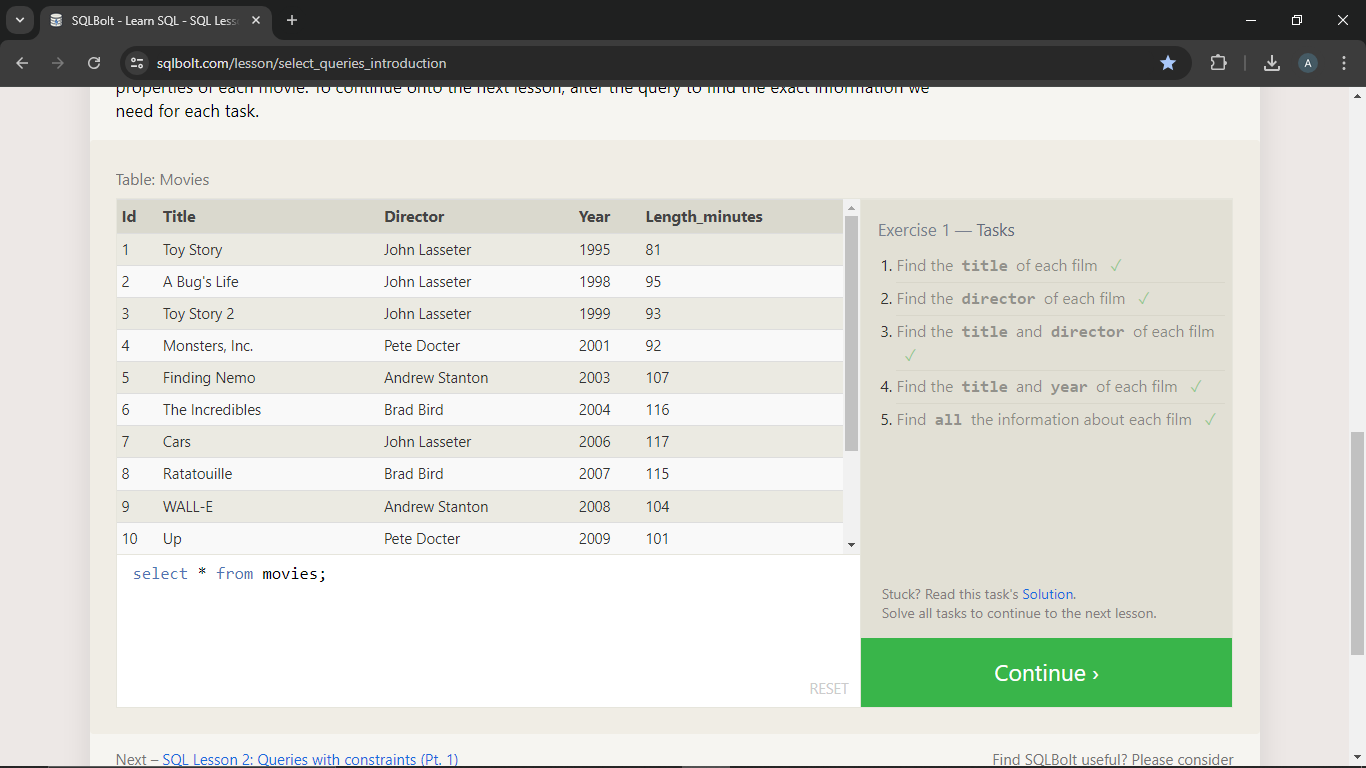
**SQL TASK**

**SQL Lesson 1: SELECT queries 101 :**

****

1. Find the **title** of each film

Query : select Title from movies;

1. Find the **director** of each film

Query : select director from movies;

1. Find the **title** and **director** of each film

Query : select title,director from movies;

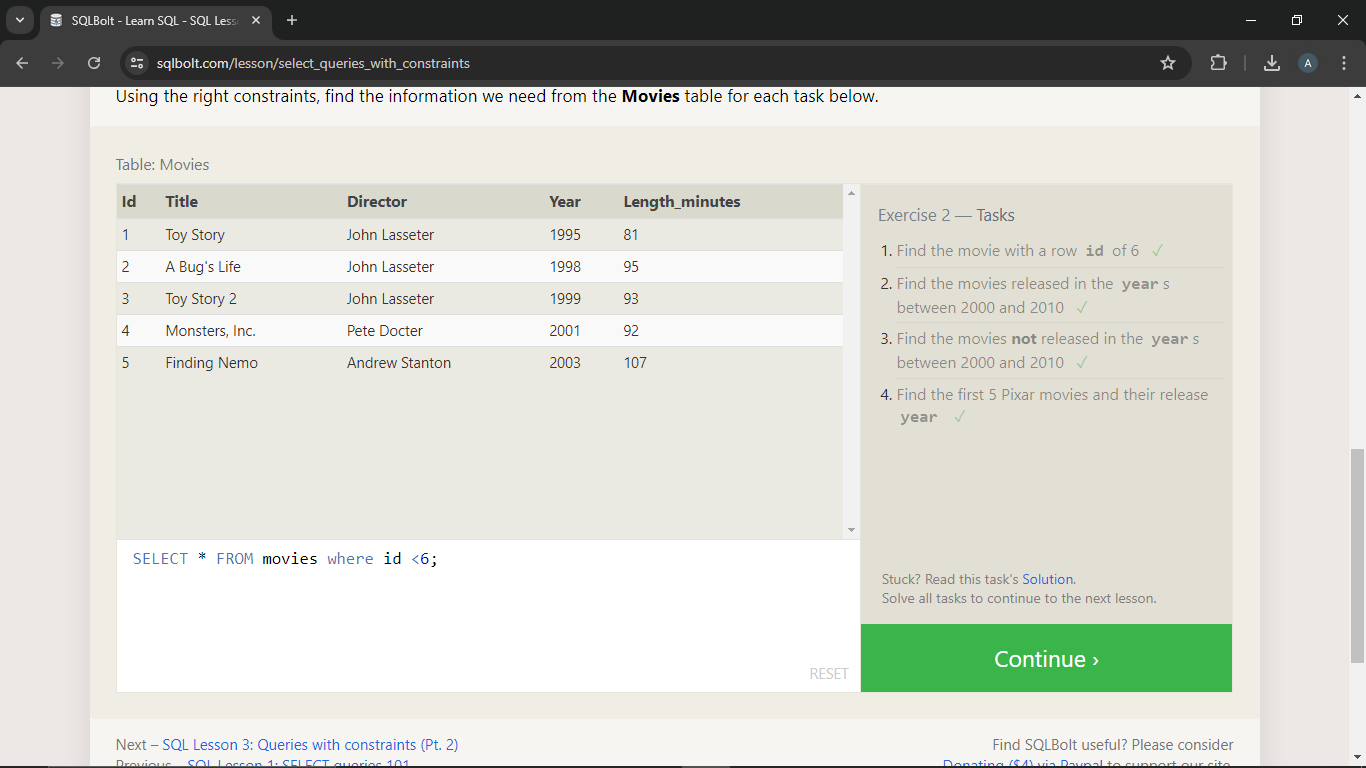
1. Find the **title** and **year** of each film

Query : select title,year from movies;

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

Query : select \* from movies;

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

****

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

Query : SELECT \* FROM movies where id =6;

