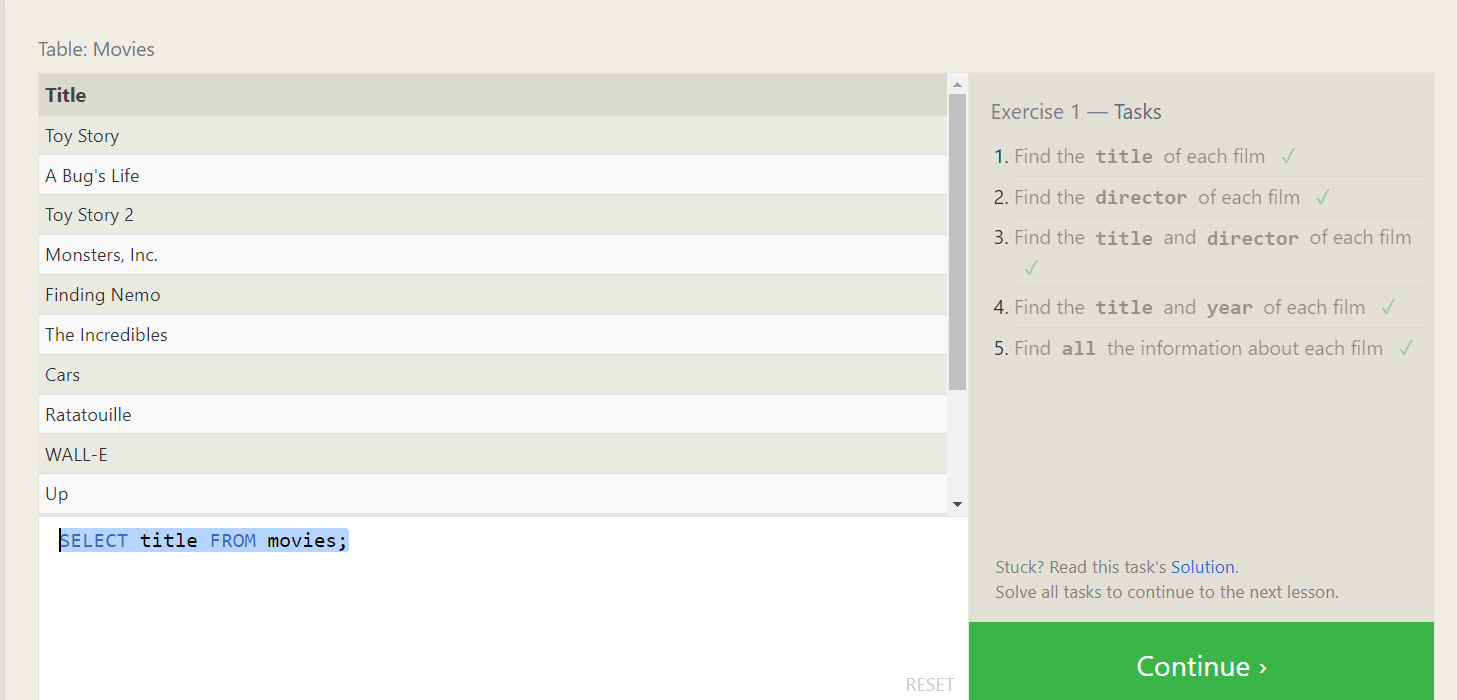
**SQL Lesson 1: SELECT queries 101**

**Exercise 1 — Tasks**

1. Find the title of each film

SELECT title FROM movies;



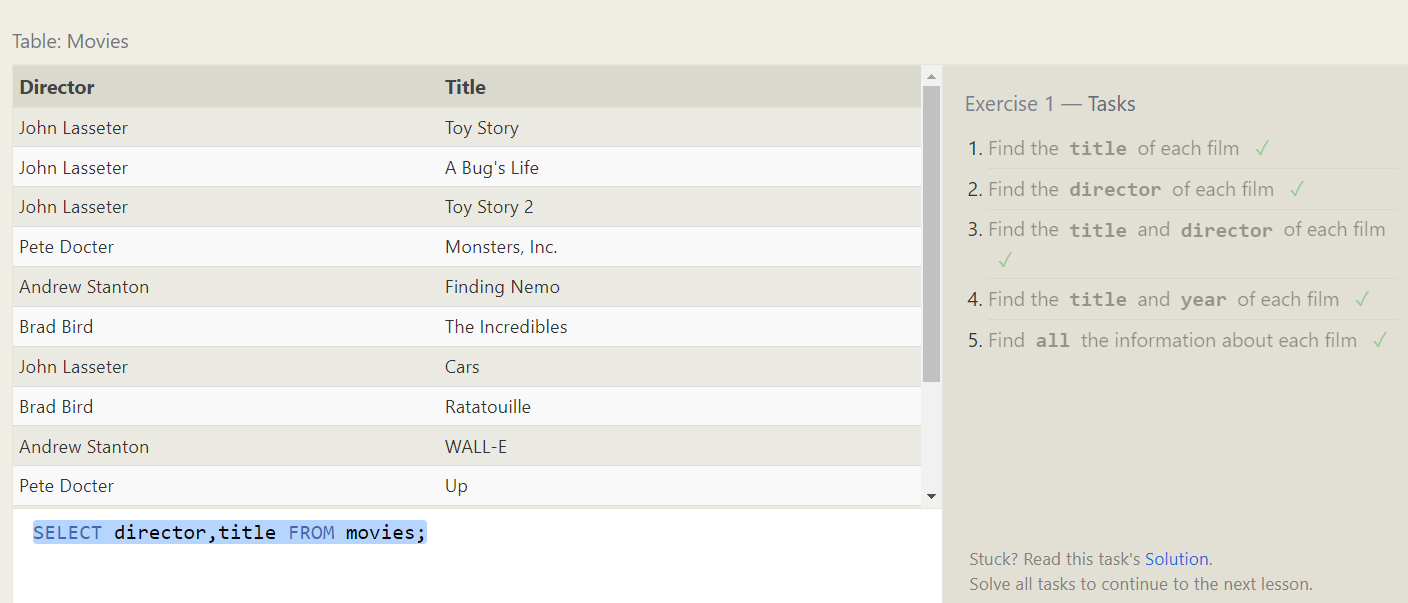
1. Find the director of each film

SELECT director FROM movies;



1. Find the title and director of each film

SELECT director,title FROM movies;



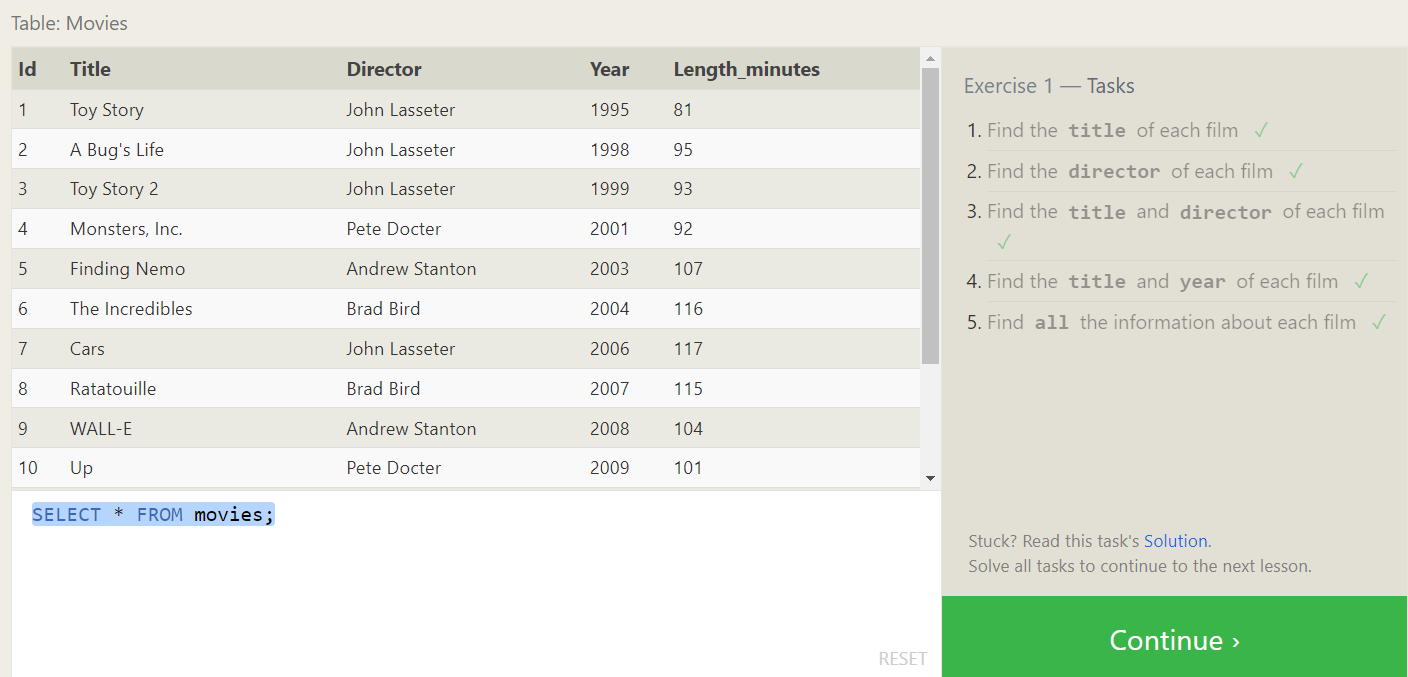
1. Find the title and year of each film

SELECT year,title FROM movies;



1. Find **all** the information about each film

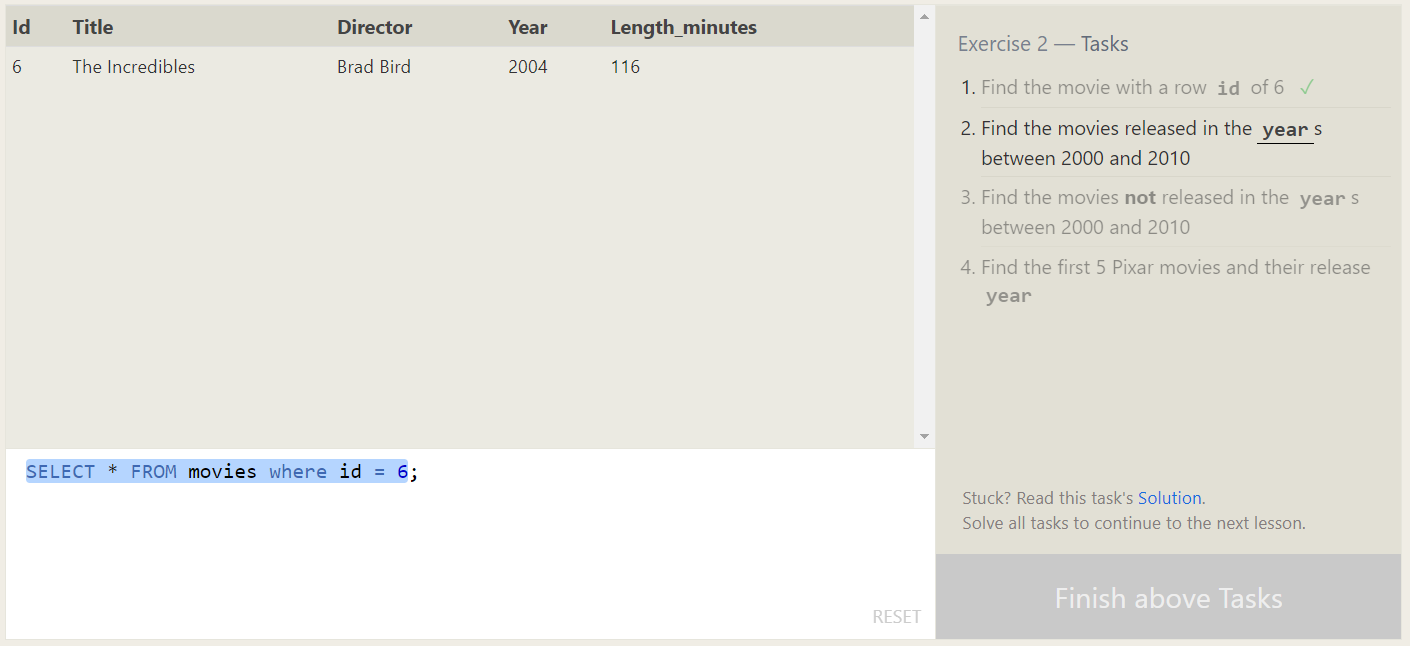
SELECT \* FROM movies;



**SQL Lesson 2: Queries with constraints**

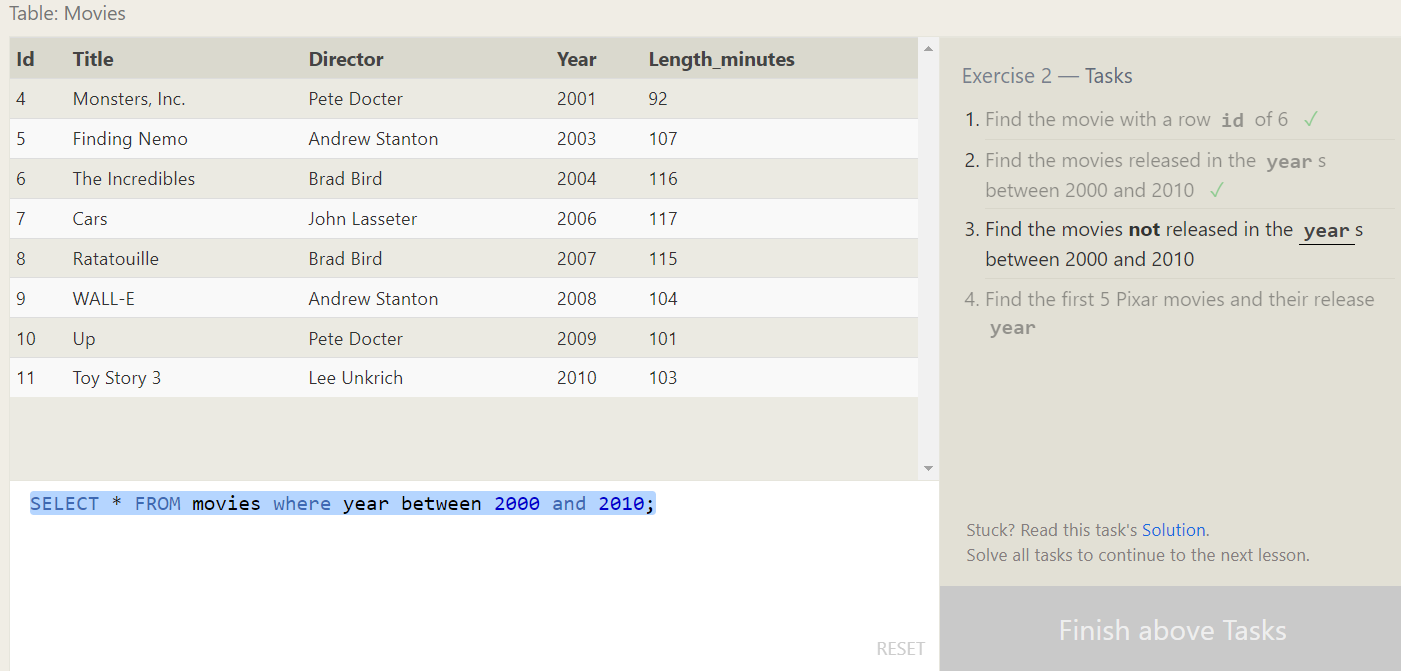
1. Find the movie with a row id of 6

SELECT \* FROM movies where id = 6;



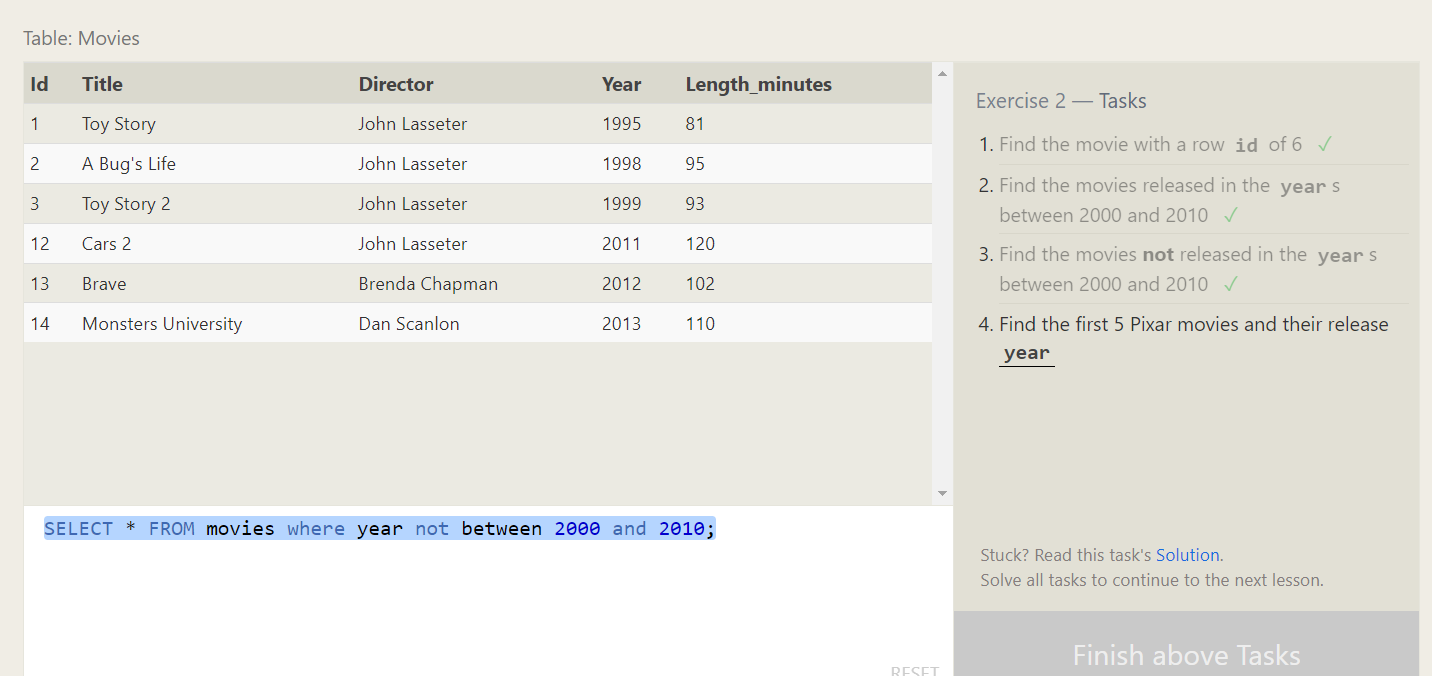
1. Find the movies released in the **year**s between 2000 and 2010

SELECT \* FROM movies where year between 2000 and 2010;



