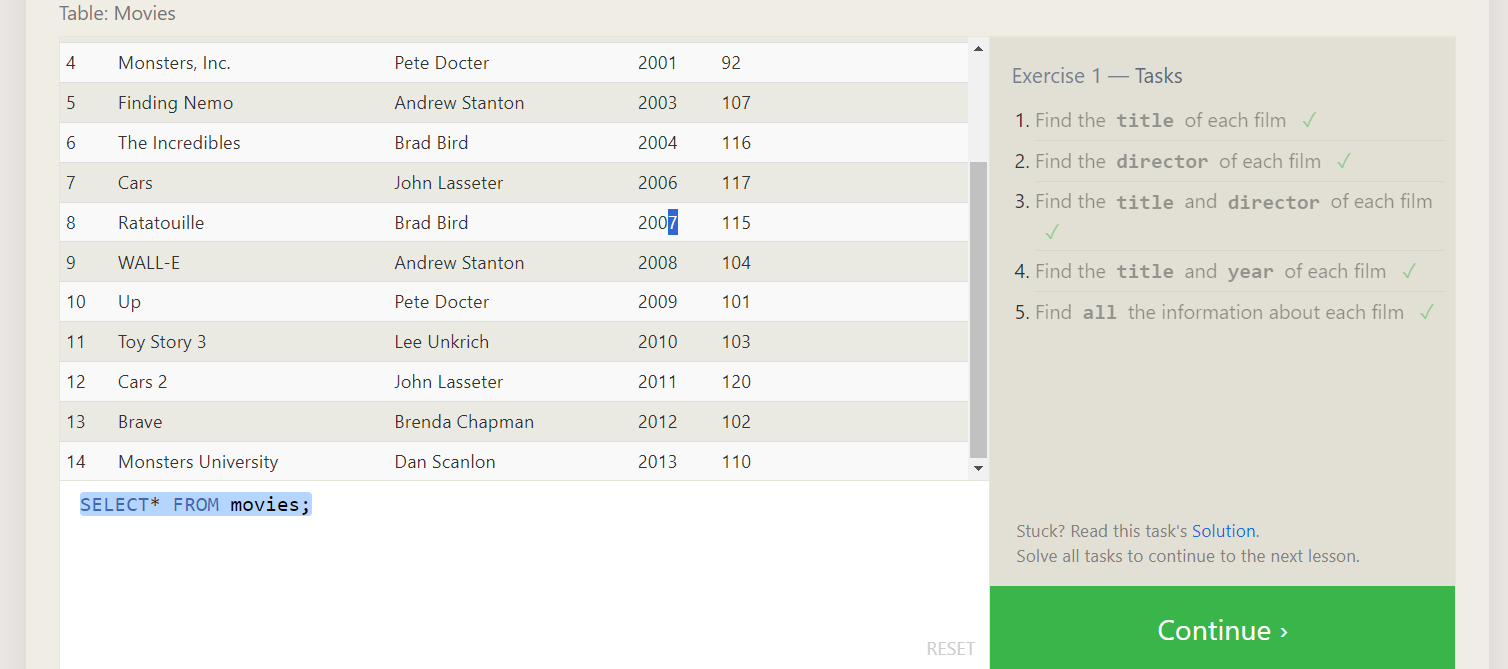
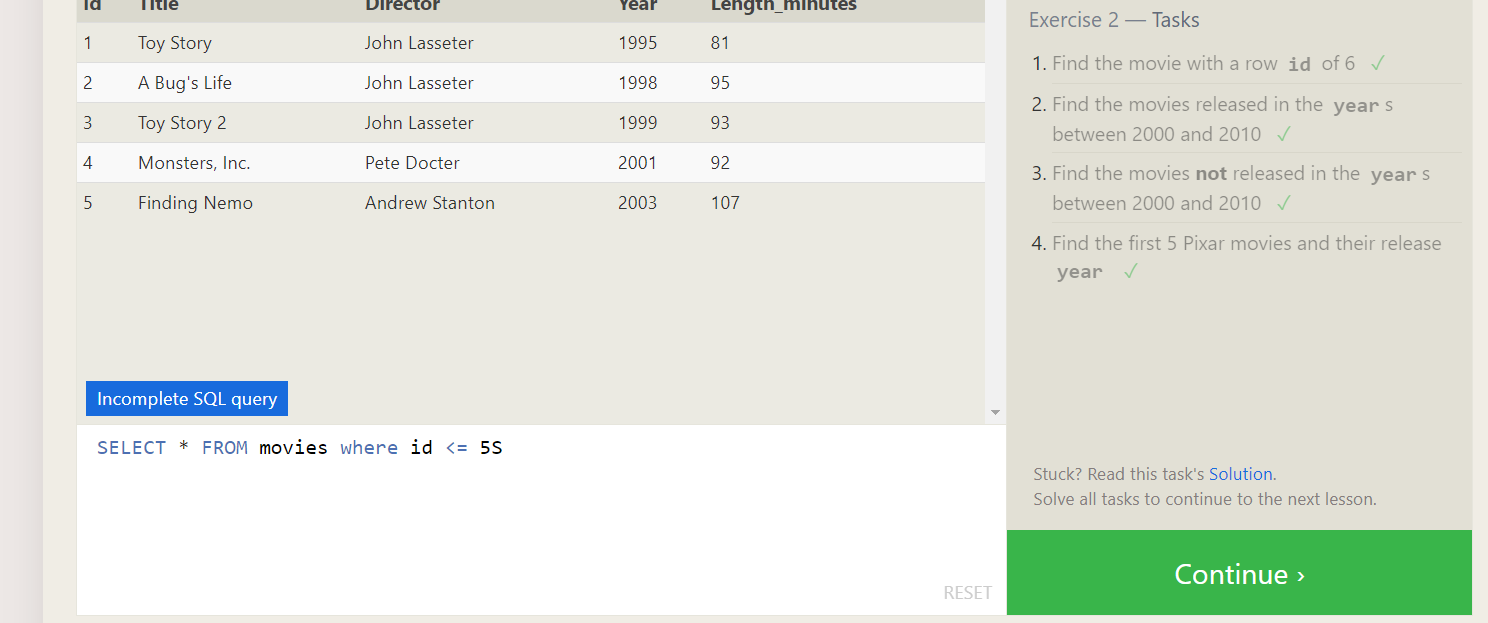
**SQL Lesson 1: SELECT queries 101**

