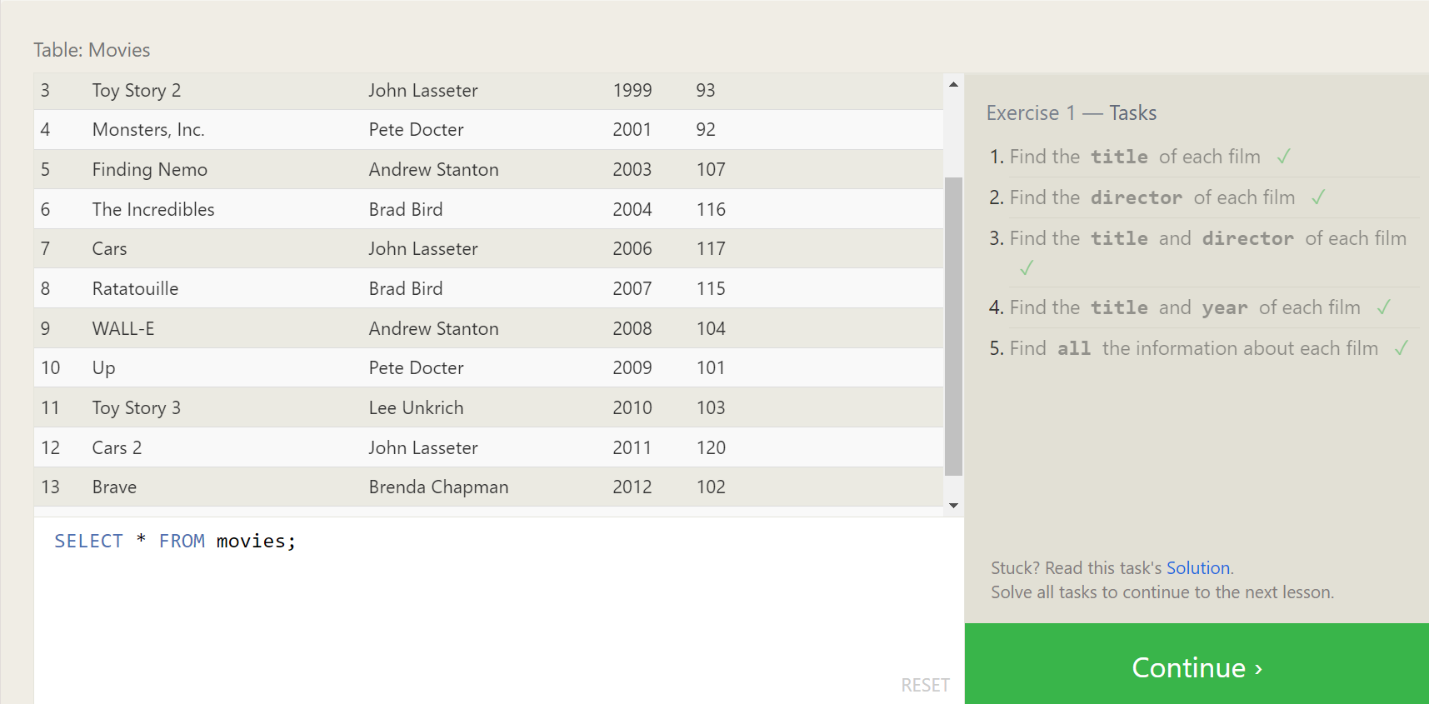
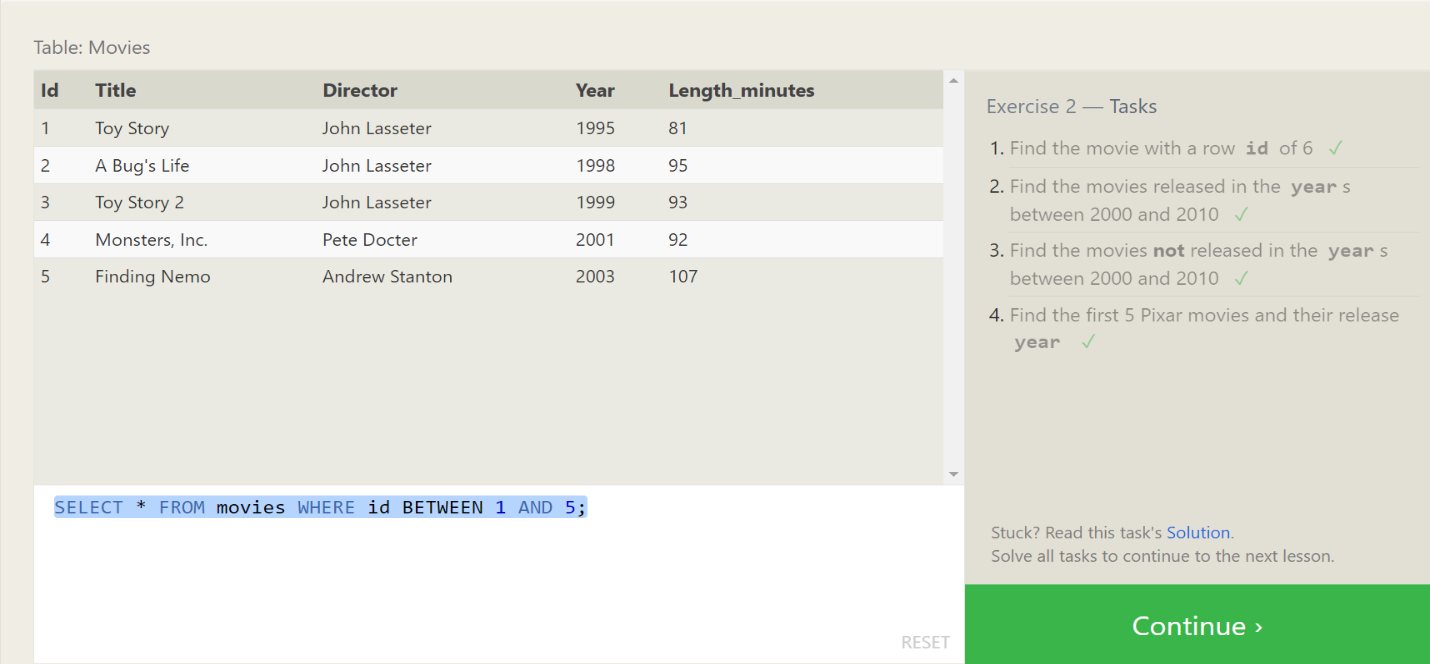
1. **SQL Lesson 1 SELECT queries 101:**

