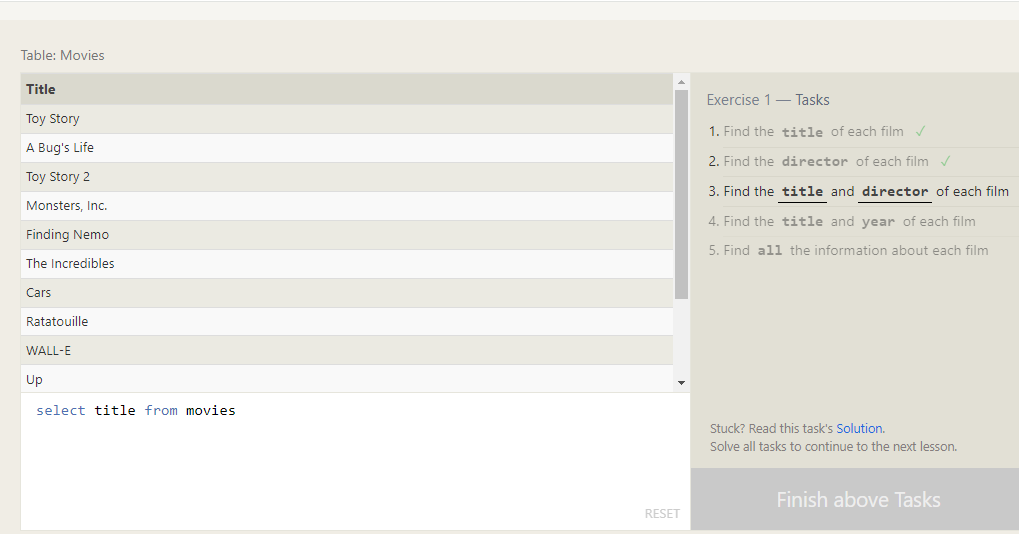
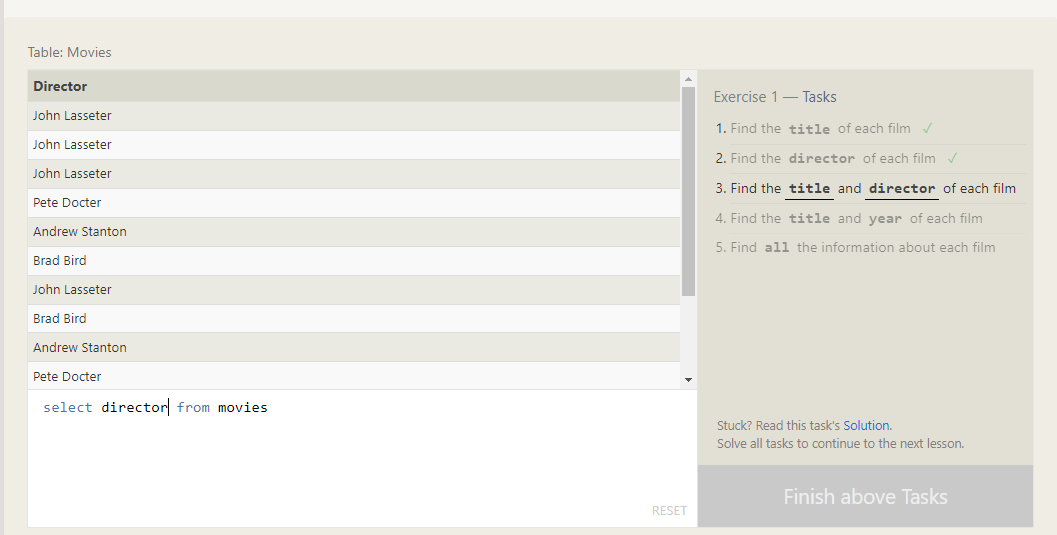
DATA BASE TASK – 1

**Excersice-1 Task:-**

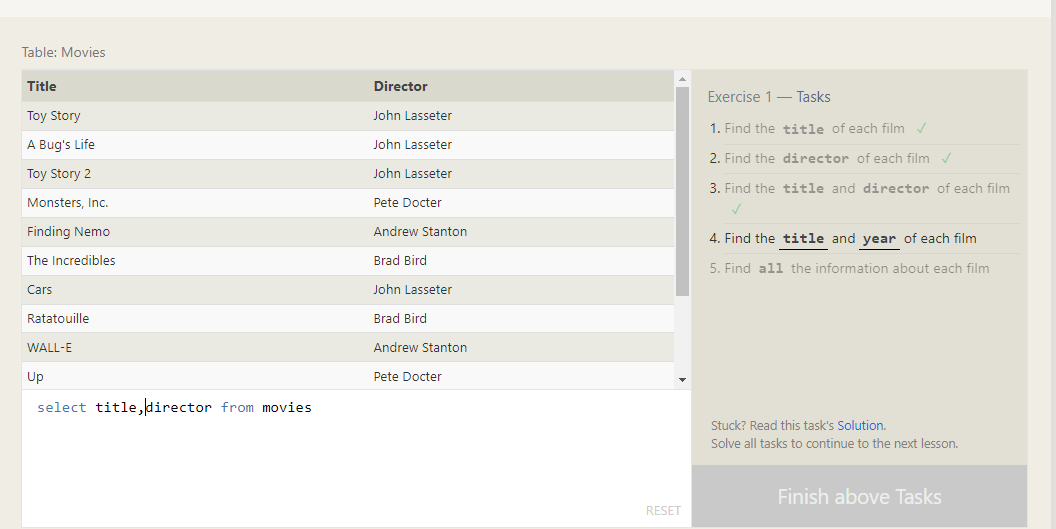
a) Query: select title from movies;



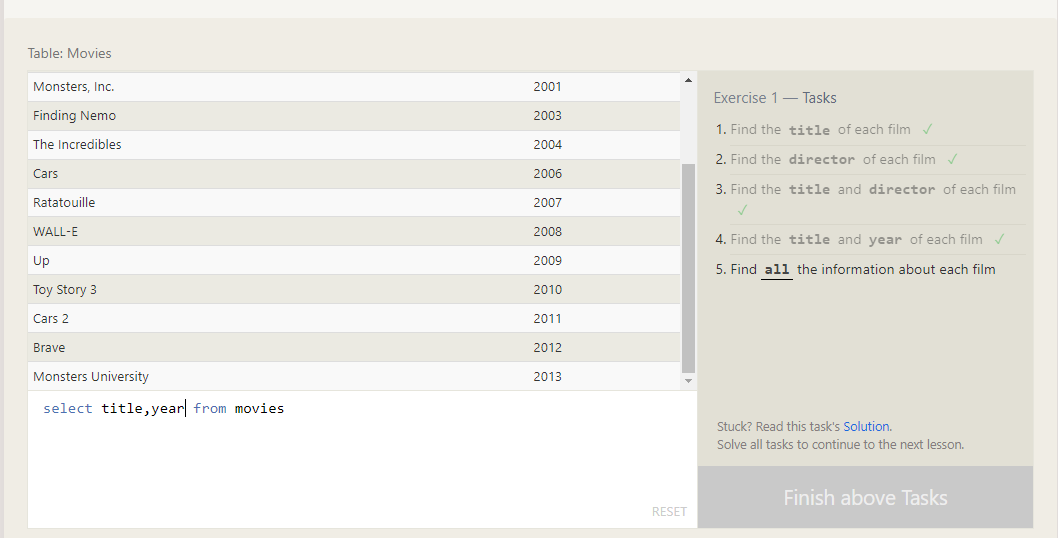
b) Query: select director from movies;



