# **SQL Task**

**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

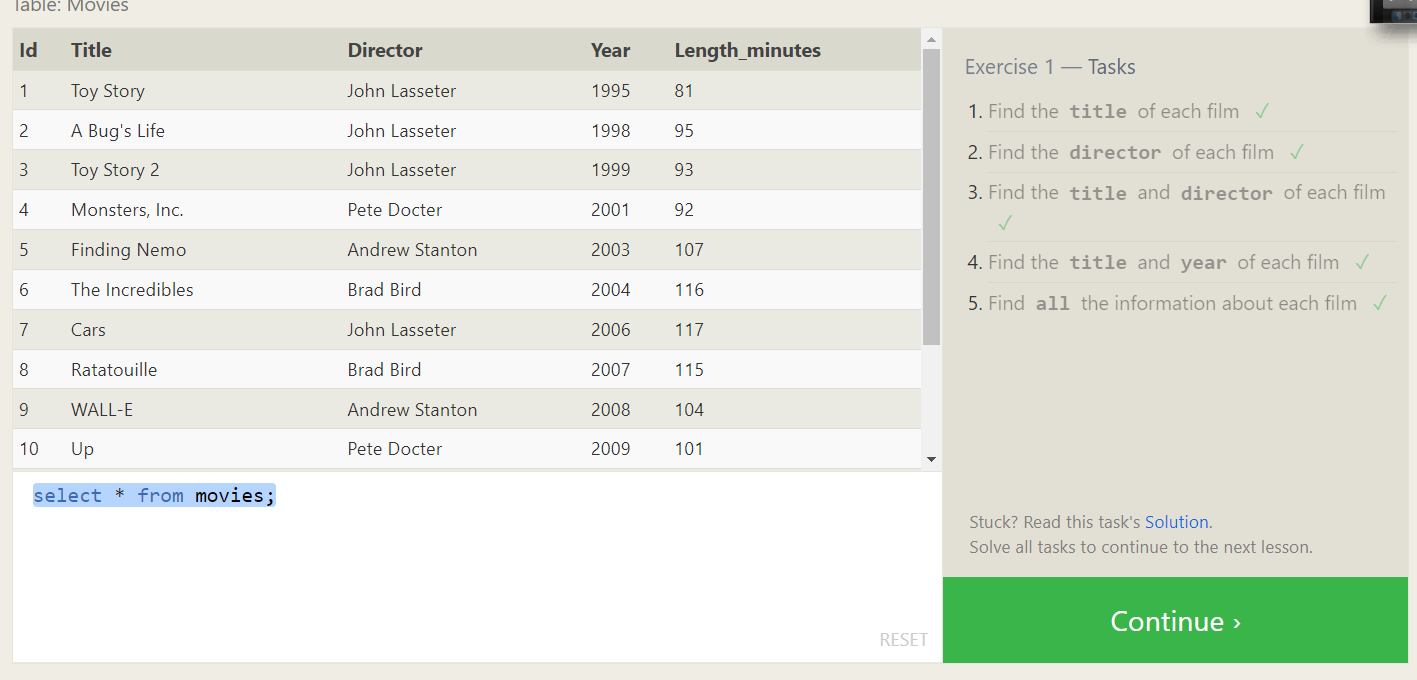
**select title,director from movies;**

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

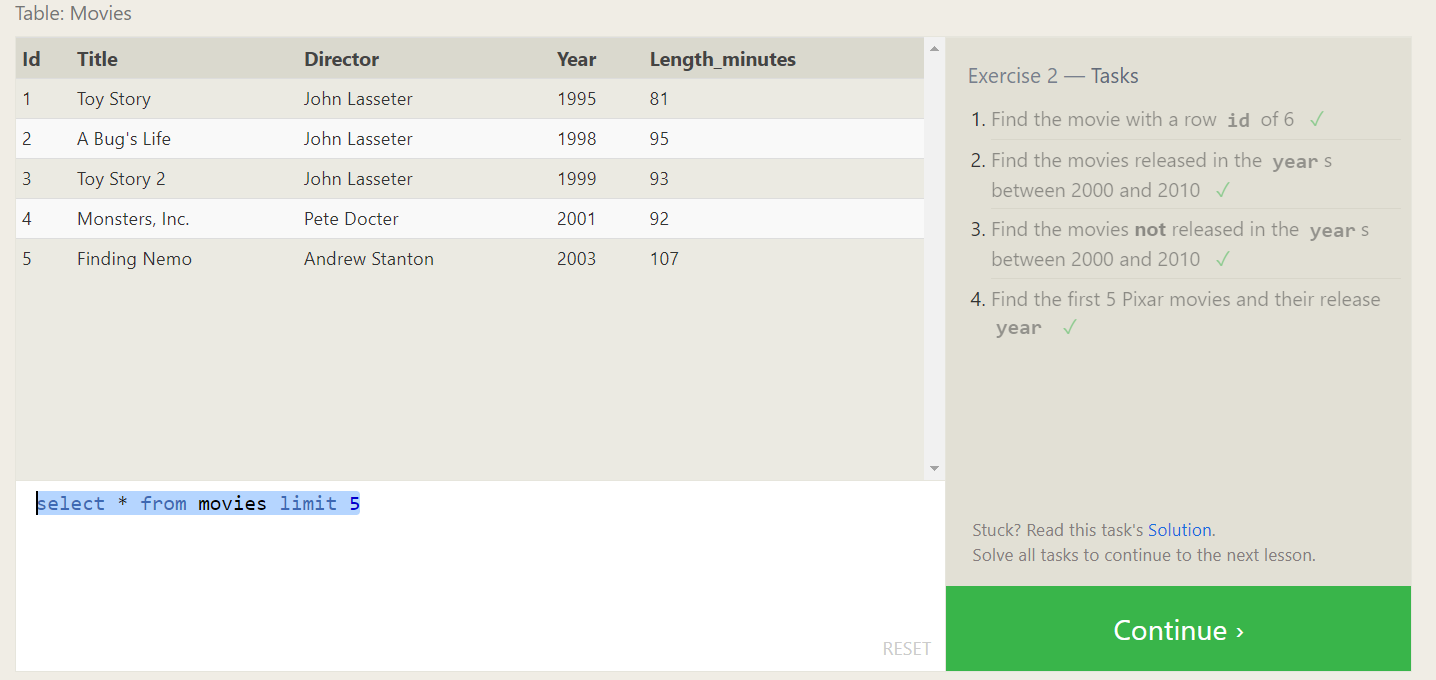
**select \* from movies where year between 2000 and 2010;**

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 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%';**

