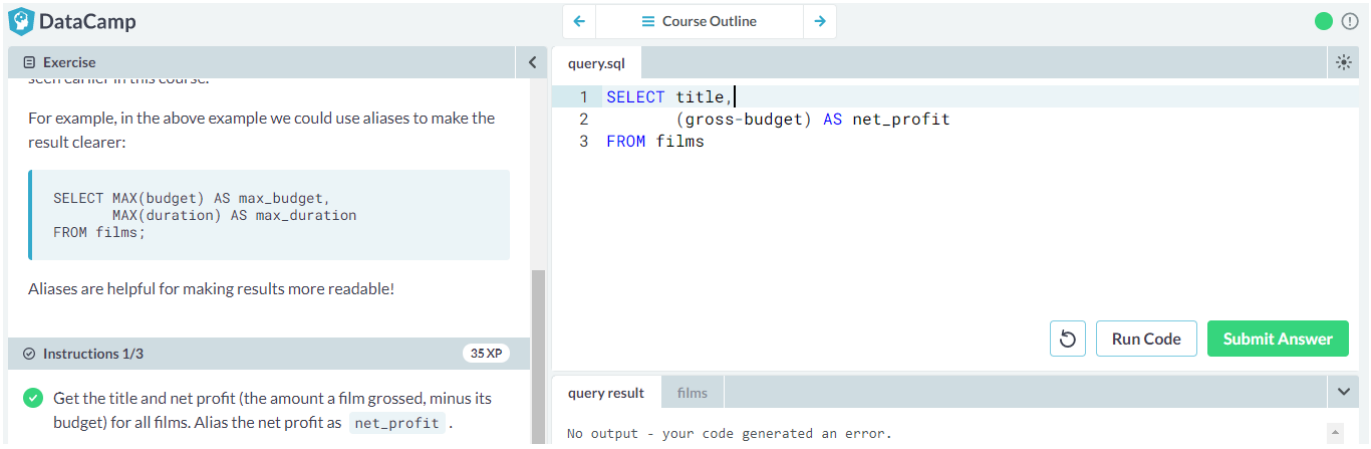

DataCamp
Course Outline
Exercise
Now it's your turn!
Instructions 4/4
25 XP
Use the SUM function to get the total amount grossed by all films made in the year 2000 or later.
Get the average amount grossed by all films whose titles start with the letter 'A'.
Get the amount grossed by the worst performing film in 1994.
Get the amount grossed by the best performing film between 2000 and 2012, inclusive.
query.sql
1 SELECT MAX(gross)
2 FROM films
3 WHERE release_year BETWEEN 2000 AND 2012
Run Code
Submit Answer
query result
films
release_year country duration language certification gross
e's Struggle Throughout the Ages 1916 USA 123 null Not Rated null
the Poorhouse 1920 USA 110 null null 3000000

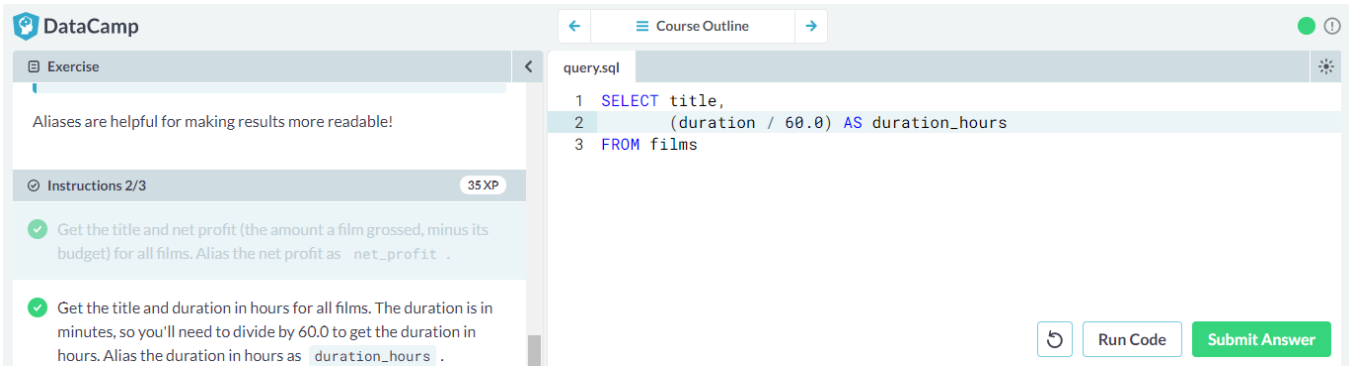

DataCamp
Course Outline
Exercise
A note on arithmetic
In addition to using aggregate functions, you can perform basic arithmetic with symbols like + , - , * , and / .
So, for example, this gives a result of 12 :
SELECT (4 * 3);
However, the following gives a result of 1 :
SELECT (4 / 3);
What's going on here?
SQL assumes that if you divide an integer by an integer, you want to get an integer back. So be careful when dividing!
Instructions
50 XP
query.sql
1
Run Code
query result
films
people
reviews
roles
No query executed yet...
Showing 0 out of 0 rows

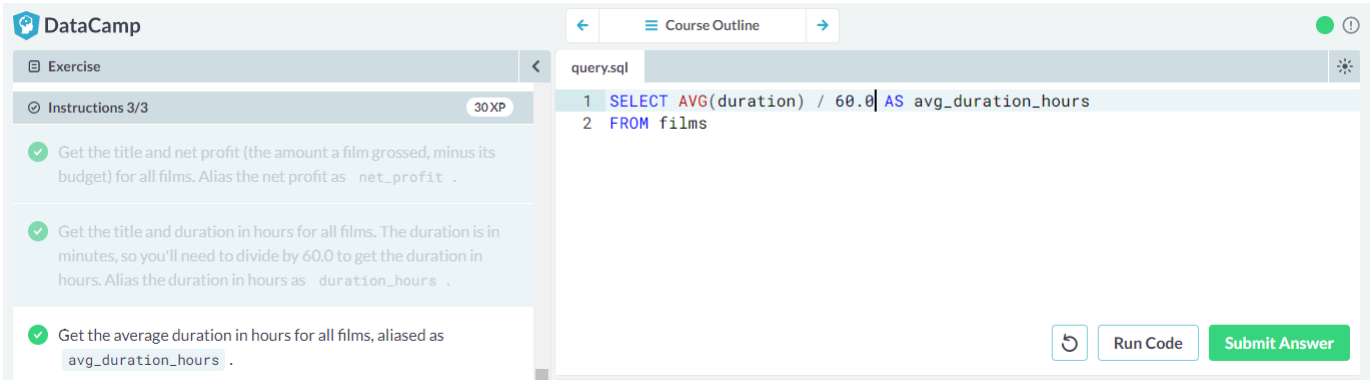


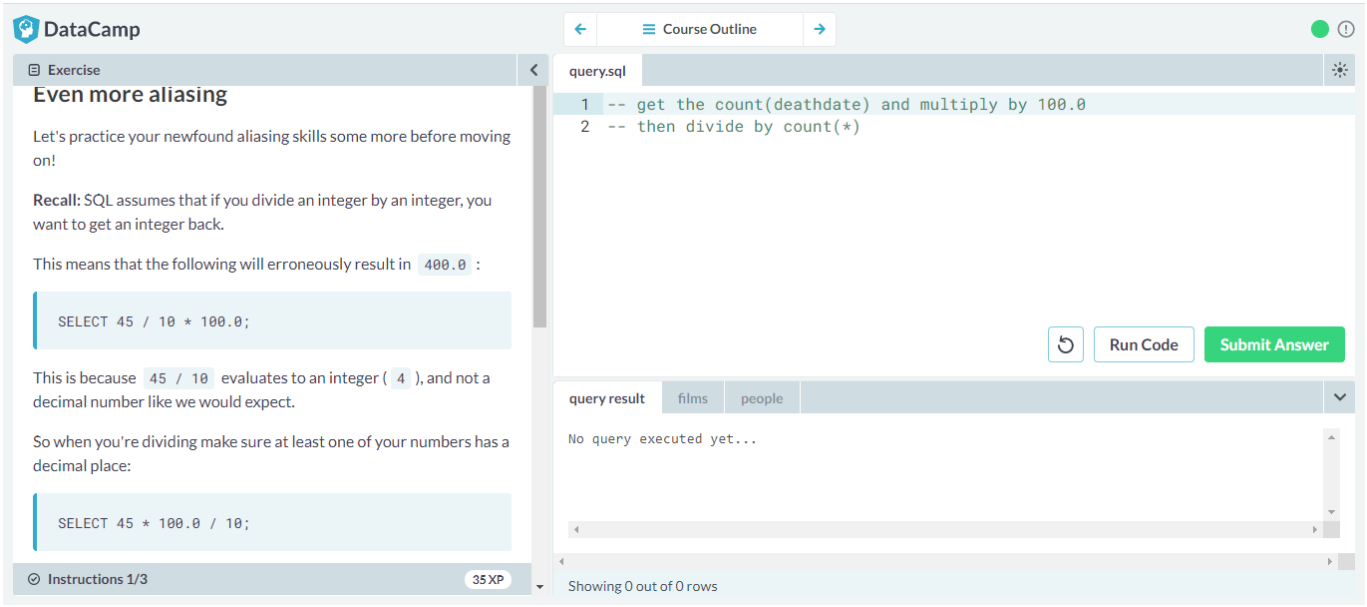
Es normal este resultado, porque SQL entiende que al divider un entero por otro entero debe dar un entero. Pero si queremos que de un resultado decimal hay que dividir 10.0/3. No se asusten ahaha

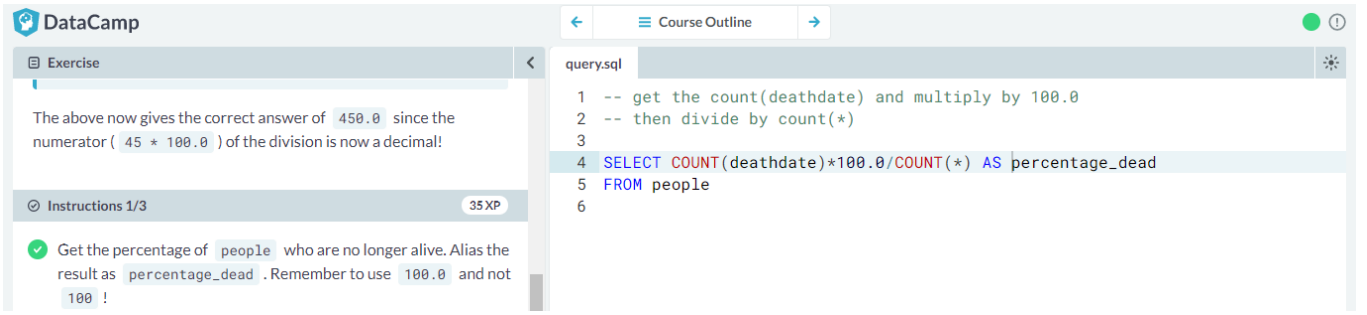


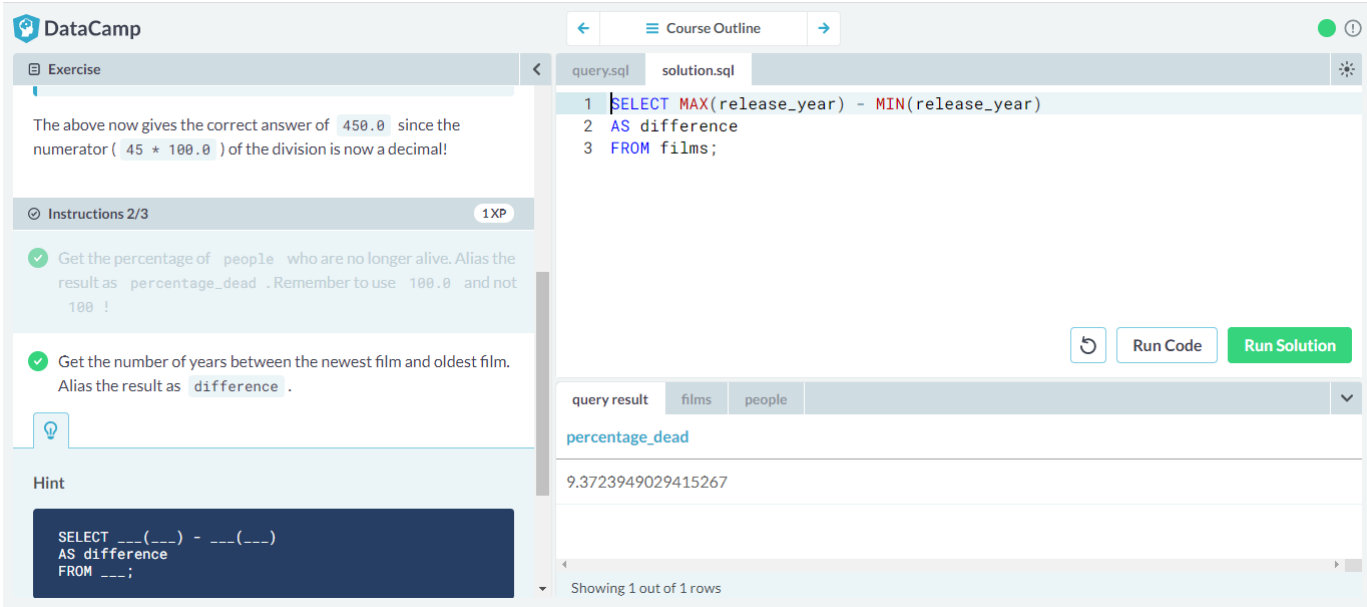

DataCamp
Course Outline
Exercise
For example, in the above example we could use aliases to make the result clearer:
SELECT MAX(budget) AS max_budget,
MAX(duration) AS max_duration
FROM films;
Aliases are helpful for making results more readable!
Instructions 1/3
35 XP
Get the title and net profit (the amount a film grossed, minus its budget) for all films. Alias the net profit as net_profit .
query.sql
1 SELECT title,
2 (gross-budget) AS net_profit
3 FROM films
Run Code
Submit Answer
query result
films
No output - your code generated an error.

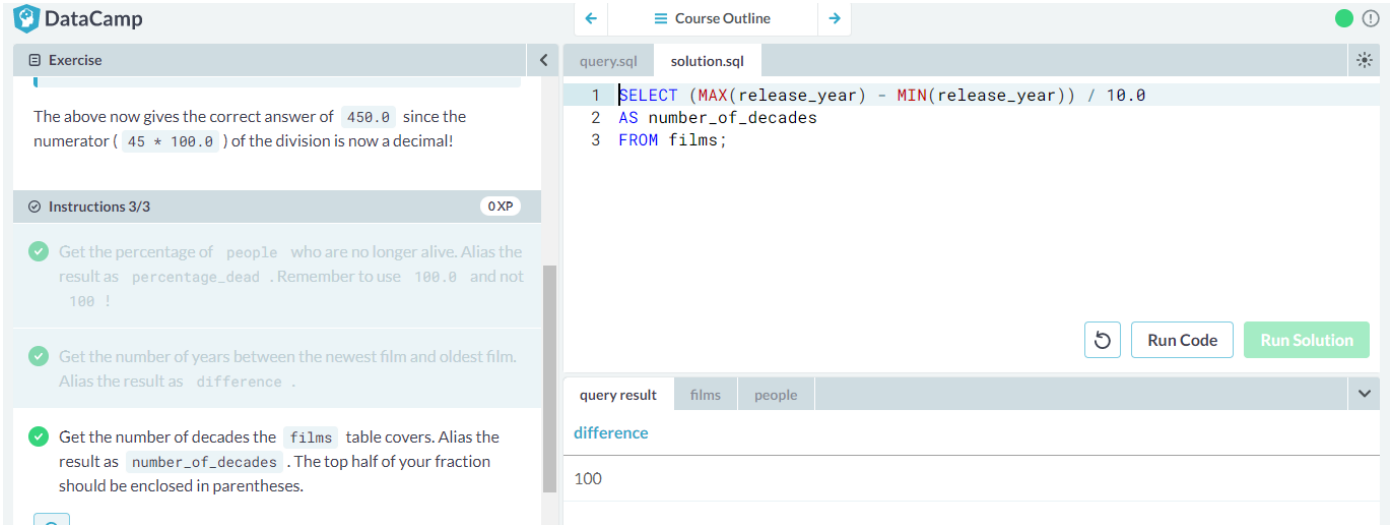

DataCamp
Course Outline
Exercise
Aliases are helpful for making results more readable!
Instructions 2/3
35 XP
Get the title and net profit (the amount a film grossed, minus its budget) for all films. Alias the net profit as net_profit .
Get the title and duration in hours for all films. The duration is in minutes, so you'll need to divide by 60.0 to get the duration in hours. Alias the duration in hours as duration_hours .
query.sql
1 SELECT title,
2 (duration / 60.0) AS duration_hours
3 FROM films
Run Code
Submit Answer

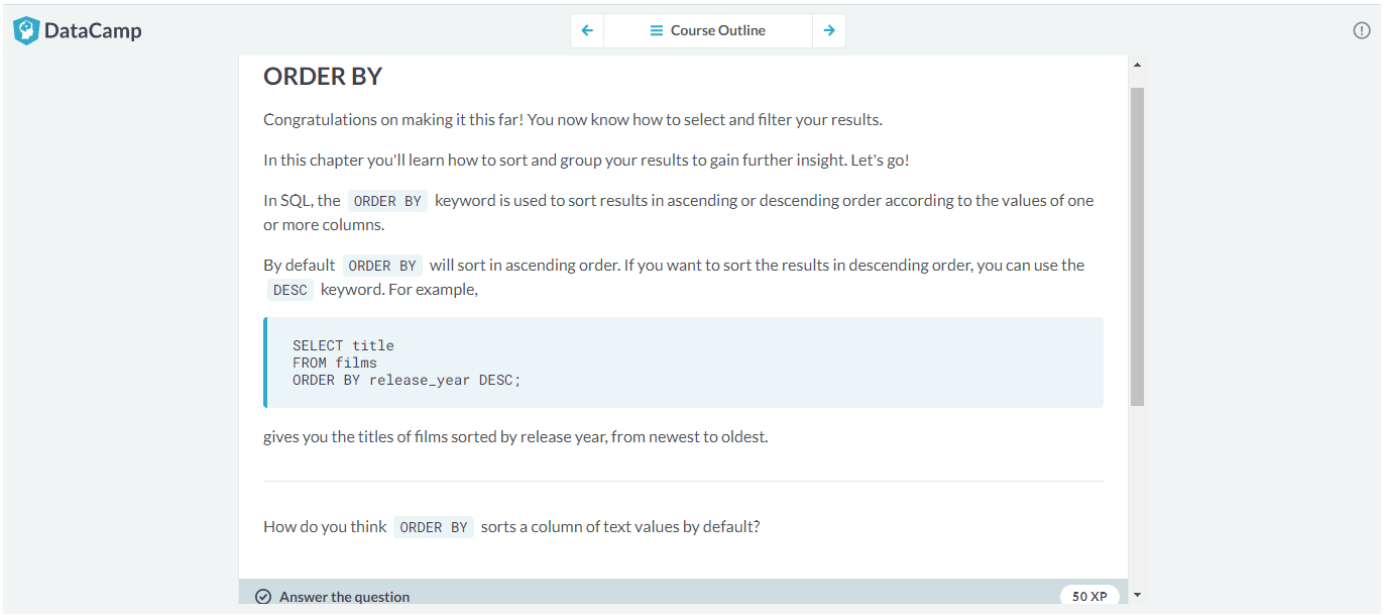

DataCamp
Course Outline
Exercise
Instructions 3/3
30 XP
Get the title and net profit (the amount a film grossed, minus its budget) for all films. Alias the net profit as net_profit .
Get the title and duration in hours for all films. The duration is in minutes, so you'll need to divide by 60.0 to get the duration in hours. Alias the duration in hours as duration_hours .
Get the average duration in hours for all films, aliased as avg_duration_hours .
query.sql
1 SELECT AVG(duration) / 60.0 AS avg_duration_hours
2 FROM films
Run Code
Submit Answer

