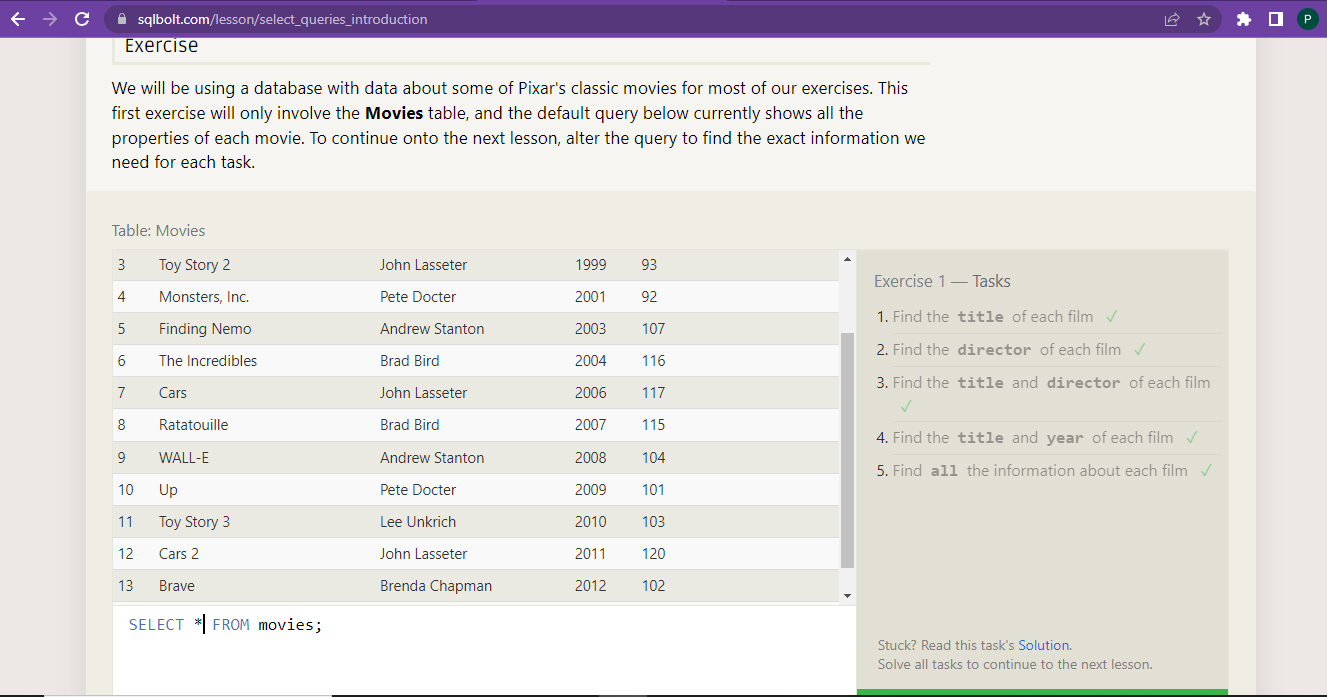
**SQL Lesson 1: SELECT queries 101**

****

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 title, director FROM movies;

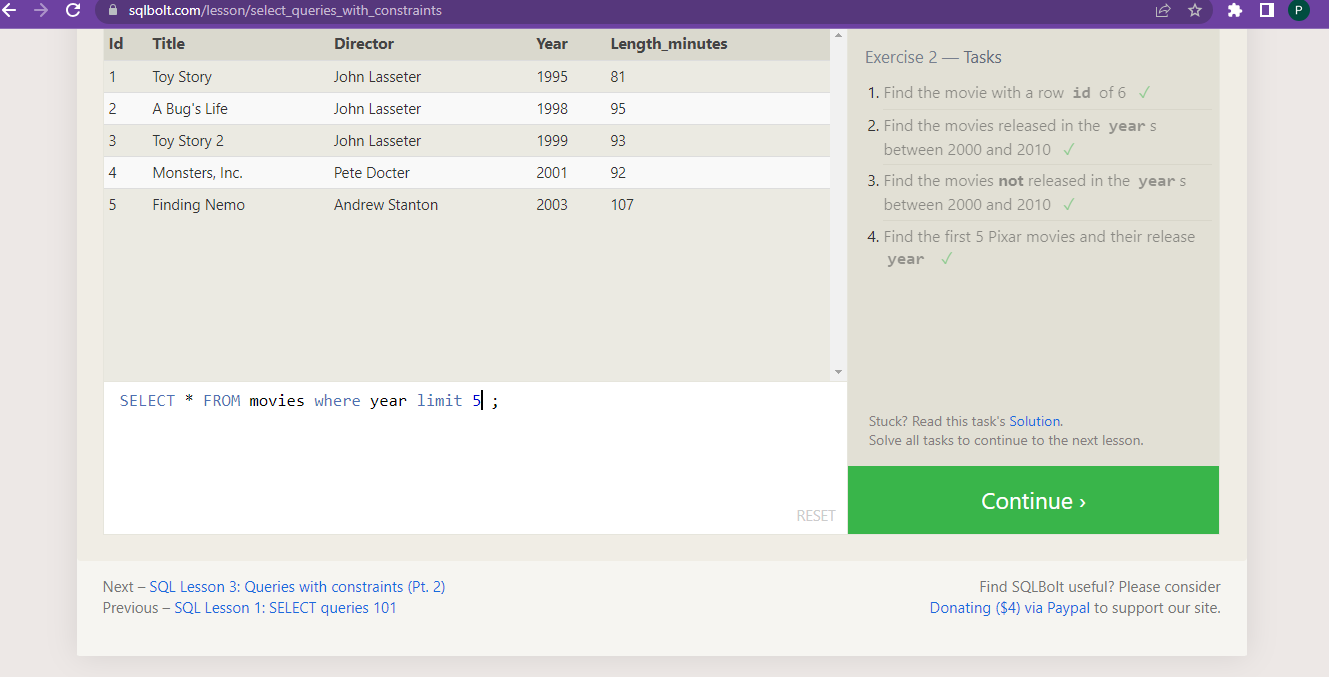
1. Find the title and year of each film ✓

SELECT title, year FROM movies;

1. Find all the information about each film ✓

SELECT \* FROM movies;

**SQL Lesson 2: Queries with constraints (Pt. 1)**

****

1. Find the movie with a row id of 6 ✓

SELECT \* FROM movies WHERE id = 6;

1. Find the movies released in the years between 2000 and 2010 ✓

SELECT \* FROM movies WHERE year BETWEEN 2000 AND 2010;

1. Find the movies not released in the years 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 WHERE year limit 5;

**SQL Lesson 3: Queries with constraints (Pt. 2)**

1. Find all the Toy Story movies ✓

SELECT \* FROM movies where title like '%toy story%';