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



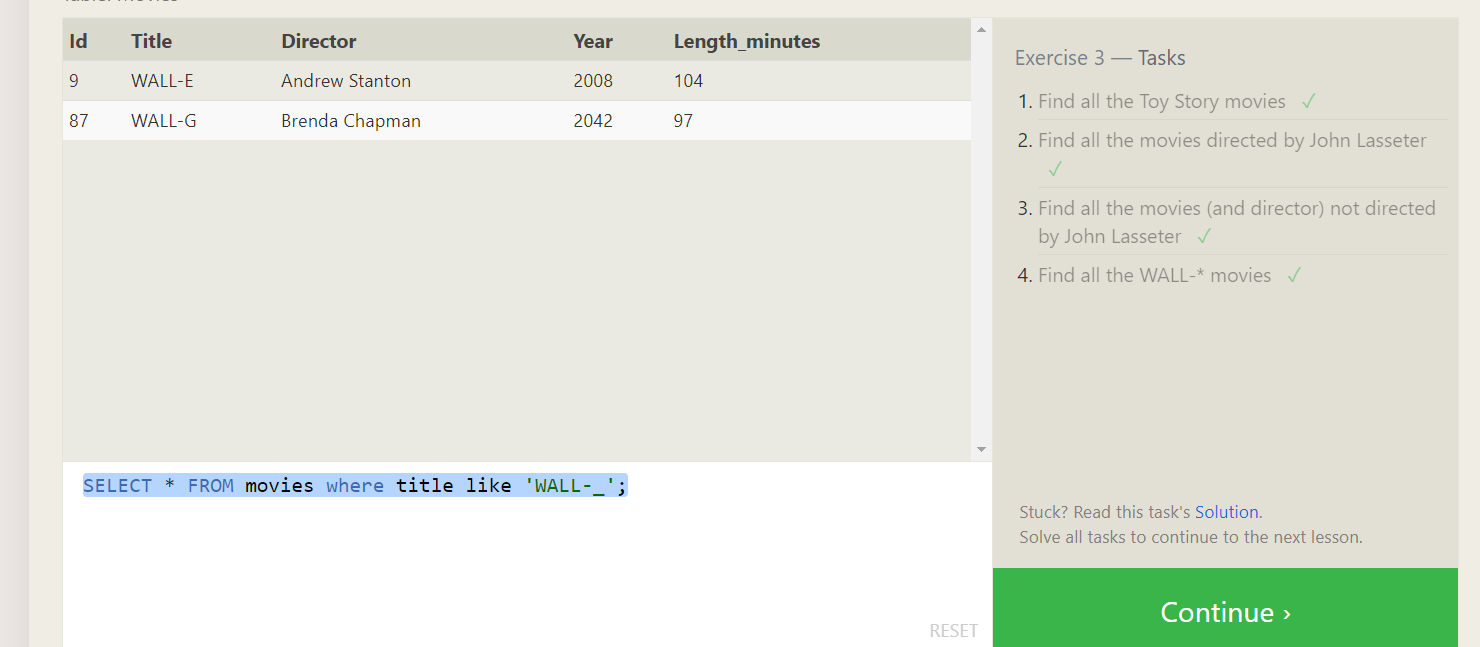
**SQL Lesson 2: Queries with constraints (Pt. 1)**

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 where id <= 5**



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

1. **SELECT \* FROM movies where title like '%Toy Story%';**
2. **SELECT \* FROM movies where director like 'johny%';**
3. **SELECT \* FROM movies where director not like 'John%';**
4. **SELECT \* FROM movies where title like 'WALL-\_';**



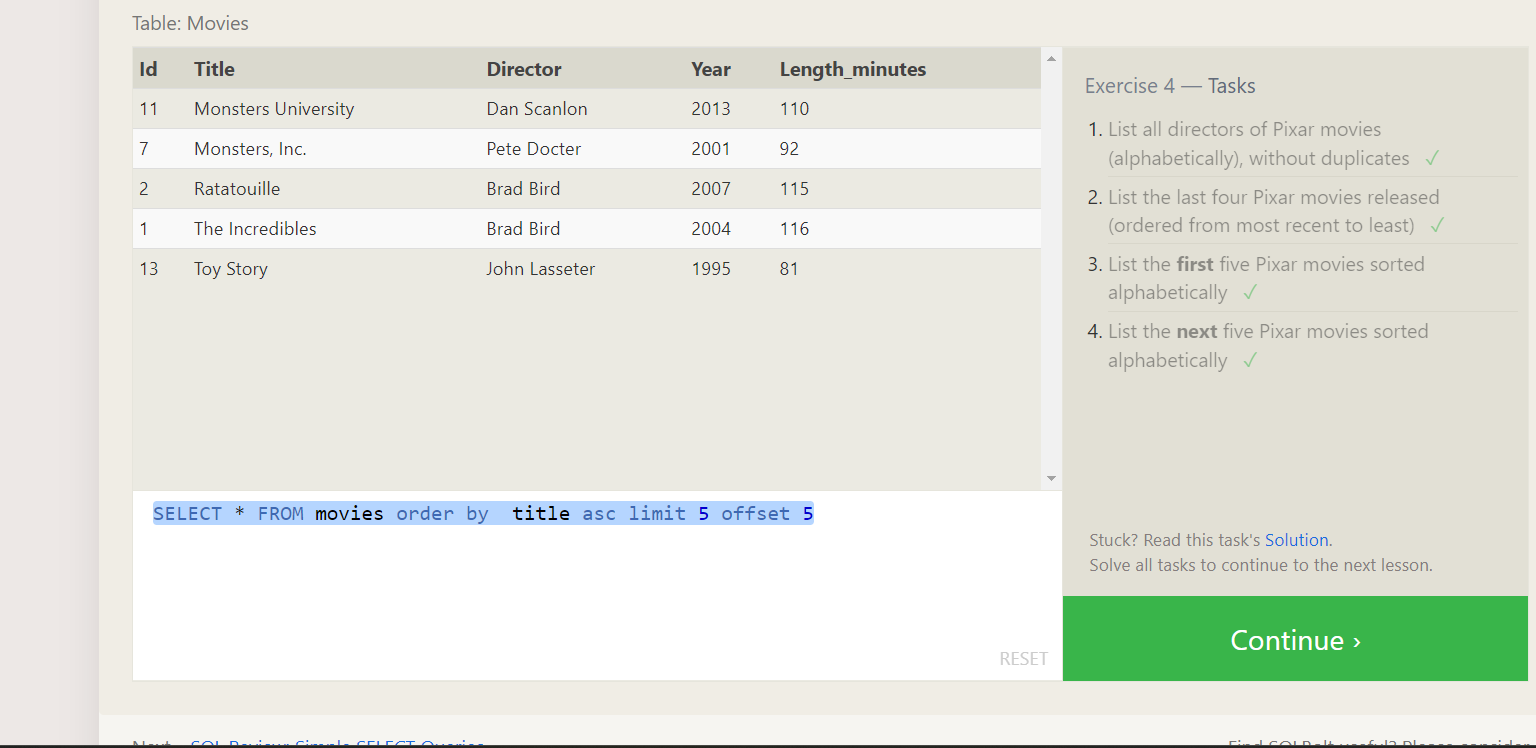
**SQL Lesson 4: Filtering and sorting Query results**