1. Find all the movies directed by John Lasseter

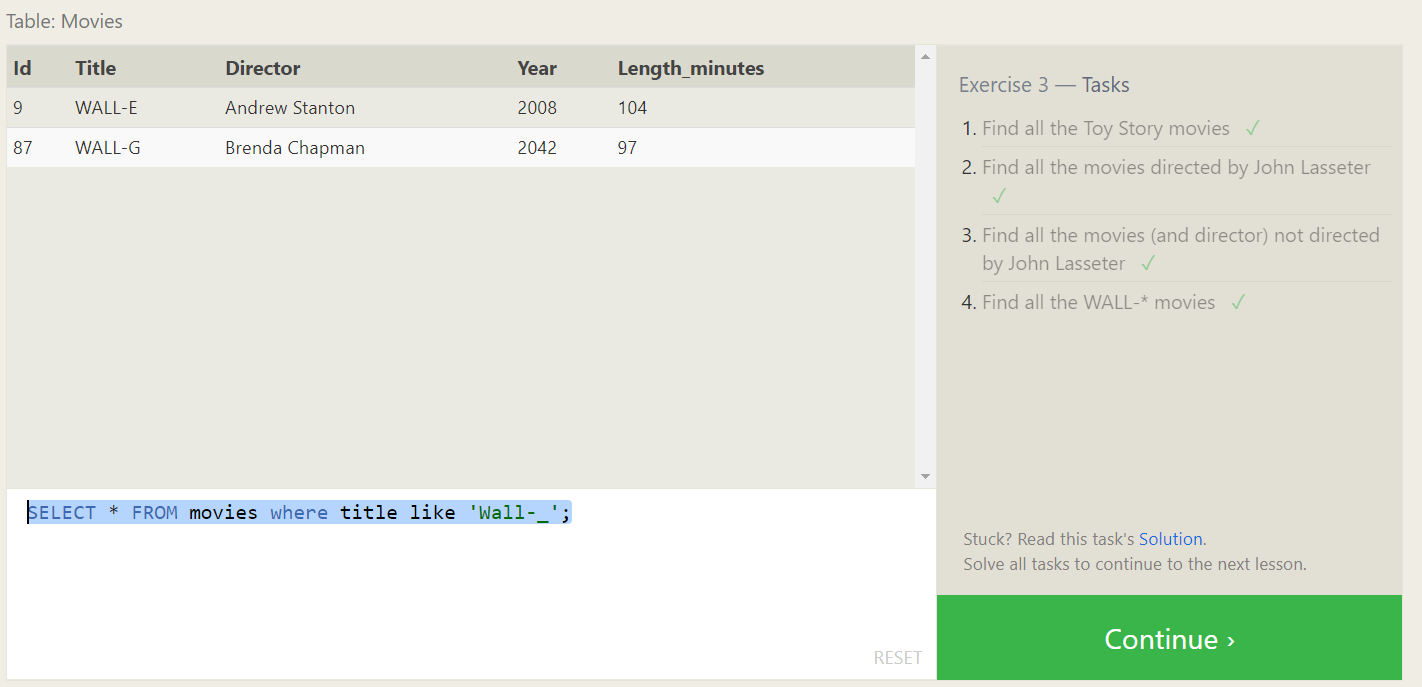
**SELECT \* FROM movies where director like 'John Lasseter%';**

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

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

1. Find all the WALL-\* movies

**SELECT \* 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 asc;**

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

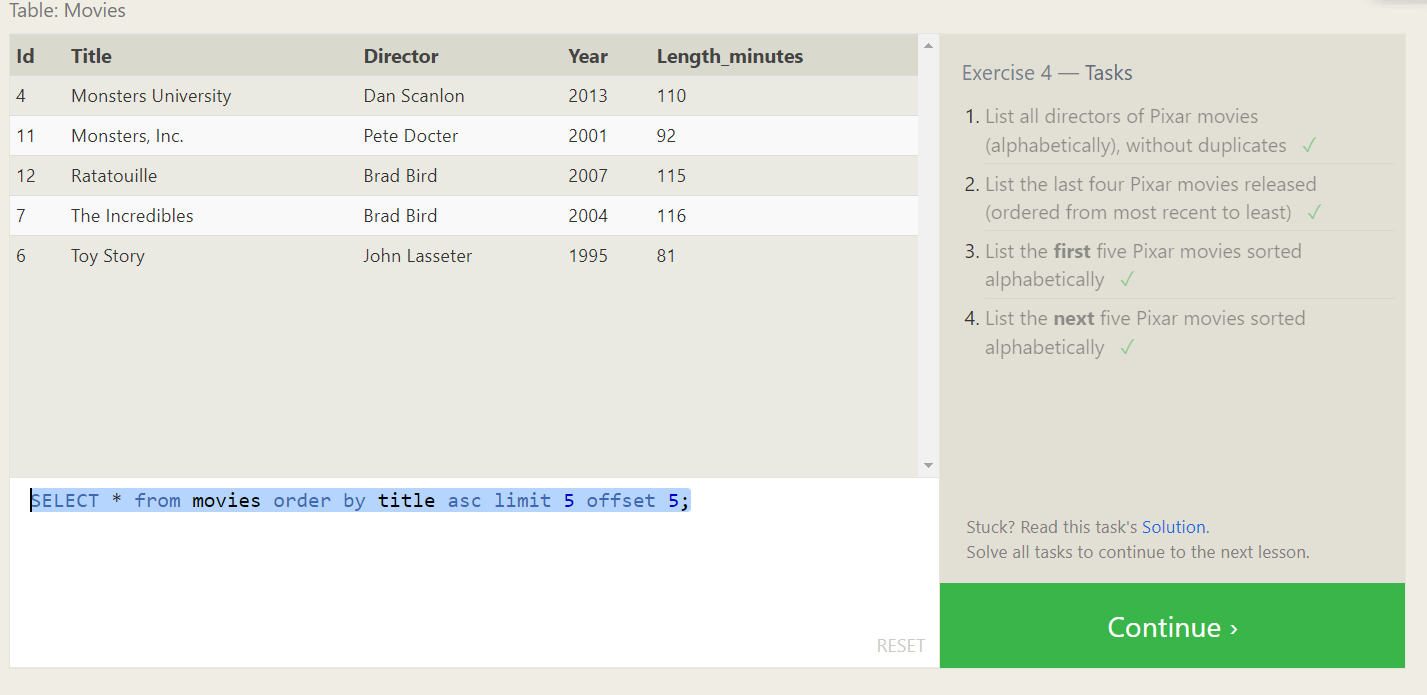
**SELECT \* from movies order by year desc limit 4;**

