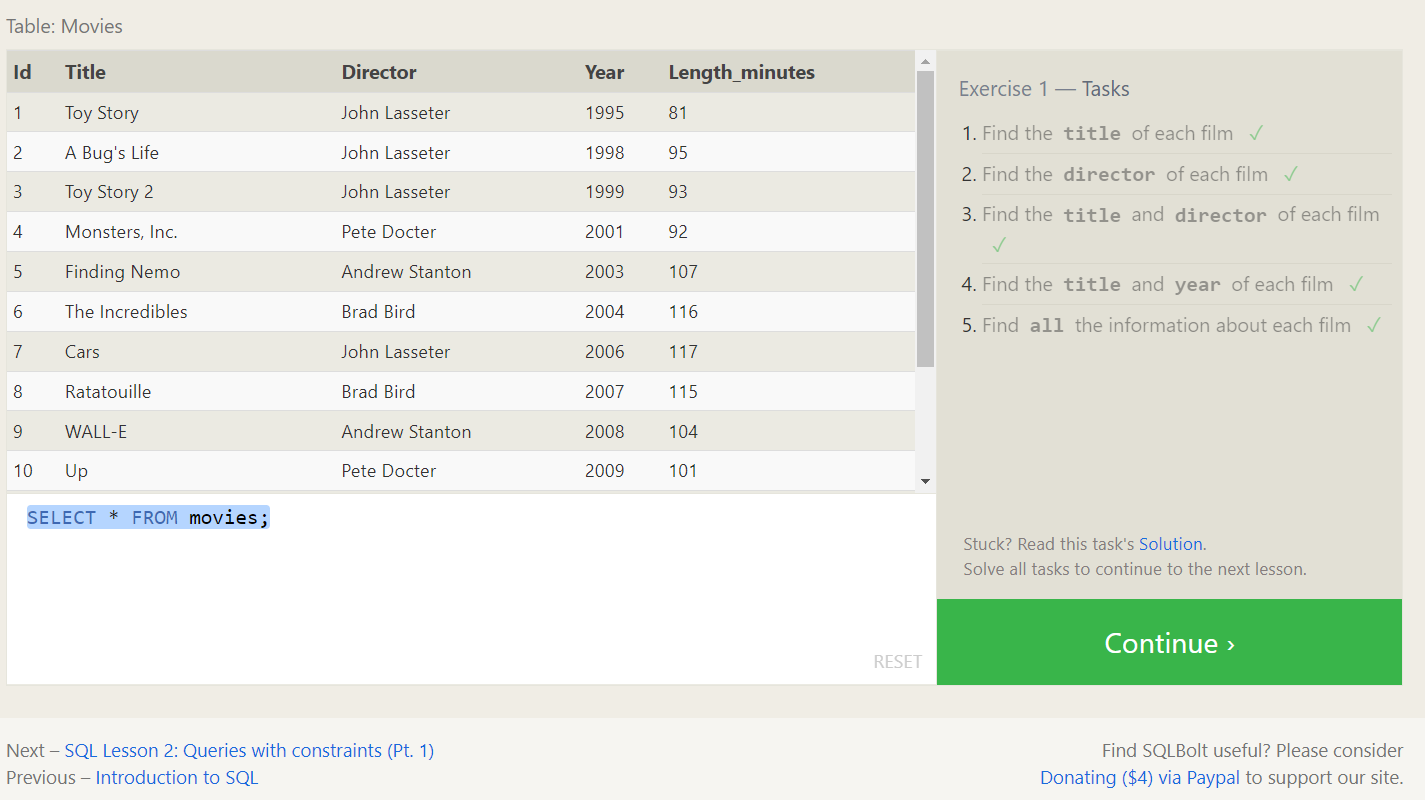
**SQL QUERIES TASK**

# **SQL Lesson 1: SELECT queries 101**



**1.SELECT title FROM movies;**

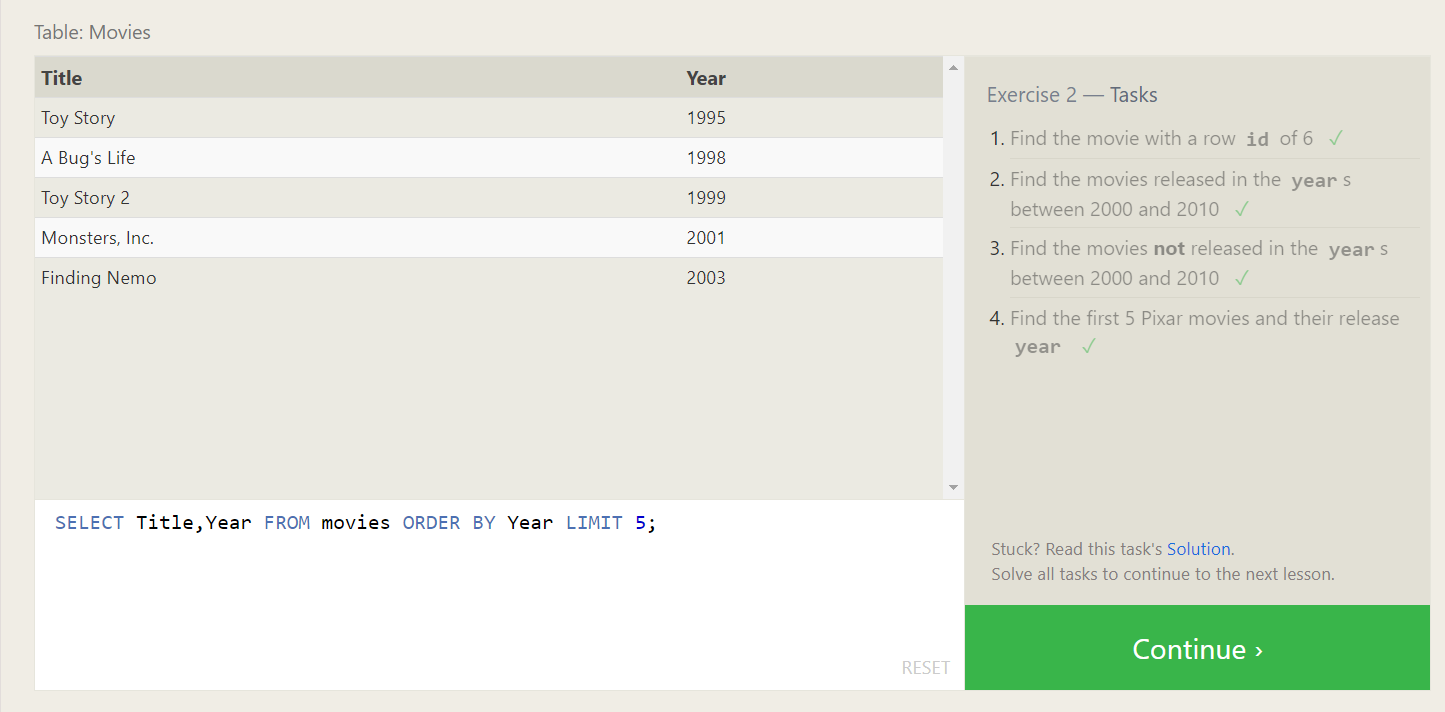
**2.SELECT director FROM movies;**

**3.SELECT title,director FROM movies;**

**4.SELECT title,year FROM movies;**

**5.SELECT \* FROM movies;**

# **SQL Lesson 2: Queries with constraints**



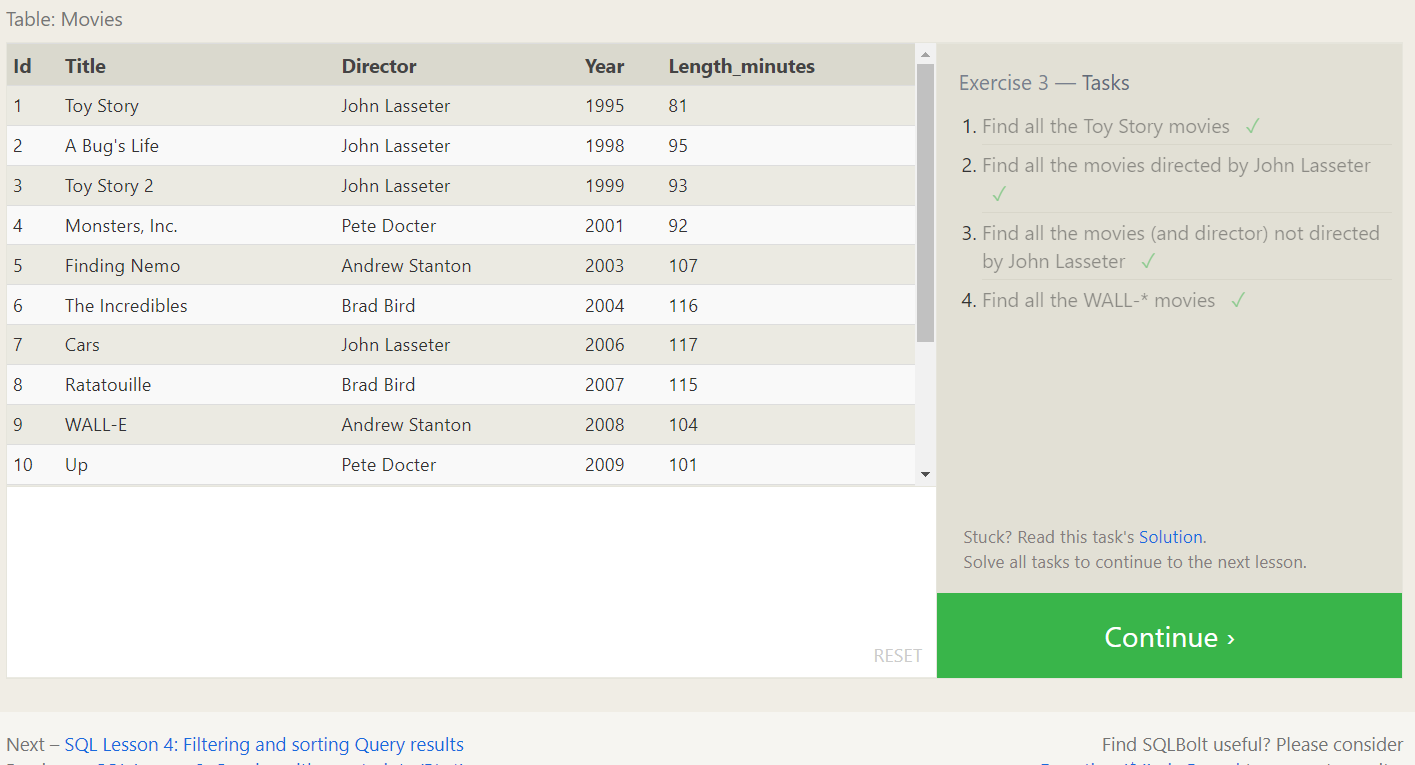
