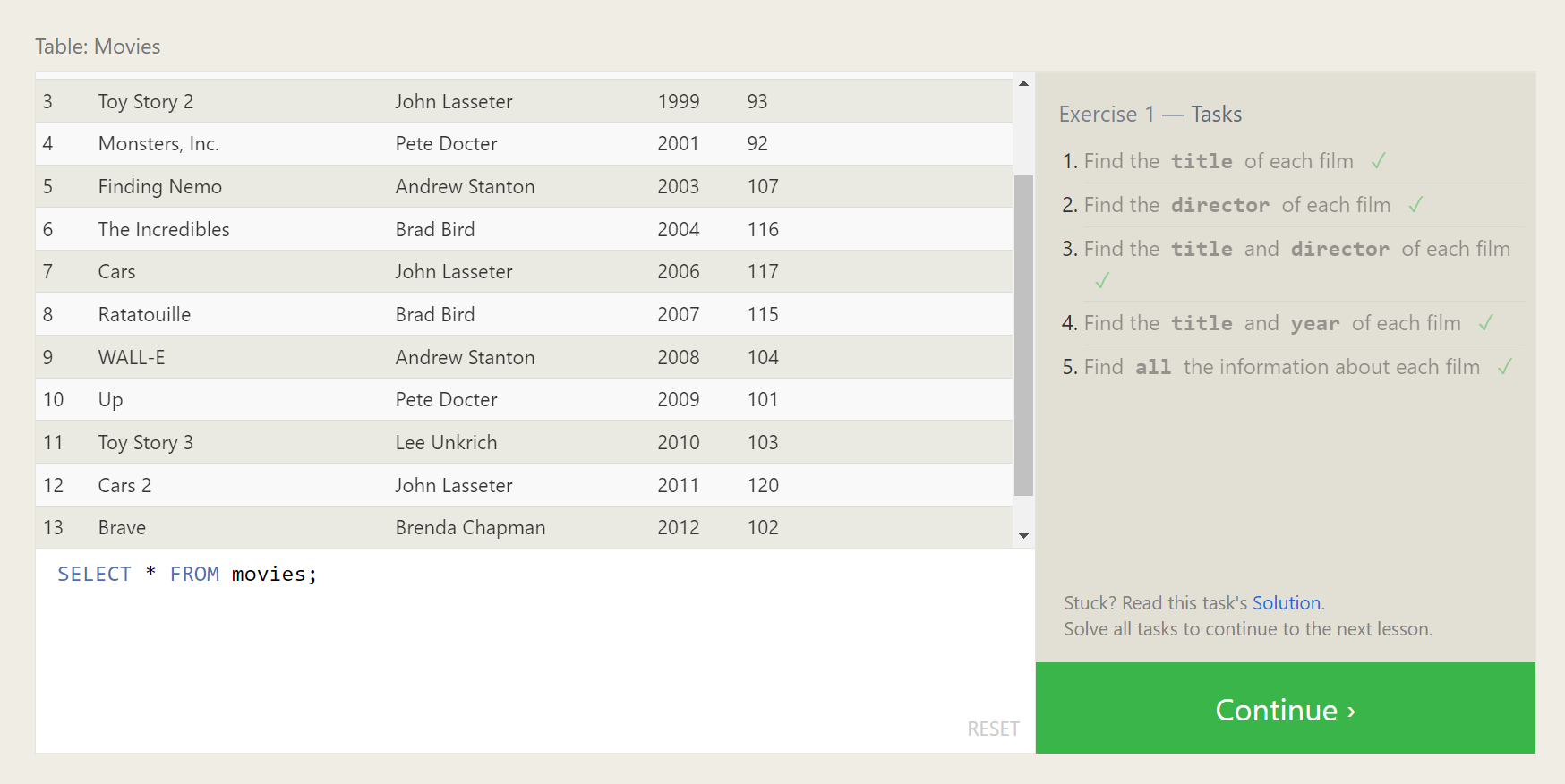
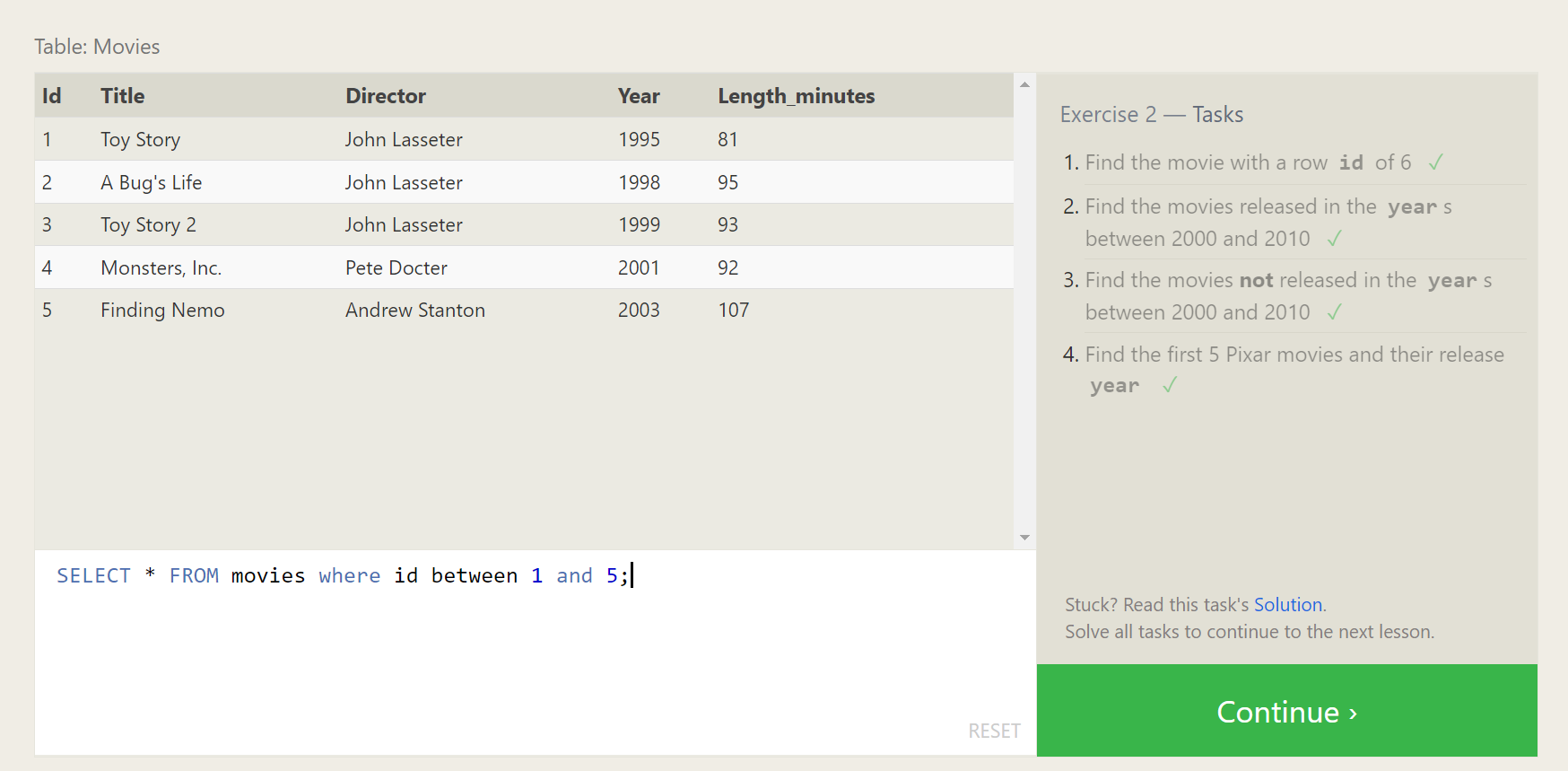
SQL Lesson 1: SELECT queries 101



Queries:

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

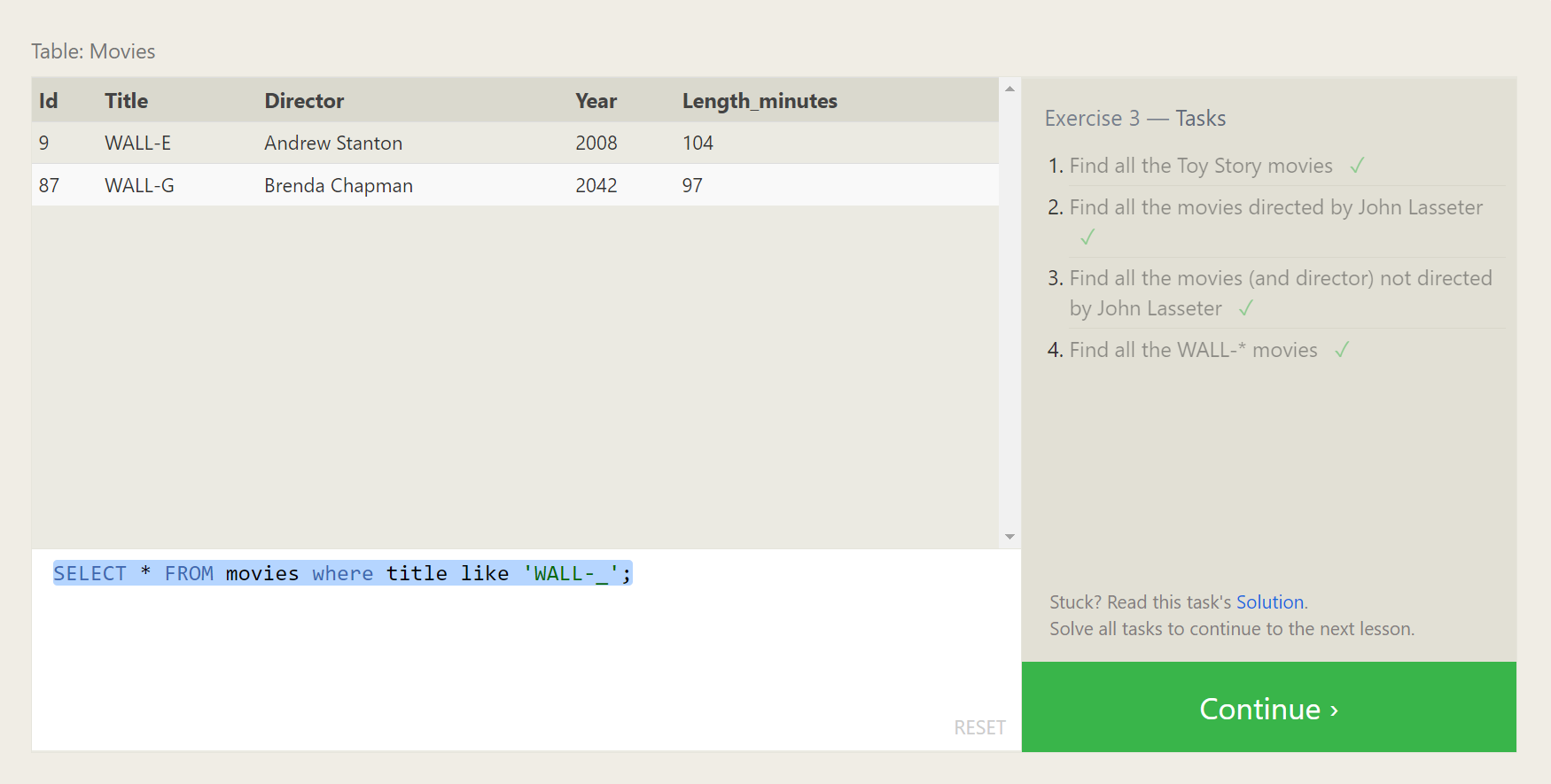
SQL Lesson 2: Queries with constraints (Pt. 1)



Queries:

1. SELECT title 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 where id between 1 and 5;

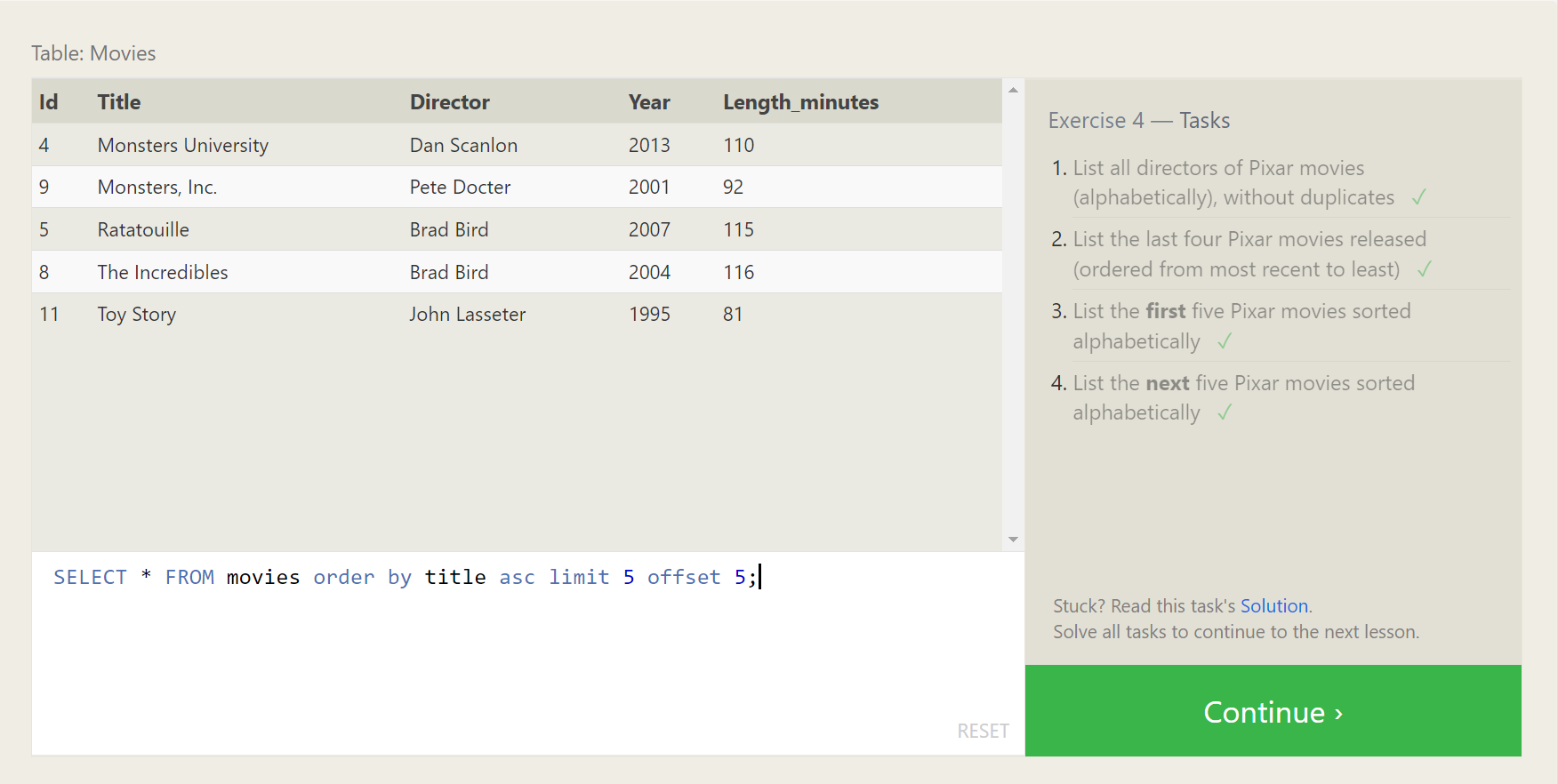
SQL Lesson 3: Queries with constraints (Pt. 2)



Queries:

1. SELECT \* FROM movies where title in ("Toy Story","Toy Story 2","Toy Story 3");
2. SELECT \* FROM movies where director = 'John Lasseter';
3. SELECT \* FROM movies where director != 'John Lasseter';
4. SELECT \* FROM movies where title like 'WALL-\_';

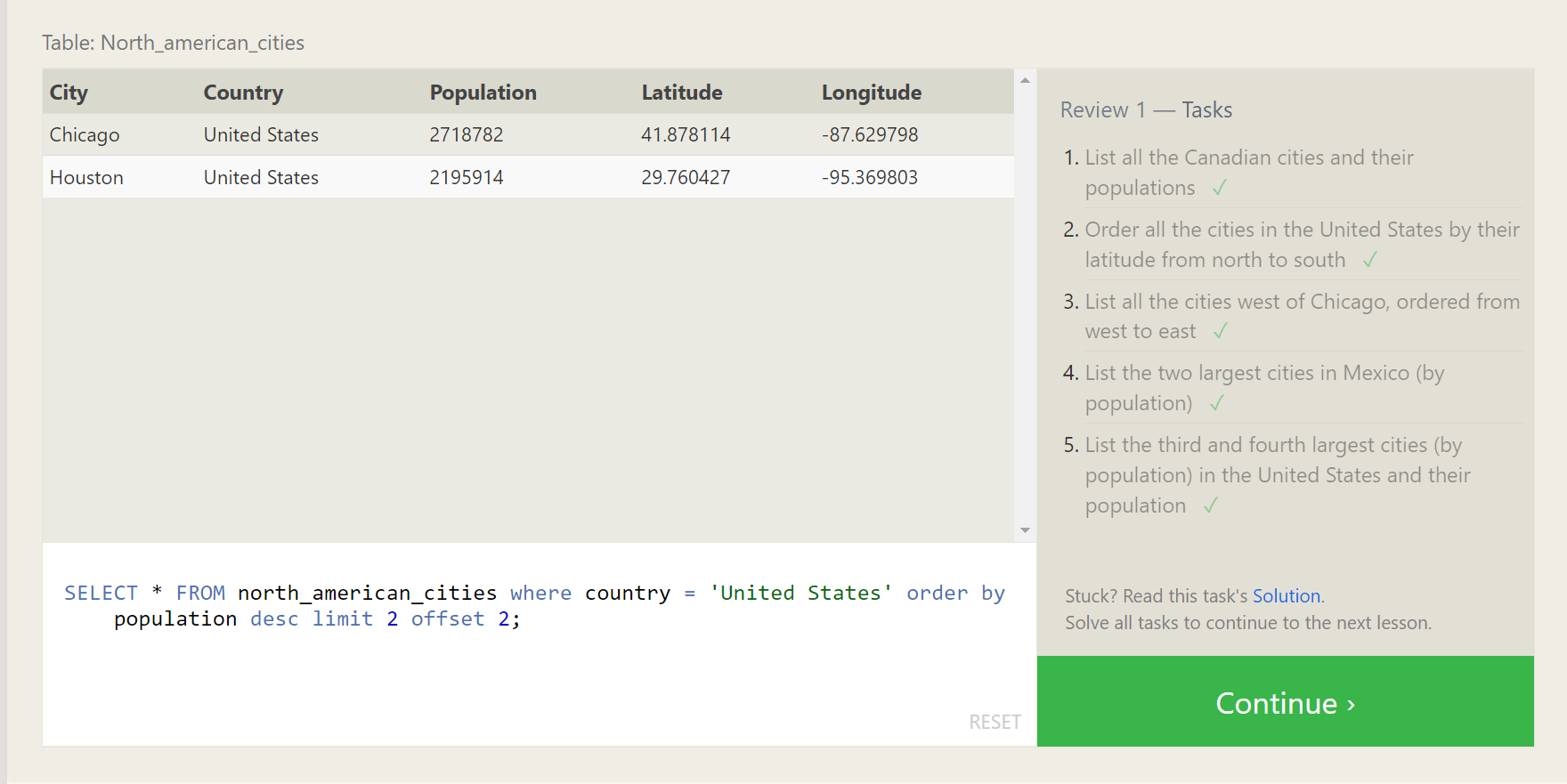
SQL Lesson 4: Filtering and sorting Query results