2 Find all the movies directed by John Lasseter ✓

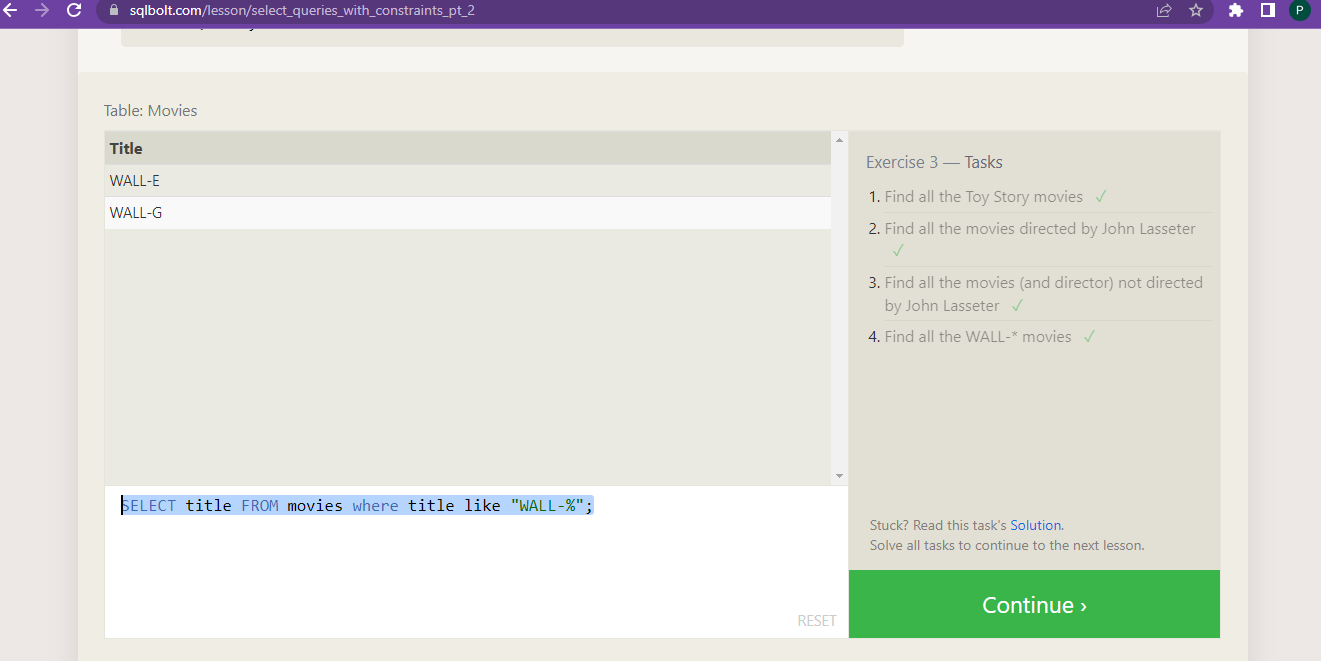
SELECT \* FROM movies where director like 'john lasseter';

3 Find all the movies (and director) not directed by John Lasseter ✓

SELECT \* FROM movies where director not like 'john lasseter';

Find all the WALL-\* movies ✓

SELECT title FROM movies where title like "WALL-%";



**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) ✓

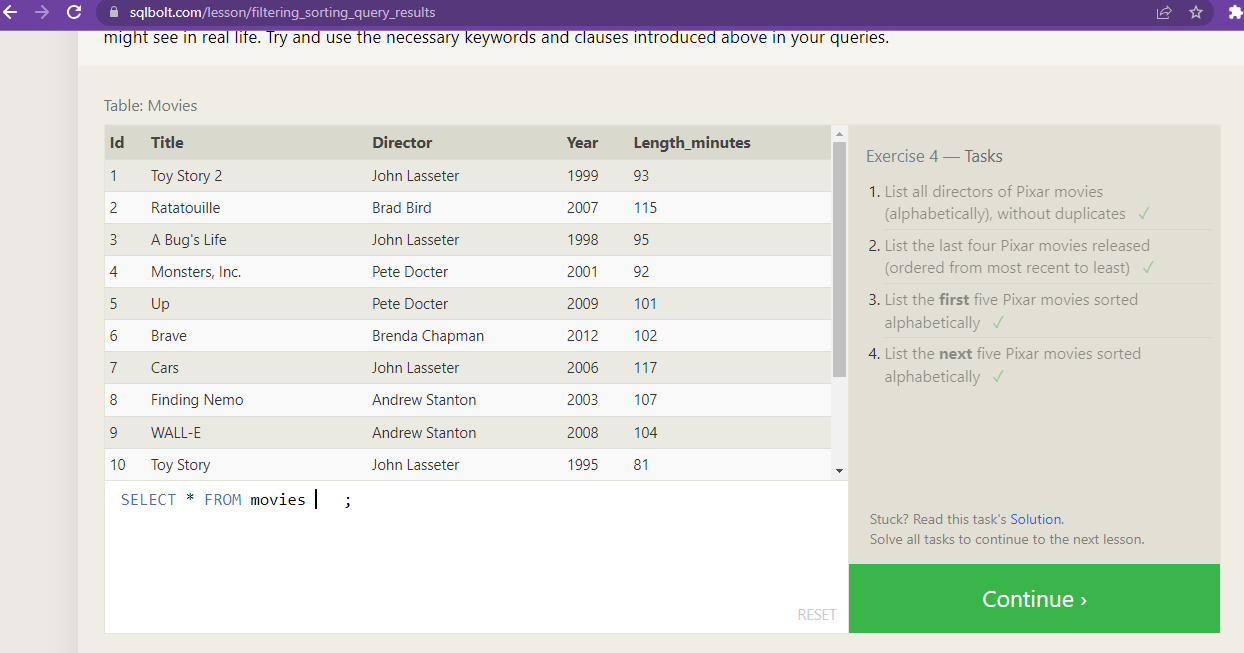
SELECT distinct title FROM movies order by year desc limit 4 ;

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

SELECT title FROM movies order by title limit 5 ;

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

SELECT title 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';

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

SELECT city FROM north\_american\_cities where country like 'united states'order by latitude desc ;