1. List the first five Pixar movies sorted alphabetically

**SELECT \* from movies order by title asc limit 5;**

1. List the next five Pixar movies sorted alphabetically

**SELECT \* from movies order by title asc limit 5 offset 5;**



**SQL Review: Simple SELECT Queries**

1. List all the Canadian cities and their populations

**SELECT \* from North\_american\_cities where country like 'canada';**

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

**SELECT \* from North\_american\_cities where country like 'united states'order by latitude desc;**

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

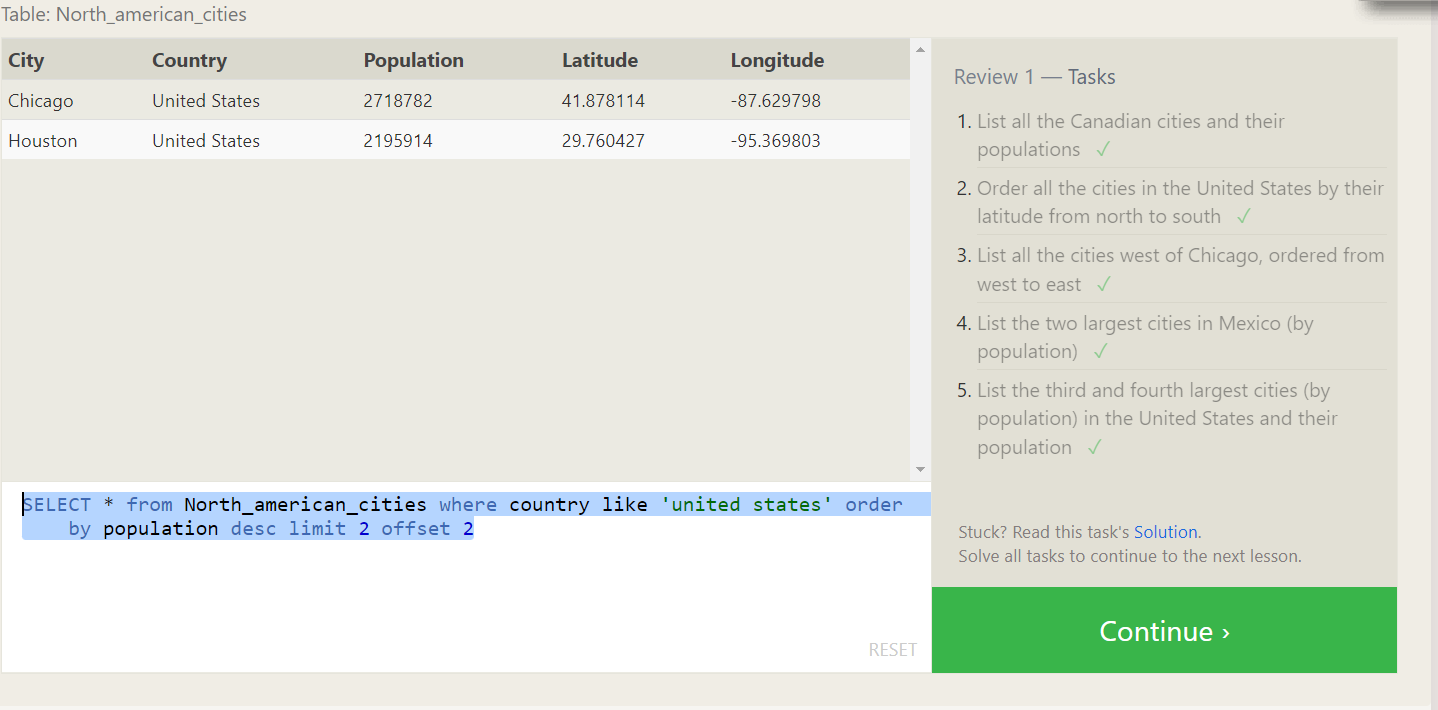
**SELECT \* from North\_american\_cities where longitude < -87.629798 order by longitude asc;**

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

**SELECT \* from North\_american\_cities where country like '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 like '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

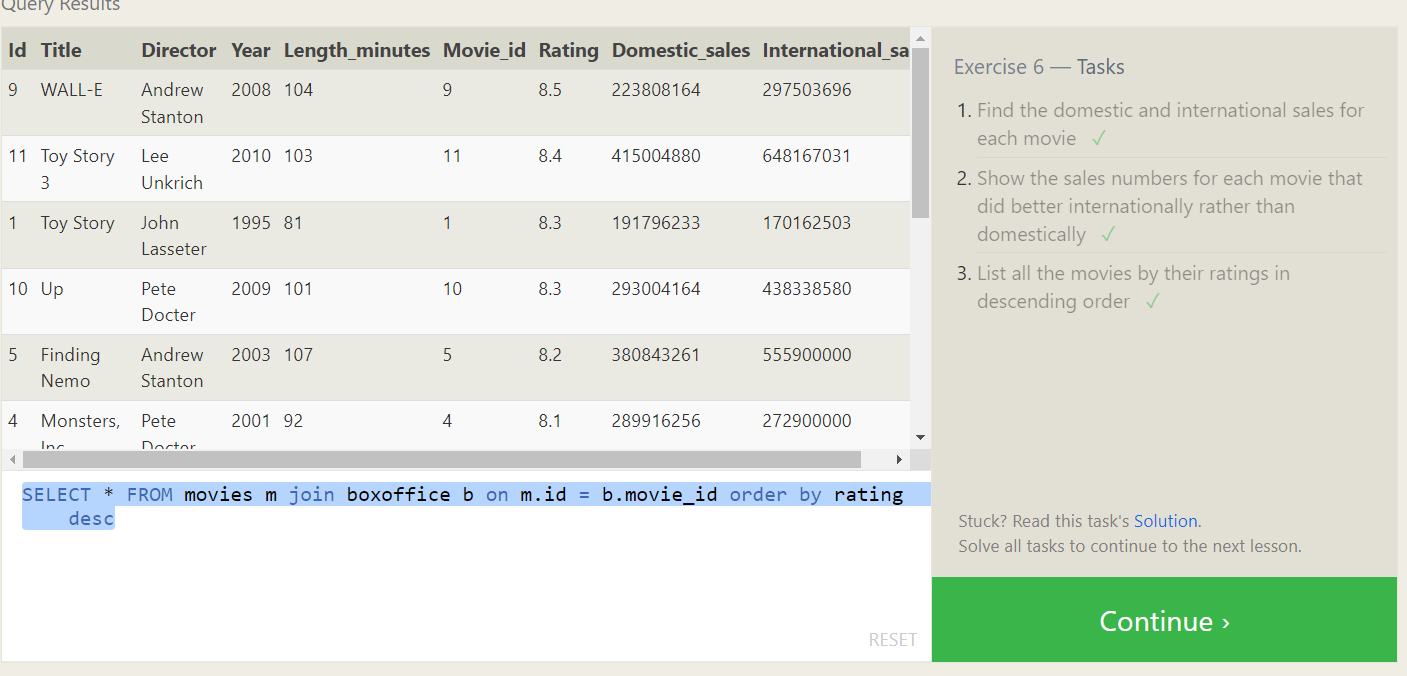
**SELECT \* FROM movies m join boxoffice b where m.id = b.movie\_id;**

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

**SELECT \* FROM movies m 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 join boxoffice b on m.id = b.movie\_id order by rating desc;**