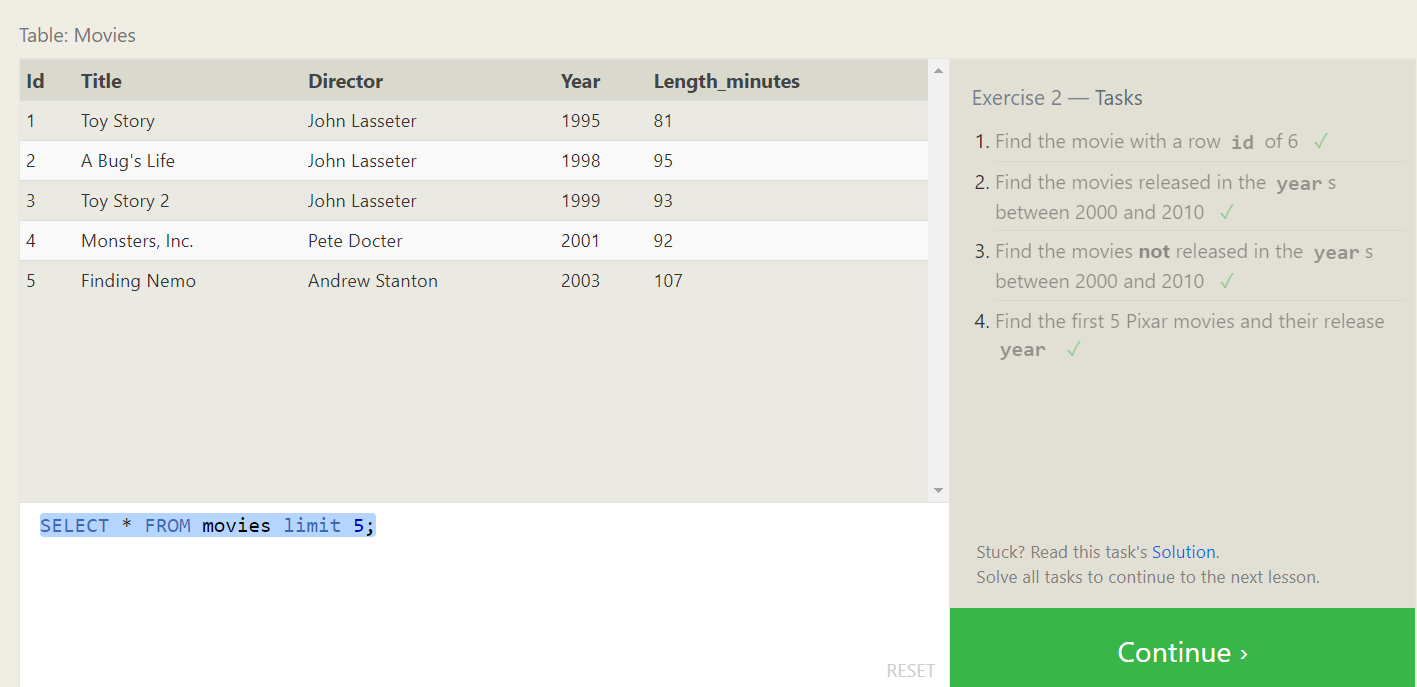
1. Find the movies **not** released in the **year**s between 2000 and 2010

SELECT \* FROM movies where year not between 2000 and 2010;



1. Find the first 5 Pixar movies and their release **year**

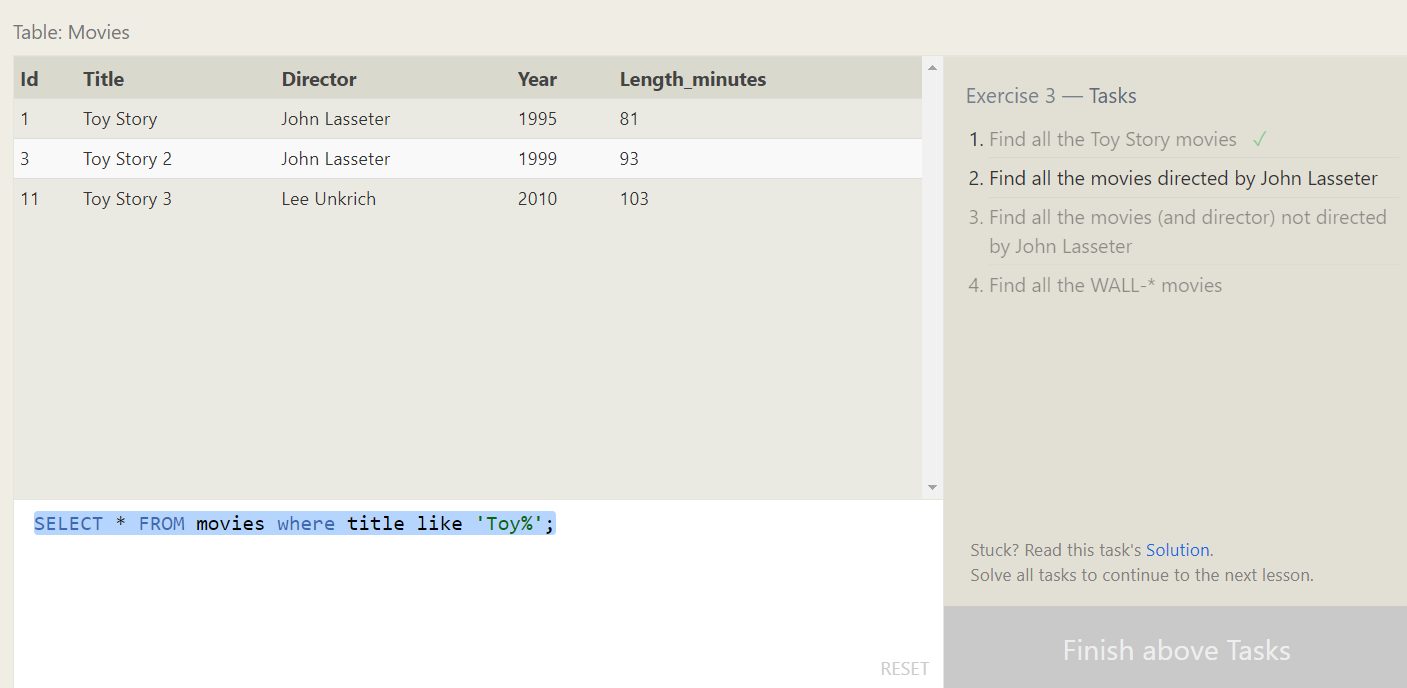
SELECT \* FROM movies limit 5;



**SQL Lesson 3: Queries with constraints**

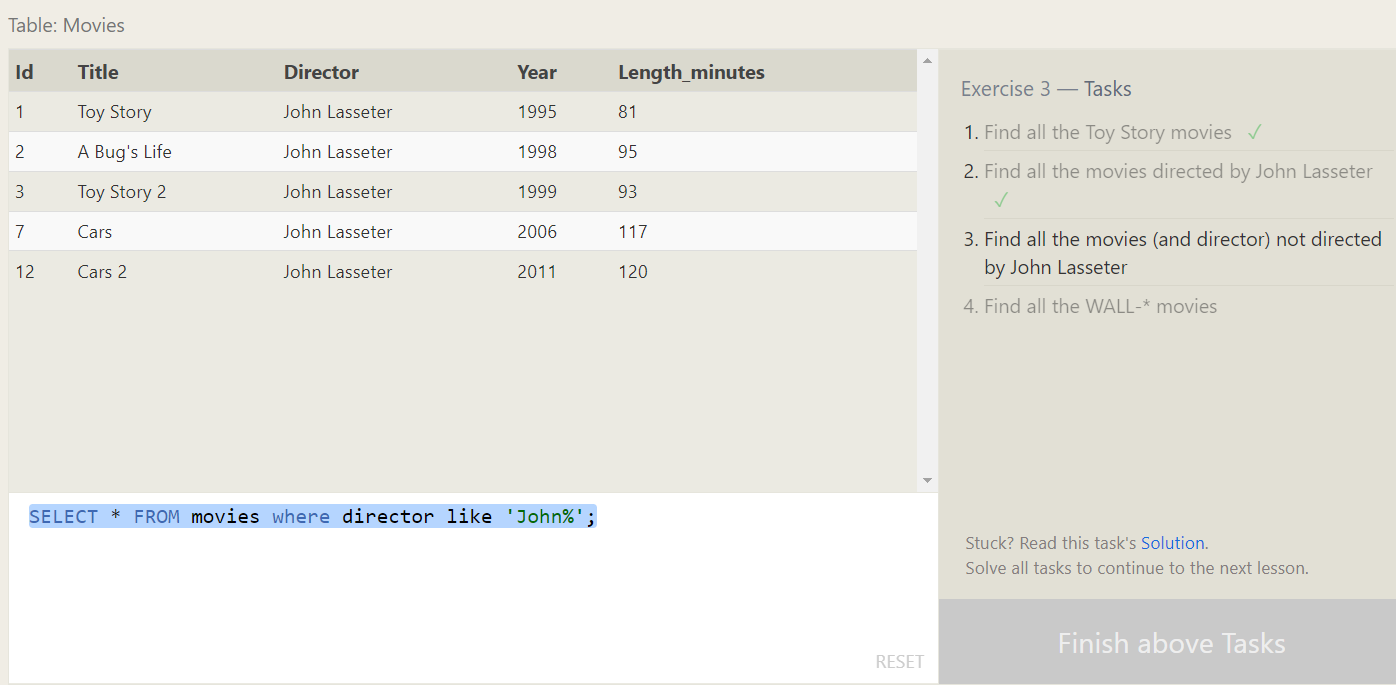
1. Find all the Toy Story movies

SELECT \* FROM movies where title like 'Toy%';



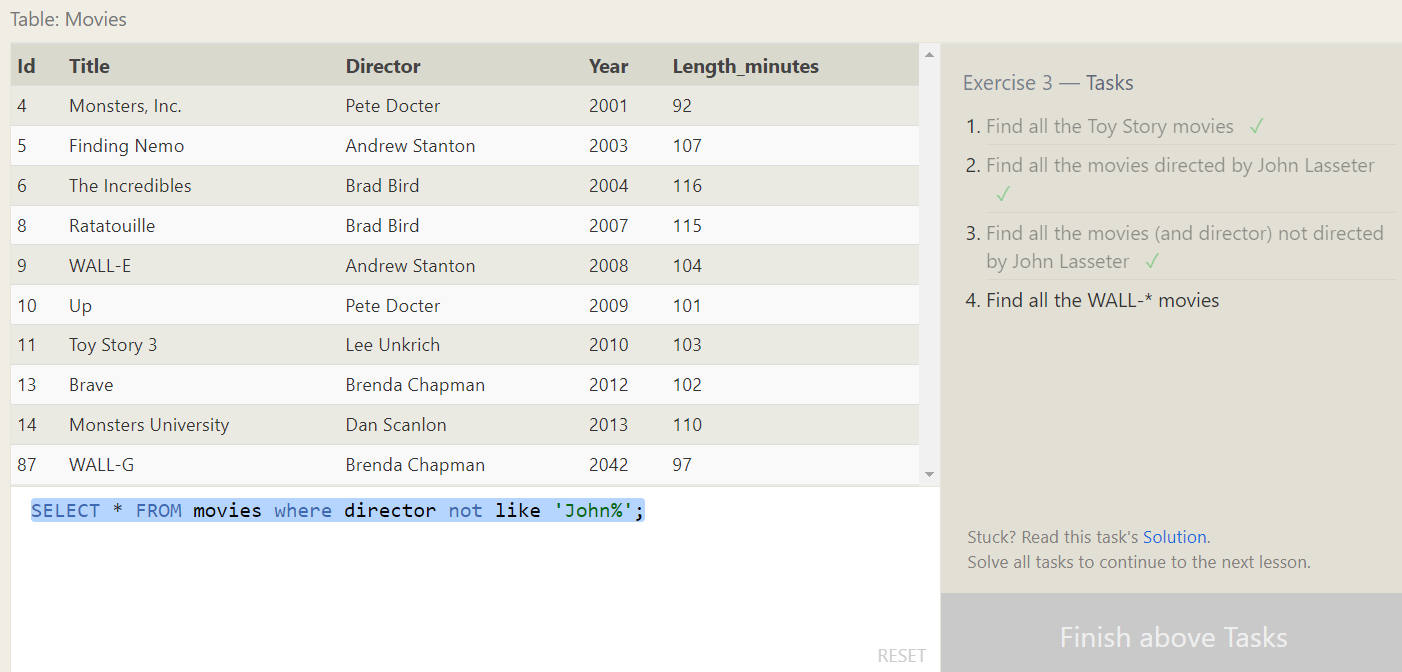
1. Find all the movies directed by John Lasseter

SELECT \* FROM movies where director like 'John%';

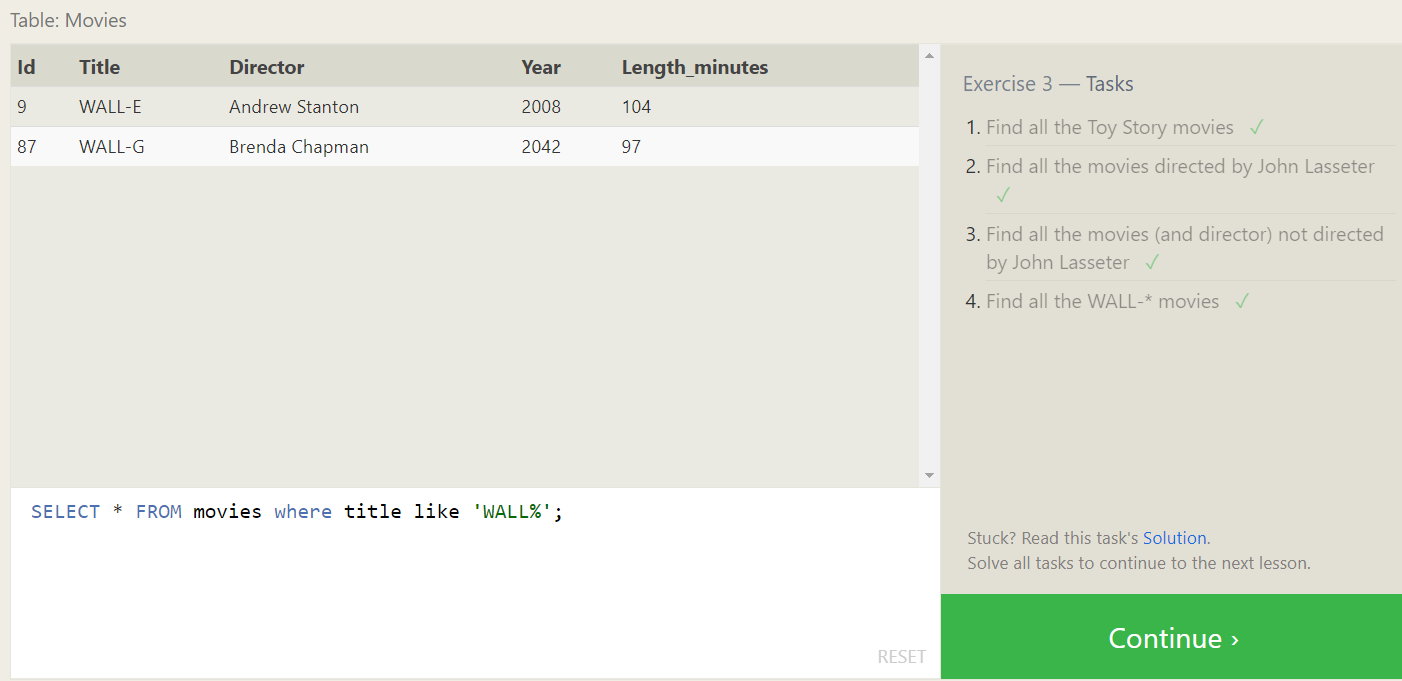


1. Find all the movies (and director) not directed by John Lasseter

SELECT \* FROM movies where director not like 'John%';



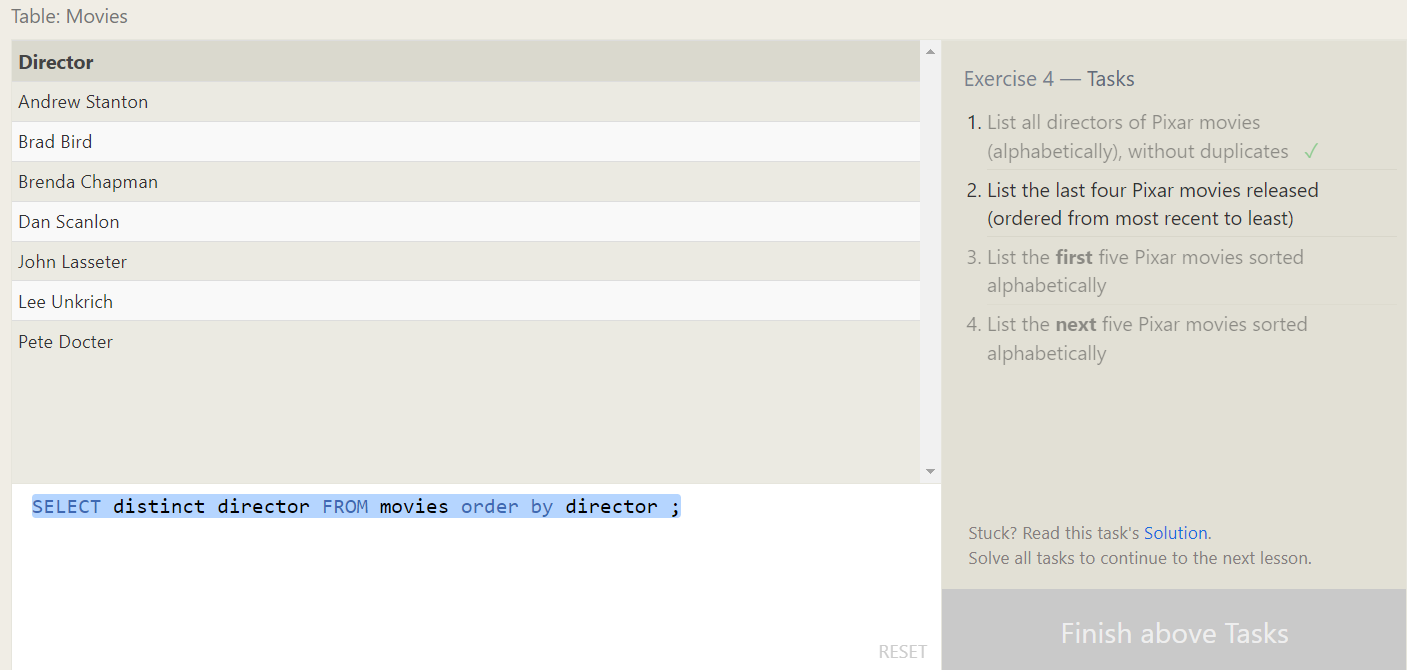
1. Find all the WALL-\* movies



SQL Lesson 4: Filtering and sorting Query results

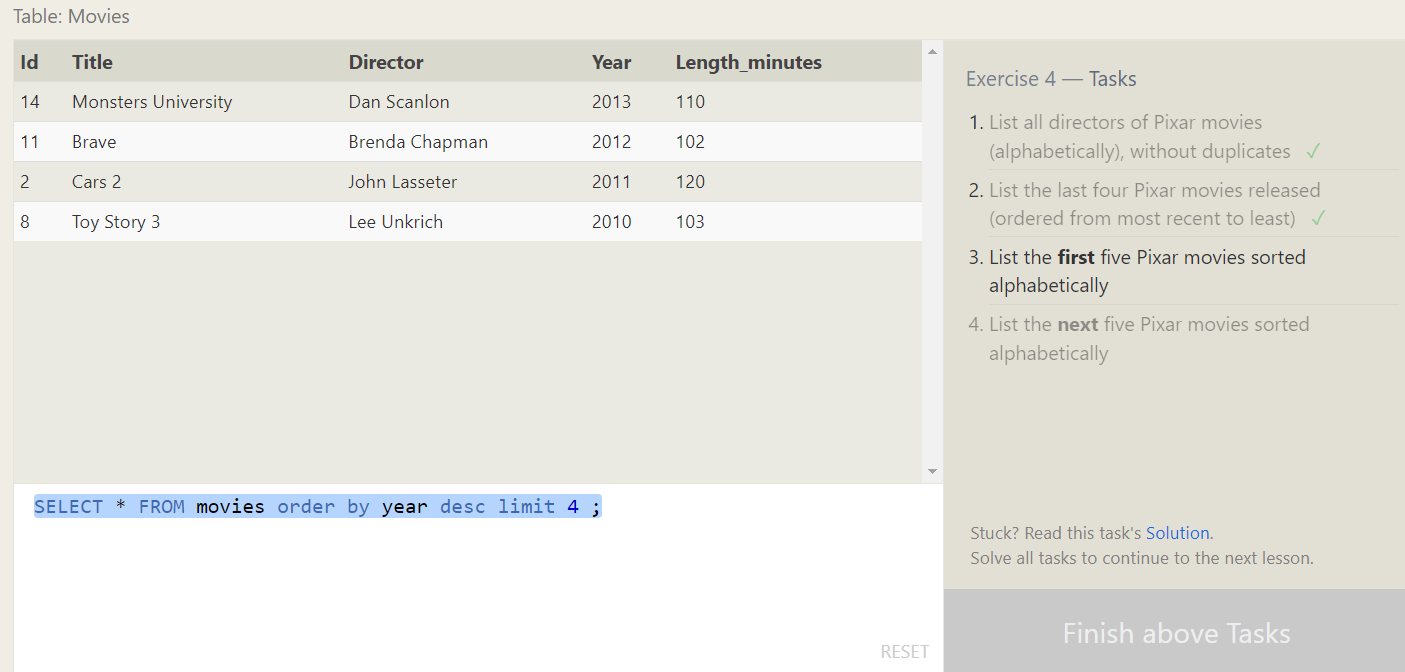
1. List all directors of Pixar movies (alphabetically), without duplicates