**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 asc limit 5**

**4.** **SELECT \* FROM movies order by title asc limit 5 offset 5**

****

**SQL Review: Simple SELECT Queries**

**1.** **SELECT city, population**

**FROM north\_american\_cities**

**WHERE country = "Canada";**

**2.** **SELECT city**

**FROM north\_american\_cities**

**WHERE country = "United States"**

**ORDER BY latitude DESC;**

**3.** **SELECT city**

**FROM north\_american\_cities**

**WHERE longitude < -87.629798**

**ORDER BY longitude;**

**4.** **SELECT city**

**FROM north\_american\_cities**

**WHERE longitude < -87.629798**

**ORDER BY longitude;**

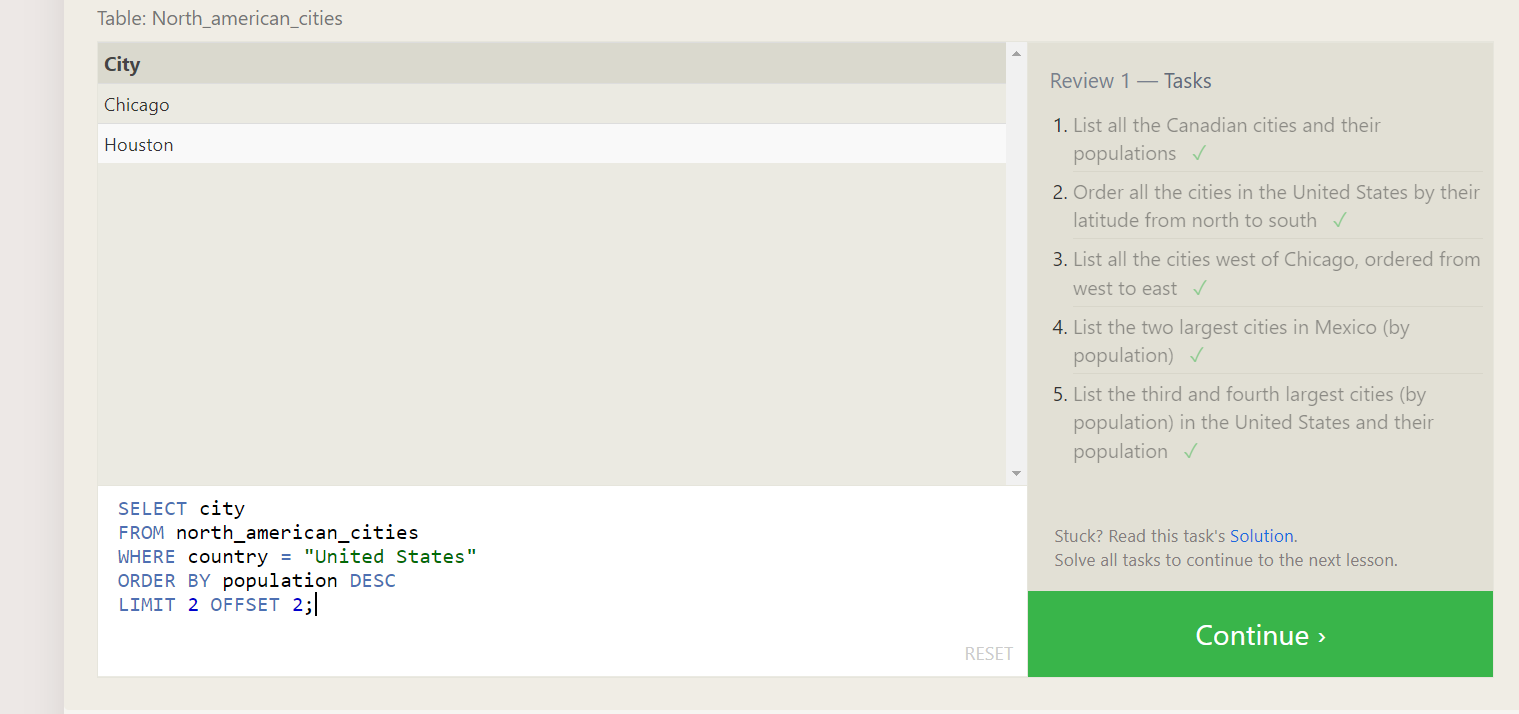
**5.** **SELECT city**

**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.** **SELECT title, domestic\_sales, international\_sales**

**FROM movies**

**2.** **SELECT title, domestic\_sales, international\_sales**

**FROM movies**

**INNER JOIN boxoffice**

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