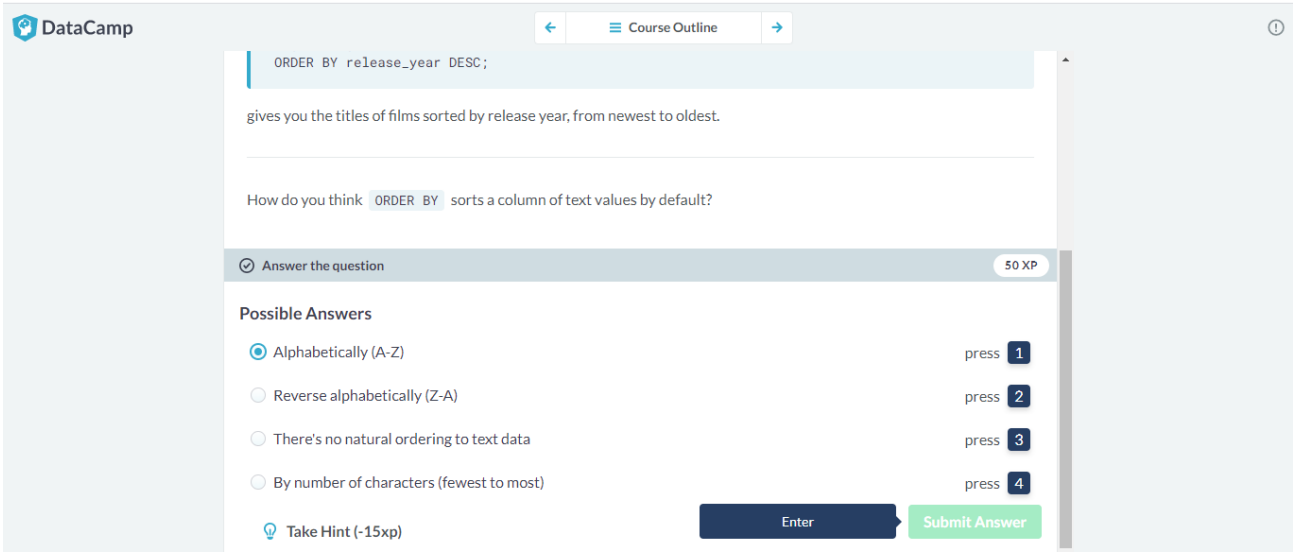

DataCamp
Course Outline
Exercise
Even more aliasing
Let's practice your newfound aliasing skills some more before moving on!
Recall: SQL assumes that if you divide an integer by an integer, you want to get an integer back.
This means that the following will erroneously result in 400.0 :
SELECT 45 / 10 * 100.0;
This is because 45 / 10 evaluates to an integer (4), and not a decimal number like we would expect.
So when you're dividing make sure at least one of your numbers has a decimal place:
SELECT 45 * 100.0 / 10;
Instructions 1/3
35 XP
query.sql
1 -- get the count(deathdate) and multiply by 100.0
2 -- then divide by count(*)
Run Code
Submit Answer
query result
films
people
No query executed yet...
Showing 0 out of 0 rows

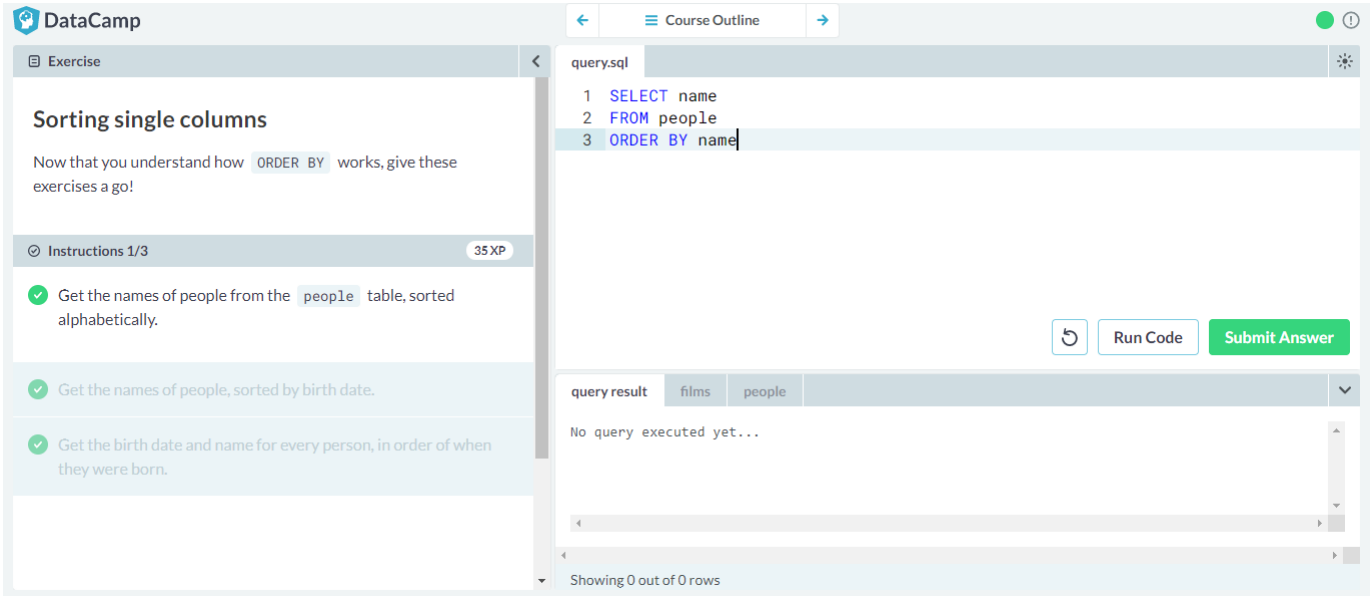

DataCamp
Course Outline
Exercise
The above now gives the correct answer of 450.0 since the numerator (45 * 100.0) of the division is now a decimal!
Instructions 1/3
35 XP
Get the percentage of people who are no longer alive. Alias the result as percentage_dead . Remember to use 100.0 and not 100 !
query.sql
1 -- get the count(deathdate) and multiply by 100.0
2 -- then divide by count(*)
3
4 SELECT COUNT(deathdate)*100.0/COUNT(*) AS percentage_dead
5 FROM people
6

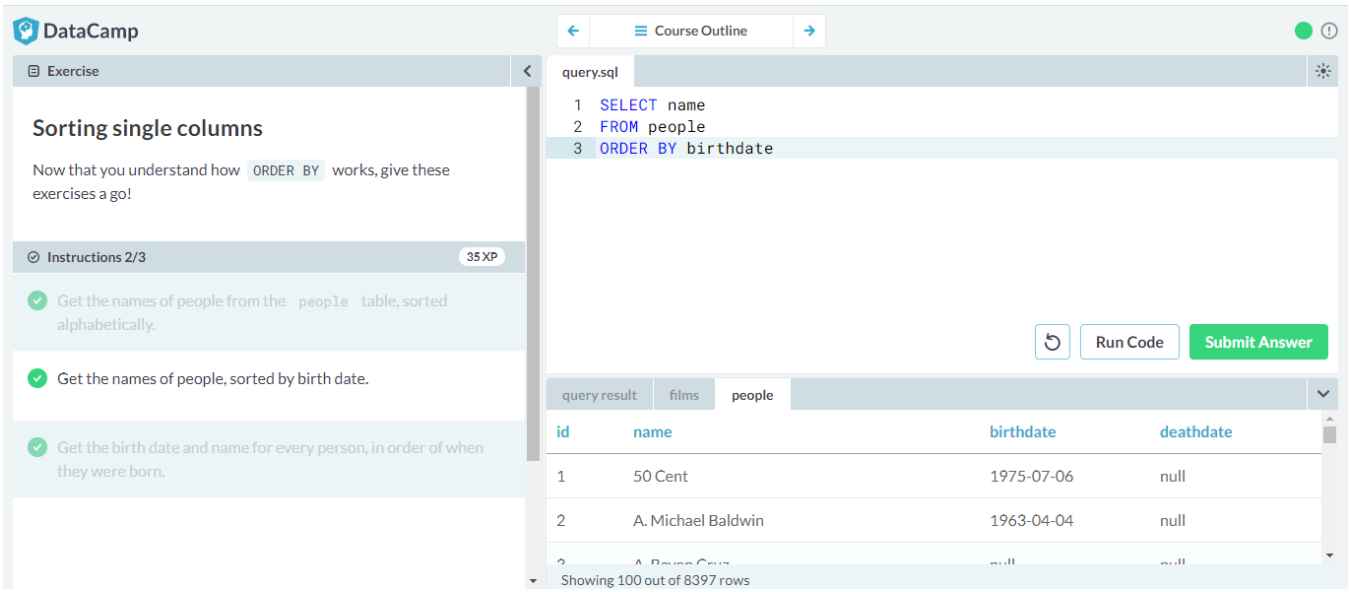

DataCamp
Course Outline
Exercise
The above now gives the correct answer of 450.0 since the numerator (45 * 100.0) of the division is now a decimal!
Instructions 2/3
1 XP
Get the percentage of people who are no longer alive. Alias the result as percentage_dead . Remember to use 100.0 and not 100 !
Get the number of years between the newest film and oldest film. Alias the result as difference .
Hint
SELECT ___(___) - ___(___)
AS difference
FROM ___;
query.sql
solution.sql
1 SELECT MAX(release_year) - MIN(release_year)
2 AS difference
3 FROM films;
Run Code
Run Solution
query result
films
people
percentage_dead
9.3723949029415267
Showing 1 out of 1 rows

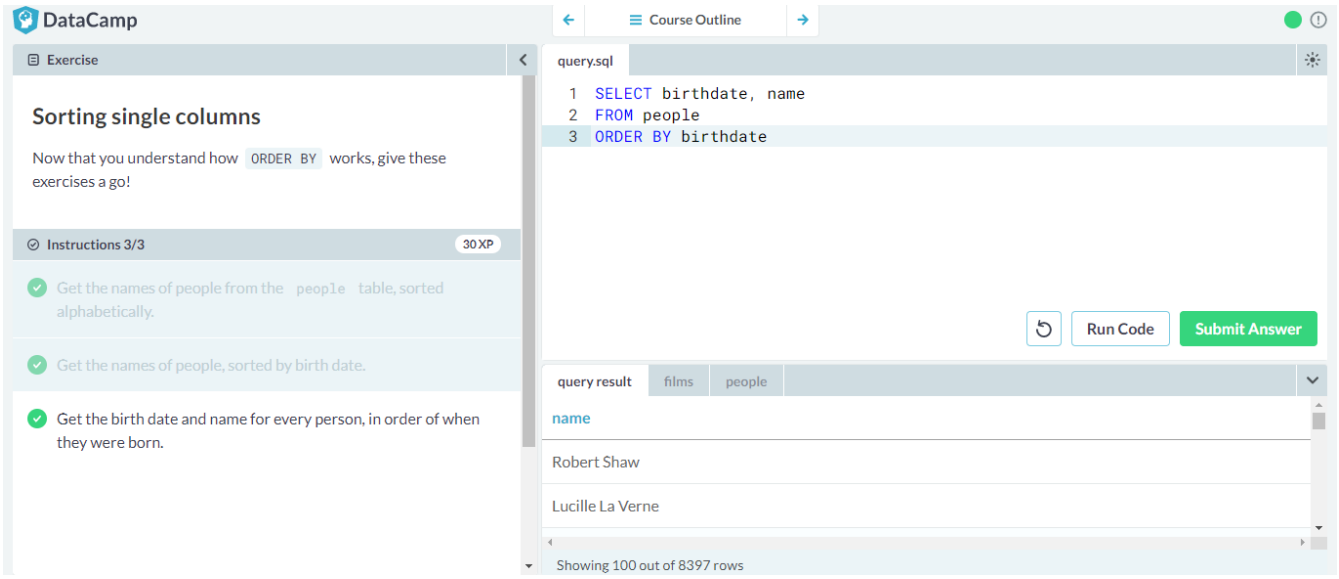

DataCamp
Course Outline
Exercise
The above now gives the correct answer of 450.0 since the numerator (45 * 100.0) of the division is now a decimal!
Instructions 3/3
0 XP
Get the percentage of people who are no longer alive. Alias the result as percentage_dead . Remember to use 100.0 and not 100 !
Get the number of years between the newest film and oldest film. Alias the result as difference .
Get the number of decades the films table covers. Alias the result as number_of_decades . The top half of your fraction should be enclosed in parentheses.
query.sql
solution.sql
1 SELECT (MAX(release_year) - MIN(release_year)) / 10.0
2 AS number_of_decades
3 FROM films;
Run Code
Run Solution
query result
films
people
difference
100

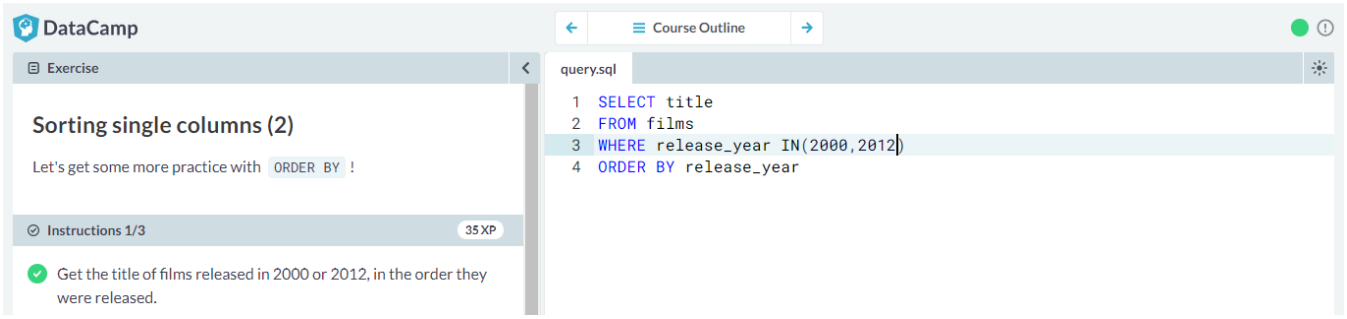

DataCamp
Course Outline
ORDER BY
Congratulations on making it this far! You now know how to select and filter your results.
In this chapter you'll learn how to sort and group your results to gain further insight. Let's go!
In SQL, the ORDER BY keyword is used to sort results in ascending or descending order according to the values of one or more columns.
By default ORDER BY will sort in ascending order. If you want to sort the results in descending order, you can use the DESC keyword. For example,
SELECT title
FROM films
ORDER BY release_year DESC;
gives you the titles of films sorted by release year, from newest to oldest.
How do you think ORDER BY sorts a column of text values by default?
Answer the question
50 XP

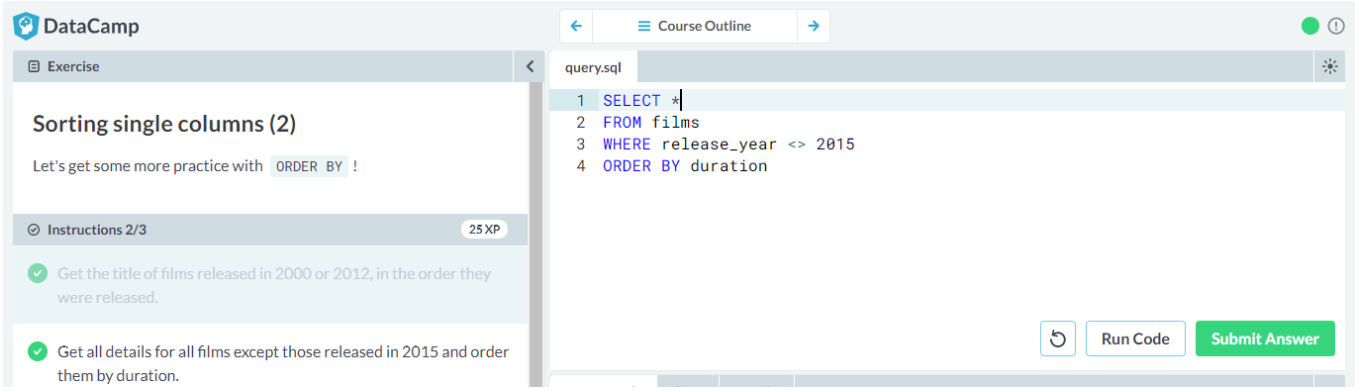

DataCamp
Course Outline
ORDER BY release_year DESC;
gives you the titles of films sorted by release year, from newest to oldest.
How do you think ORDER BY sorts a column of text values by default?
Answer the question
50 XP
Possible Answers
Alphabetically (A-Z)
press 1
Reverse alphabetically (Z-A)
press 2
There's no natural ordering to text data
press 3
By number of characters (fewest to most)
press 4
Take Hint (-15xp)
Enter
Submit Answer

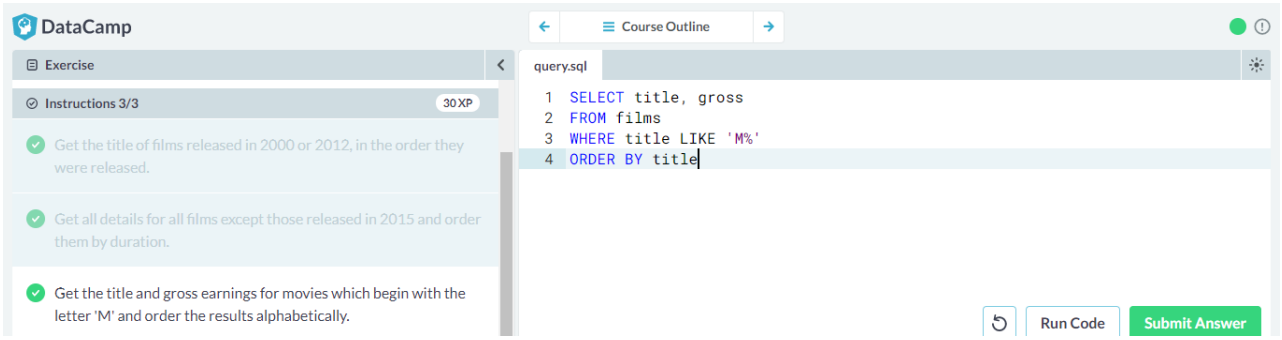

DataCamp
Course Outline
Exercise
Sorting single columns
Now that you understand how ORDER BY works, give these exercises a go!
Instructions 1/3
35 XP
Get the names of people from the people table, sorted alphabetically.
Get the names of people, sorted by birth date.
Get the birth date and name for every person, in order of when they were born.
query.sql
1 SELECT name
2 FROM people
3 ORDER BY name
Run Code
Submit Answer
query result
films
people
No query executed yet...
Showing 0 out of 0 rows