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

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

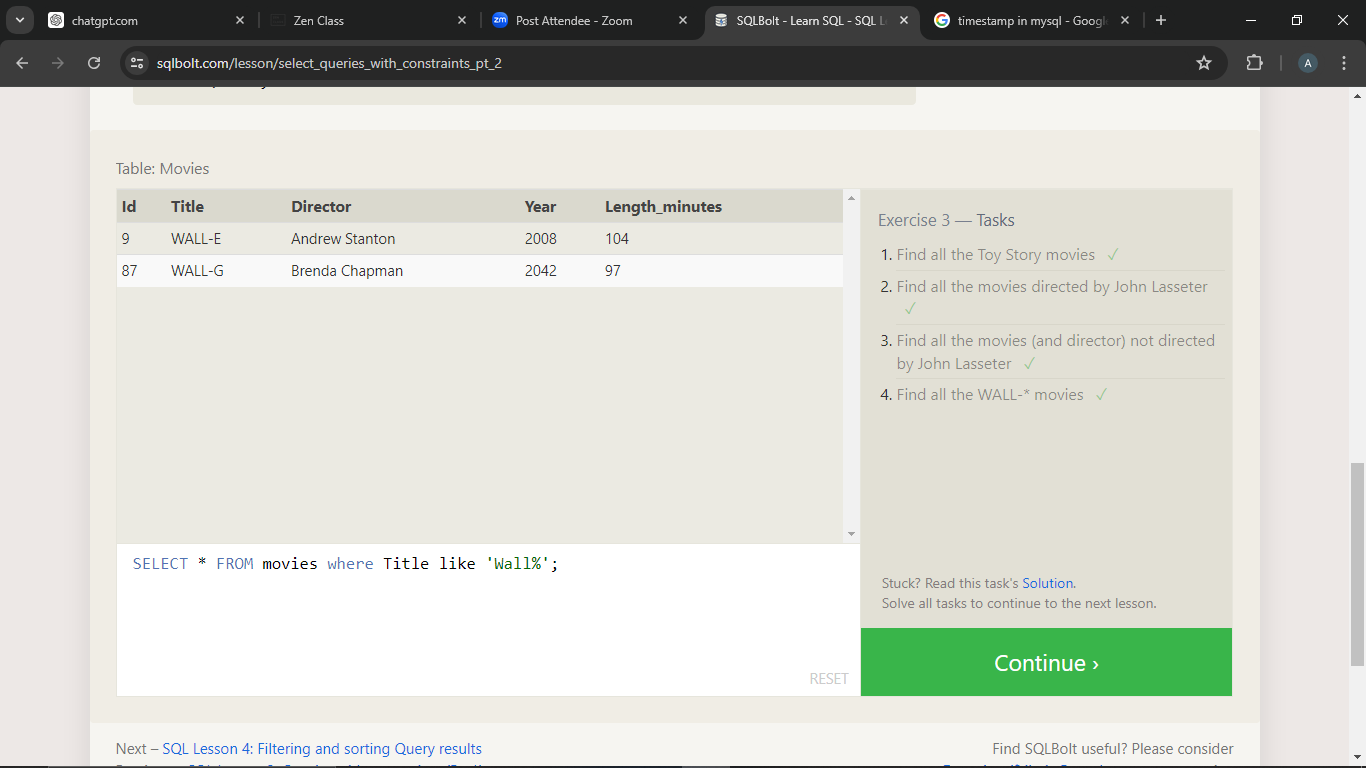
1. Find the movies **not** released in the **year**s between 2000 and 2010

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

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

**Query :** **SELECT \* FROM movies where id <6;**

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

****

1. Find all the Toy Story movies

Query : SELECT \* FROM movies where Title like 'Toy Story%';

1. Find all the movies directed by John Lasseter

Query : SELECT \* FROM movies where Director like 'John Lasseter';

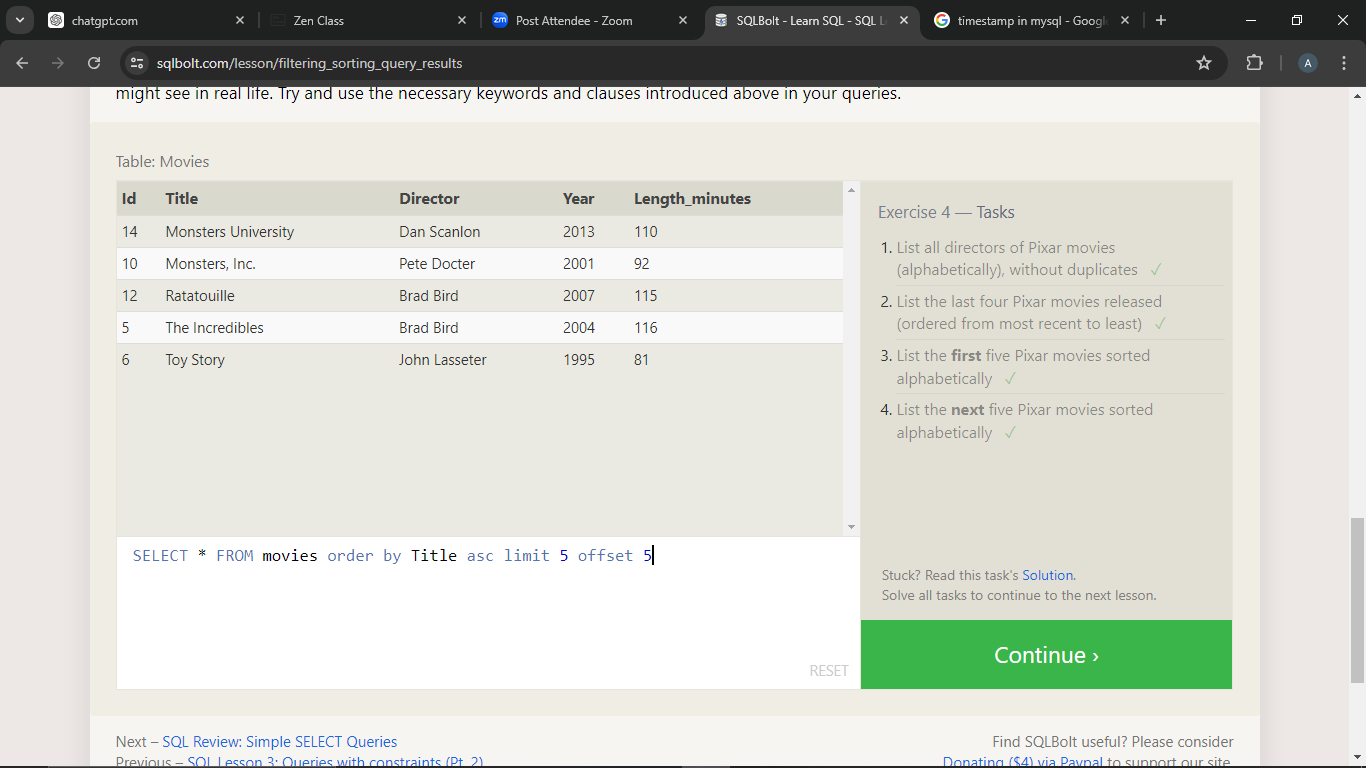
1. Find all the movies (and director) not directed by John Lasseter

Query : SELECT Title,Director FROM movies where not Director like 'John Lasseter';

1. Find all the WALL-\* movies

Query : 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

Query : SELECT Distinct Director FROM movies order by Director asc

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

Query : SELECT \* FROM movies where year > 2009 order by year desc

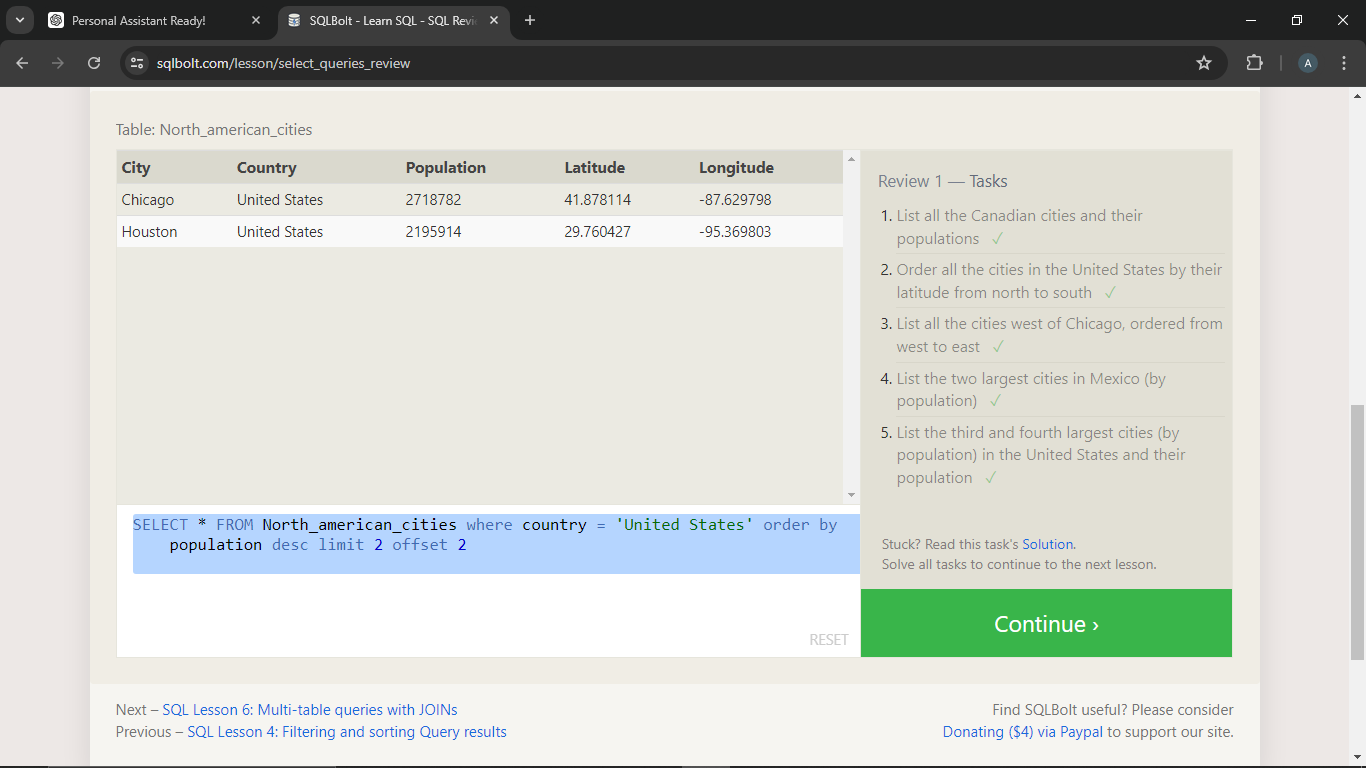
1. List the **first** five Pixar movies sorted alphabetically