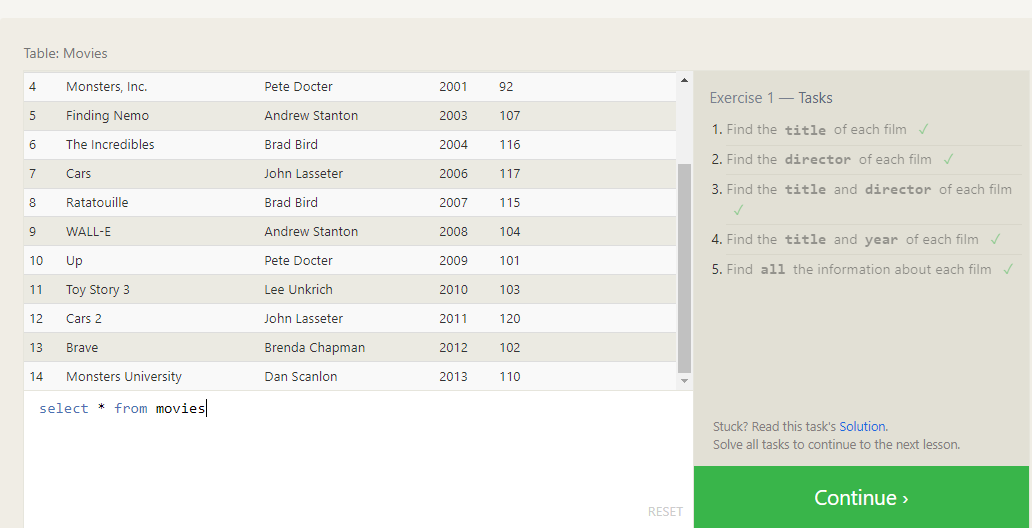
c) Query: select title,director from movies;



d) Query: select title,year from movies;



e) Query: select \* from movies;

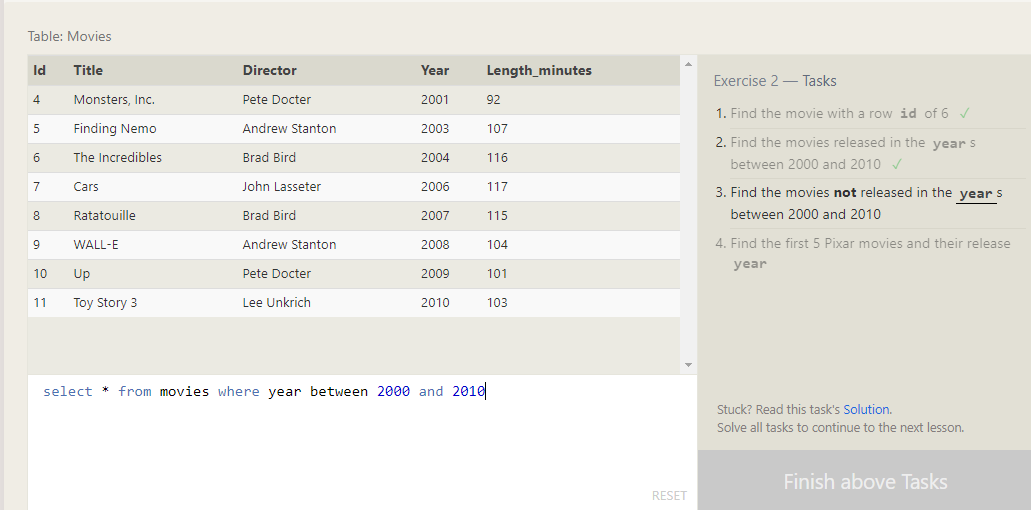


**Excersice-2 Task**

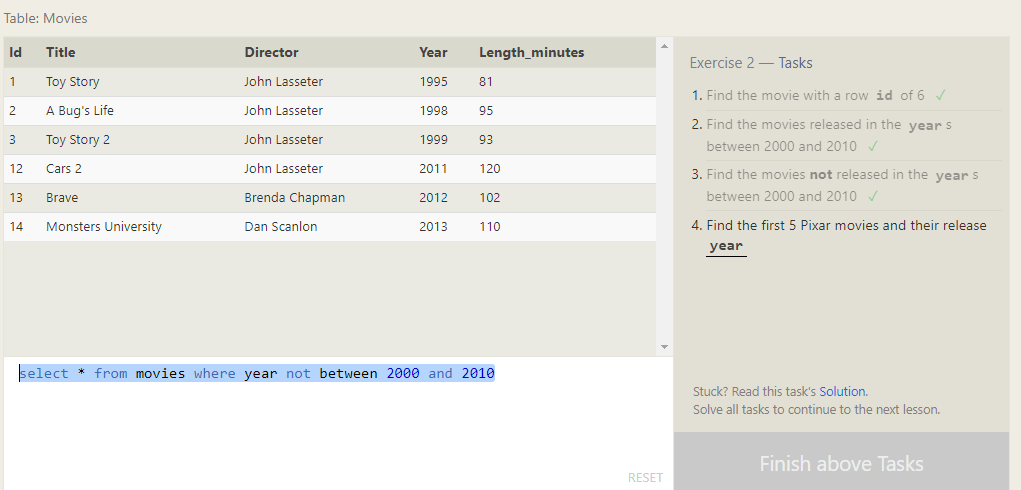
a) Query: select \* from movies where id=6;



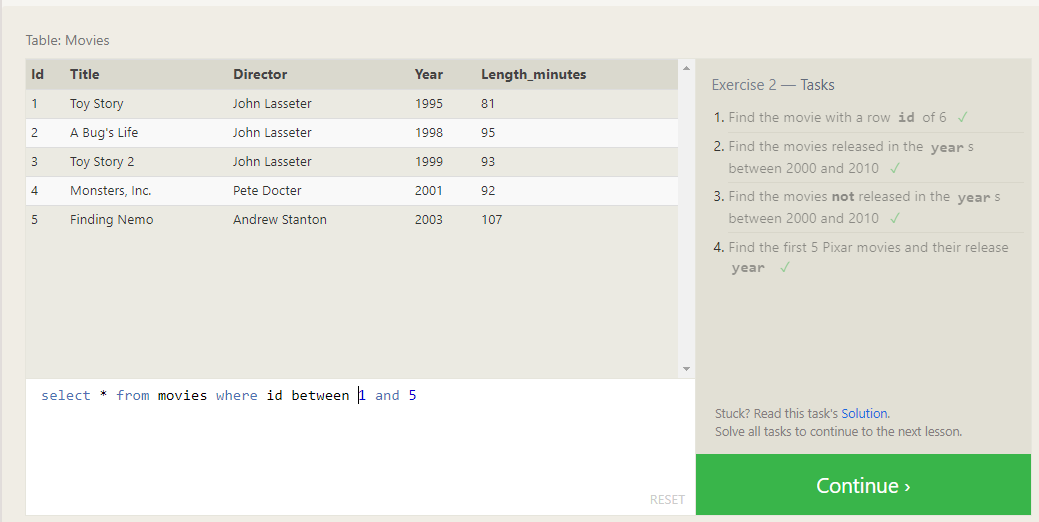
b) Query: select \* from movies where between 2000 and 2010;



c) Query: select \* from movies where year not between 2000 and 2010;



