**MY-SQL TASKS**

**SQL Lesson 1: SELECT queries 101**

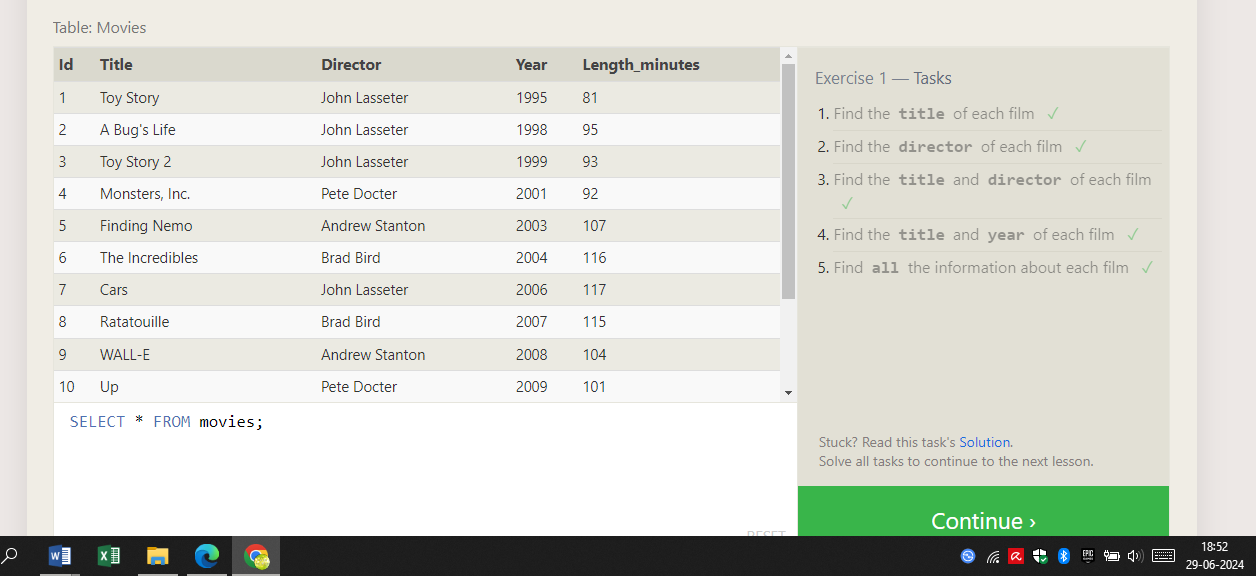
**Questions:**

1. Find the title of each film
2. Find the director of each film
3. Find the title and director of each film
4. Find the title and year of each film
5. Find all the information about each film

**Answers:**

1. SELECT title FROM movies;
2. SELECT director FROM movies;
3. SELECT title,director FROM movies;
4. SELECT title,year FROM movies;
5. SELECT \* FROM movies;

**Screenshot:**



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

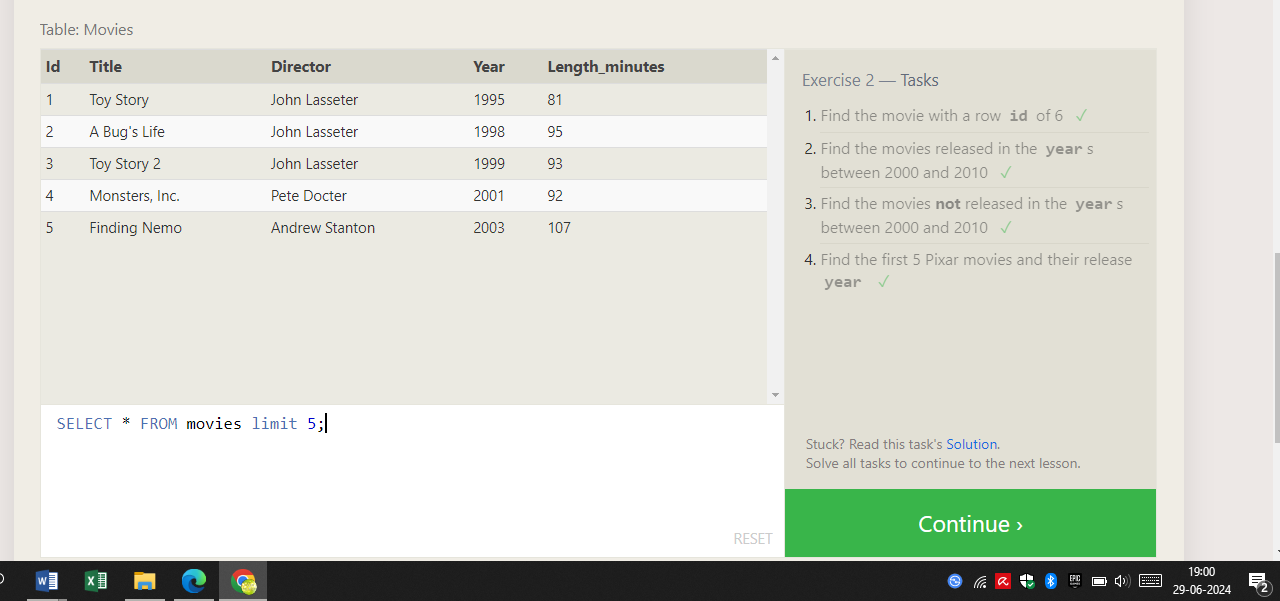
**Questions:**

1. Find the movie with a row id of 6
2. Find the movies released in the years between 2000 and 2010
3. Find the movies not released in the years between 2000 and 2010
4. Find the first 5 Pixar movies and their release year

**Answers:**

1. SELECT \* FROM movies where id=6;
2. SELECT \* FROM movies where year between 2000 and 2010;
3. SELECT \* FROM movies where year not between 2000 and 2010;
4. SELECT \* FROM movies limit 5;

**Screenshots:**

****

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

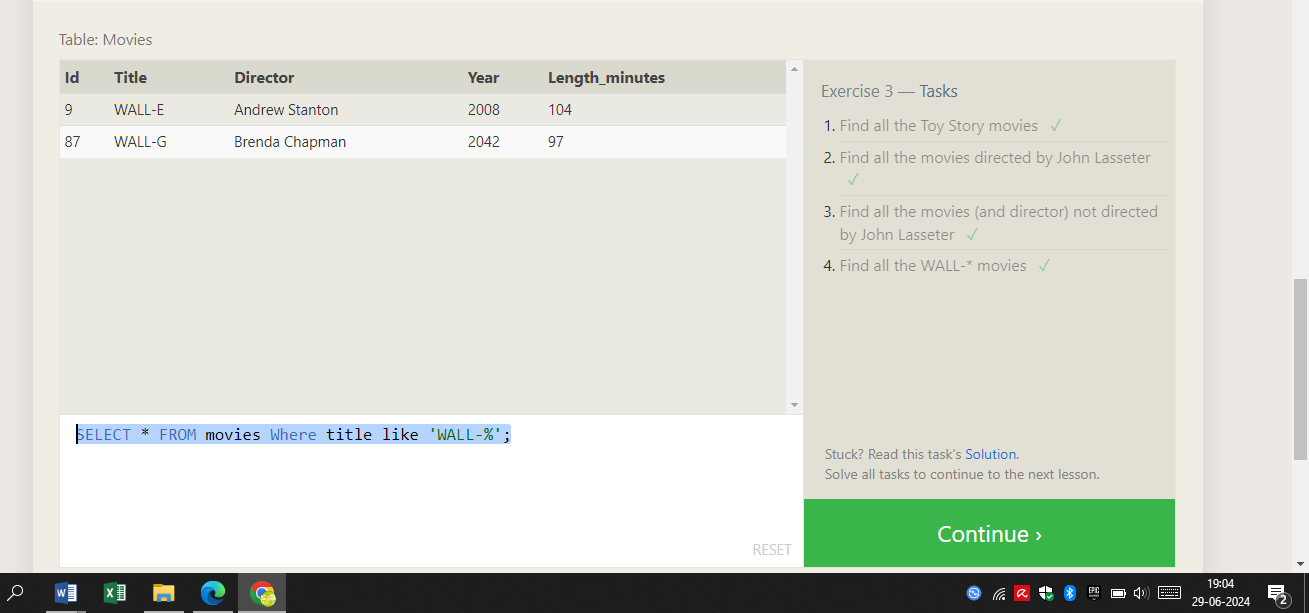
**Questions:**

1. Find all the Toy Story movies
2. Find all the movies directed by John Lasseter
3. Find all the movies (and director) not directed by John Lasseter
4. Find all the WALL-\* movies