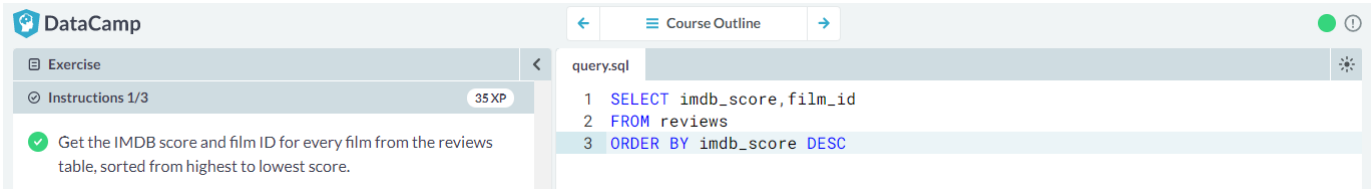

DataCamp
Course Outline
Exercise
Sorting single columns
Now that you understand how ORDER BY works, give these exercises a go!
Instructions 2/3
35 XP
Get the names of people from the people table, sorted alphabetically.
Get the names of people, sorted by birth date.
Get the birth date and name for every person, in order of when they were born.
query.sql
1 SELECT name
2 FROM people
3 ORDER BY birthdate
Run Code
Submit Answer
query result
films
people
id
name
birthdate
deathdate
1
50 Cent
1975-07-06
null
2
A. Michael Baldwin
1963-04-04
null
Showing 100 out of 8397 rows

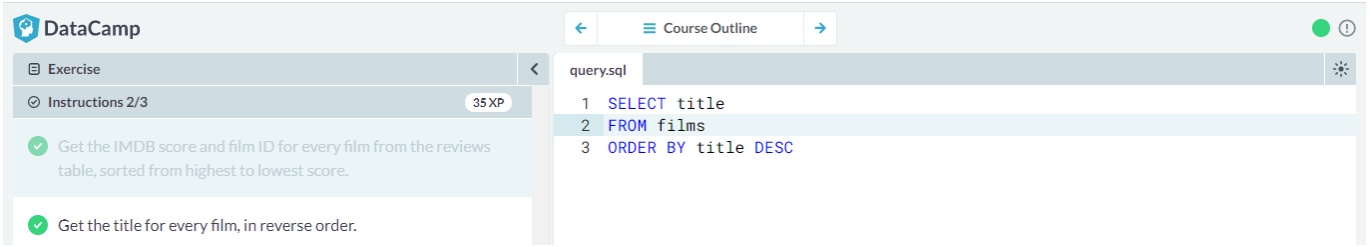

DataCamp
Course Outline
Exercise
Sorting single columns
Now that you understand how ORDER BY works, give these exercises a go!
Instructions 3/3
30 XP
Get the names of people from the people table, sorted alphabetically.
Get the names of people, sorted by birth date.
Get the birth date and name for every person, in order of when they were born.
query.sql
SELECT birthdate, name
FROM people
ORDER BY birthdate
Run Code
Submit Answer
query result
films
people
name
Robert Shaw
Lucille La Verne
Showing 100 out of 8397 rows

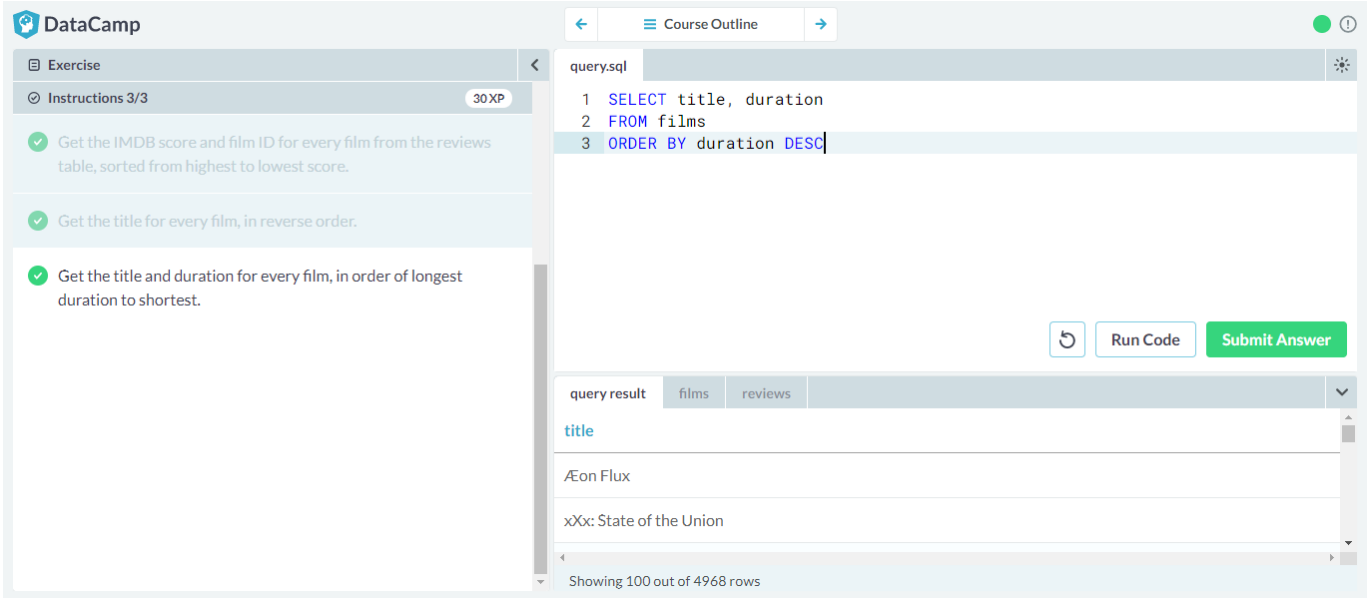

DataCamp
Course Outline
Exercise
Sorting single columns (2)
Let's get some more practice with ORDER BY !
Instructions 1/3
35 XP
Get the title of films released in 2000 or 2012, in the order they were released.
query.sql
SELECT title
FROM films
WHERE release_year IN(2000,2012)
ORDER BY release_year

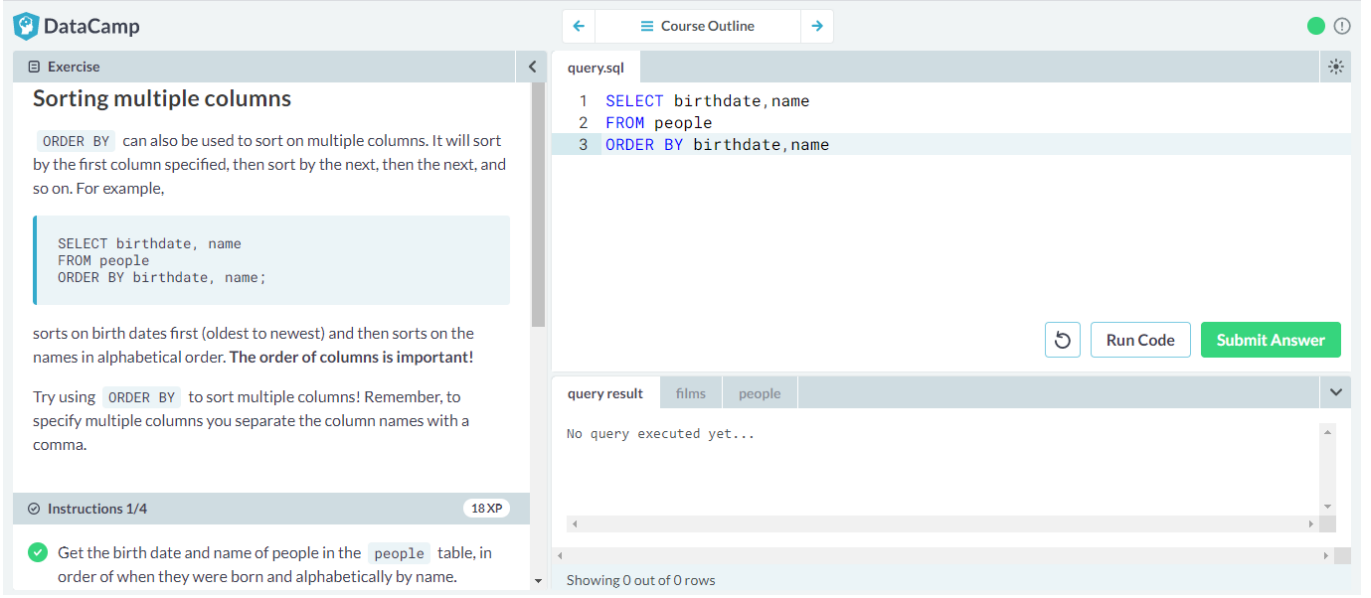

DataCamp
Course Outline
Exercise
Sorting single columns (2)
Let's get some more practice with ORDER BY !
Instructions 2/3
25 XP
Get the title of films released in 2000 or 2012, in the order they were released.
Get all details for all films except those released in 2015 and order them by duration.
query.sql
SELECT *
FROM films
WHERE release_year <> 2015
ORDER BY duration
Run Code
Submit Answer

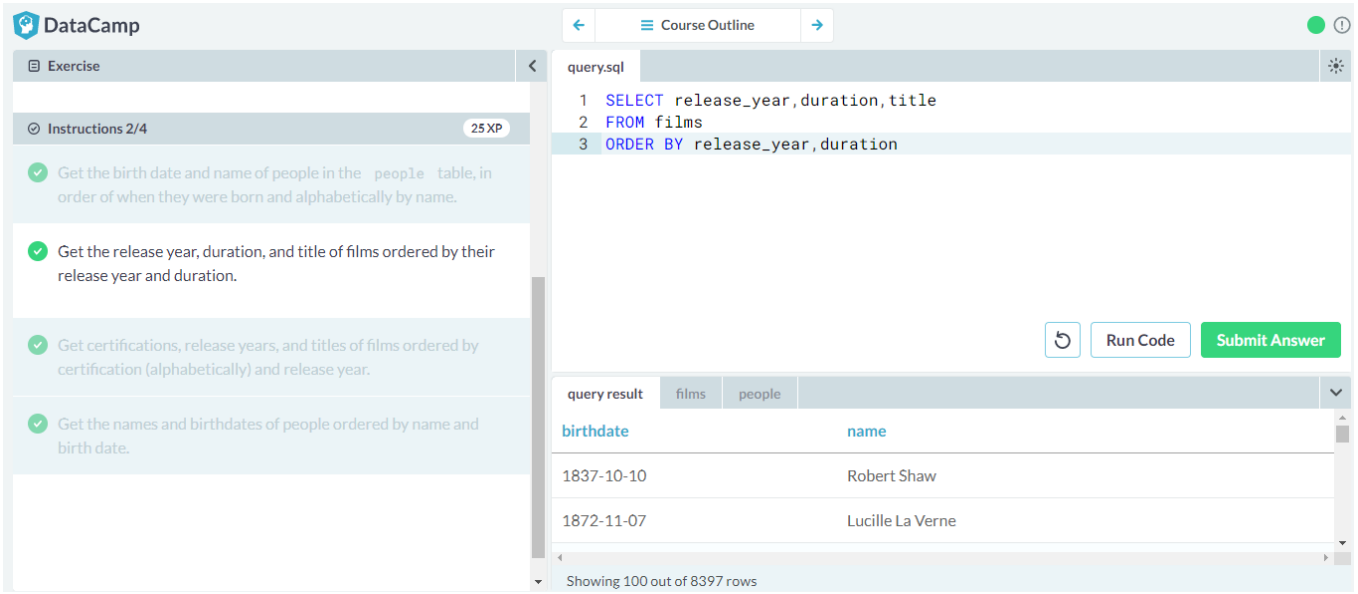

DataCamp
Course Outline
Exercise
Instructions 3/3
30 XP
Get the title of films released in 2000 or 2012, in the order they were released.
Get all details for all films except those released in 2015 and order them by duration.
Get the title and gross earnings for movies which begin with the letter 'M' and order the results alphabetically.
query.sql
SELECT title, gross
FROM films
WHERE title LIKE 'M%'
ORDER BY title
Run Code
Submit Answer

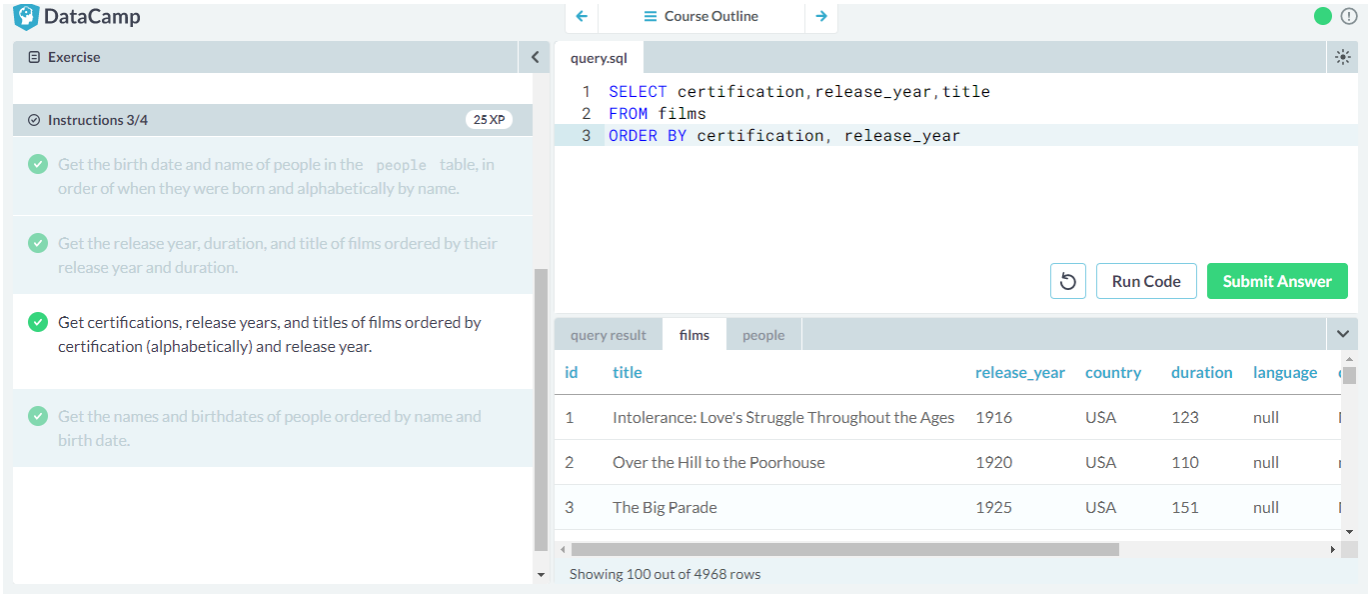

DataCamp
Course Outline
Exercise
Instructions 1/3
35 XP
Get the IMDB score and film ID for every film from the reviews table, sorted from highest to lowest score.
query.sql
1 SELECT imdb_score,film_id
2 FROM reviews
3 ORDER BY imdb_score DESC

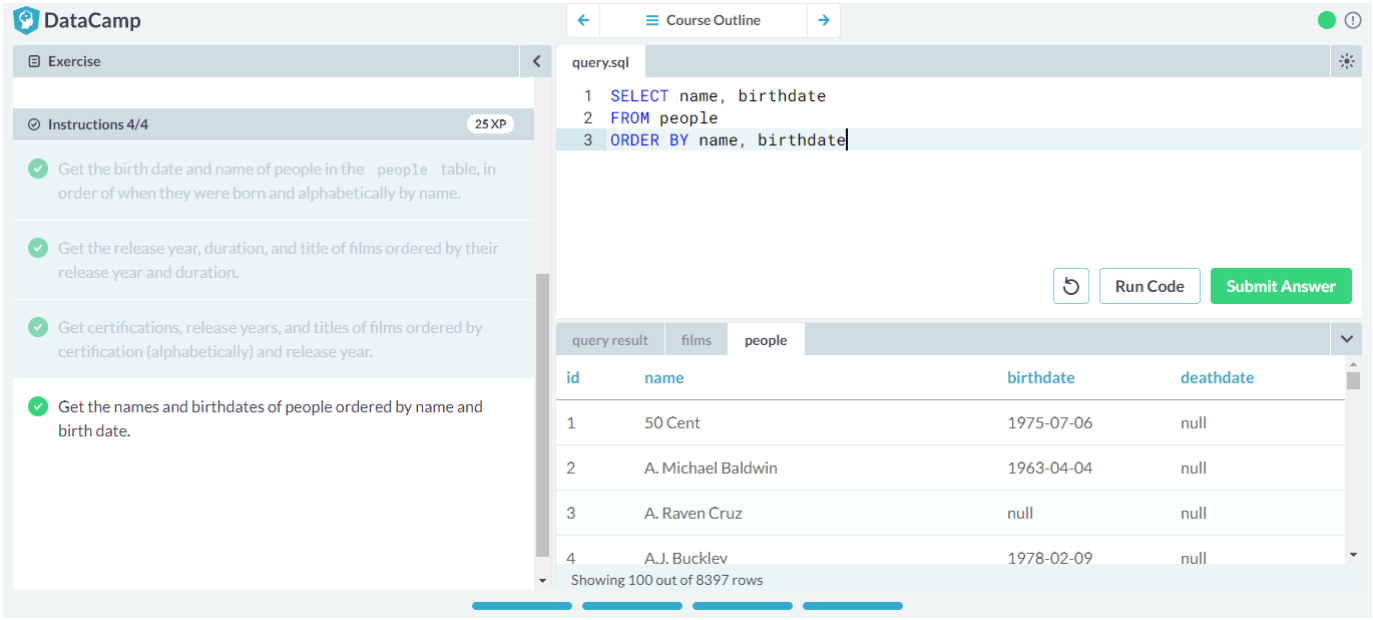

DataCamp
Course Outline
Exercise
Instructions 2/3
35 XP
Get the IMDB score and film ID for every film from the reviews table, sorted from highest to lowest score.
Get the title for every film, in reverse order.
query.sql
1 SELECT title
2 FROM films
3 ORDER BY title DESC