**1.SELECT \* FROM movies where id is 6**

**2.SELECT \* FROM movies WHERE Year BETWEEN 2000 AND 2010;**

**3.SELECT \* FROM movies WHERE Year NOT BETWEEN 2000 AND 2010;**

**4.SELECT Title,Year FROM movies ORDER BY Year LIMIT 5;**

# **SQL Lesson 3: Queries with constraints**

****

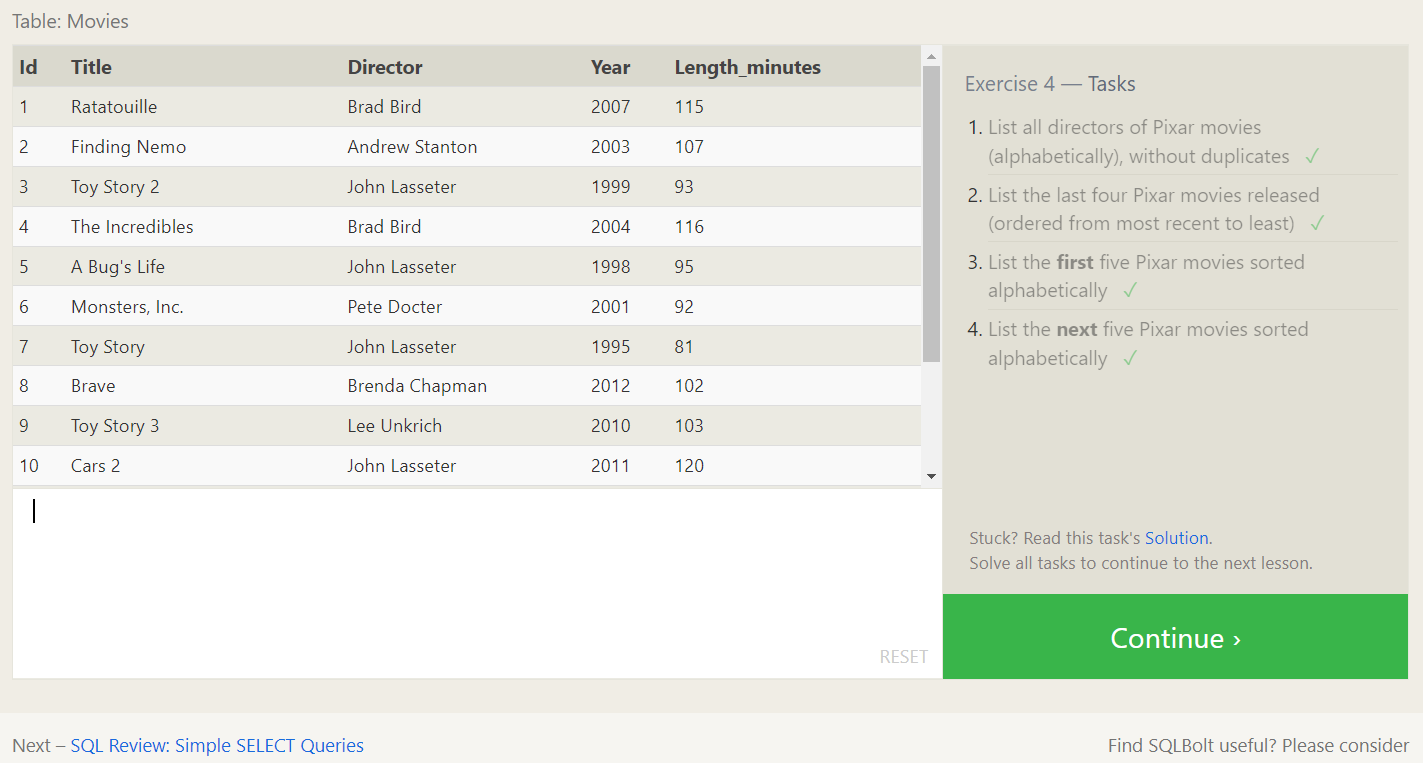
**1.SELECT \* FROM movies WHERE Title LIKE 'Toy Story%';**

**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**



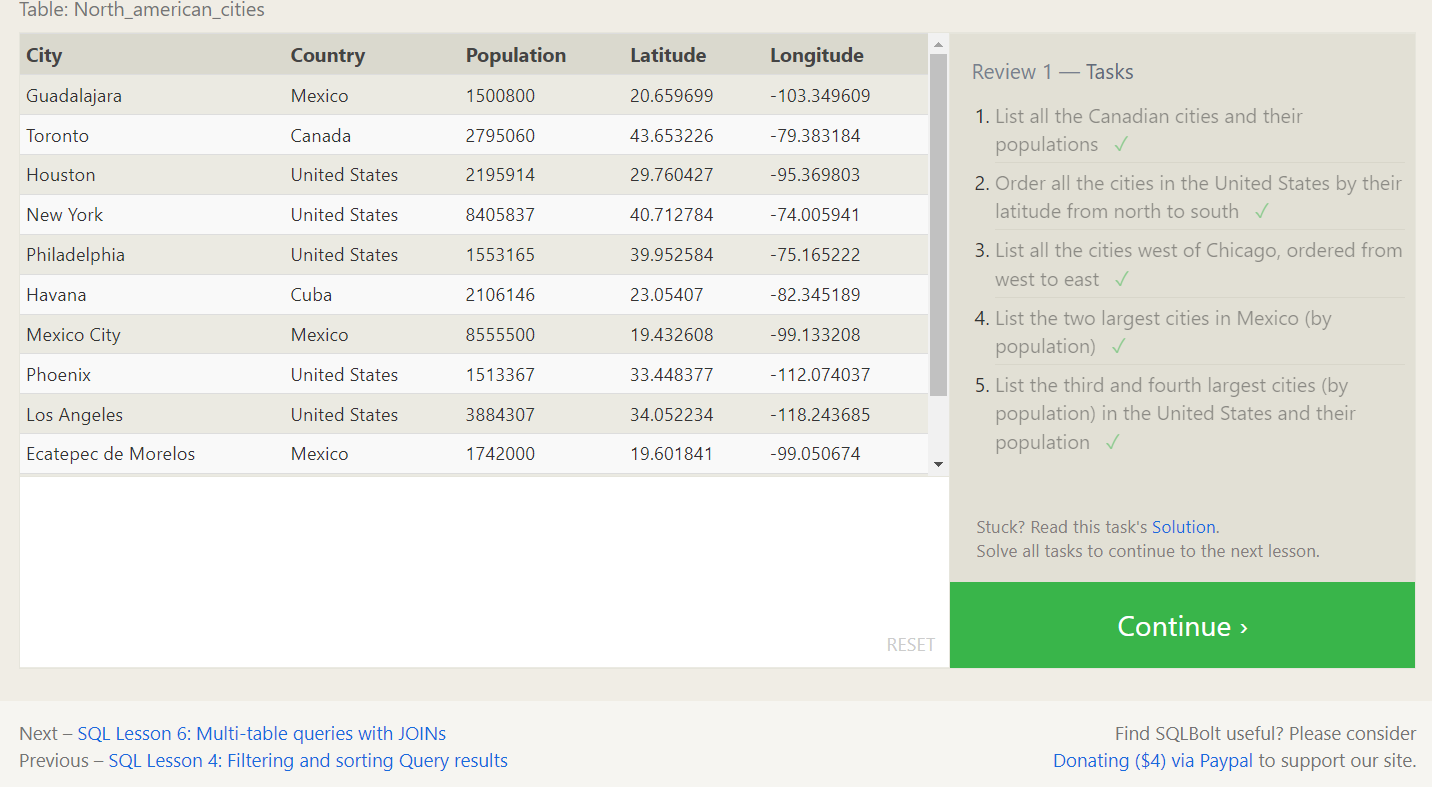
**1.SELECT DISTINCT Director FROM movies ORDER BY Director;**