Query : SELECT \* FROM movies order by Title asc limit 5

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

Query : 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

Query : 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

Query : SELECT \* 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

Query : SELECT \* FROM North\_american\_cities WHERE Longitude < -87.629798 ORDER BY Longitude ASC;

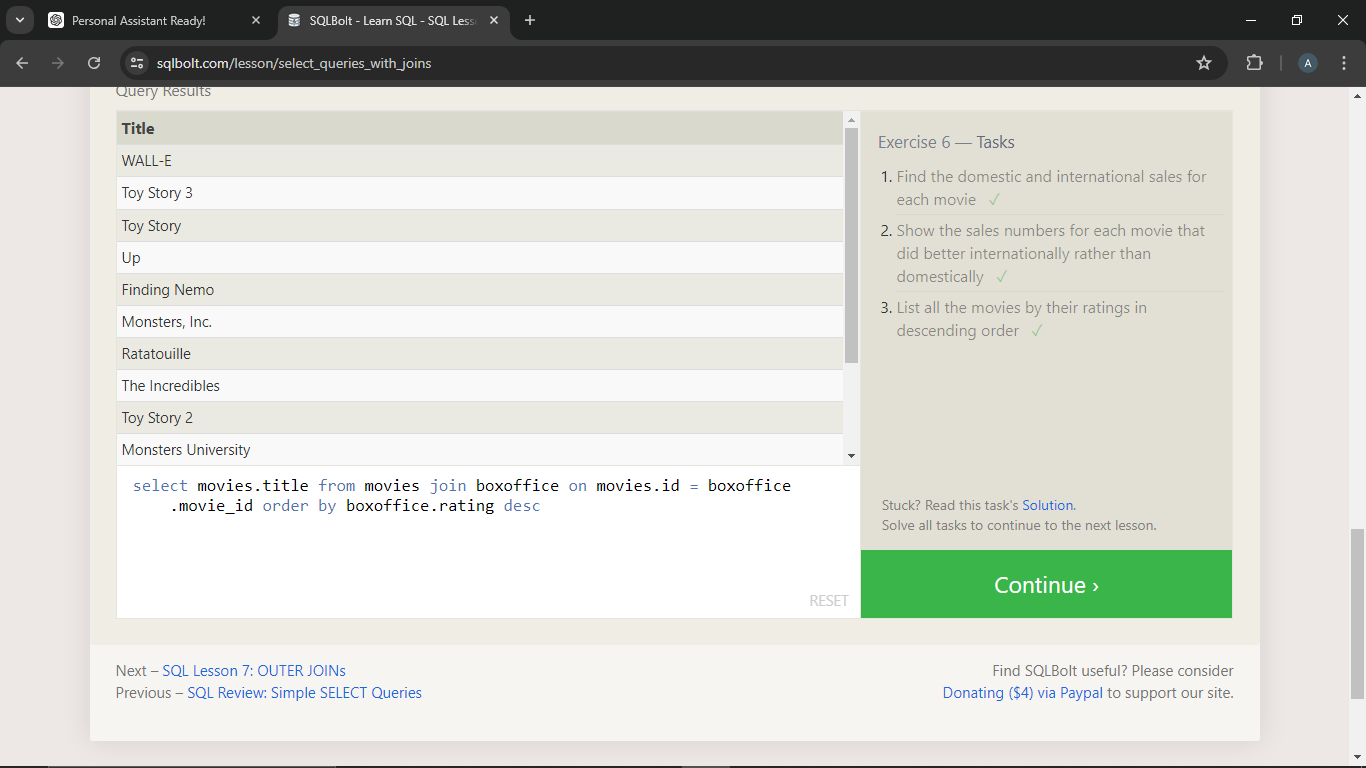
1. List the two largest cities in Mexico (by population)

Query : 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

Query : 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

Query : SELECT Domestic\_sales,international\_sales,title FROM movies join boxoffice on movies.id = boxoffice.movie\_id

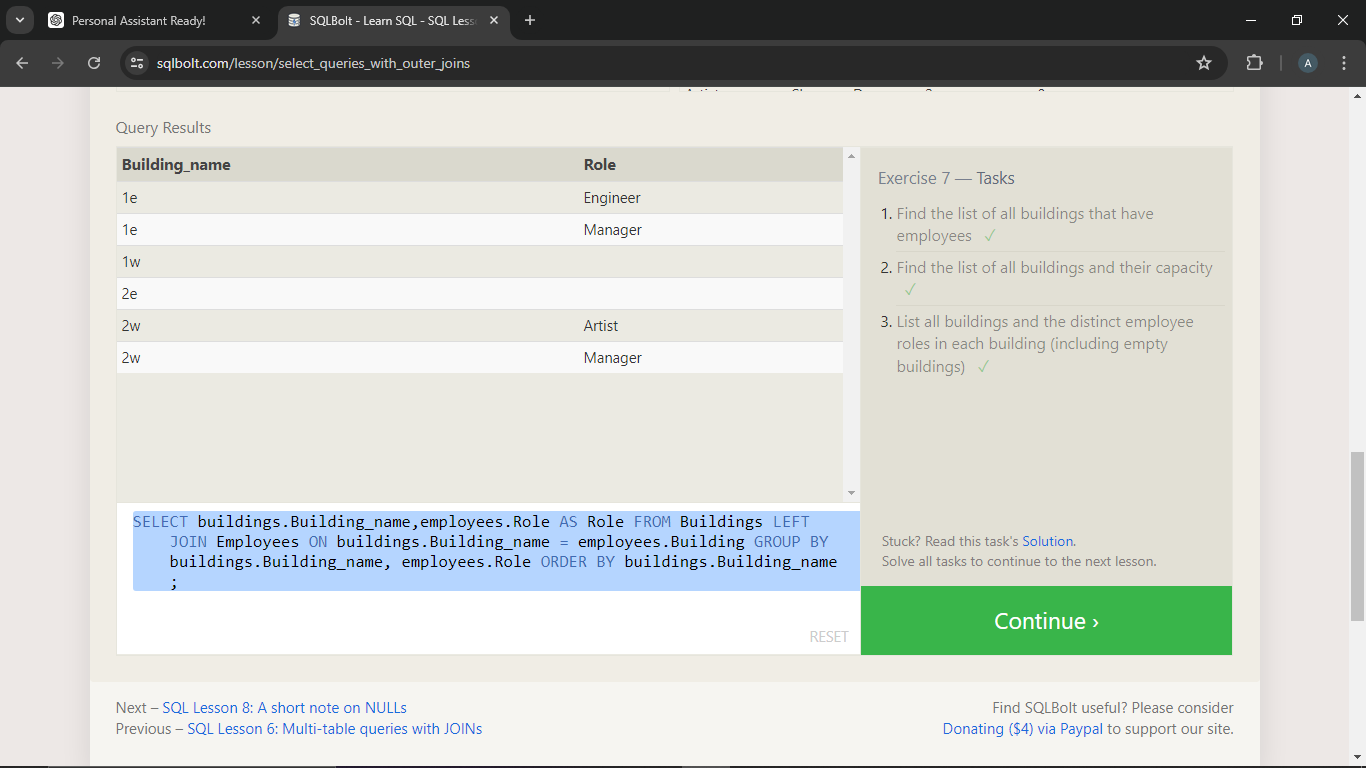
1. Show the sales numbers for each movie that did better internationally rather than domestically

