SQL Lesson 1: SELECT queries 101

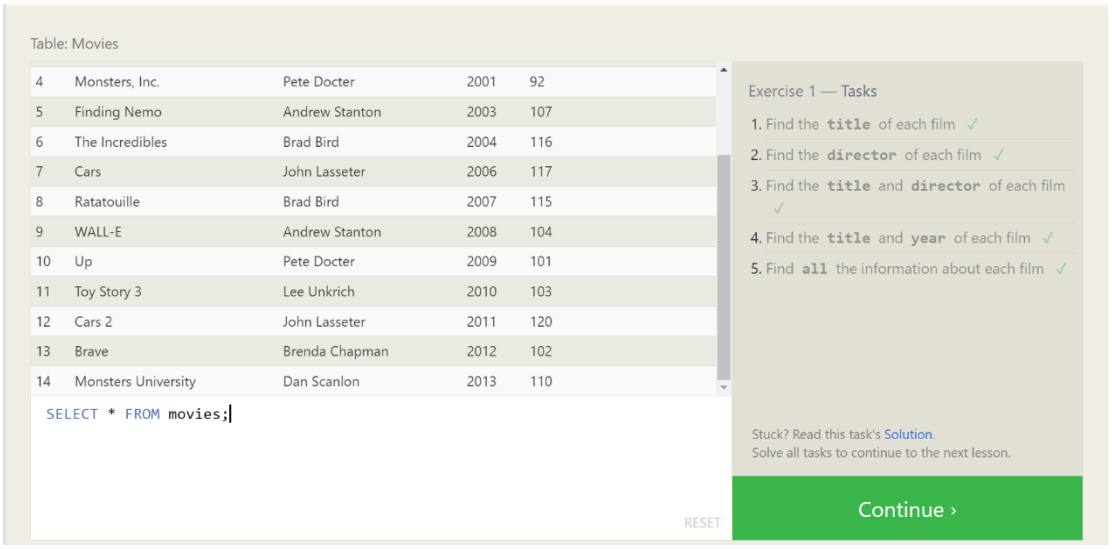


Table: Movies

4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

```
SELECT * FROM movies;
```

RESET

Exercise 1 — Tasks

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 ✓

Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

Continue ›

Exercise 1 — Tasks

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

   SELECT title FROM movies;

2. Find the **director** of each film

   SELECT director FROM movies;

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

   SELECT title, director FROM movies;

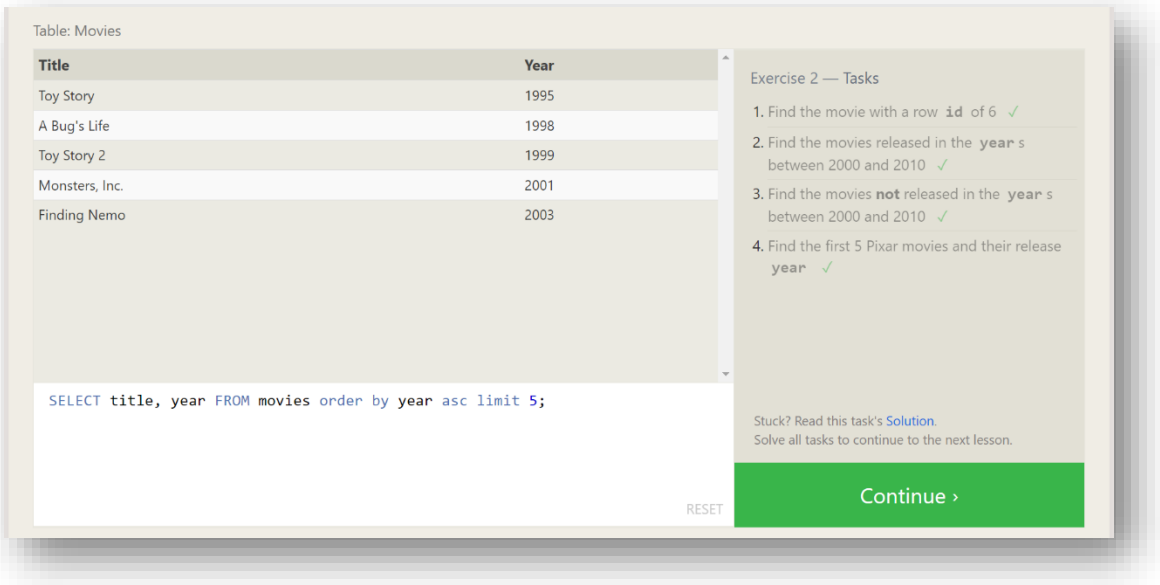
4. Find the **title** and **year** of each film