****

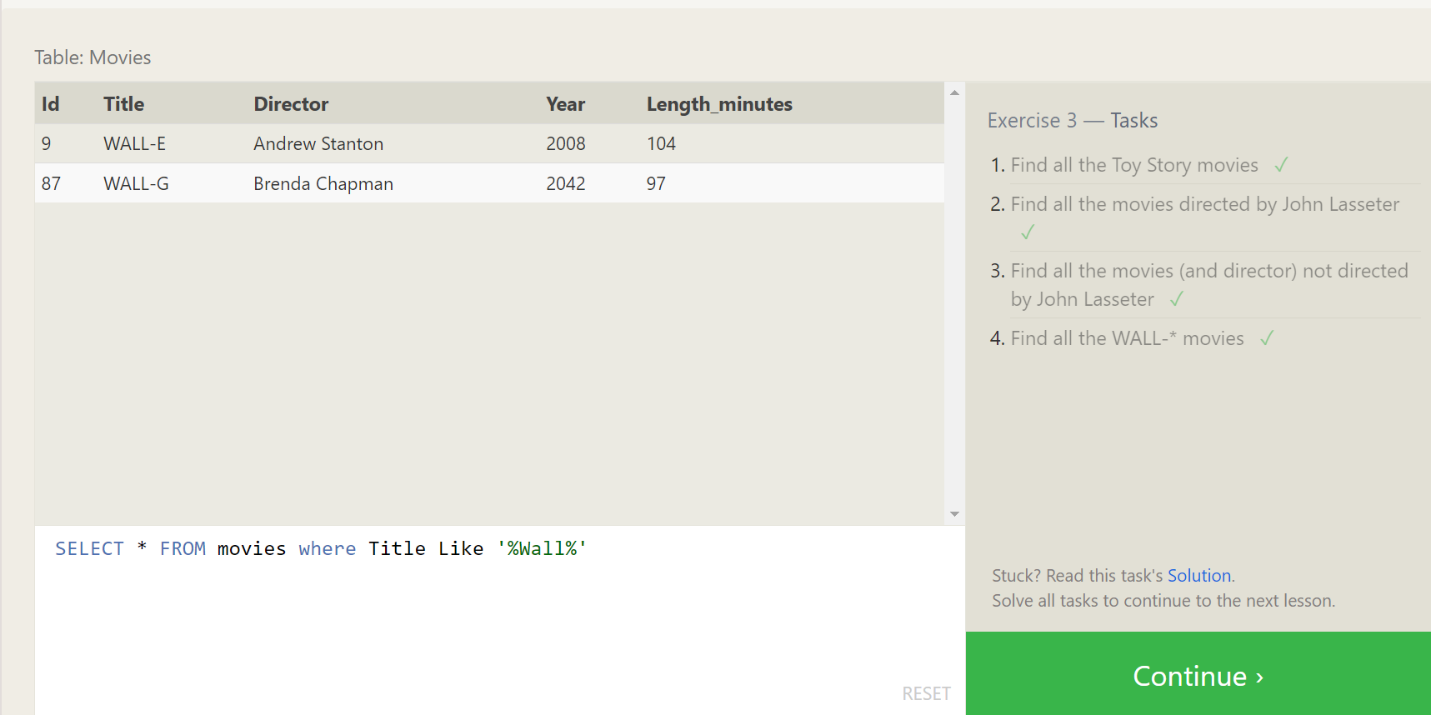
**Answer:**

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



**Answer:**

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 BETWEEN 1 AND 5;
5. **SQL Lesson 3: Queries with constraints (Pt. 2)**



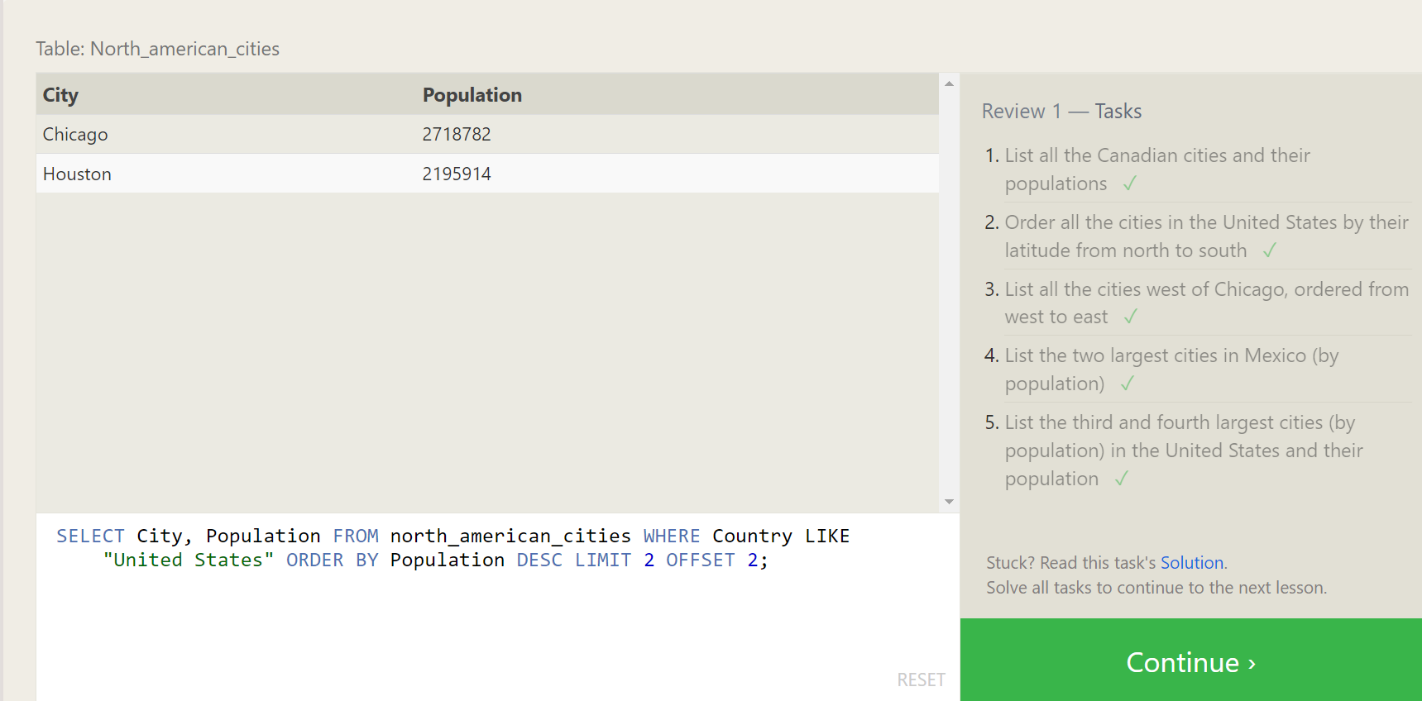
**Answer:**

1. SELECT \* FROM movies where Title Like '%Toy Story%';
2. SELECT \* FROM movies where Director Like '%john Lasseter%';
3. SELECT \* FROM movies where Director NOT Like '%john Lasseter%';
4. SELECT \* FROM movies where Title Like '%Wall%';
5. **SQL Lesson 4: Filtering and sorting Query results:**