Query : SELECT Domestic\_sales,international\_sales,title FROM movies join boxoffice on movies.id = boxoffice.movie\_id where boxoffice.international\_sales > boxoffice.domestic\_sales

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

Query : select movies.title from movies join boxoffice on movies.id = boxoffice.movie\_id order by boxoffice.rating desc

**SQL Lesson 7: OUTER JOINs :**



1. Find the list of all buildings that have employees

Query : select distinct building from employees

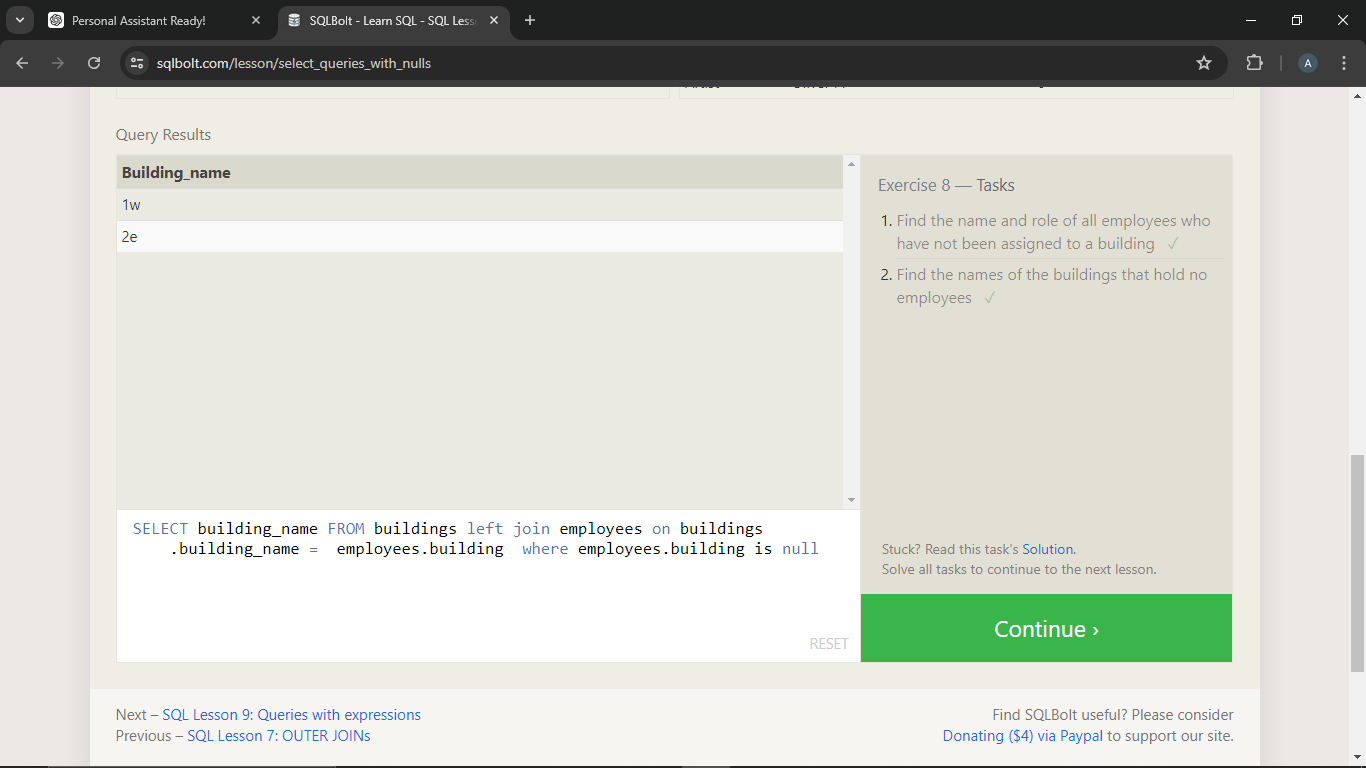
1. Find the list of all buildings and their capacity

Query : select distinct building\_name,capacity from buildings

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

Query : SELECT buildings.Building\_name,employees.Role AS Role FROM Buildings LEFT JOIN Employees ON buildings.Building\_name = employees.Building GROUP BY buildings.Building\_name, employees.Role ORDER BY buildings.Building\_name;

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



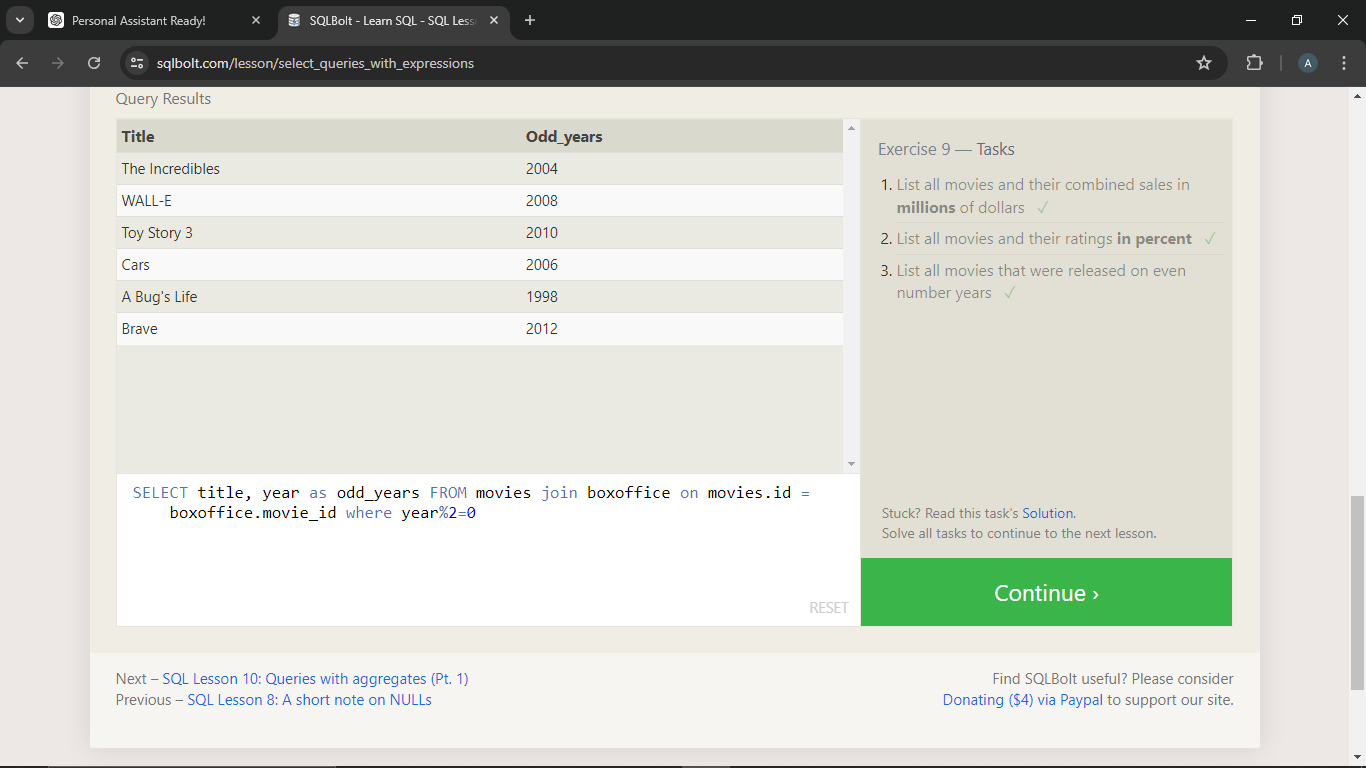
1. Find the name and role of all employees who have not been assigned to a building