**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;**

# **SQL Review: Simple SELECT Queries**



**1.SELECT City,population FROM North\_american\_cities WHERE country = '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 < (SELECT Longitude FROM North\_american\_cities**

**WHERE City = 'Chicago') ORDER BY Longitude ASC;**

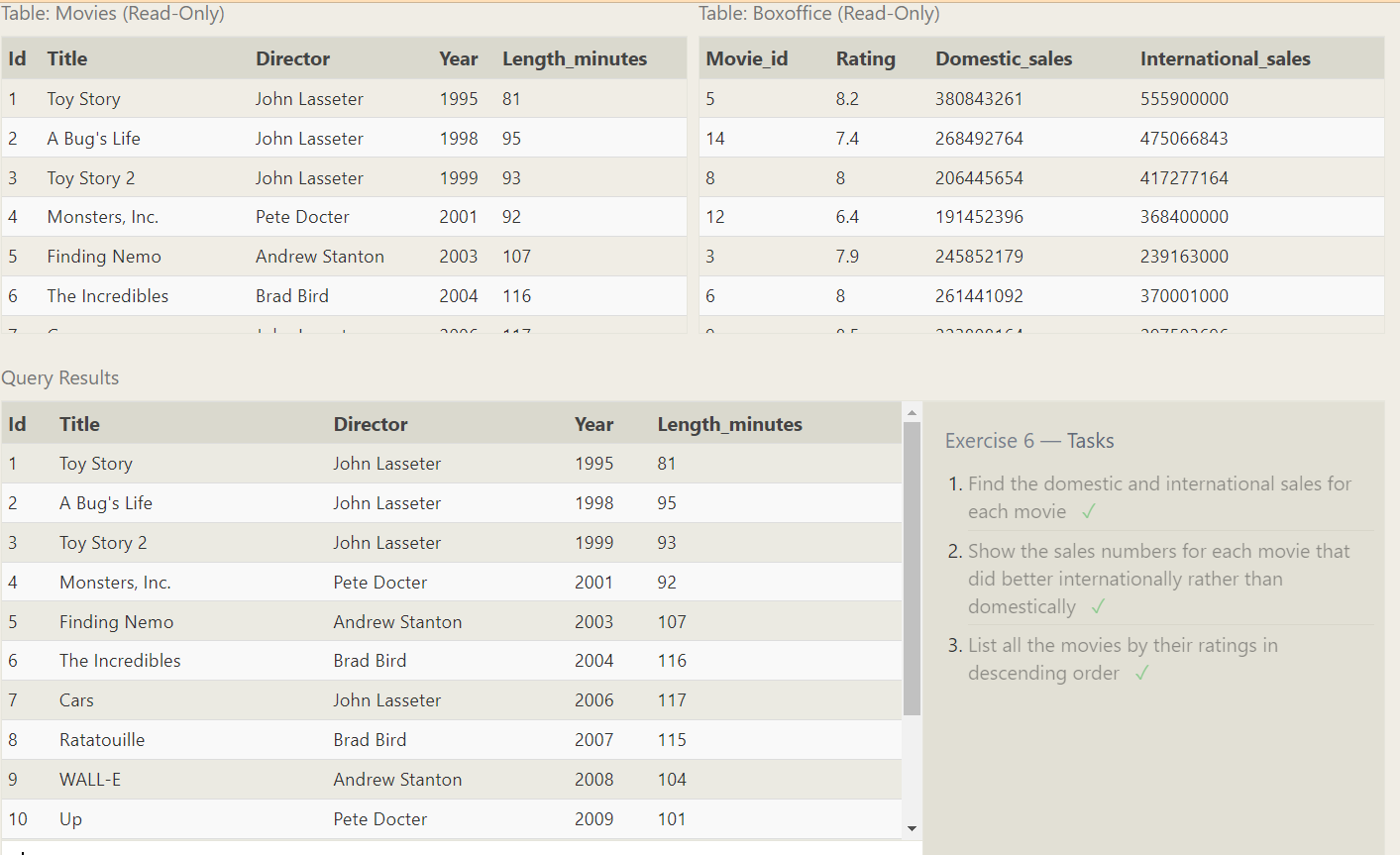
**4.SELECT City, Population FROM North\_american\_cities WHERE Country = 'Mexico'**

**ORDER BY Population DESC LIMIT 2;**

**5.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**



**1.SELECT m.Title, b.Domestic\_sales, b.International\_sales**

**FROM Movies m JOIN Boxoffice b ON m.Id = b.Movie\_id;**

**2.SELECT m.Title, b.Domestic\_sales, b.International\_sales**

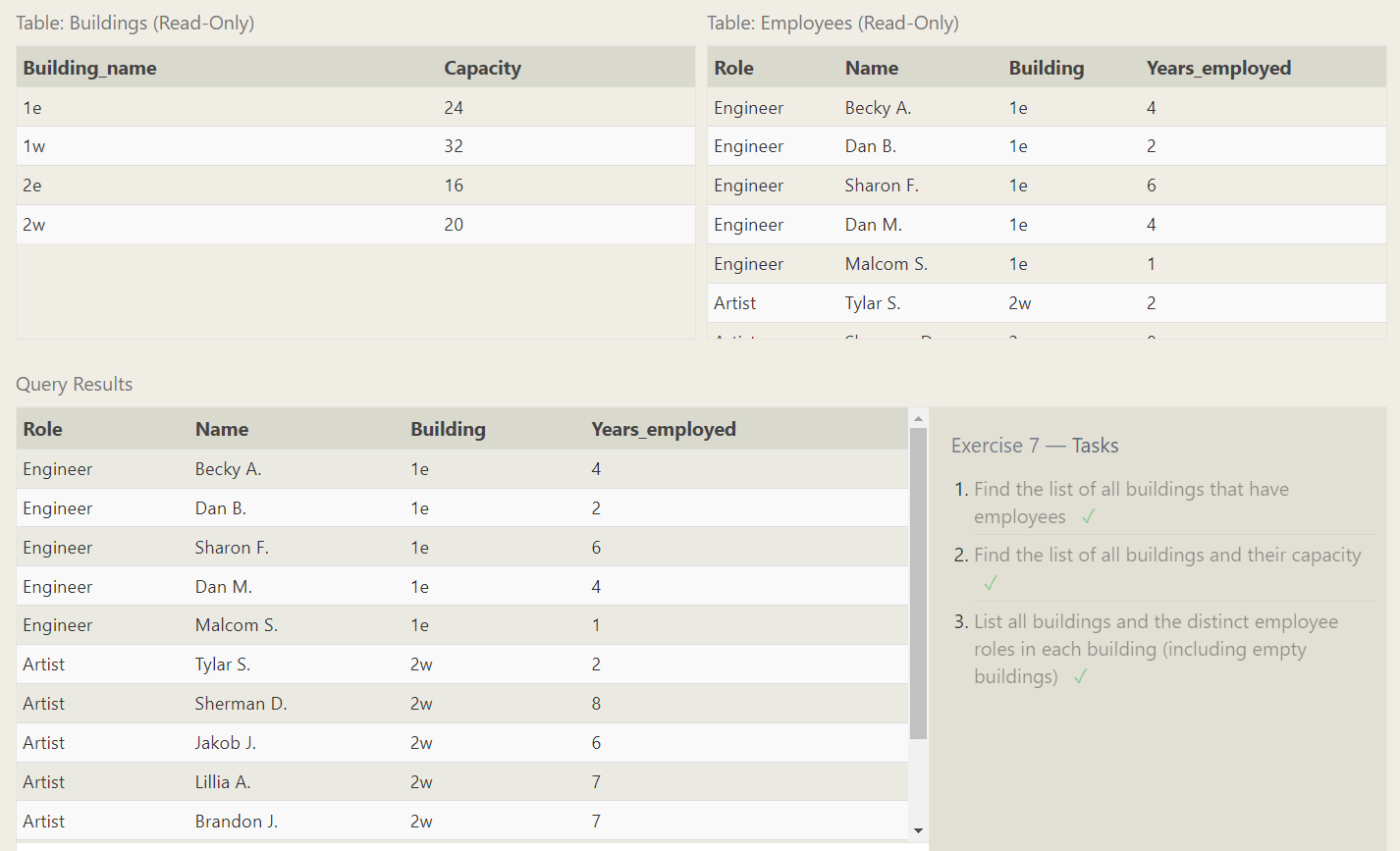
**FROM Movies m JOIN Boxoffice b ON m.Id = b.Movie\_id**