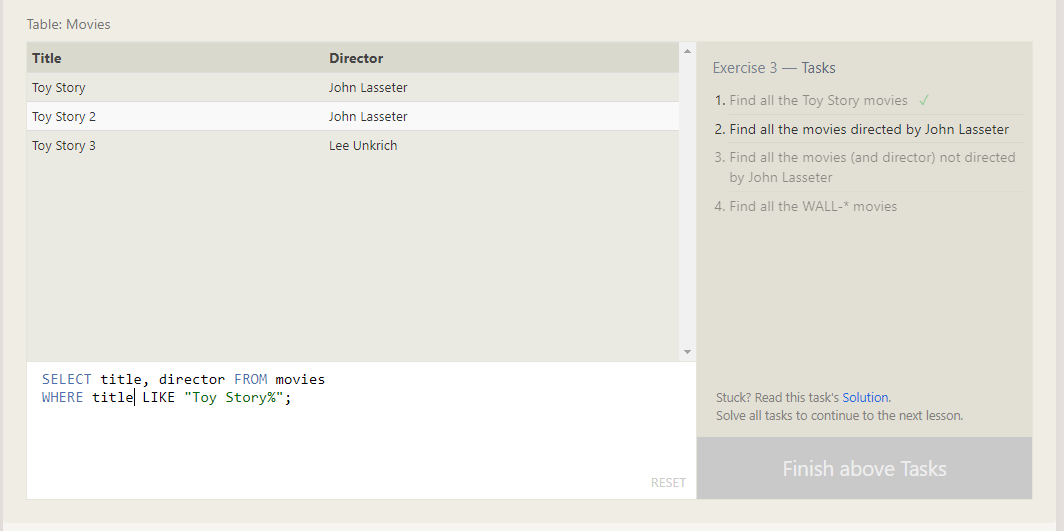
d) Query: select \* from movies where id between 1 and 5;



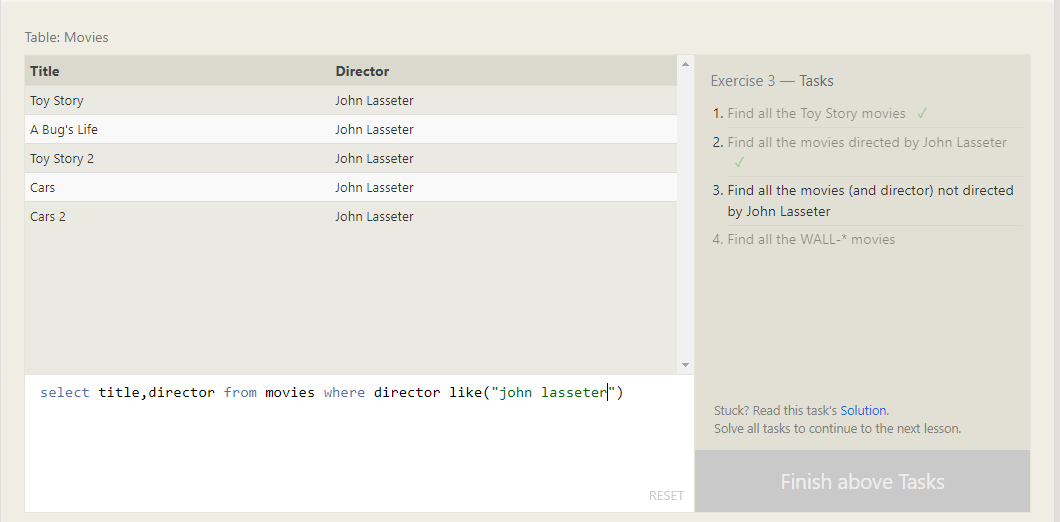
**Excersice-3 Task**

a) Query: SELECT title, director FROM movies

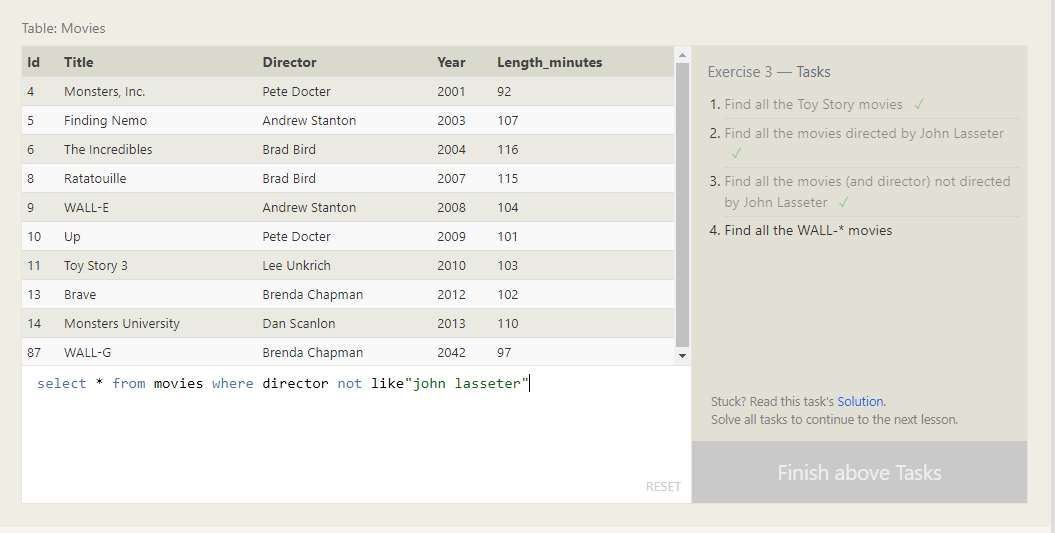
WHERE title LIKE "Toy Story%";