Queries:

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

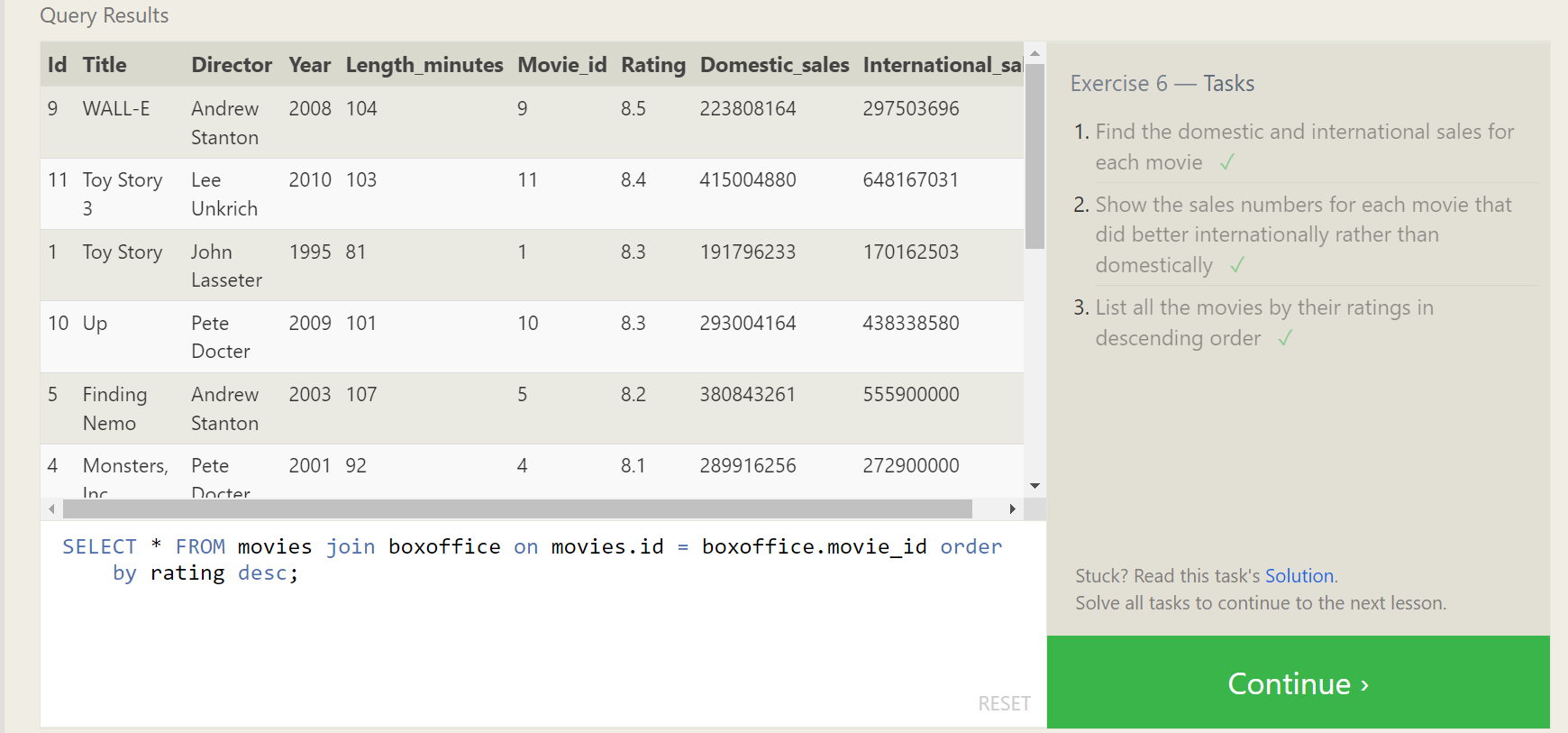
SQL Review: Simple SELECT Queries



Queries:

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 < (select longitude from north\_american\_cities where city = 'Chicago') order by longitude asc;
4. SELECT \* FROM north\_american\_cities where country = '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;

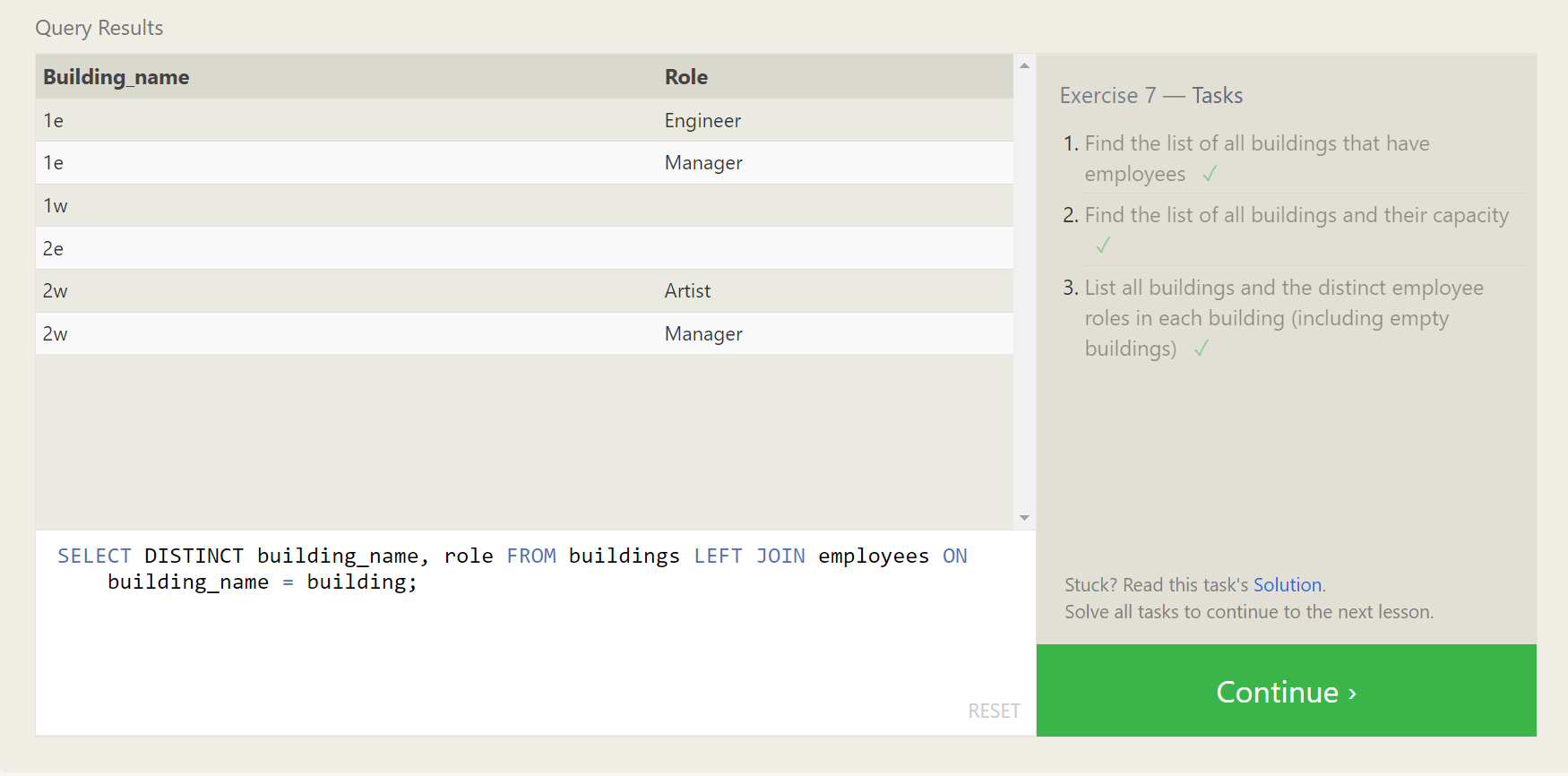
SQL Lesson 6: Multi-table queries with JOINs