   SELECT title, year FROM movies;

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

   SELECT * FROM movies;

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



Exercise 2 — Tasks

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

   SELECT title FROM movies where id=6;

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

   SELECT * FROM movies where year between 2000 and 2010;

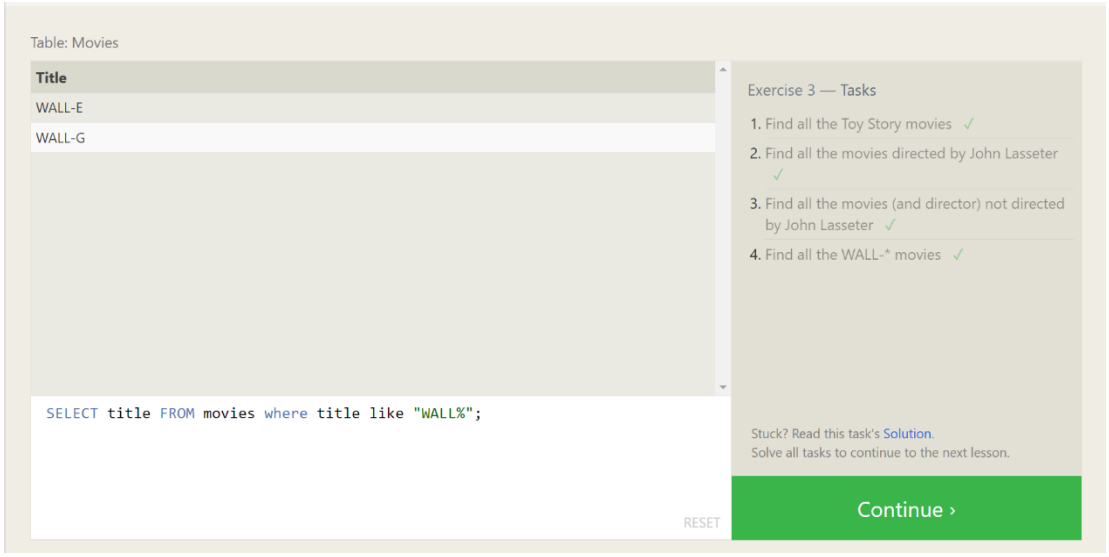
3. Find the movies **not** released in the **year**s between 2000 and 2010

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

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

   SELECT title, year FROM movies order by year asc limit 5;

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



Exercise 3 — Tasks

1. Find all the Toy Story movies

   SELECT title FROM movies where title like "Toy Story%";

2. Find all the movies directed by John Lasseter

   SELECT title FROM movies where director like "John%";

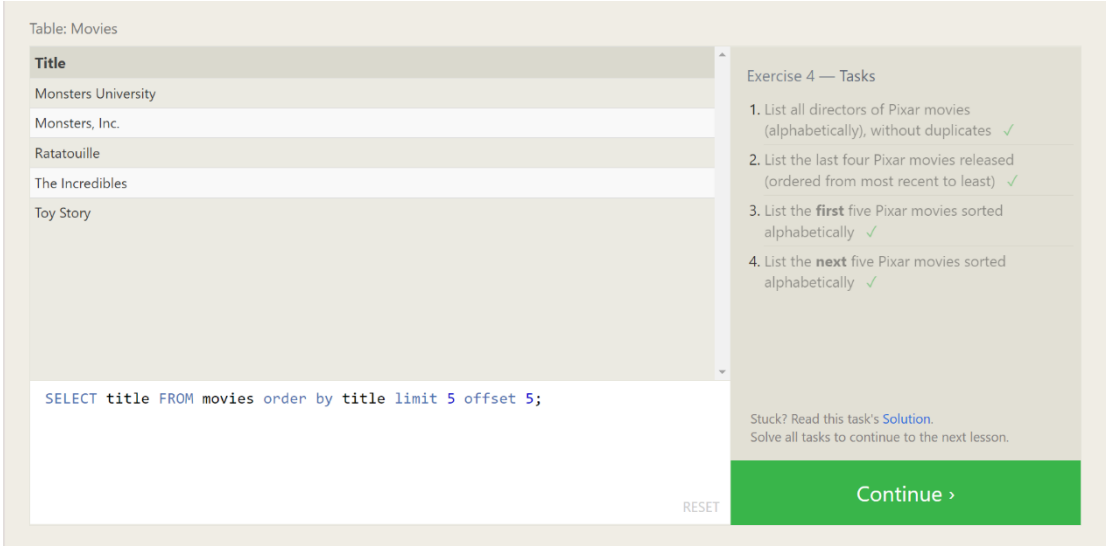
3. Find all the movies (and director) not directed by John Lasseter