**SQL Lesson 7: OUTER JOINs**

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

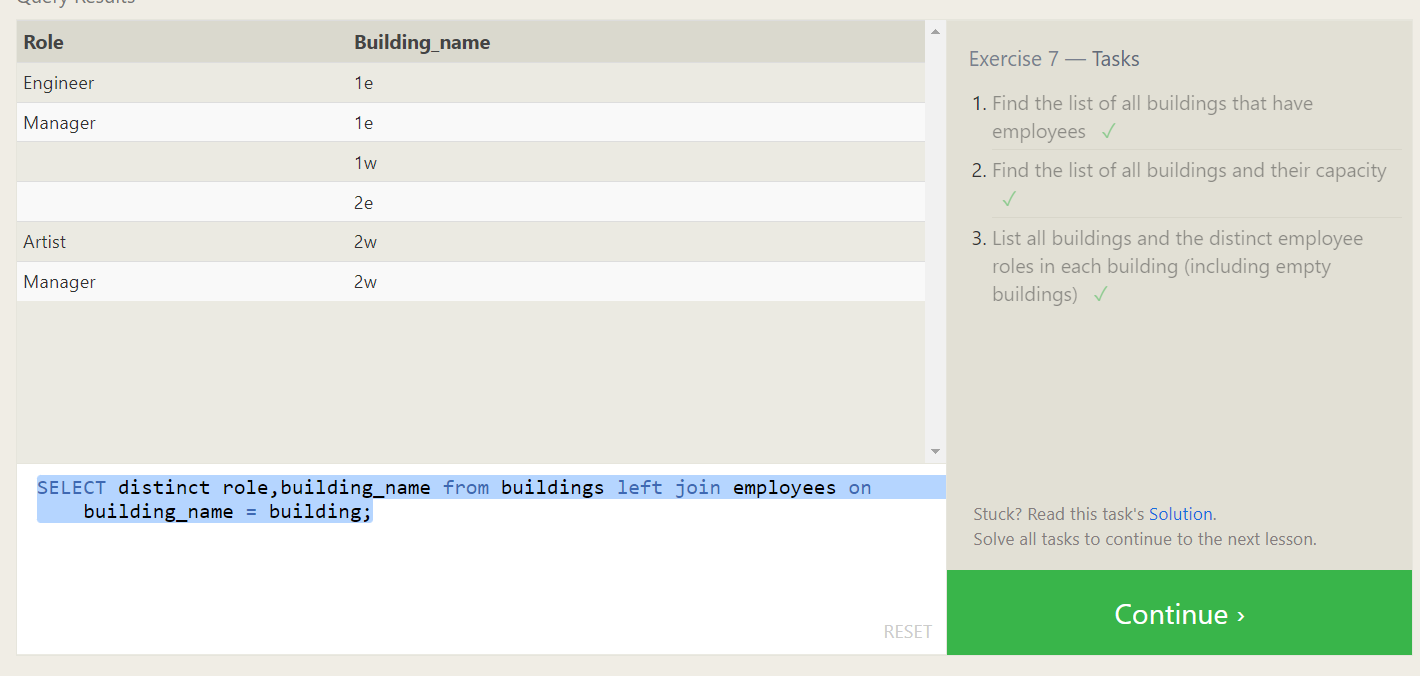
**SELECT distinct building** **FROM employees;**

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 role,building\_name from buildings left join employees on building\_name = building;**



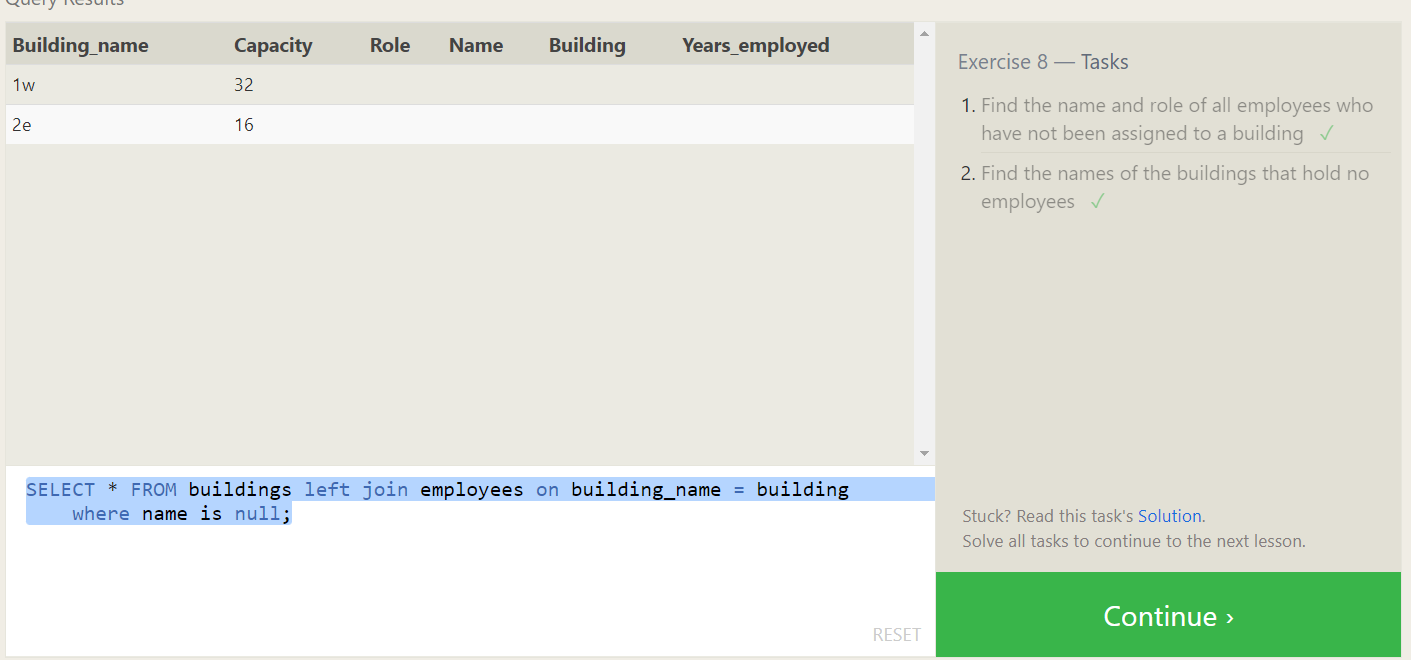
**SQL Lesson 8: A short note on NULLs**

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

**SELECT \* FROM employees where building is null;**

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

**SELECT \* FROM buildings left join employees on building\_name = building where name 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) / 1000000 AS sales**