**Answers:**

1. SELECT \* FROM movies Where Title like 'Toy%';
2. SELECT \* FROM movies Where director='John Lasseter';
3. SELECT \* FROM movies Where director<>'John Lasseter';
4. SELECT \* FROM movies Where title like 'WALL-%';

**Screenshots:**

****

**SQL Lesson 4: Filtering and sorting Query results**

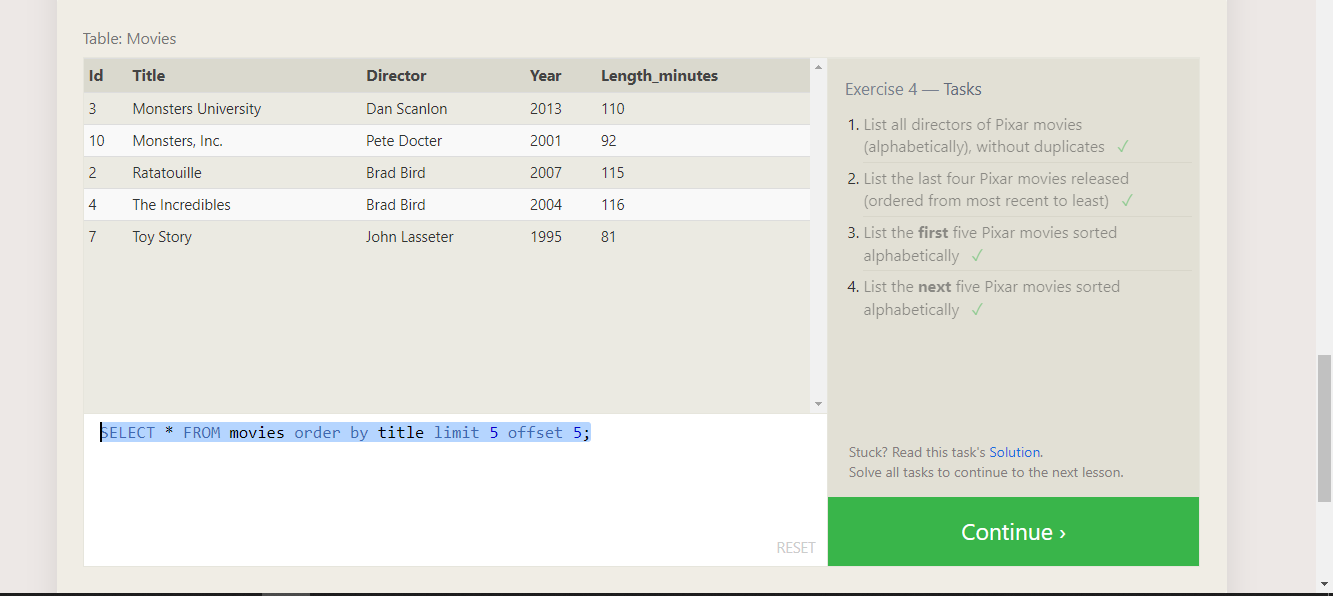
**Questions:**

1. List all directors of Pixar movies (alphabetically), without duplicates
2. List the last four Pixar movies released (ordered from most recent to least)
3. List the **first** five Pixar movies sorted alphabetically
4. List the **next** five Pixar movies sorted alphabetically

**Answers:**

1. SELECT distinct director FROM movies order by director;
2. SELECT \* FROM movies order by year desc limit 4;
3. SELECT \* FROM movies order by title limit 5;
4. SELECT \* FROM movies order by title limit 5 offset 5;

Screenshots:



**SQL Review: Simple SELECT Queries**

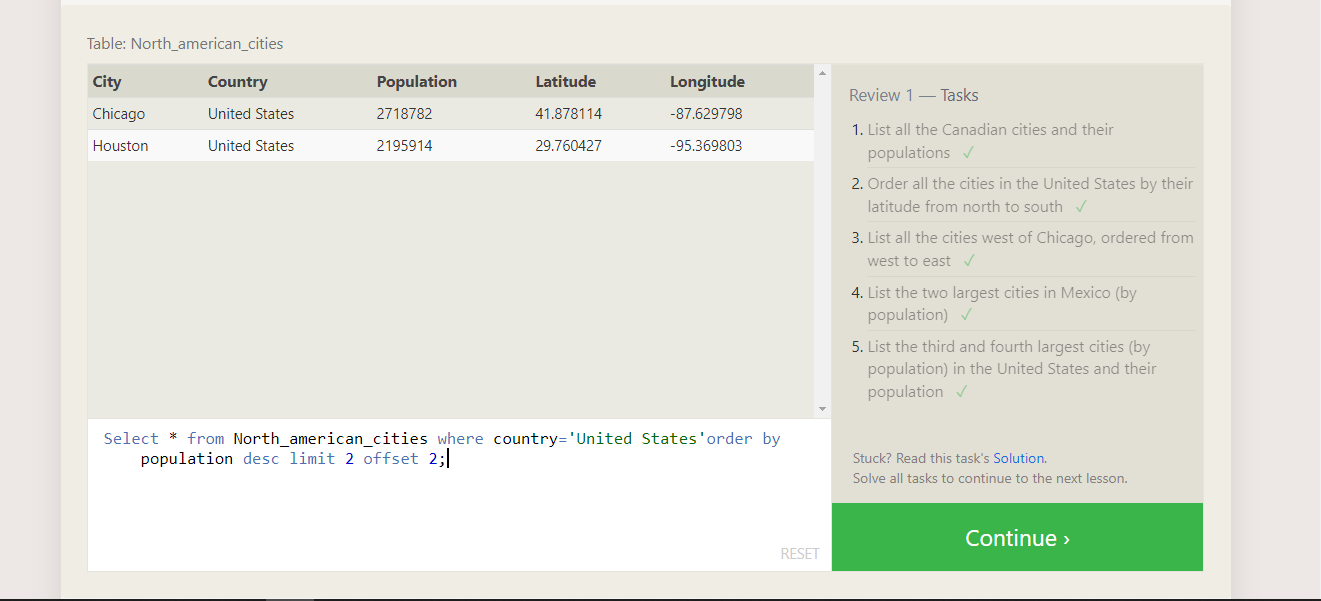
**Questions:**

1. List all the Canadian cities and their populations
2. Order all the cities in the United States by their latitude from north to south
3. List all the cities west of Chicago, ordered from west to east
4. List the two largest cities in Mexico (by population)
5. List the third and fourth largest cities (by population) in the United States and their population

**Answers:**

1. **SELECT \* FROM north\_american\_cities where Country='Canada';**
2. Select \* from North\_american\_cities where country='United States'order by latitude desc;
3. Select \* from North\_american\_cities where longitude <-87.629798 order by longitude;
4. SELECT city, population FROM north\_american\_cities WHERE country LIKE "Mexico" ORDER BY population DESC LIMIT 2;
5. Select \* from North\_american\_cities where country='United States'order by population desc limit 2 offset 2;