****

**Answer:**

1. SELECT distinct Director FROM movies ORDER BY Director ASC;
2. SELECT Title, Year FROM movies ORDER BY year DESC LIMIT 4;
3. SELECT Title FROM Movies ORDER BY Title ASC LIMIT 5;
4. SELECT Title FROM Movies ORDER BY Title ASC LIMIT 5 OFFSET 5;
5. **SQL Review: Simple SELECT Queries:**

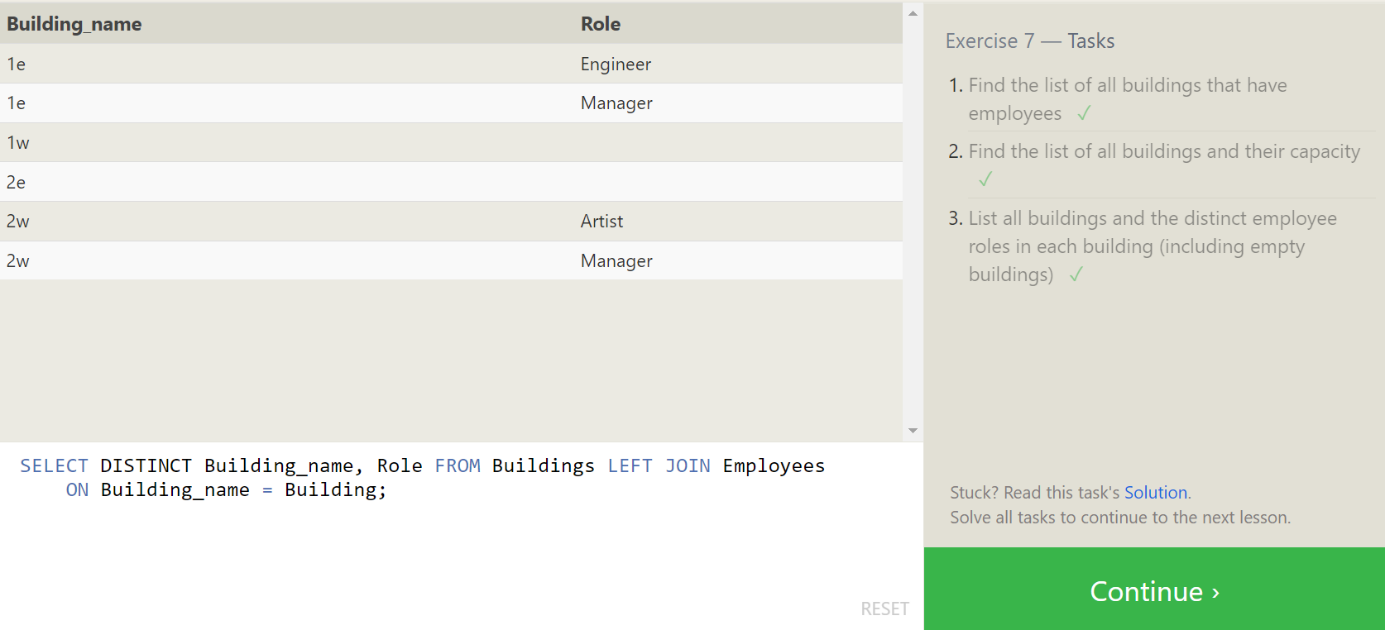
****

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 < -87.629798 ORDER BY longitude ASC;
4. SELECT City, Population FROM north\_american\_cities WHERE Country LIKE "Mexico" ORDER BY Population DESC LIMIT 2;
5. SELECT City, Population FROM north\_american\_cities WHERE Country LIKE "United States" ORDER BY Population DESC LIMIT 2 OFFSET 2;
6. **SQL Lesson 6: Multi-table queries with JOINs:**