**FROM movies**

**JOIN boxoffice**

**ON movies.id = boxoffice.movie\_id;**

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

**SELECT title, (Rating \*10)**

**FROM movies**

**JOIN boxoffice**

**ON movies.id = boxoffice.movie\_id;**

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

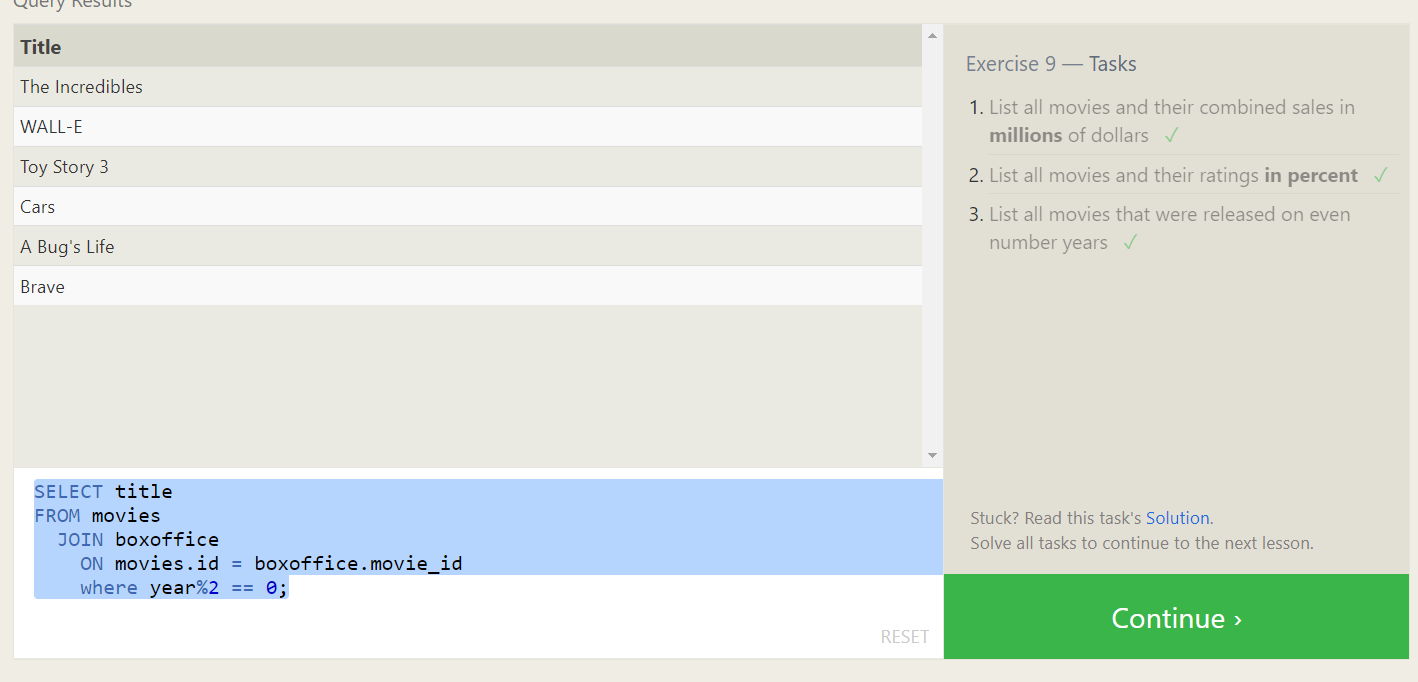
**SELECT title**

**FROM movies**

**JOIN boxoffice**

**ON movies.id = boxoffice.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.

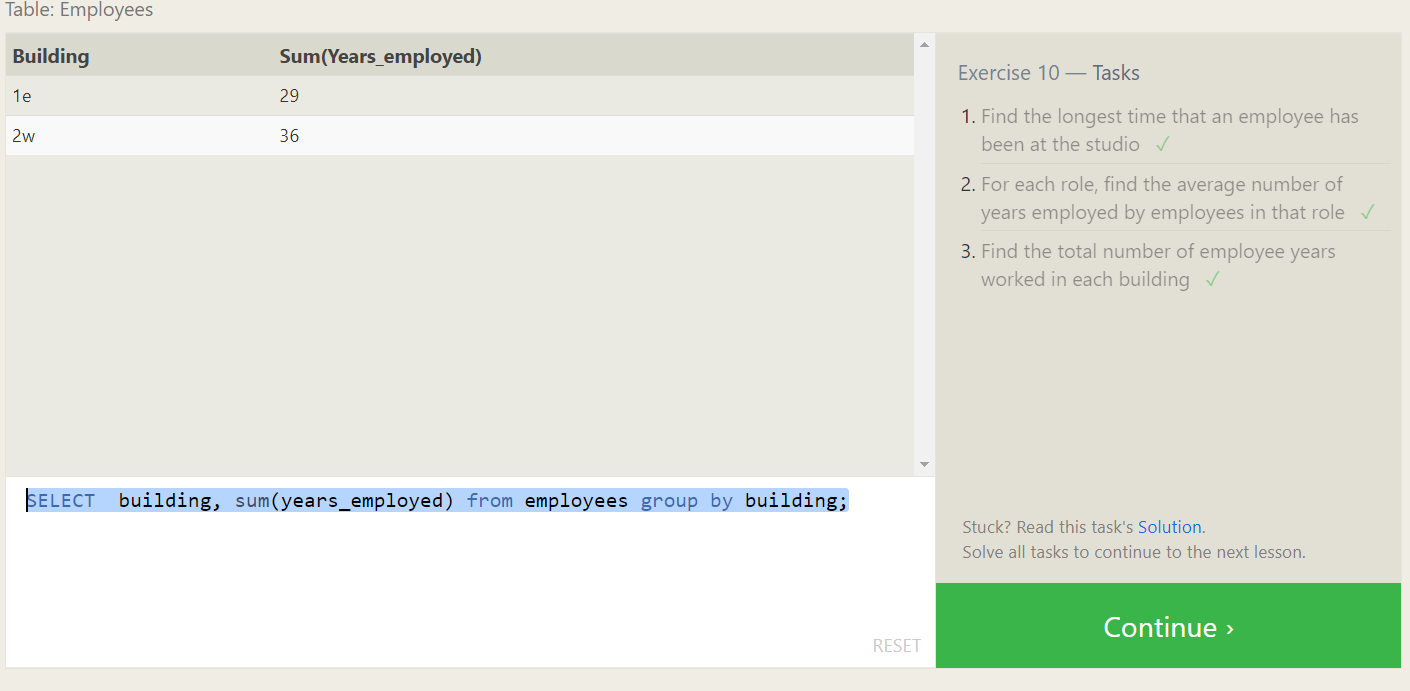
**SELECT max(Years\_employed) from employees;**

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

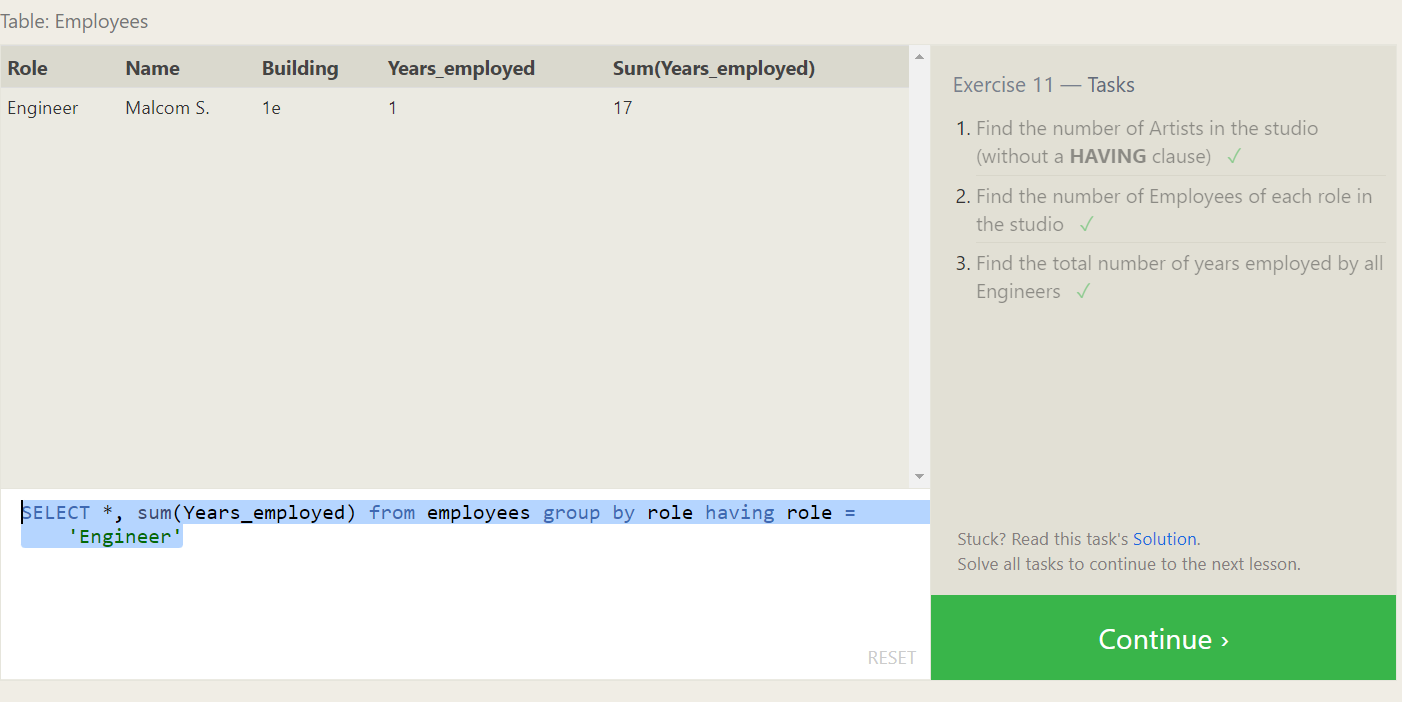
**SELECT count() FROM employees where role = 'Artist';**