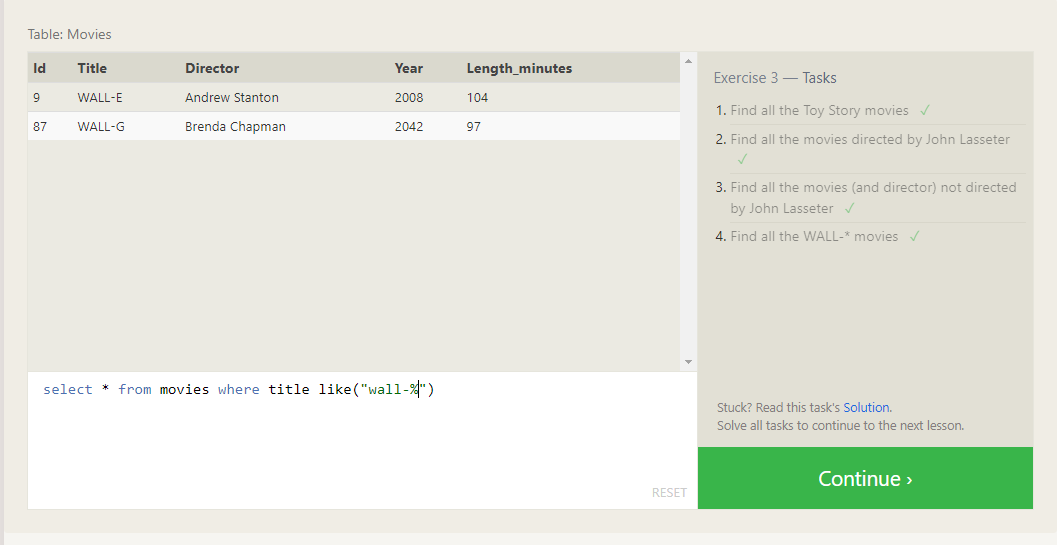
b) Query: select title,director from movies where director like("john lasseter");



c) Query: select \* from movies where director not like "john lasseter";

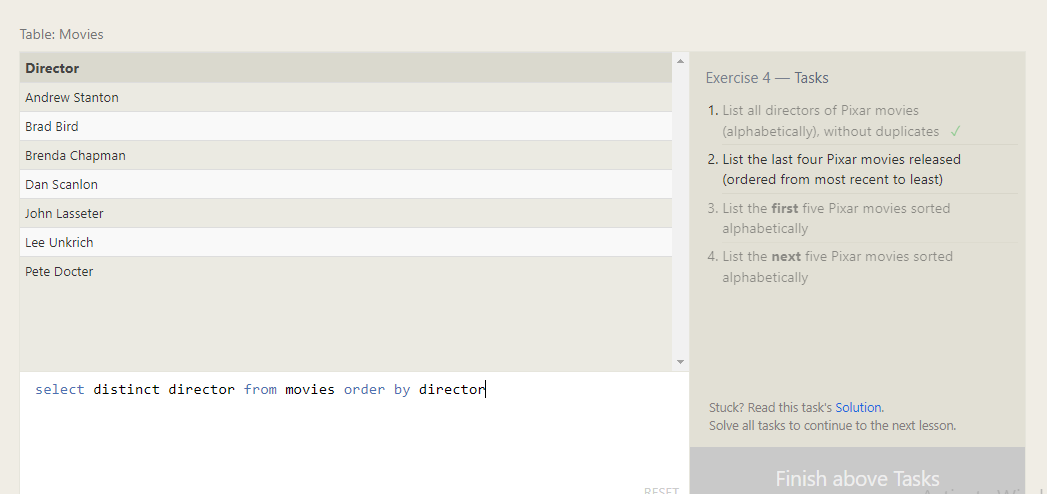


d.) Query: select \* from movies where title like("wall-%");

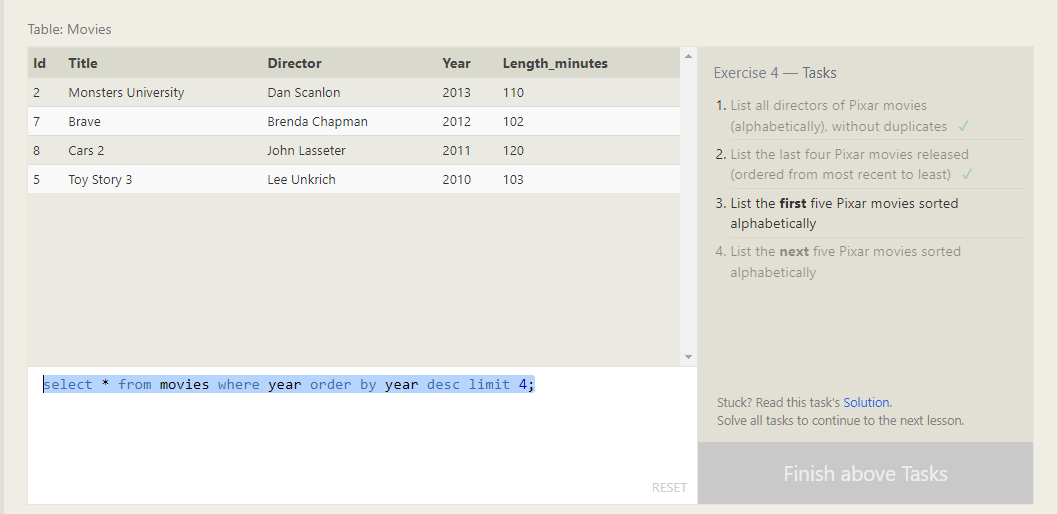


**Excersice-4 Task**

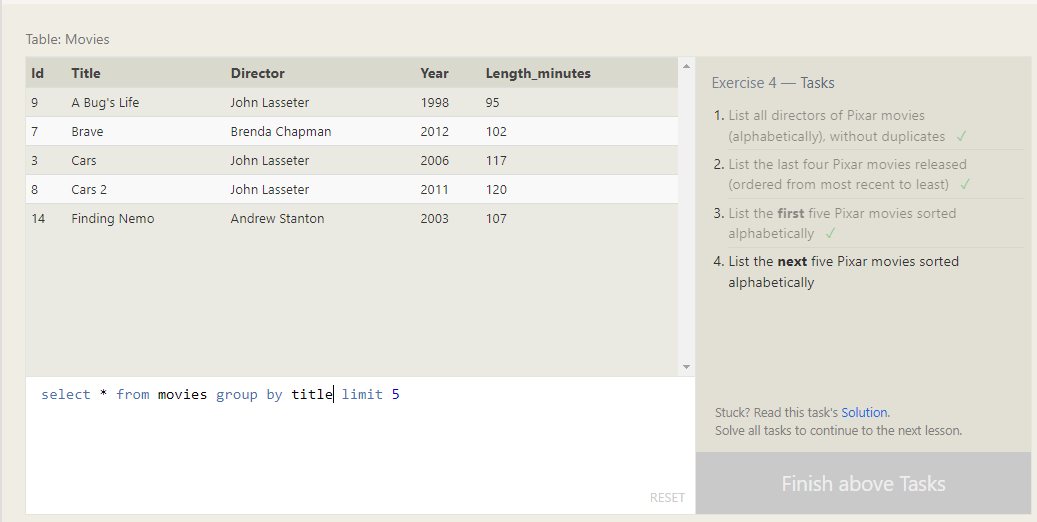
a.) Query: select distinct director from movies order by director;



b.) Query: select \* from movies where year order by year desc limit 4;



c.) Query: select \* from movies group by title limit 5;