**WHERE international\_sales > domestic\_sales;**

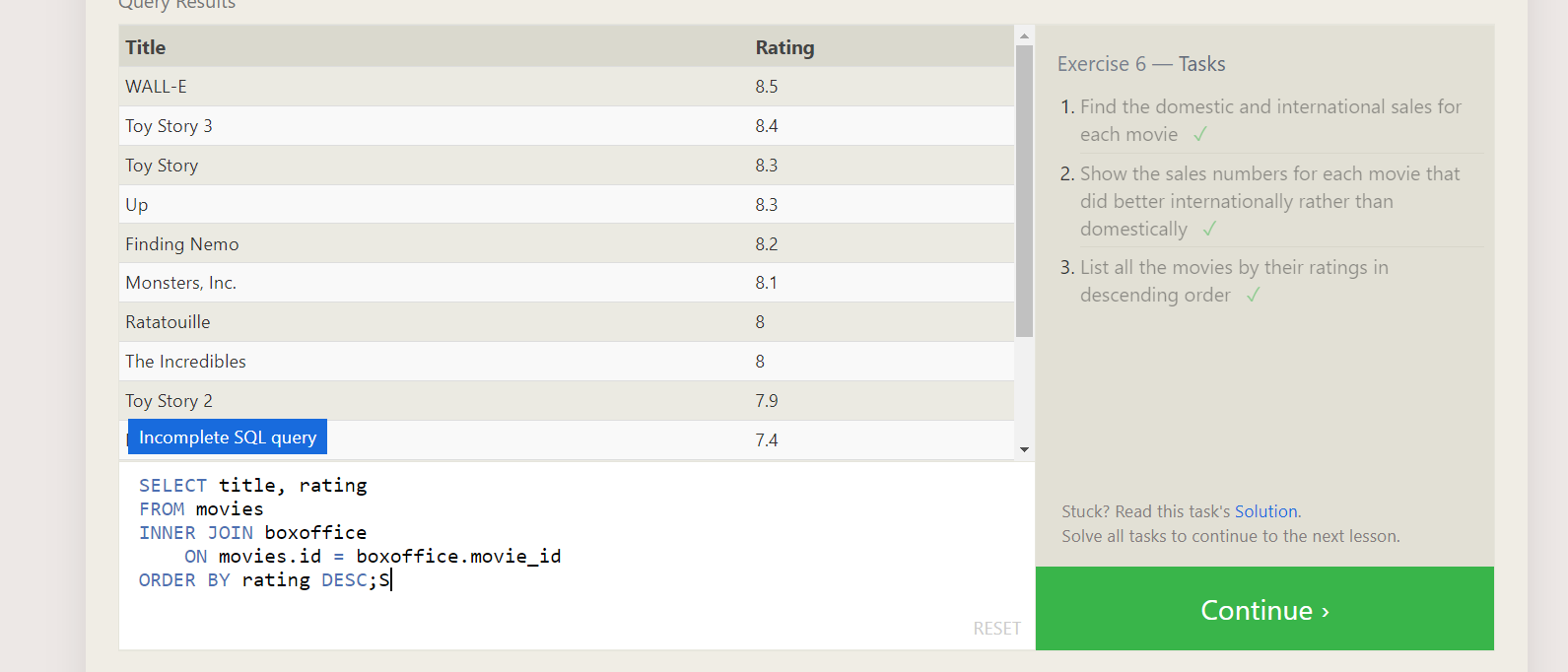
**3.** **SELECT title, rating**

**FROM movies**

**INNER JOIN boxoffice**

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

**ORDER BY rating DESC;**

****

**SQL Lesson 7: OUTER JOINs**

**1.** **SELECT DISTINCT building FROM employees;**

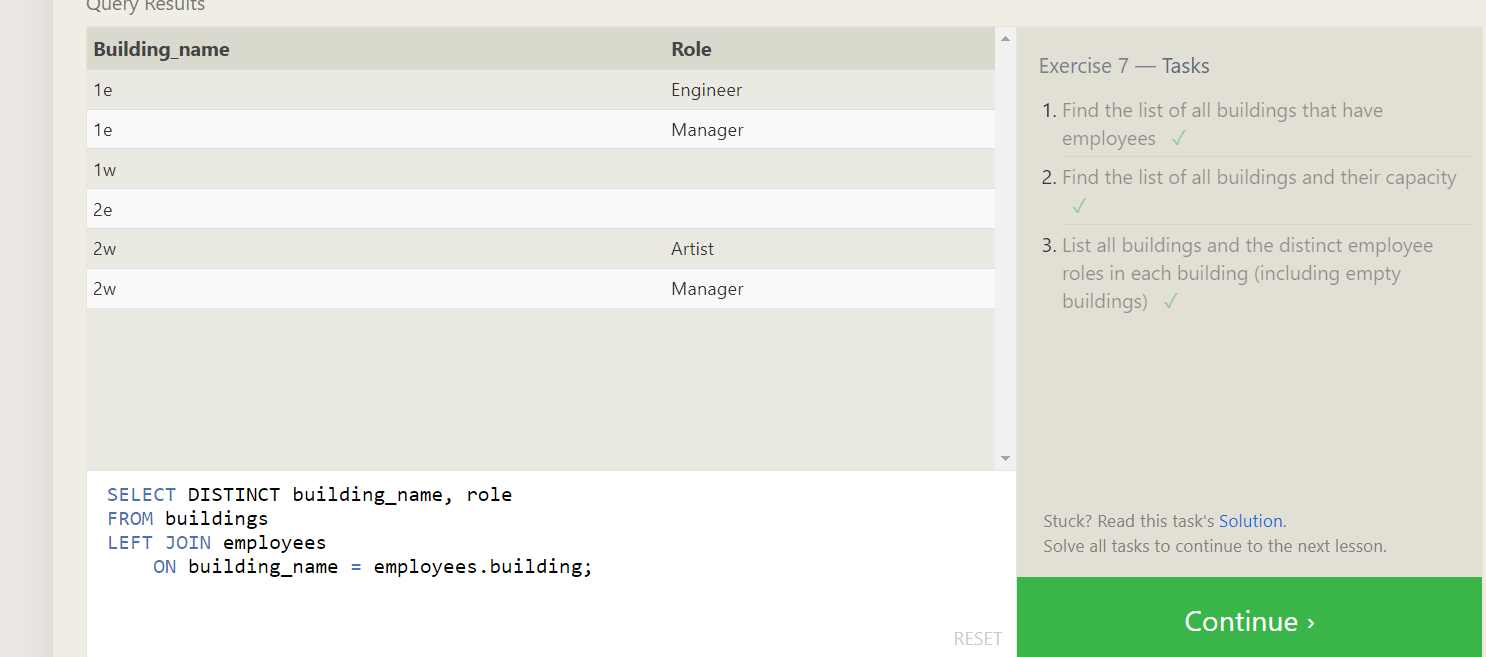
**2.** **SELECT \* FROM buildings;**

**3.** **SELECT DISTINCT building\_name, role**

**FROM buildings**

**LEFT JOIN employees**

**ON building\_name = employees.building;**

****

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

1. **SELECT name, role FROM employees WHERE building IS NULL;**

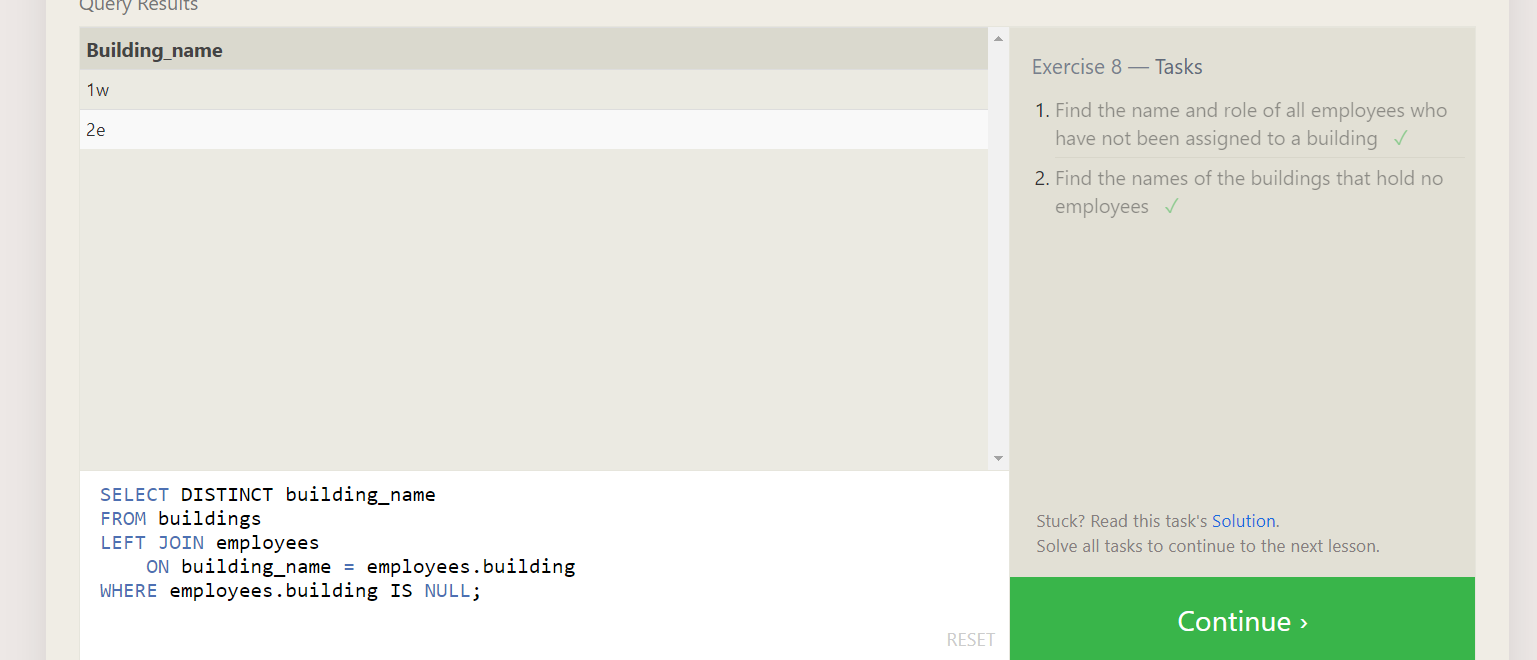
**2. SELECT DISTINCT building\_name**

**FROM buildings**

**LEFT JOIN employees**

**ON building\_name = employees.building**

**WHERE employees.building IS NULL;**

****

**SQL Lesson 9: Queries with expressions**

**1.