****

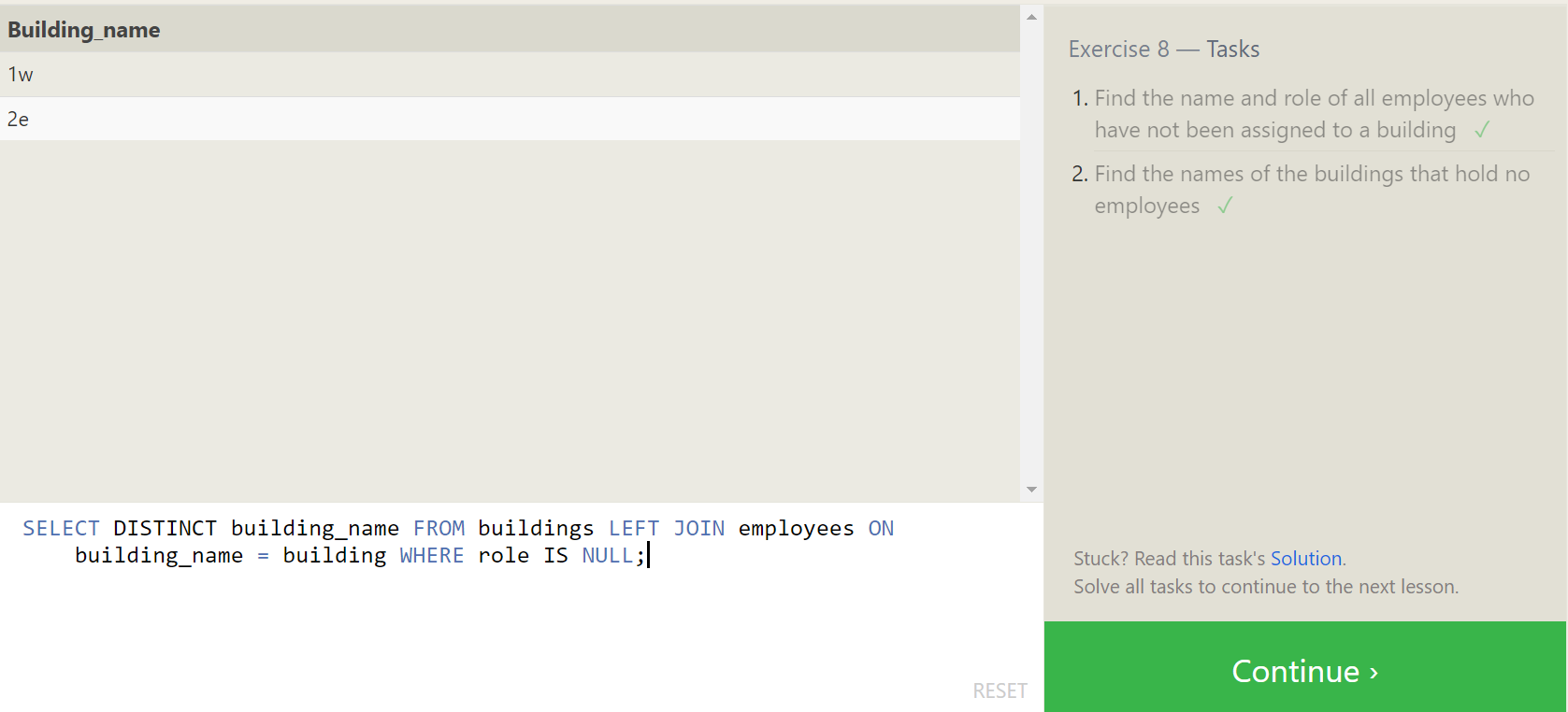
**Answer:**

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, Rating FROM Movies JOIN Boxoffice ON Movies.id = Boxoffice.Movie\_id ORDER BY Rating DESC;
4. **SQL Lesson 7: OUTER JOINs:**



**Answer:**

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;
4. **SQL Lesson 8: A short note on NULLs:**