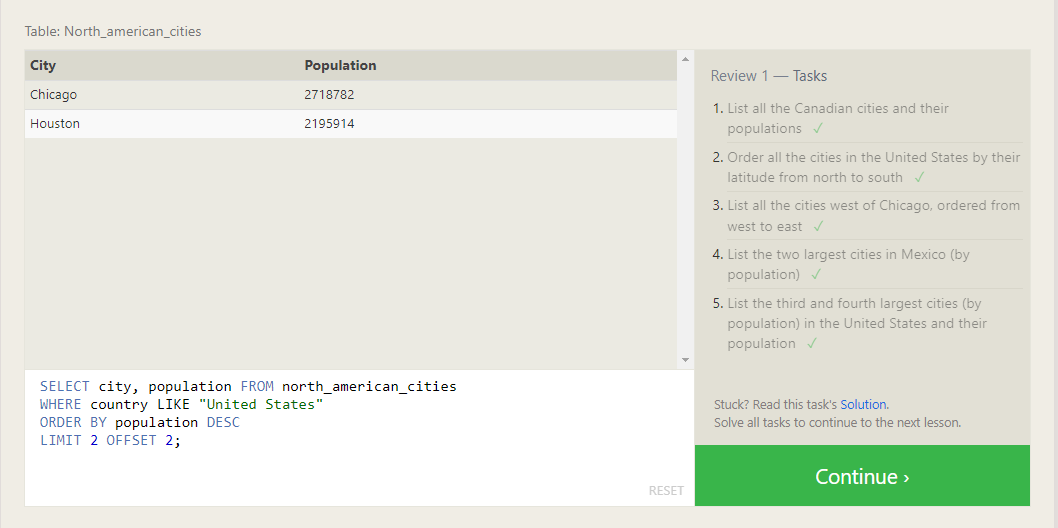


d.) Query : SELECT title FROM movies ORDER BY title ASC LIMIT 5 OFFSET 5;



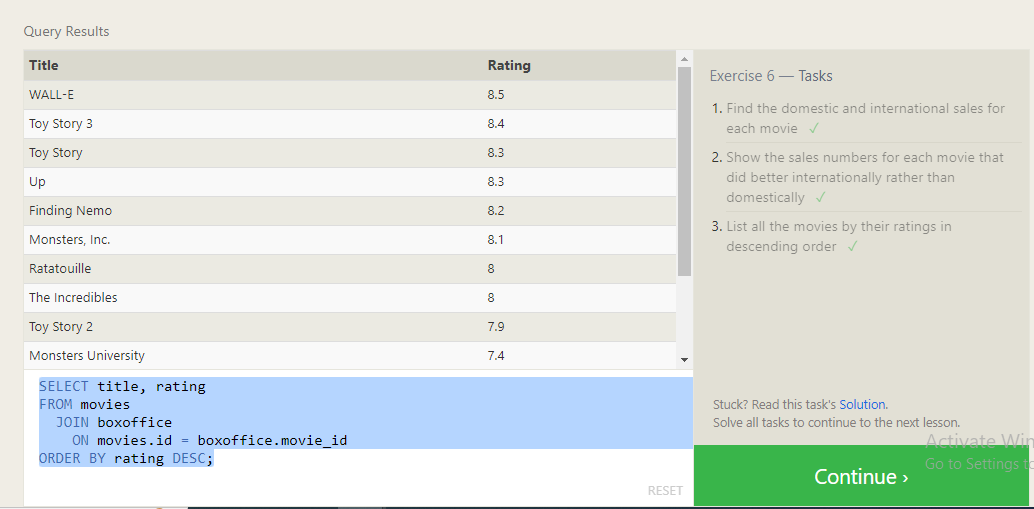
**Excersice-5 Task**

1. select city from north\_american\_cities where country like("canada");
2. SELECT city, latitude FROM north\_american\_cities WHERE country = "United States" ORDER BY latitude DESC;
3. SELECT city, longitude FROM north\_american\_cities WHERE longitude < -87.629798 ORDER BY longitude ASC;
4. select \* from north\_american\_cities where country like("mexico") order by population desc limit 2;
5. select \* from north\_american\_cities where country like("mexico") order by population desc limit 2;



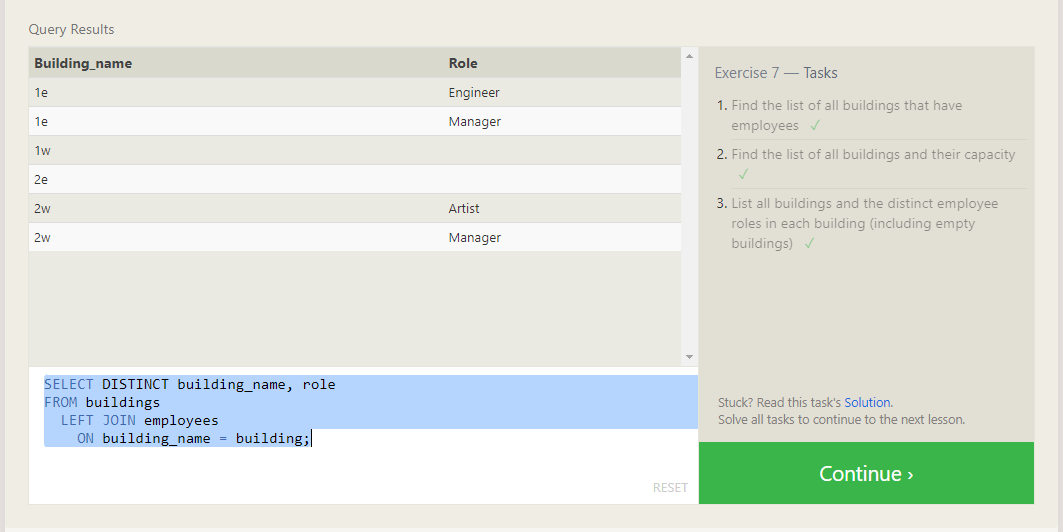
**SQL Lesson 6: Multi-table queries with JOIN**

1. SELECT title, domestic\_sales, international\_sales FROM movies JOIN boxoffice ON movies.id = boxoffice.movie\_id;
2. SELECT title, domestic\_sales, international\_sales FROM movies JOIN boxoffice ON movies.id = boxoffice.movie\_id WHERE international\_sales > domestic\_sales;
3. SELECT title, domestic\_sales, international\_sales FROM movies JOIN boxoffice ON movies.id = boxoffice.movie\_id WHERE international\_sales > domestic\_sales;