**Screenshots:**

****

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

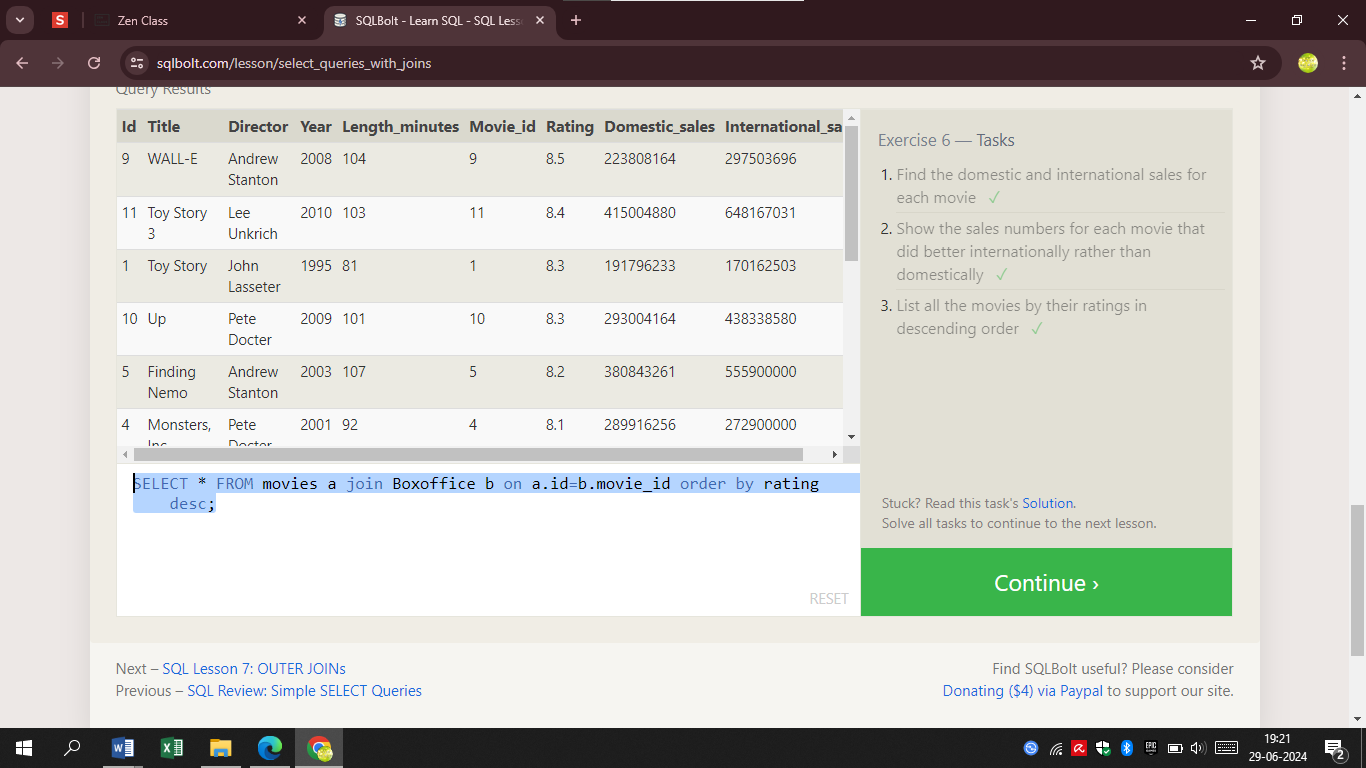
**Questions:**

1. Find the domestic and international sales for each movie
2. Show the sales numbers for each movie that did better internationally rather than domestically
3. List all the movies by their ratings in descending order

Answers:

1. SELECT \* FROM movies a join Boxoffice b on a.id=b.movie\_id;
2. SELECT \* FROM movies a join Boxoffice b on a.id=b.movie\_id where international\_sales > Domestic\_sales;
3. SELECT \* FROM movies a join Boxoffice b on a.id=b.movie\_id order by rating desc;

Screenshots:



**SQL Lesson 7: OUTER JOINs**

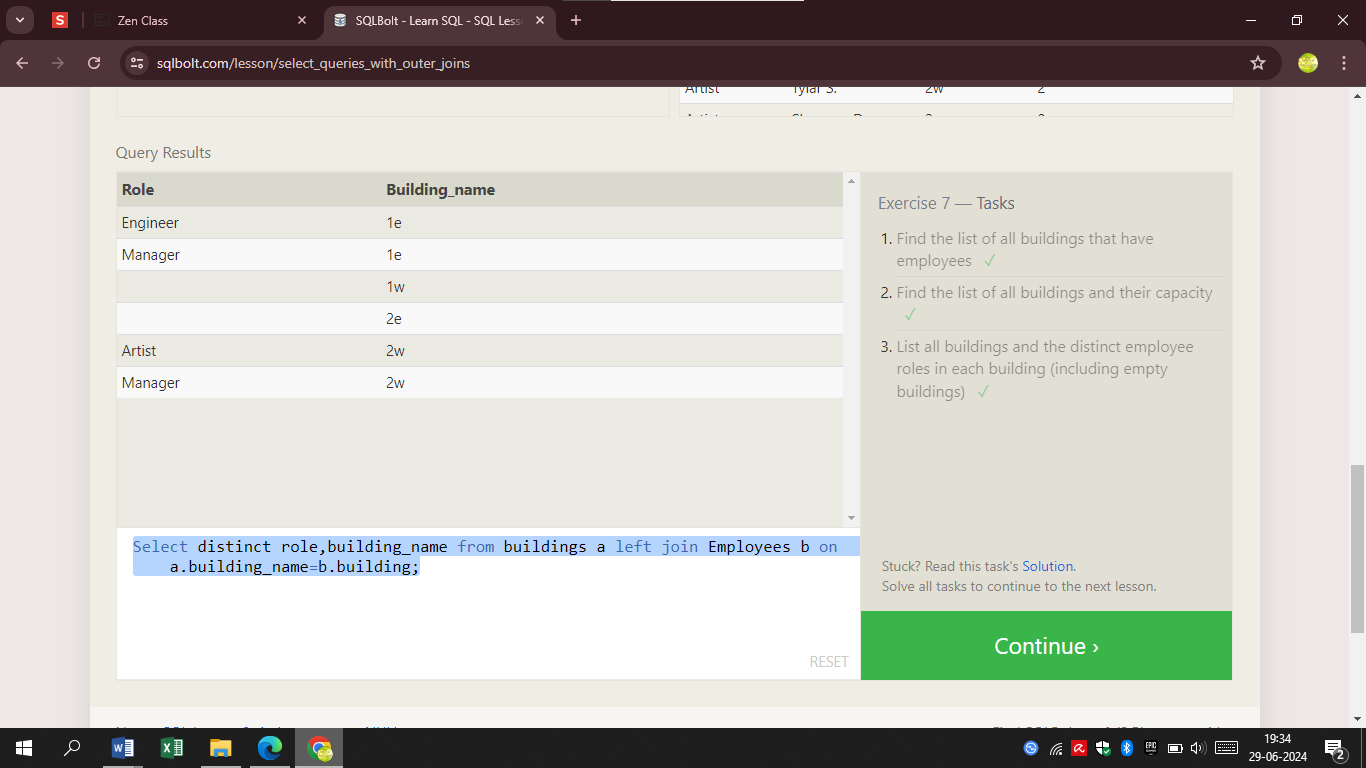
**Questions:**

1. Find the list of all buildings that have employees
2. Find the list of all buildings and their capacity
3. List all buildings and the distinct employee roles in each building (including empty buildings)

**Answers:**

1. SELECT DISTINCT building FROM employees;
2. Select \* from buildings;
3. Select distinct role,building\_name from buildings a left join Employees b on a.building\_name=b.building;