Query : SELECT name,role FROM employees where building is null;

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

Query : SELECT building\_name FROM buildings left join employees on buildings.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

Query : SELECT title,(domestic\_sales + international\_sales)/1000000 as compines\_sales FROM movies join boxoffice on movies.id = boxoffice.movie\_id

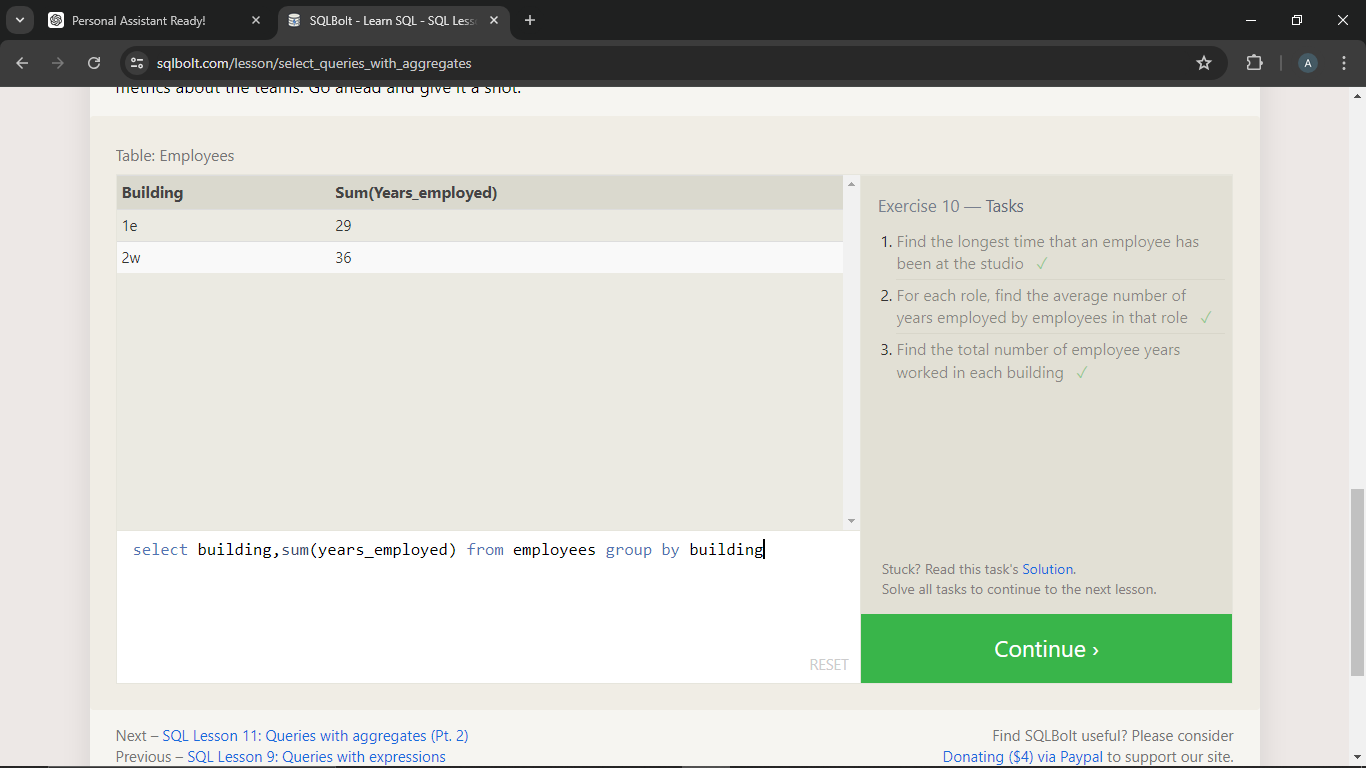
1. List all movies and their ratings **in percent**

**Query :** **SELECT title,(rating)\*10 as percent FROM movies join boxoffice on movies.id = boxoffice.movie\_id**

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

Query :SELECT title, year as odd\_years 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

Query : SELECT \*,max(years\_employed) FROM employees

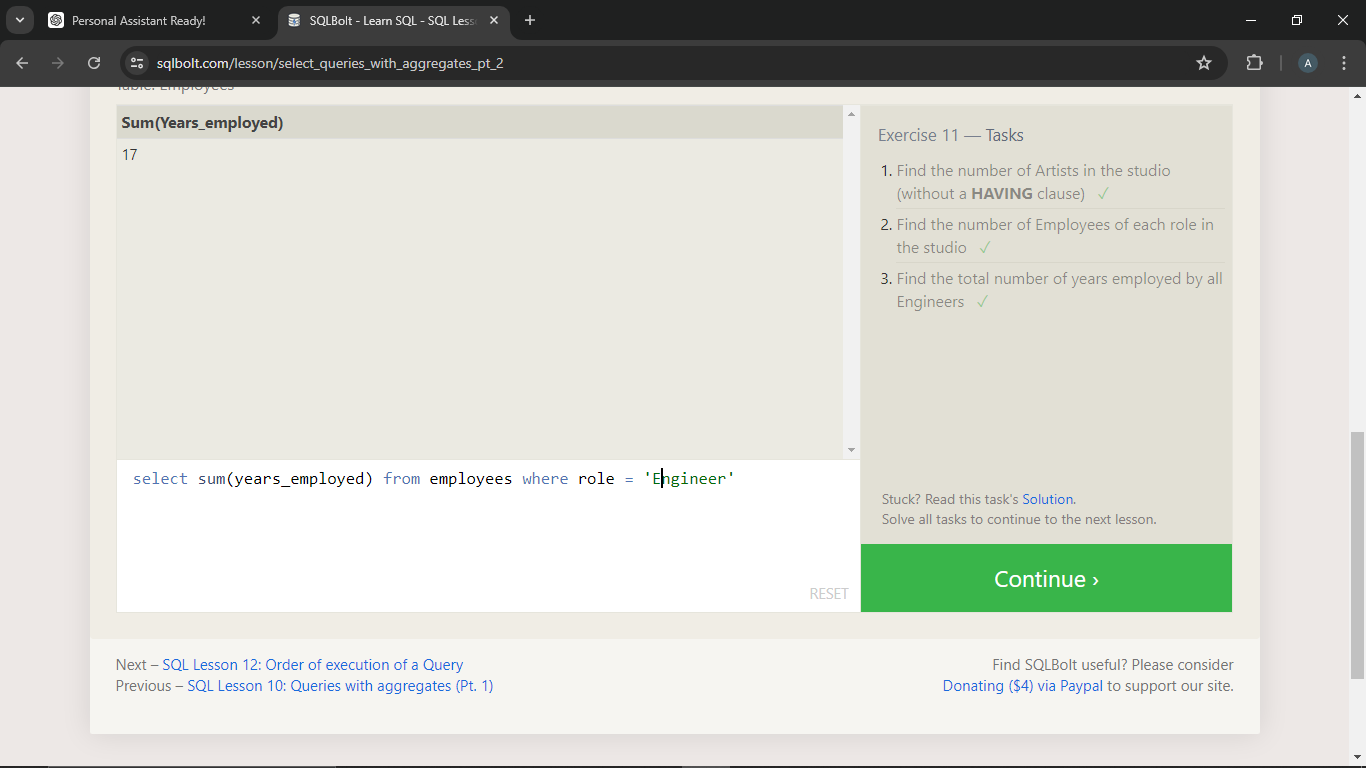
1. For each role, find the average number of years employed by employees in that role

Query : SELECT role,avg(years\_employed) FROM employees group by role

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

Query : 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)