SELECT distinct director FROM movies order by director ;



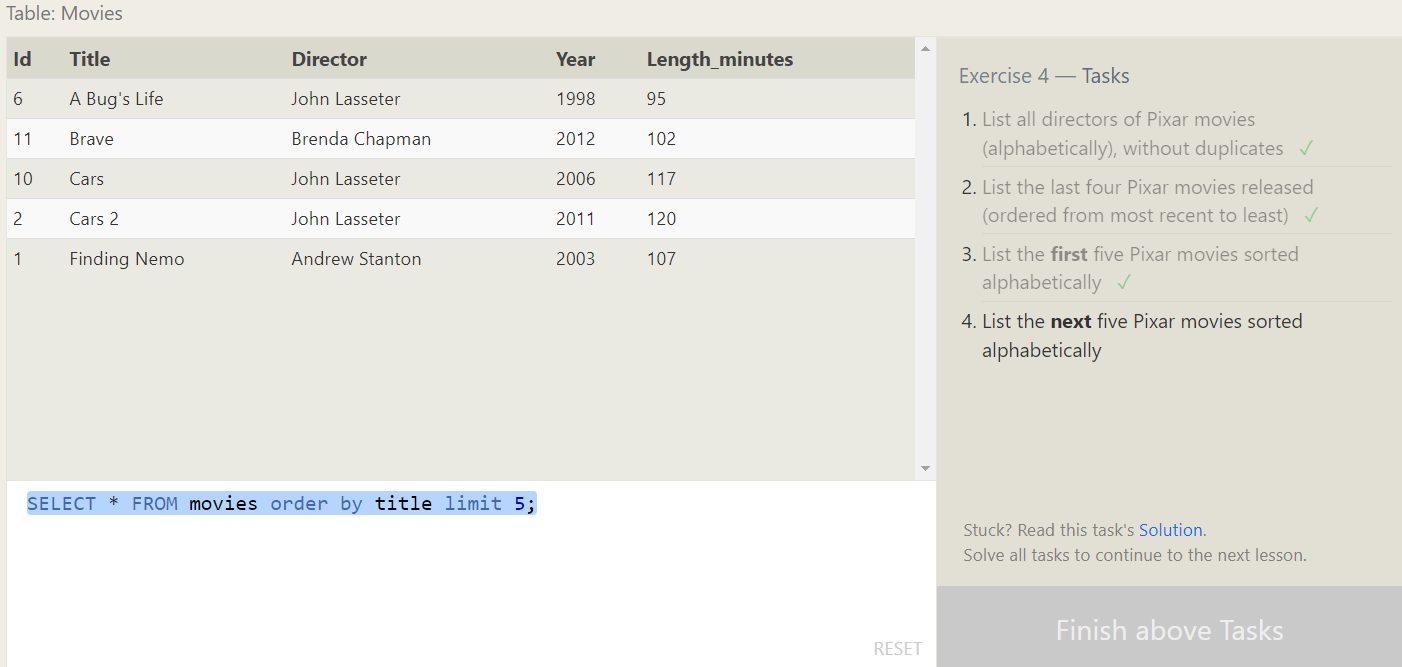
1. List the last four Pixar movies released (ordered from most recent to least)

SELECT \* FROM movies order by year desc limit 4 ;



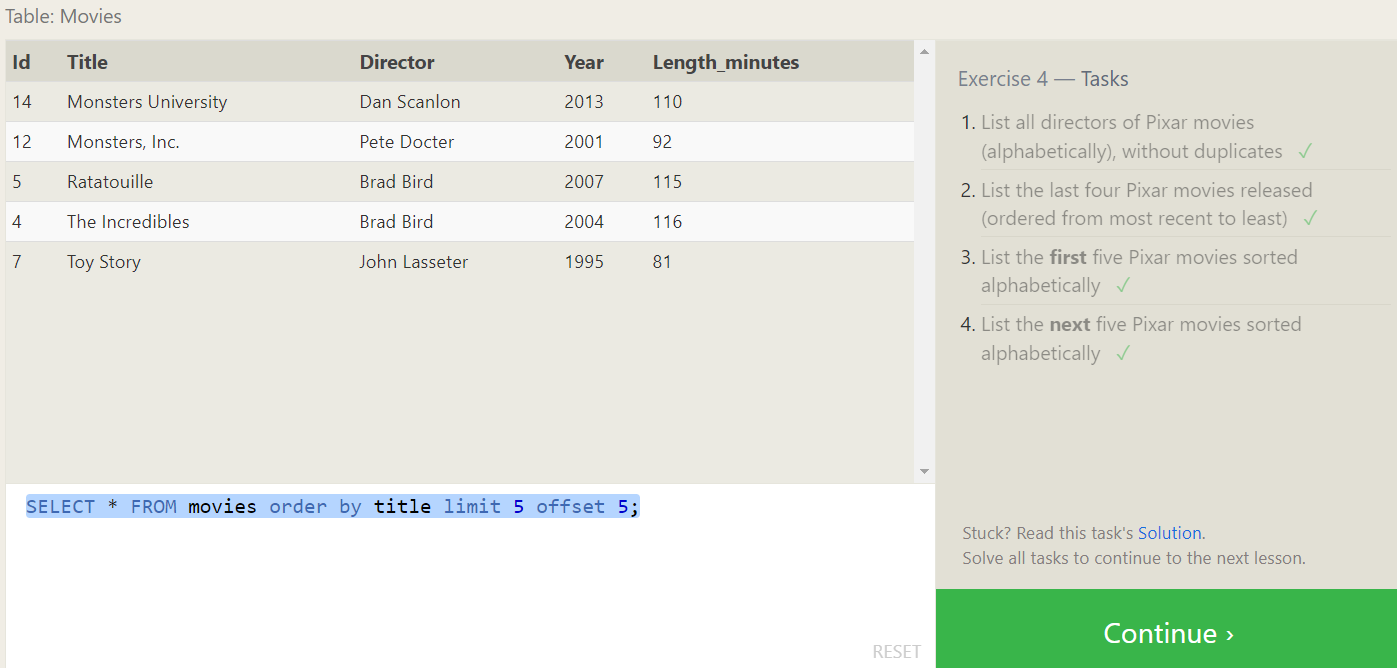
1. List the **first** five Pixar movies sorted alphabetically

SELECT \* FROM movies order by title limit 5;



1. List the **next** five Pixar movies sorted alphabetically

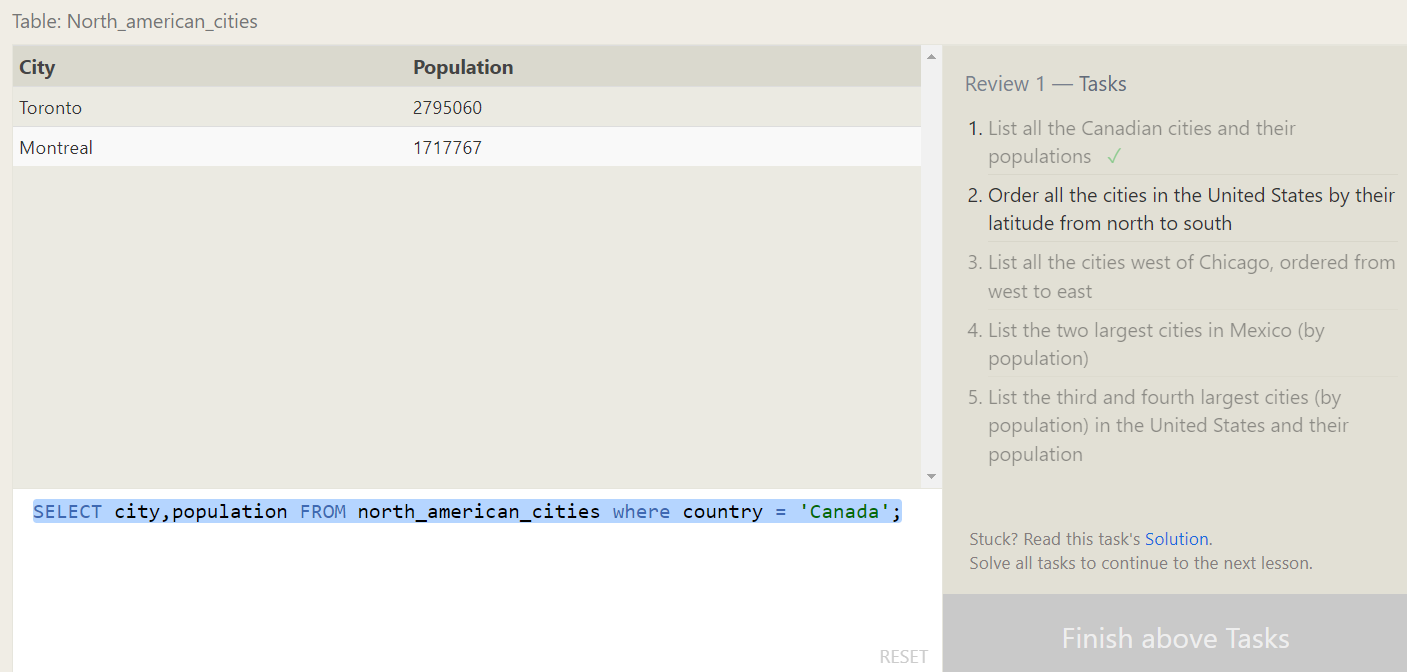
SELECT \* FROM movies order by title limit 5 offset 5;



SQL Review: Simple SELECT Queries

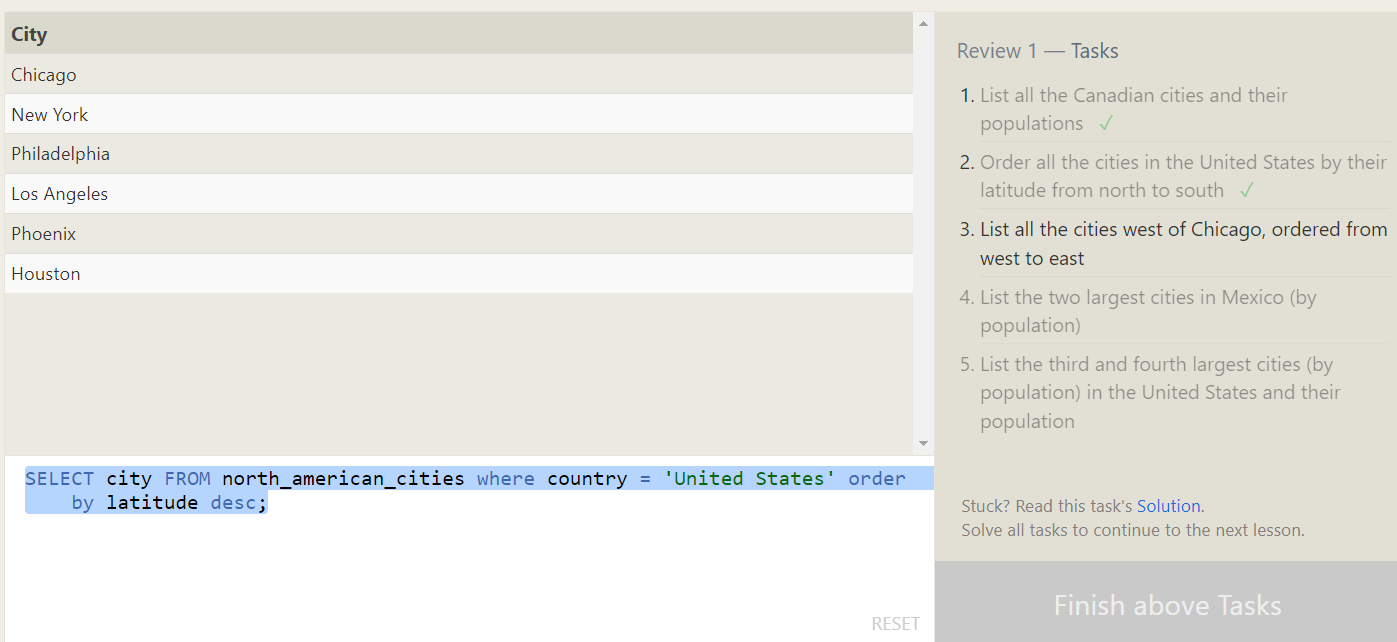
1. List all the Canadian cities and their populations

SELECT city,population FROM north\_american\_cities where country = 'Canada';



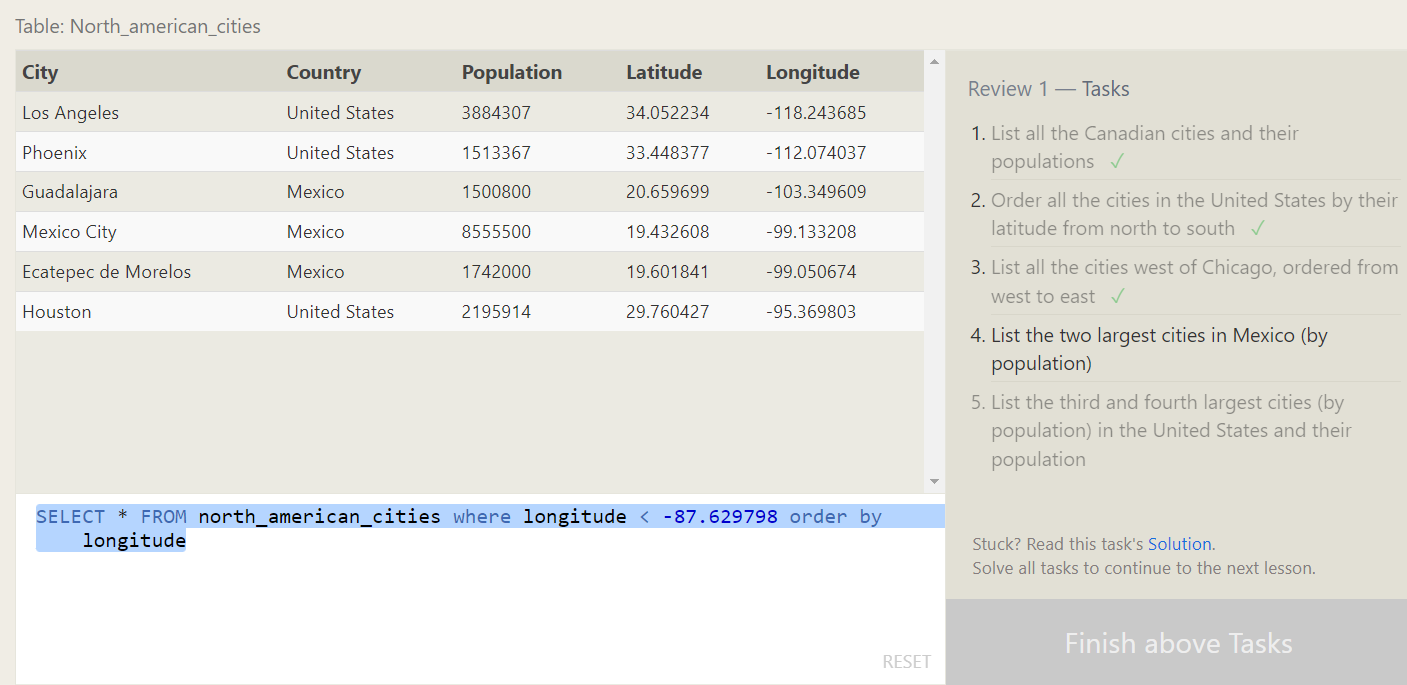
1. Order all the cities in the United States by their latitude from north to south

SELECT city FROM north\_american\_cities where country = 'United States' order by latitude desc;



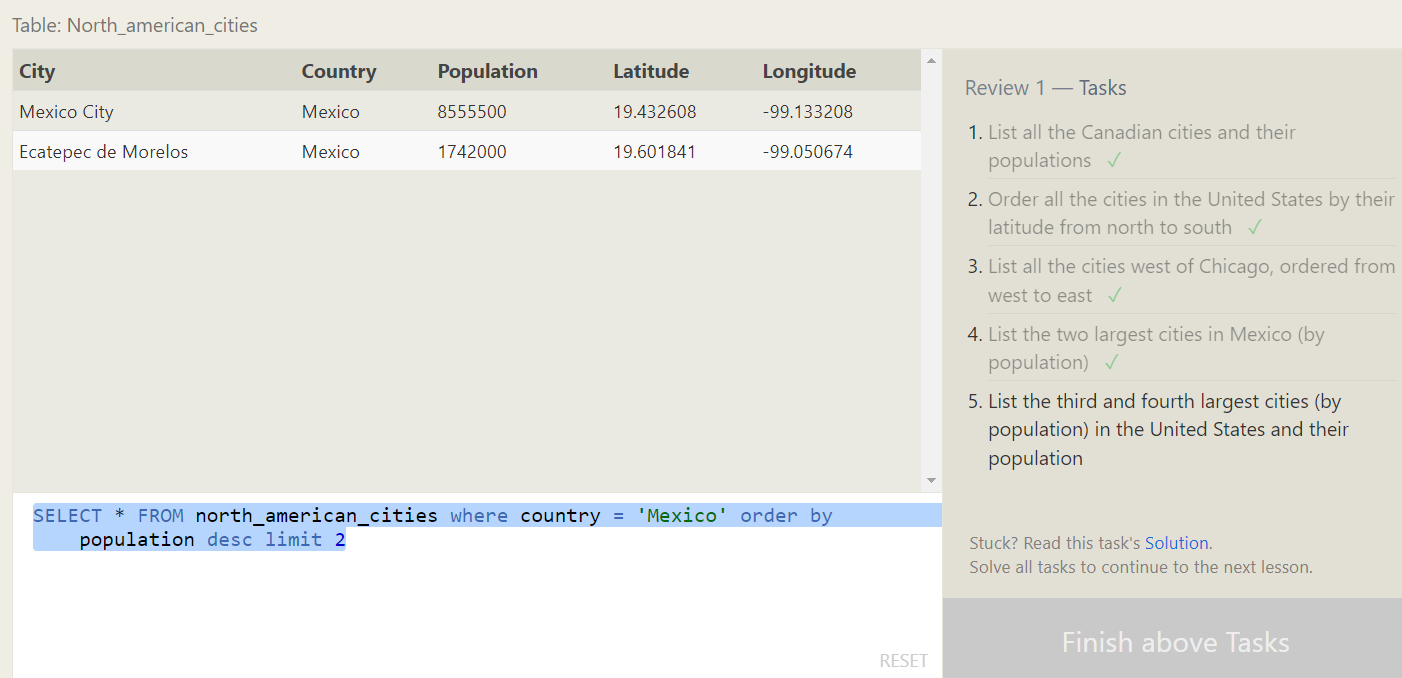
1. List all the cities west of Chicago, ordered from west to east