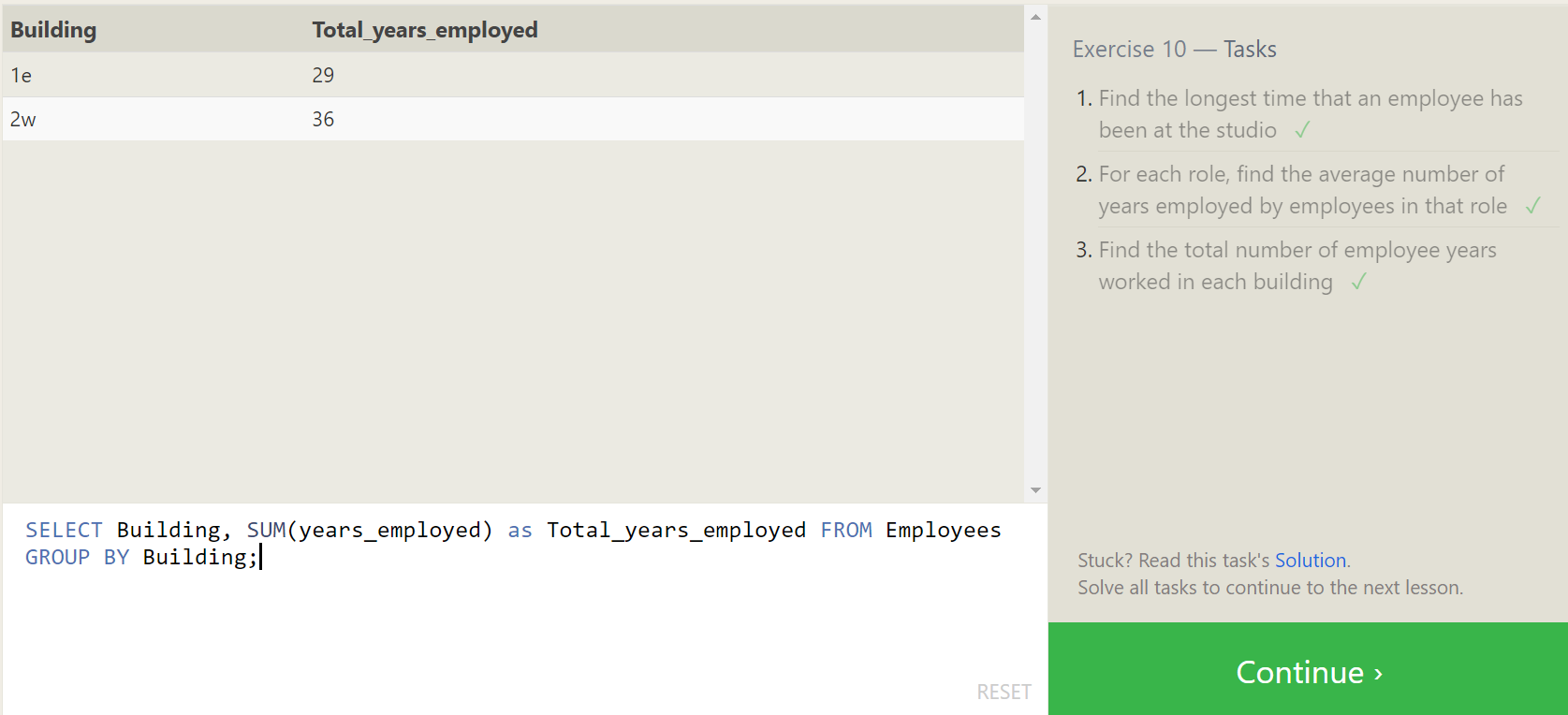
**Answer:**

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 NULL;
3. **SQL Lesson 9: Queries with expressions:**