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

**SELECT \*, count() from employees group by role;**

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

**SELECT \*, 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() FROM movies group by director;**

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

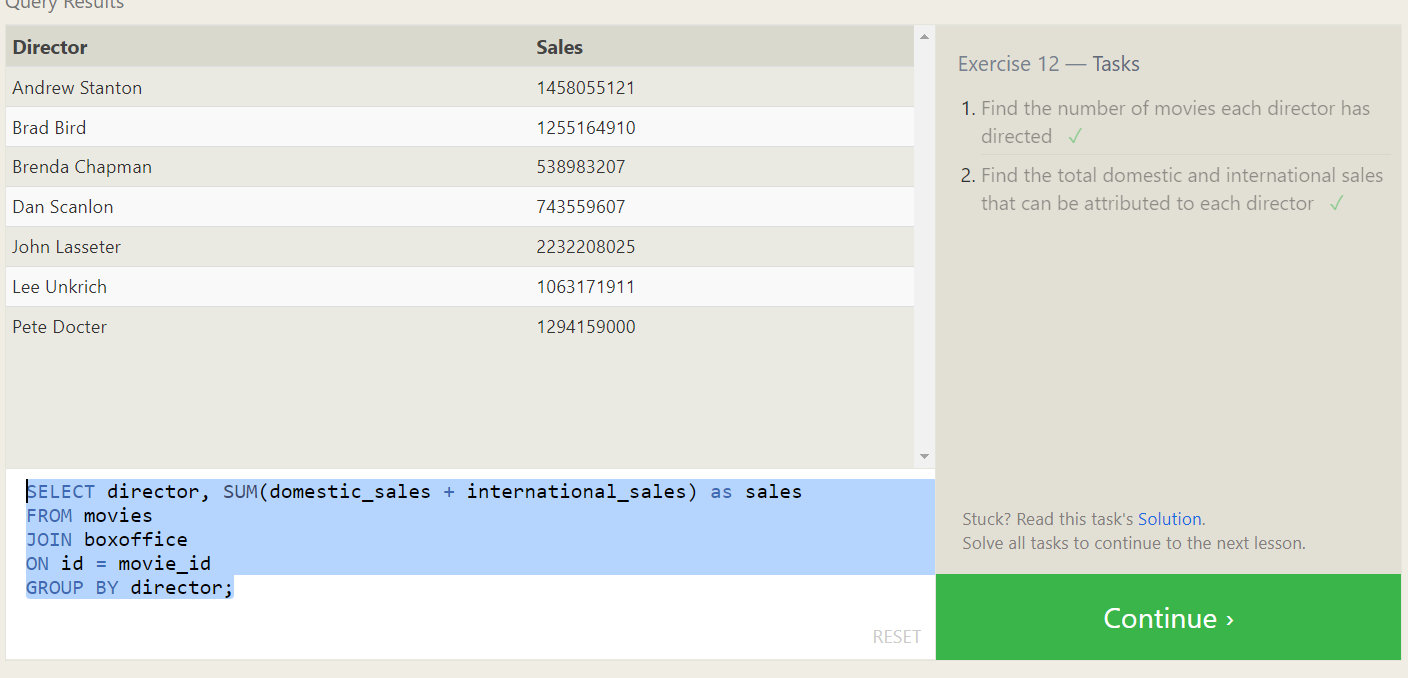
**SELECT director, SUM(domestic\_sales + international\_sales) as sales**