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

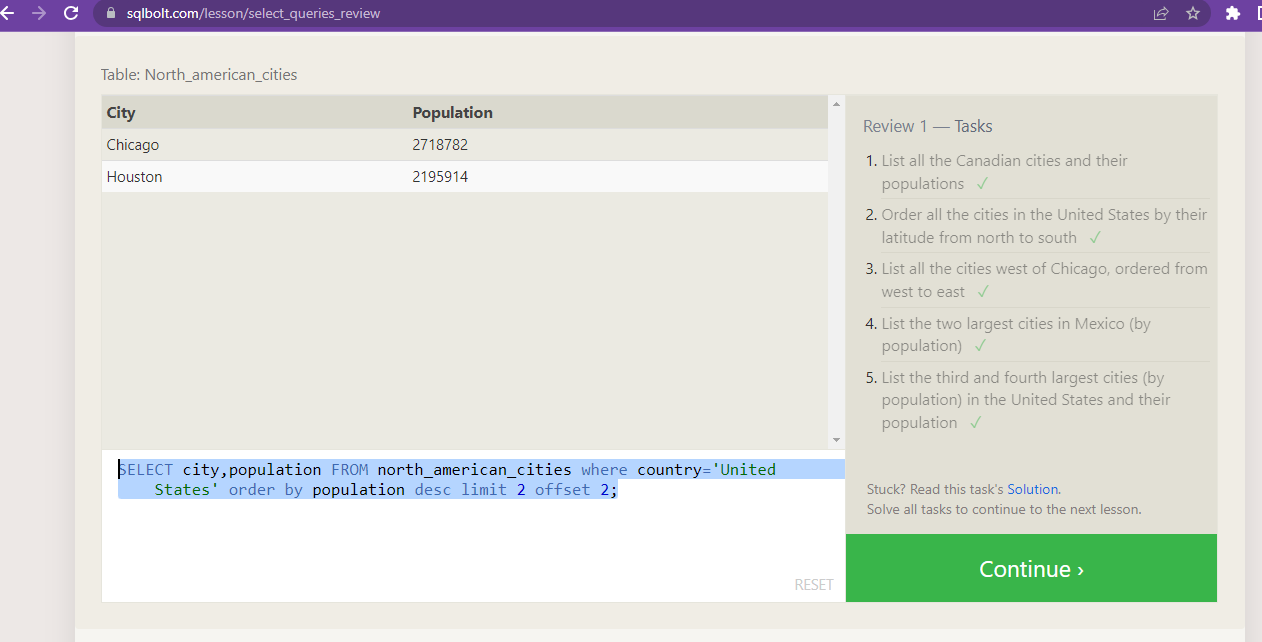
SELECT city FROM north\_american\_cities where longitude <-87.929798 order by longitude ;

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

SELECT city 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 city,population 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 ✓

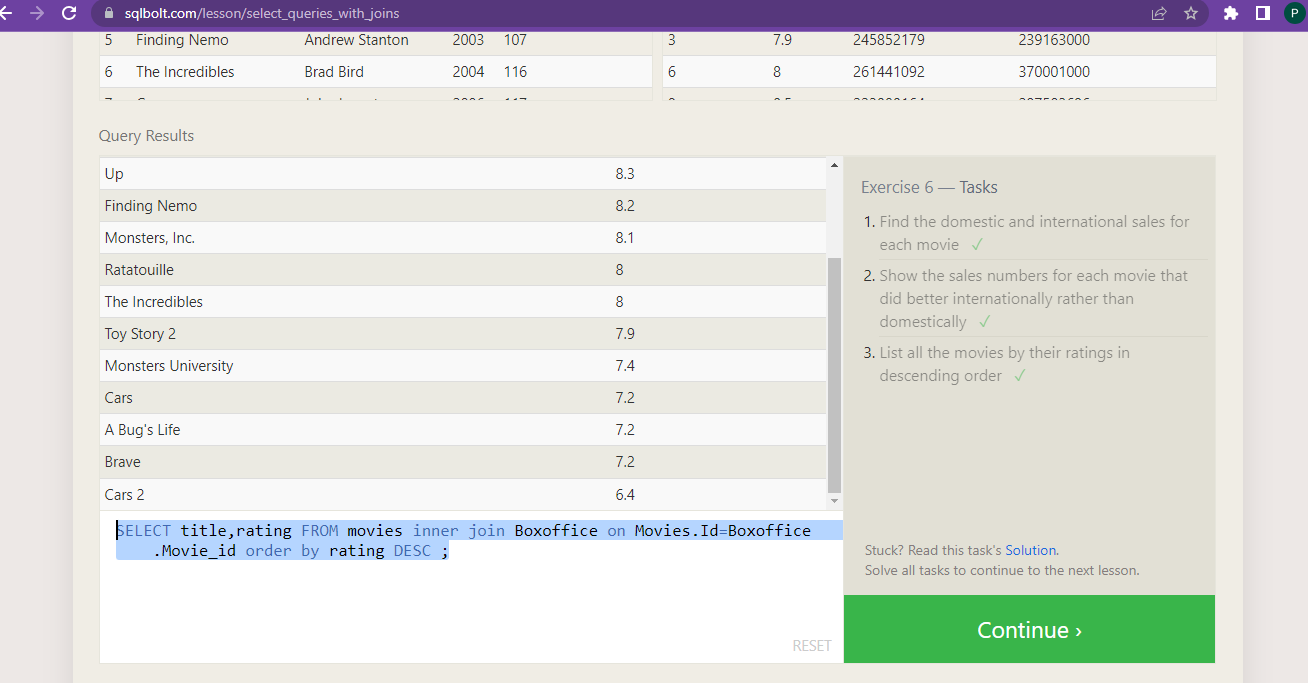
SELECT title, Domestic\_sales, International\_sales FROM movies inner join boxoffice on movies.id=Boxoffice.Movie\_id;

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

SELECT title, Domestic\_sales, International\_sales FROM movies inner join boxoffice on movies.id = boxoffice.movie\_id where International\_sales > Domestic\_sales;

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

SELECT title,rating FROM movies inner join Boxoffice on Movies.Id=Boxoffice.Movie\_id order by rating DESC ;



**SQL Lesson 7: OUTER JOINs**

1. Find the list of all buildings that have employees ✓

SELECT distinct building FROM employees;