** **SELECT DISTINCT**

**title,**

**(domestic\_sales + international\_sales) / 1000000 AS sales**

**FROM movies**

**INNER JOIN boxoffice**

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

**2.** **SELECT DISTINCT**

**title,**

**(rating \* 10) AS rate\_percent**

**FROM movies**

**INNER JOIN boxoffice**

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

**3.** **SELECT title FROM movies WHERE year % 2 = 0;**

****

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

**1.SELECT MAX(years\_employed) AS maximum\_years\_employed**

**FROM employees;**

**2.** **SELECT role,**

**AVG(years\_employed) AS average\_number\_of\_years**

**FROM employees**

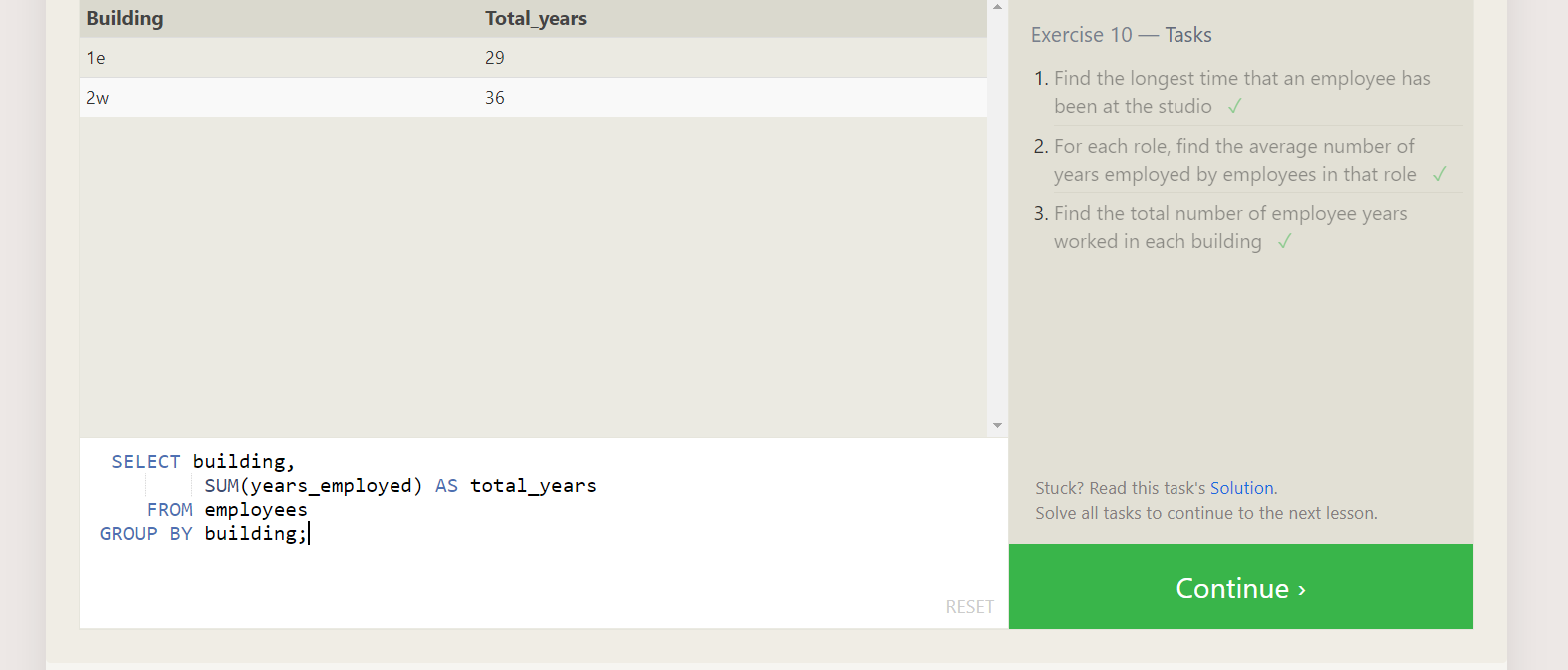
**GROUP BY role;**

**3. SELECT building,**

**SUM(years\_employed) AS total\_years**

**FROM employees**

**GROUP BY building;**

****

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

**1** **SELECT COUNT(name) AS number\_of\_artists**

**FROM employees**

**WHERE role = 'Artist';**

**2 SELECT role,**

**COUNT(name) AS employees**

**FROM employees**

**GROUP by role;**

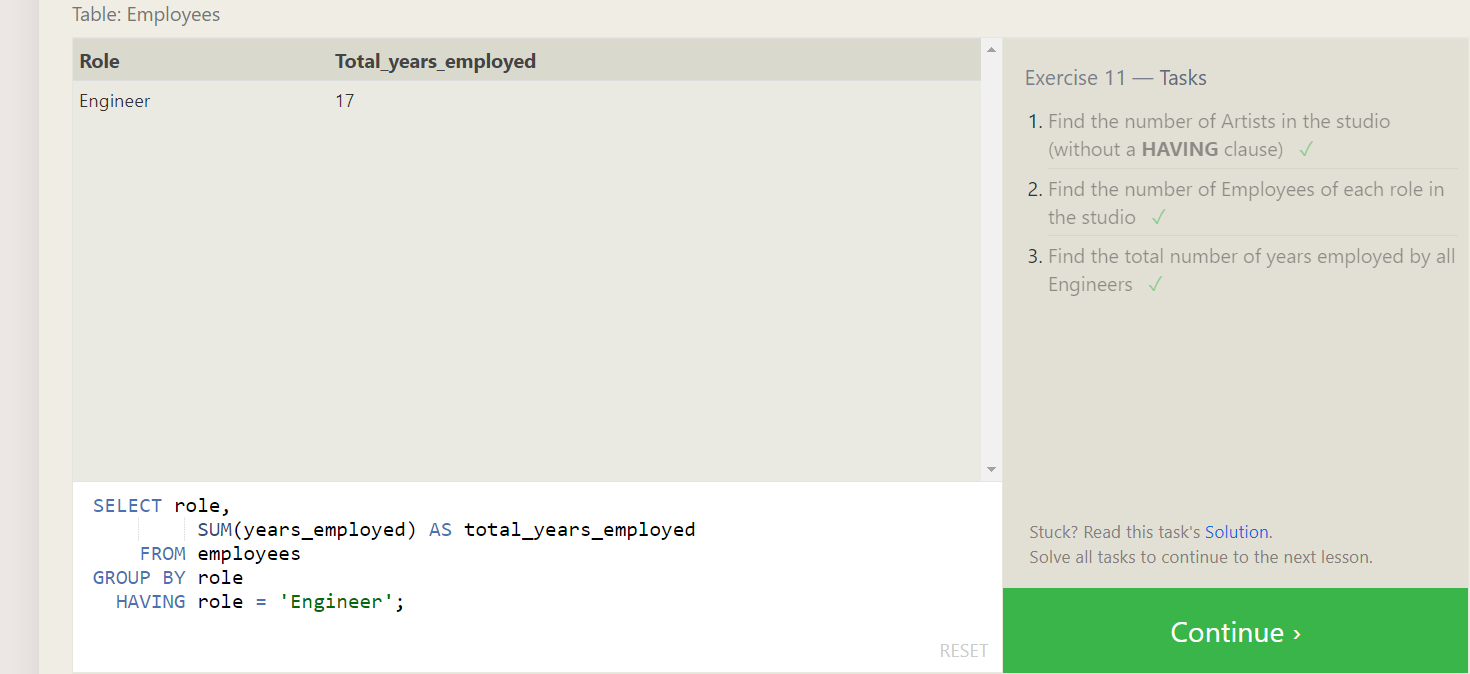