**FROM movies**

**JOIN boxoffice**

**ON id = movie\_id**

**GROUP BY director;**



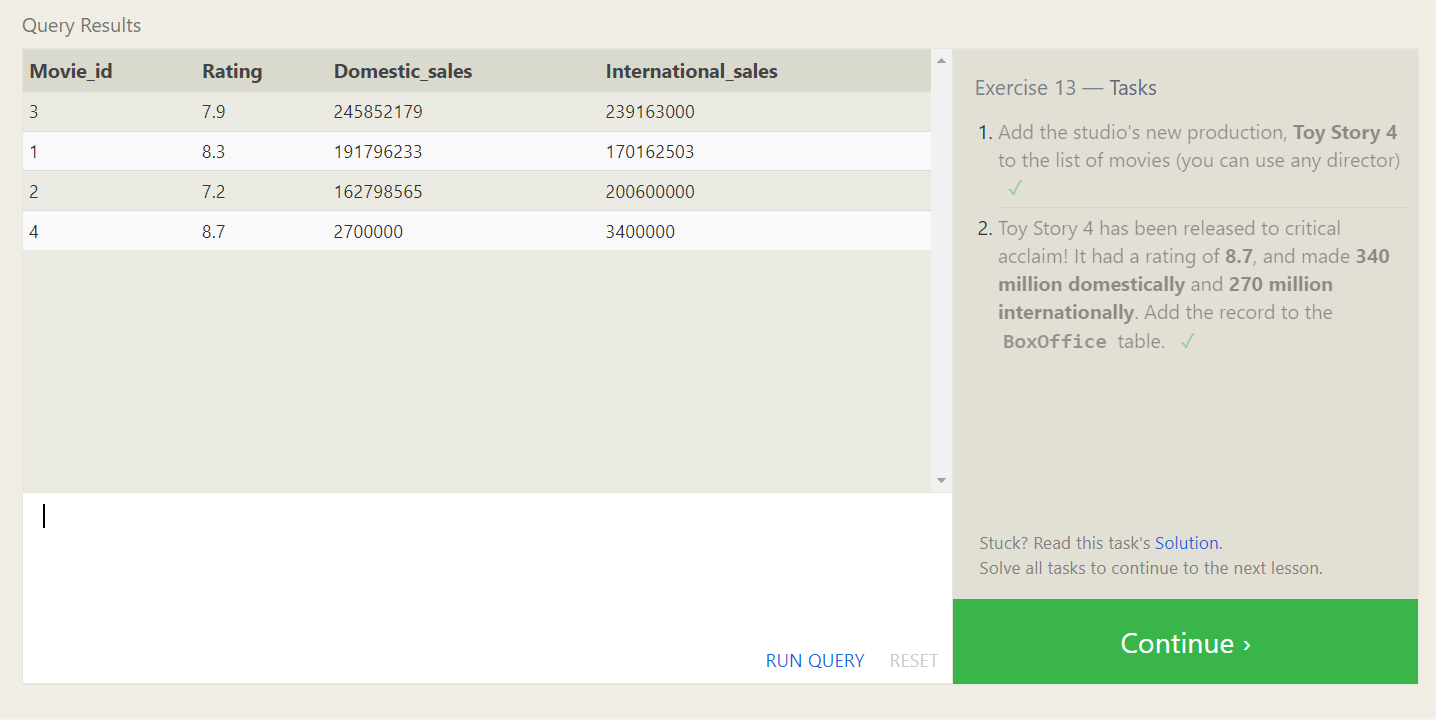
**SQL Lesson 13: Inserting rows**

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',2020,200);**

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(4,8.7,2700000,3400000);**



**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 id=2;**

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

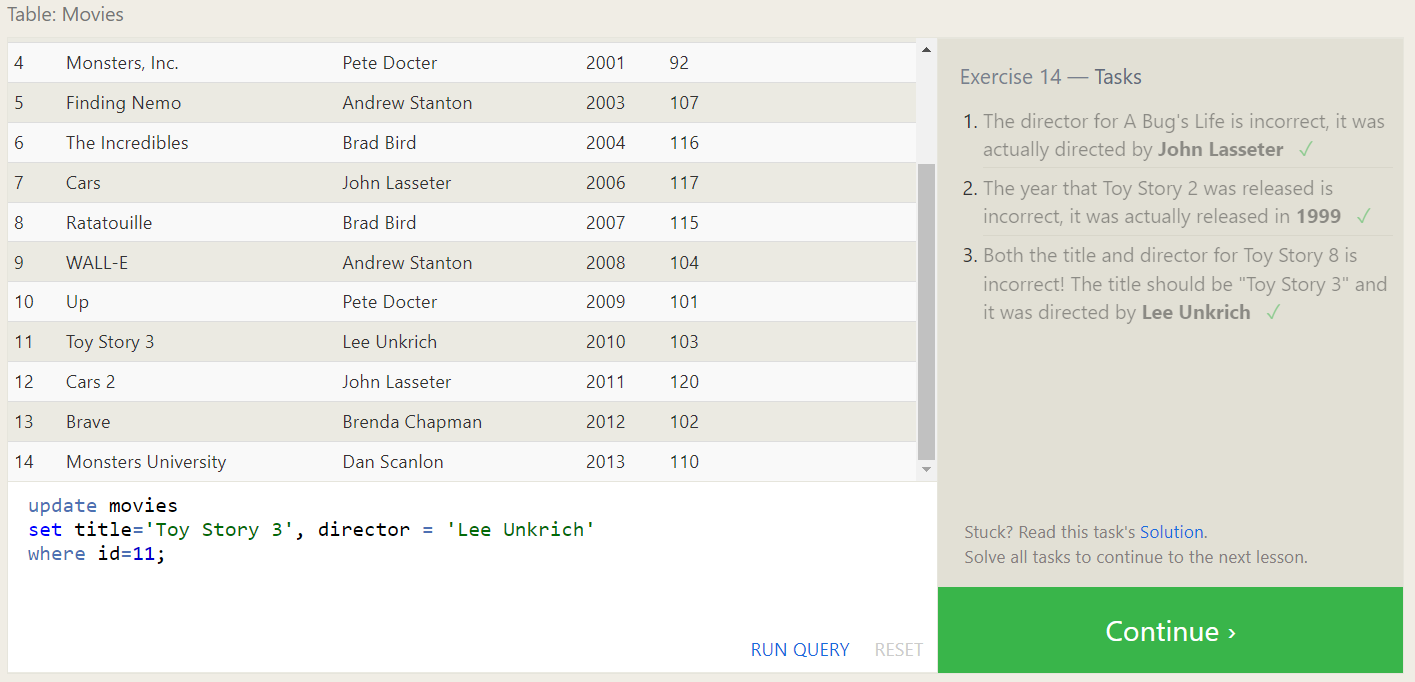
**update movies set year=1999 where id=3;**