   SELECT title, director FROM movies where director not like "John%";

4. Find all the WALL-* movies

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

SQL Lesson 4: Filtering and sorting Query results



Exercise 4 — Tasks

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

   SELECT distinct director FROM movies order by director;

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

   SELECT title FROM movies order by year desc limit 4;

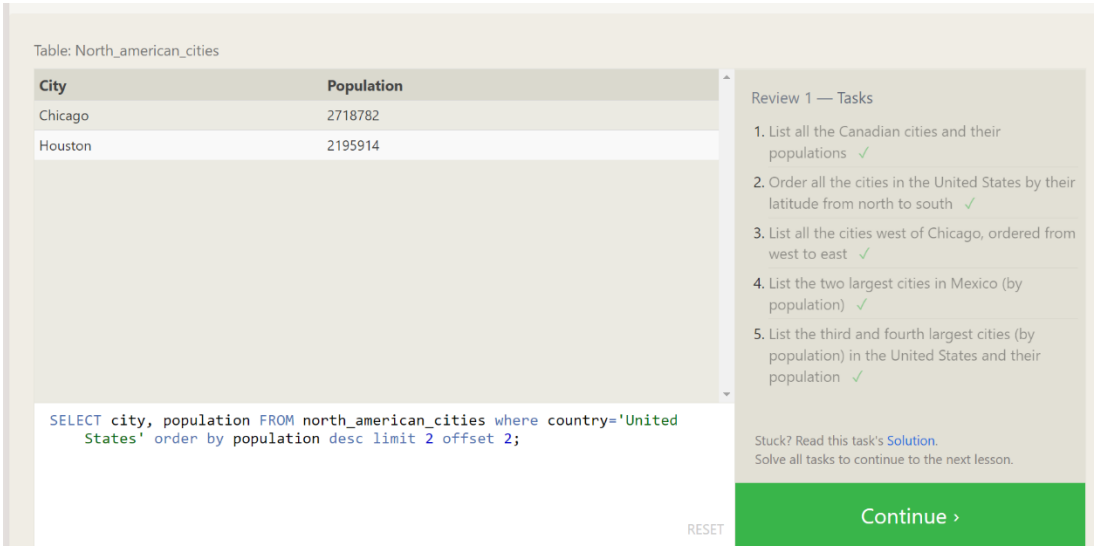
3. List the **first** five Pixar movies sorted alphabetically

   SELECT title FROM movies order by title limit 5;

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

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

SQL Review: Simple SELECT Queries



Review 1 — Tasks

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='United States' order by latitude desc;

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

   SELECT city, longitude FROM north_american_cities where longitude<-87.629798 order by longitude;

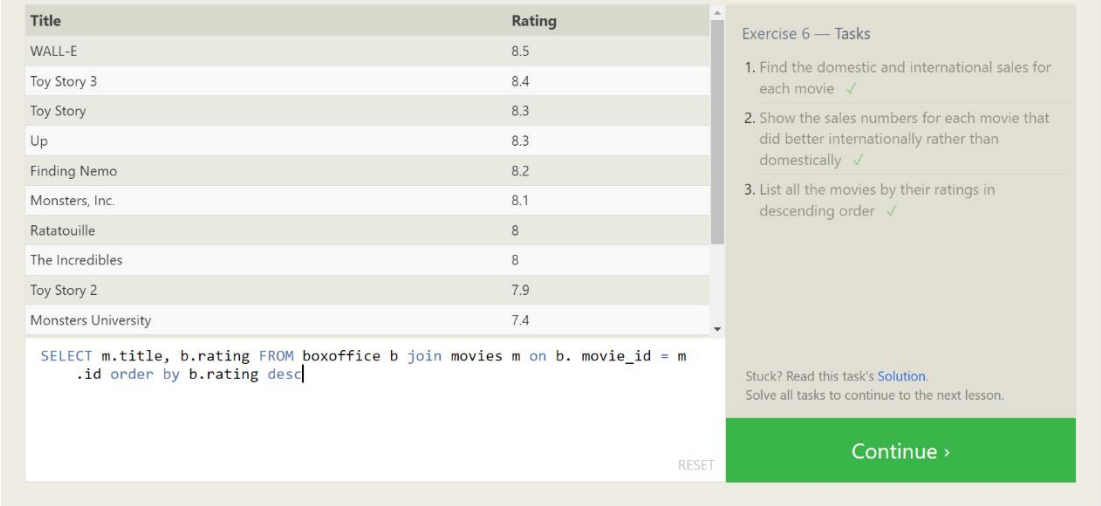
4. List the two largest cities in Mexico (by population)