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

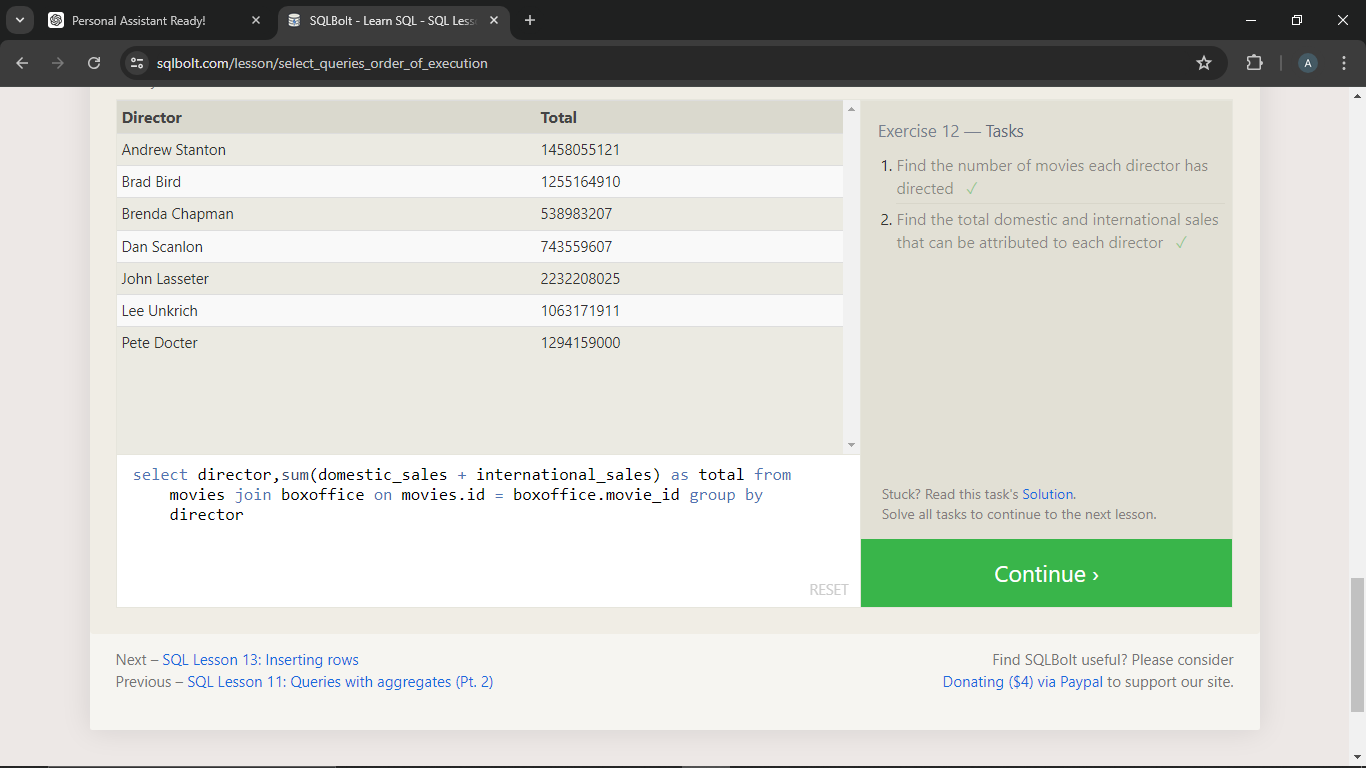
1. Find the number of Employees of each role in the studio

Query : select count(name),role from employees group by role

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

Query : select sum(years\_employed) from employees where role = 'Engineer'

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

****

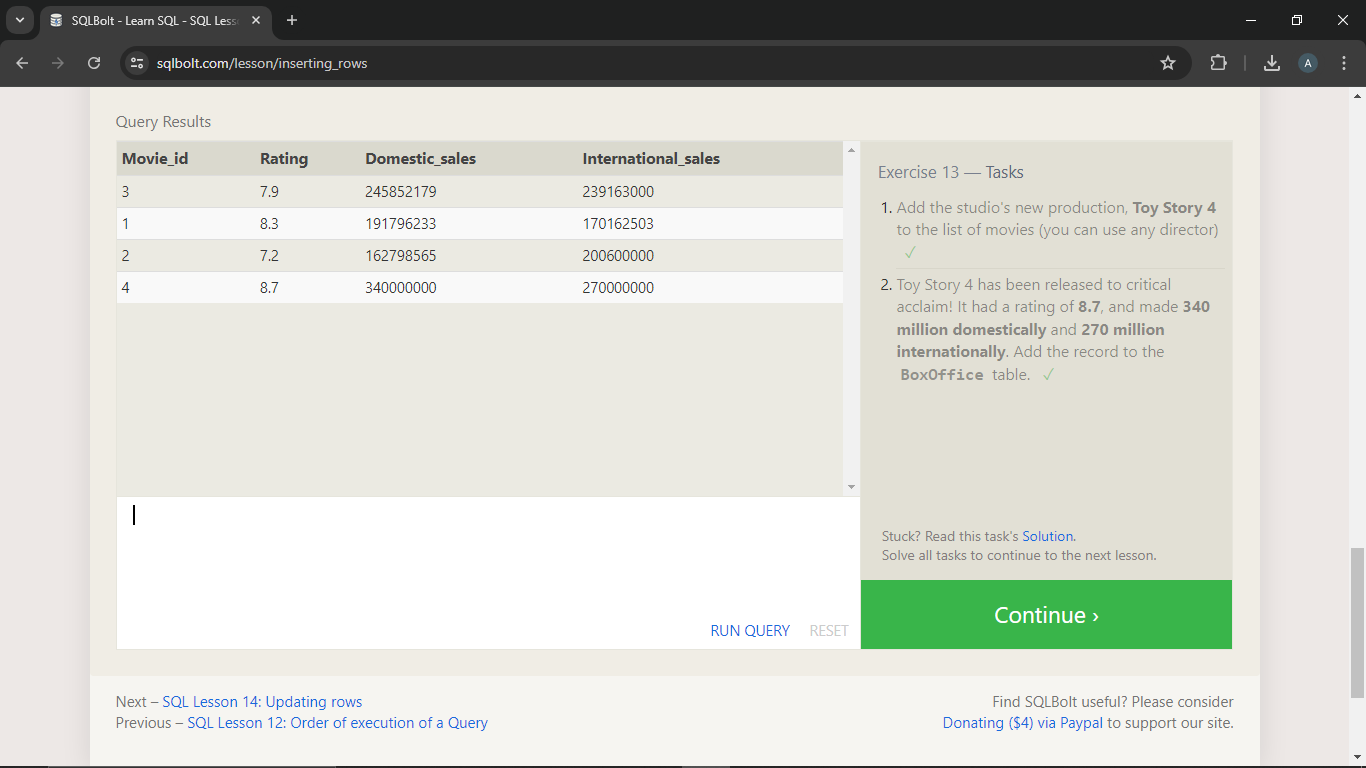
1. Find the number of movies each director has directed

Query : select director,count(title) from movies group by director

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

Query : select director,sum(domestic\_sales + international\_sales) as total from movies join boxoffice on movies.id = boxoffice.movie\_id group by director

SQL Lesson 13: Inserting rows :



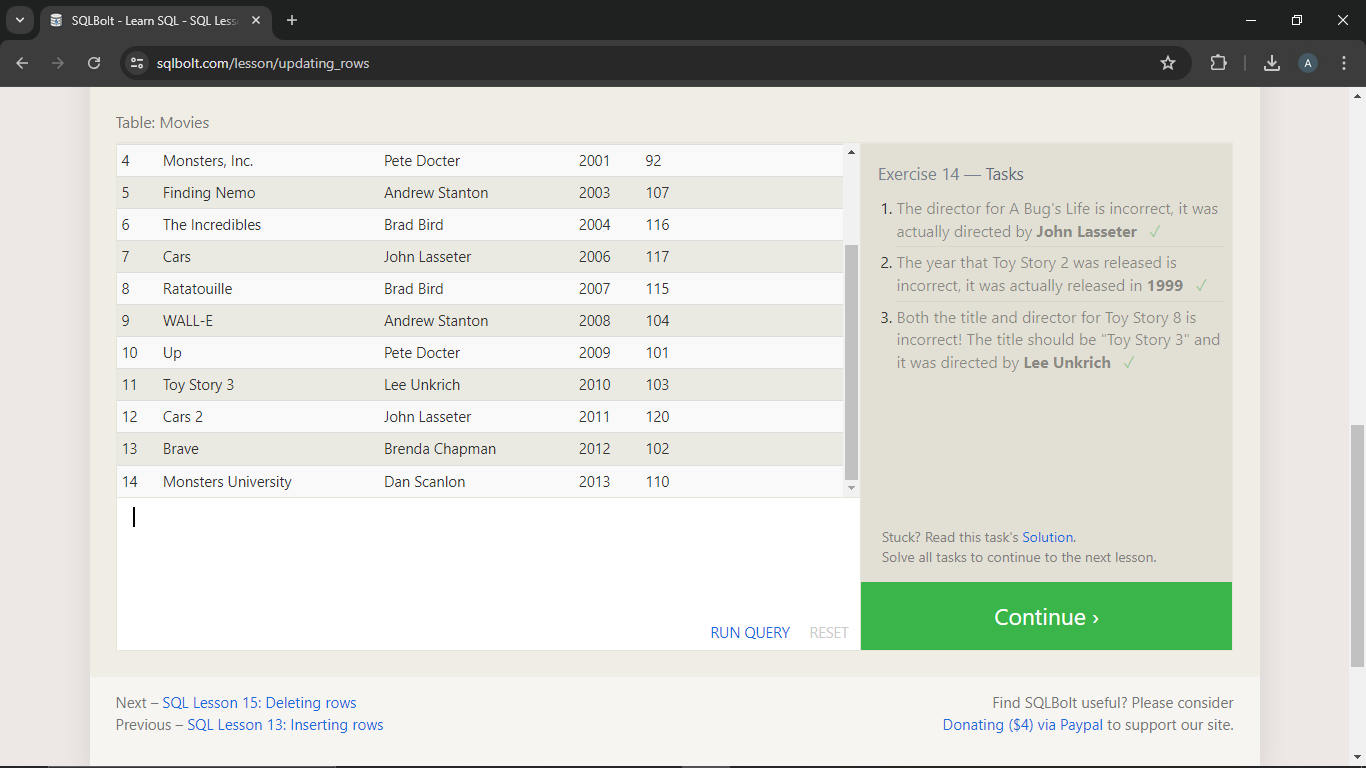
1. Add the studio's new production, **Toy Story 4** to the list of movies (you can use any director)