Queries:

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

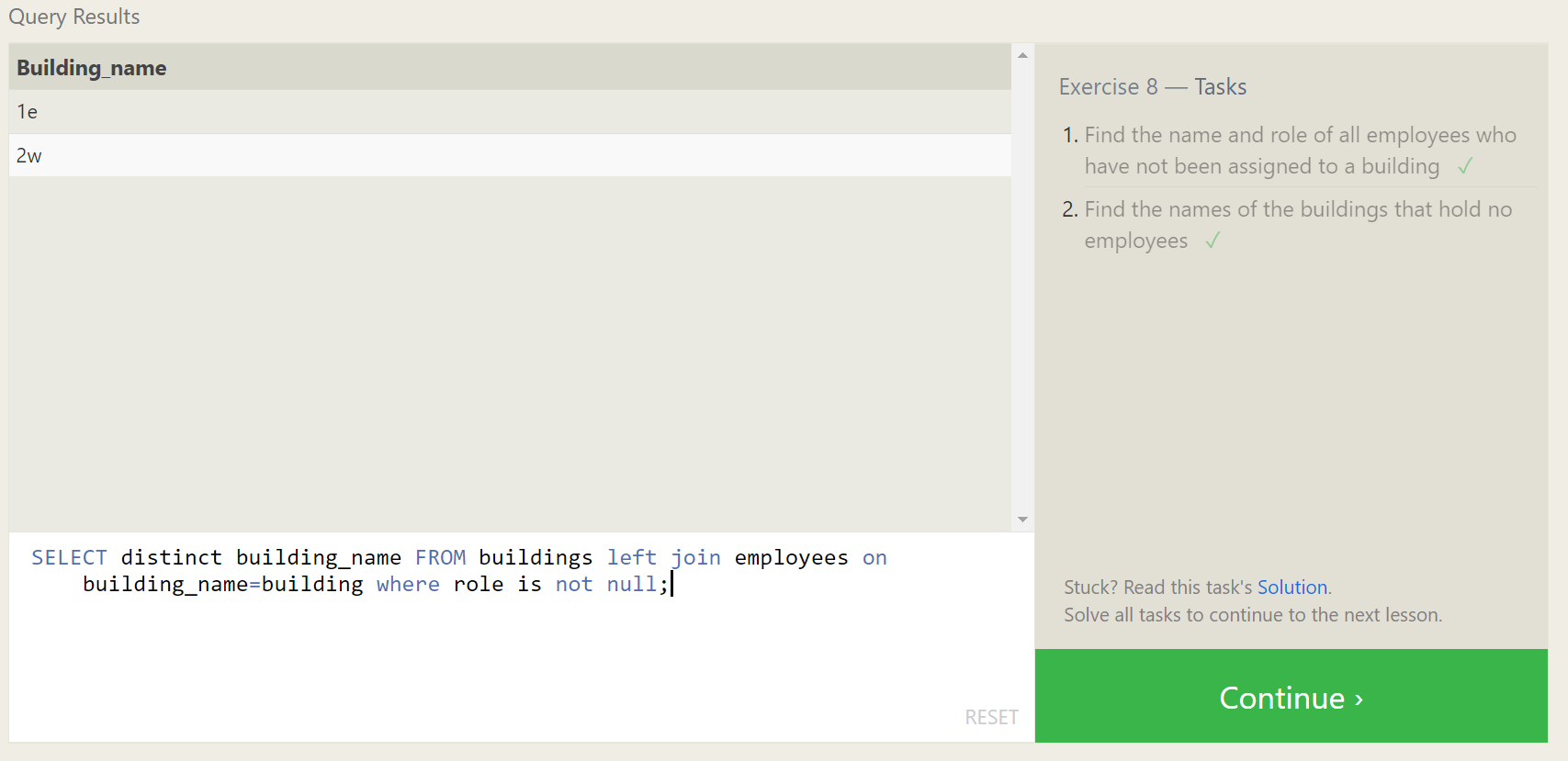
SQL Lesson 7: OUTER JOINs



Queries:

1. SELECT building FROM employees join buildings on buildings.building\_name = employees.building group by building;
2. SELECT building\_name,capacity FROM buildings;
3. SELECT DISTINCT building\_name, role FROM buildings LEFT JOIN employees ON building\_name = building;

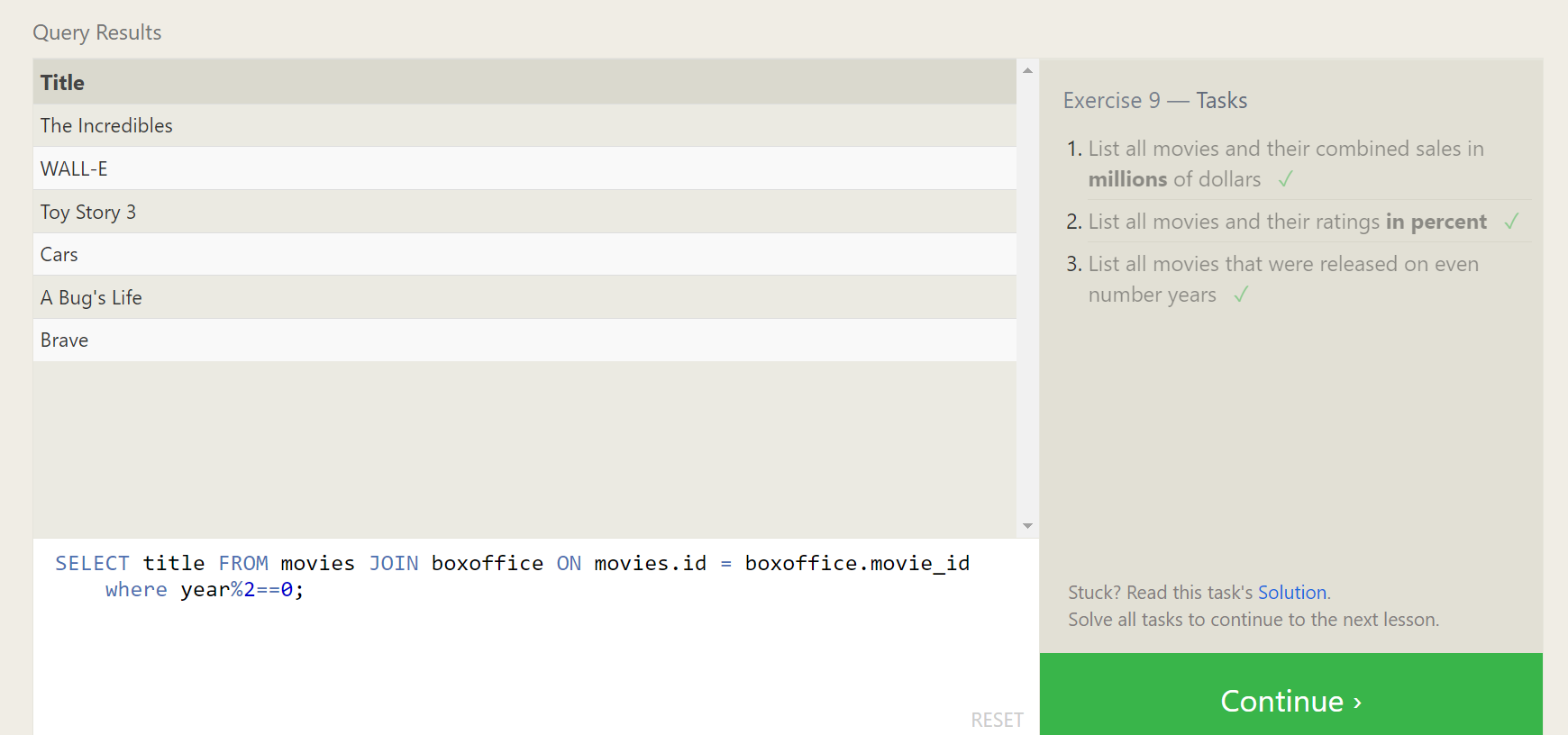
SQL Lesson 8: A short note on NULLs



Queries:

1. SELECT name,role FROM employees where building is null;
2. SELECT distinct building\_name FROM buildings left join employees on building\_name=building where role is not null;