SELECT \* FROM north\_american\_cities where longitude < -87.629798 order by longitude



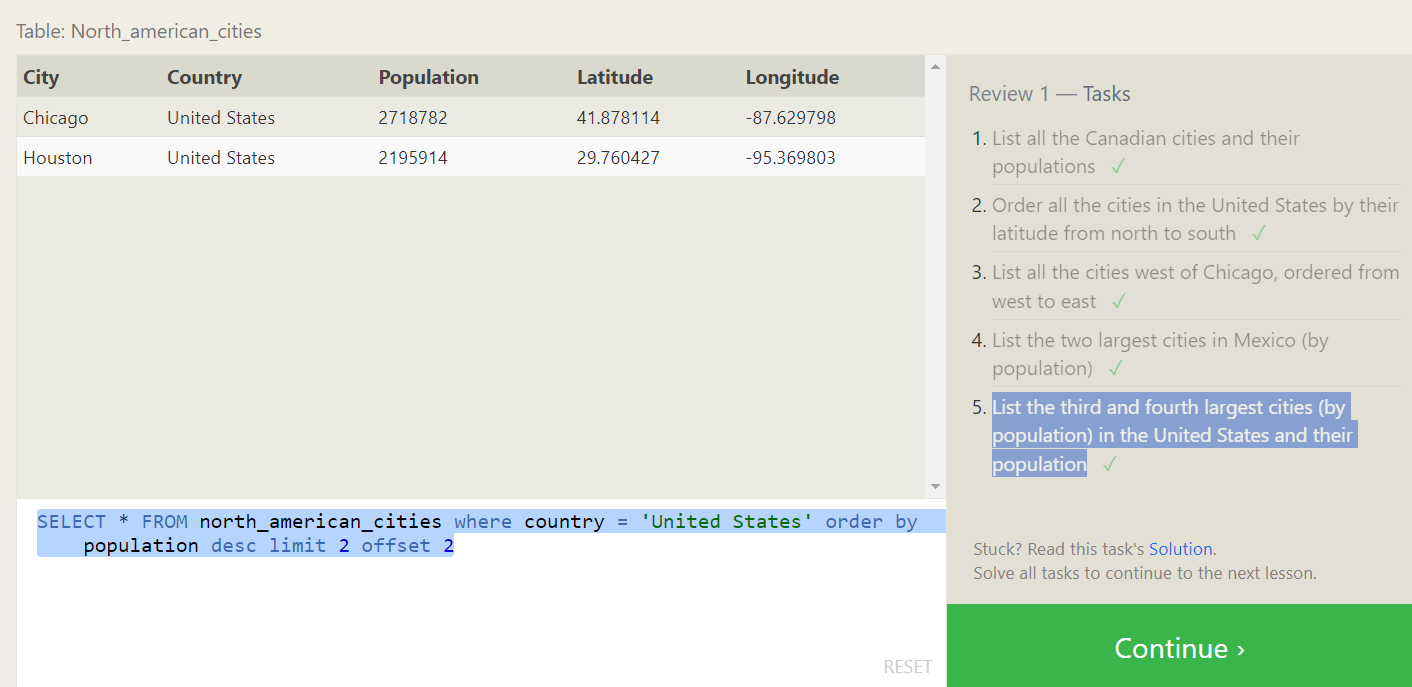
1. List the two largest cities in Mexico (by population)

SELECT \* FROM north\_american\_cities where country = 'Mexico' order by population desc limit 2



1. List the third and fourth largest cities (by population) in the United States and their population

SELECT \* FROM north\_american\_cities where country = 'United States' order by population desc limit 2 offset 2



**SQL Lesson 6: Multi-table queries with JOINs**

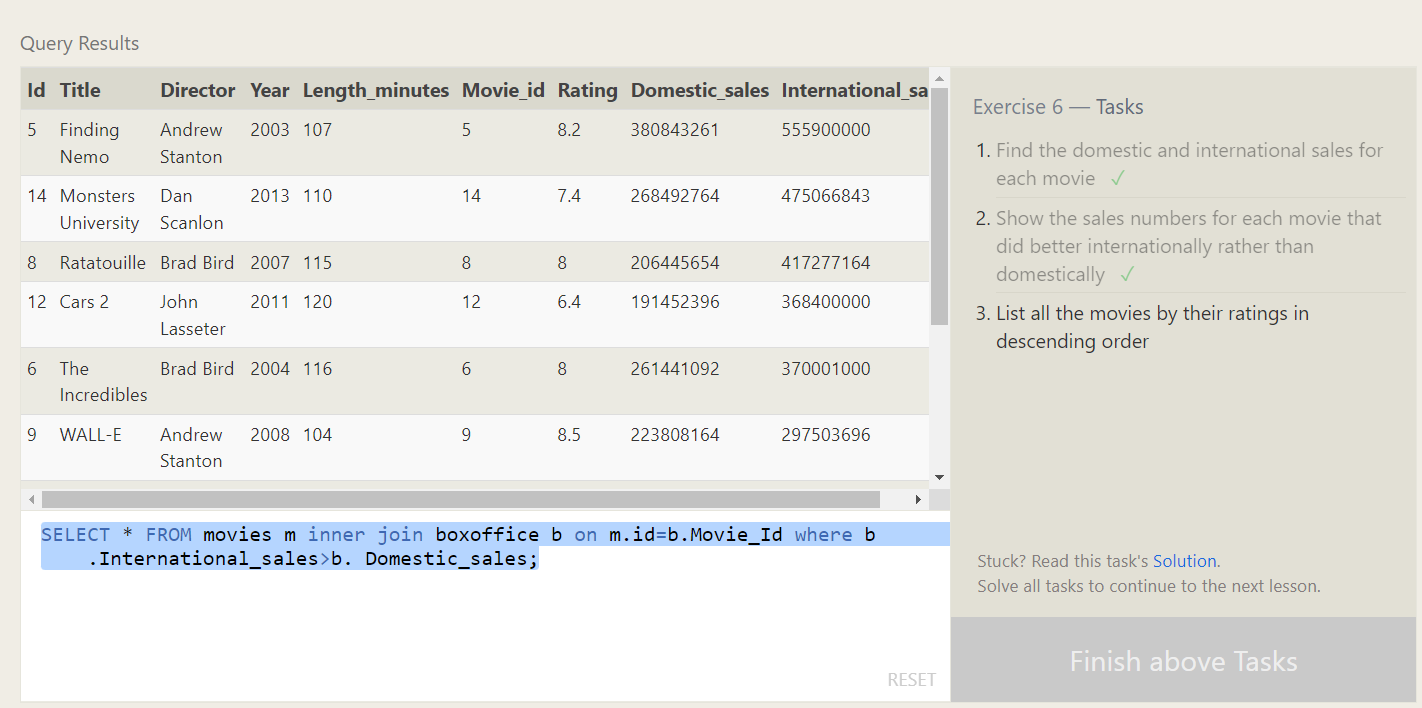
1. Find the domestic and international sales for each movie

SELECT \* FROM movies m inner join boxoffice b on m.id= b.movie\_id;



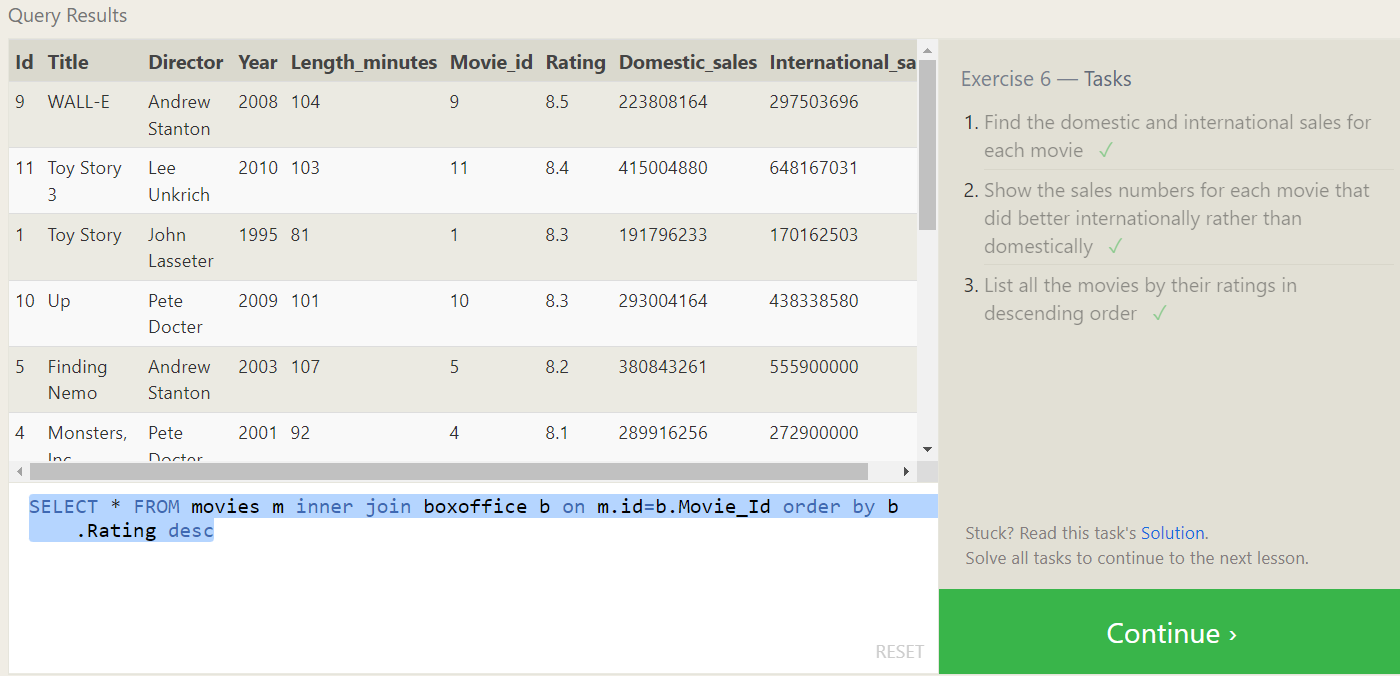
1. Show the sales numbers for each movie that did better internationally rather than domestically

SELECT \* FROM movies m inner join boxoffice b on m.id=b.Movie\_Id where b.International\_sales>b. Domestic\_sales;



1. List all the movies by their ratings in descending order

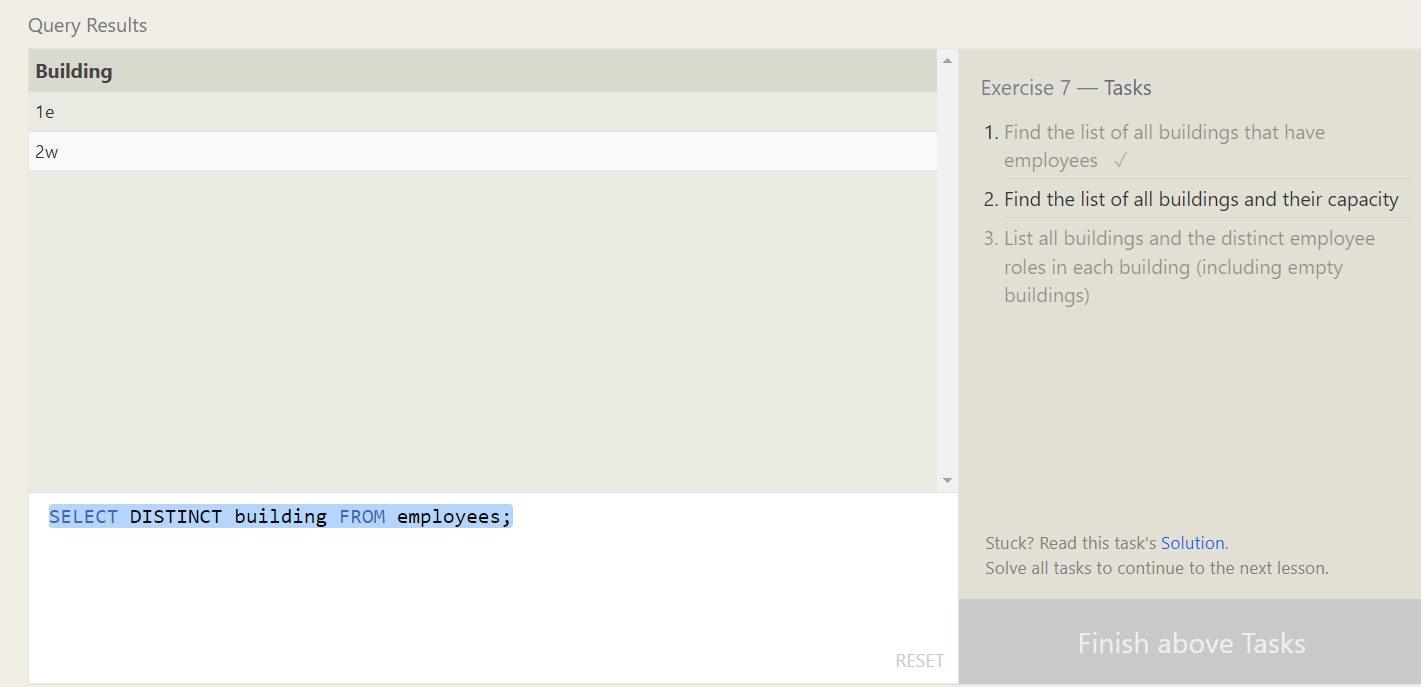
SELECT \* FROM movies m inner join boxoffice b on m.id=b.Movie\_Id order by b.Rating desc



**SQL Lesson 7: OUTER JOINs**

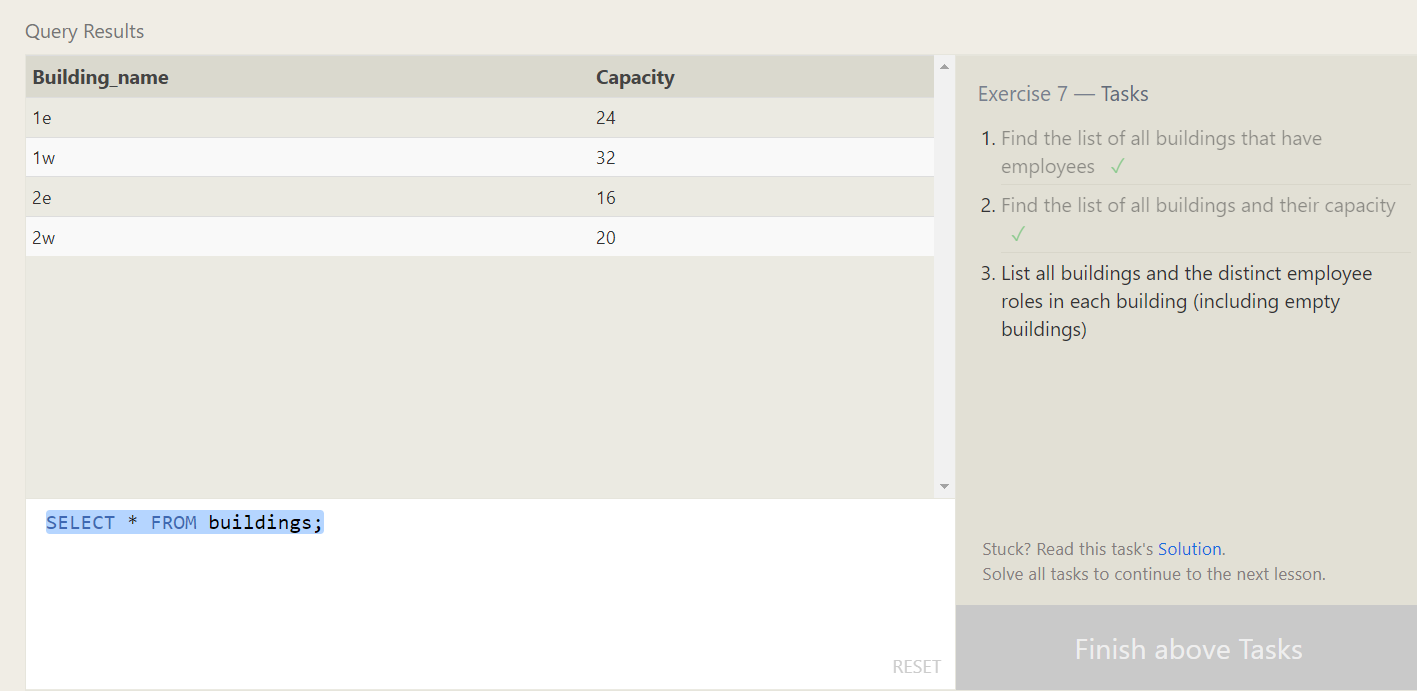
1. Find the list of all buildings that have employees

SELECT DISTINCT building FROM employees;



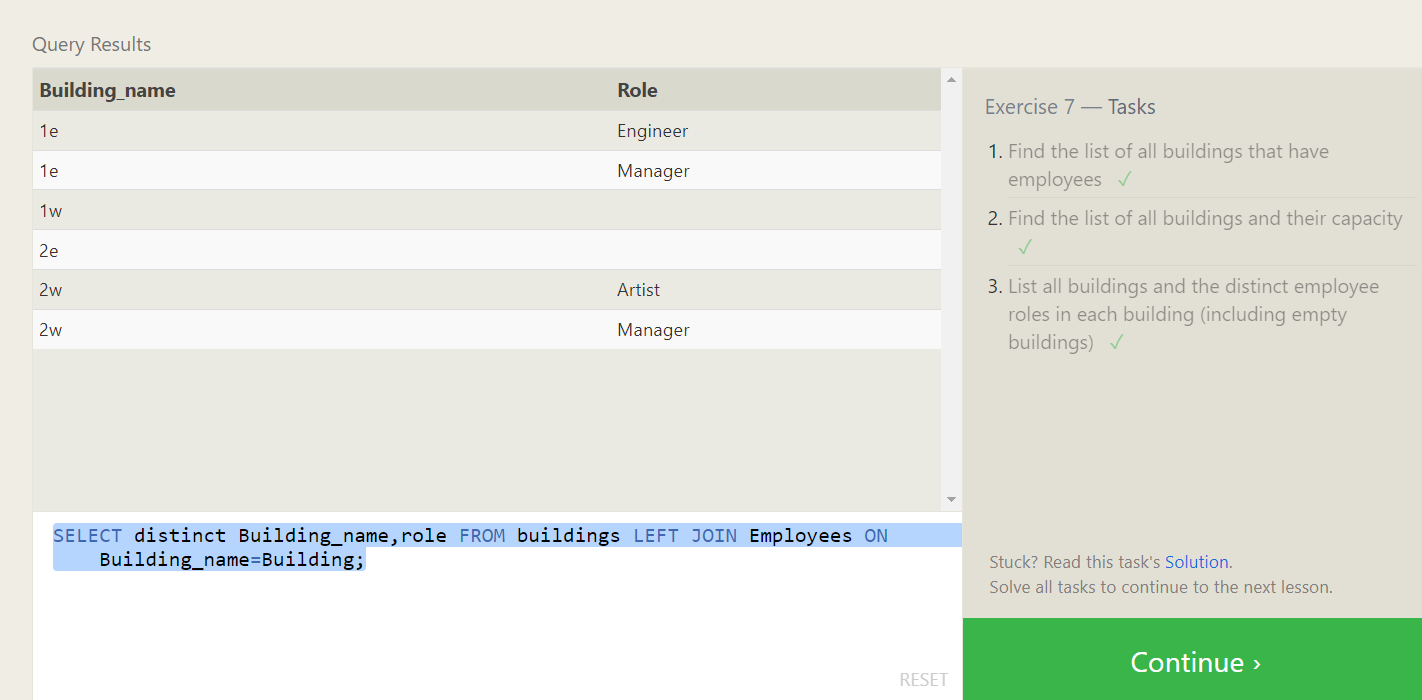
1. Find the list of all buildings and their capacity