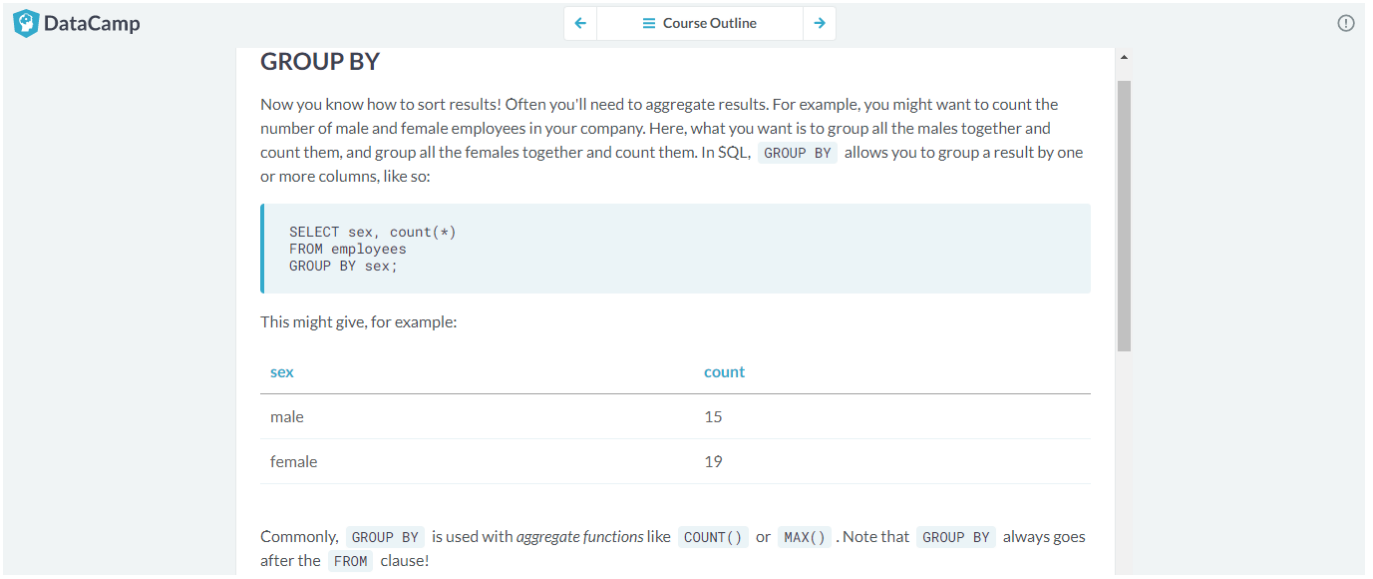

DataCamp
Course Outline
Exercise
Instructions 3/3
30 XP
Get the IMDB score and film ID for every film from the reviews table, sorted from highest to lowest score.
Get the title for every film, in reverse order.
Get the title and duration for every film, in order of longest duration to shortest.
query.sql
1 SELECT title, duration
2 FROM films
3 ORDER BY duration DESC
Run Code
Submit Answer
query result
films
reviews
title
Æon Flux
xXx: State of the Union
Showing 100 out of 4968 rows


DataCamp
Course Outline
Exercise
Sorting multiple columns
ORDER BY can also be used to sort on multiple columns. It will sort by the first column specified, then sort by the next, then the next, and so on. For example,
SELECT birthdate, name
FROM people
ORDER BY birthdate, name;
sorts on birth dates first (oldest to newest) and then sorts on the names in alphabetical order. The order of columns is important!
Try using ORDER BY to sort multiple columns! Remember, to specify multiple columns you separate the column names with a comma.
Instructions 1/4
18 XP
Get the birth date and name of people in the people table, in order of when they were born and alphabetically by name.
query.sql
1 SELECT birthdate,name
2 FROM people
3 ORDER BY birthdate,name
Run Code
Submit Answer
query result
films
people
No query executed yet...
Showing 0 out of 0 rows


DataCamp
Course Outline
Exercise
Instructions 2/4
25 XP
Get the birth date and name of people in the people table, in order of when they were born and alphabetically by name.
Get the release year, duration, and title of films ordered by their release year and duration.
Get certifications, release years, and titles of films ordered by certification (alphabetically) and release year.
Get the names and birthdates of people ordered by name and birth date.
query.sql
1 SELECT release_year,duration,title
2 FROM films
3 ORDER BY release_year,duration
Run Code
Submit Answer
query result
films
people
birthdate
name
1837-10-10
Robert Shaw
1872-11-07
Lucille La Verne
Showing 100 out of 8397 rows


DataCamp
Course Outline
Exercise
Instructions 3/4
25 XP
Get the birth date and name of people in the people table, in order of when they were born and alphabetically by name.
Get the release year, duration, and title of films ordered by their release year and duration.
Get certifications, release years, and titles of films ordered by certification (alphabetically) and release year.
Get the names and birthdates of people ordered by name and birth date.
query.sql
1 SELECT certification,release_year,title
2 FROM films
3 ORDER BY certification, release_year
Run Code
Submit Answer
query result
films
people
id
title
release_year
country
duration
language
1
Intolerance: Love's Struggle Throughout the Ages
1916
USA
123
null
2
Over the Hill to the Poorhouse
1920
USA
110
null
3
The Big Parade
1925
USA
151
null
Showing 100 out of 4968 rows





GROUP BY

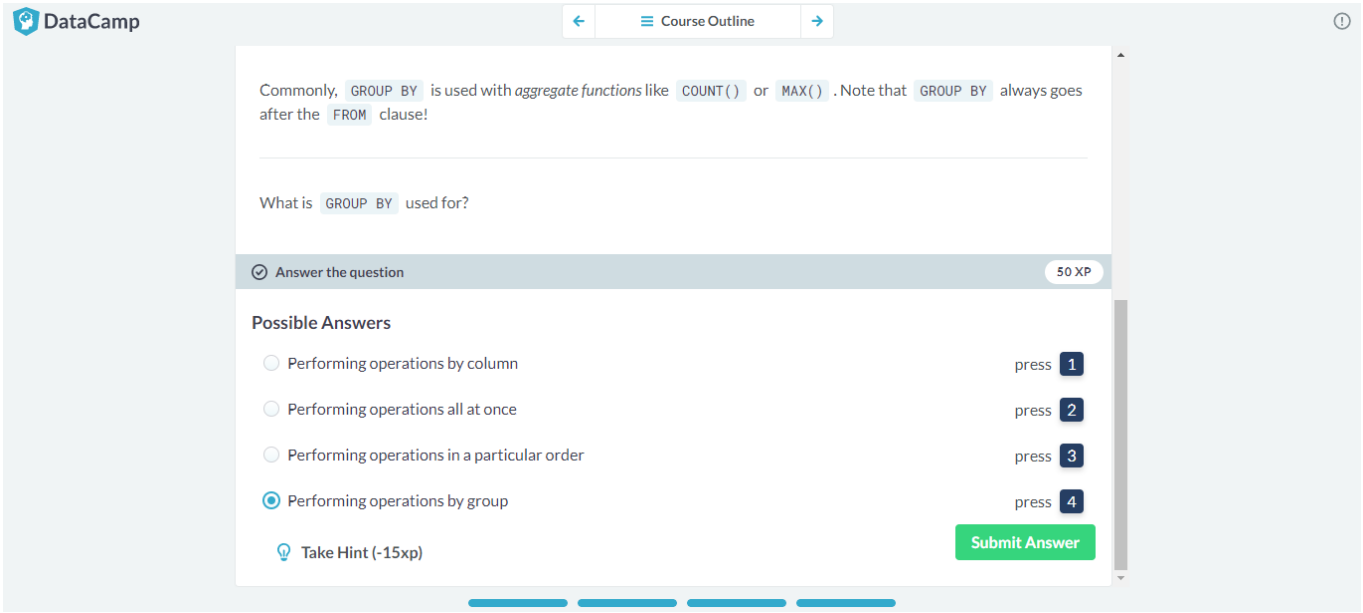
Now you know how to sort results! Often you'll need to aggregate results. For example, you might want to count the number of male and female employees in your company. Here, what you want is to group all the males together and count them, and group all the females together and count them. In SQL, `GROUP BY` allows you to group a result by one or more columns, like so:

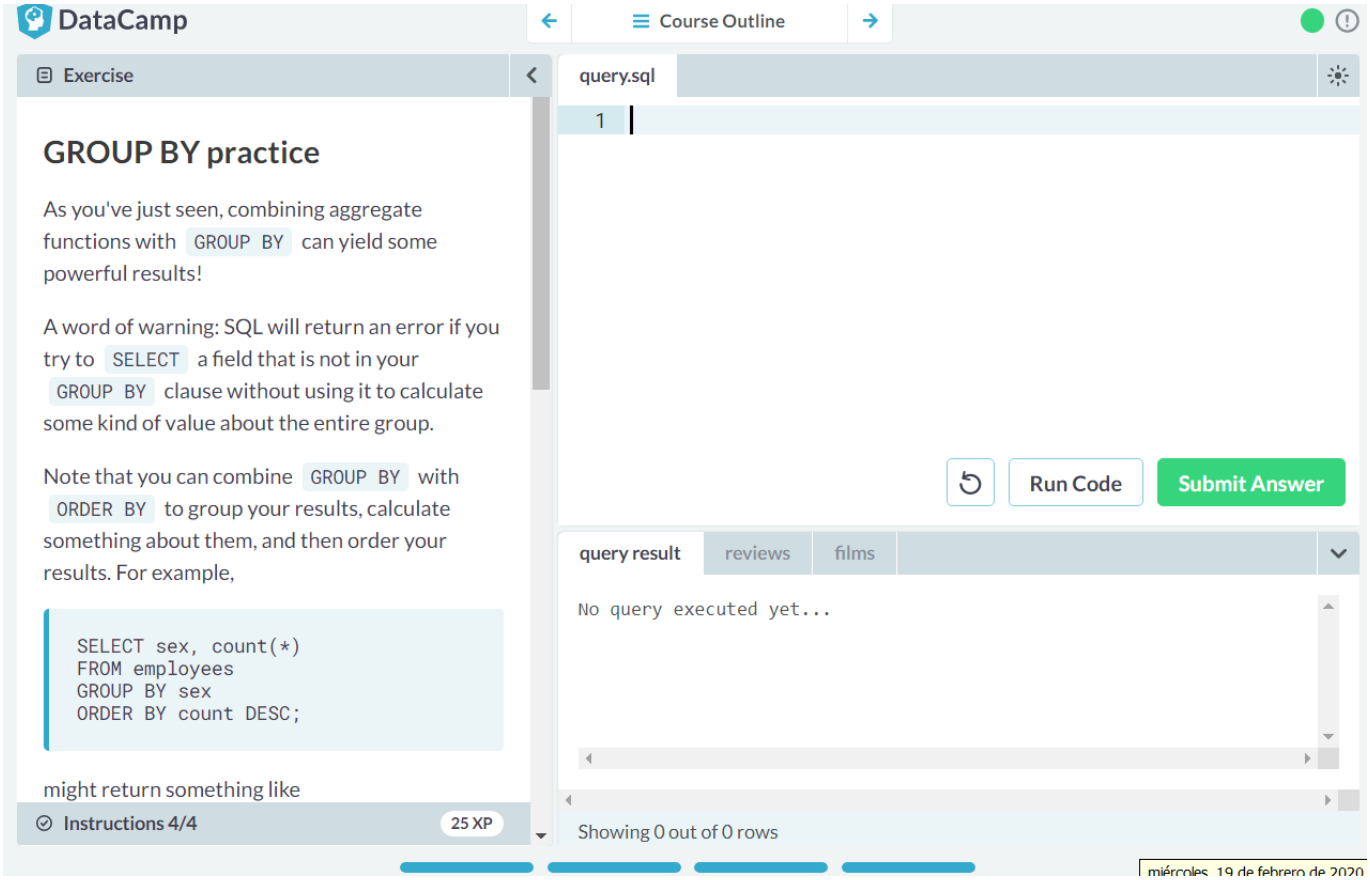
```
SELECT sex, count(*)
FROM employees
GROUP BY sex;
```

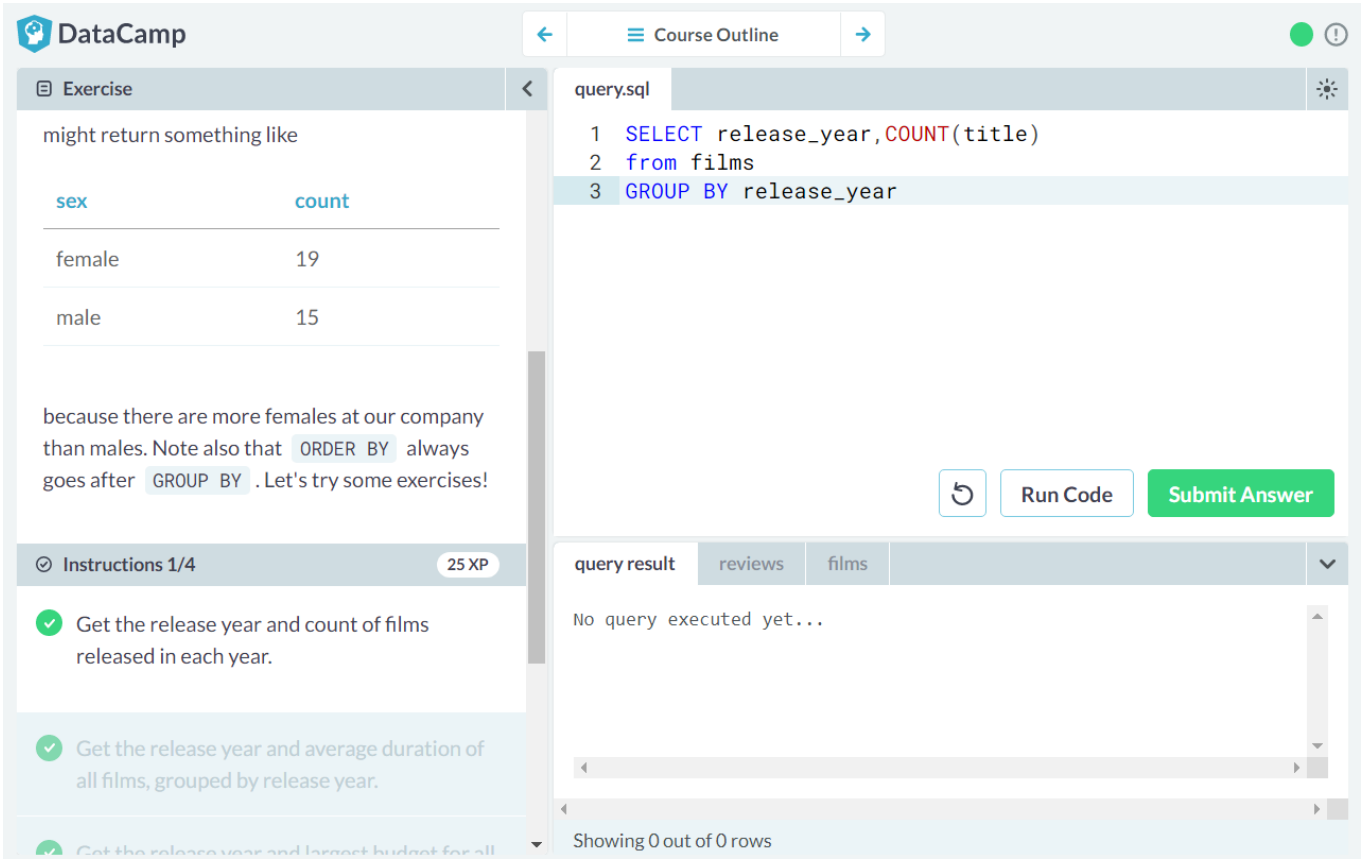
This might give, for example:

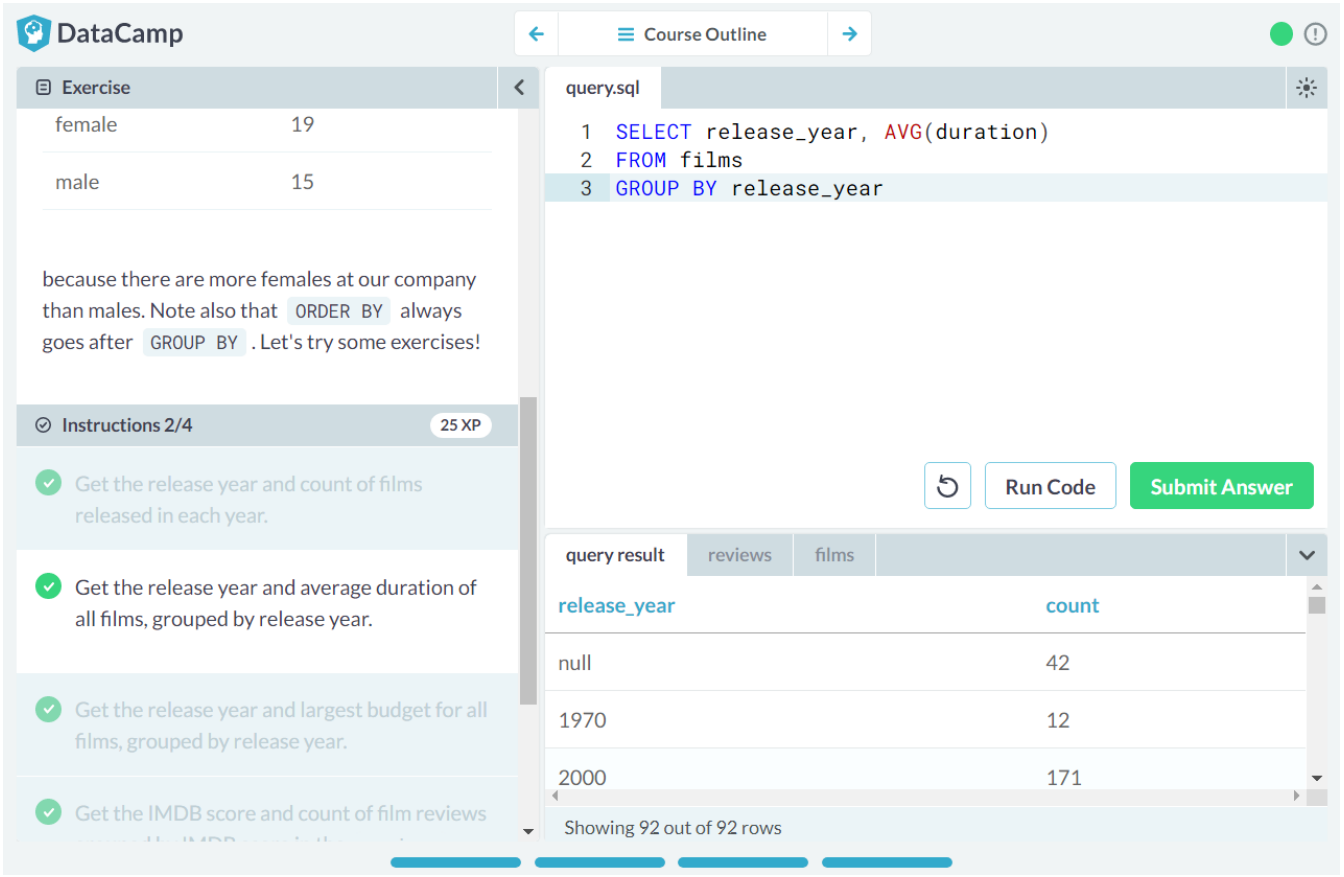
sex	count
male	15
female	19

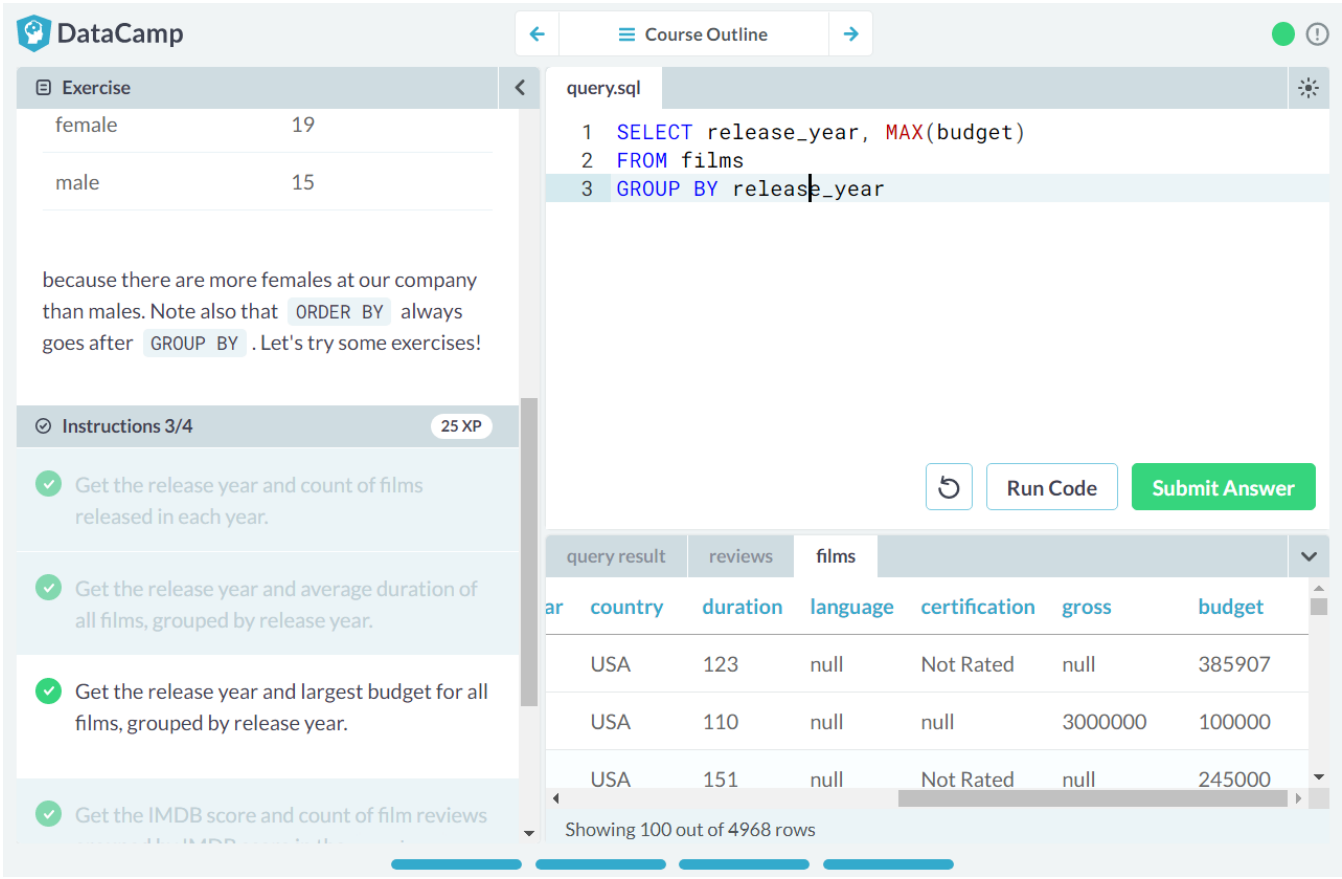
Commonly, `GROUP BY` is used with *aggregate functions* like `COUNT()` or `MAX()`. Note that `GROUP BY` always goes after the `FROM` clause!

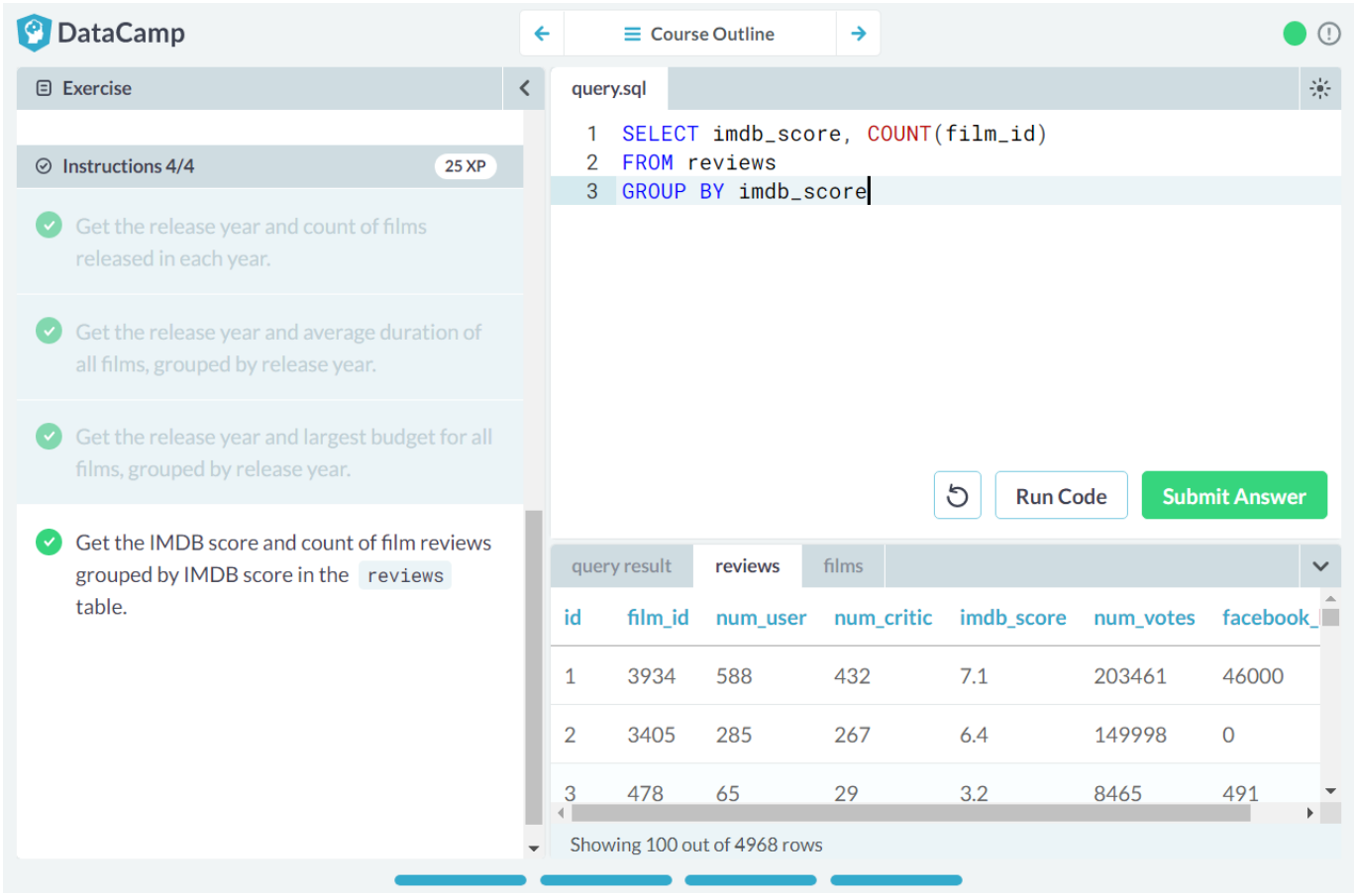

DataCamp
Course Outline
Commonly, GROUP BY is used with aggregate functions like COUNT() or MAX(). Note that GROUP BY always goes after the FROM clause!
What is GROUP BY used for?
Answer the question
50 XP
Possible Answers
Performing operations by column
press 1
Performing operations all at once
press 2
Performing operations in a particular order
press 3
Performing operations by group
press 4
Take Hint (-15xp)
Submit Answer

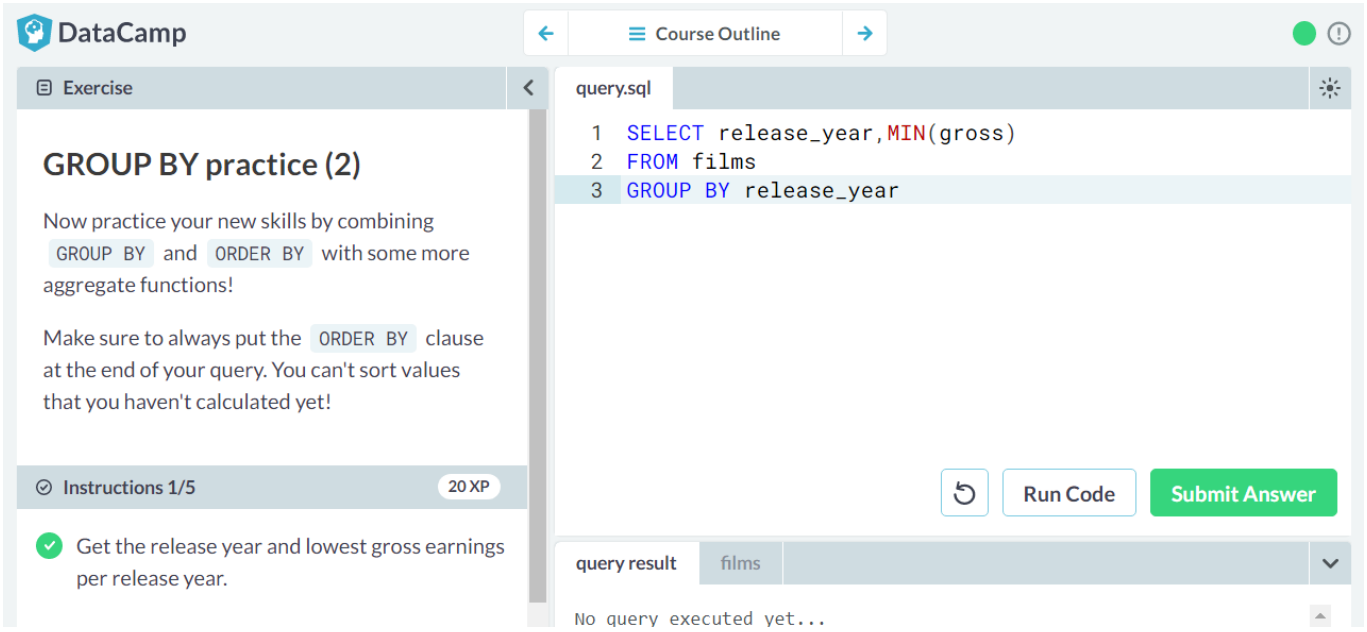

DataCamp
Course Outline
Exercise
query.sql
1
GROUP BY practice
As you've just seen, combining aggregate functions with GROUP BY can yield some powerful results!
A word of warning: SQL will return an error if you try to SELECT a field that is not in your GROUP BY clause without using it to calculate some kind of value about the entire group.
Note that you can combine GROUP BY with ORDER BY to group your results, calculate something about them, and then order your results. For example,
SELECT sex, count(*)
FROM employees
GROUP BY sex
ORDER BY count DESC;
might return something like
Run Code
Submit Answer
query result
reviews
films
No query executed yet...
Instructions 4/4
25 XP
Showing 0 out of 0 rows
miércoles, 19 de febrero de 2020

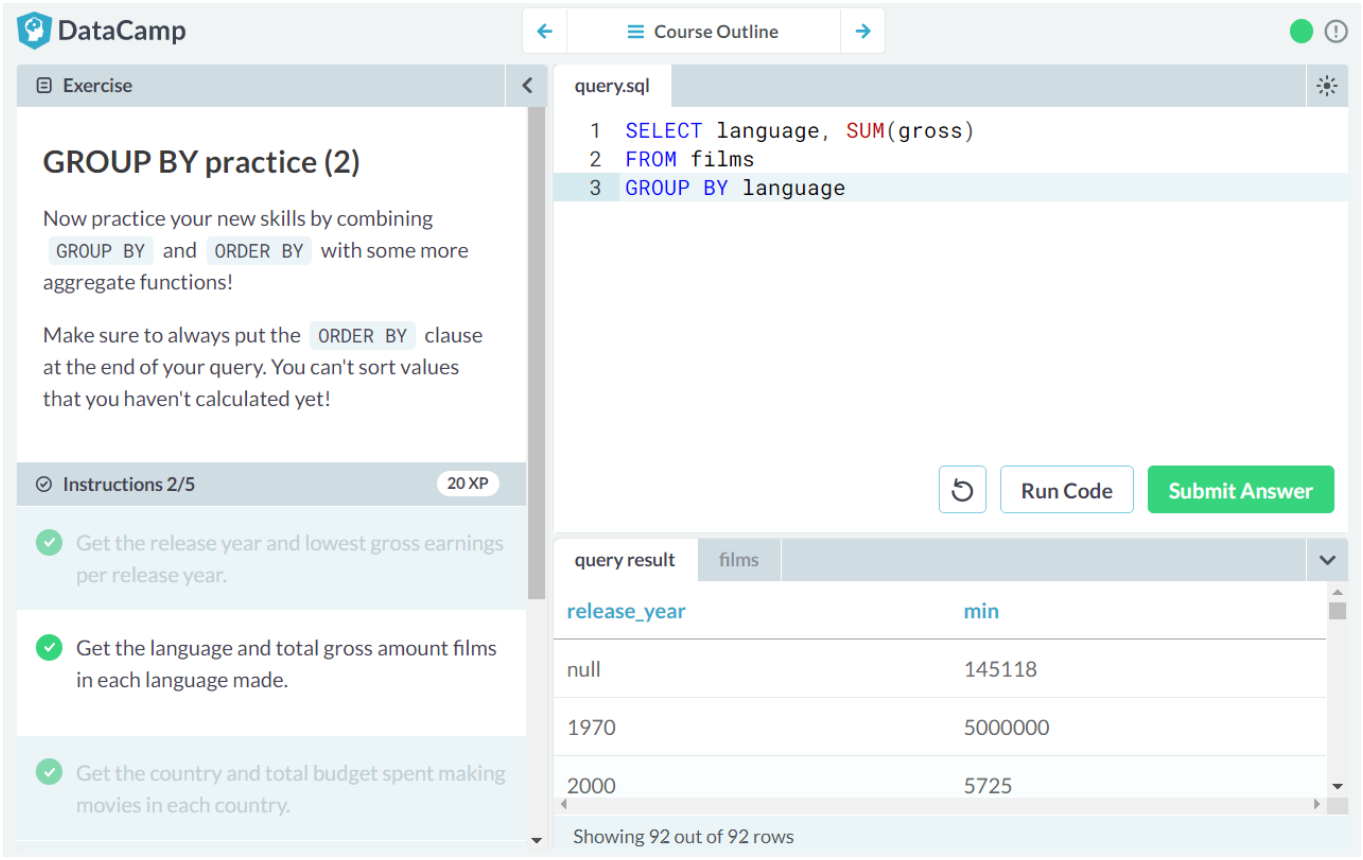

DataCamp
Course Outline
Exercise
might return something like
sex
count
female
19
male
15
because there are more females at our company than males. Note also that ORDER BY always goes after GROUP BY . Let's try some exercises!
Instructions 1/4
25 XP
Get the release year and count of films released in each year.
Get the release year and average duration of all films, grouped by release year.
query.sql
1 SELECT release_year,COUNT(title)
2 from films
3 GROUP BY release_year
Run Code
Submit Answer
query result
reviews
films
No query executed yet...
Showing 0 out of 0 rows

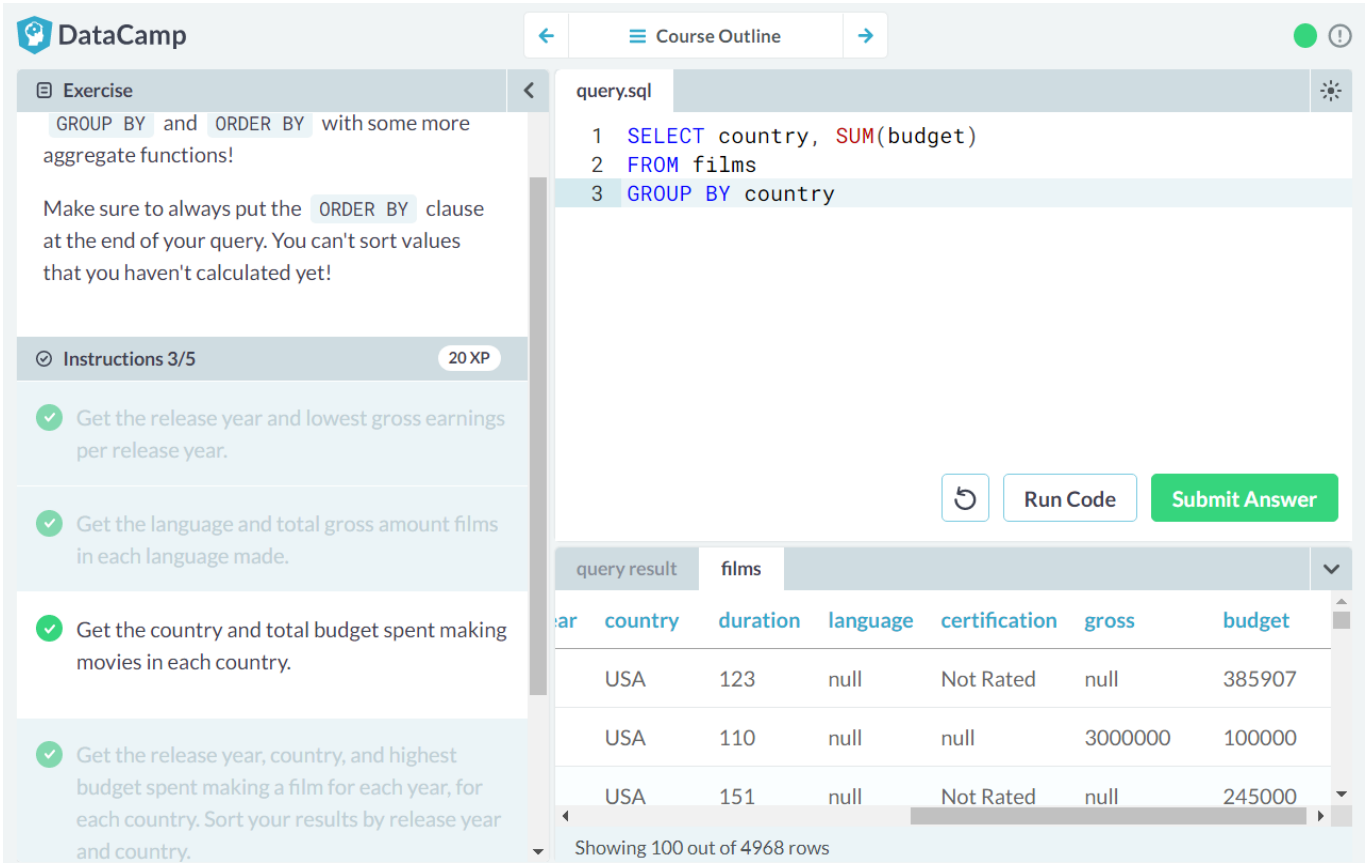

DataCamp
Course Outline
Exercise
female 19
male 15
because there are more females at our company than males. Note also that ORDER BY always goes after GROUP BY . Let's try some exercises!
Instructions 2/4
25 XP
Get the release year and count of films released in each year.
Get the release year and average duration of all films, grouped by release year.
Get the release year and largest budget for all films, grouped by release year.
Get the IMDB score and count of film reviews
query.sql
1 SELECT release_year, AVG(duration)
2 FROM films
3 GROUP BY release_year
Run Code
Submit Answer
query result
reviews
films
release_year
count
null 42
1970 12
2000 171
Showing 92 out of 92 rows