Query : insert into movies values(4,'Toy Story 4','Abu Iron',2021,98);

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.

Query : insert into Boxoffice values (4,8.7,340000000,270000000)

SQL Lesson 14: Updating rows :

****

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

Query : UPDATE movies SET director = 'John Lasseter' WHERE id = 2;

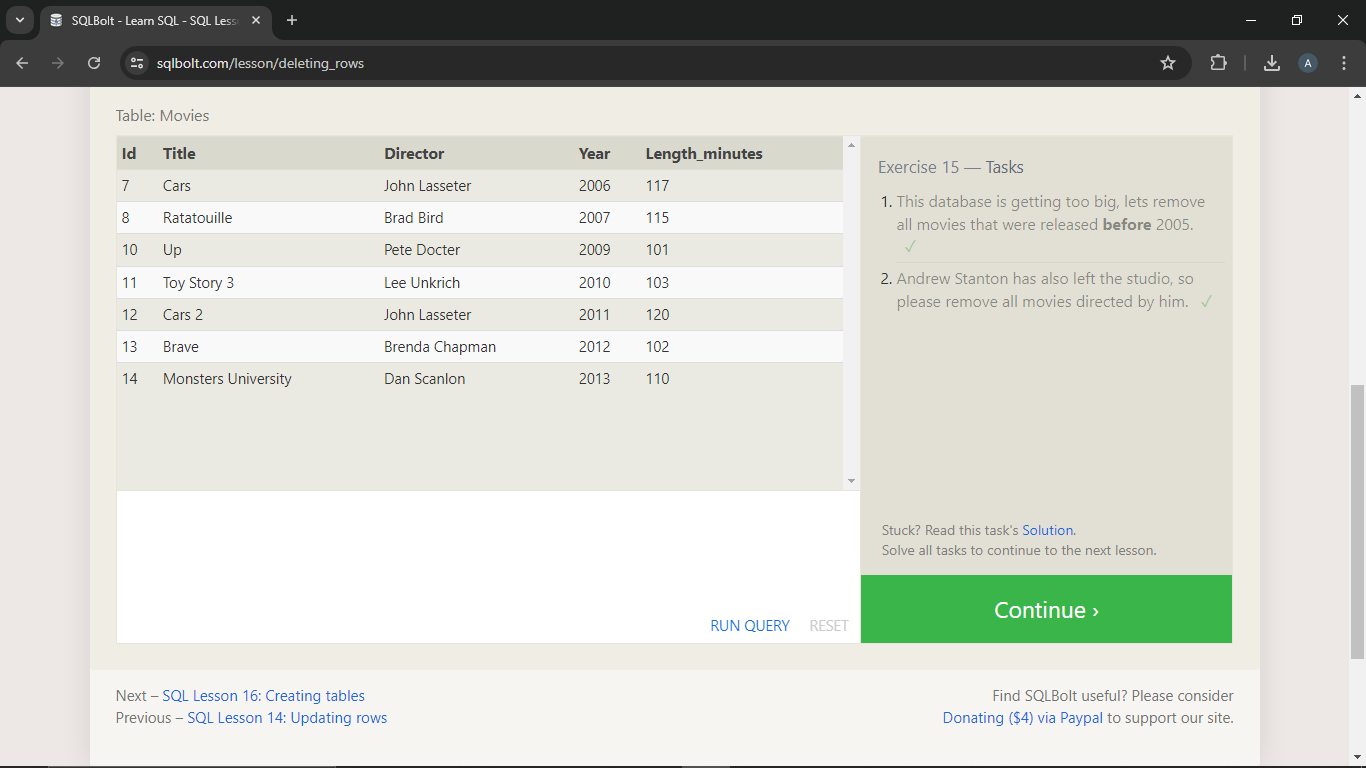
1. The year that Toy Story 2 was released is incorrect, it was actually released in **1999**

Query : update movies set year = 1999 where id = 3;

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**

Query : update movies set title = 'Toy Story 3', director = 'Lee Unkrich' where title = 'Toy Story 8';

**SQL Lesson 15: Deleting rows :**

****

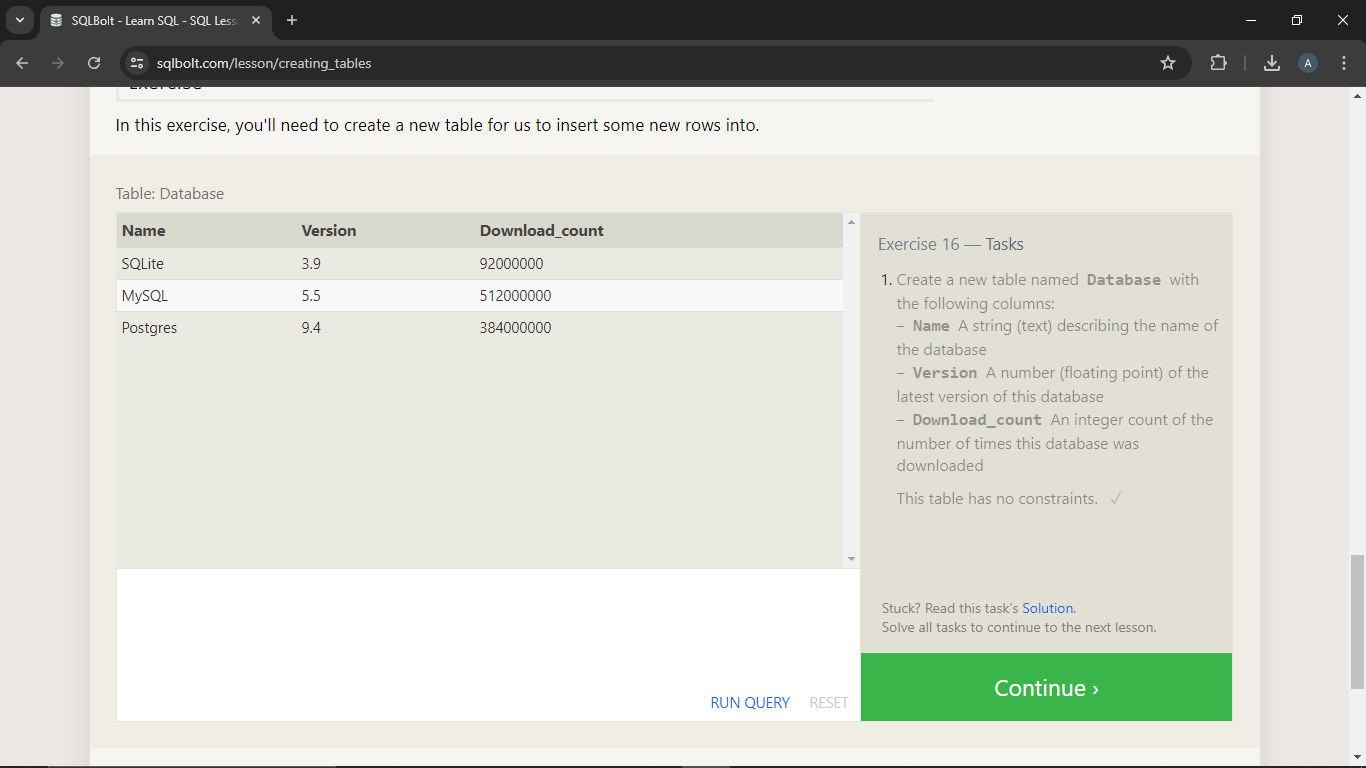
1. This database is getting too big, lets remove all movies that were released **before** 2005.