   SELECT city FROM north_american_cities where country='Mexico' order by population desc limit 2;

5. 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



Title	Rating
WALL-E	8.5
Toy Story 3	8.4
Toy Story	8.3
Up	8.3
Finding Nemo	8.2
Monsters, Inc.	8.1
Ratatouille	8
The Incredibles	8
Toy Story 2	7.9
Monsters University	7.4

```
SELECT m.title, b.rating FROM boxoffice b join movies m on b. movie_id = m
    .id order by b.rating desc
```

RESET

Exercise 6 — Tasks

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 ✓

Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

Continue ›

Exercise 6 — Tasks

1. Find the domestic and international sales for each movie

   SELECT m.title, b.domestic_sales, international_sales FROM boxoffice b join movies m on b. movie_id = m.id;

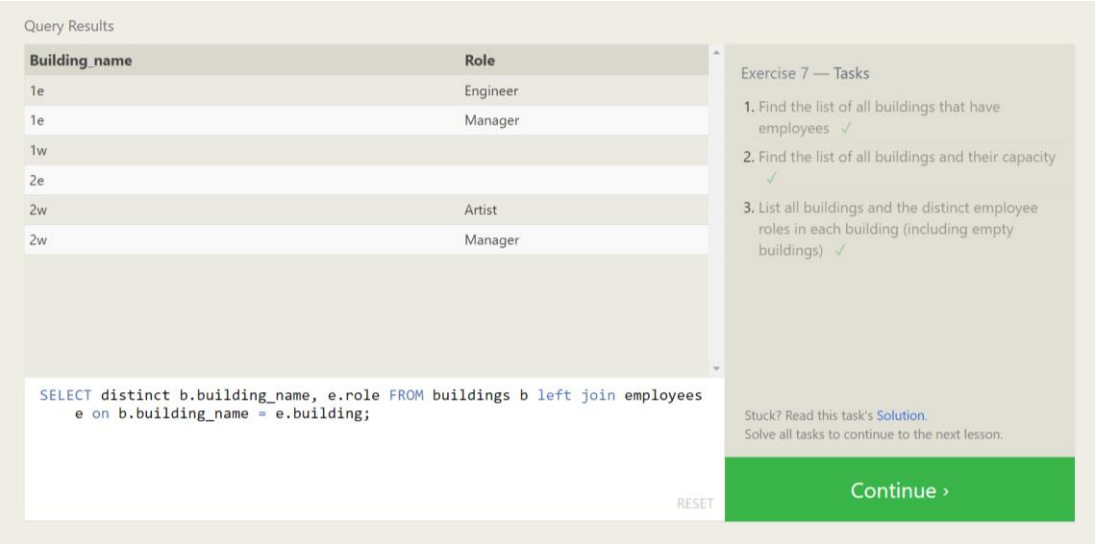
2. Show the sales numbers for each movie that did better internationally rather than domestically

   SELECT m.title, b.domestic_sales, b.international_sales FROM boxoffice b join movies m on b. movie_id = m.id where b.international_sales>b.domestic_sales;

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

   SELECT m.title, b.rating FROM boxoffice b join movies m on b. movie_id = m.id order by b.rating desc

SQL Lesson 7: OUTER JOINs



Exercise 7 — Tasks

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

   SELECT distinct building FROM employees;