**3. SELECT role,**

**SUM(years\_employed) AS total\_years\_employed**

**FROM employees**

**GROUP BY role**

**HAVING role = 'Engineer';**

****

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

**1.** **SELECT director,**

**COUNT(title) AS number\_of\_movies**

**FROM movies**

**GROUP BY director;**

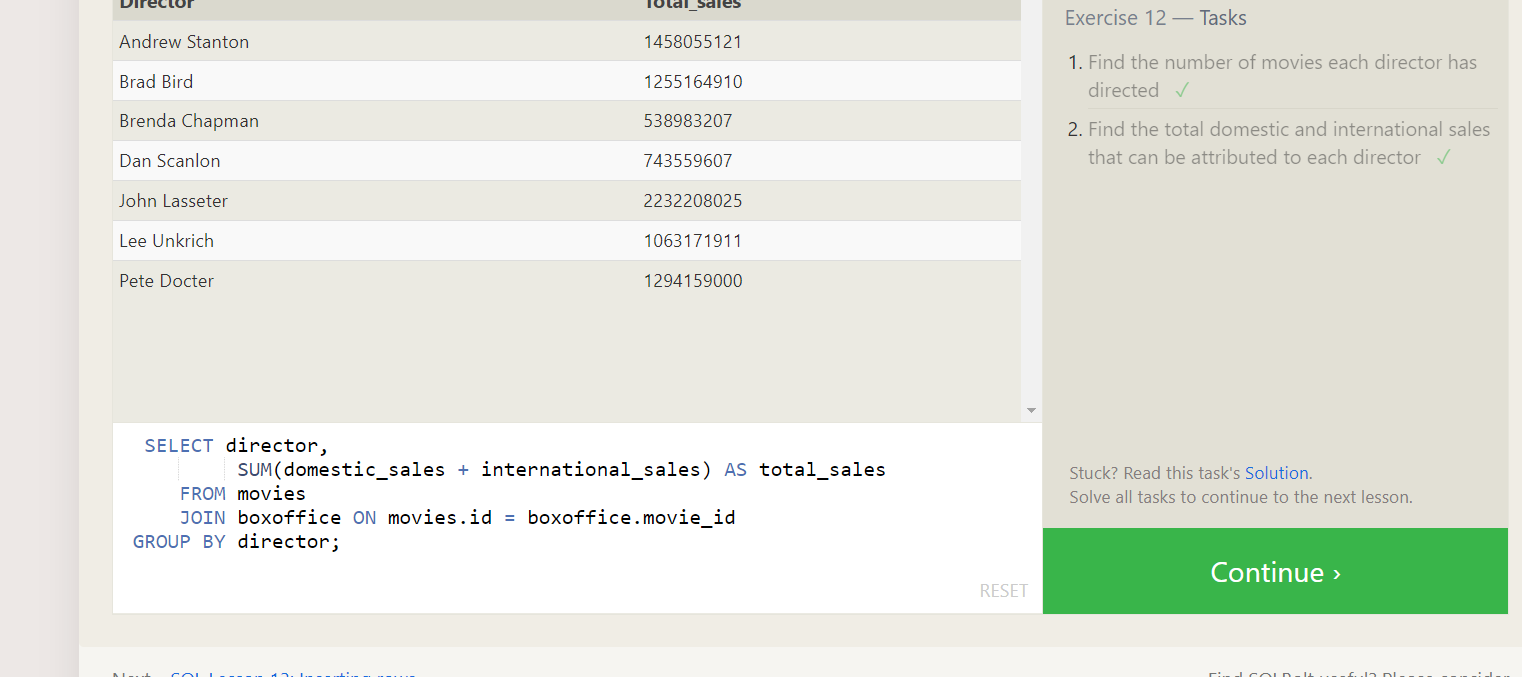
**2. SELECT director,**

**SUM(domestic\_sales + international\_sales) AS total\_sales**

**FROM movies**

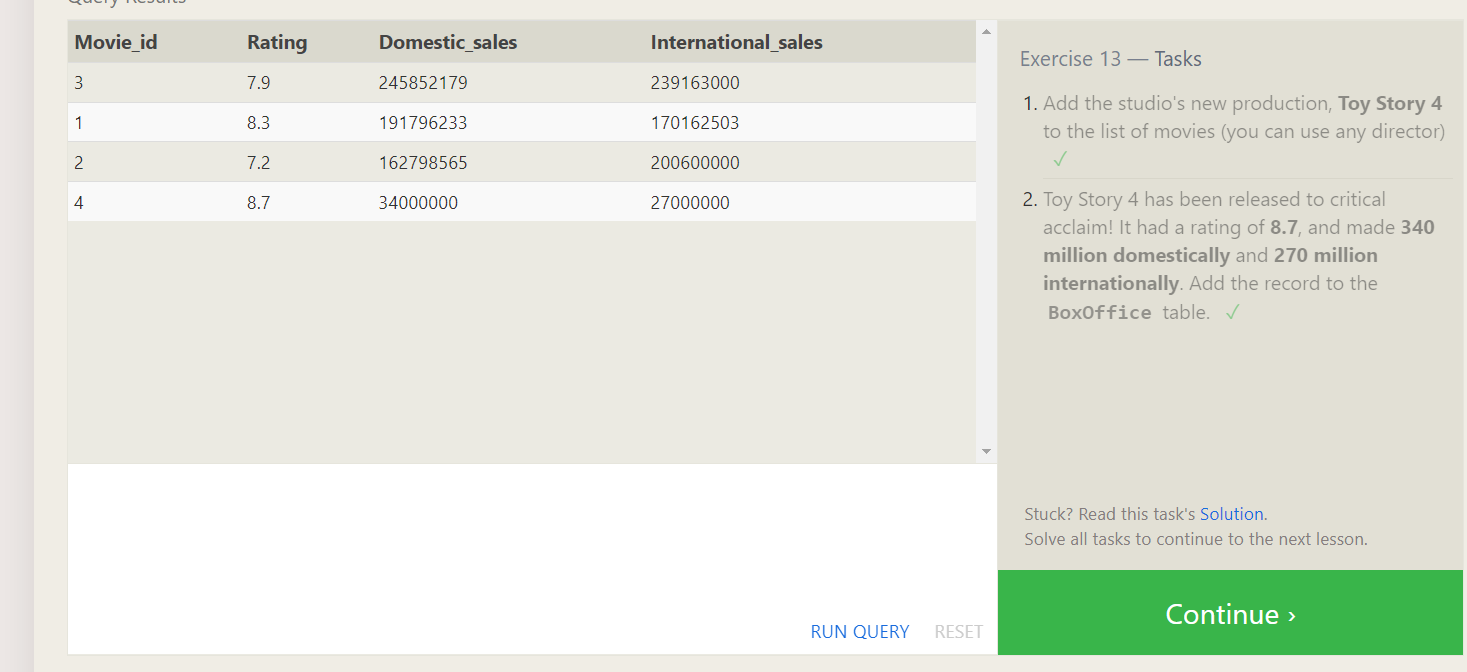
**JOIN boxoffice ON movies.id = boxoffice.movie\_id**

**GROUP BY director;**

****

**SQL Lesson 13: Inserting rows**

**1.insert into movies values(4,’Toy story 4’,’** **John Lasseter’,2024,78)**

**2.** insert into Boxoffice(4,8.7,34000000,27000000)****

**SQL Lesson 14: Updating rows**

1. update movies

set director = 'John Lasseter'

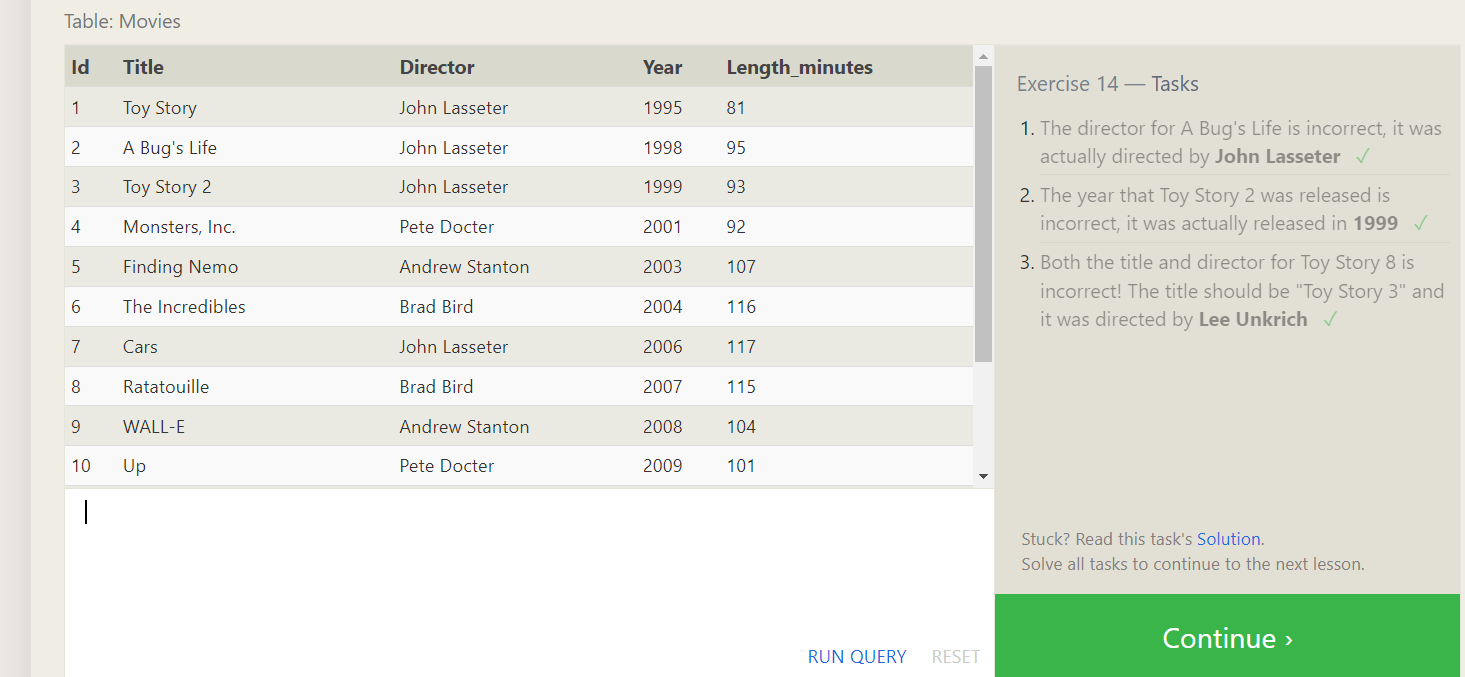
where id = 2;