SELECT \* FROM buildings;



1. List all buildings and the distinct employee roles in each building (including empty buildings)

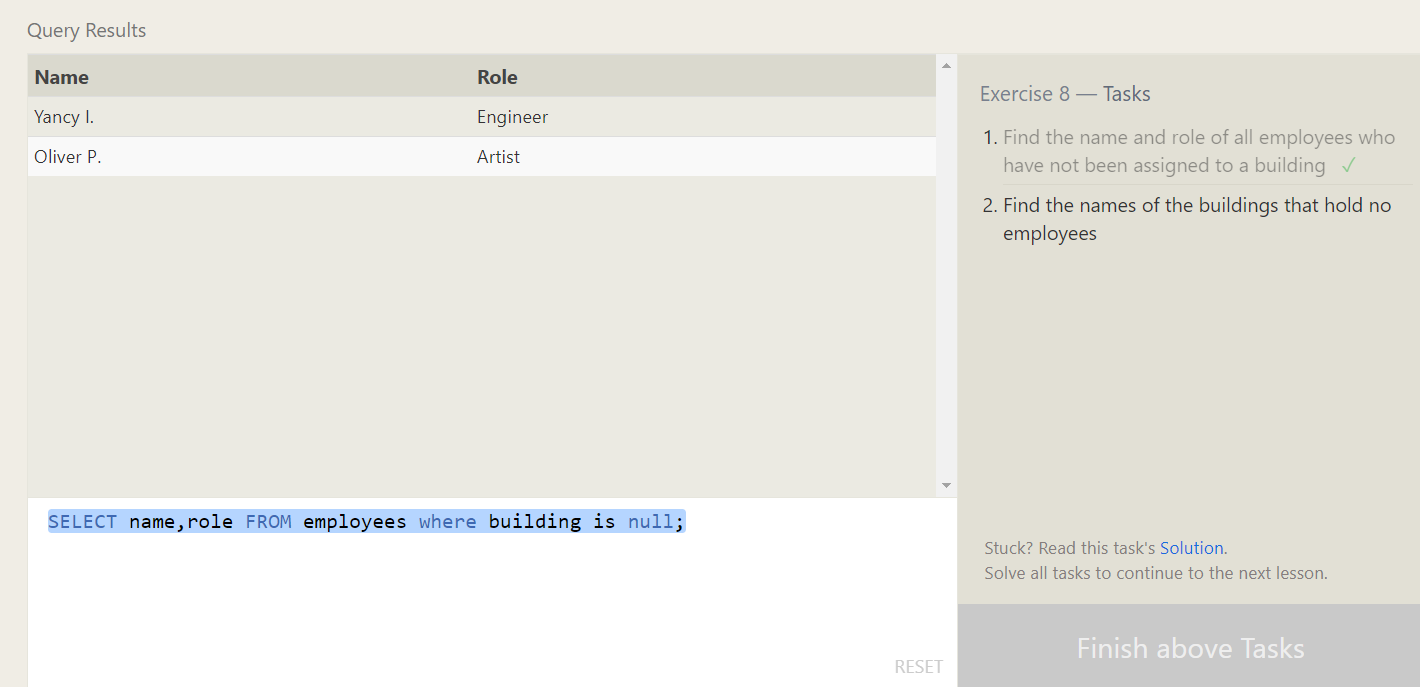
SELECT distinct Building\_name,role FROM buildings LEFT JOIN Employees ON Building\_name=Building;



**SQL Lesson 8: A short note on NULLs**

1. Find the name and role of all employees who have not been assigned to a building

SELECT name,role FROM employees where building is null;



1. Find the names of the buildings that hold no employees

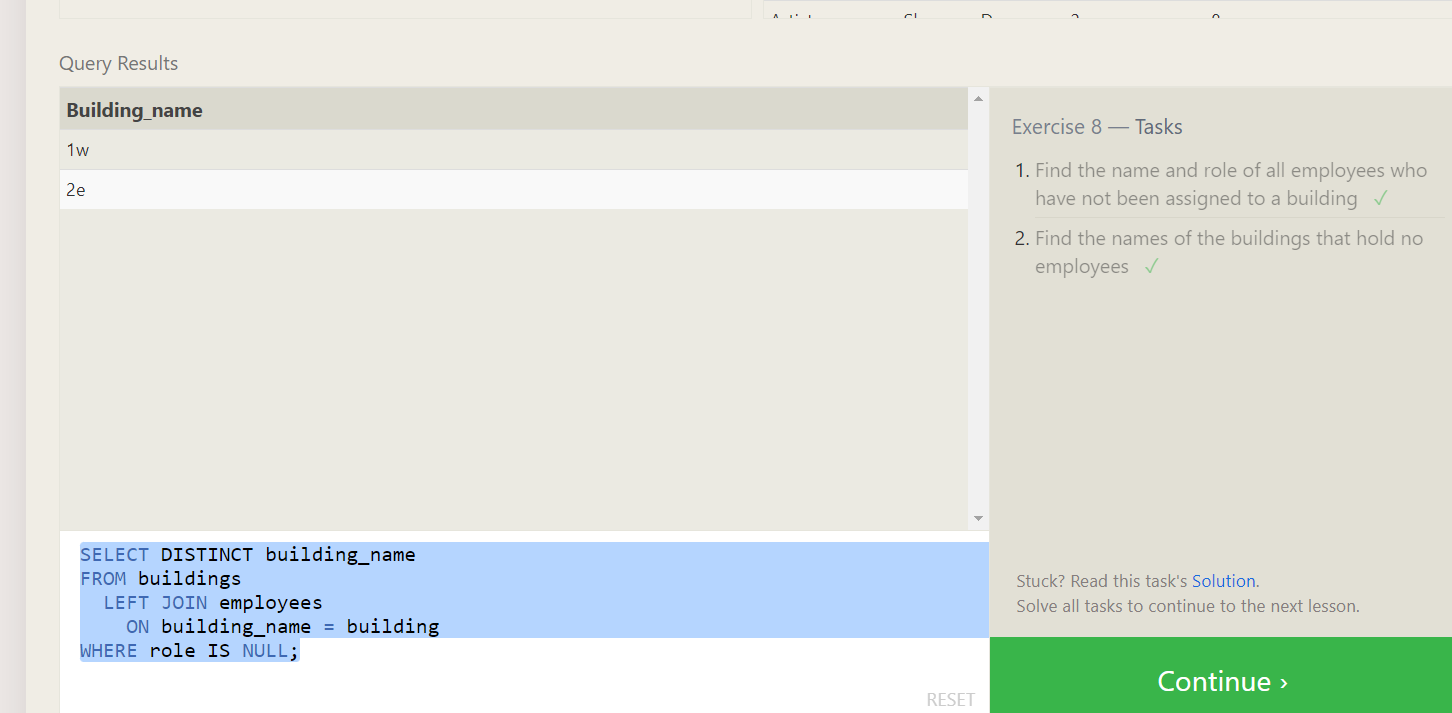
SELECT DISTINCT building\_name

FROM buildings

LEFT JOIN employees

ON building\_name = building

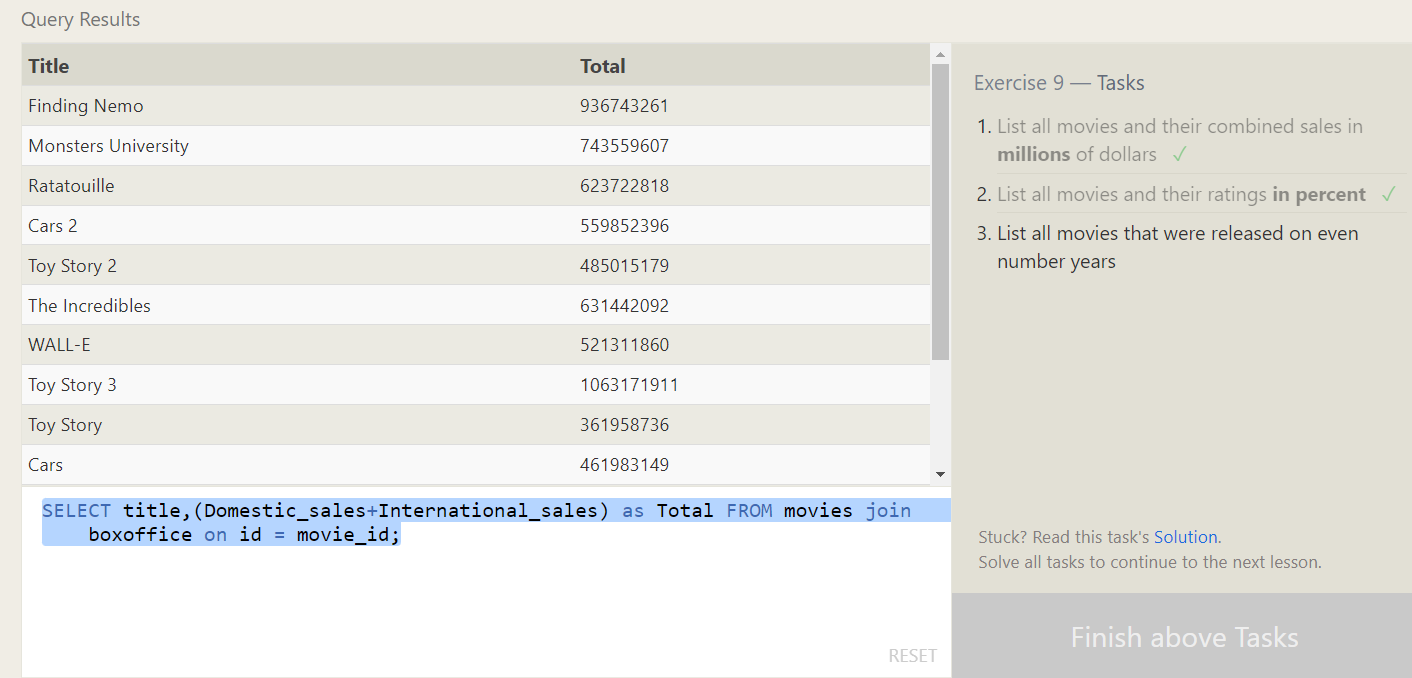
WHERE role IS NULL;



**SQL Lesson 9: Queries with expressions**

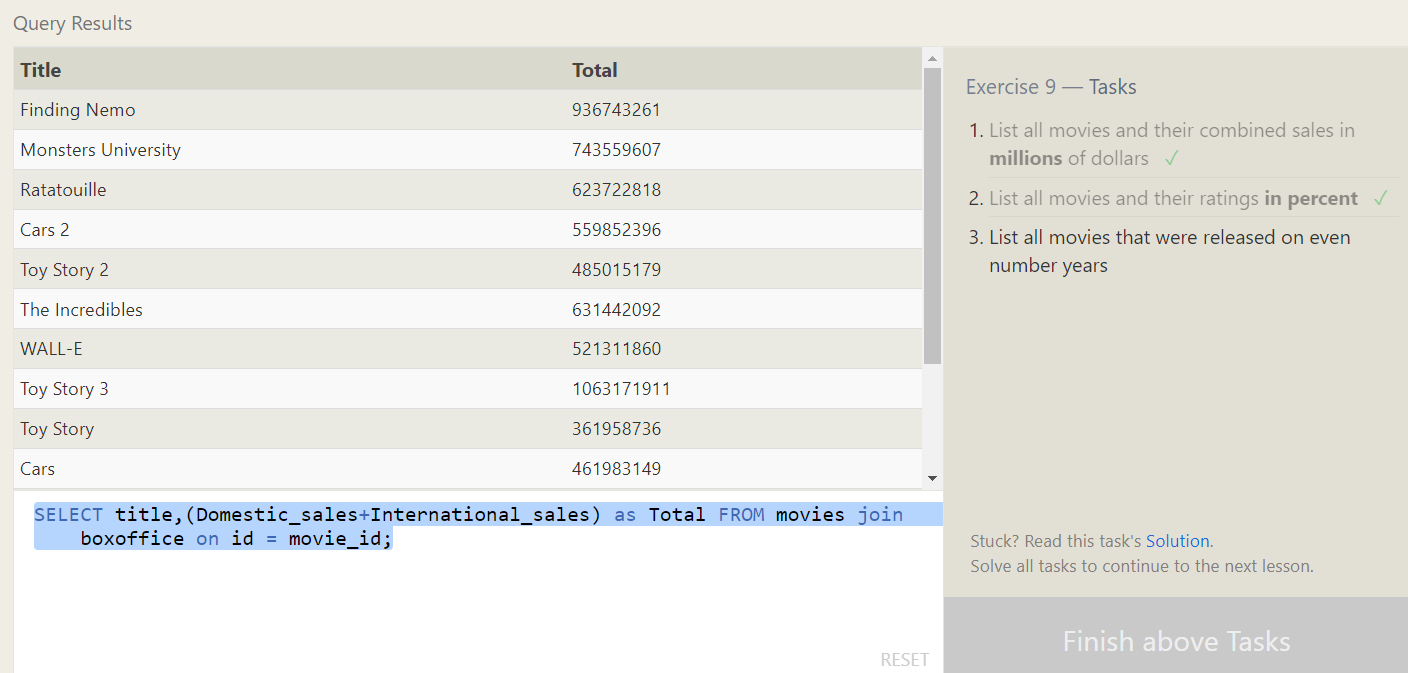
1. List all movies and their combined sales in **millions** of dollars

SELECT title,(Domestic\_sales+International\_sales) as Total FROM movies join boxoffice on id = movie\_id



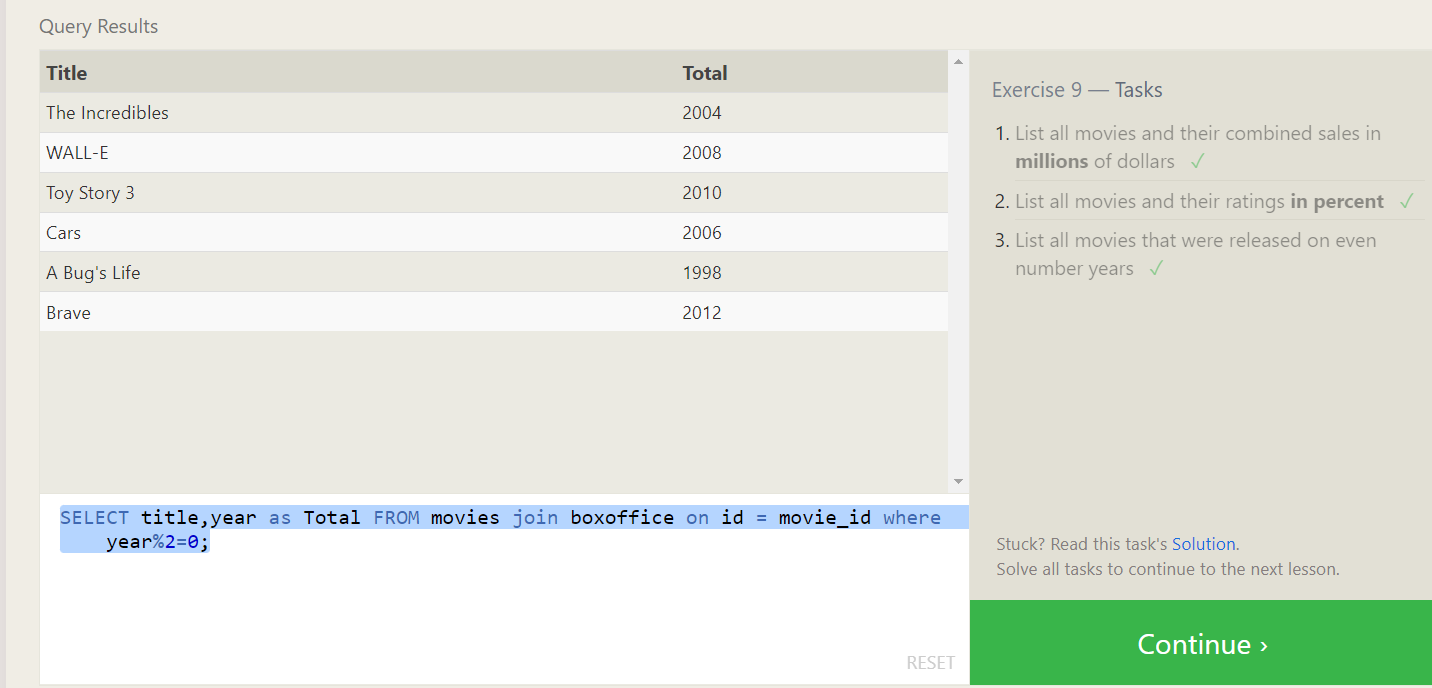
1. List all movies and their ratings **in percent**

SELECT title,(Domestic\_sales+International\_sales) as Total FROM movies join boxoffice on id = movie\_id;



1. List all movies that were released on even number years

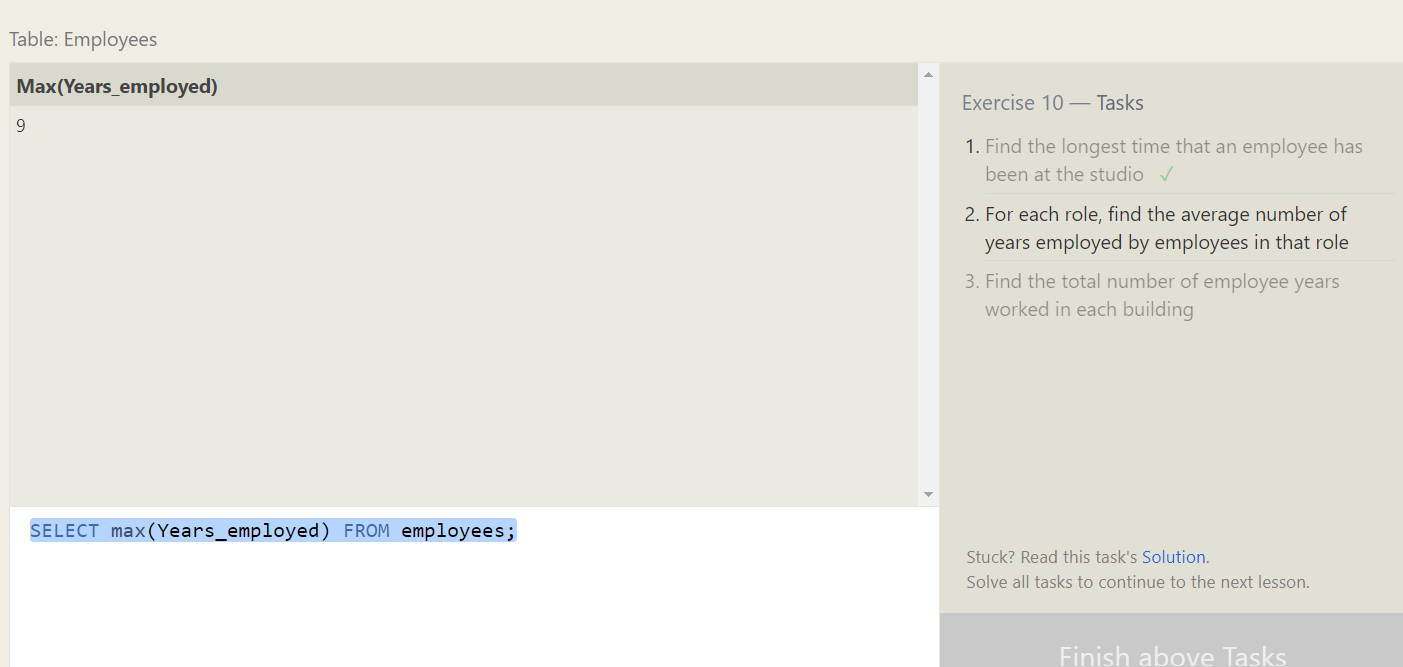
SELECT title,year as Total FROM movies join boxoffice on id = movie\_id where year%2=0;