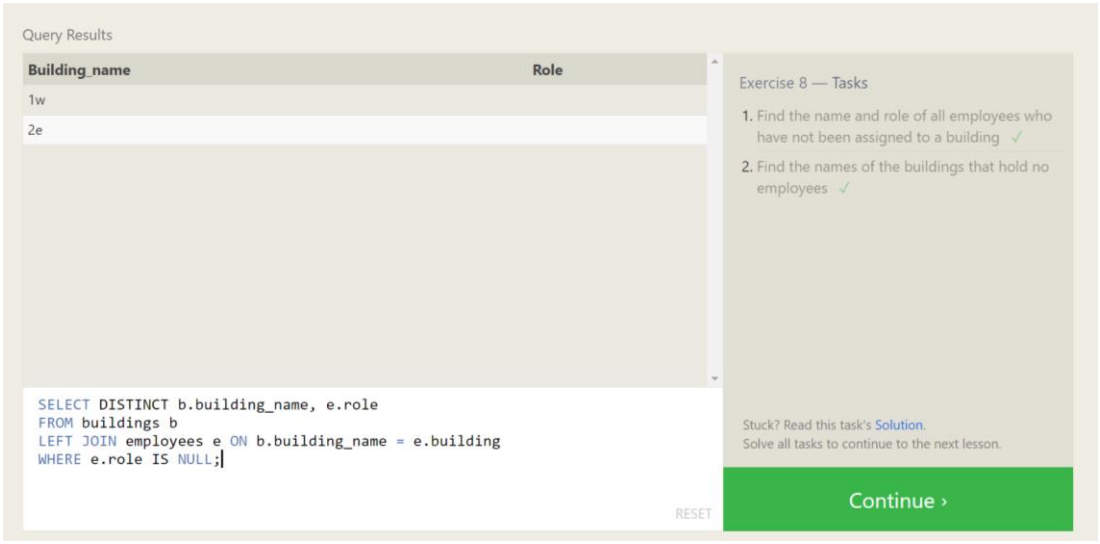
2. Find the list of all buildings and their capacity.

   SELECT building_name, capacity FROM buildings;

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

   SELECT distinct b.building_name, e.role FROM buildings b left join employees e on b.building_name = e.building;

SQL Lesson 8: A short note on NULLs



Exercise 8 — Tasks

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;

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

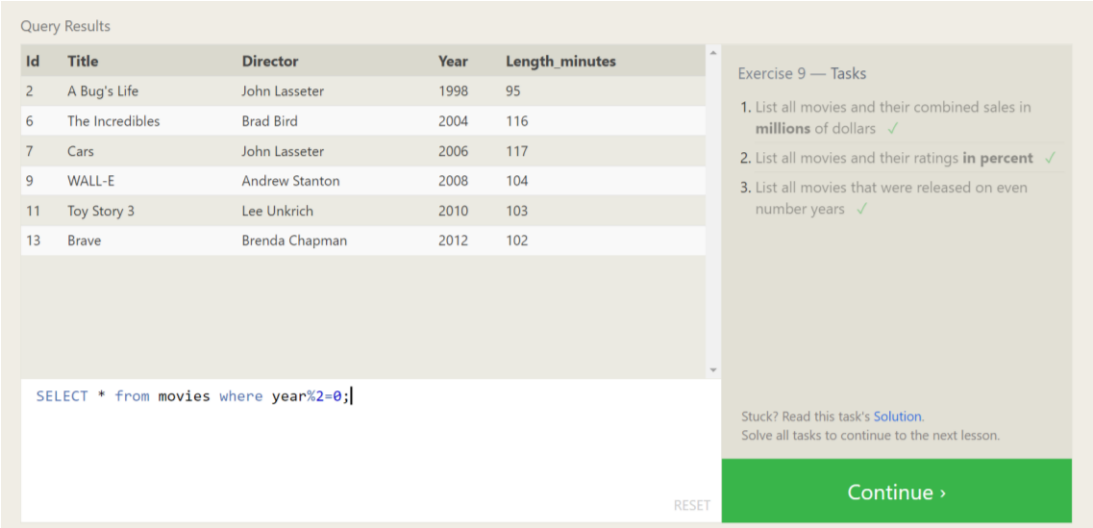
   SELECT DISTINCT b.building_name, e.role

   FROM buildings b

   LEFT JOIN employees e ON b.building_name = e.building

   WHERE e.role IS NULL;

SQL Lesson 9: Queries with expressions



Exercise 9 — Tasks

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

   SELECT m.title ,((b.domestic_sales+ b.international_sales)/10000000) as combined_sales FROM movies m join boxoffice b on m.id = b.movie_id;