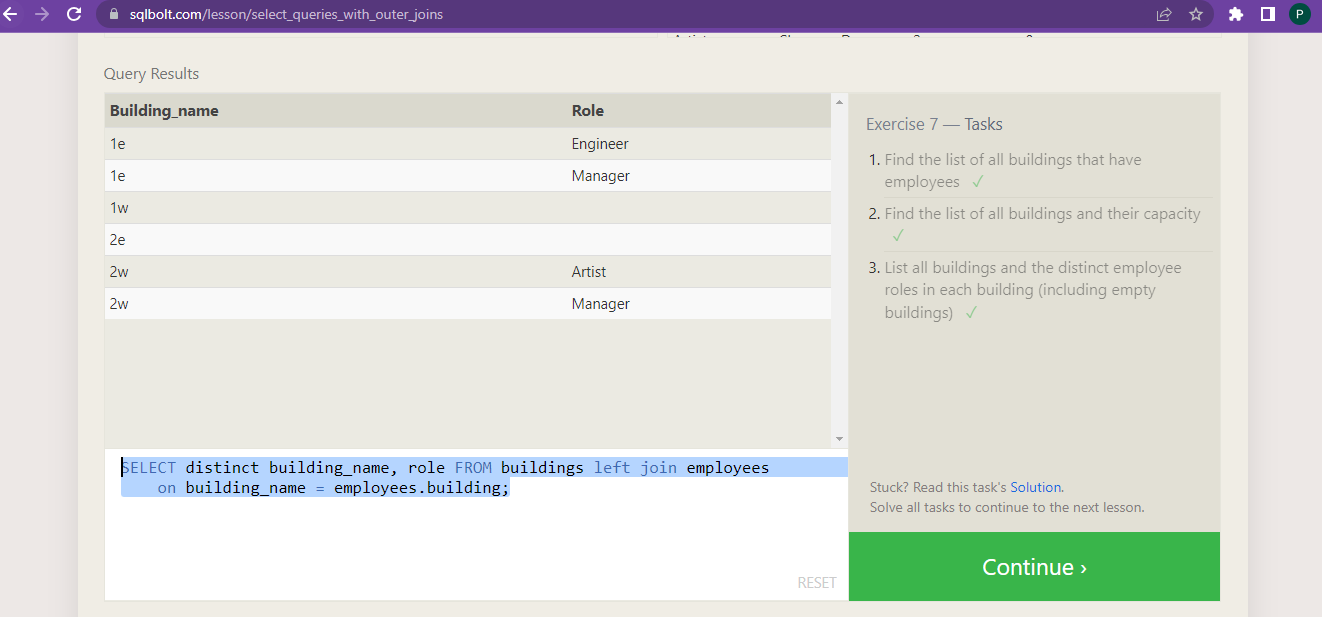
1. Find the list of all buildings and their capacity ✓

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 = employees.building;



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

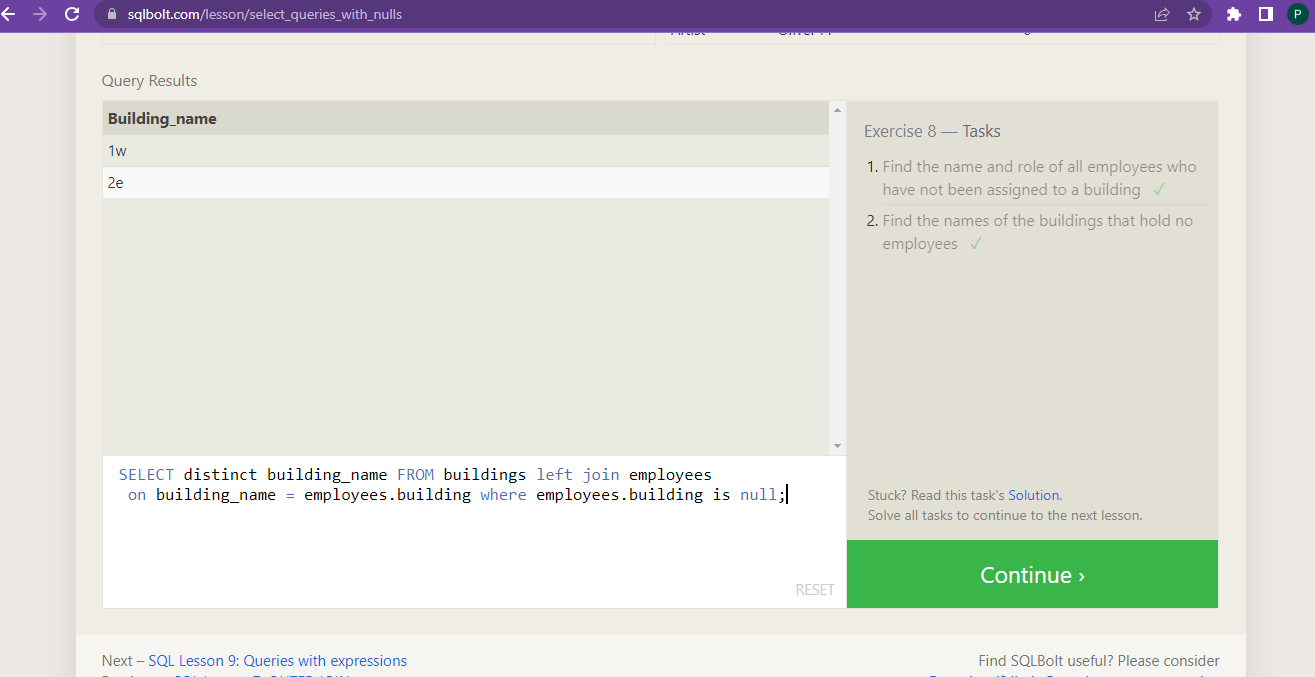
1. Find the name and role of all employees who have not been assigned to a building ✓

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 = employees.building where employees.building is null;



**SQL Lesson 9: Queries with expressions**

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

SELECT distinct title, (domestic\_sales + international\_sales) / 1000000 as sales FROM movies inner join boxoffice on movies.id = boxoffice.movie\_id;

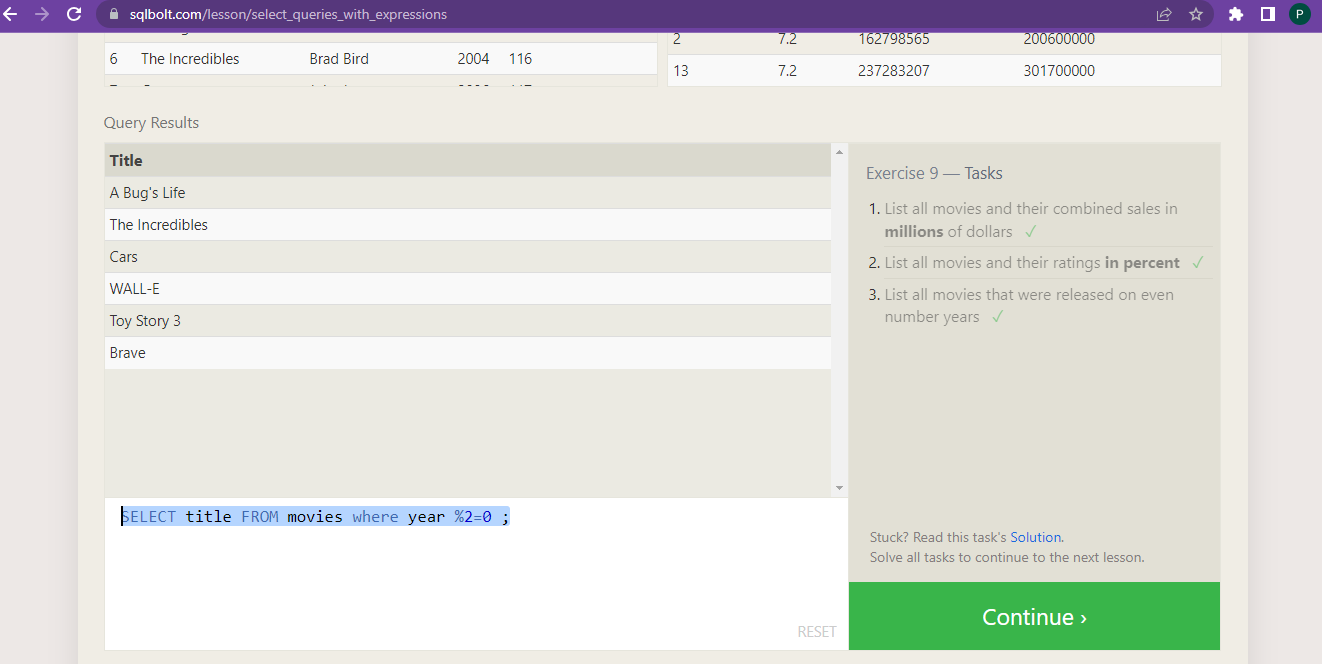
1. List all movies and their ratings **in percent** ✓