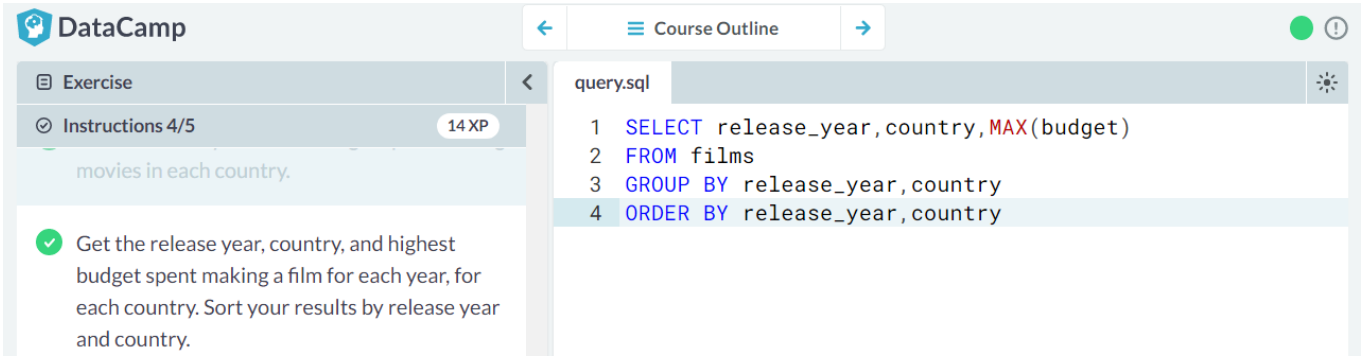

DataCamp
Course Outline
Exercise
female 19
male 15
because there are more females at our company than males. Note also that ORDER BY always goes after GROUP BY . Let's try some exercises!
Instructions 3/4
25 XP
Get the release year and count of films released in each year.
Get the release year and average duration of all films, grouped by release year.
Get the release year and largest budget for all films, grouped by release year.
Get the IMDB score and count of film reviews
query.sql
1 SELECT release_year, MAX(budget)
2 FROM films
3 GROUP BY release_year
Run Code
Submit Answer
query result
reviews
films
ar country duration language certification gross budget
USA 123 null Not Rated null 385907
USA 110 null null 3000000 100000
USA 151 null Not Rated null 245000
Showing 100 out of 4968 rows

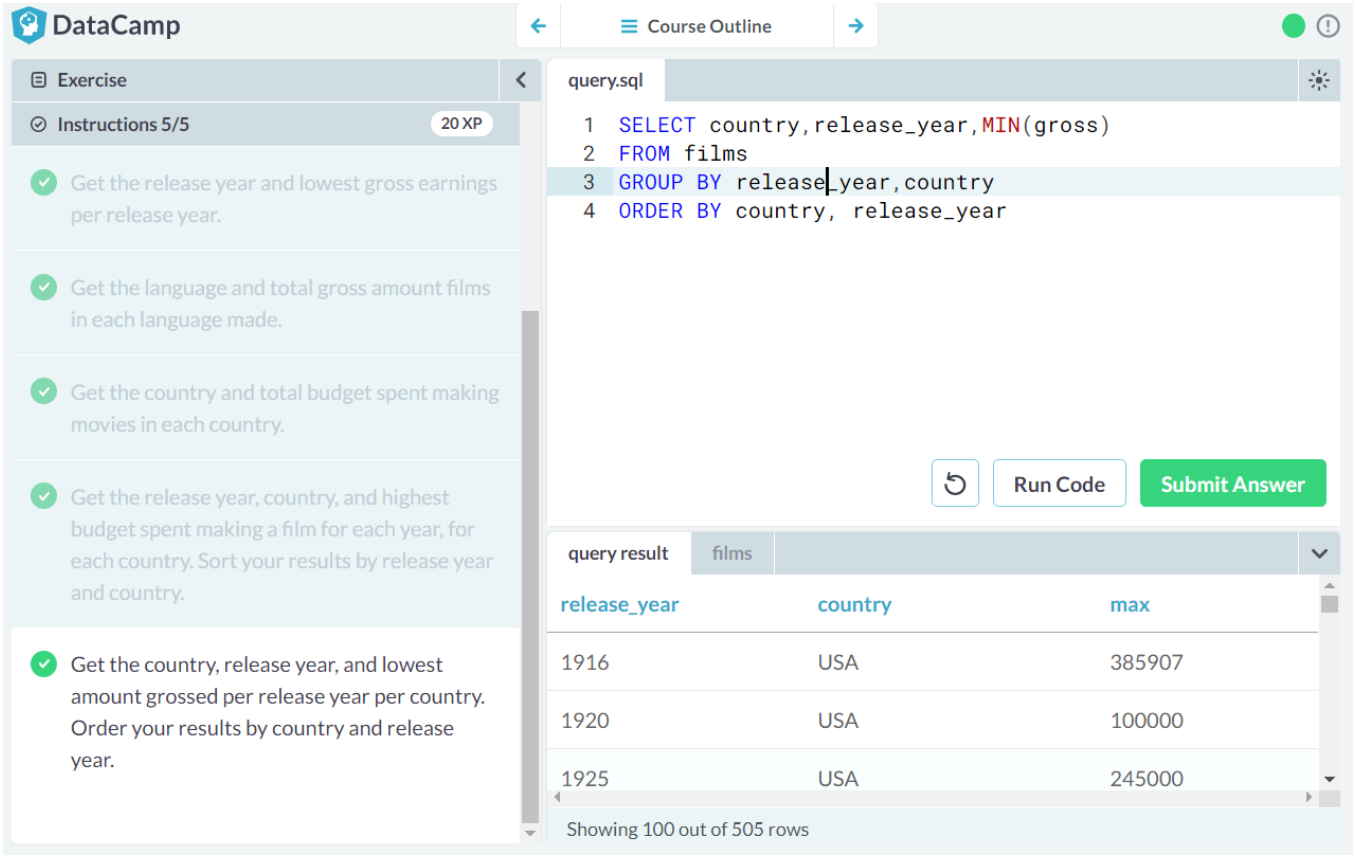

DataCamp
Course Outline
Exercise
Instructions 4/4
25 XP
Get the release year and count of films released in each year.
Get the release year and average duration of all films, grouped by release year.
Get the release year and largest budget for all films, grouped by release year.
Get the IMDB score and count of film reviews grouped by IMDB score in the reviews table.
query.sql
SELECT imdb_score, COUNT(film_id)
FROM reviews
GROUP BY imdb_score
Run Code
Submit Answer
query result
reviews
films
id film_id num_user num_critic imdb_score num_votes facebook_
1 3934 588 432 7.1 203461 46000
2 3405 285 267 6.4 149998 0
3 478 65 29 3.2 8465 491
Showing 100 out of 4968 rows

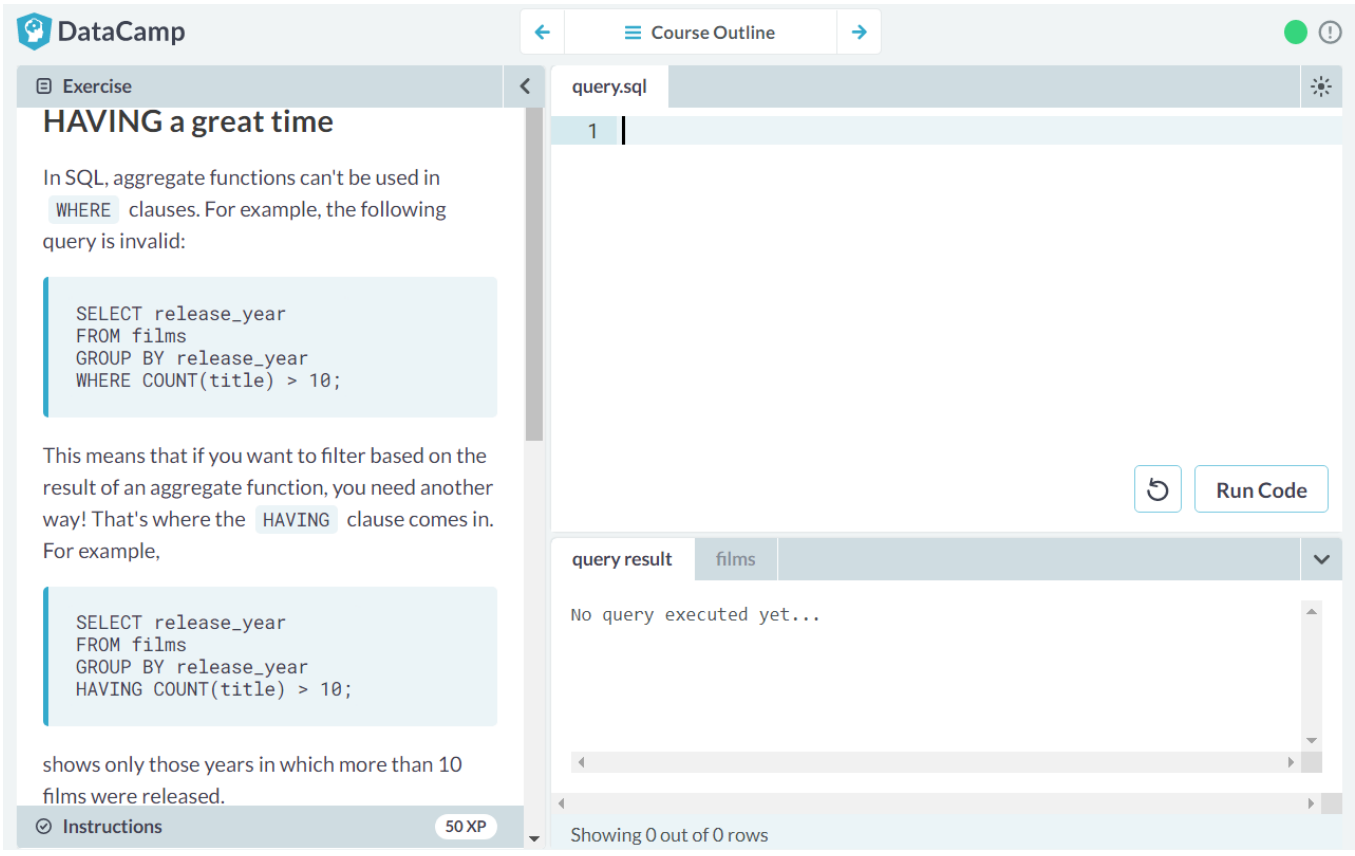

DataCamp
Course Outline
Exercise
GROUP BY practice (2)
Now practice your new skills by combining GROUP BY and ORDER BY with some more aggregate functions!
Make sure to always put the ORDER BY clause at the end of your query. You can't sort values that you haven't calculated yet!
Instructions 1/5
20 XP
Get the release year and lowest gross earnings per release year.
query.sql
SELECT release_year,MIN(gross)
FROM films
GROUP BY release_year
Run Code
Submit Answer
query result
films
No query executed yet...

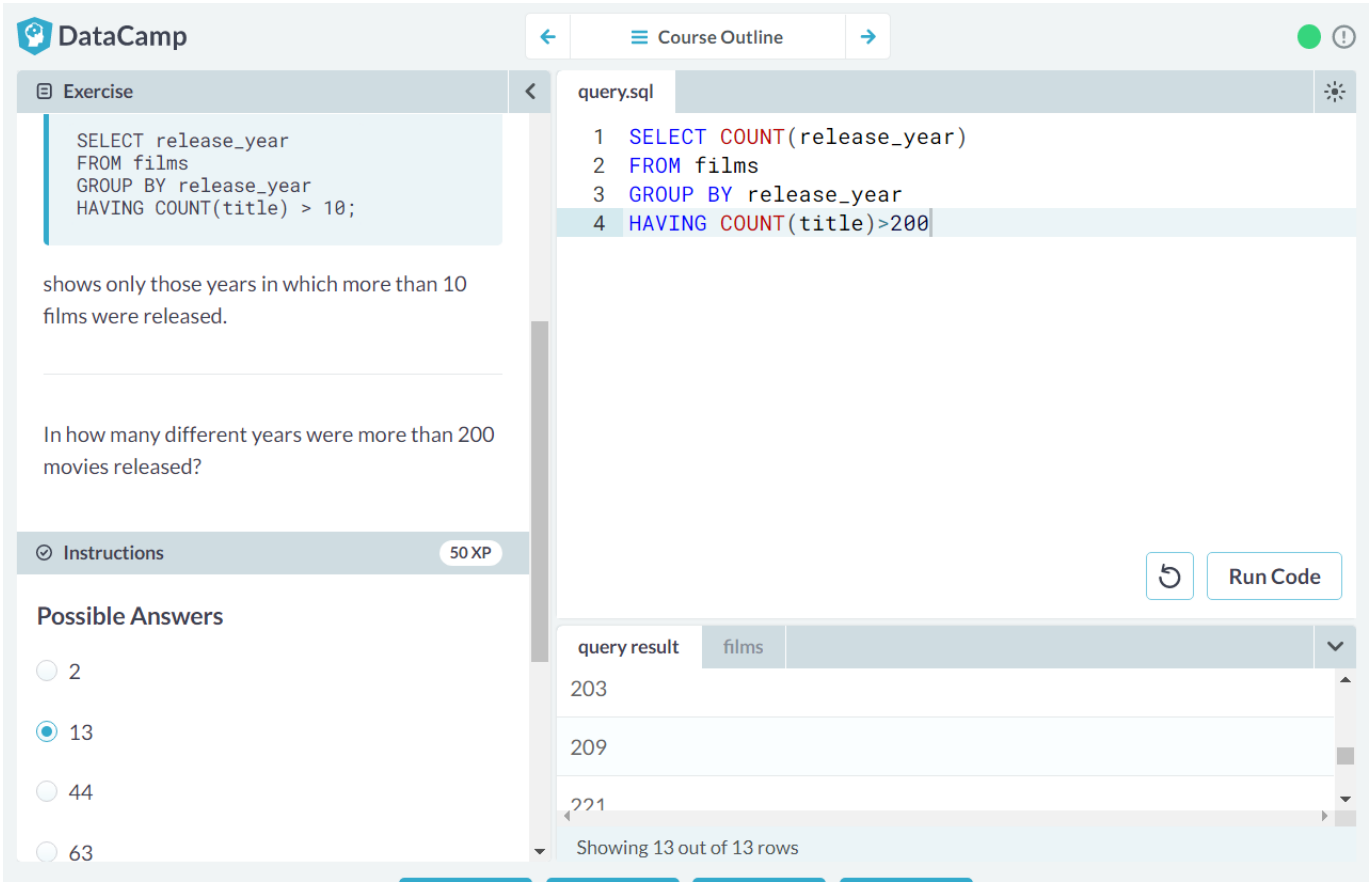

DataCamp
Course Outline
Exercise
GROUP BY practice (2)
Now practice your new skills by combining GROUP BY and ORDER BY with some more aggregate functions!
Make sure to always put the ORDER BY clause at the end of your query. You can't sort values that you haven't calculated yet!
Instructions 2/5
20 XP
Get the release year and lowest gross earnings per release year.
Get the language and total gross amount films in each language made.
Get the country and total budget spent making movies in each country.
query.sql
1 SELECT language, SUM(gross)
2 FROM films
3 GROUP BY language
Run Code
Submit Answer
query result
films
release_year
min
null
145118
1970
5000000
2000
5725
Showing 92 out of 92 rows