2. update movies

set year= 1999

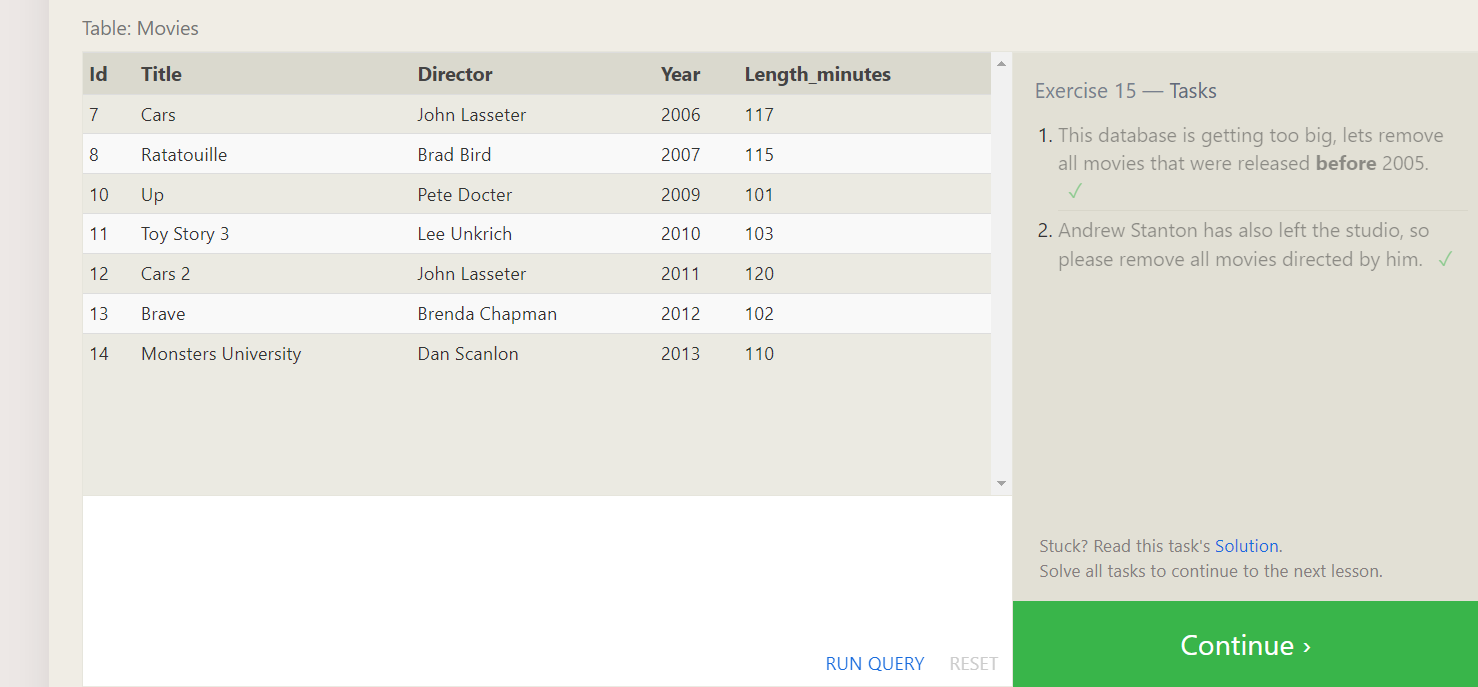
where id = 3; 3. Update movies set director =’ **Lee Unkrich’,title= "Toy Story 3"**