SELECT distinct title, (rating \* 10) as rate\_percent FROM movies

inner join boxoffice on movies.id = boxoffice.movie\_id;

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

SELECT title FROM movies 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 sum(Years\_employed) as [longest time],Name from

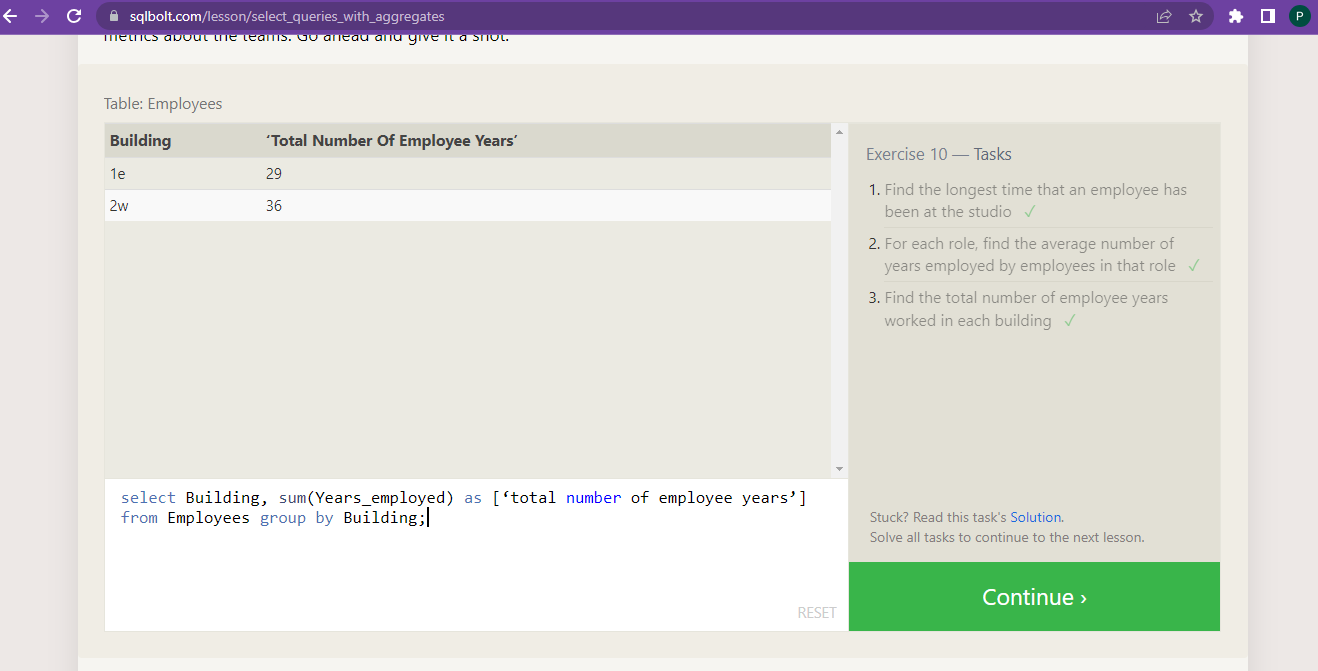
Employees group by Name order by [Longest Time] desc limit 1;

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;

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

select Building, sum(Years\_employed) as [‘total number of employee years’]   
from Employees group by Building;



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

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

SELECT role, COUNT(\*) as Number\_of\_artists

FROM employees

WHERE role = "Artist";

2. Find the number of Employees of each role in the studio ✓

select Role, count(Role) as [number of Employees] from Employees group by Role;

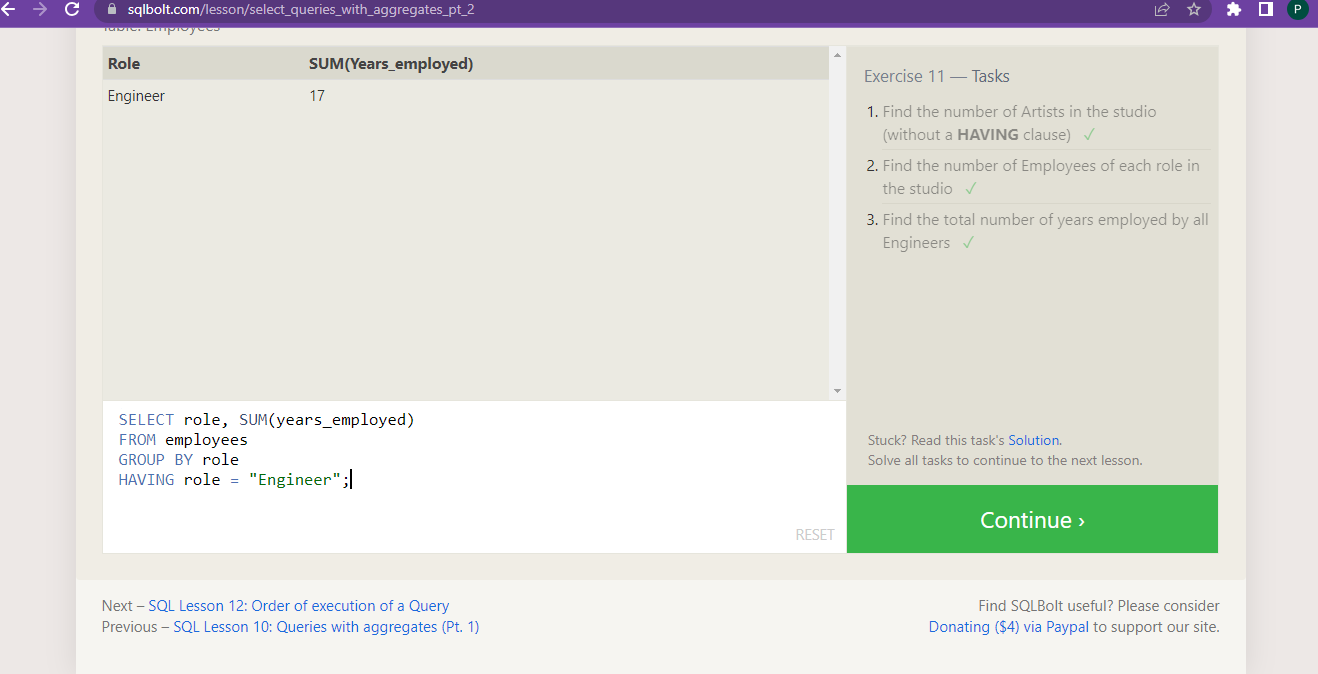
3. Find the total number of years employed by all Engineers ✓