**Screenshots:**



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

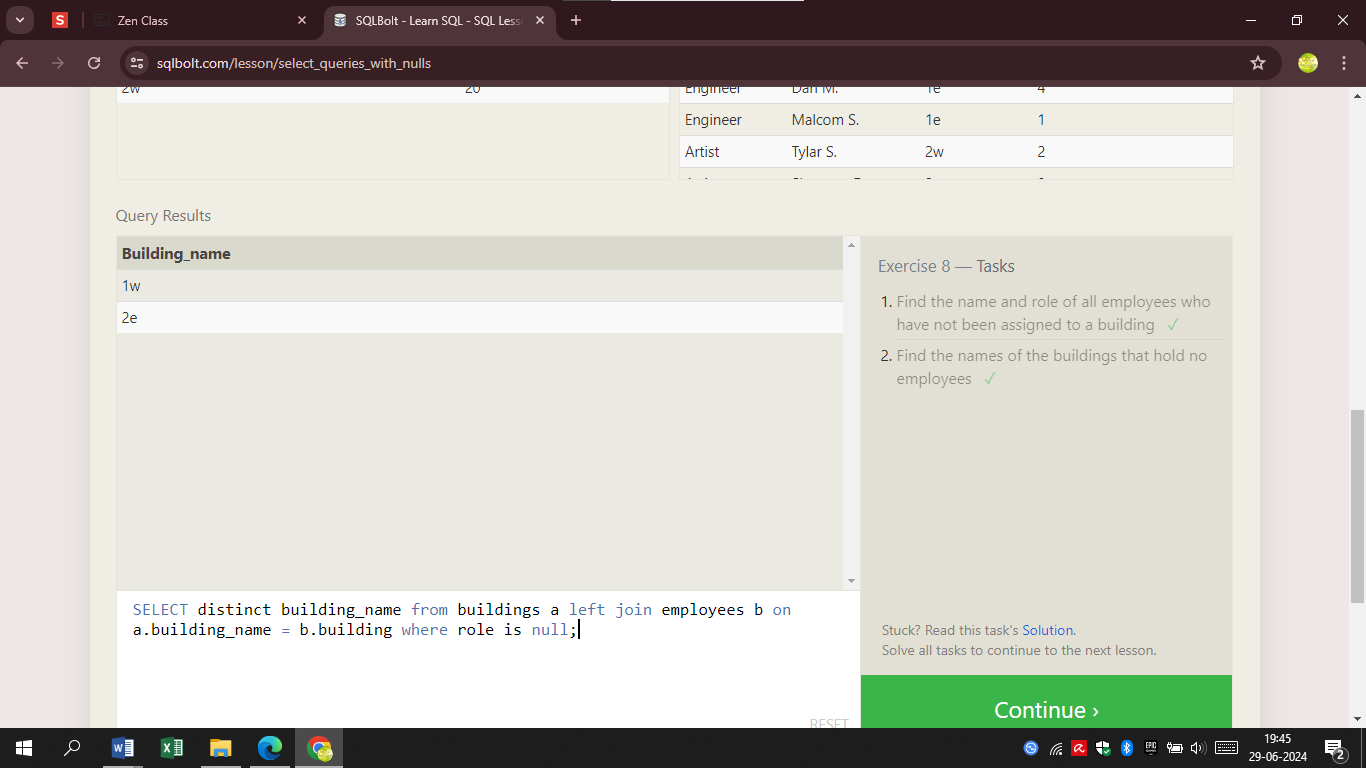
**Questions:**

1. Find the name and role of all employees who have not been assigned to a building
2. Find the names of the buildings that hold no employees

Answers:

1. SELECT \* FROM employees where building is null;
2. SELECT distinct building\_name from buildings a left join employees b on

a.building\_name = b.building where role is null;



**SQL Lesson 9: Queries with expressions**

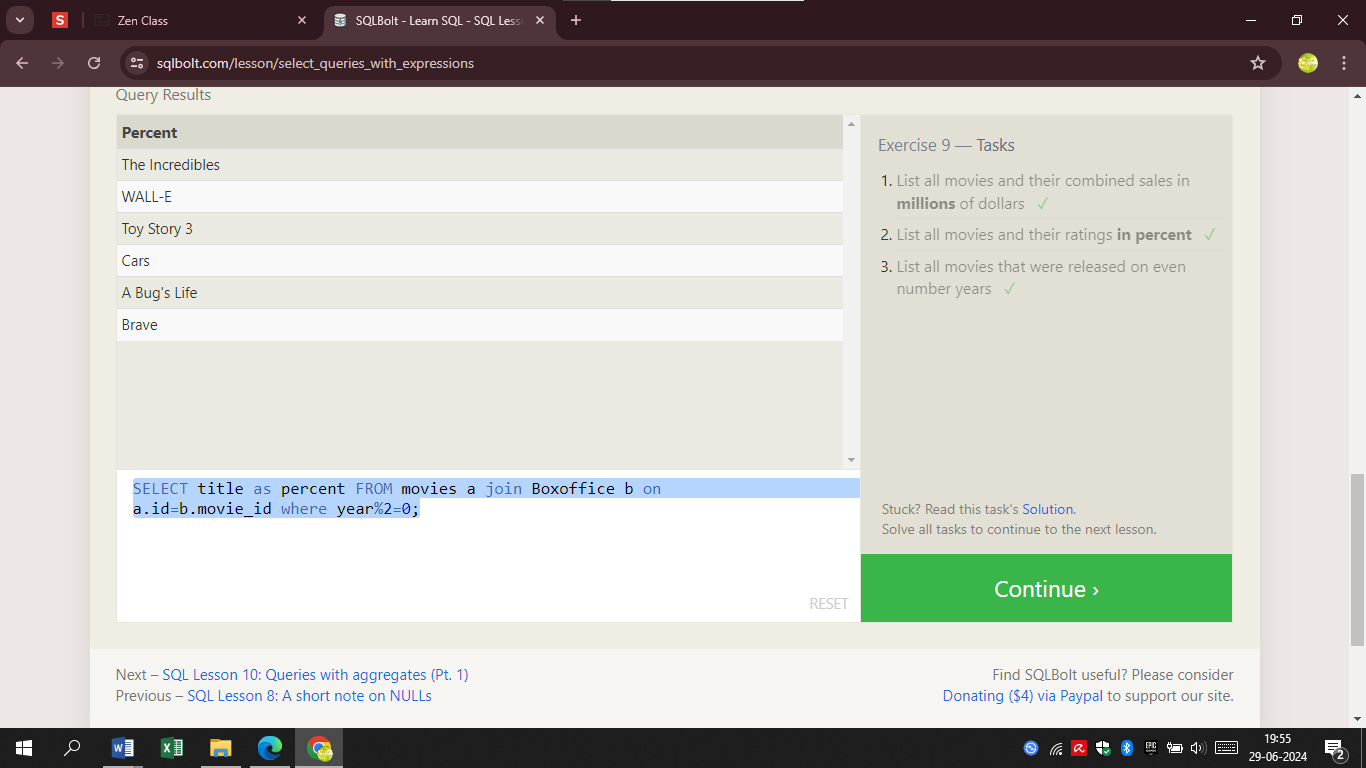
**Questions:**

1. List all movies and their combined sales in **millions** of dollars
2. List all movies and their ratings **in percent**
3. List all movies that were released on even number years

Answers:

1. SELECT title, (Domestic\_sales + International\_sales)/1000000 as tot\_sales FROM movies a join Boxoffice b on a.id=b.movie\_id;
2. SELECT title,rating\*100 as percent FROM movies a join Boxoffice b on a.id=b.movie\_id;
3. SELECT title as percent FROM movies a join Boxoffice b on a.id=b.movie\_id where year%2=0;

Screenshots:



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

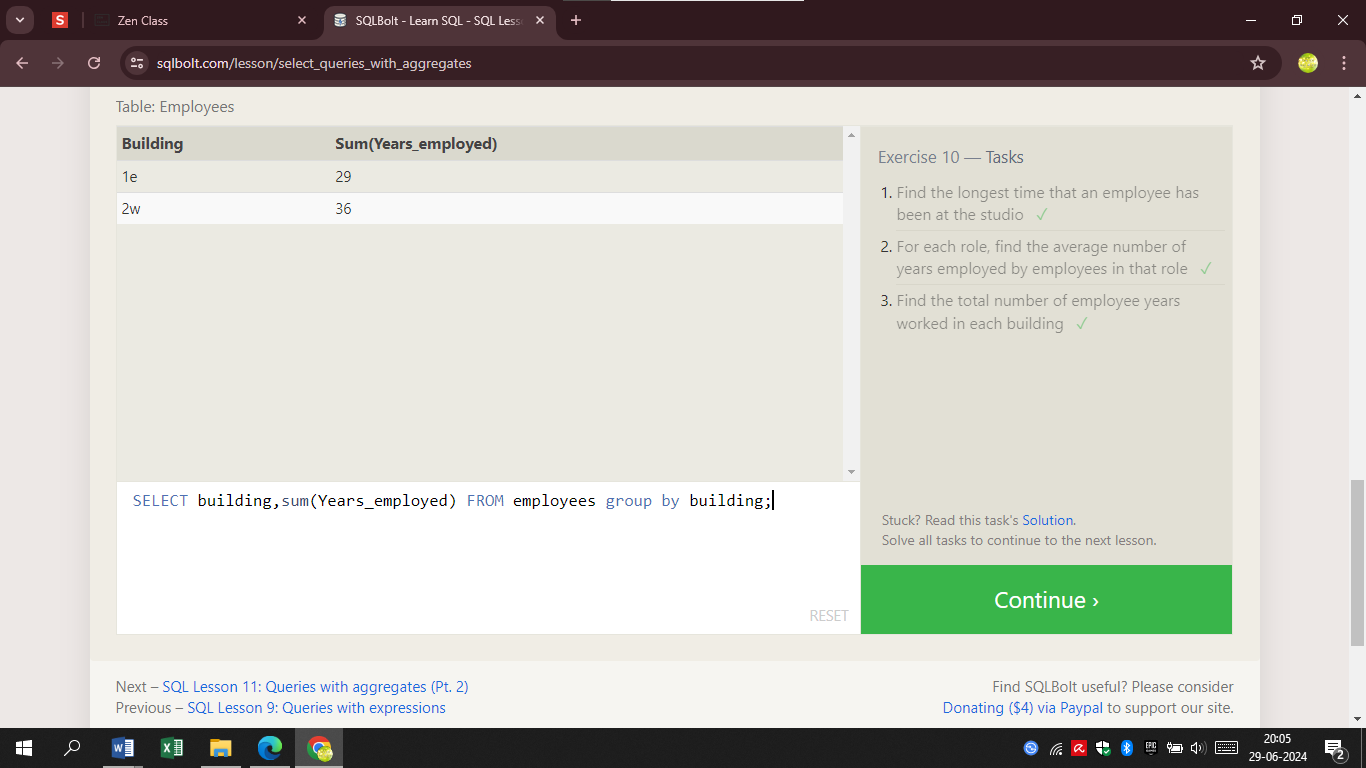
**Questions:**

1. Find the longest time that an employee has been at the studio
2. For each role, find the average number of years employed by employees in that role
3. Find the total number of employee years worked in each building

**Answers:**

1. SELECT max(Years\_employed) FROM employees;
2. SELECT role,avg(Years\_employed) FROM employees group by role;
3. SELECT building,sum(Years\_employed) FROM employees group by building;

**Screenshots:**



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

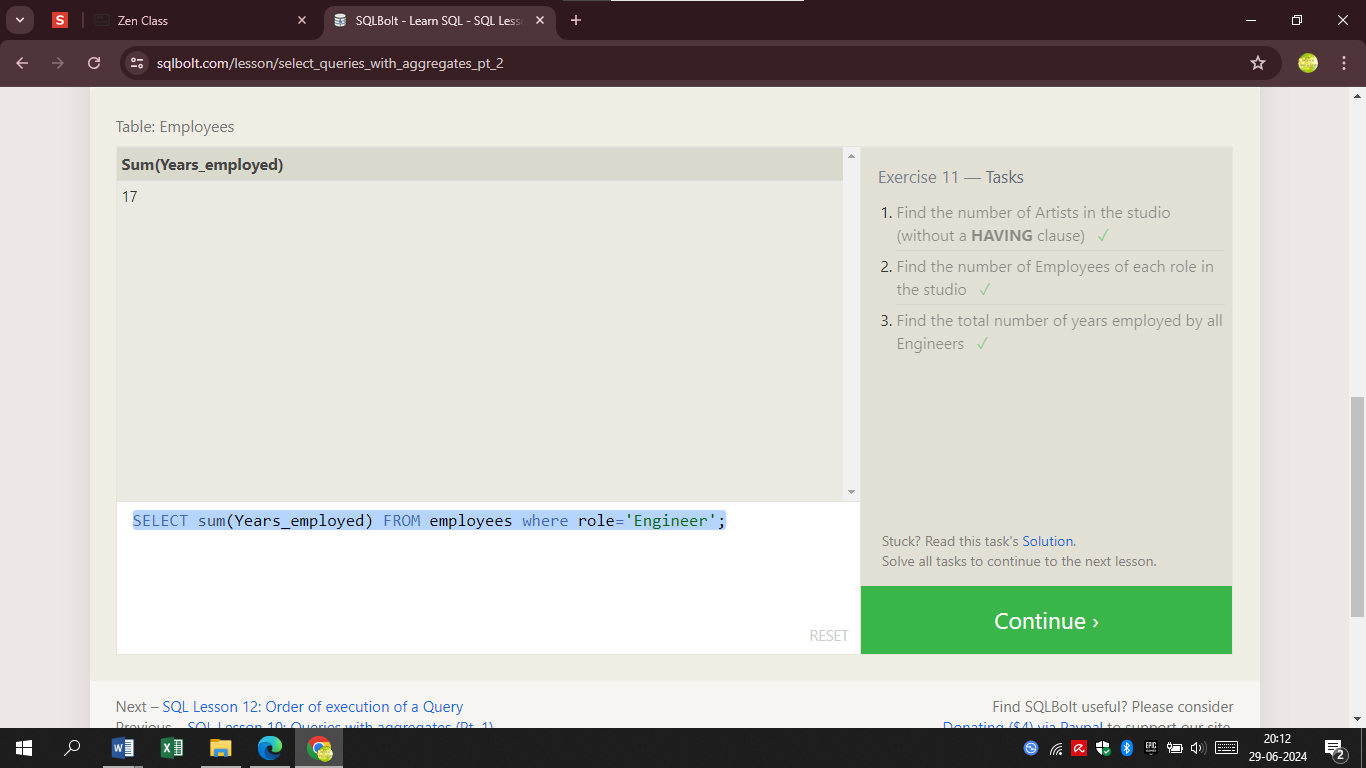
**Questions:**

1. Find the number of Artists in the studio (without a **HAVING** clause)
2. Find the number of Employees of each role in the studio
3. Find the total number of years employed by all Engineers

**Answers:**

1. SELECT count(role) FROM employees where role='Artist';
2. SELECT role,count(role) FROM employees group by role;
3. SELECT sum(Years\_employed) FROM employees where role='Engineer';