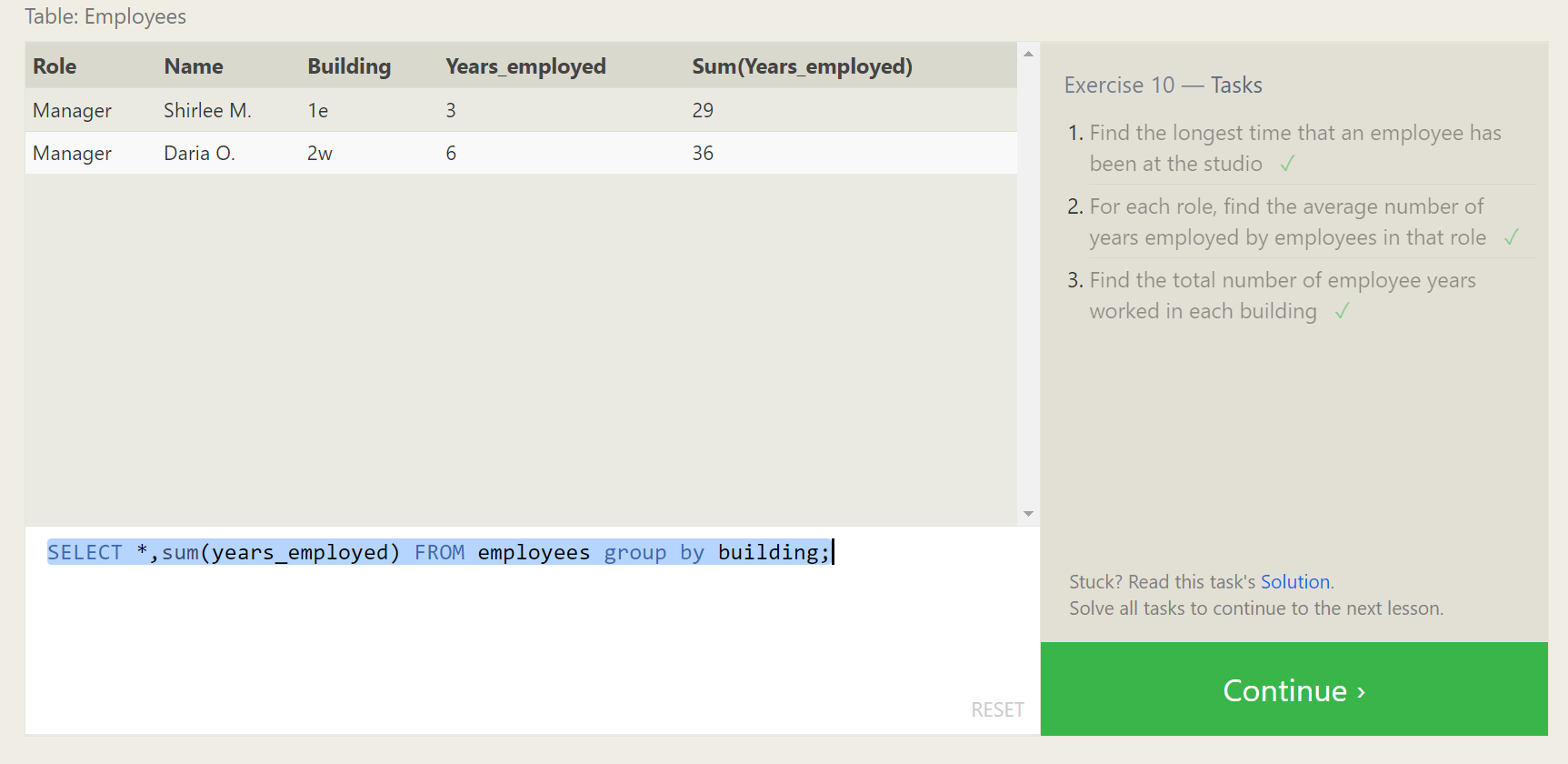
SQL Lesson 9: Queries with expressions



Queries:

1. SELECT title, (domestic\_sales + international\_sales) / 1000000 AS millions FROM movies JOIN boxoffice ON movies.id = boxoffice.movie\_id;
2. SELECT title, (rating) \* 10 AS Ratings FROM movies JOIN boxoffice ON movies.id = boxoffice.movie\_id;
3. SELECT title FROM movies JOIN boxoffice ON movies.id = boxoffice.movie\_id where year%2==0;

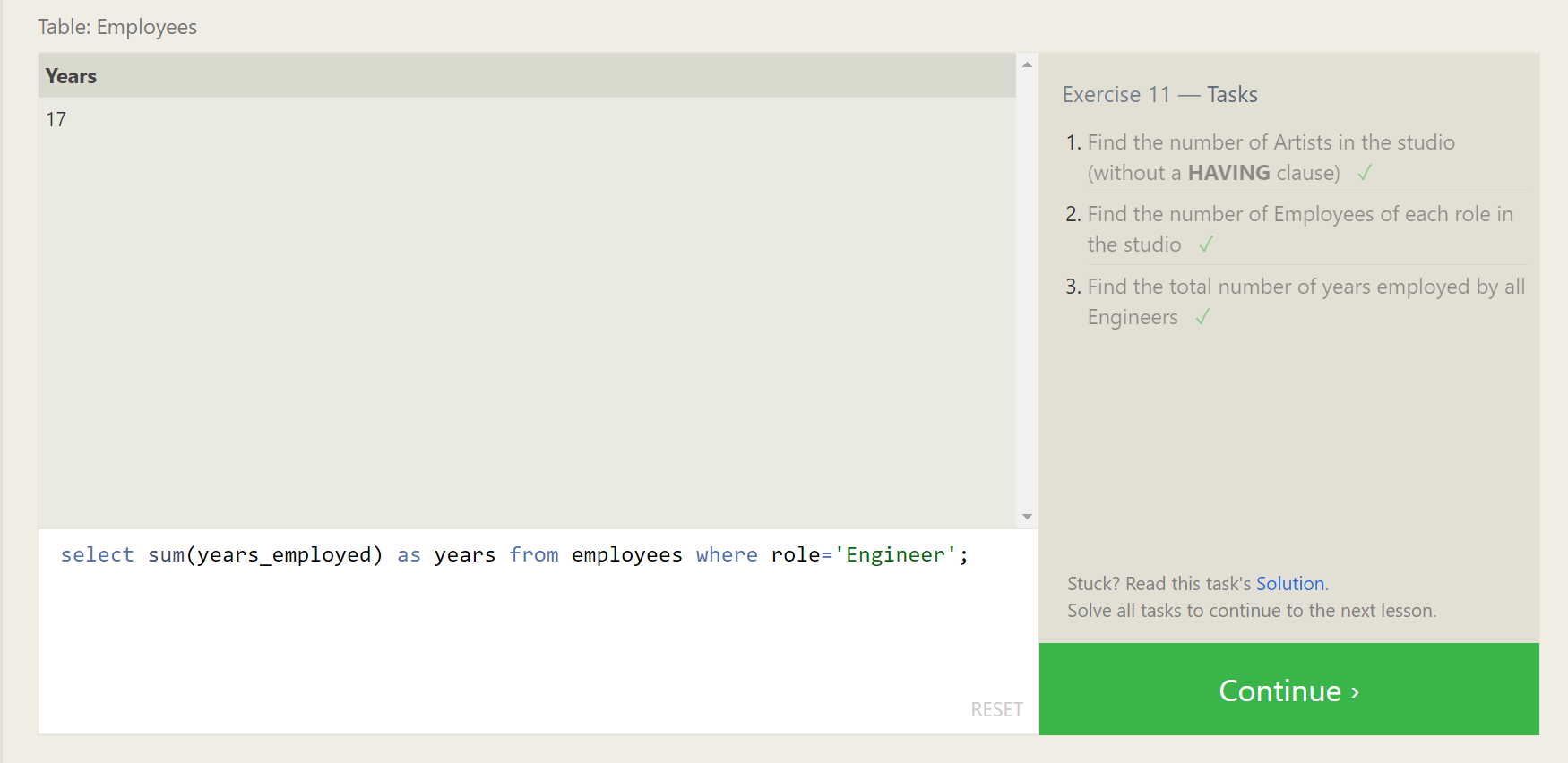
SQL Lesson 10: Queries with aggregates (Pt. 1)



Queries:

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

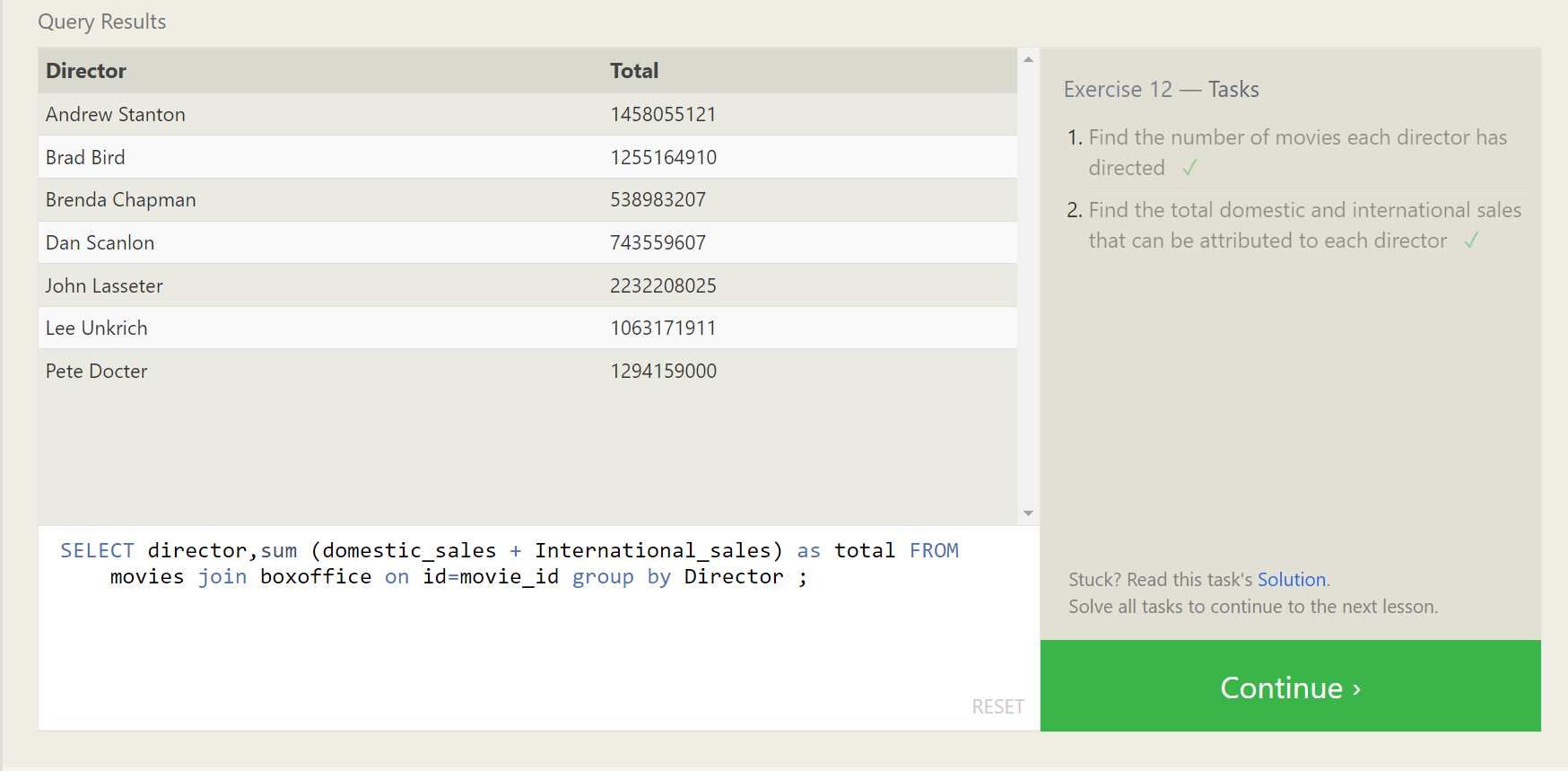
SQL Lesson 11: Queries with aggregates (Pt. 2)



Queries:

1. SELECT count(role) FROM employees where role = 'Artist' ;
2. select role,count(role) from employees group by role;
3. SELECT role, SUM(years\_employed) FROM employees GROUP BY role HAVING role = "Engineer";

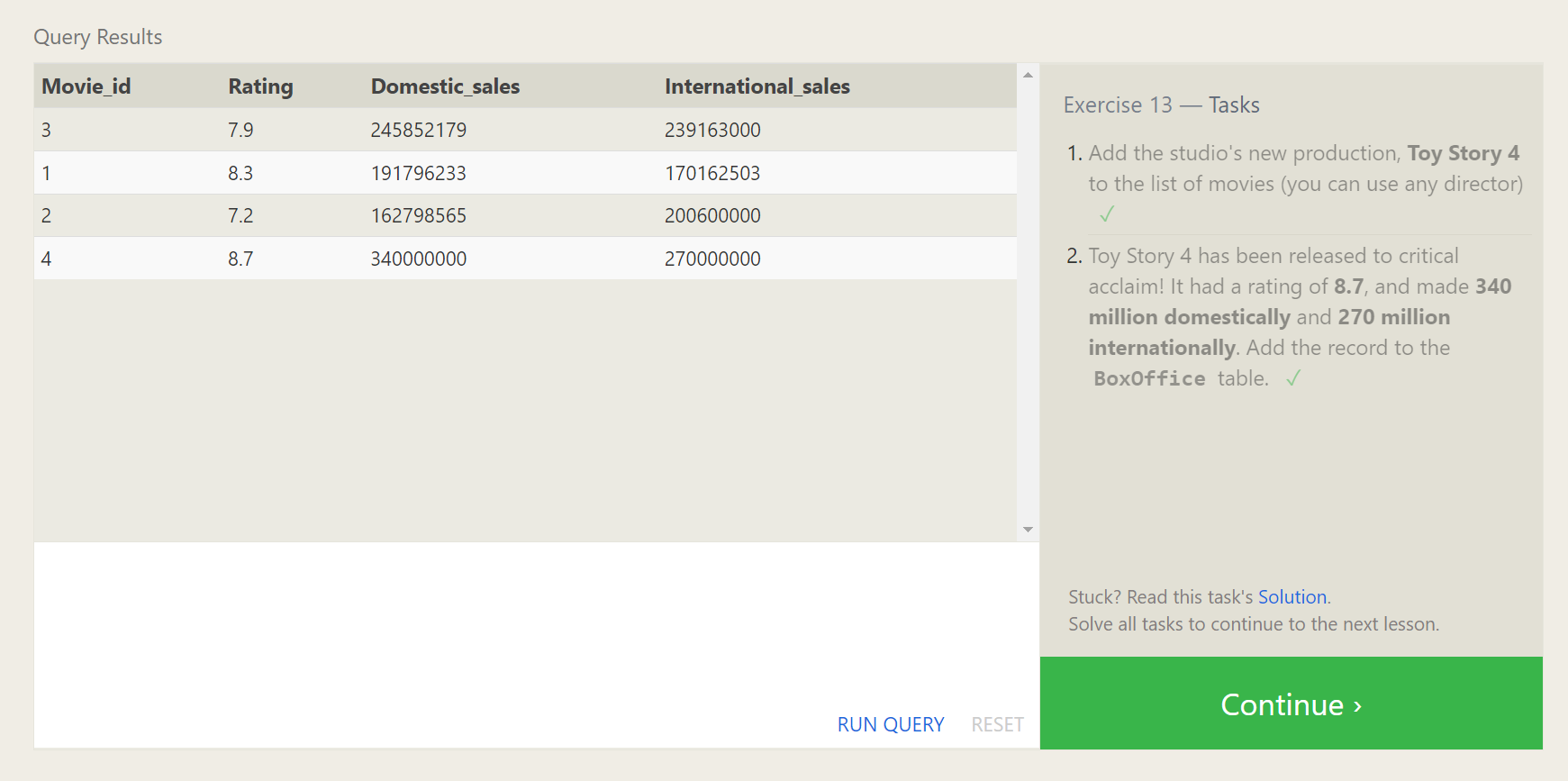
SQL Lesson 12: Order of execution of a Query



Queries:

1. SELECT Director, count(director) FROM movies group by director;
2. SELECT director, sum (domestic\_sales + International\_sales) as total FROM movies join boxoffice on id=movie\_id group by Director ;