SELECT role, SUM(years\_employed)

FROM employees

GROUP BY role

HAVING role = "Engineer";



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

1. Find the number of movies each director has directed ✓

select Director, count(Title) as [number of movies] from Movies

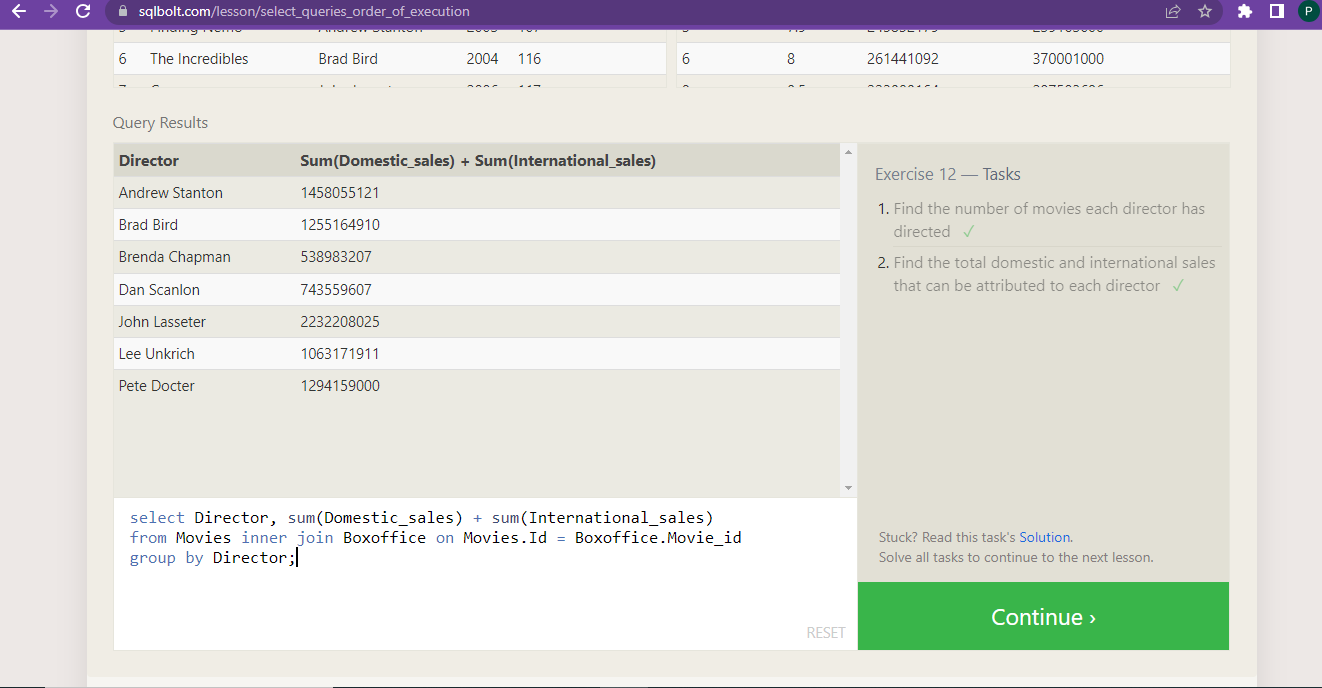
group by Director;

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

select Director, sum(Domestic\_sales) + sum(International\_sales)

from Movies inner join Boxoffice on Movies.Id = Boxoffice.Movie\_id

group by Director;



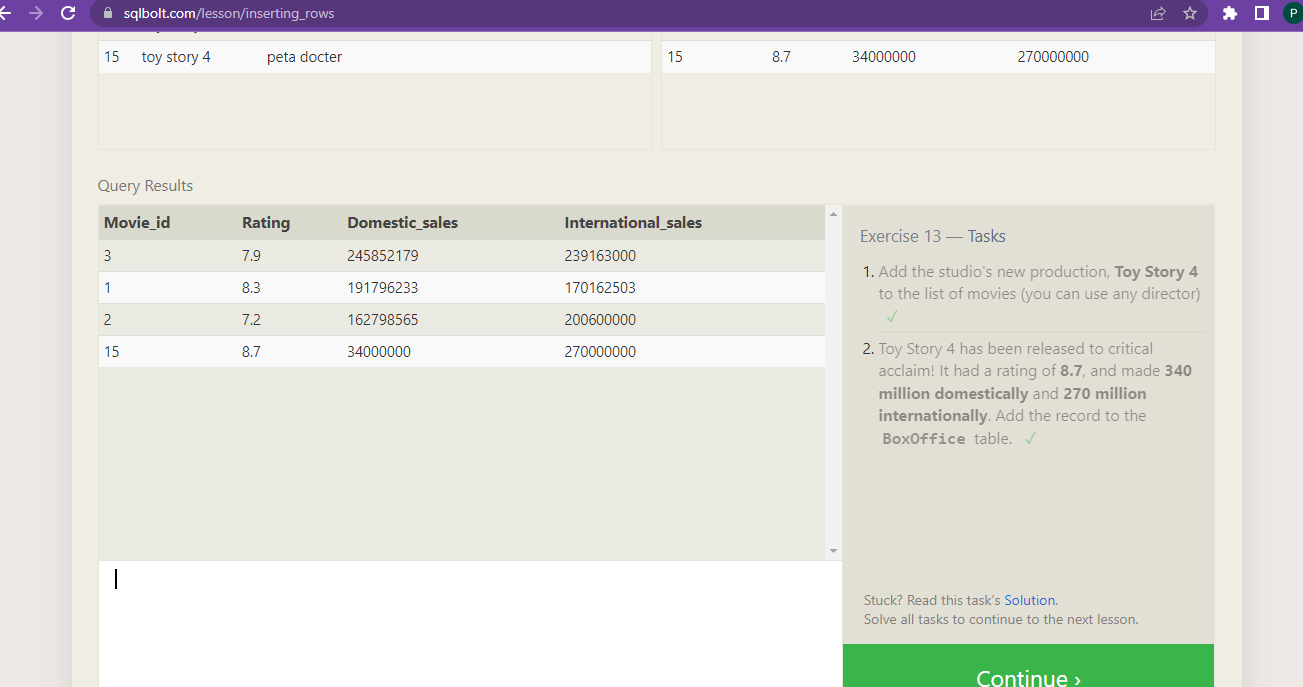
**SQL Lesson 13: Inserting rows**

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

Insert into Movies (Title, director) values(‘Toy Story 4’, ‘peta docter’);