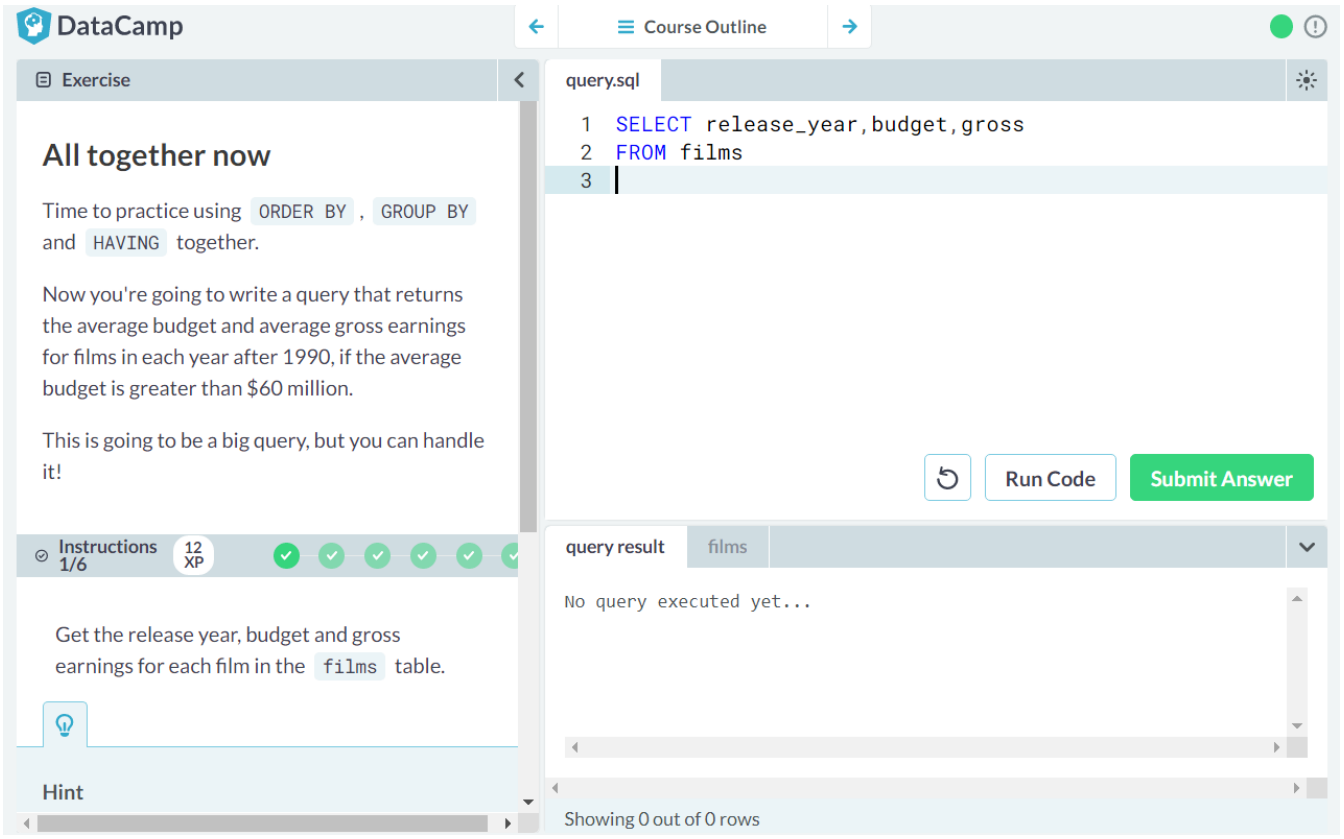

DataCamp
Course Outline
Exercise
GROUP BY and ORDER BY with some more aggregate functions!
Make sure to always put the ORDER BY clause at the end of your query. You can't sort values that you haven't calculated yet!
Instructions 3/5
20 XP
Get the release year and lowest gross earnings per release year.
Get the language and total gross amount films in each language made.
Get the country and total budget spent making movies in each country.
Get the release year, country, and highest budget spent making a film for each year, for each country. Sort your results by release year and country.
query.sql
1 SELECT country, SUM(budget)
2 FROM films
3 GROUP BY country
Run Code
Submit Answer
query result
films
ar country duration language certification gross budget
USA 123 null Not Rated null 385907
USA 110 null null 3000000 100000
USA 151 null Not Rated null 245000
Showing 100 out of 4968 rows

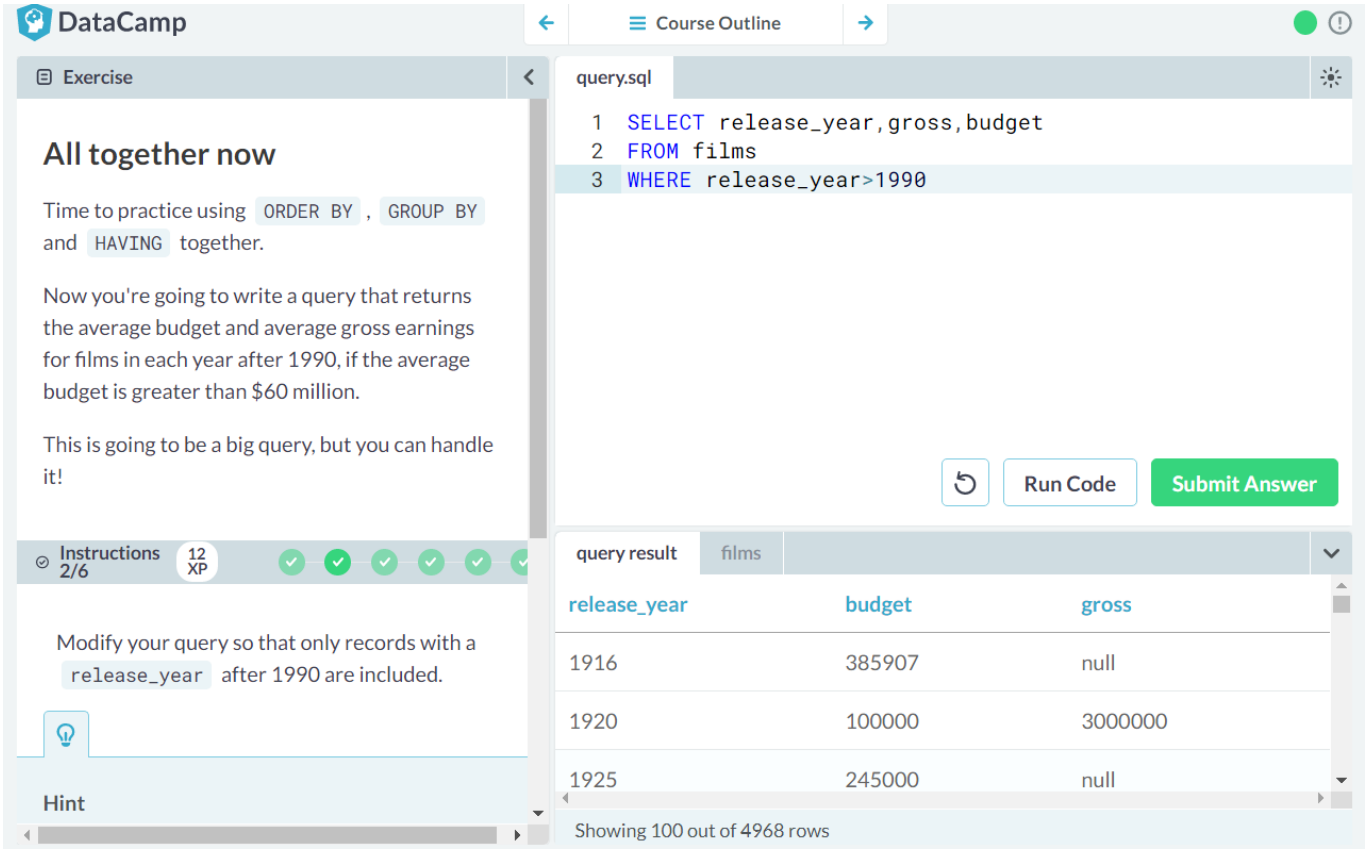

DataCamp
Course Outline
Exercise
Instructions 4/5
14 XP
movies in each country.
Get the release year, country, and highest budget spent making a film for each year, for each country. Sort your results by release year and country.
query.sql
1 SELECT release_year,country,MAX(budget)
2 FROM films
3 GROUP BY release_year,country
4 ORDER BY release_year,country

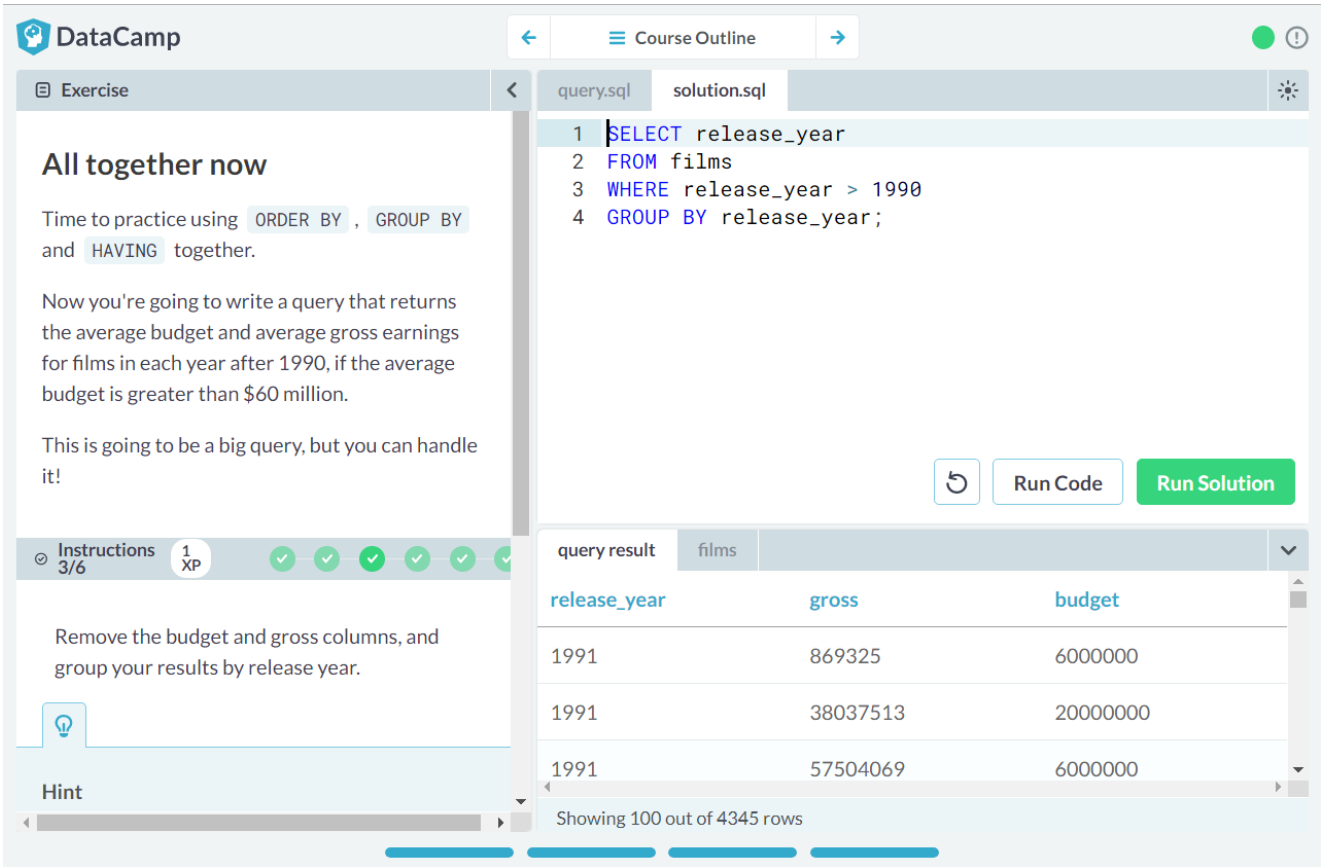

DataCamp
Course Outline
Exercise
Instructions 5/5
20 XP
Get the release year and lowest gross earnings per release year.
Get the language and total gross amount films in each language made.
Get the country and total budget spent making movies in each country.
Get the release year, country, and highest budget spent making a film for each year, for each country. Sort your results by release year and country.
Get the country, release year, and lowest amount grossed per release year per country. Order your results by country and release year.
query.sql
1 SELECT country,release_year,MIN(gross)
2 FROM films
3 GROUP BY release_year,country
4 ORDER BY country, release_year
Run Code
Submit Answer
query result
films
release_year
country
max
1916
USA
385907
1920
USA
100000
1925
USA
245000
Showing 100 out of 505 rows

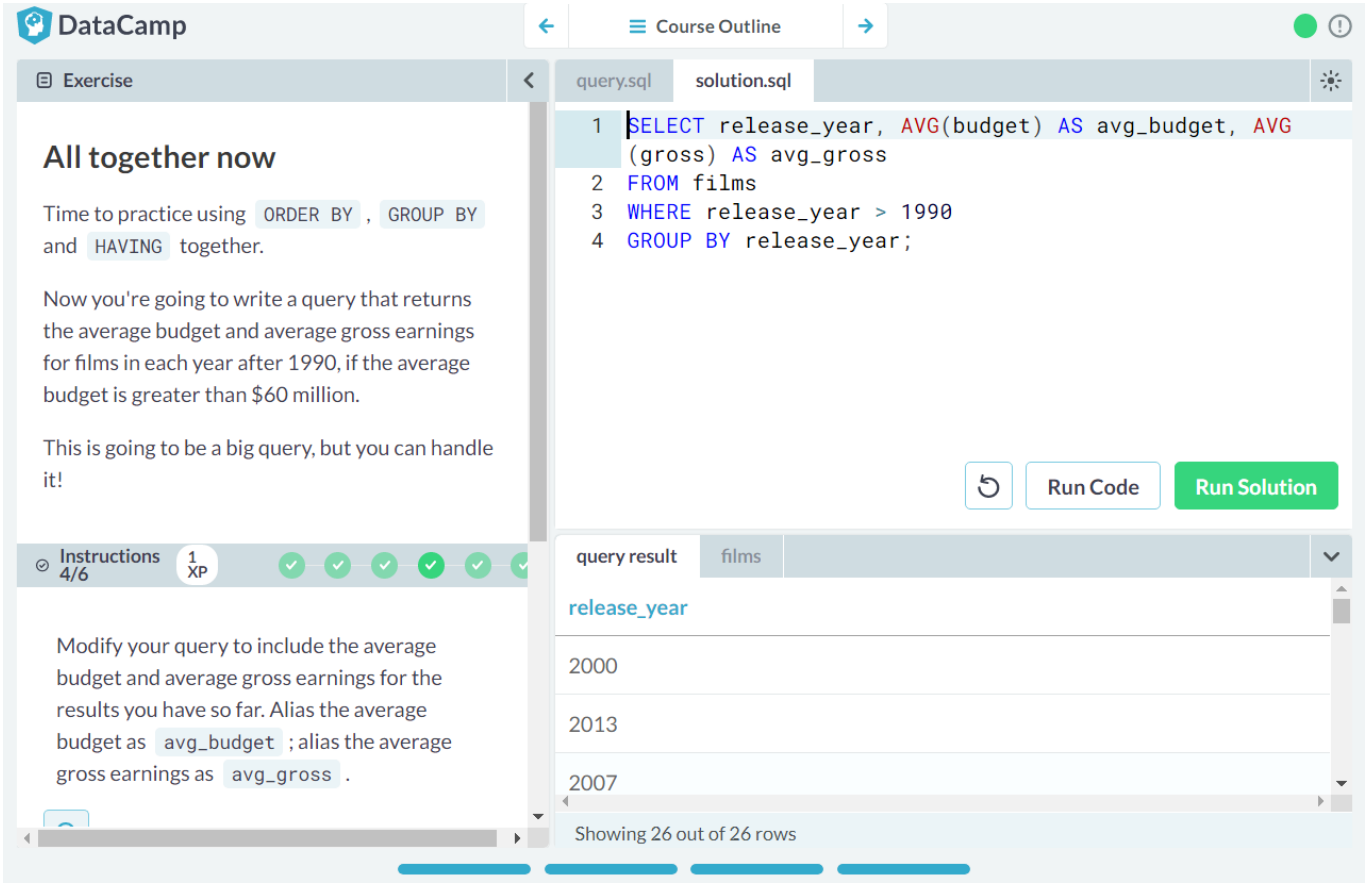

DataCamp
Course Outline
Exercise
HAVING a great time
In SQL, aggregate functions can't be used in WHERE clauses. For example, the following query is invalid:
SELECT release_year
FROM films
GROUP BY release_year
WHERE COUNT(title) > 10;
This means that if you want to filter based on the result of an aggregate function, you need another way! That's where the HAVING clause comes in. For example,
SELECT release_year
FROM films
GROUP BY release_year
HAVING COUNT(title) > 10;
shows only those years in which more than 10 films were released.
Instructions
50 XP
query.sql
1
Run Code
query result
films
No query executed yet...
Showing 0 out of 0 rows

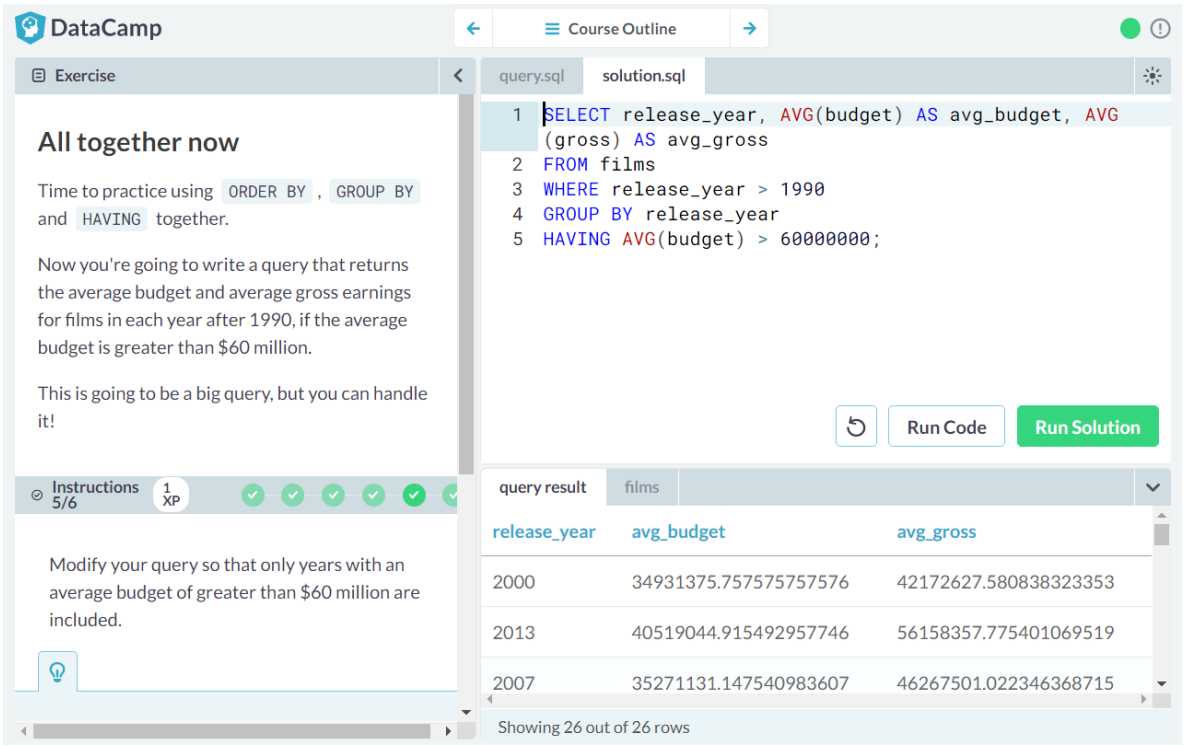

DataCamp
Course Outline
Exercise
SELECT release_year
FROM films
GROUP BY release_year
HAVING COUNT(title) > 10;
shows only those years in which more than 10 films were released.
In how many different years were more than 200 movies released?
Instructions
50 XP
Possible Answers
2
13
44
63
query.sql
1 SELECT COUNT(release_year)
2 FROM films
3 GROUP BY release_year
4 HAVING COUNT(title)>200
Run Code
query result
films
203
209
221
Showing 13 out of 13 rows