**SQL Lesson 10: Queries with aggregates (Pt. 1)**

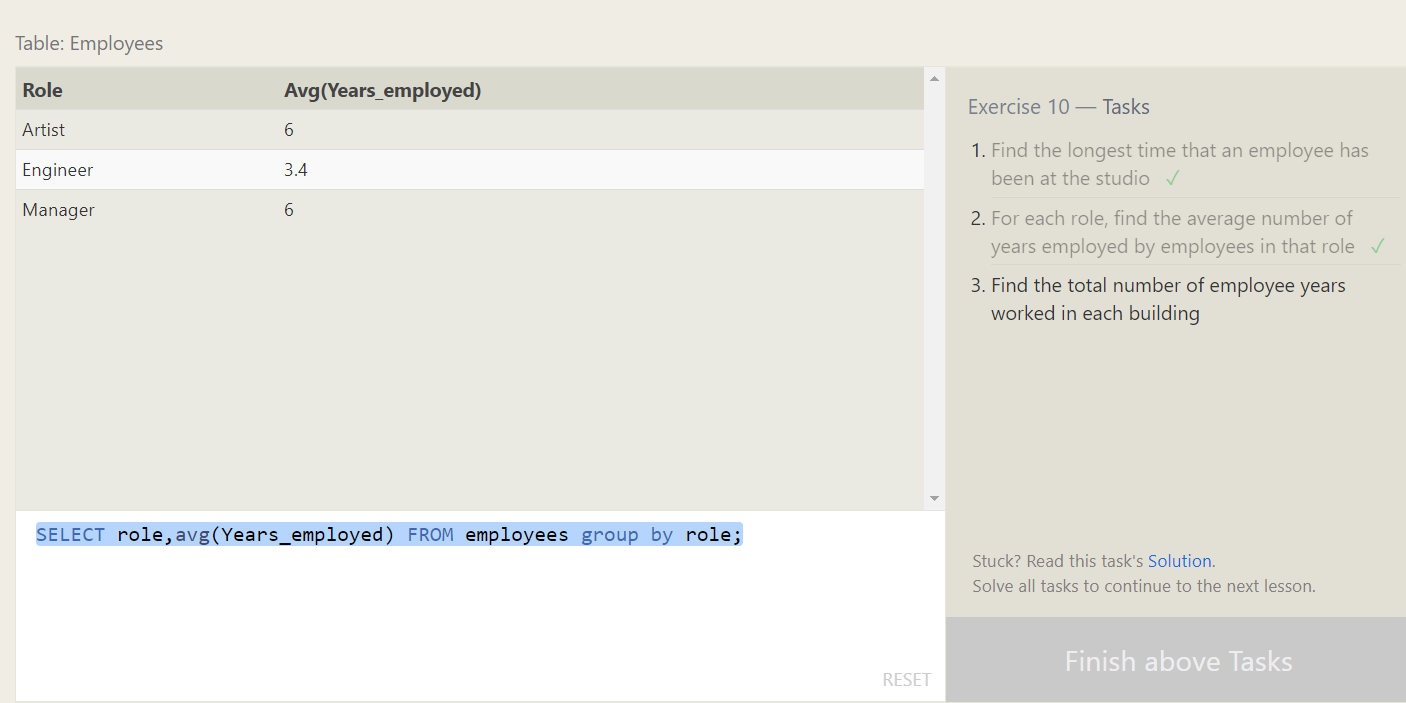
1. Find the longest time that an employee has been at the studio

SELECT max(Years\_employed) FROM employees;



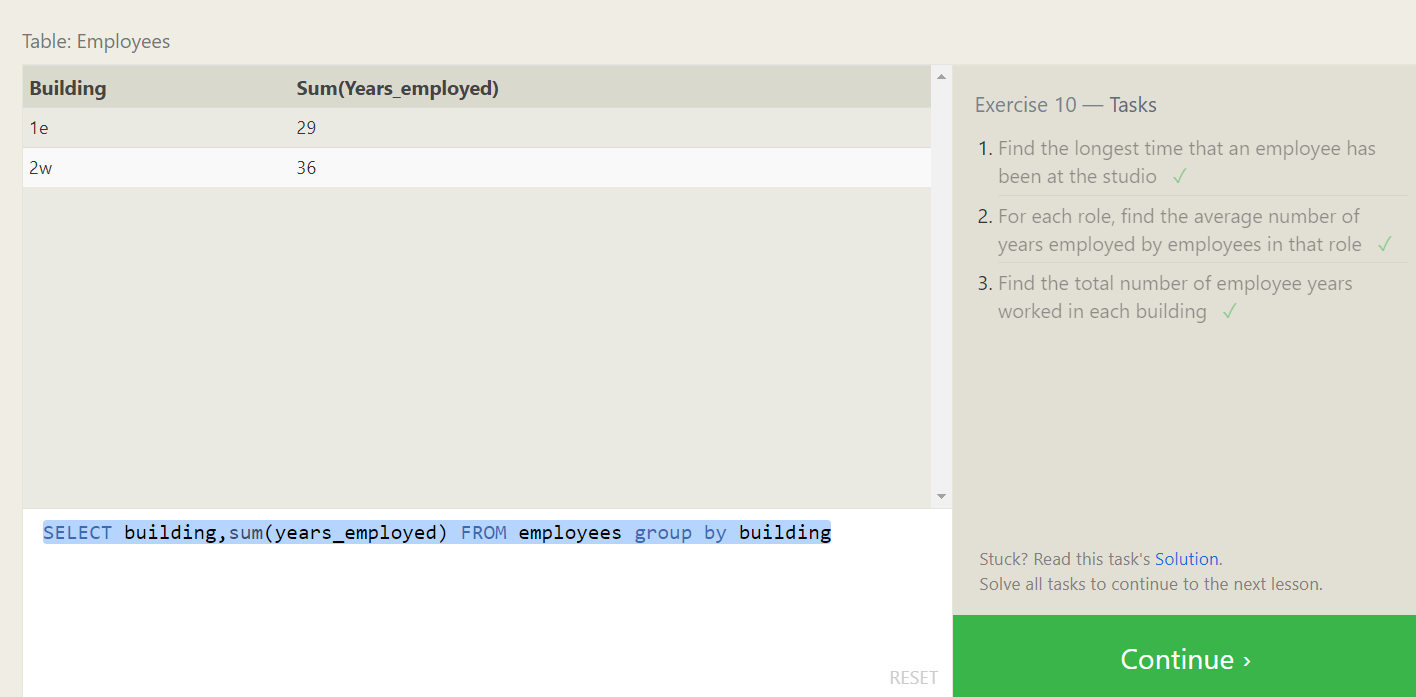
1. For each role, find the average number of years employed by employees in that role

SELECT role,avg(Years\_employed) FROM employees group by role;



1. Find the total number of employee years worked in each building

SELECT building,sum(years\_employed) FROM employees group by building



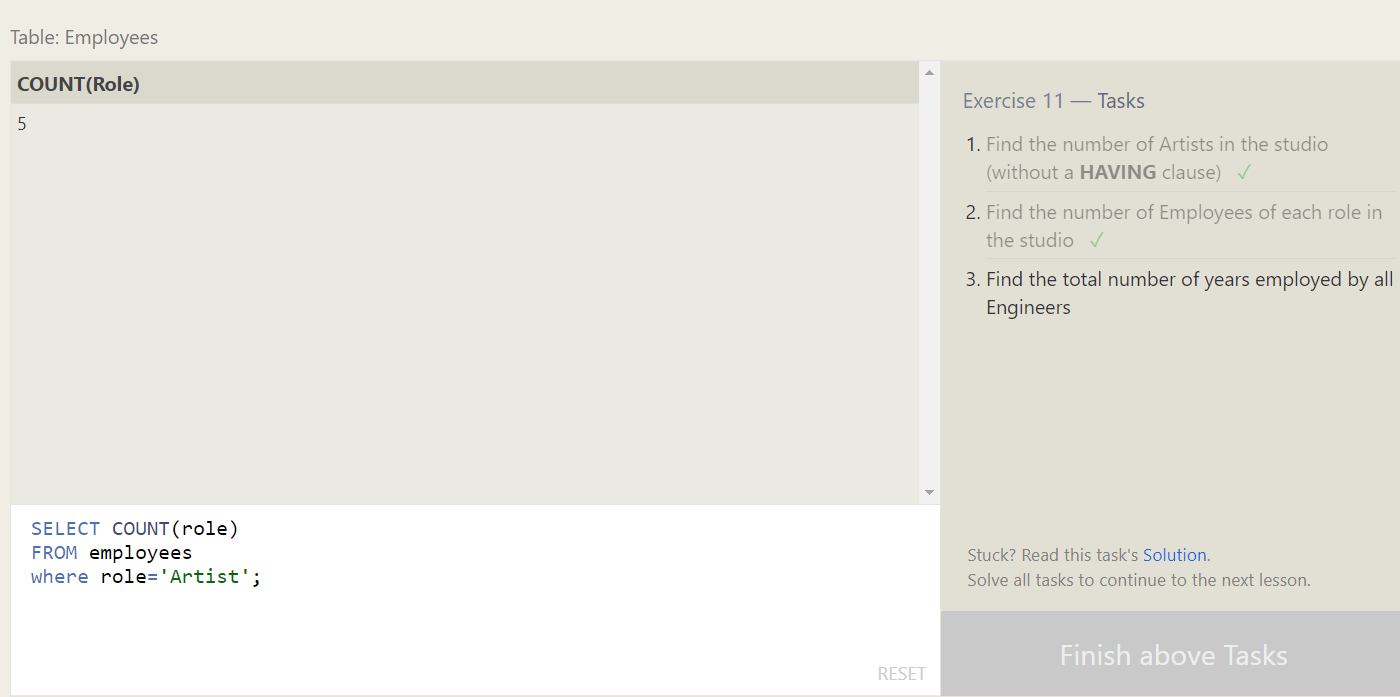
SQL Lesson 11: Queries with aggregates (Pt. 2)

1. 1. Find the number of Artists in the studio (without a **HAVING** clause)

SELECT COUNT(role)

FROM employees

where role='Artist';

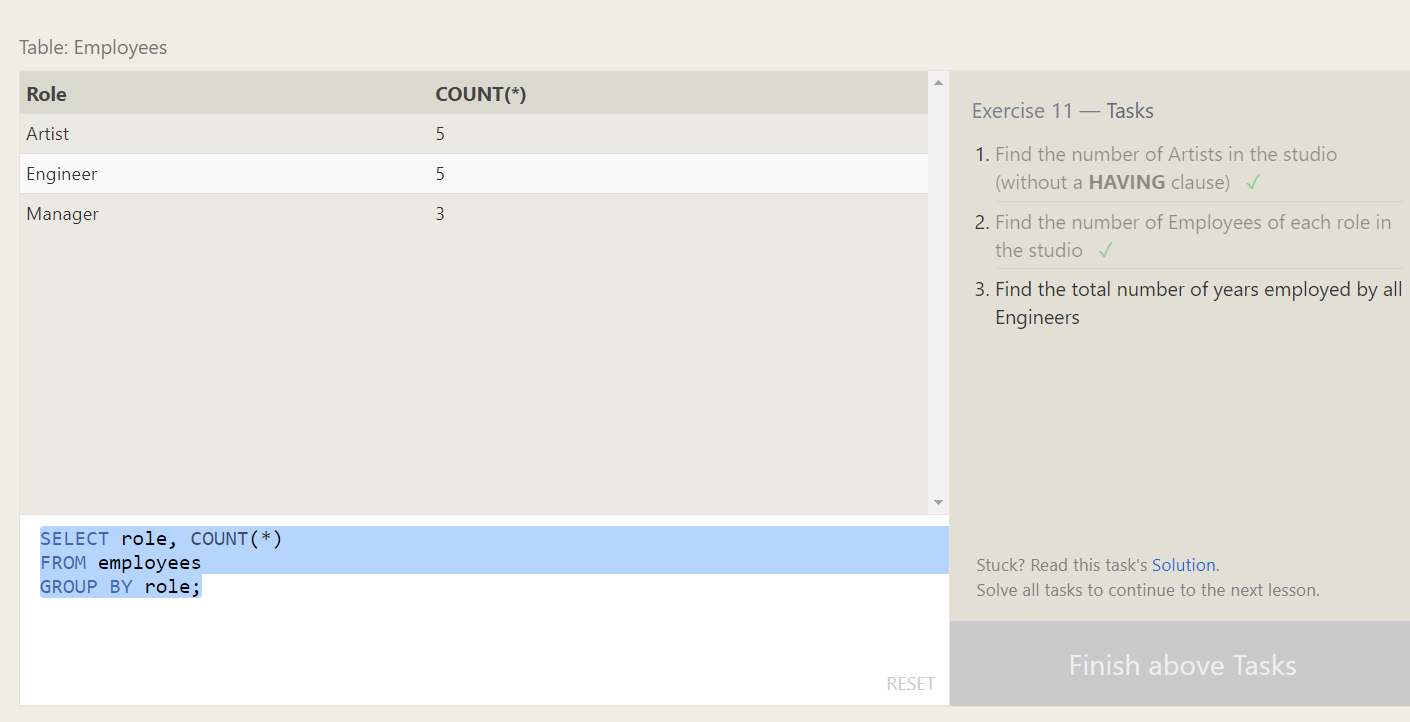


1. Find the number of Employees of each role in the studio

SELECT role, COUNT(\*)

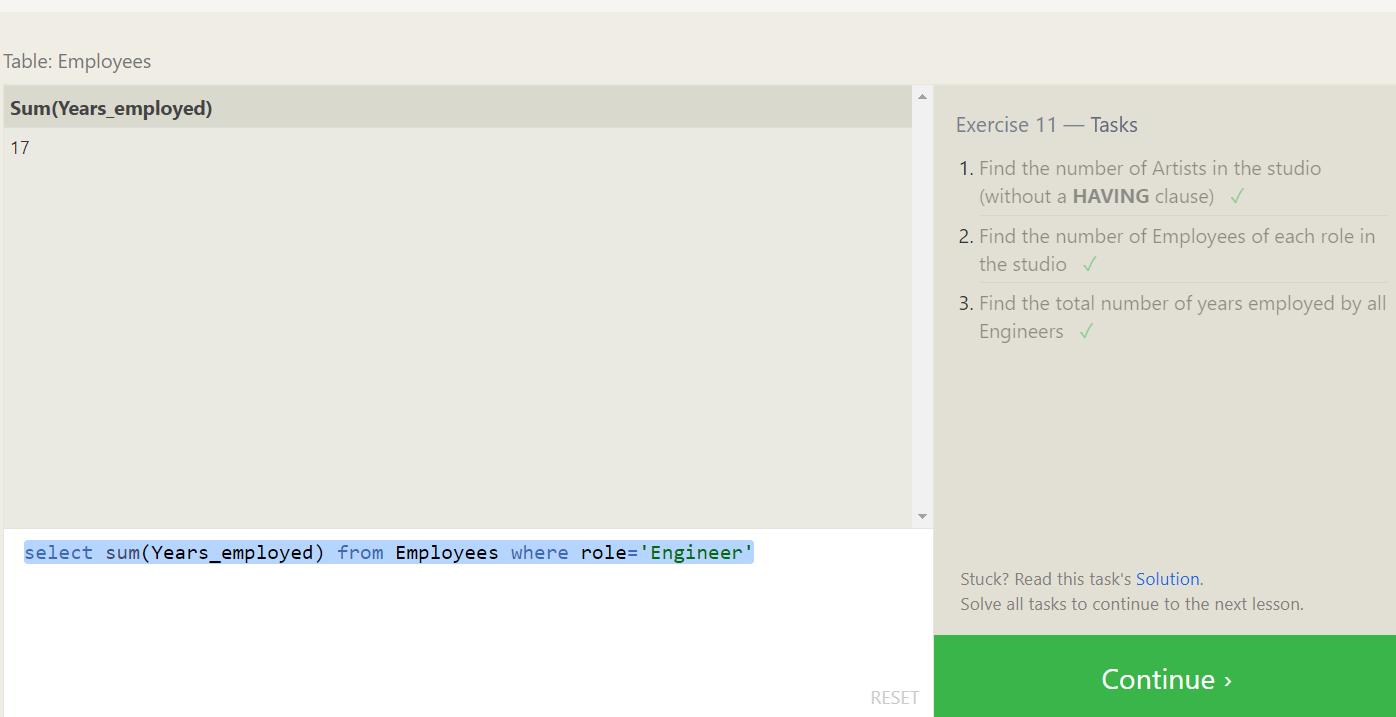
FROM employees

GROUP BY role;



1. Find the total number of years employed by all Engineers

select sum(Years\_employed) from Employees where role='Engineer'



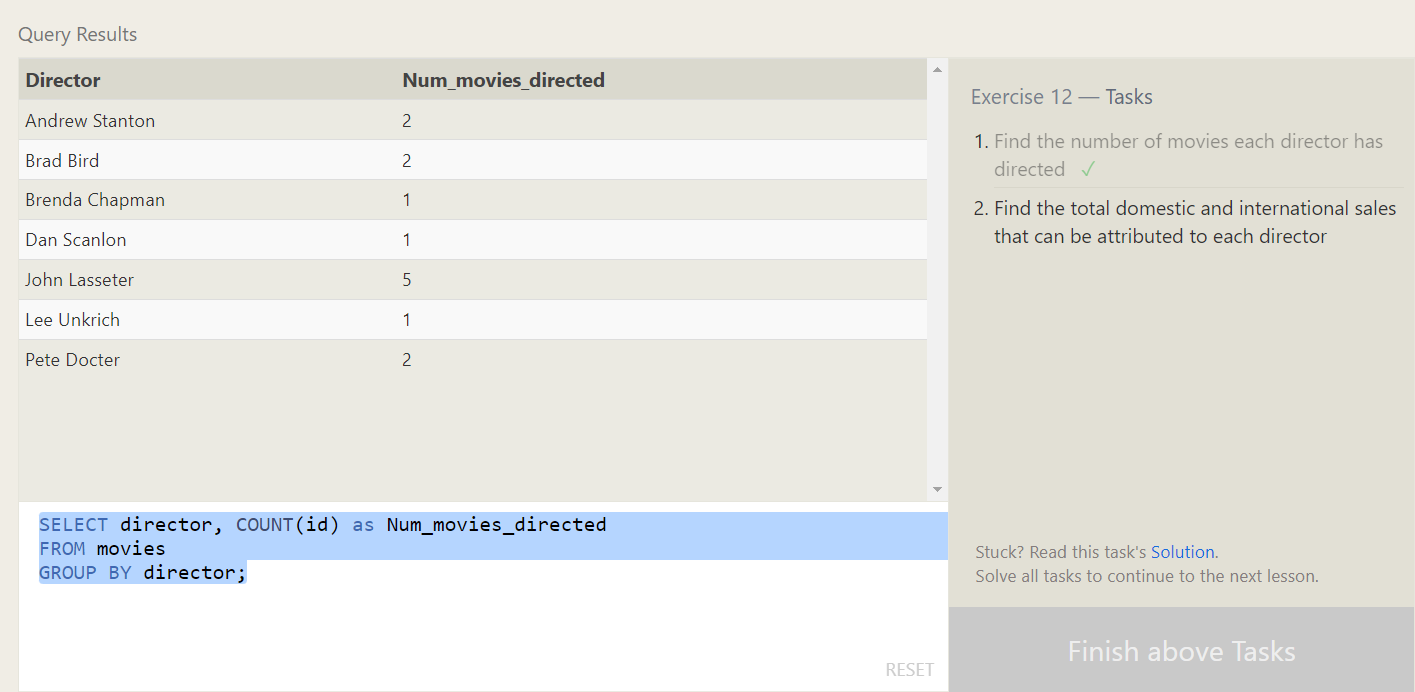
**SQL Lesson 12: Order of execution of a Query**