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 (Movie\_id,rating,Domestic\_sales,International\_sales) values('15',' 8.7','34000000','270000000');



**SQL Lesson 14: Updating rows**

1. The director for A Bug's Life is incorrect, it was actually directed by **John Lasseter** ✓

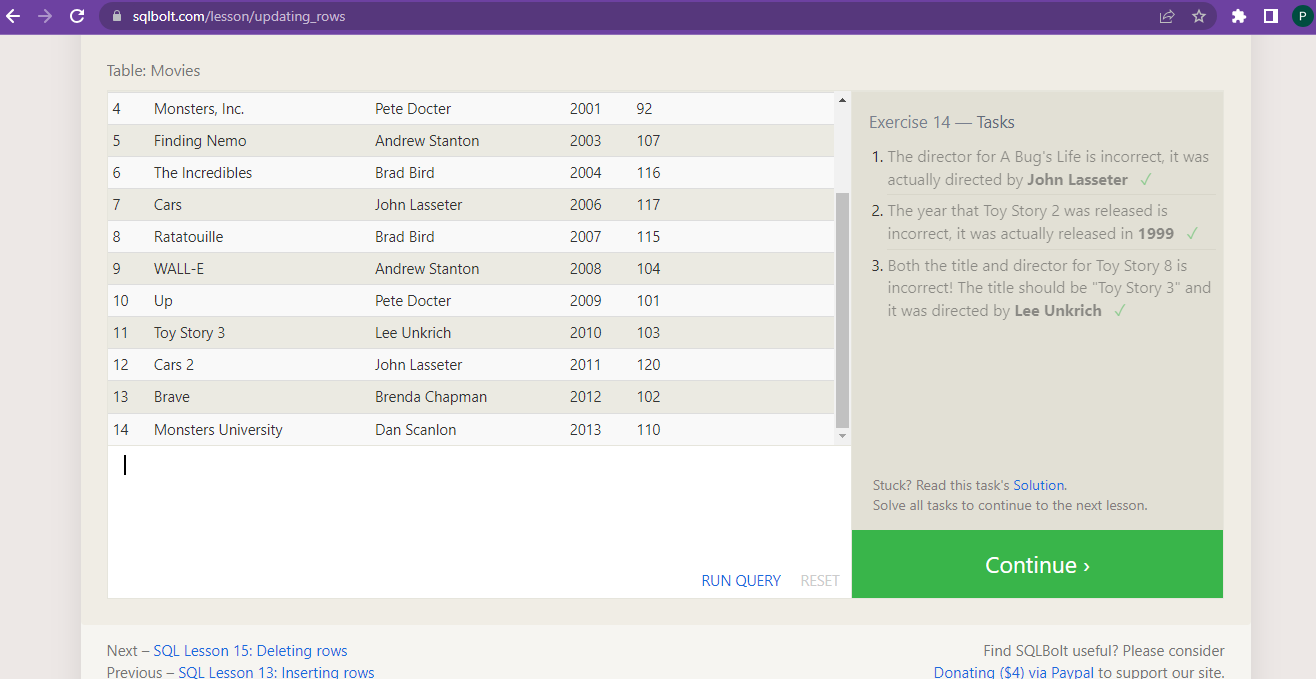
update Movies set Director = ‘John Lasseter’ where Title = “A Bug’s Life”;

1. The year that Toy Story 2 was released is incorrect, it was actually released in **1999** ✓

Update Movies set Year=’1999’ where Title = “Toy Story 2”;

1. 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** ✓

Update Movies set Title = ‘Toy Story 3’, Director = ‘Lee Unkrich’ where Title = “Toy Story 8”;



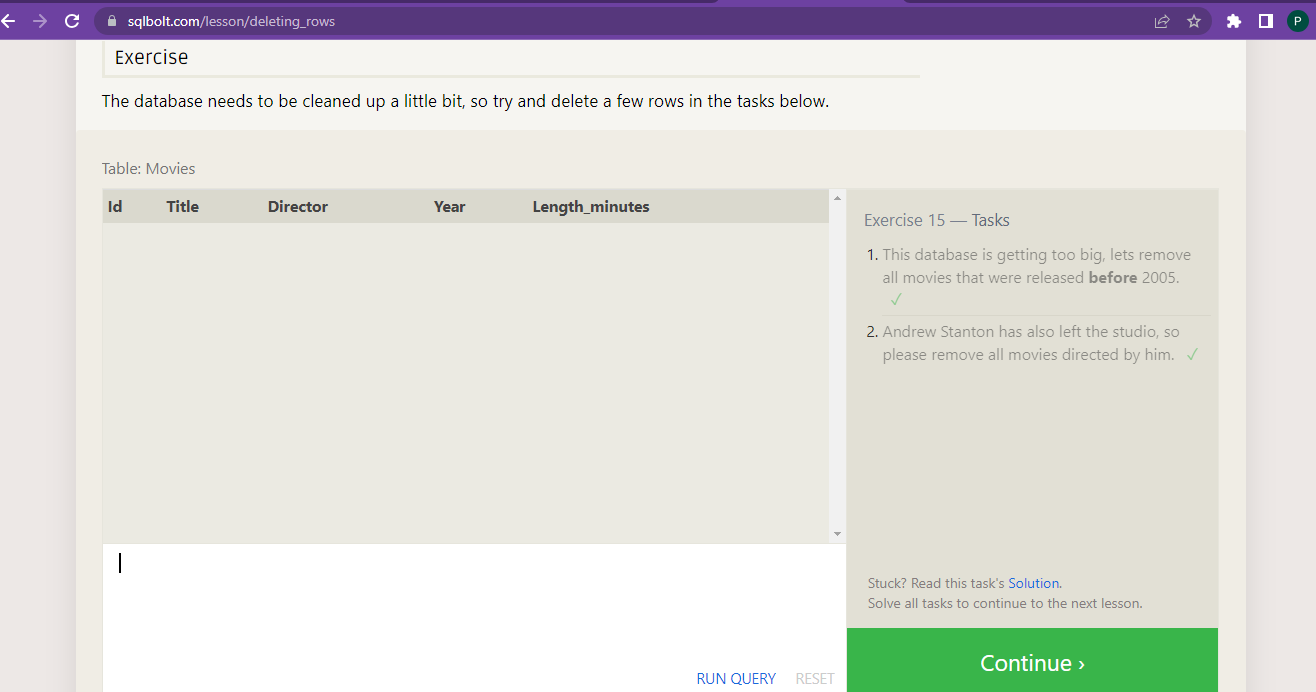
**SQL Lesson 15: Deleting rows**

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

Delete from Movies where year<2005;