**WHERE b.International\_sales > b.Domestic\_sales;**

**3.SELECT m.Title, b.Rating FROM Movies m JOIN Boxoffice b ON m.Id = b.Movie\_id**

**ORDER BY b.Rating DESC;**

# **SQL Lesson 7: OUTER JOINs**



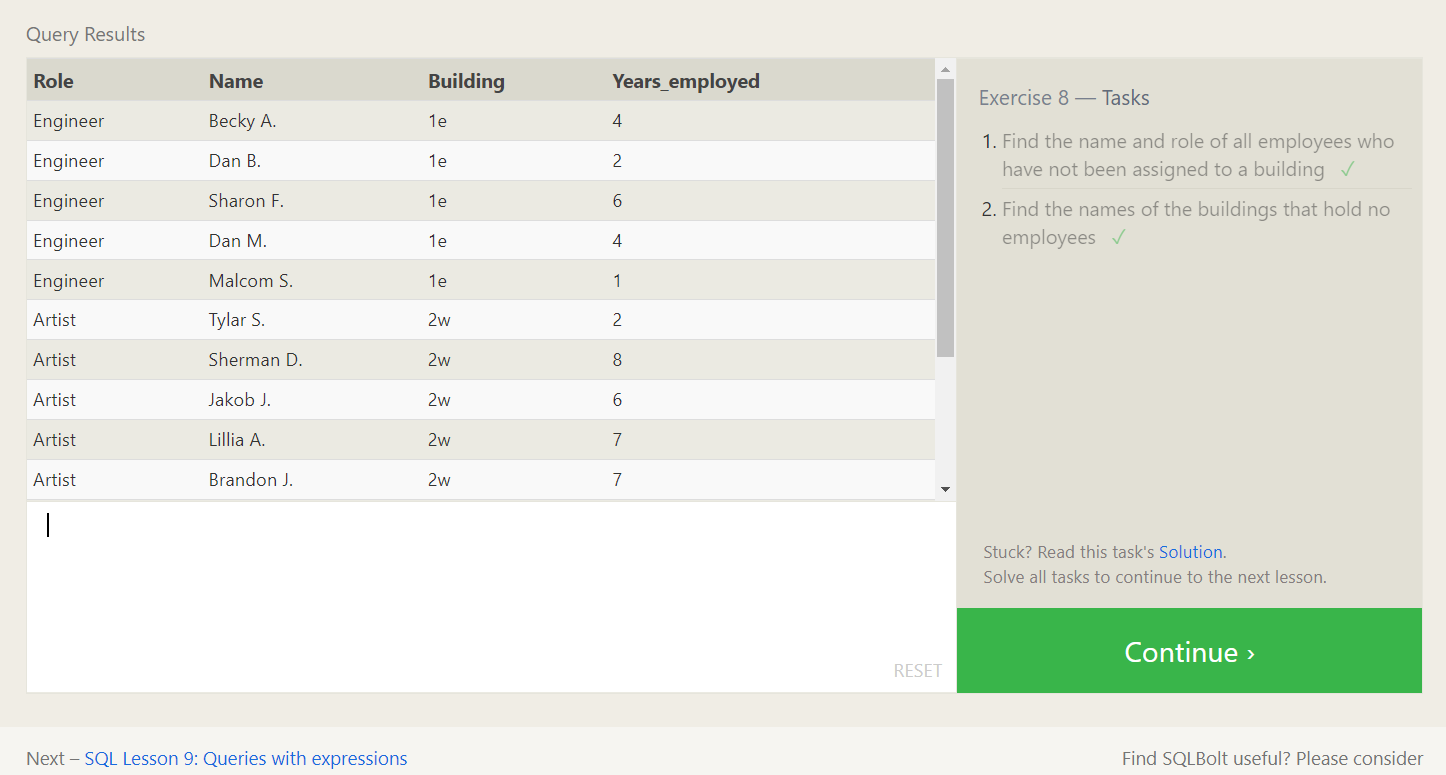
**1.SELECT DISTINCT Building FROM Employees;**

**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**



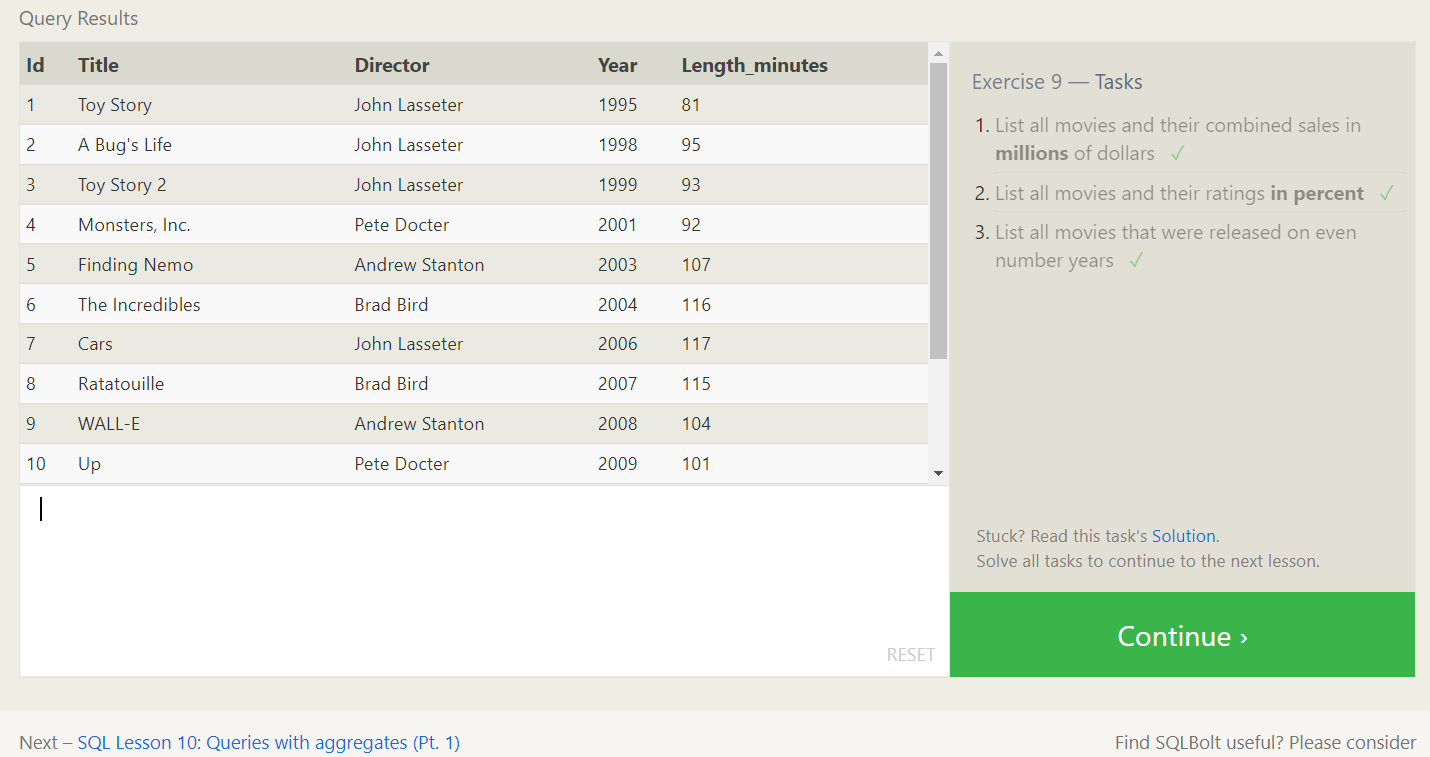
**1.SELECT Name, Role FROM Employees WHERE Building IS NULL;**