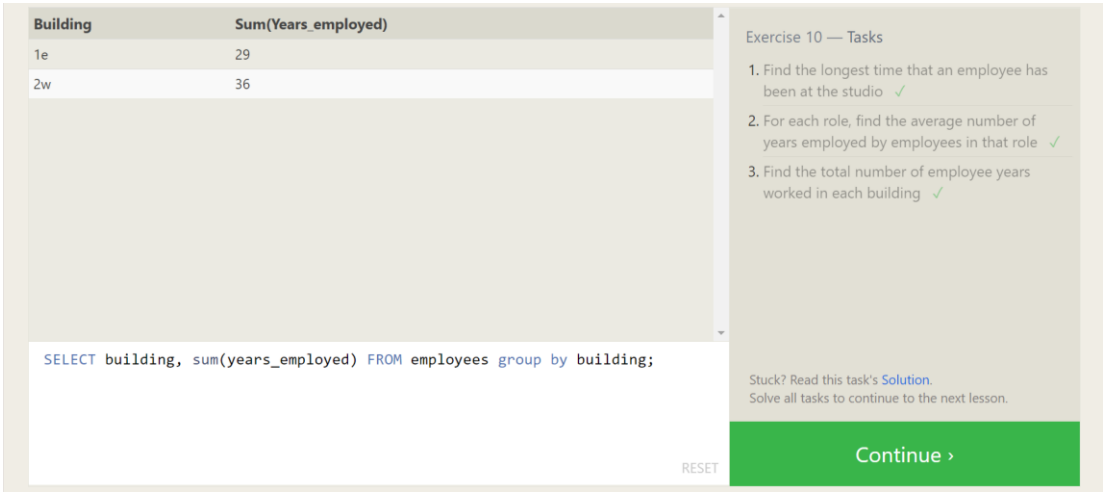
2. List all movies and their ratings **in percent**

   SELECT m.title ,(b.rating)*10 as ratings FROM movies m join boxoffice b on m.id = b.movie_id;

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

   SELECT * from movies where year%2=0;

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



Exercise 10 — Tasks

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

   SELECT MAX(years_employed) FROM employees;

2. 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) FROM employees group by building;

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

Exercise 11 — Tasks

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

   SELECT COUNT(name) FROM employees where role='Artist';

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

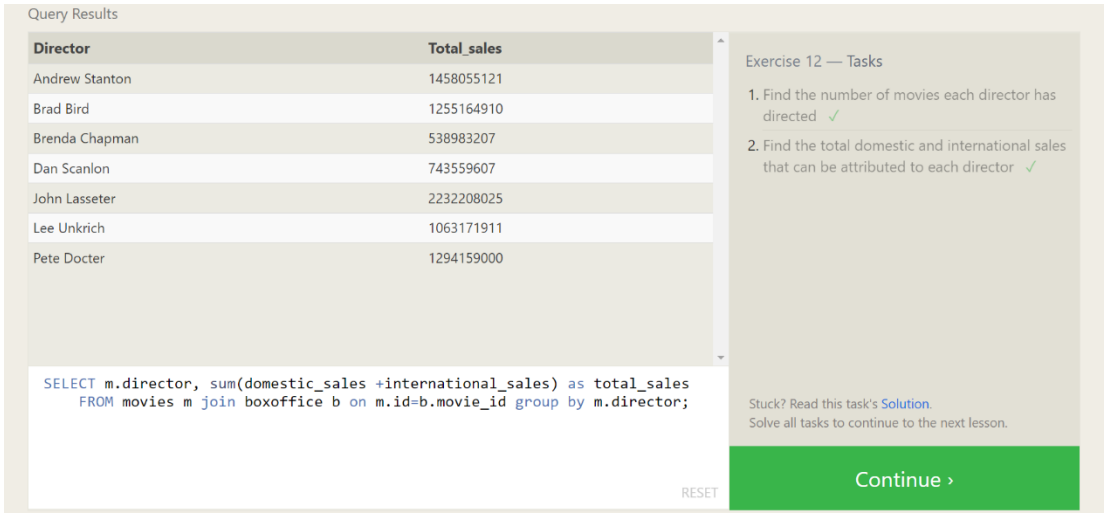
   SELECT role, COUNT(name) FROM employees group by role;