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

Delete from Movies where Director=’ Andrew Stanton’;



**SQL Lesson 16: Creating tables**

1. 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

This table has no constraints. ✓

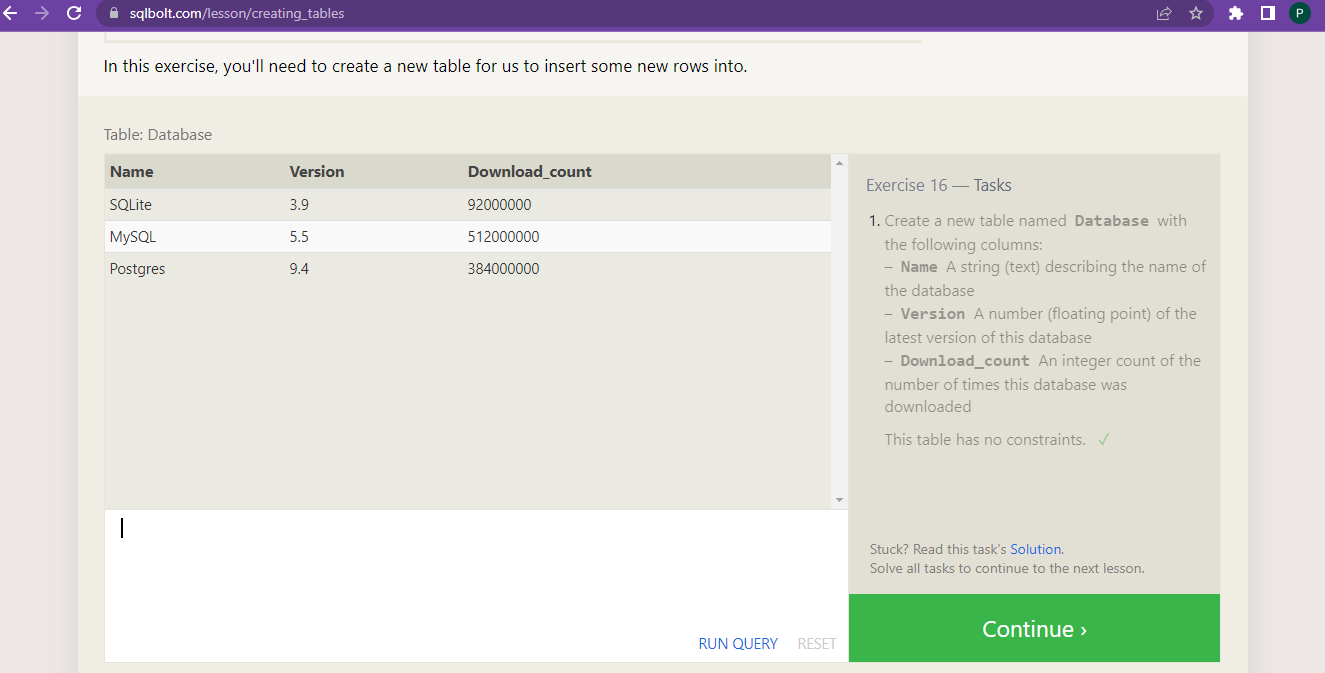
Create table Database (

Name text,

Version float,

Download\_count int,

);



**SQL Lesson 17: Altering tables**

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

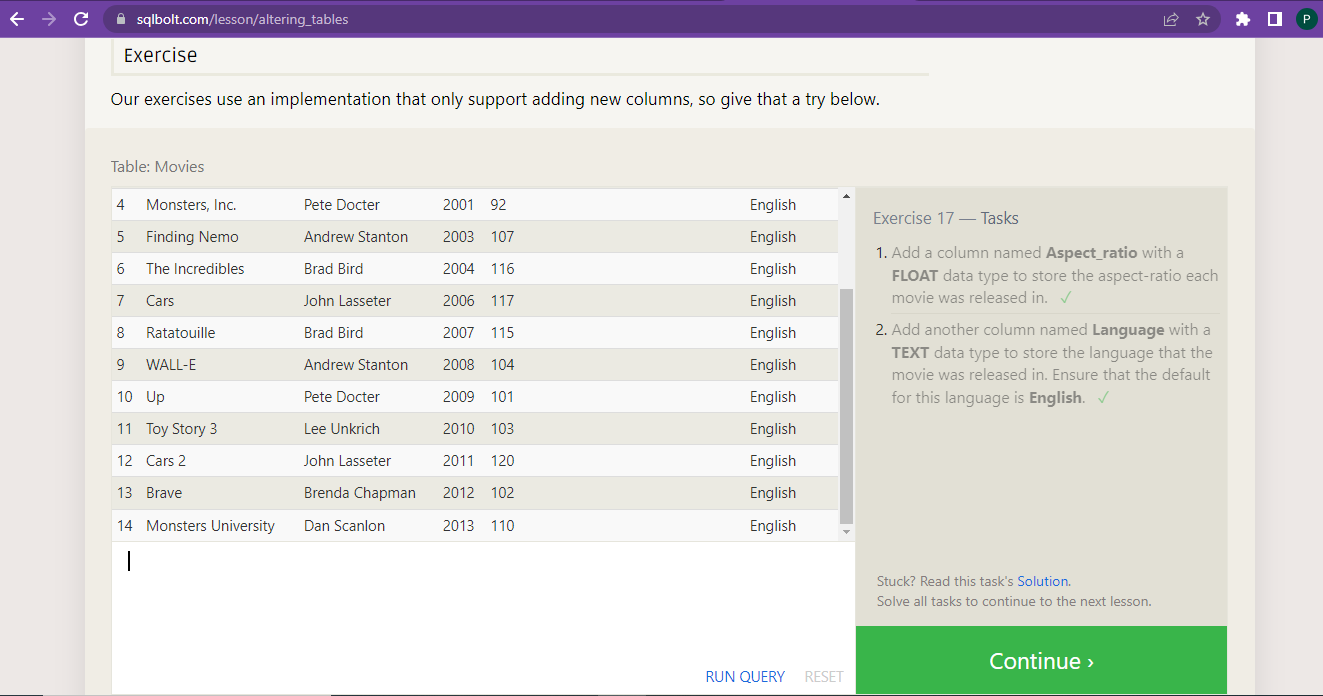
Alter table Movies add Aspect\_ratio float;

1. 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**. ✓

Alter table Movies

Add Language text

Default English:



**SQL Lesson 18: Dropping tables**

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

Drop table Movies;

2. And drop the **BoxOffice** table as well ✓

Drop table BoxOffice;