**2.SELECT Building\_name FROM Buildings**

**WHERE Building\_name NOT IN (SELECT DISTINCT Building FROM Employees**

**WHERE Building IS NOT NULL);**

# **SQL Lesson 9: Queries with expressions**



**1.SELECT m.Title, (b.Domestic\_sales + b.International\_sales) / 1000000 AS**

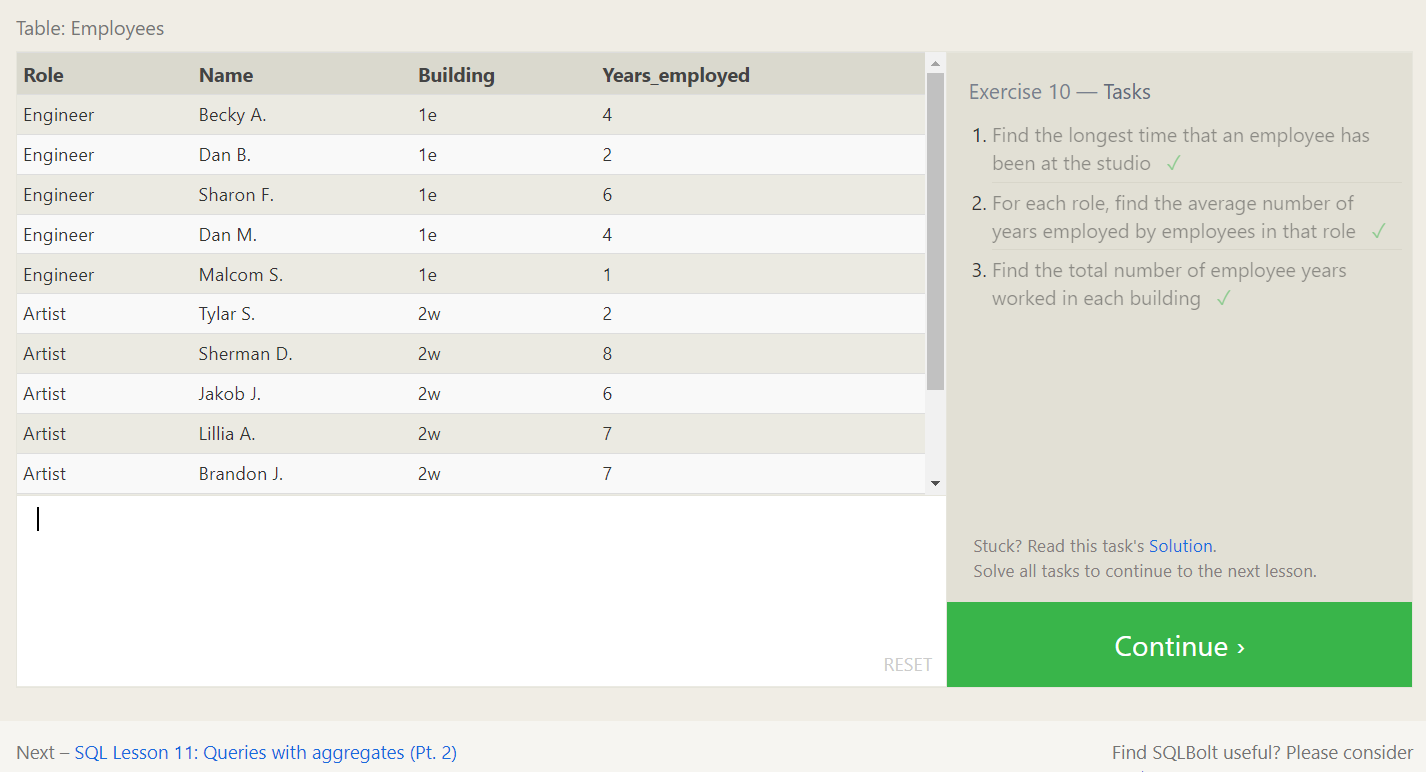
**Combined\_sales FROM Movies m JOIN Boxoffice b ON m.Id = b.Movie\_id;**

**2.SELECT title, rating \* 10 AS rating\_percent FROM movies JOIN boxoffice**

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

**3.SELECT \* FROM Movies WHERE Year % 2 = 0;**

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



**1.SELECT MAX(Years\_employed) AS Longest\_time FROM Employees;**

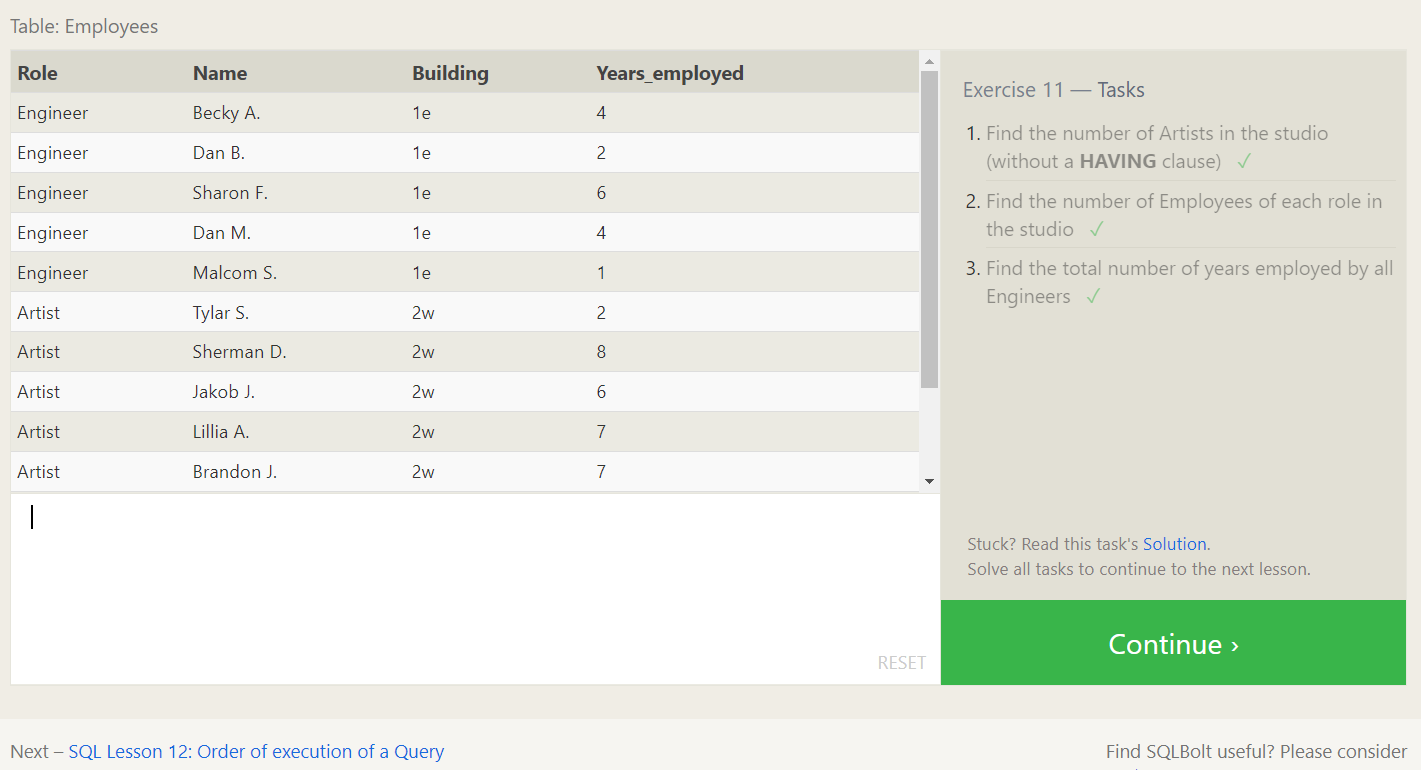
**2.SELECT Role, AVG(Years\_employed) AS Average\_years FROM Employees**