**id=11;**



**SQL Lesson 15: Deleting rows**

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



**SQL Lesson 16: Creating tables**

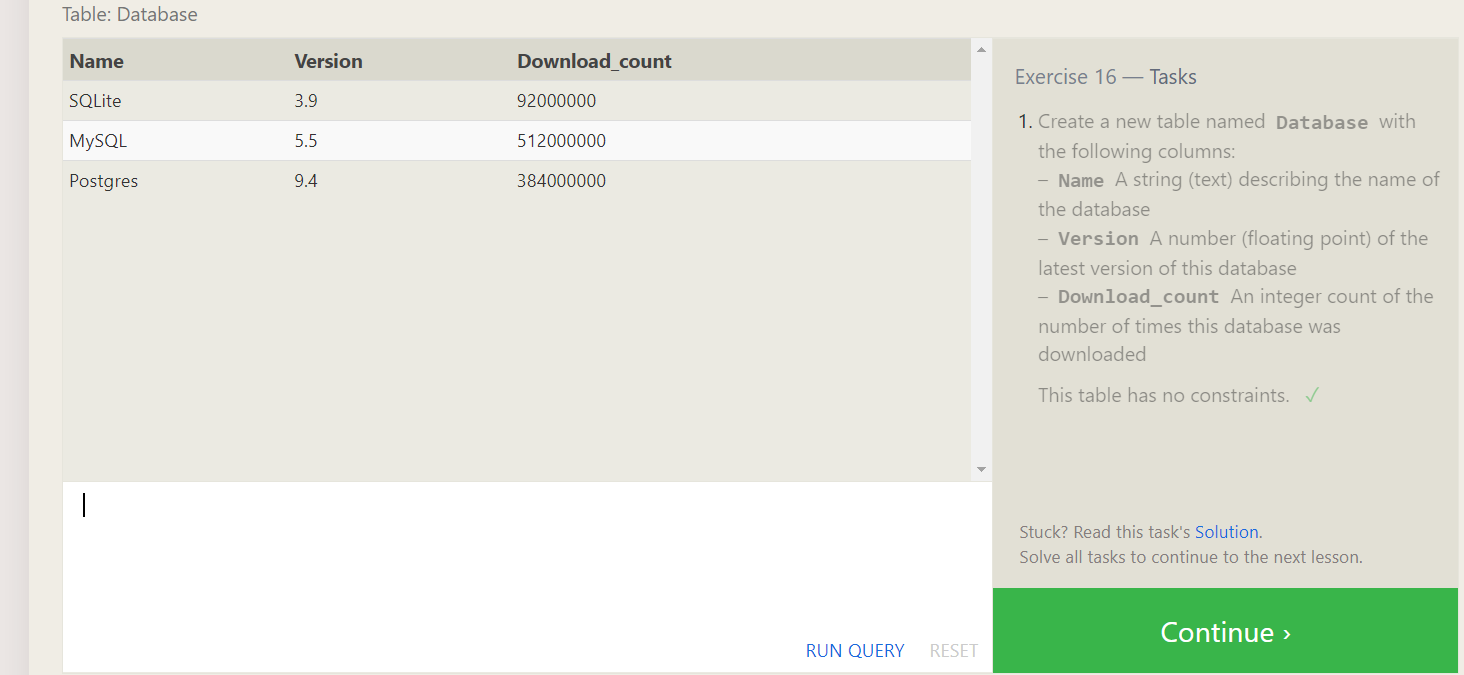
**1.** **create table database (**

**name varchar(255),**

**version decimal,**

**Download\_count int**

**)**

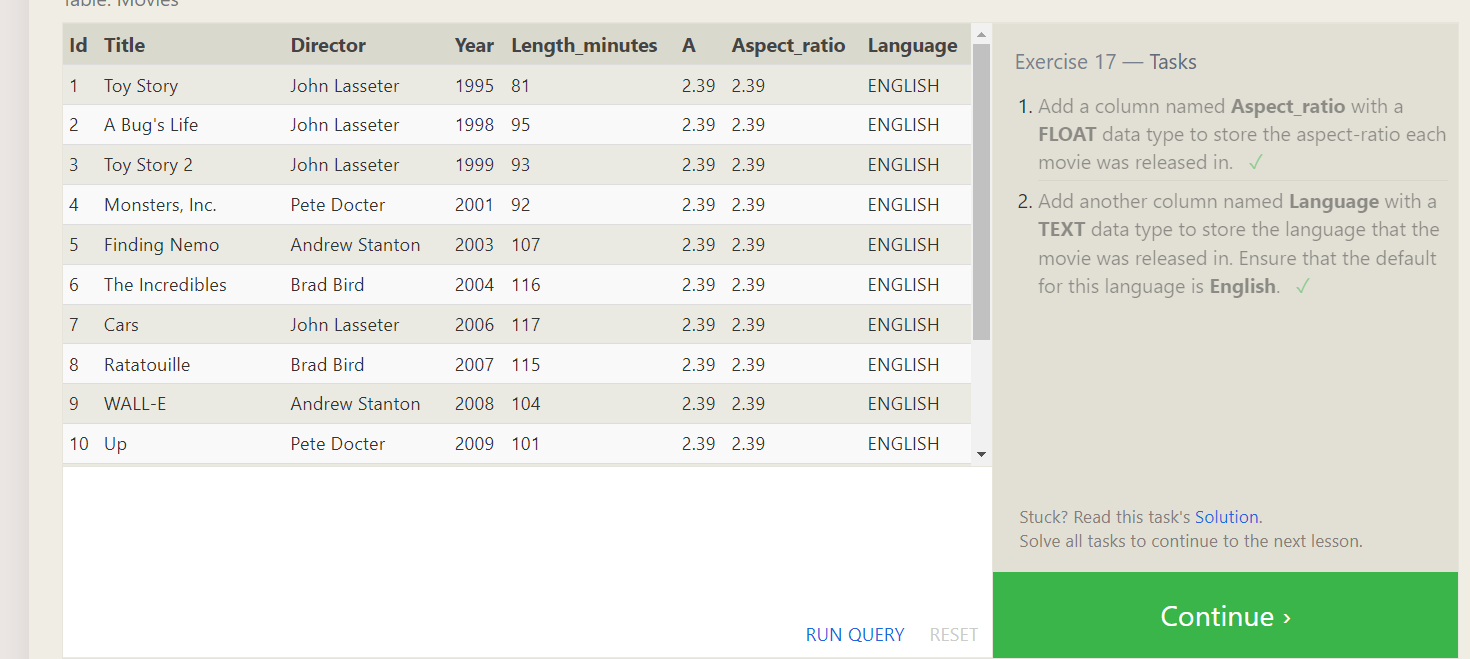


**SQL Lesson 17: Altering tables**

**1.** **ALTER TABLE movies**

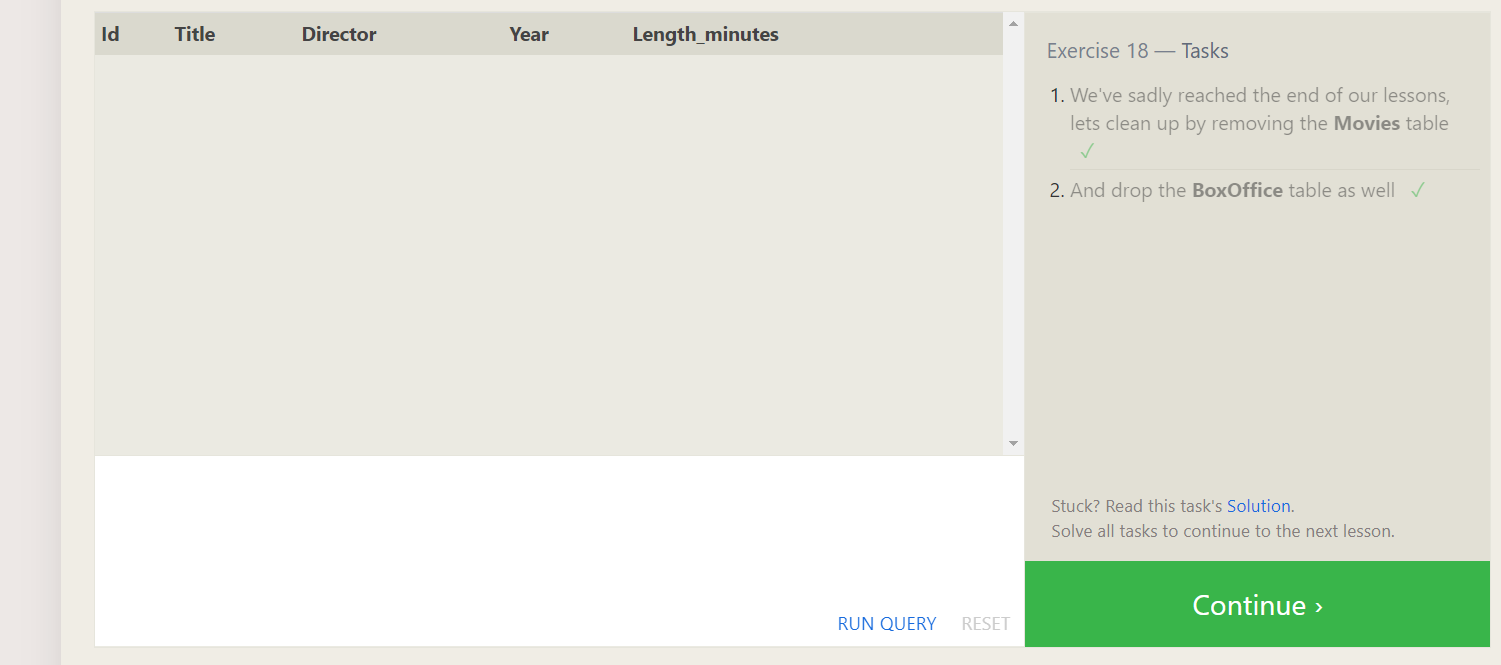
**ADD COLUMN Aspect\_ratio FLOAT default 2.39;**

**2.alter table movies add language varchar(255) default ,ENGLISH;**

****

**SQL Lesson 18: Dropping tables**

1. **drop table movies**
2. **drop table BoxOffice**

****