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



Exercise 12 — Tasks

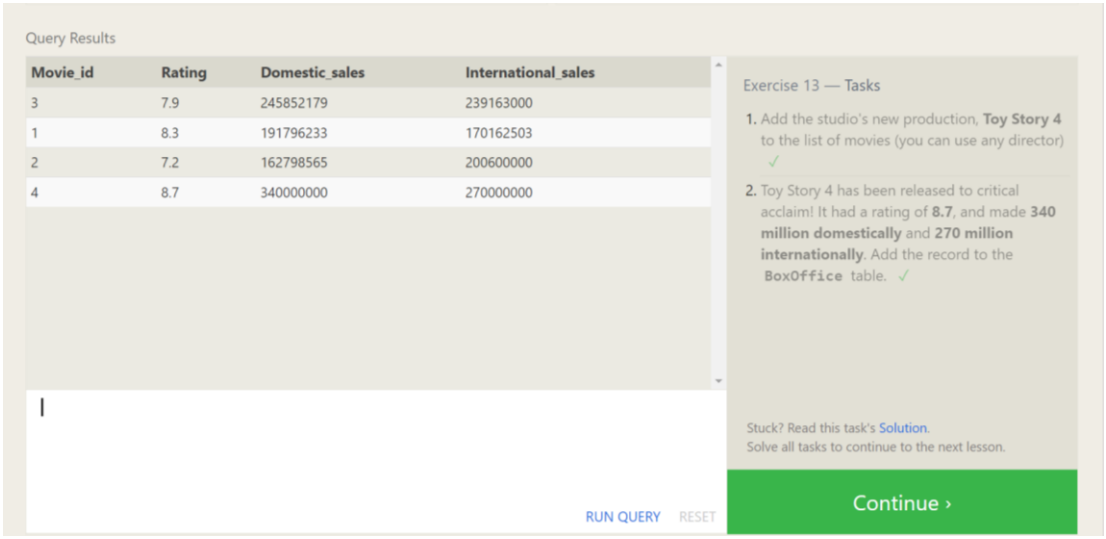
1. Find the number of movies each director has directed

   SELECT director, count(director) FROM movies group by director;

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

   SELECT m.director, sum(domestic_sales +international_sales) as total_sales FROM movies m join boxoffice b on m.id=b.movie_id group by m.director;

Exercise 13 — Tasks



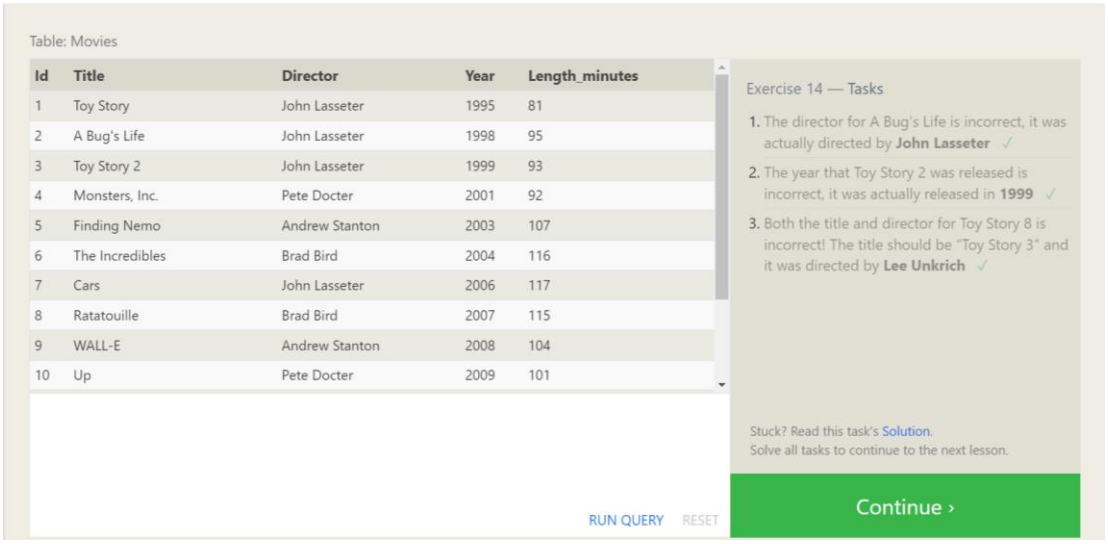
1. Add the studio's new production, **Toy Story 4** to the list of movies (you can use any director)

   INSERT into movies values (4, 'Toy Story 4',' John Lasseter',2003,93);

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.