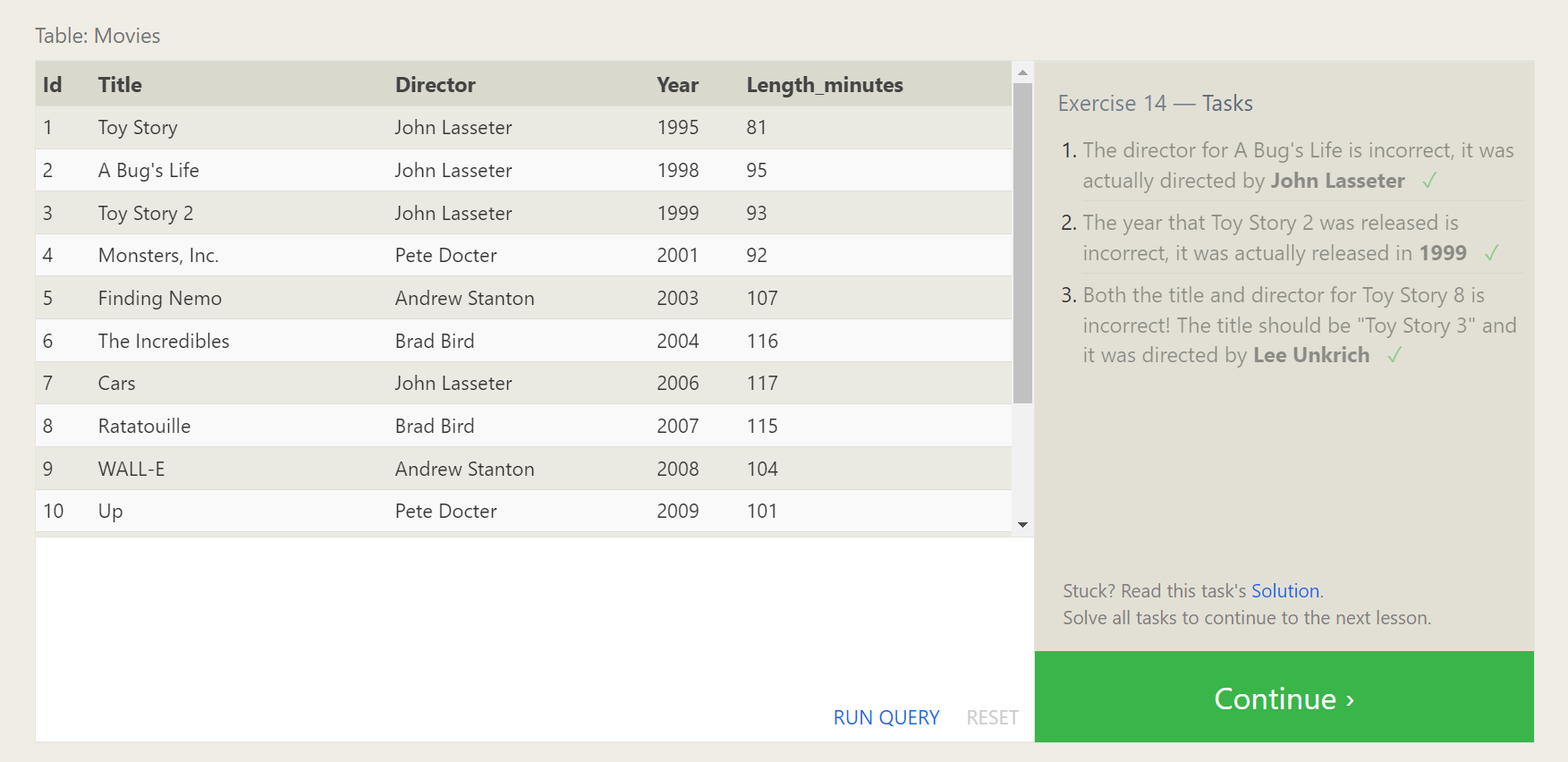
SQL Lesson 13: Inserting rows



Queries:

1. INSERT INTO movies VALUES (4, "Toy Story 4", "Pete Docter", 2015, 90);
2. INSERT INTO boxoffice VALUES (4, 8.7, 340000000, 270000000);

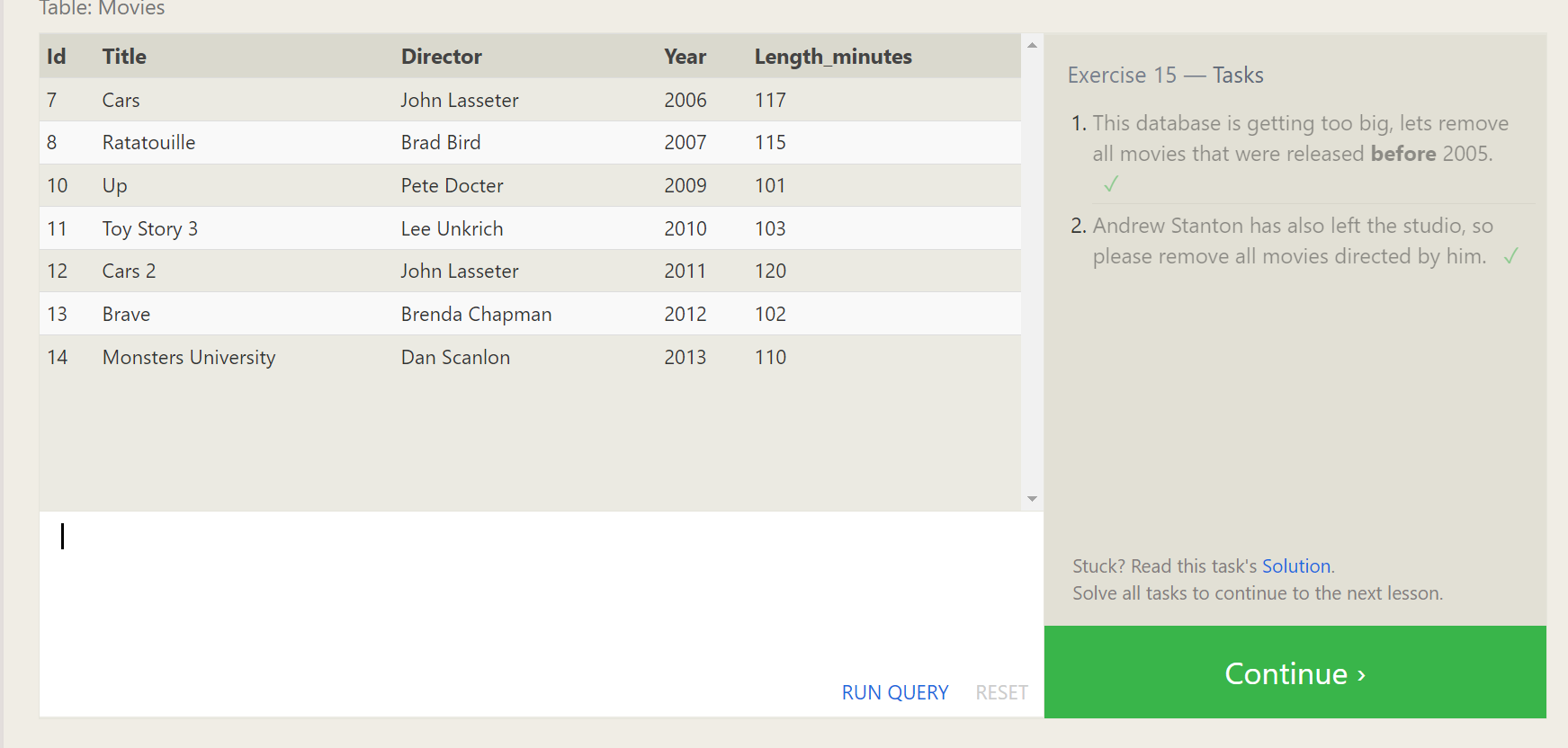
SQL Lesson 14: Updating rows



Queries:

1. update Movies set Director = 'John Lasseter' where Title = "A Bug's Life"
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"

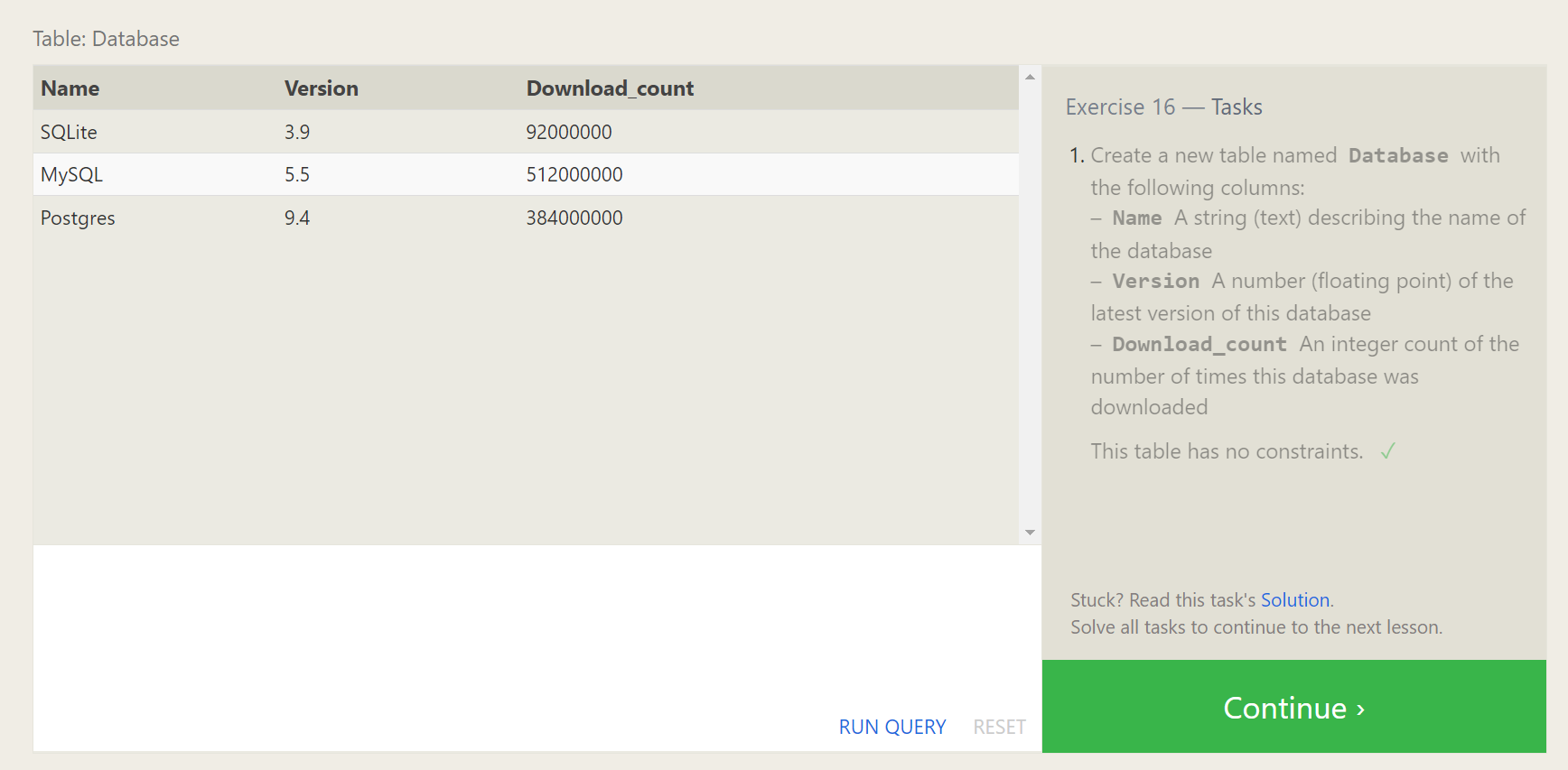
SQL Lesson 15: Deleting rows



Queries:

1. delete from Movies where year<2005;
2. delete from movies where director=”Andrew Santon”;

SQL Lesson 16: Creating tables



Queries:

1. create table Database (

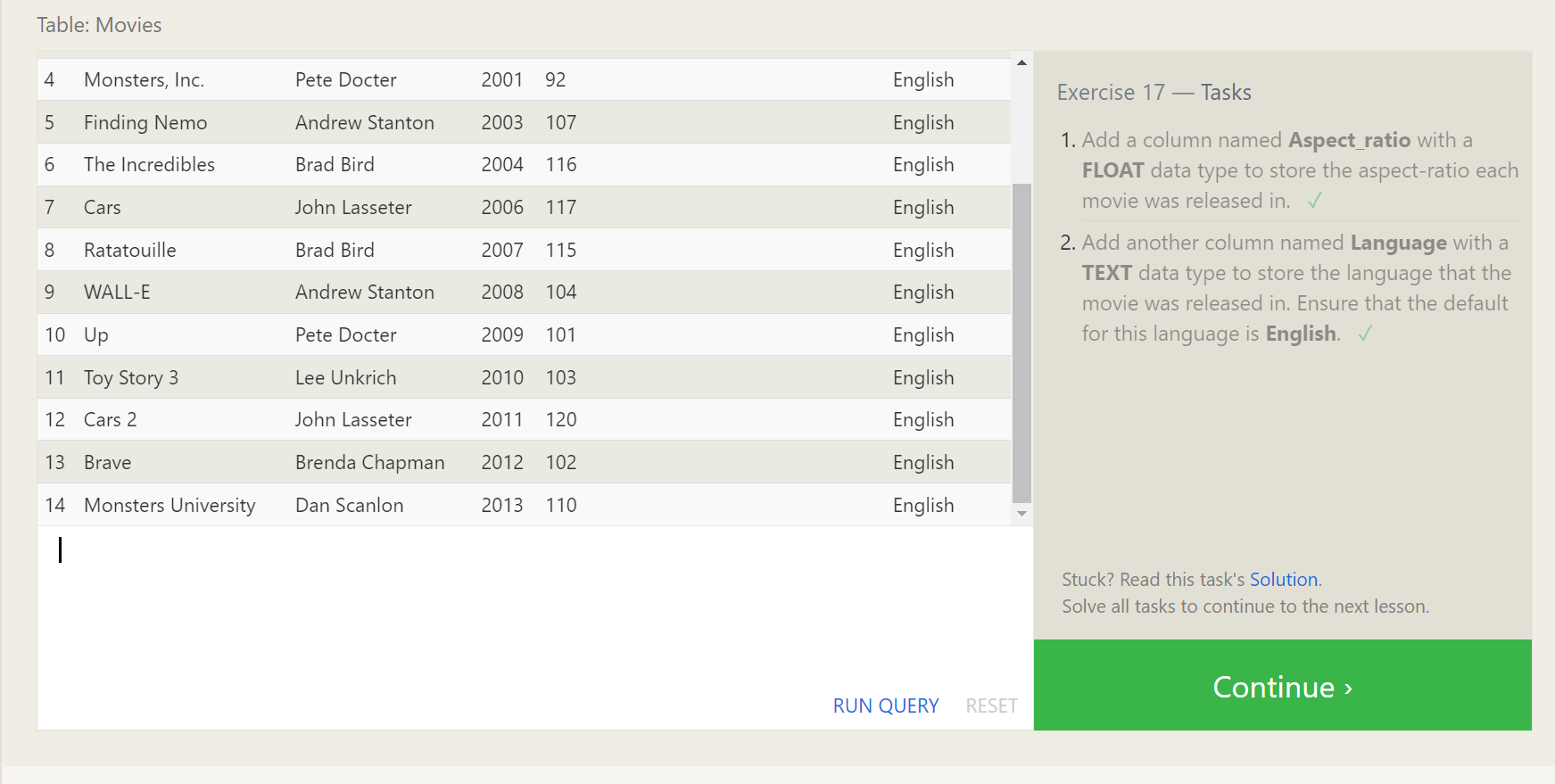
Name text,

Version float,

Download\_count int

)

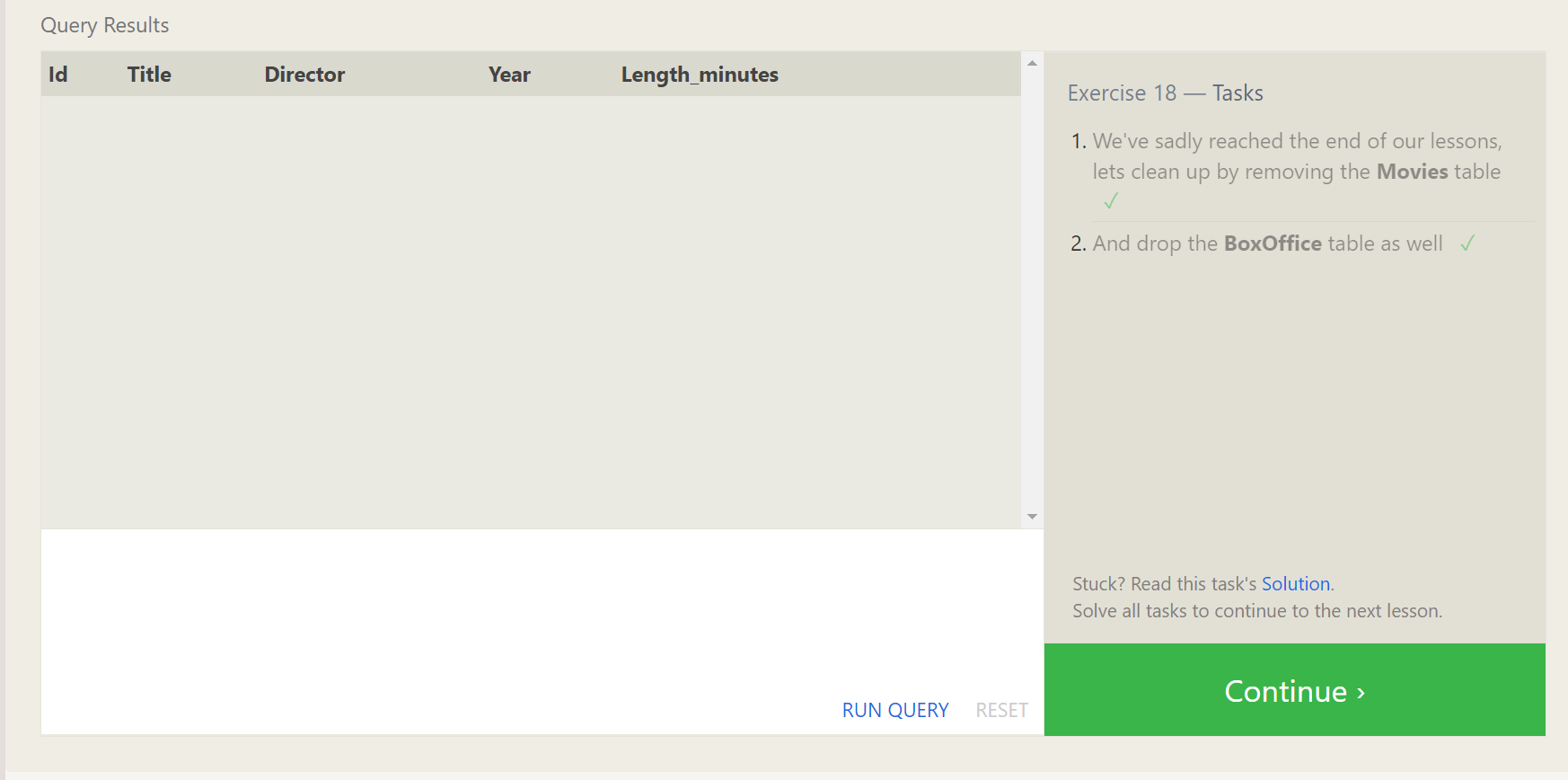
SQL Lesson 17: Altering tables



Queries:

1. alter table Movies add Aspect\_ratio float;
2. alter table Movies add Language text default English;

SQL Lesson 18: Dropping tables



Queries:

1. drop table if exists Movies;
2. drop table if exists BoxOffice;