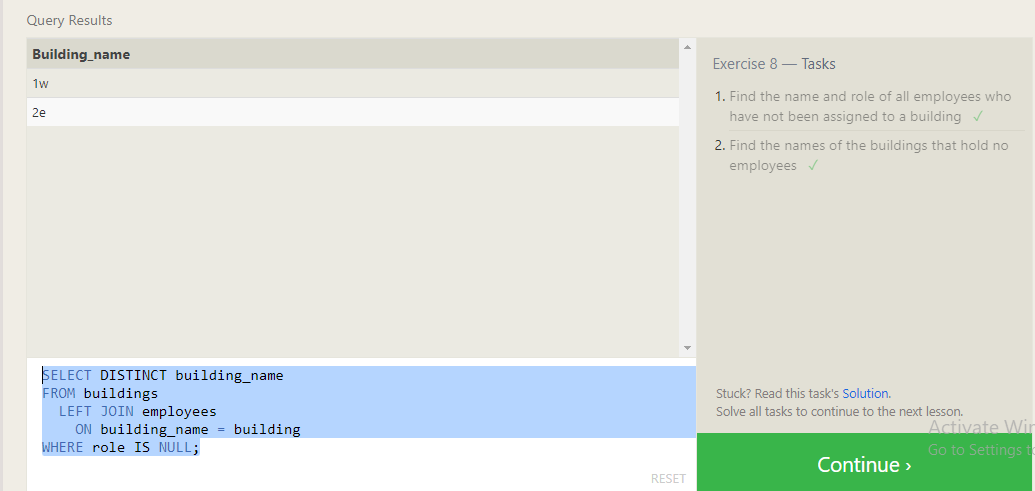
**SQL Lesson 7: OUTER JOIN**

1. SELECT DISTINCT building FROM employees;
2. SELECT \* FROM buildings;
3. SELECT DISTINCT building\_name, role FROM buildings LEFT JOIN employees ON building\_name = building;



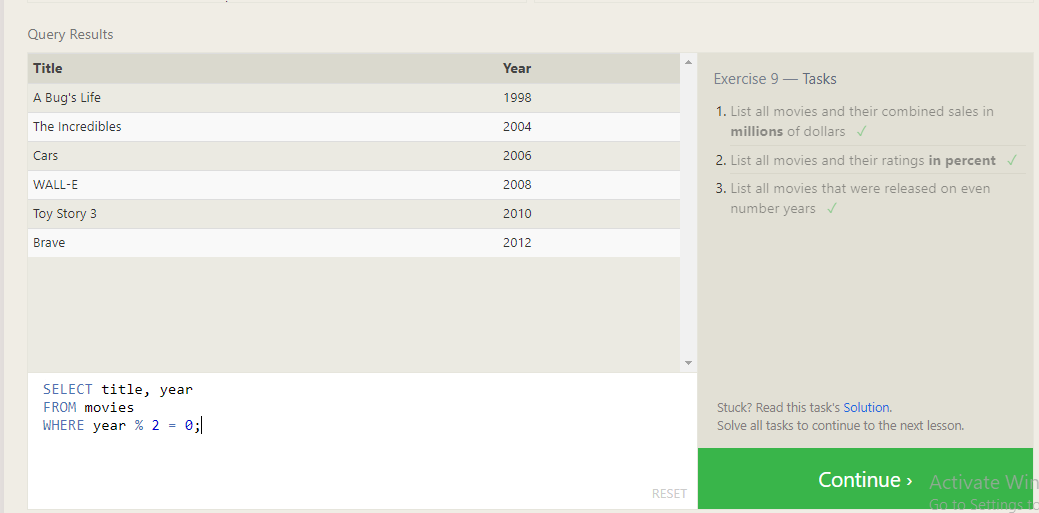
**SQL Lesson 8: A short note on NULL**

1. SELECT name, role FROM employees WHERE building IS NULL;
2. SELECT DISTINCT building\_name;
3. FROM buildings LEFT JOIN employees ON building\_name = building WHERE role IS NULL;



**SQL Lesson 9: Queries with expressions**

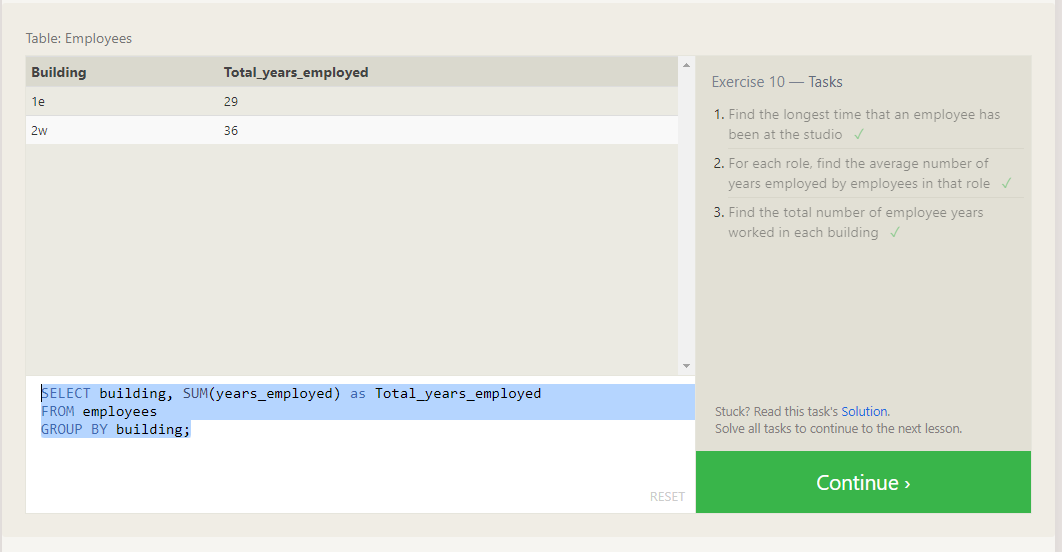
1. SELECT title, (domestic\_sales + international\_sales) / 1000000 AS gross\_sales\_millions FROM movies JOIN boxoffice ON movies.id = boxoffice.movie\_id;
2. SELECT title, rating \* 10 AS rating\_percent FROM movies JOIN boxoffice ON movies.id = boxoffice.movie\_id;
3. SELECT title, year FROM movies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



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

1. SELECT MAX(years\_employed) as Max\_years\_employed FROM employees;
2. SELECT role, AVG(years\_employed) as Average\_years\_employed FROM employees GROUP BY role;
3. SELECT building, SUM(years\_employed) as Total\_years\_employed

FROM employees GROUP BY building;