**Screenshots:**



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

**Questions:**

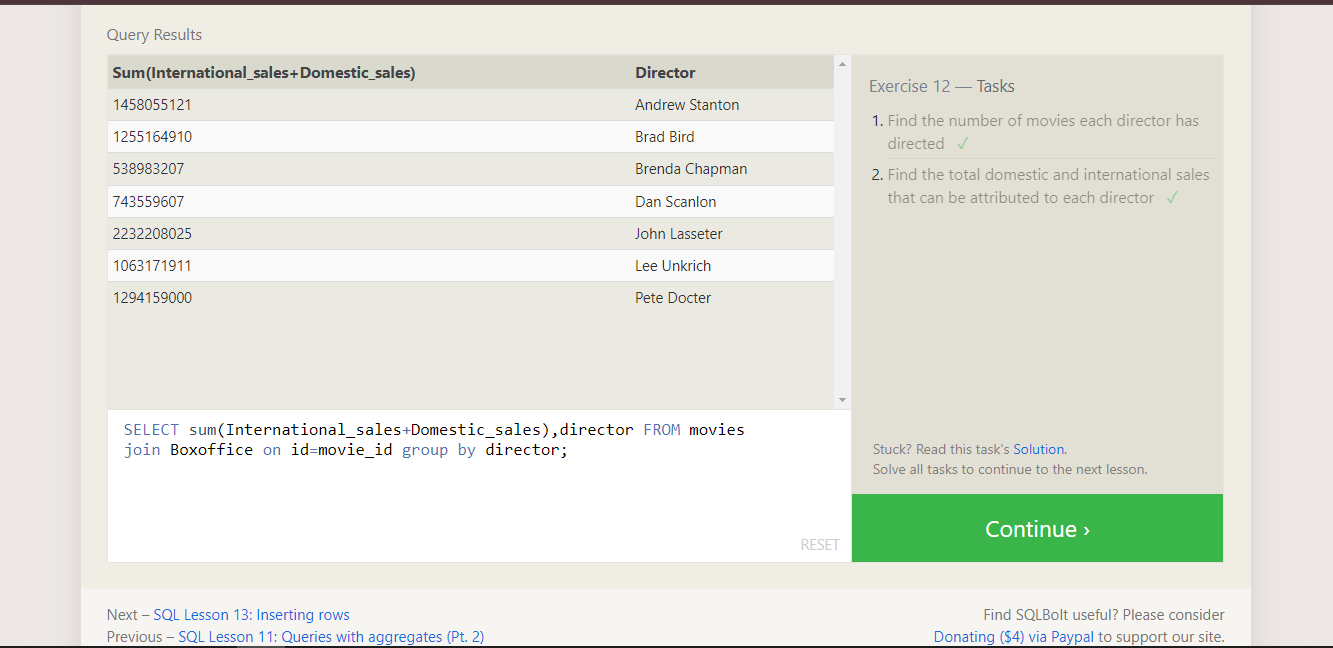
1. Find the number of movies each director has directed
2. Find the total domestic and international sales that can be attributed to each director

**Answers:**

1. SELECT count(title),director FROM movies group by director;
2. SELECT sum(International\_sales+Domestic\_sales),director FROM movies

join Boxoffice on id=movie\_id group by director;

**Screenshots:**



**SQL Lesson 13: Inserting rows**

**Questions:**

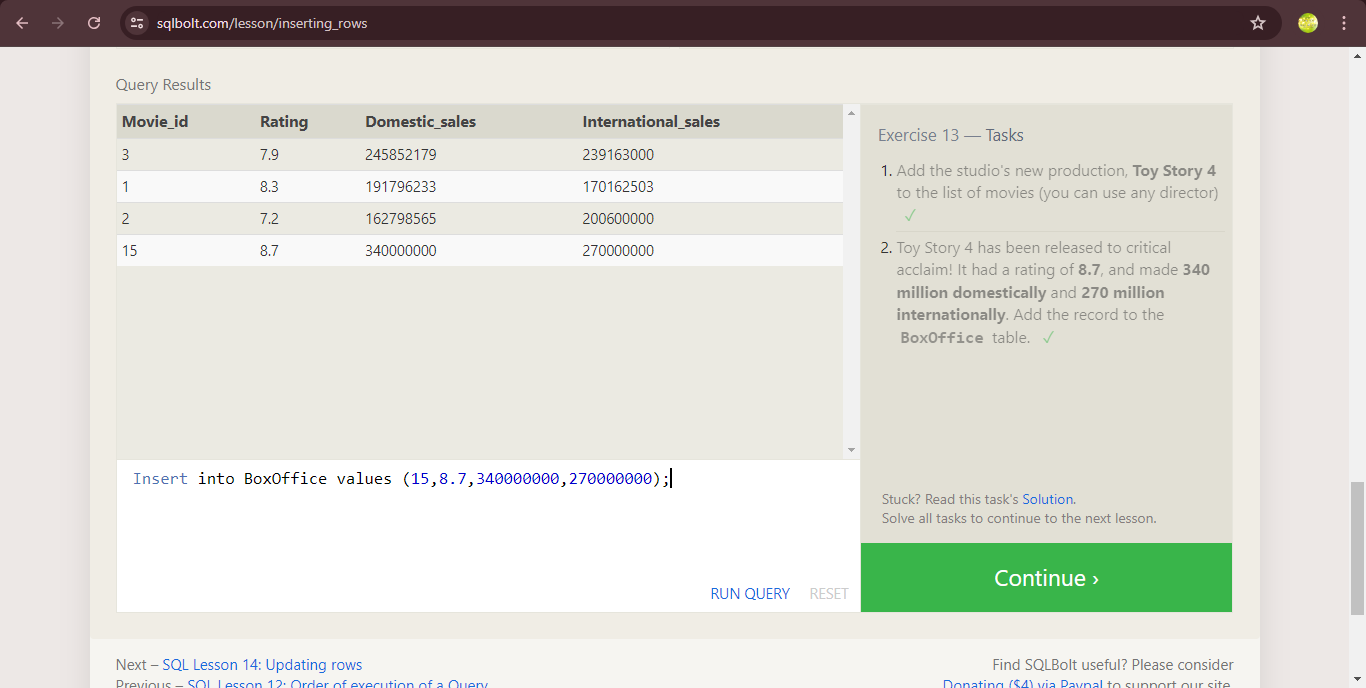
1. Add the studio's new production, Toy Story 4 to the list of movies (you can use any director)
2. Toy Story 4 has been released to critical acclaim! It had a rating of 8.7, and made 340 million domestically and 270 million internationally. Add the record to the BoxOffice table.