1. Find the number of movies each director has directed

SELECT director, COUNT(id) as Num\_movies\_directed

FROM movies

GROUP BY director;



1. Find the total domestic and international sales that can be attributed to each director

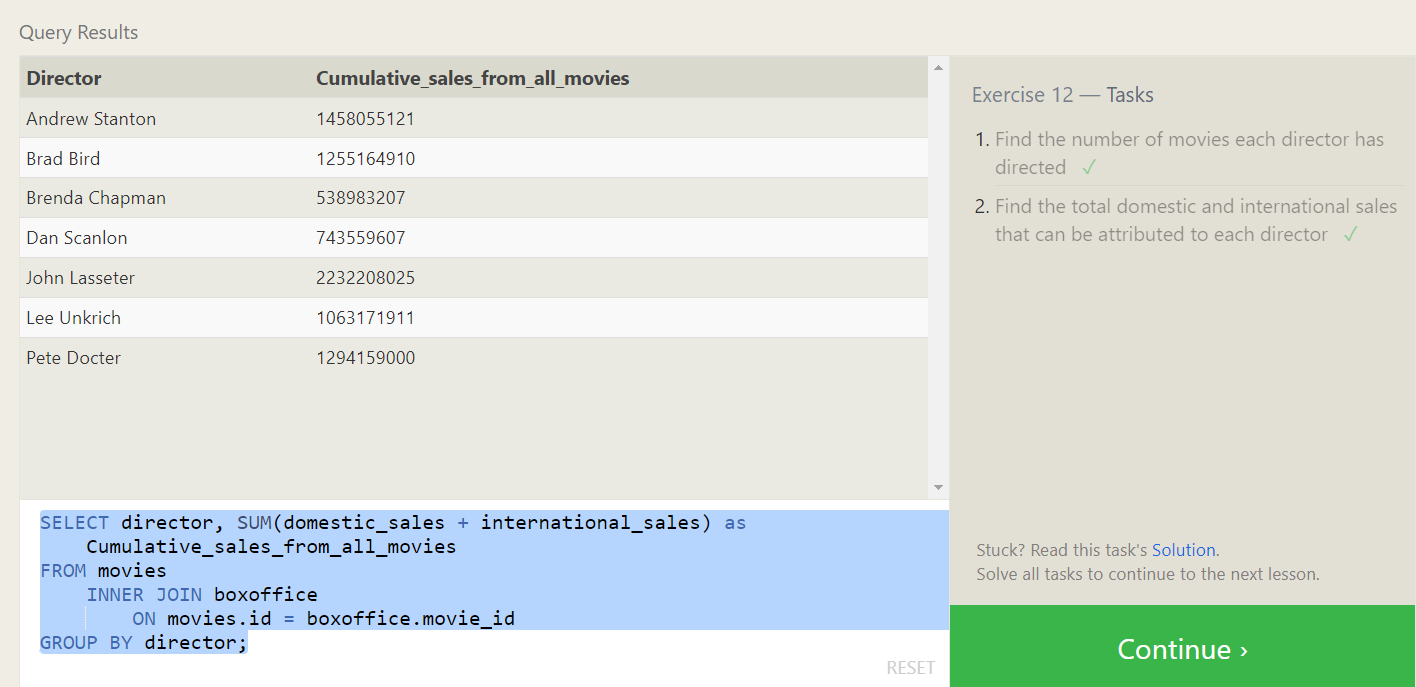
SELECT director, SUM(domestic\_sales + international\_sales) as Cumulative\_sales\_from\_all\_movies

FROM movies

INNER JOIN boxoffice

ON movies.id = boxoffice.movie\_id

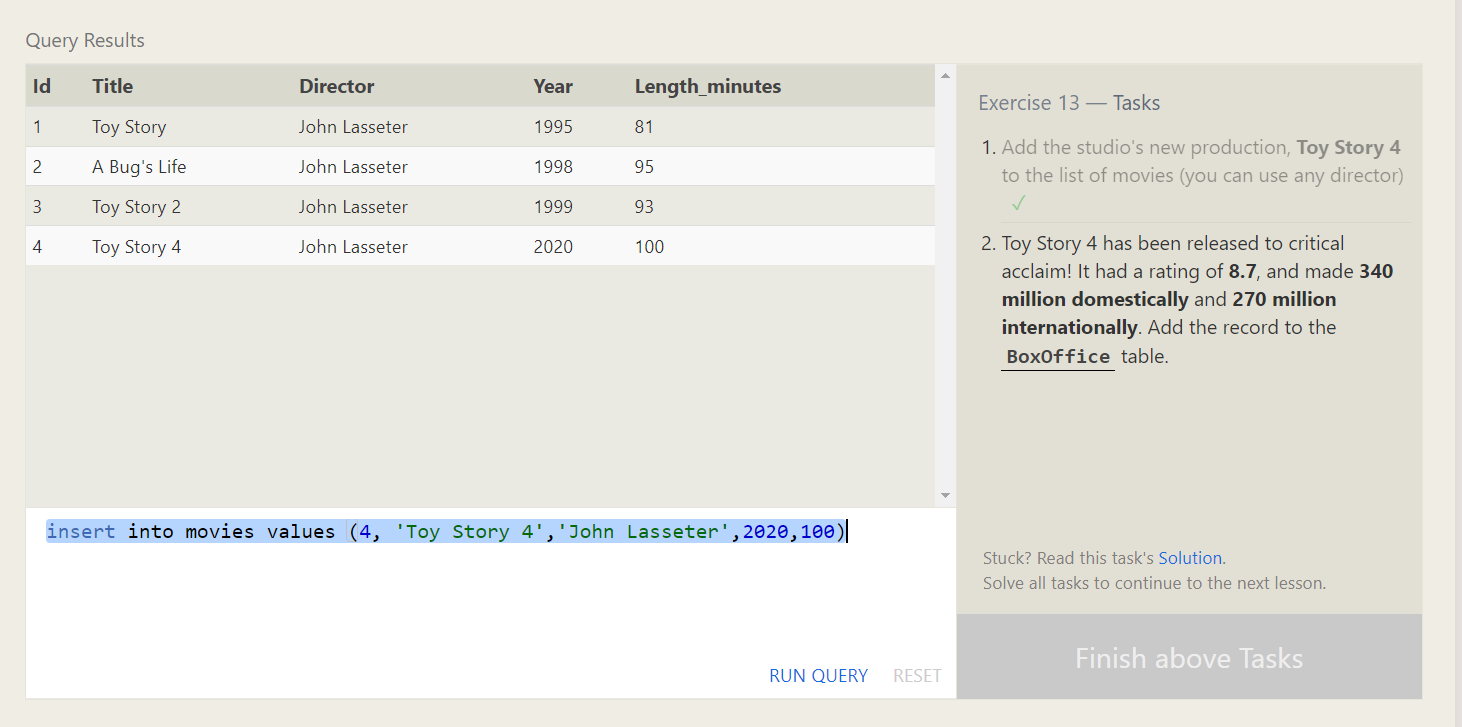
GROUP BY director;



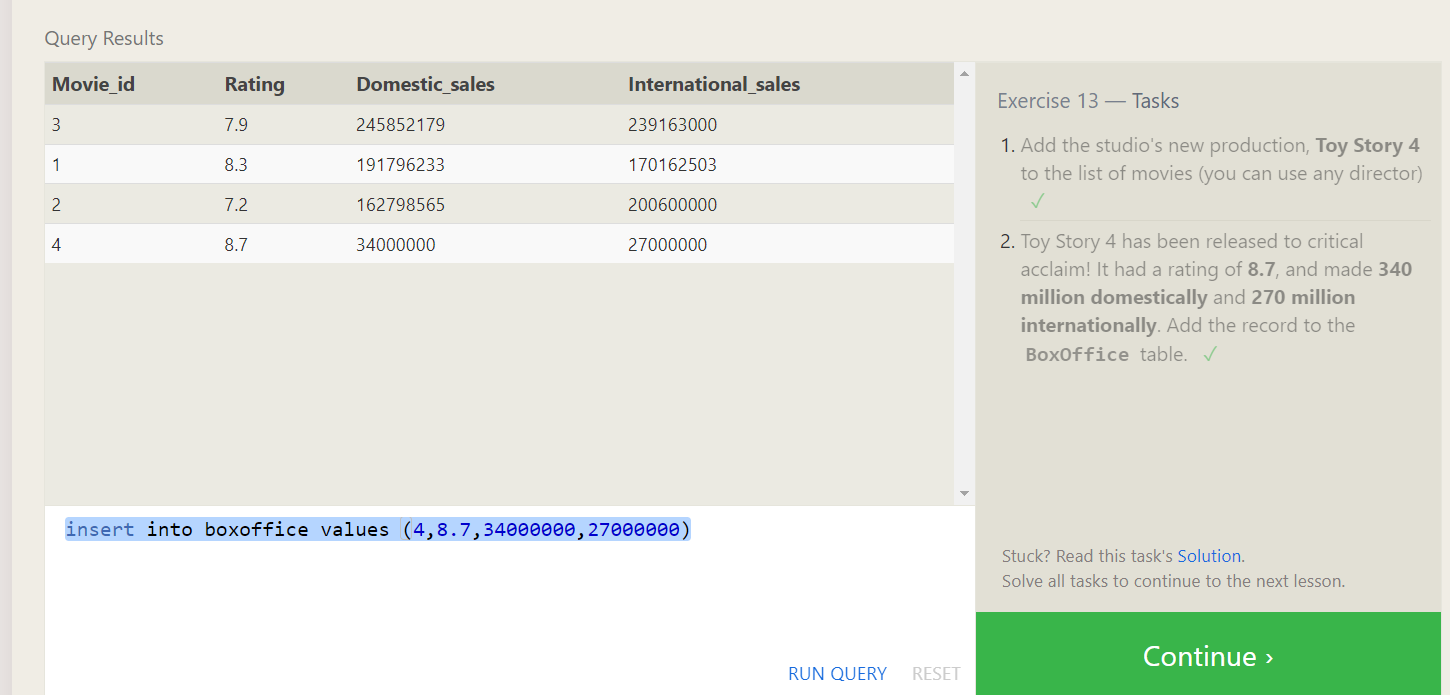
**SQL Lesson 13: Inserting rows**

1. Add the studio's new production, **Toy Story 4** to the list of movies (you can use any director)

insert into movies values (4, 'Toy Story 4','John Lasseter',2020,100)



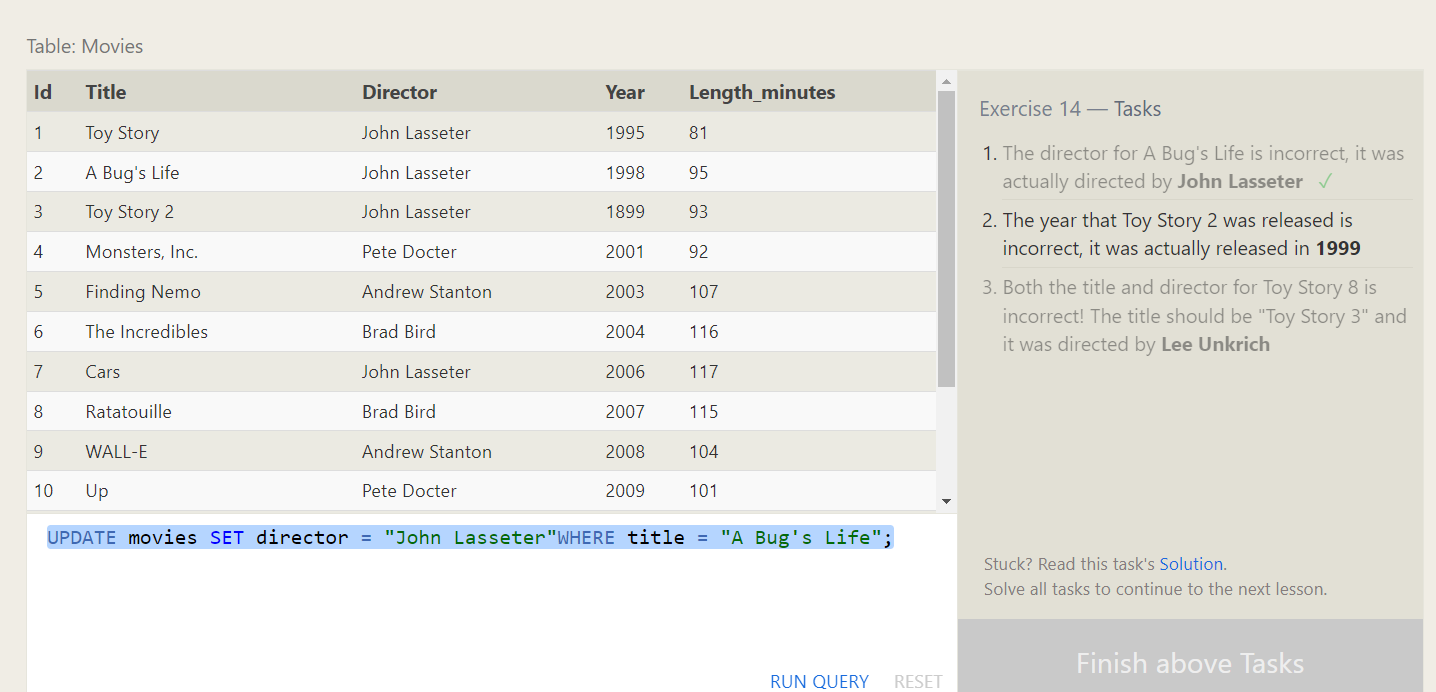
1. Toy Story 4 has been released to critical acclaim! It had a rating of **8.7**, and made **340 million domestically** and **270 million internationally**. Add the record to the **BoxOffice** table.

insert into boxoffice values (4,8.7,34000000,27000000) 

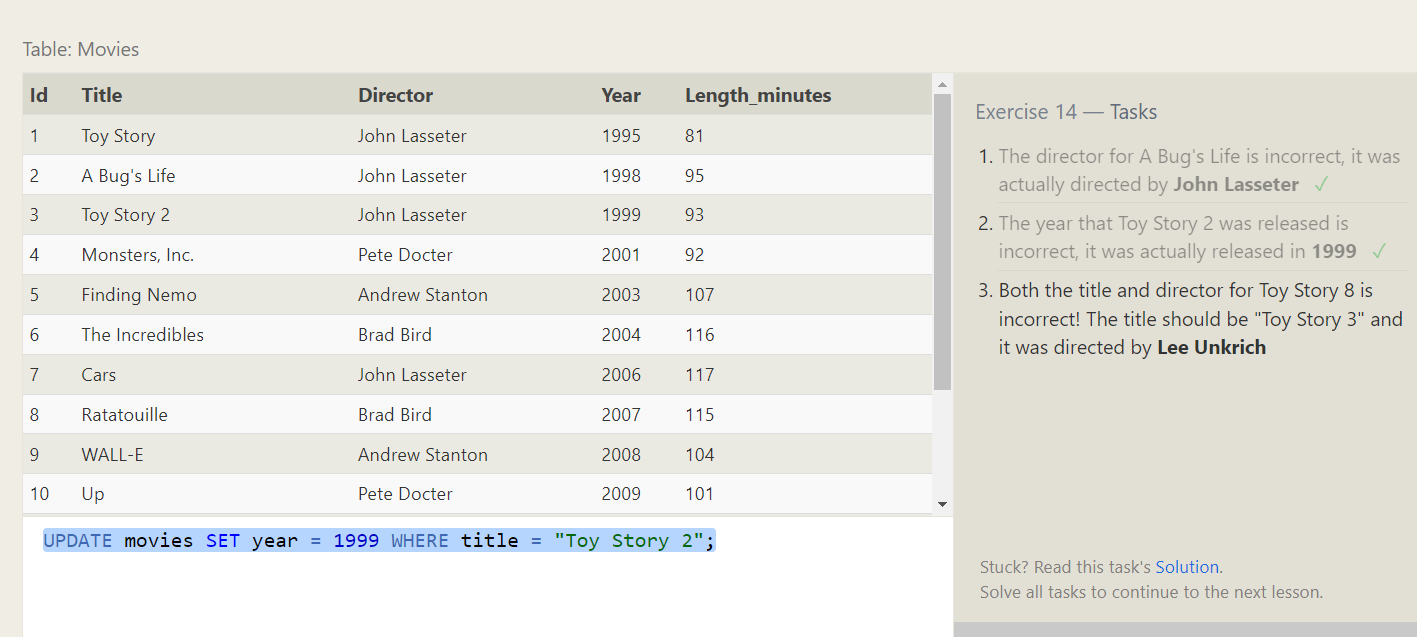
**SQL Lesson 14: Updating rows**

Q: The director for A Bug's Life is incorrect, it was actually directed by John Lasseter

A: UPDATE movies SET director = "John Lasseter"WHERE title = "A Bug's Life";

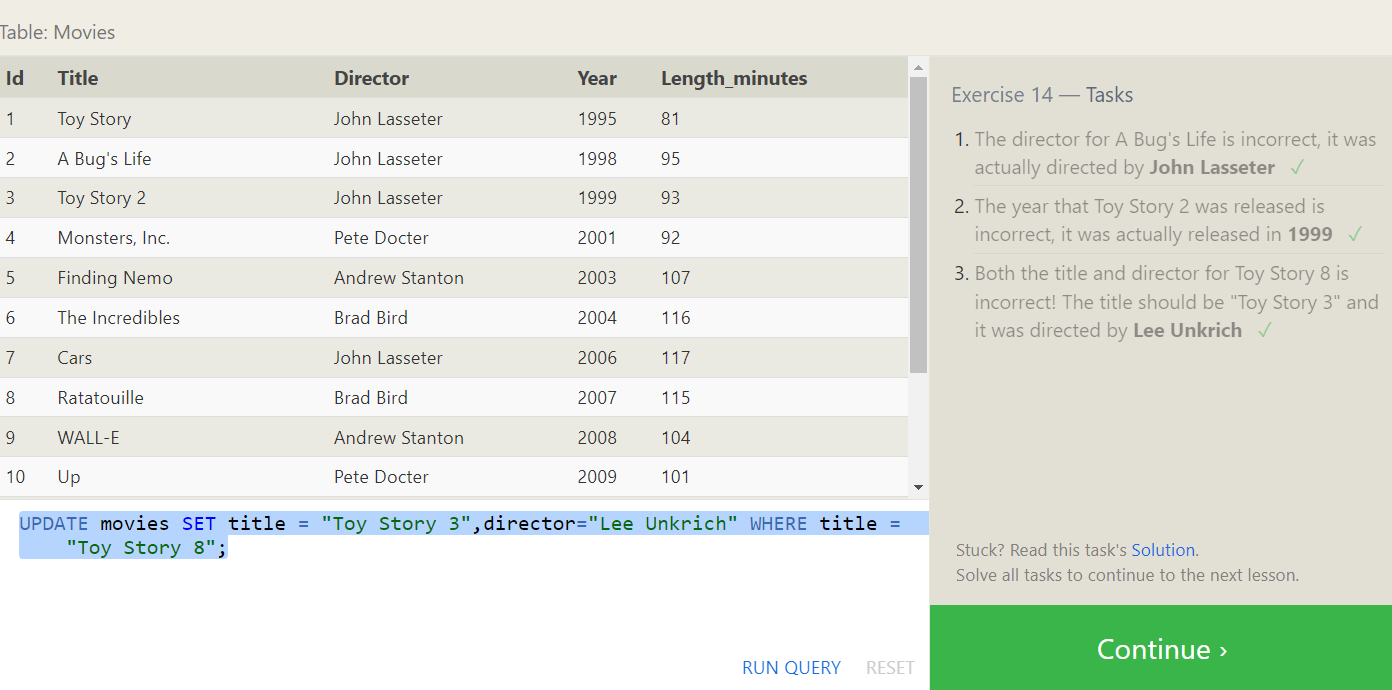


Q: The year that Toy Story 2 was released is incorrect, it was actually released in **1999**