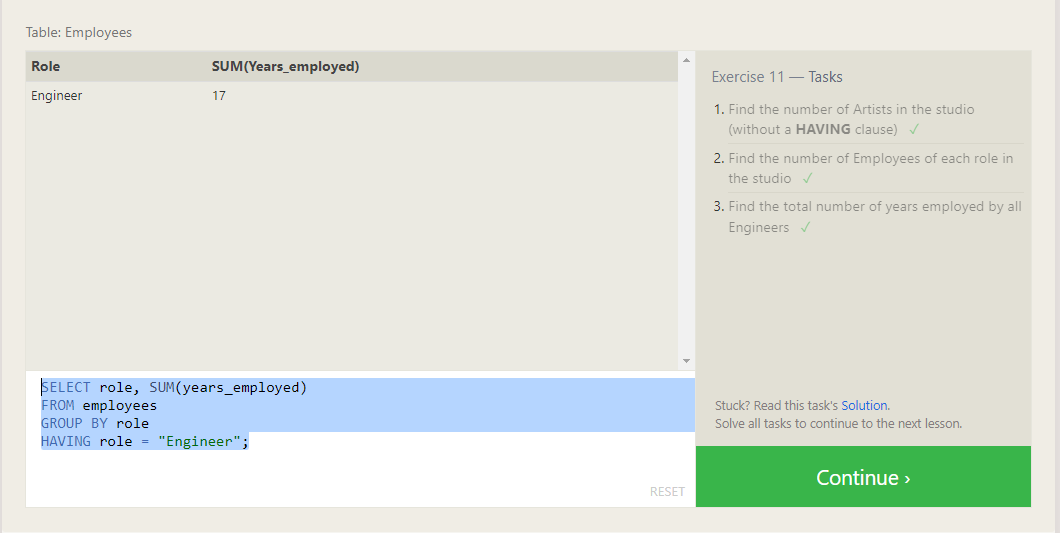


**SQL Lesson 11: Queries with aggregates (Pt. 2)**

1. SELECT role, COUNT(\*) as Number\_of\_artists

FROM employees WHERE role = "Artist";

1. SELECT role, COUNT(\*) FROM employees GROUP BY role;
2. SELECT role, SUM(years\_employed) FROM employees GROUP BY role HAVING role = "Engineer";

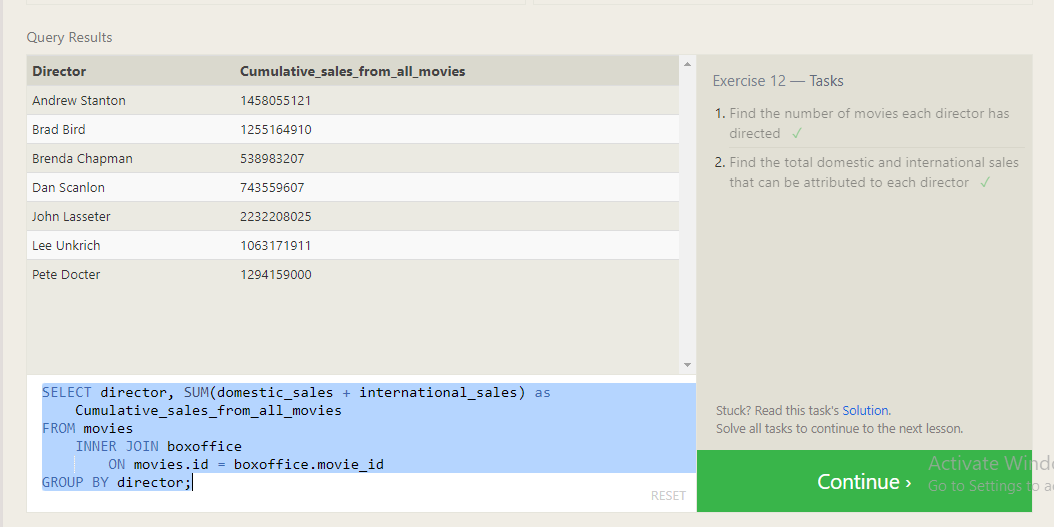


**SQL Lesson 12: Order of execution of a Query**

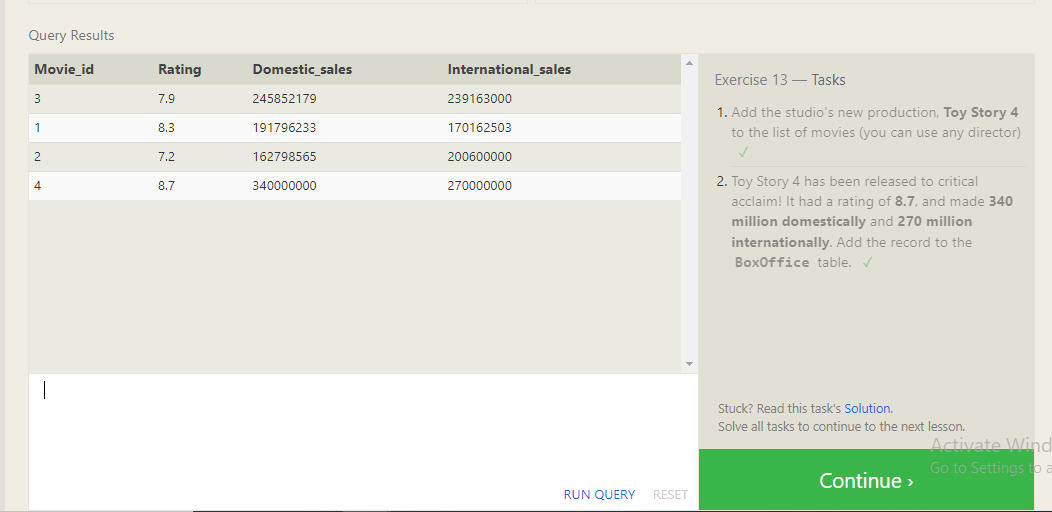
1. SELECT director, COUNT(id) as Num\_movies\_directed

FROM movies GROUP BY director;

1. SELECT director, SUM(domestic\_sales + international\_sales) as Cumulative\_sales\_from\_all\_movies FROM movies INNER JOIN boxoffice ON movies.id = boxoffice.movie\_id GROUP BY director;