3. 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 id=11;**



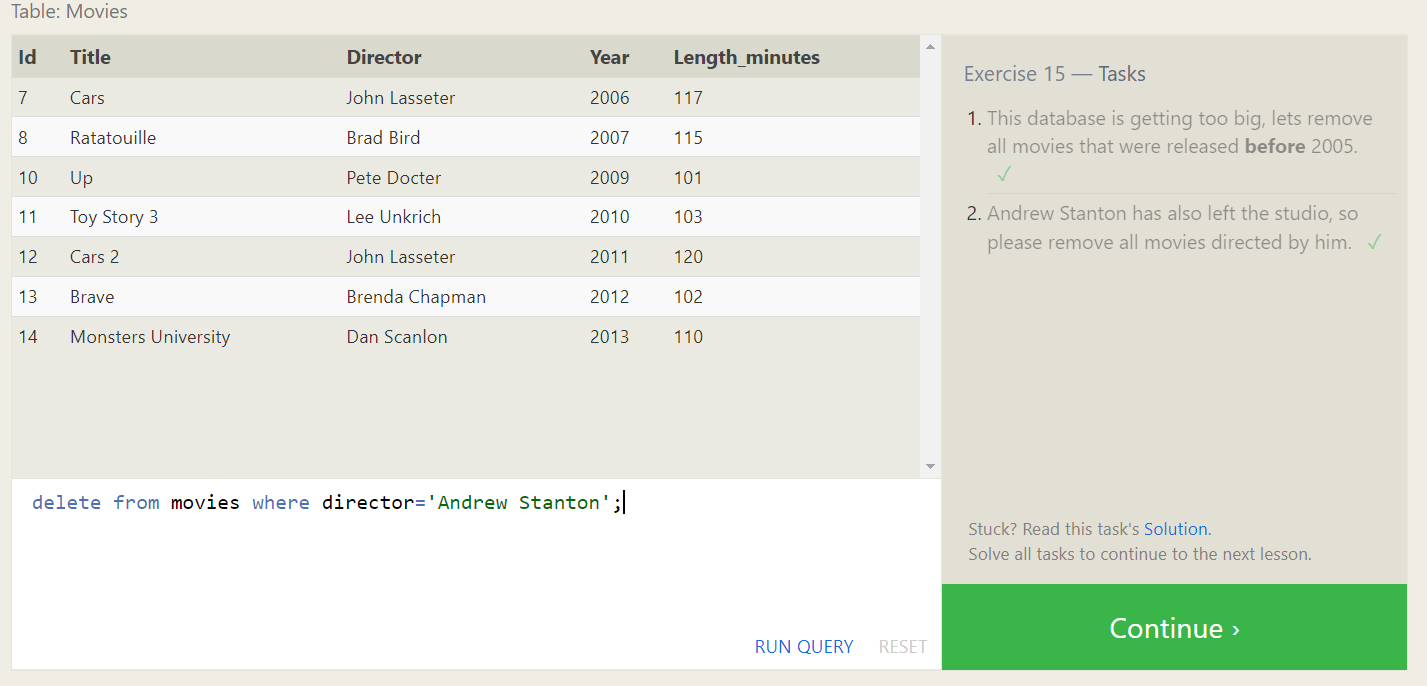
**SQL Lesson 15: Deleting rows**

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 (**

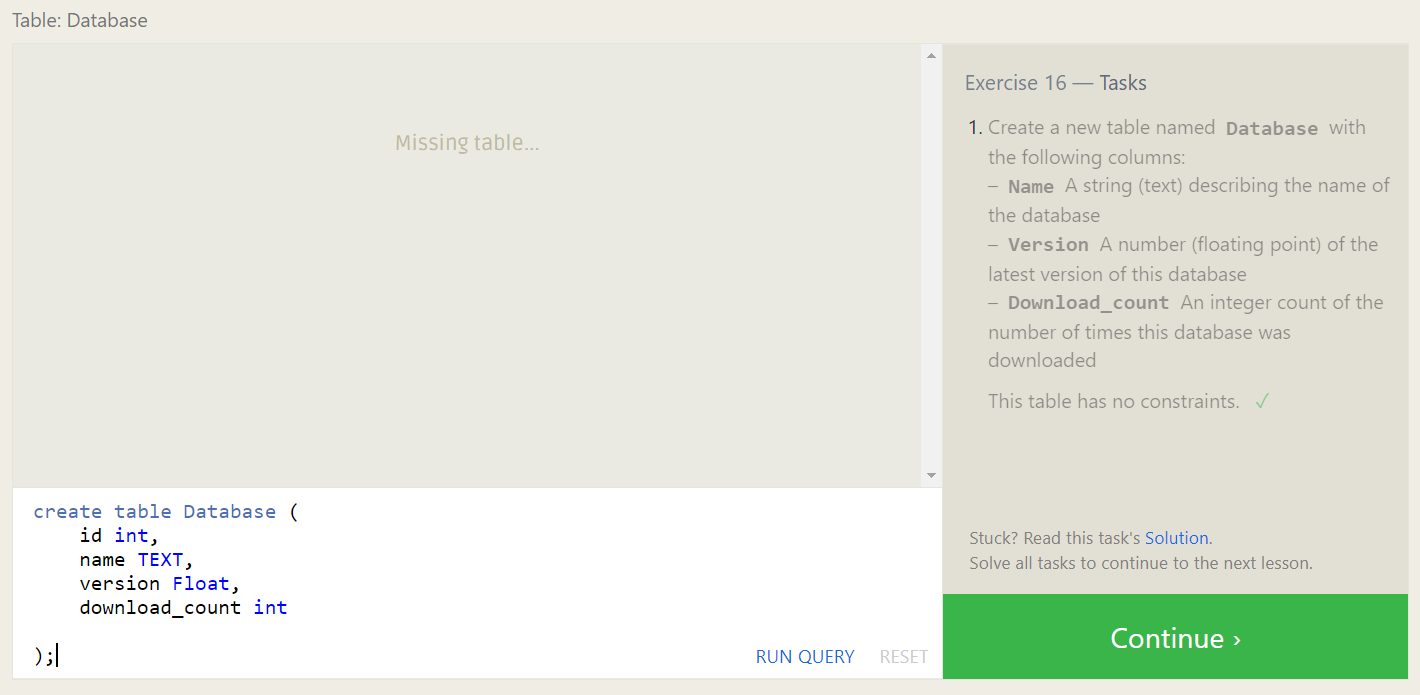
**id int,**

**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.

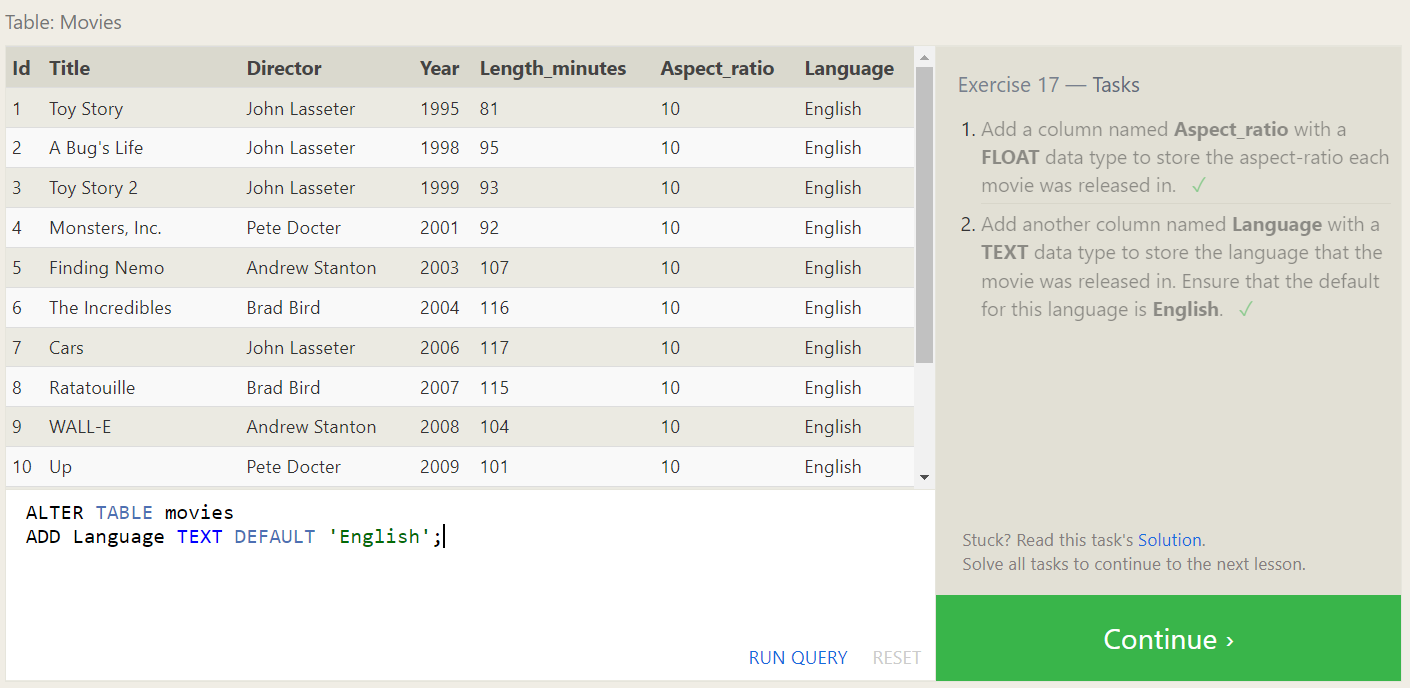
**ALTER TABLE movies**

**ADD Aspect\_ratio Float DEFAULT 10.0;**

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;**

1. And drop the BoxOffice table as well

**drop table boxoffice;**