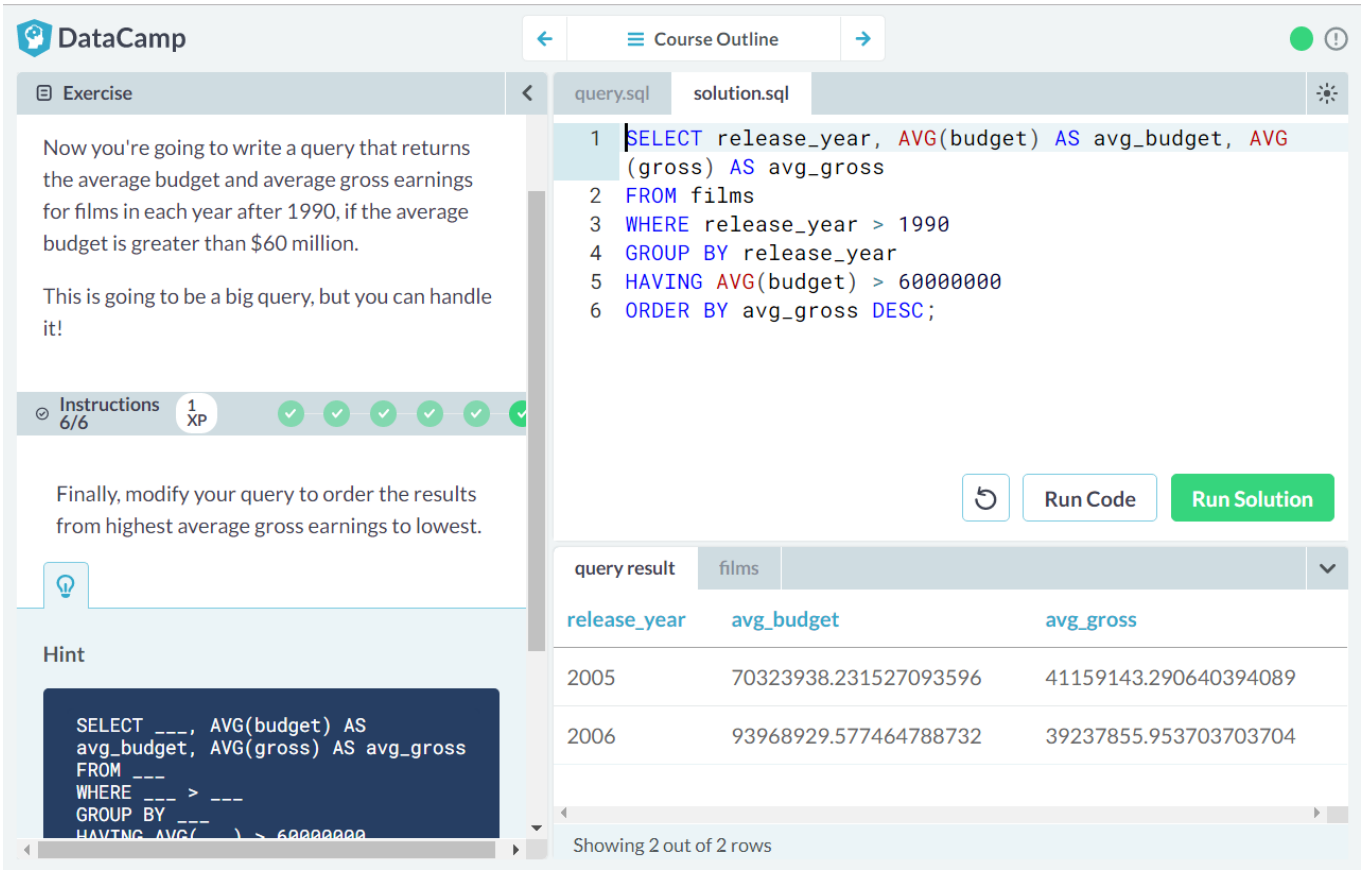

DataCamp
Course Outline
Exercise
All together now
Time to practice using ORDER BY , GROUP BY and HAVING together.
Now you're going to write a query that returns the average budget and average gross earnings for films in each year after 1990, if the average budget is greater than $60 million.
This is going to be a big query, but you can handle it!
Instructions 1/6
12 XP
Get the release year, budget and gross earnings for each film in the films table.
Hint
query.sql
1 SELECT release_year,budget,gross
2 FROM films
3
Run Code
Submit Answer
query result
films
No query executed yet...
Showing 0 out of 0 rows

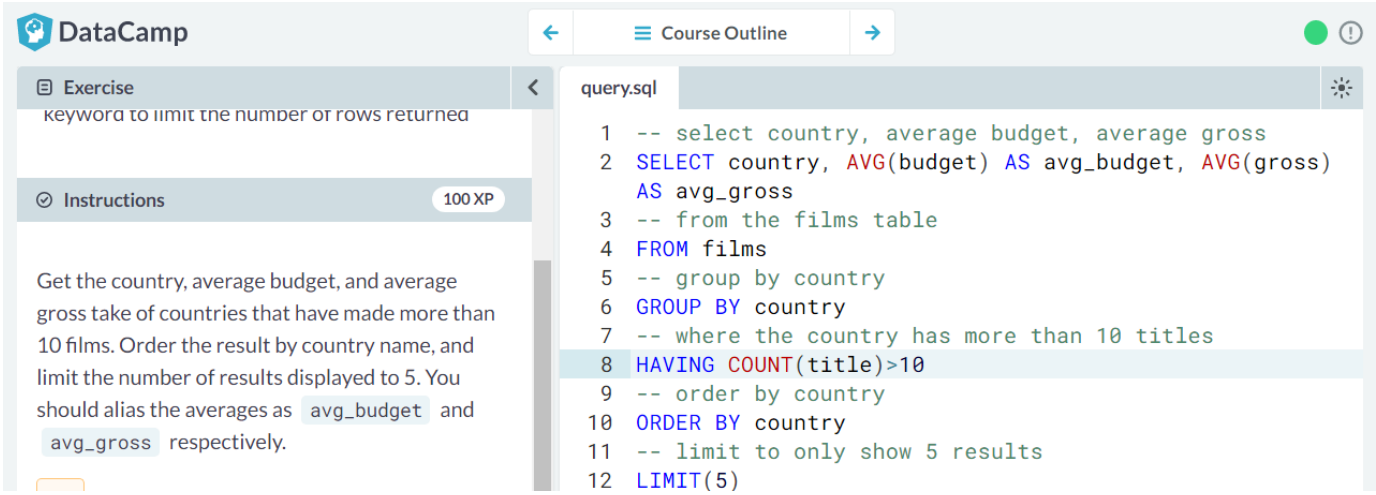

DataCamp
Course Outline
Exercise
All together now
Time to practice using ORDER BY , GROUP BY and HAVING together.
Now you're going to write a query that returns the average budget and average gross earnings for films in each year after 1990, if the average budget is greater than $60 million.
This is going to be a big query, but you can handle it!
Instructions 2/6
12 XP
Modify your query so that only records with a release_year after 1990 are included.
Hint
query.sql
1 SELECT release_year,gross,budget
2 FROM films
3 WHERE release_year>1990
Run Code
Submit Answer
query result
films
release_year
budget
gross
1916
385907
null
1920
100000
3000000
1925
245000
null
Showing 100 out of 4968 rows

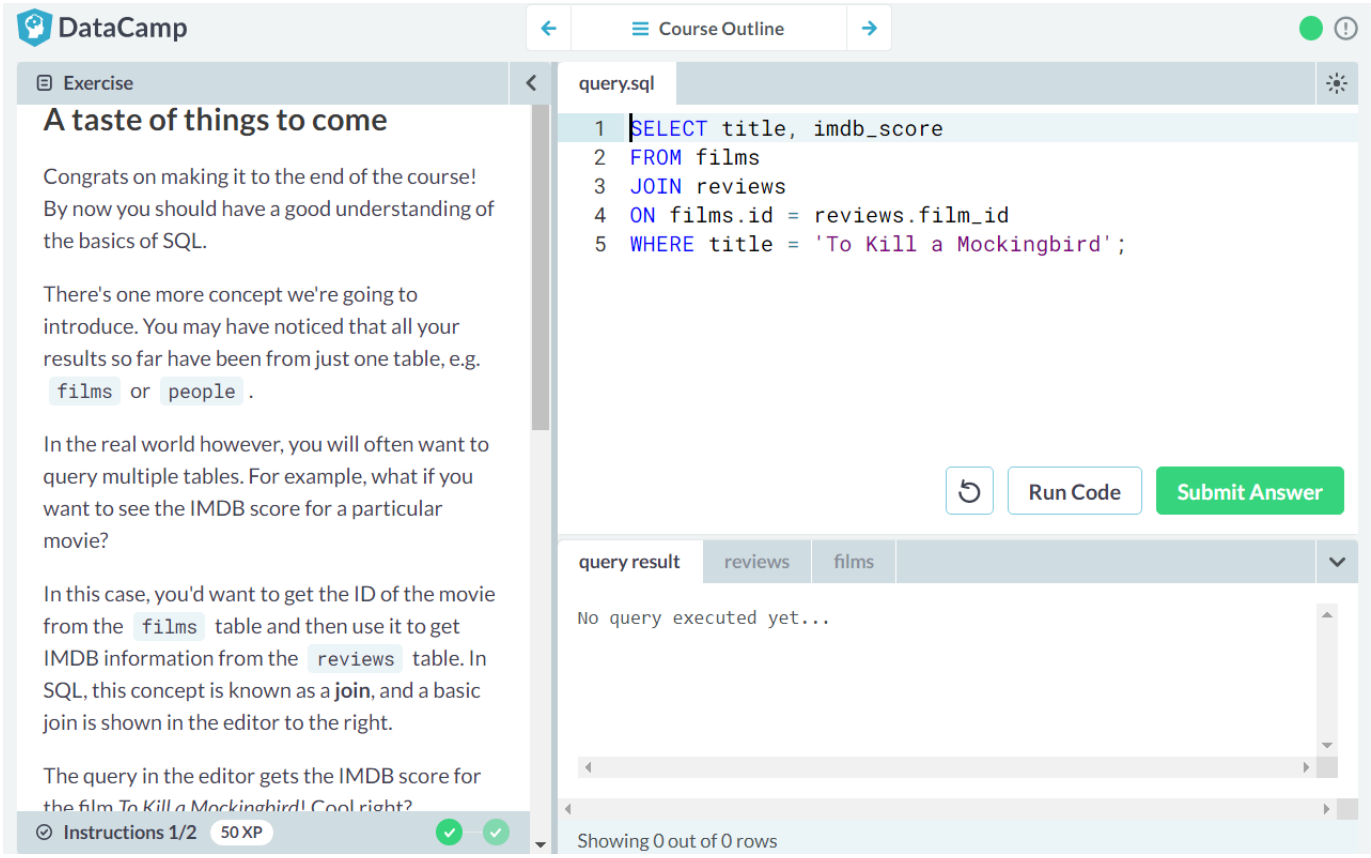

DataCamp
Course Outline
Exercise
All together now
Time to practice using ORDER BY , GROUP BY and HAVING together.
Now you're going to write a query that returns the average budget and average gross earnings for films in each year after 1990, if the average budget is greater than $60 million.
This is going to be a big query, but you can handle it!
Instructions 3/6
1 XP
Remove the budget and gross columns, and group your results by release year.
Hint
query.sql
solution.sql
1 SELECT release_year
2 FROM films
3 WHERE release_year > 1990
4 GROUP BY release_year;
Run Code
Run Solution
query result
films
release_year
gross
budget
1991
869325
6000000
1991
38037513
20000000
1991
57504069
6000000
Showing 100 out of 4345 rows

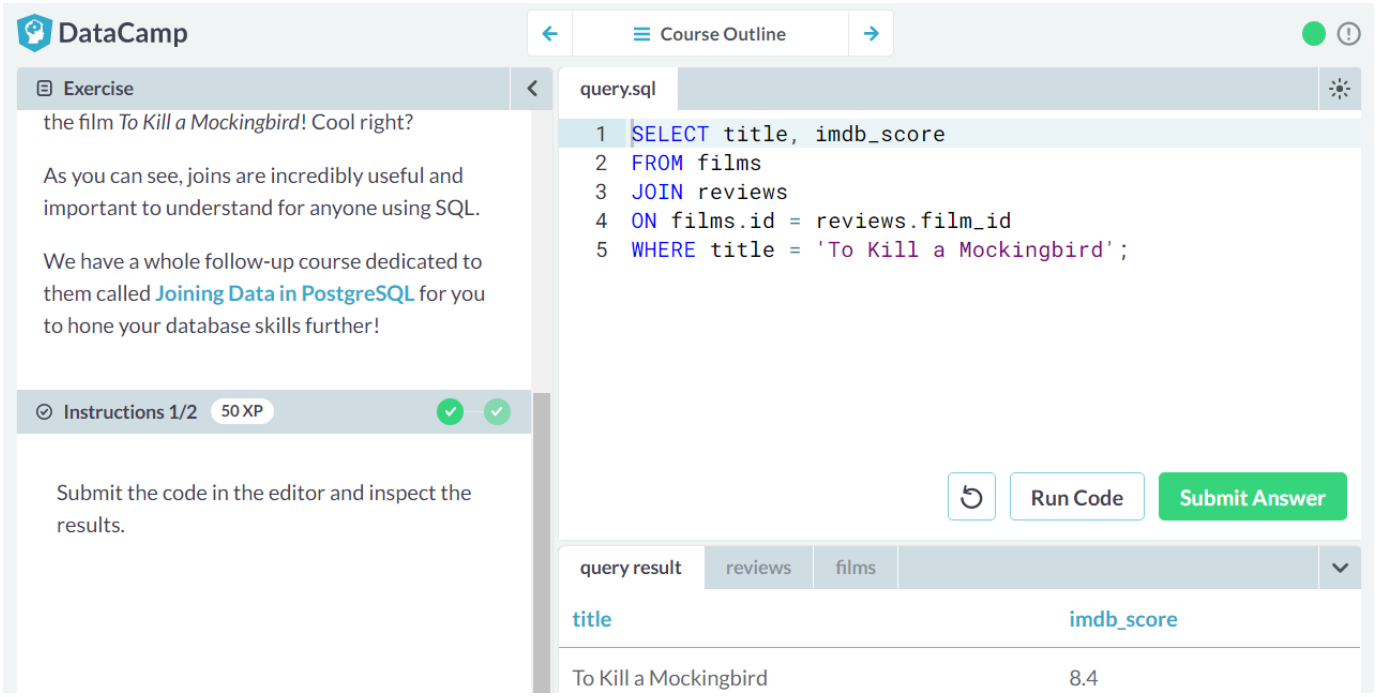

DataCamp
Course Outline
Exercise
All together now
Time to practice using ORDER BY , GROUP BY and HAVING together.
Now you're going to write a query that returns the average budget and average gross earnings for films in each year after 1990, if the average budget is greater than $60 million.
This is going to be a big query, but you can handle it!
Instructions 4/6
1 XP
Modify your query to include the average budget and average gross earnings for the results you have so far. Alias the average budget as avg_budget ; alias the average gross earnings as avg_gross .
query.sql
solution.sql
1 SELECT release_year, AVG(budget) AS avg_budget, AVG (gross) AS avg_gross
2 FROM films
3 WHERE release_year > 1990
4 GROUP BY release_year;
Run Code
Run Solution
query result
films
release_year
2000
2013
2007
Showing 26 out of 26 rows


DataCamp
Course Outline
Exercise
All together now
Time to practice using ORDER BY , GROUP BY and HAVING together.
Now you're going to write a query that returns the average budget and average gross earnings for films in each year after 1990, if the average budget is greater than $60 million.
This is going to be a big query, but you can handle it!
Instructions 5/6
1 XP
Modify your query so that only years with an average budget of greater than $60 million are included.
query.sql
solution.sql
1 SELECT release_year, AVG(budget) AS avg_budget, AVG (gross) AS avg_gross
2 FROM films
3 WHERE release_year > 1990
4 GROUP BY release_year
5 HAVING AVG(budget) > 60000000;
Run Code
Run Solution
query result
films
release_year
avg_budget
avg_gross
2000
34931375.757575757576
42172627.580838323353
2013
40519044.915492957746
56158357.775401069519
2007
35271131.147540983607
46267501.022346368715
Showing 26 out of 26 rows


DataCamp
Course Outline
Exercise
Now you're going to write a query that returns the average budget and average gross earnings for films in each year after 1990, if the average budget is greater than $60 million.
This is going to be a big query, but you can handle it!
Instructions 6/6
1 XP
Finally, modify your query to order the results from highest average gross earnings to lowest.
Hint
SELECT ___, AVG(budget) AS avg_budget, AVG(gross) AS avg_gross
FROM ___
WHERE ___ > ___
GROUP BY ___
query.sql
solution.sql
1 SELECT release_year, AVG(budget) AS avg_budget, AVG (gross) AS avg_gross
2 FROM films
3 WHERE release_year > 1990
4 GROUP BY release_year
5 HAVING AVG(budget) > 60000000
6 ORDER BY avg_gross DESC;
Run Code
Run Solution
query result
films
release_year
avg_budget
avg_gross
2005
70323938.231527093596
41159143.290640394089
2006
93968929.577464788732
39237855.953703703704
Showing 2 out of 2 rows


DataCamp
Course Outline
Exercise
keyword to limit the number of rows returned
Instructions
100 XP
Get the country, average budget, and average gross take of countries that have made more than 10 films. Order the result by country name, and limit the number of results displayed to 5. You should alias the averages as avg_budget and avg_gross respectively.
query.sql
1 -- select country, average budget, average gross
2 SELECT country, AVG(budget) AS avg_budget, AVG(gross) AS avg_gross
3 -- from the films table
4 FROM films
5 -- group by country
6 GROUP BY country
7 -- where the country has more than 10 titles
8 HAVING COUNT(title)>10
9 -- order by country
10 ORDER BY country
11 -- limit to only show 5 results
12 LIMIT(5)


DataCamp
Course Outline
Exercise
A taste of things to come
Congrats on making it to the end of the course! By now you should have a good understanding of the basics of SQL.
There's one more concept we're going to introduce. You may have noticed that all your results so far have been from just one table, e.g. films or people .
In the real world however, you will often want to query multiple tables. For example, what if you want to see the IMDB score for a particular movie?
In this case, you'd want to get the ID of the movie from the films table and then use it to get IMDB information from the reviews table. In SQL, this concept is known as a join, and a basic join is shown in the editor to the right.
The query in the editor gets the IMDB score for the film To Kill a Mockingbird! Cool right?
Instructions 1/2
50 XP
query.sql
1 SELECT title, imdb_score
2 FROM films
3 JOIN reviews
4 ON films.id = reviews.film_id
5 WHERE title = 'To Kill a Mockingbird';
Run Code
Submit Answer
query result
reviews
films
No query executed yet...
Showing 0 out of 0 rows


DataCamp
Course Outline
Exercise
the film To Kill a Mockingbird! Cool right?
As you can see, joins are incredibly useful and important to understand for anyone using SQL.
We have a whole follow-up course dedicated to them called Joining Data in PostgreSQL for you to hone your database skills further!
Instructions 1/2
50 XP
Submit the code in the editor and inspect the results.
query.sql
1 SELECT title, imdb_score
2 FROM films
3 JOIN reviews
4 ON films.id = reviews.film_id
5 WHERE title = 'To Kill a Mockingbird';
Run Code
Submit Answer
query result
reviews
films
title
imdb_score
To Kill a Mockingbird
8.4