**Answers:**

**Insert into movies values (15,'Toy Story 4','Lee Unkrich','2024',100);**

**Insert into BoxOffice values (15,8.7,340000000,270000000);**

**Screenshots:**

****

**SQL Lesson 14: Updating rows**

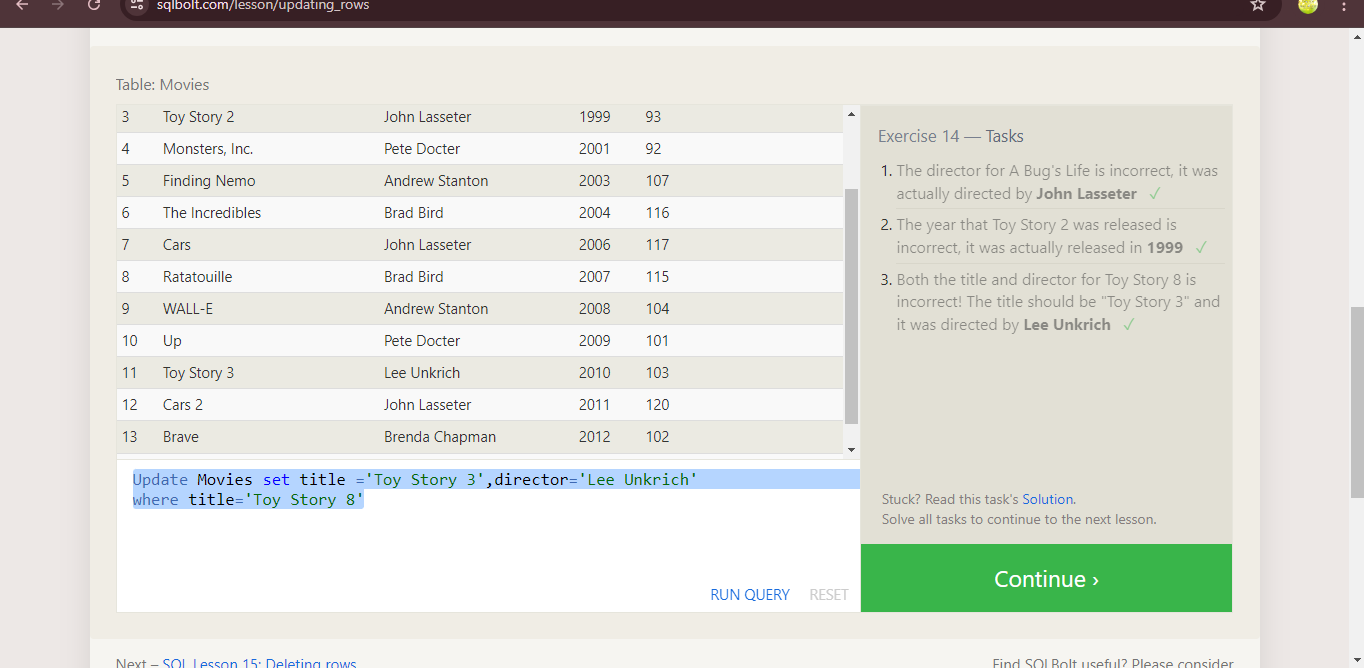
**Questions:**

1. The director for A Bug's Life is incorrect, it was actually directed by **John Lasseter**
2. The year that Toy Story 2 was released is incorrect, it was actually released in **1999**
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**

**Answers:**

1. Update Movies set director ='John Lasseter' where id=2
2. Update Movies set Year =1999 where title='Toy Story 2'
3. Update Movies set title ='Toy Story 3',director='Lee Unkrich' where title='Toy Story 8'

**Screenshots:**

****

**SQL Lesson 15: Deleting rows**

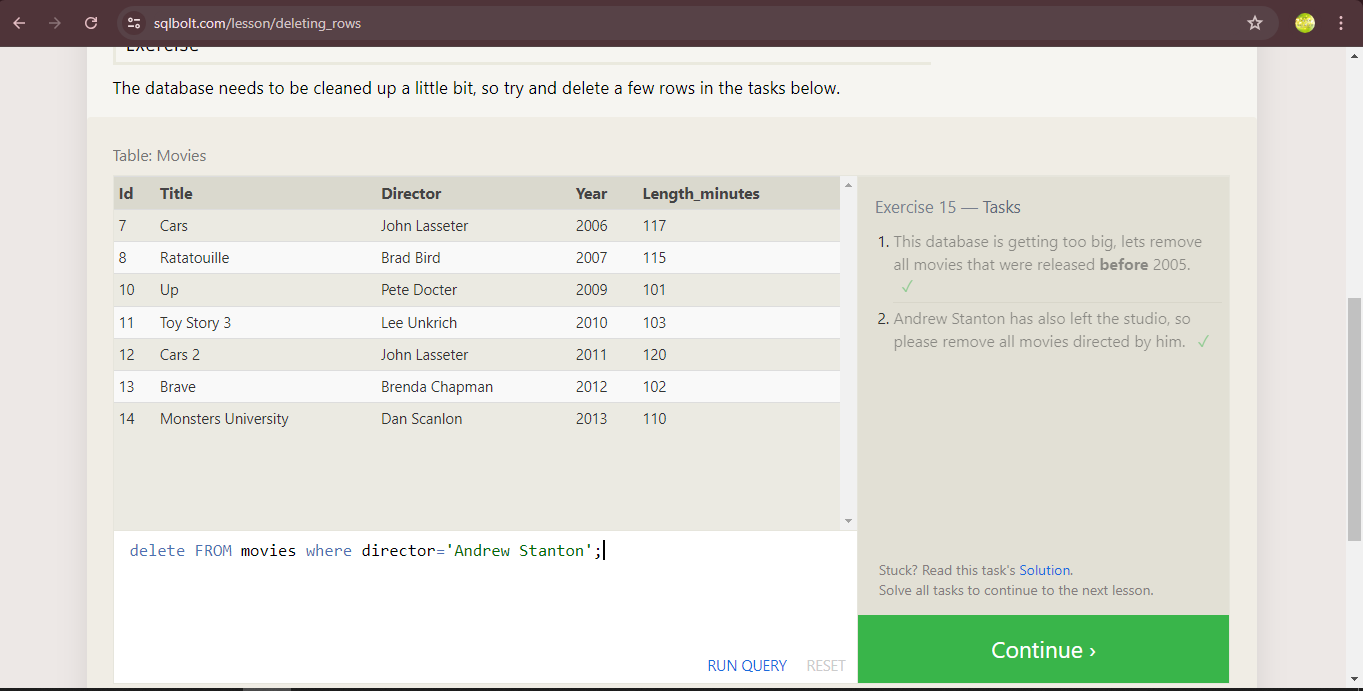
**Questions:**

1. This database is getting too big, lets remove all movies that were released **before** 2005.
2. Andrew Stanton has also left the studio, so please remove all movies directed by him.

**Answers:**

1. **delete FROM movies where year<2005;**
2. **delete FROM movies where director='Andrew Stanton';**

**Screenshots:**

****

**SQL Lesson 16: Creating tables**

**Questions:**

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

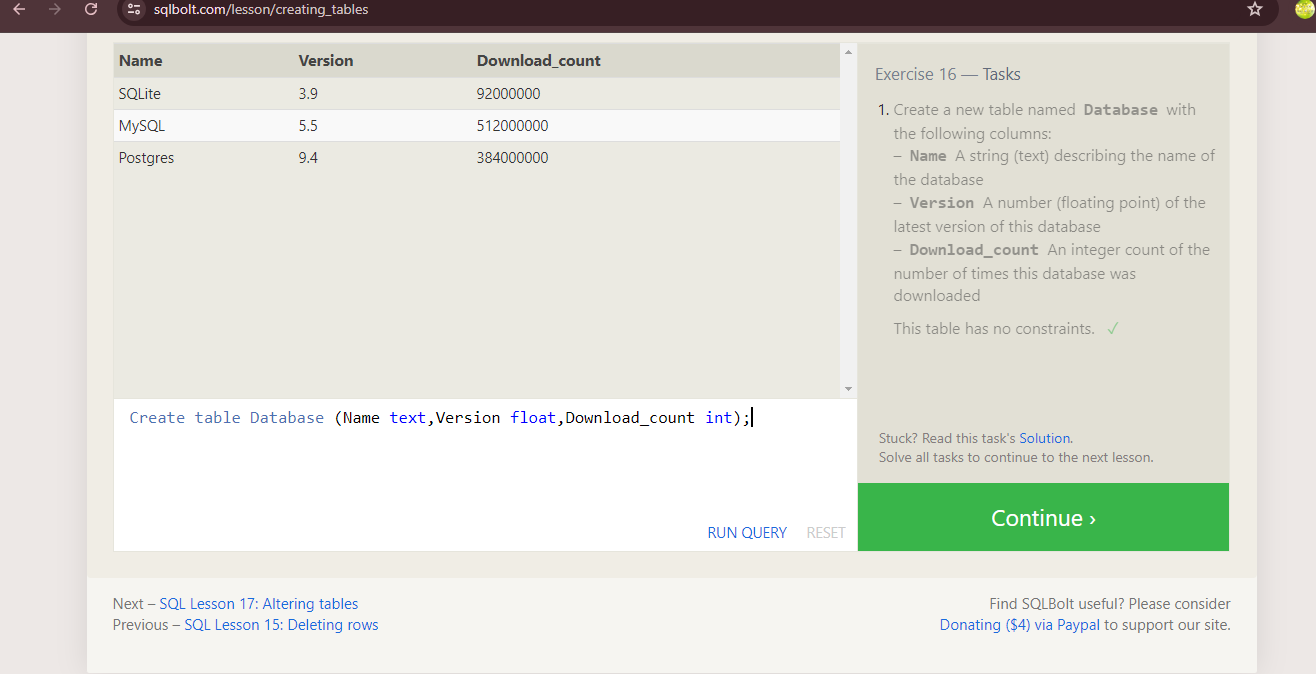
– **Name** A string (text) describing the name of the database  
– **Version** A number (floating point) of the latest version of this database  
– **Download\_count** An integer count of the number of times this database was downloaded

This table has no constraints.