Query : delete from movies where year < 2005;

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

Query : delete from movies where director ='Andrew Stanton';

**SQL Lesson 16: Creating tables :**

****

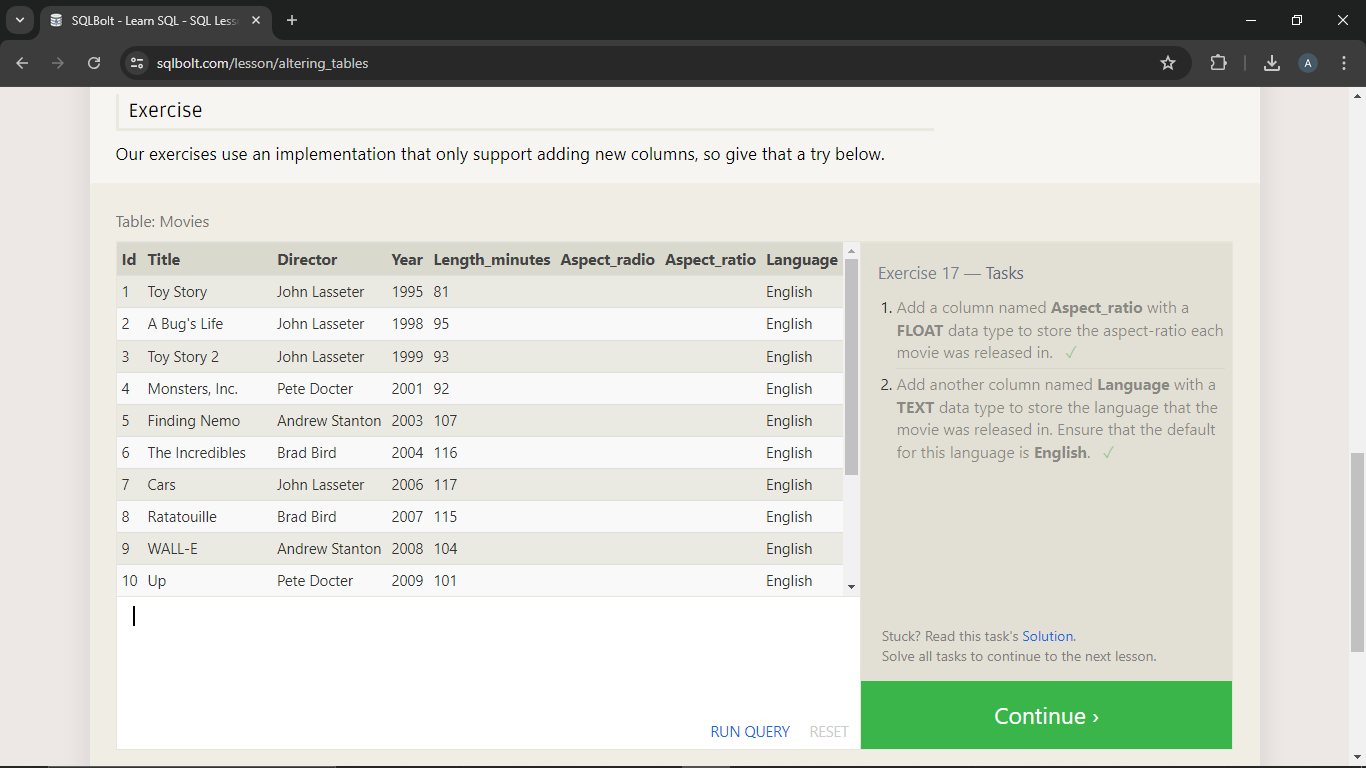
1. Create a new table named **Database** with the following columns:

– **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.

Query : create table database(name varchar(255), version float, download\_count int);

**SQL Lesson 17: Altering tables**

****

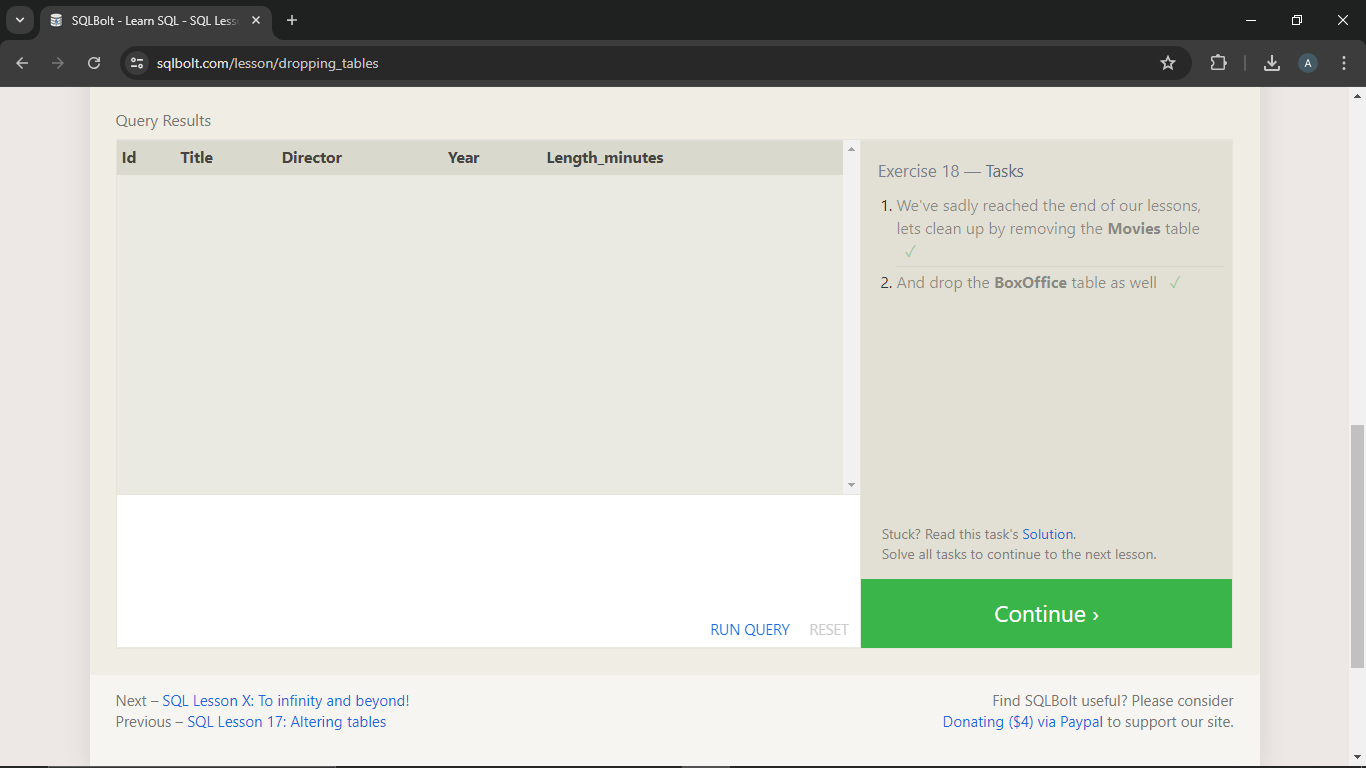
1. Add a column named **Aspect\_ratio** with a **FLOAT** data type to store the aspect-ratio each movie was released in.

Query : 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**.

Query : alter table movies add language varchar(255) 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

Query : drop table movies;

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

Query : drop table boxoffice;