**GROUP BY Role;**

**3.SELECT Building, SUM(Years\_employed) AS Total\_years\_worked**

**FROM Employees GROUP BY Building;**

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



**1.SELECT COUNT(\*) AS Number\_of\_Artists FROM Employees**

**WHERE Role = 'Artist';**

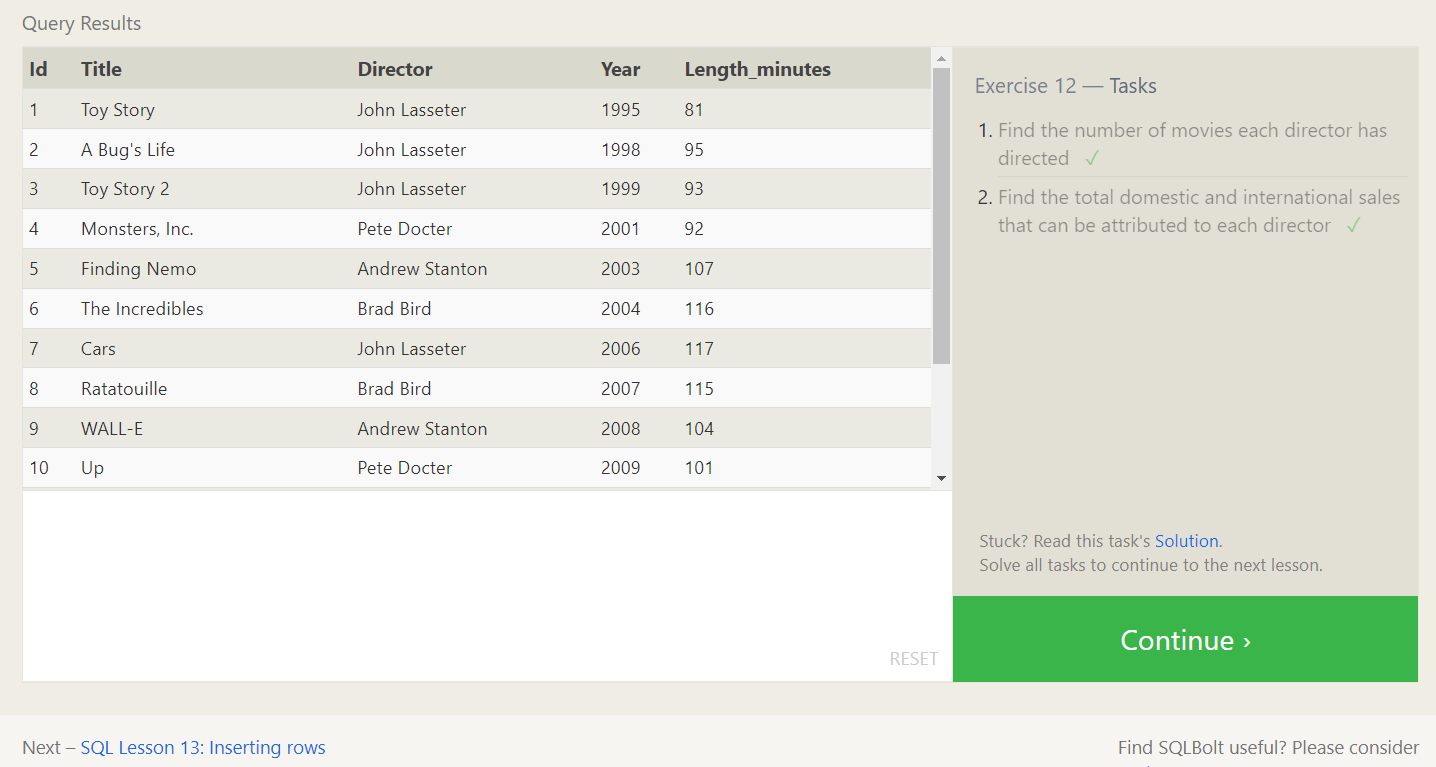
**2.SELECT Role, COUNT(\*) AS Number\_of\_Employees FROM Employees**

**GROUP BY Role;**

**3.SELECT SUM(Years\_employed) AS Total\_Years\_Employed FROM Employees**

**WHERE Role = 'Engineer';**

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



**1.SELECT Director, COUNT(\*) AS Movies\_Directed FROM Movies**

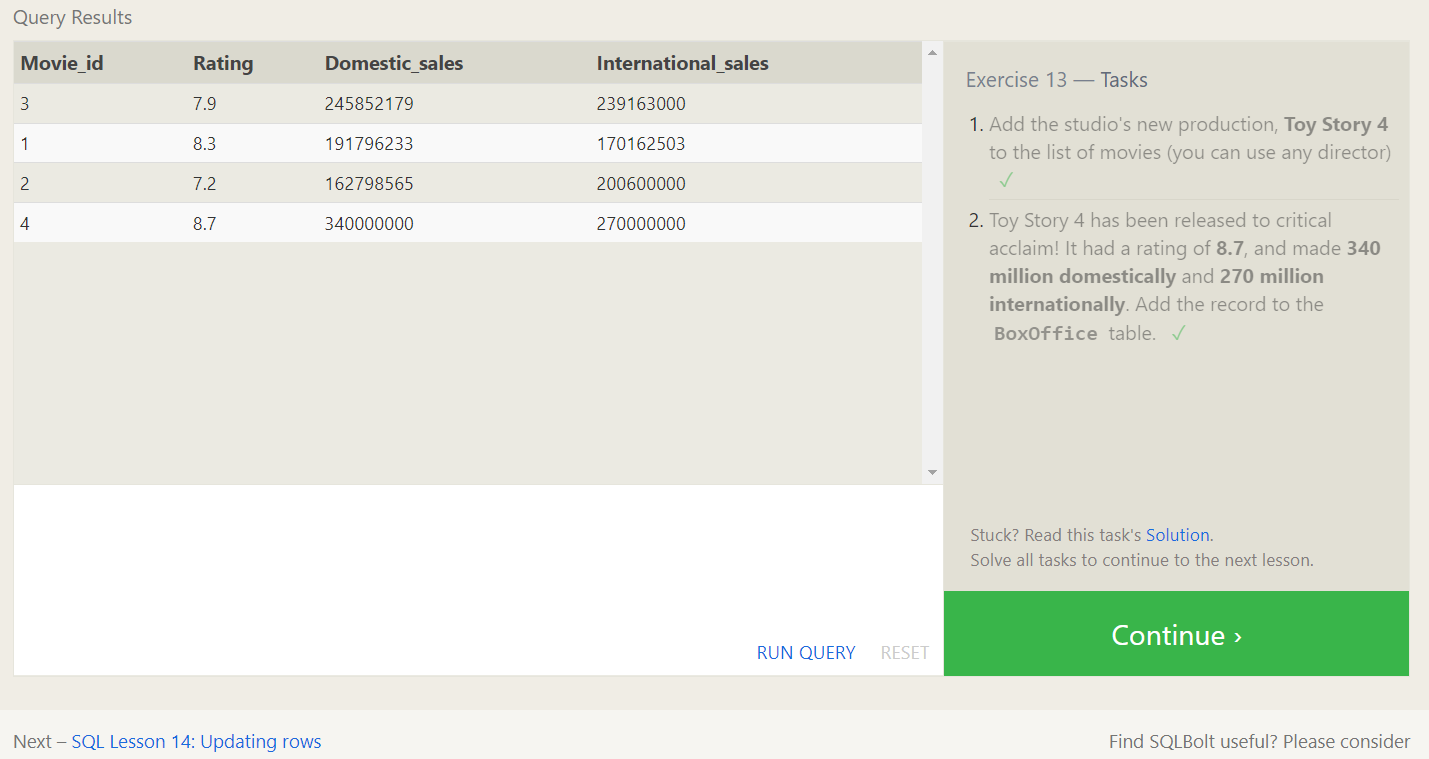
**GROUP BY Director;**

**2.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**



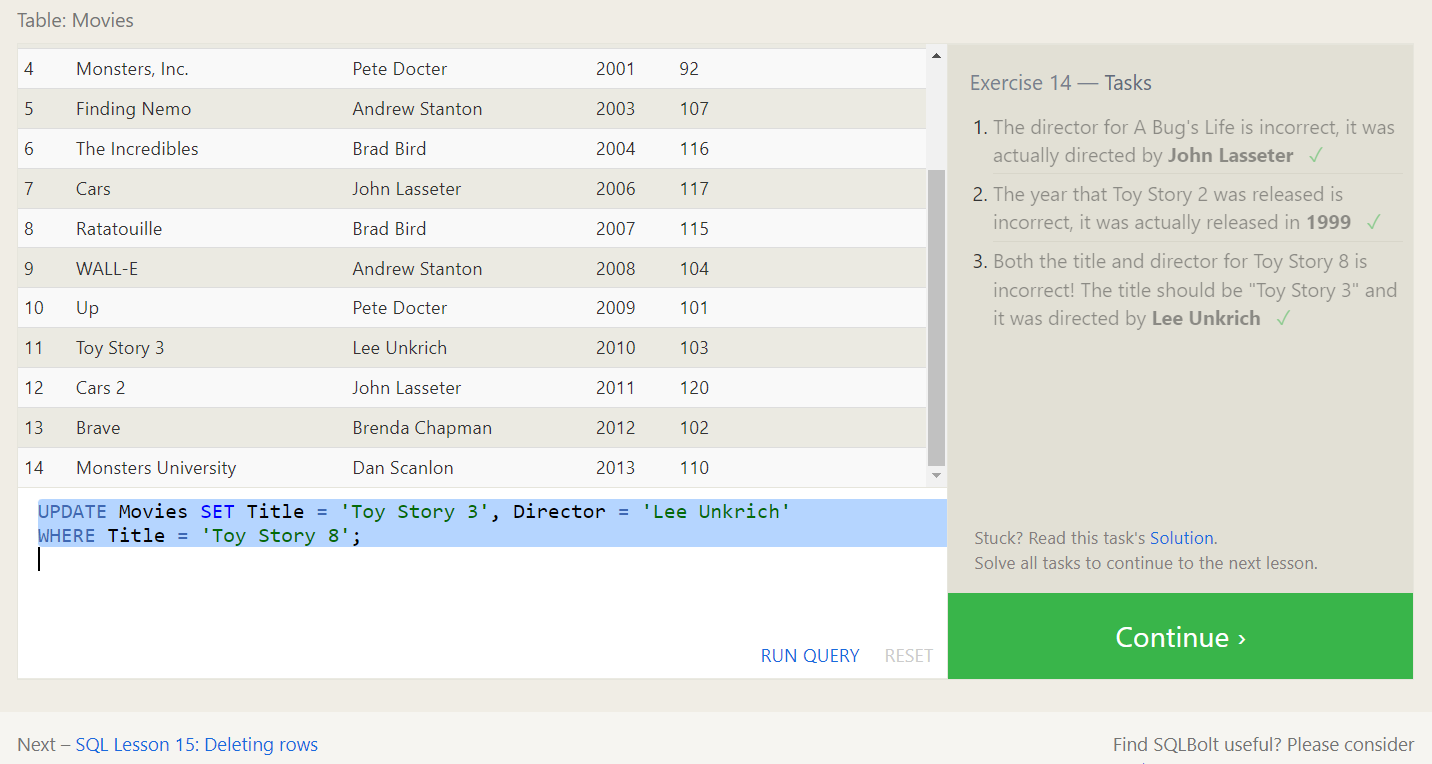
**1.INSERT INTO Movies (Title, Director, Year, Length\_minutes)**