A: UPDATE movies SET year = 1999 WHERE title = "Toy Story 2";

Q: Both the title and director for Toy Story 8 is incorrect! The title should be "Toy Story 3" and it was directed by **Lee Unkrich**

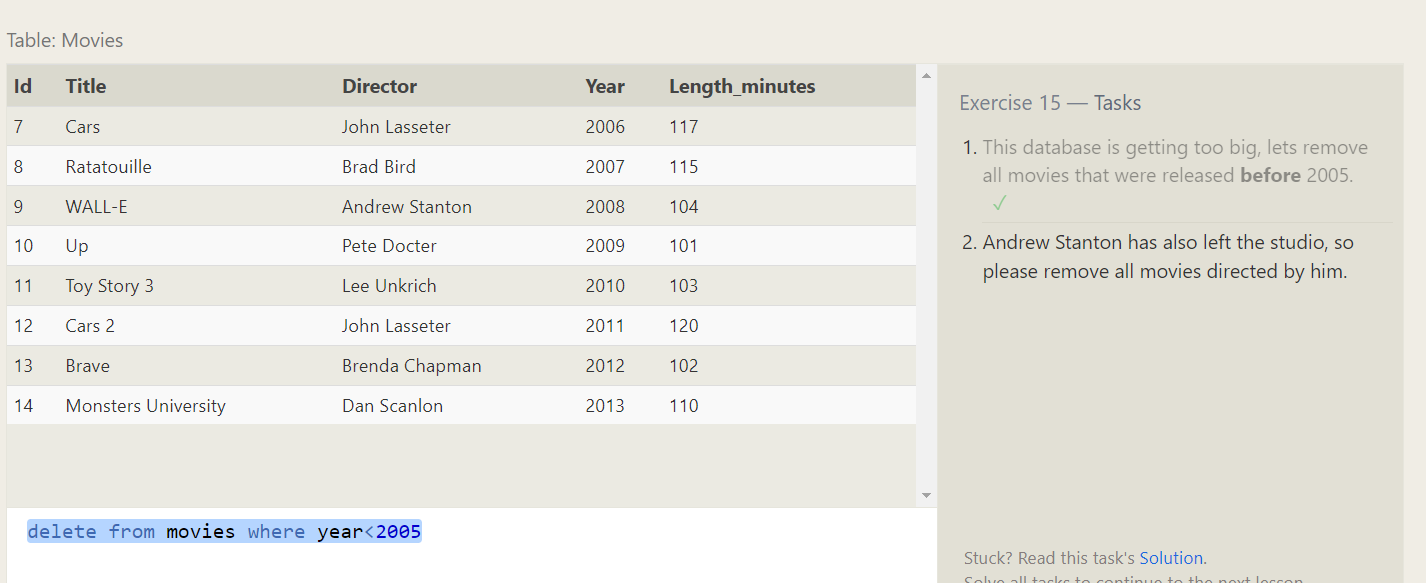
A: UPDATE movies SET title = "Toy Story 3",director="Lee Unkrich" WHERE title = "Toy Story 8";



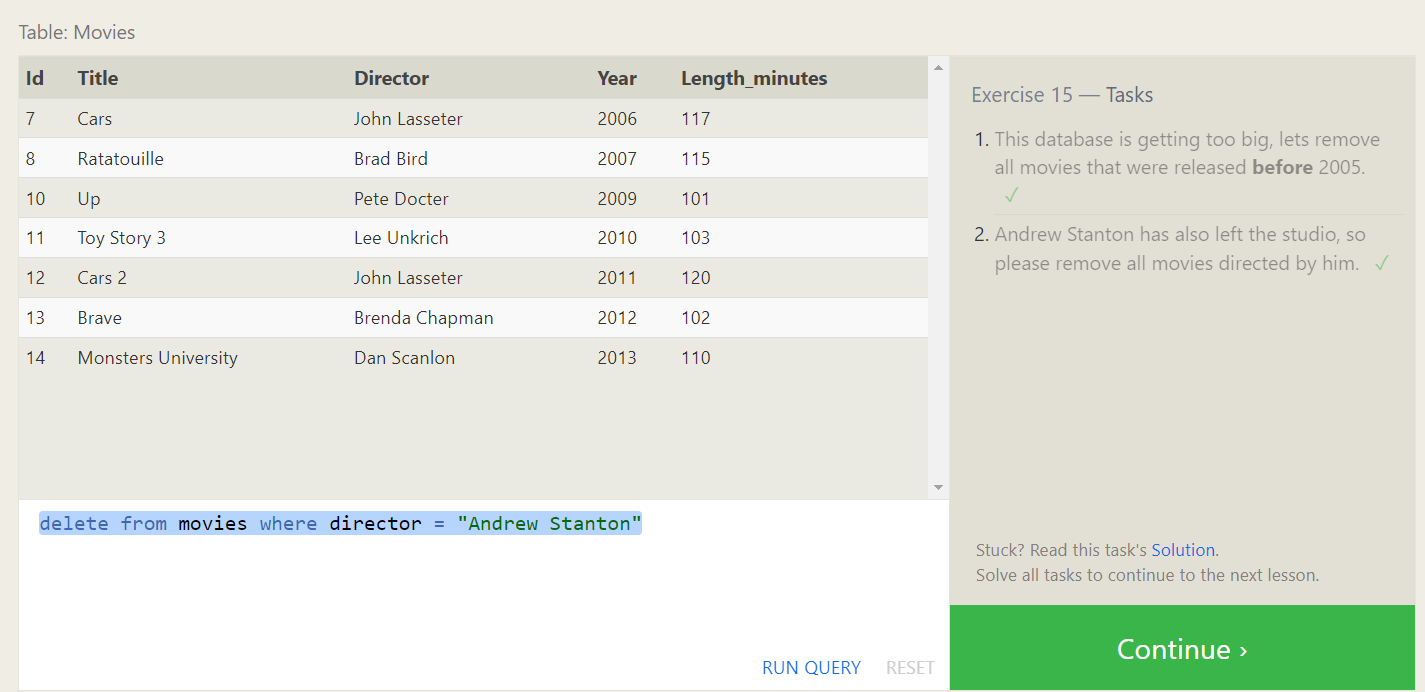
**SQL Lesson 15: Deleting rows**

Q: This database is getting too big, lets remove all movies that were released **before** 2005.

A: delete from movies where year<2005



Q: Andrew Stanton has also left the studio, so please remove all movies directed by him.

A: delete from movies where director = "Andrew Stanton"

**SQL Lesson 16: Creating tables**

Q: Create a new table named Database with the following columns:

– Name A string (text) describing the name of the database

– Version A number (floating point) of the latest version of this database

– Download\_count An integer count of the number of times this database was downloaded

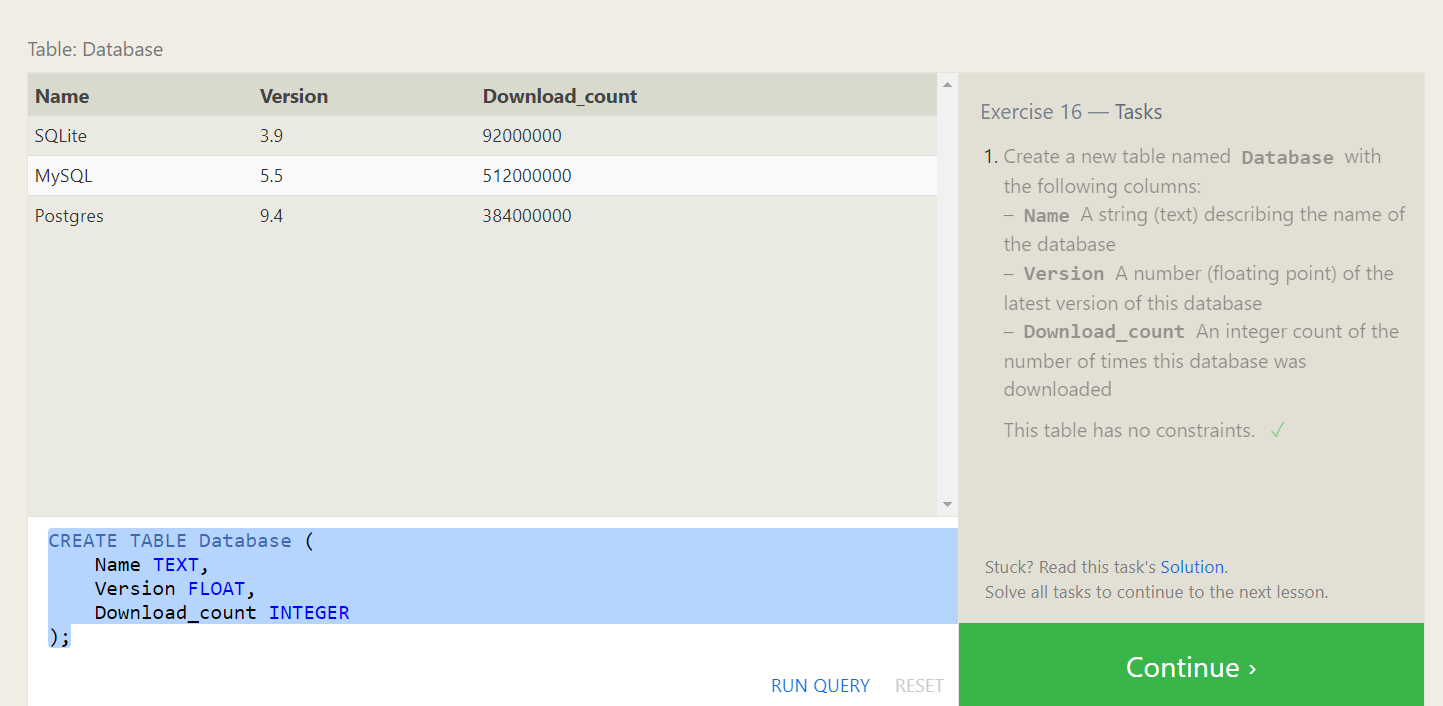
A: CREATE TABLE Database (

Name TEXT,

Version FLOAT,

Download\_count INTEGER

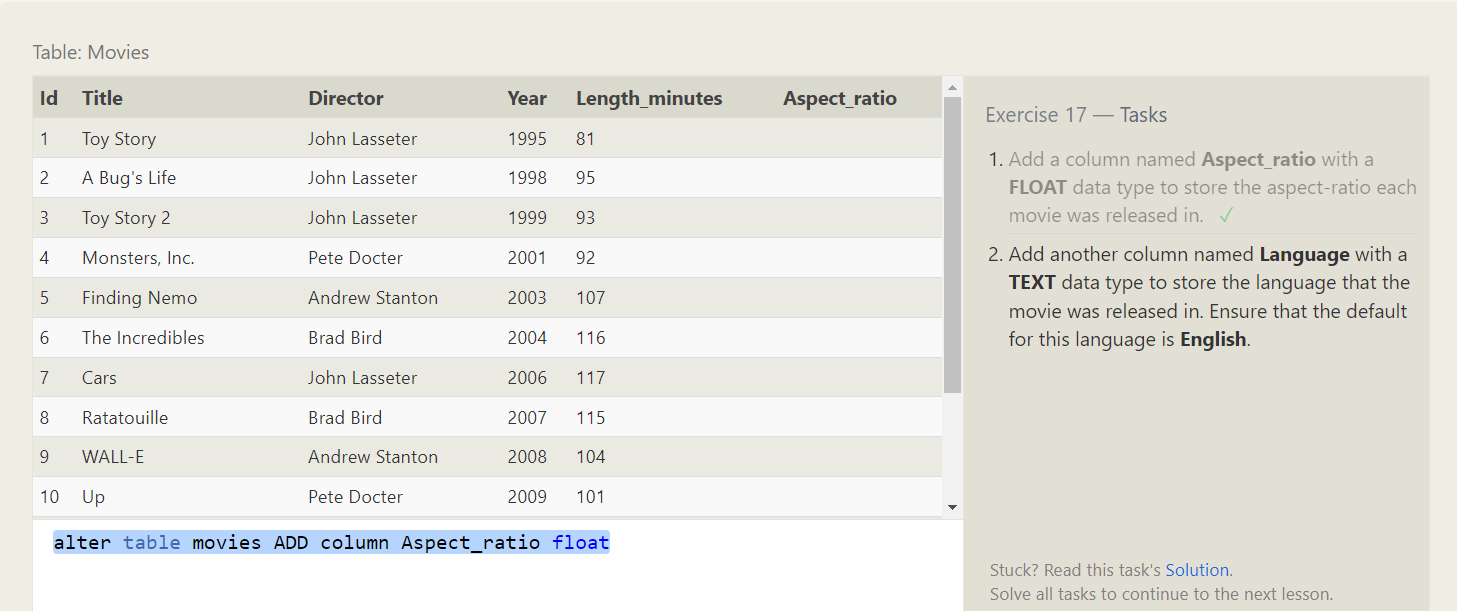
);



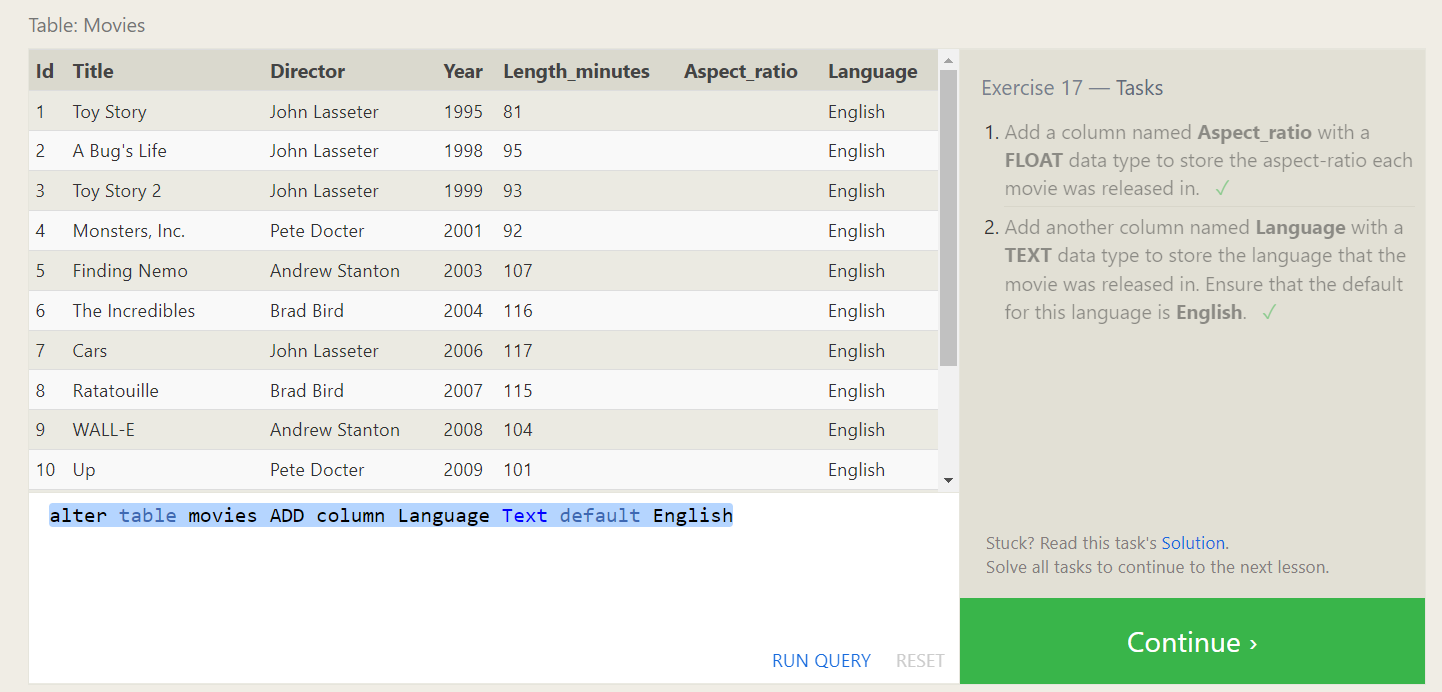
**SQL Lesson 17: Altering tables**

1. Q: Add a column named **Aspect\_ratio** with a **FLOAT** data type to store the aspect-ratio each movie was released in.

A: alter table movies ADD column Aspect\_ratio float



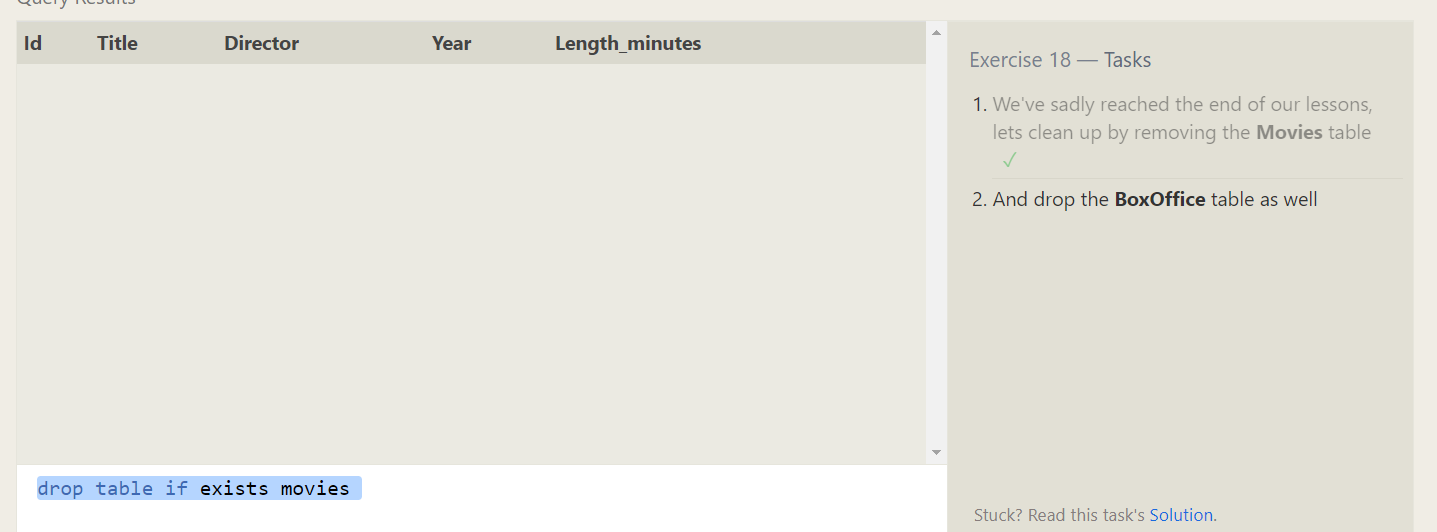
Q: Add another column named **Language** with a **TEXT** data type to store the language that the movie was released in. Ensure that the default for this language is **English**

A: alter table movies ADD column Language Text default English

**SQL Lesson 18: Dropping tables**

1. Q: We've sadly reached the end of our lessons, lets clean up by removing the **Movies** table

A: drop table if exists movies



1. Q: And drop the **BoxOffice** table as well

A: drop table if exists BoxOffice