   INSERT into boxoffice values(4,8.7,340*1000000,270*1000000);

SQL Lesson 14: Updating rows



Exercise 14 — Tasks

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';

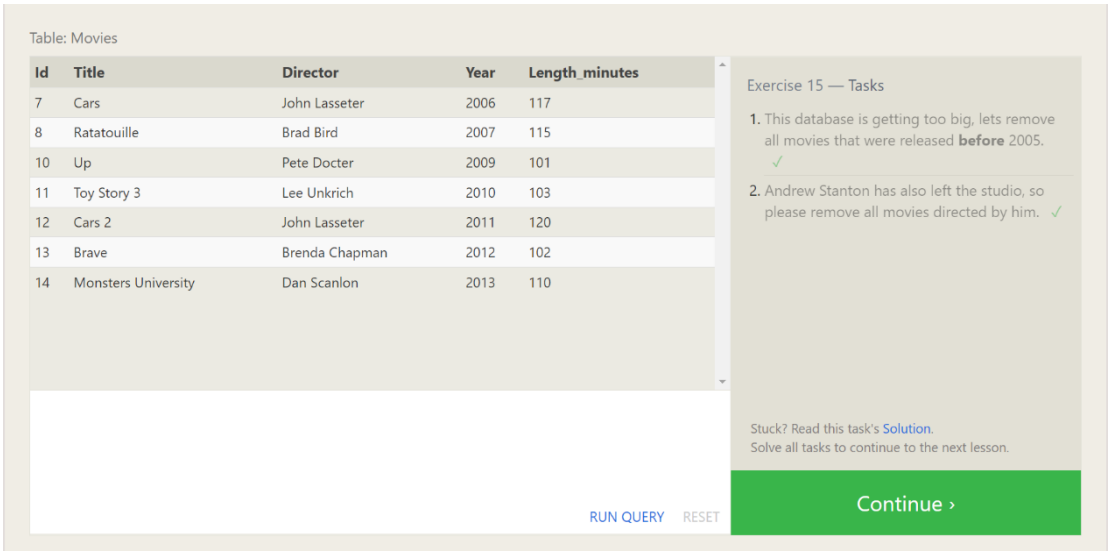
2. The year that Toy Story 2 was released is incorrect, it was actually released in **1999**

   UPDATE movies set year=1999 where title="Toy Story 2";

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 title='Toy Story 8';

SQL Lesson 15: Deleting rows



Exercise 15 — Tasks

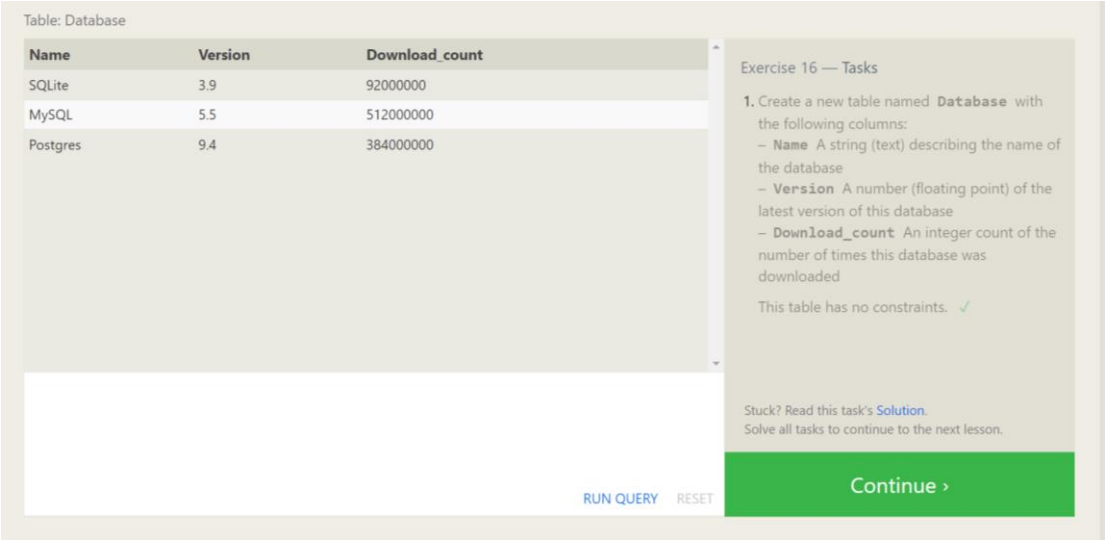
1. This database is getting too big, lets remove all movies that were released **before** 2005.

   DELETE FROM movies where year<2005;

2. 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



Exercise 16 — Tasks

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 varchar(255),
Version float,
Download_count int
);
```