**VALUES ('Toy Story 4', 'Any Director', 2022, 0);**

**2.INSERT INTO Boxoffice (Movie\_id, Rating, Domestic\_sales, International\_sales)**

**VALUES ((SELECT Id FROM Movies WHERE Title = 'Toy Story 4'), 8.7, 340000000, 270000000);**

# **SQL Lesson 14: Updating rows**



**1.UPDATE Movies SET Director = 'John Lasseter'**

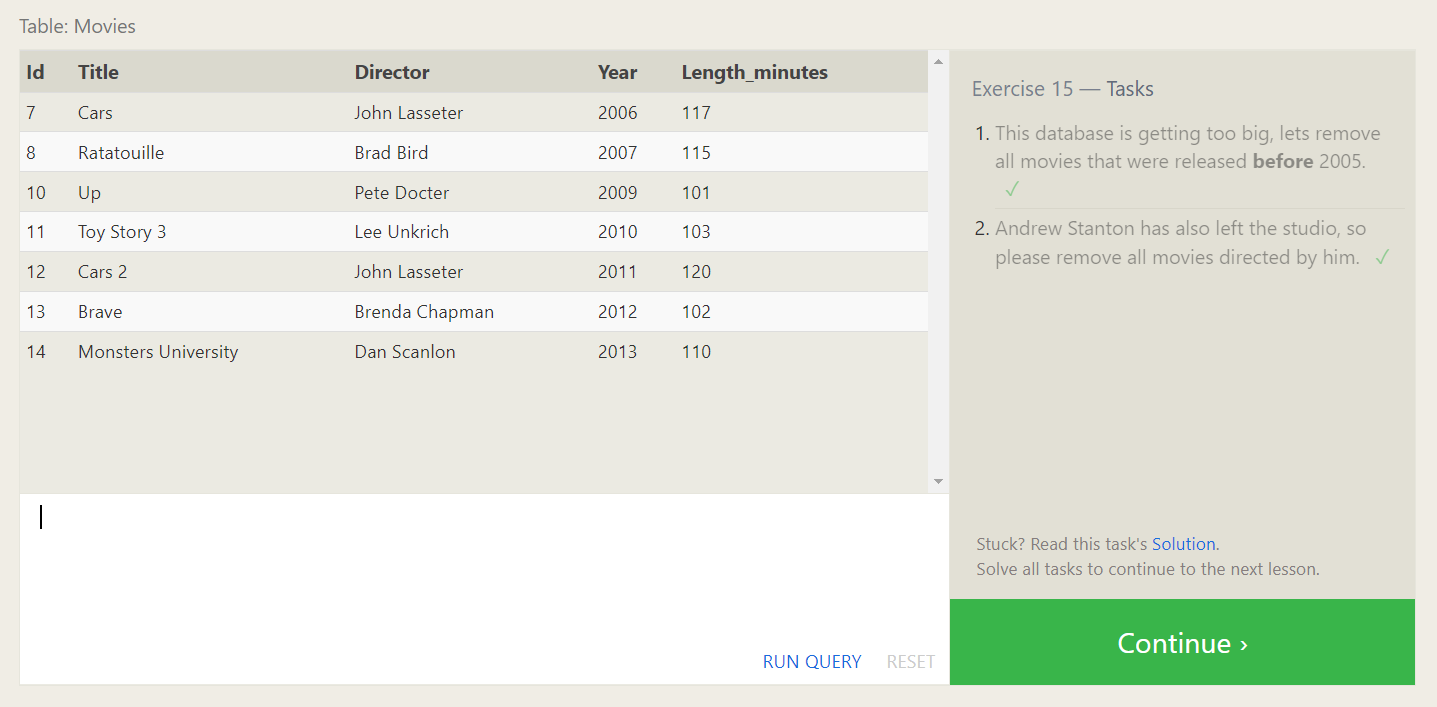
**WHERE Title = 'A Bug''s Life';**

**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**



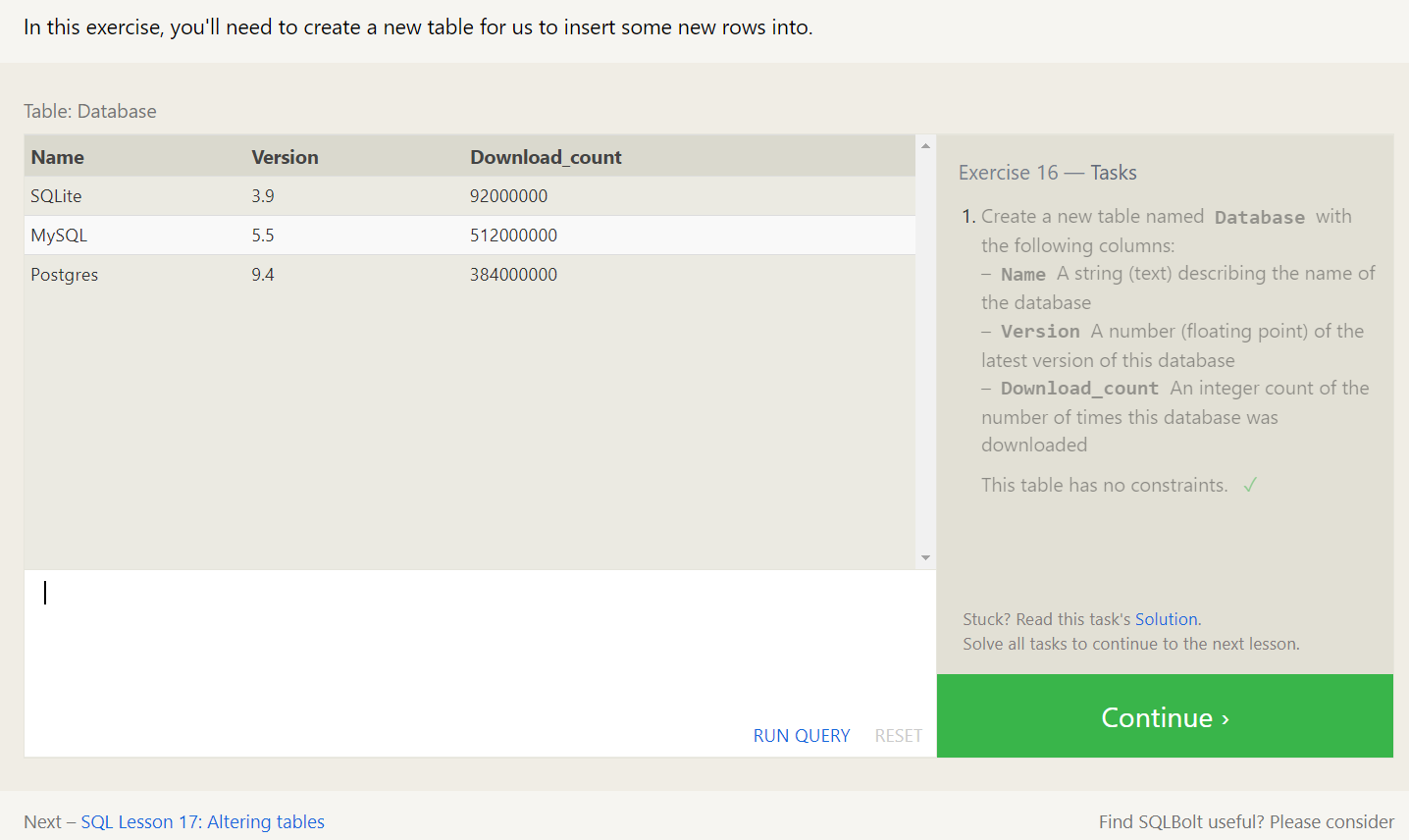
1.DELETE FROM Movies

WHERE Year < 2005;