**Answers:**

Create table Database (Name text,Version float,Download\_count int);

**Screenshots:**

****

**SQL Lesson 17: Altering tables**

**Questions:**

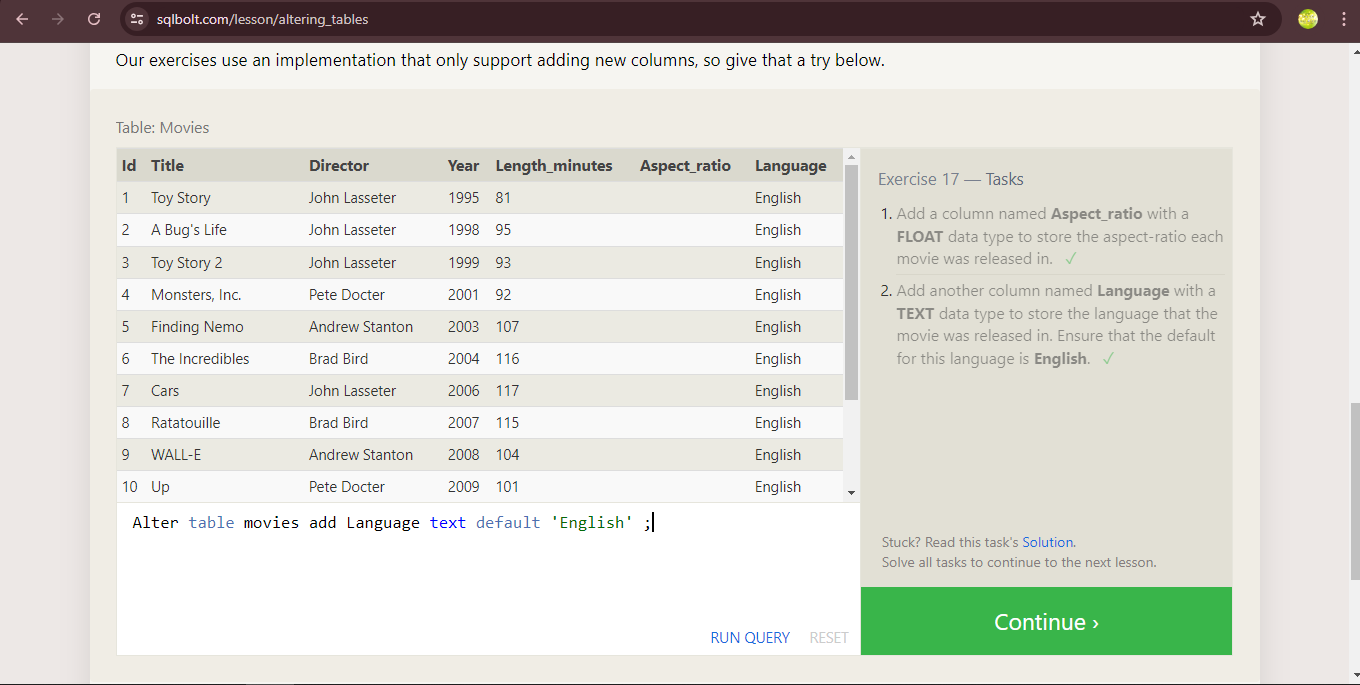
1. Add a column named **Aspect\_ratio** with a **FLOAT** data type to store the aspect-ratio each movie was released in.
2. Add another column named **Language** with a **TEXT** data type to store the language that the movie was released in. Ensure that the default for this language is **English**.

**Answers:**

Alter table movies add Aspect\_ratio float ;

Alter table movies add Language text default 'English' ;

**Screenshots:**

****

**SQL Lesson 18: Dropping tables**

**Questions:**

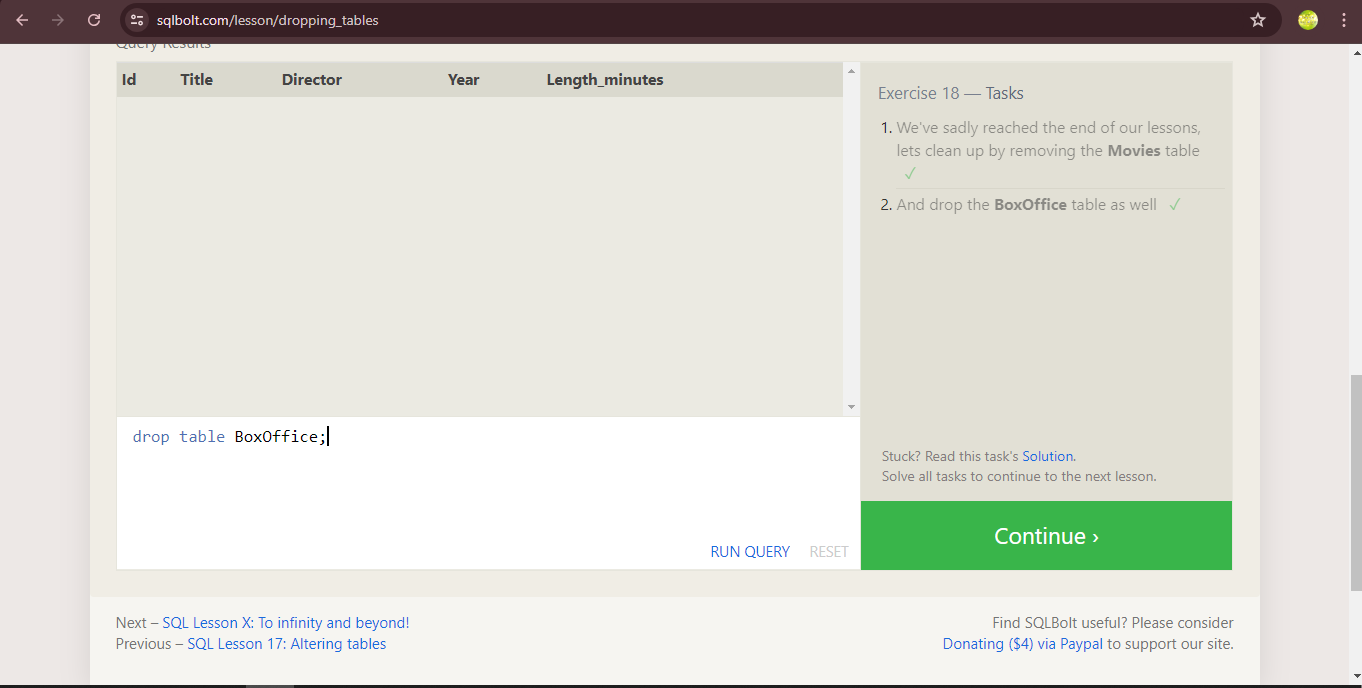
1. We've sadly reached the end of our lessons, lets clean up by removing the **Movies** table
2. And drop the **BoxOffice** table as well

**Answers:**

**drop table movies;**

**drop table BoxOffice;**

**Screenshots:**

****

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END OF DOCUMENT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*