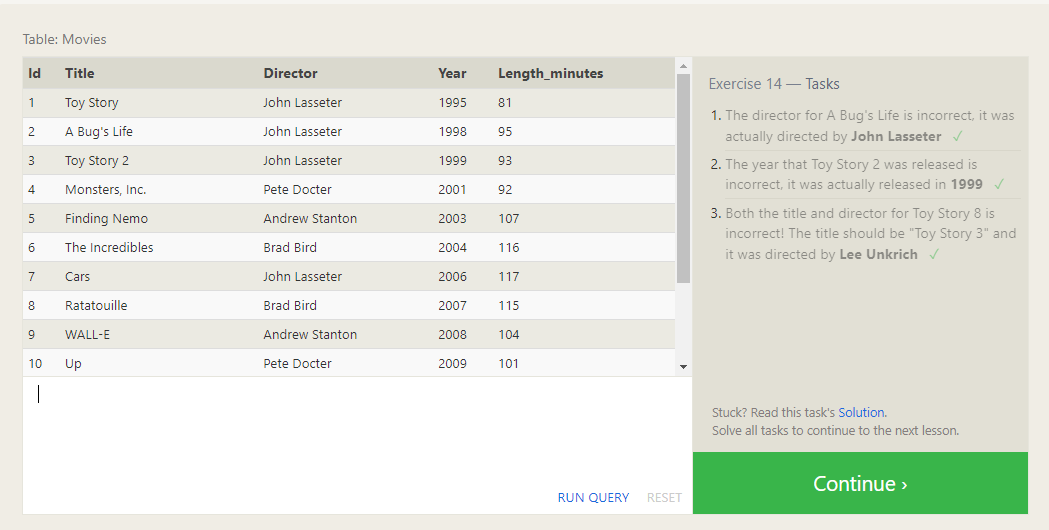


**SQL Lesson 13: Inserting rows**



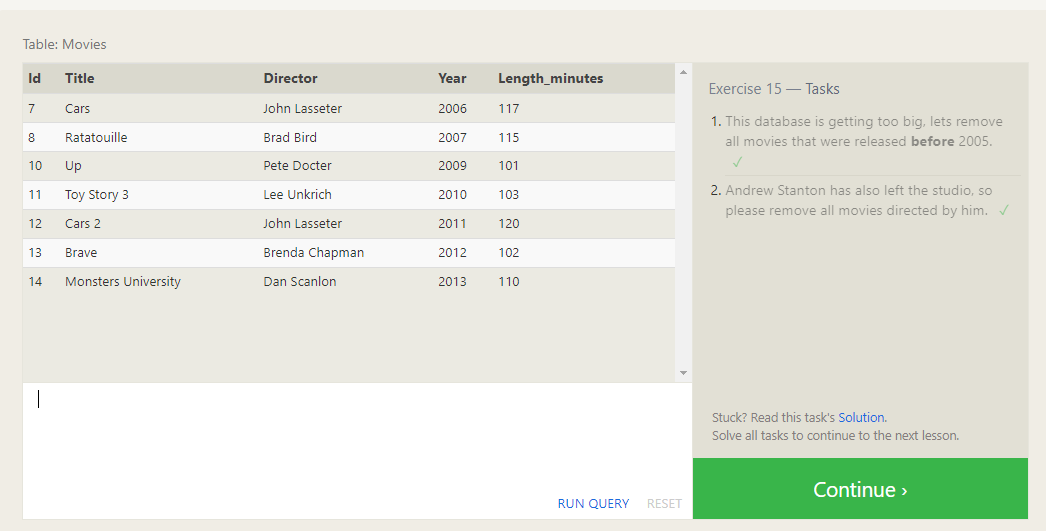
**SQL Lesson 14: Updating rows**

1. UPDATE movies SET director = "John Lasseter" WHERE id = 2;
2. UPDATE movies SET year = 1999 WHERE id = 3;
3. UPDATE movies SET title = "Toy Story 3", director = "Lee Unkrich" WHERE id = 11;



**SQL Lesson 15: Deleting rows**

1. DELETE FROM movies where year < 2005;
2. DELETE FROM movies where director = "Andrew Stanton";



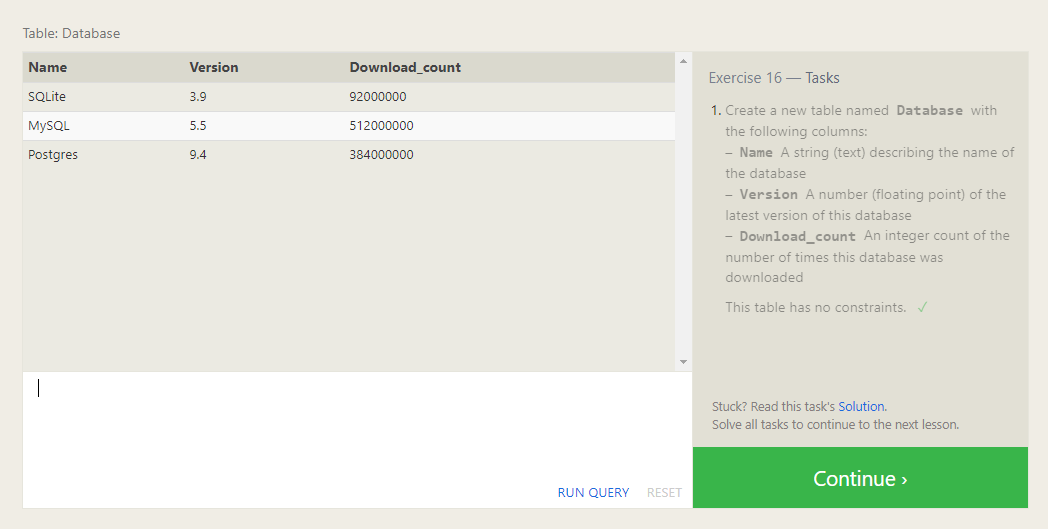
**SQL Lesson 16: Creating tables**

1. CREATE TABLE Database (

Name TEXT,

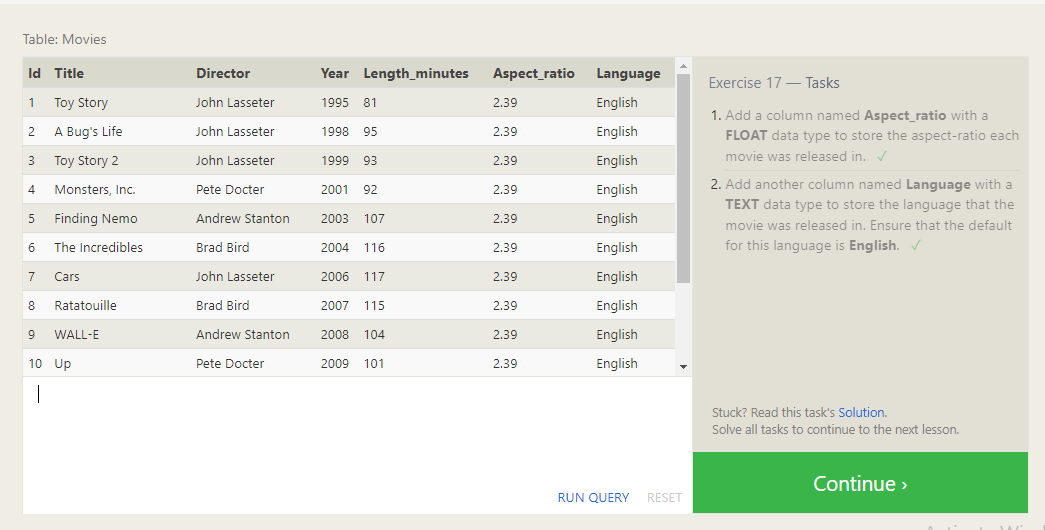
Version FLOAT,

Download\_count INTEGER);



**SQL Lesson 17: Altering tables**

1. ALTER TABLE Movies ADD COLUMN Aspect\_ratio FLOAT DEFAULT 2.39;
2. ALTER TABLE Movies ADD COLUMN Language TEXT DEFAULT "English";



**SQL Lesson 18: Dropping tables**

1. DROP TABLE Movies;
2. DROP TABLE BoxOffice;