2.DELETE FROM Movies

WHERE Director = 'Andrew Stanton';

# **SQL Lesson 16: Creating tables**



**1.CREATE TABLE Database (**

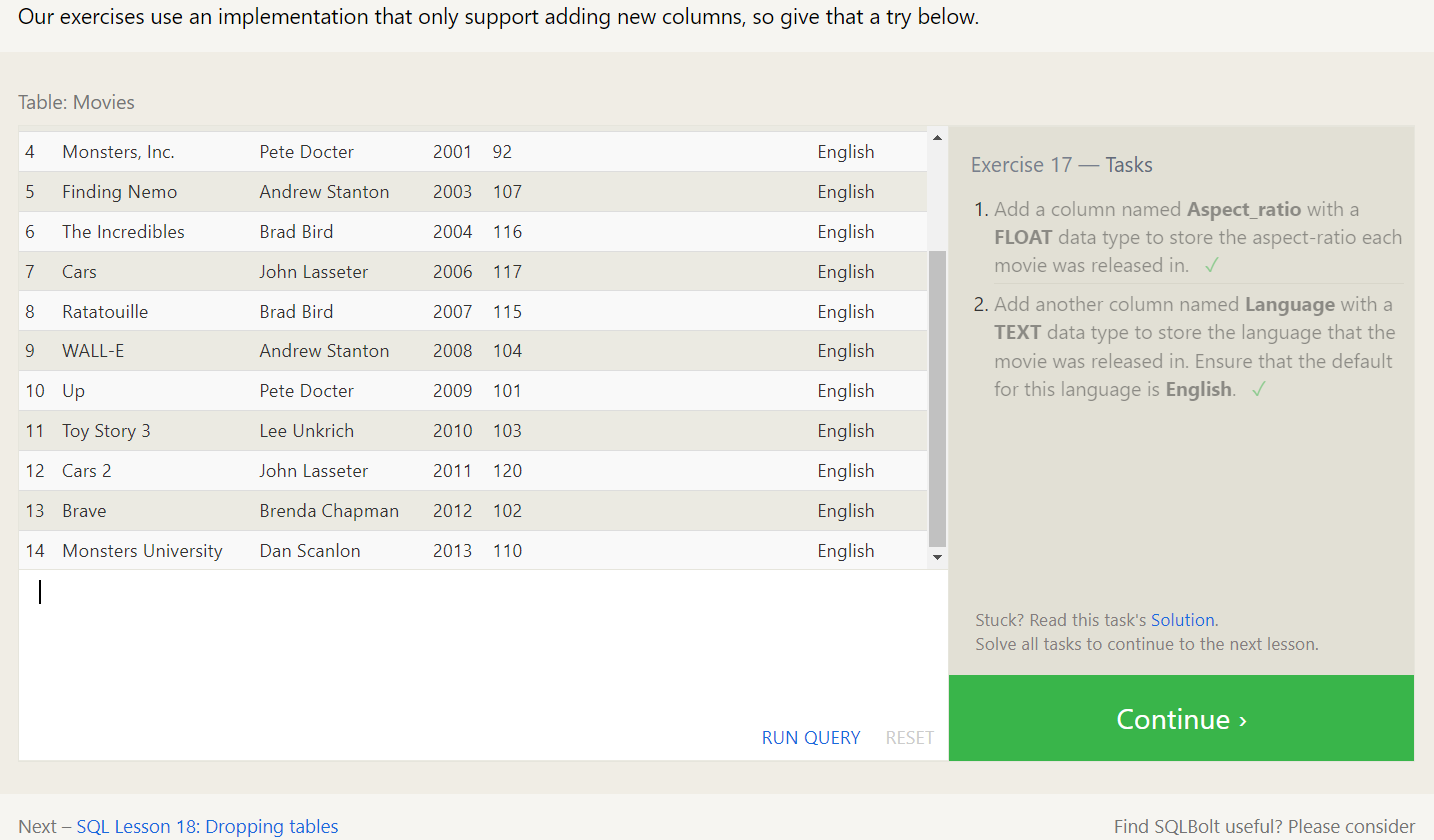
**Name TEXT,**

**Version FLOAT,**

**Download\_count INT**

**);**

# **SQL Lesson 17: Altering tables**



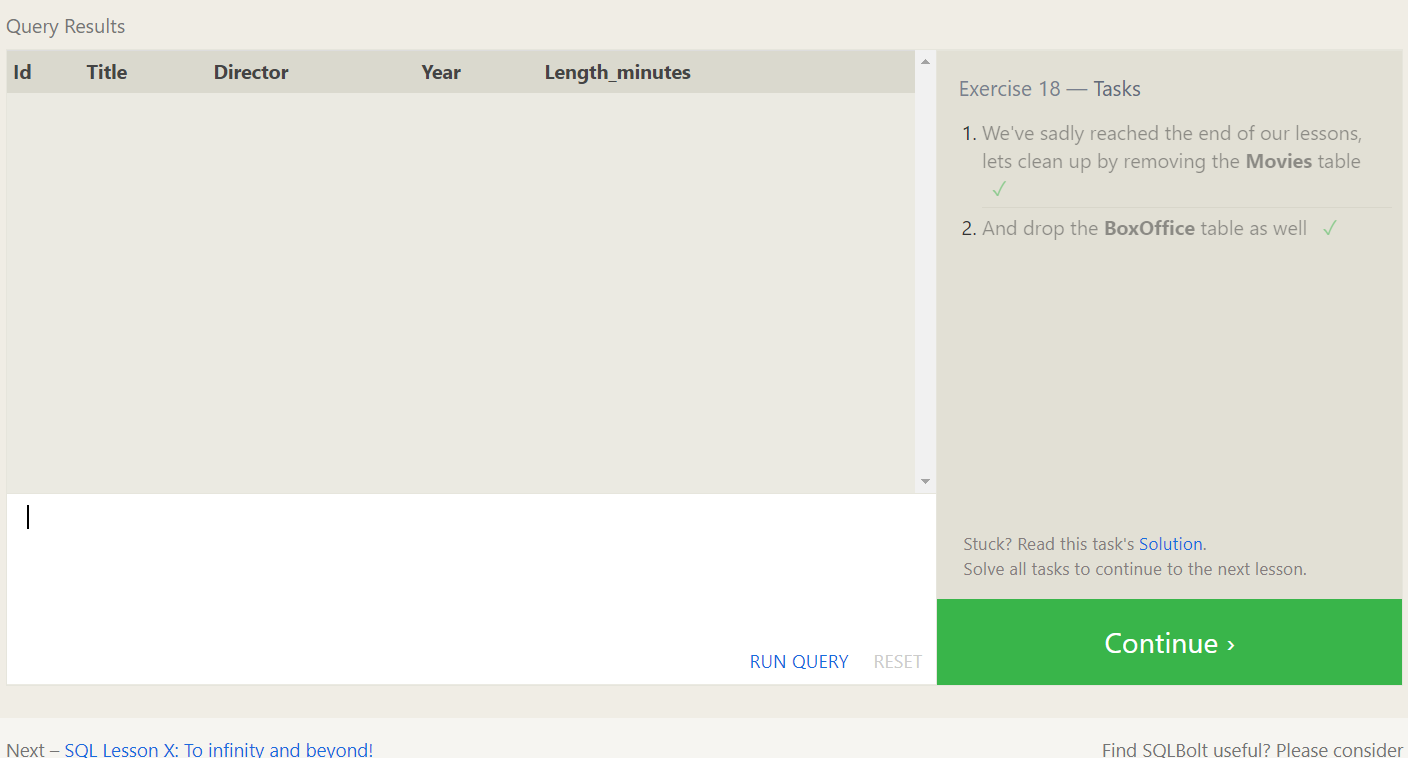
**1.ALTER TABLE Movies**

**ADD COLUMN Aspect\_ratio FLOAT;**

**2.ALTER TABLE Movies**

**ADD COLUMN Language TEXT DEFAULT 'English';**

# **SQL Lesson 18: Dropping tables**



**1.DROP TABLE IF EXISTS Movies;**

**2.DROP TABLE IF EXISTS BoxOffice;**