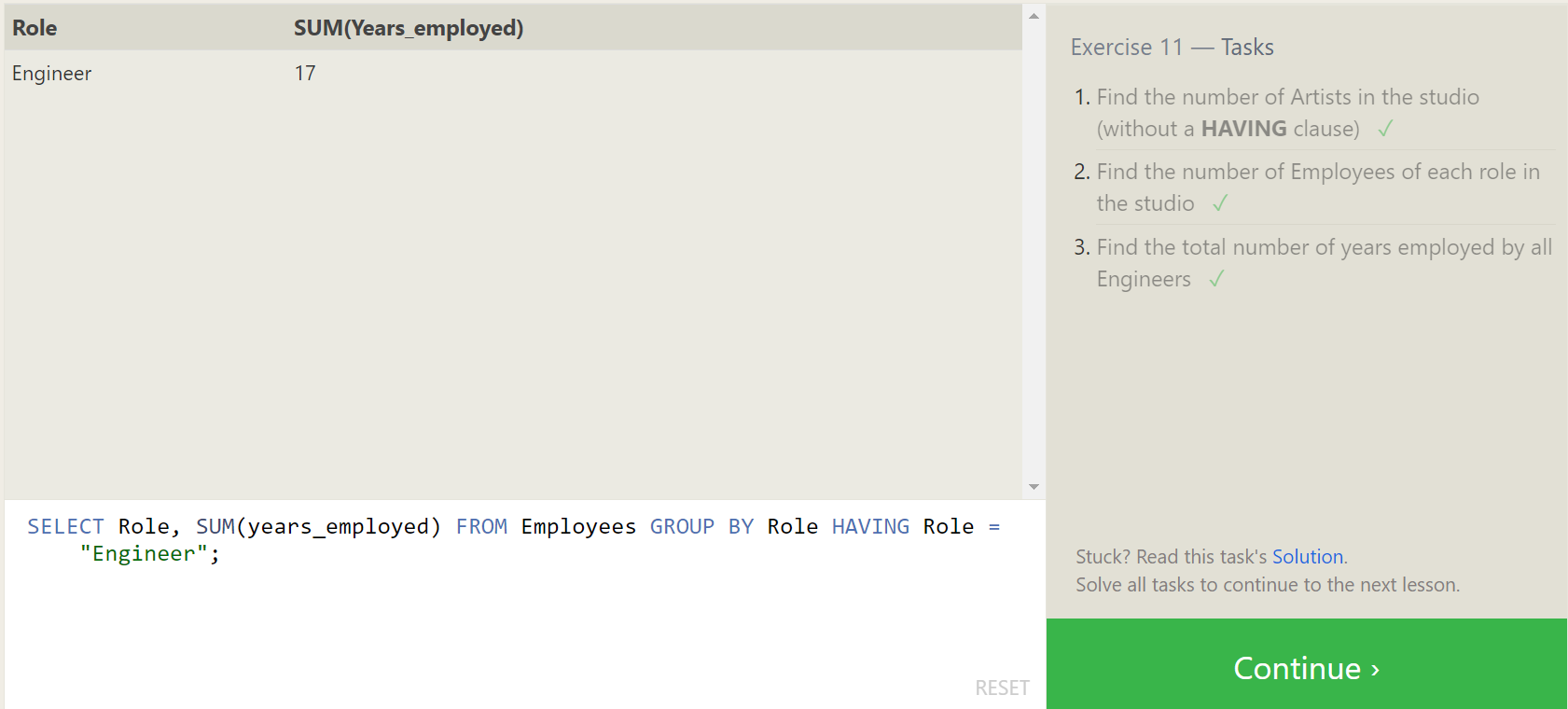
**Answer:**

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



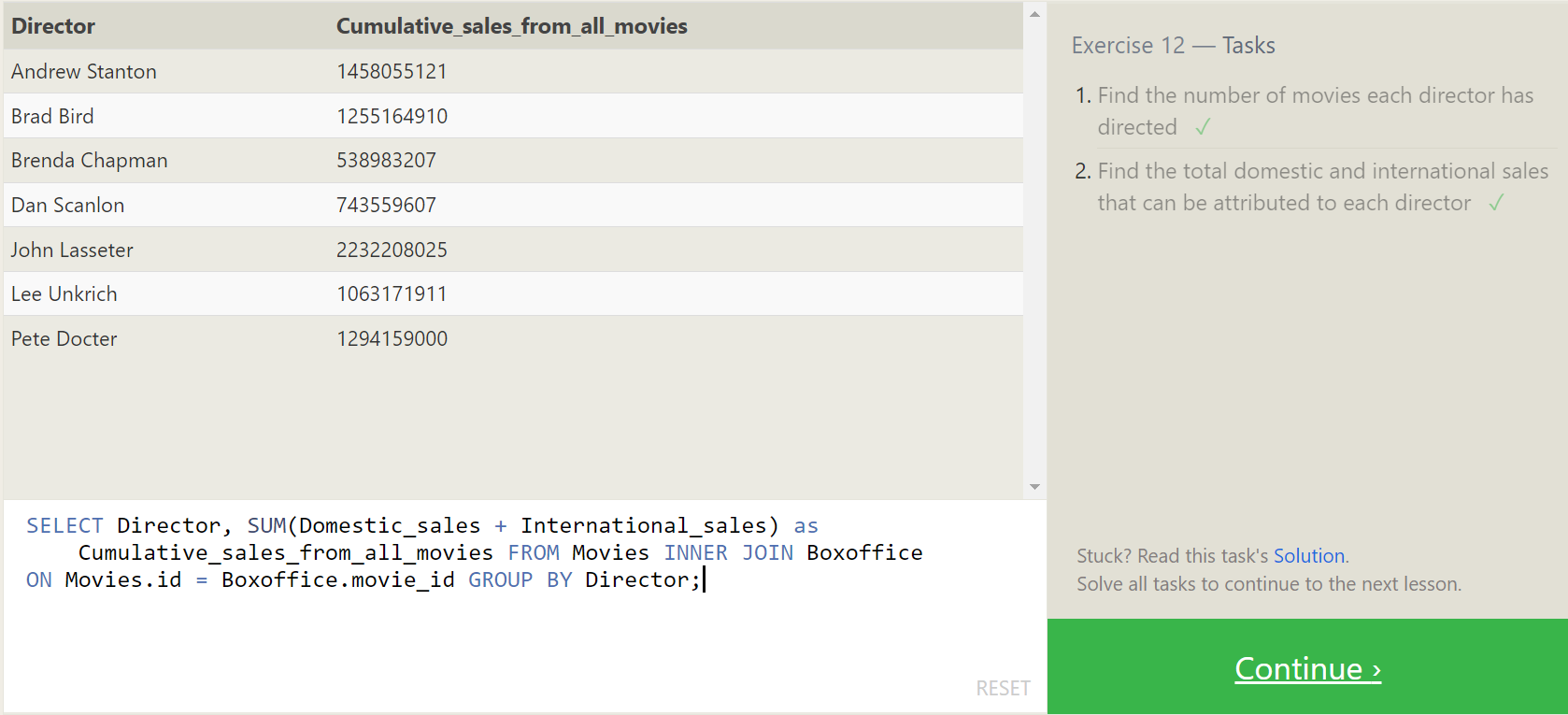
**Answer:**

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

****

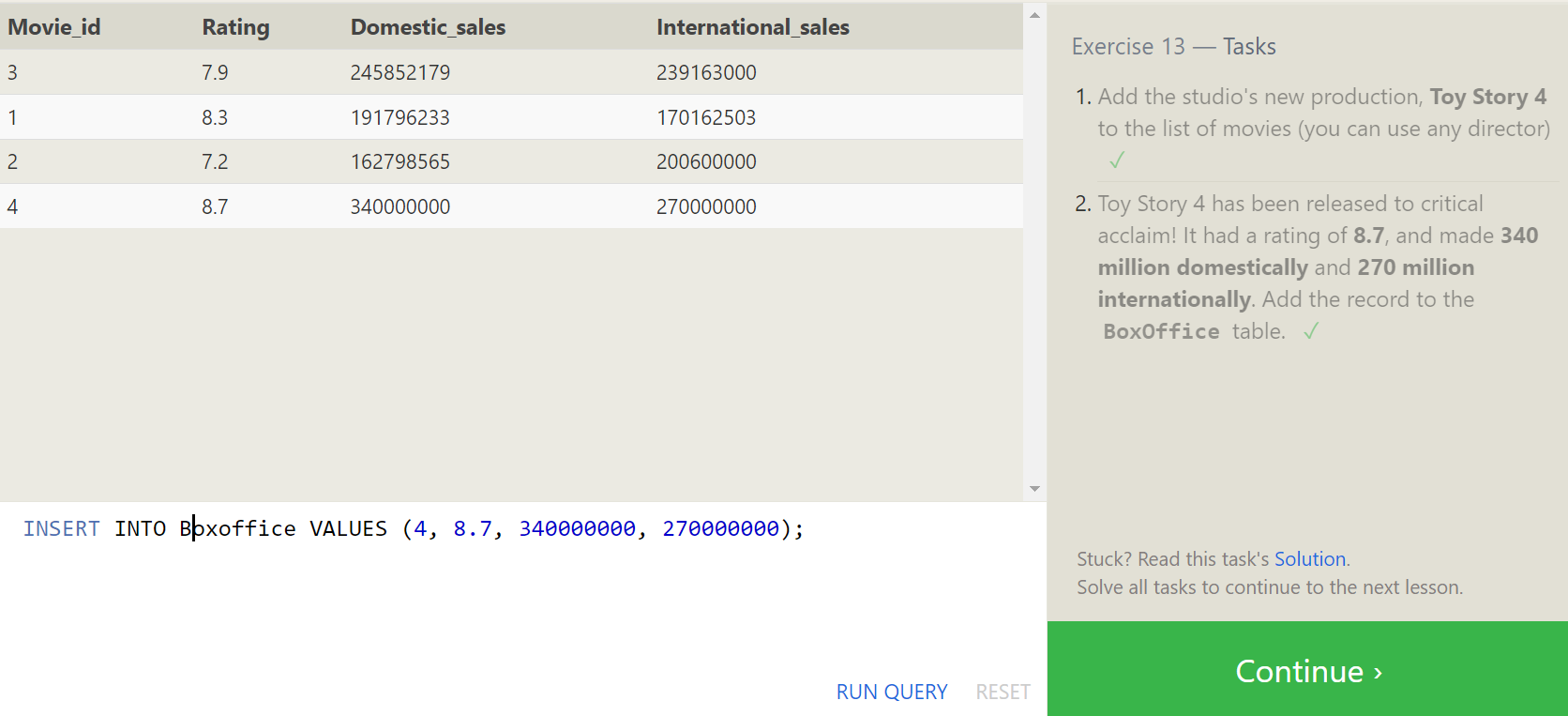
**Answer:**

1. SELECT Role, COUNT(\*) as Number\_of\_artists FROM Employees WHERE Role = "Artist";
2. SELECT Role, COUNT(\*)FROM Employees GROUP BY Role;
3. SELECT Role, SUM(years\_employed) FROM Employees GROUP BY Role HAVING Role = "Engineer";
4. **SQL Lesson 12: Order of execution of a Query:**

****

**Answer:**

1. SELECT Director, COUNT(id) as Num\_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;
3. **SQL Lesson 13: Inserting rows:**

****

**Answer:**

1. INSERT INTO Movies VALUES (4, "Toy Story 4", "El Directore", 2015, 90);
2. INSERT INTO Boxoffice VALUES (4, 8.7, 340000000, 270000000);
3. **SQL Lesson 14: Updating rows:**

****

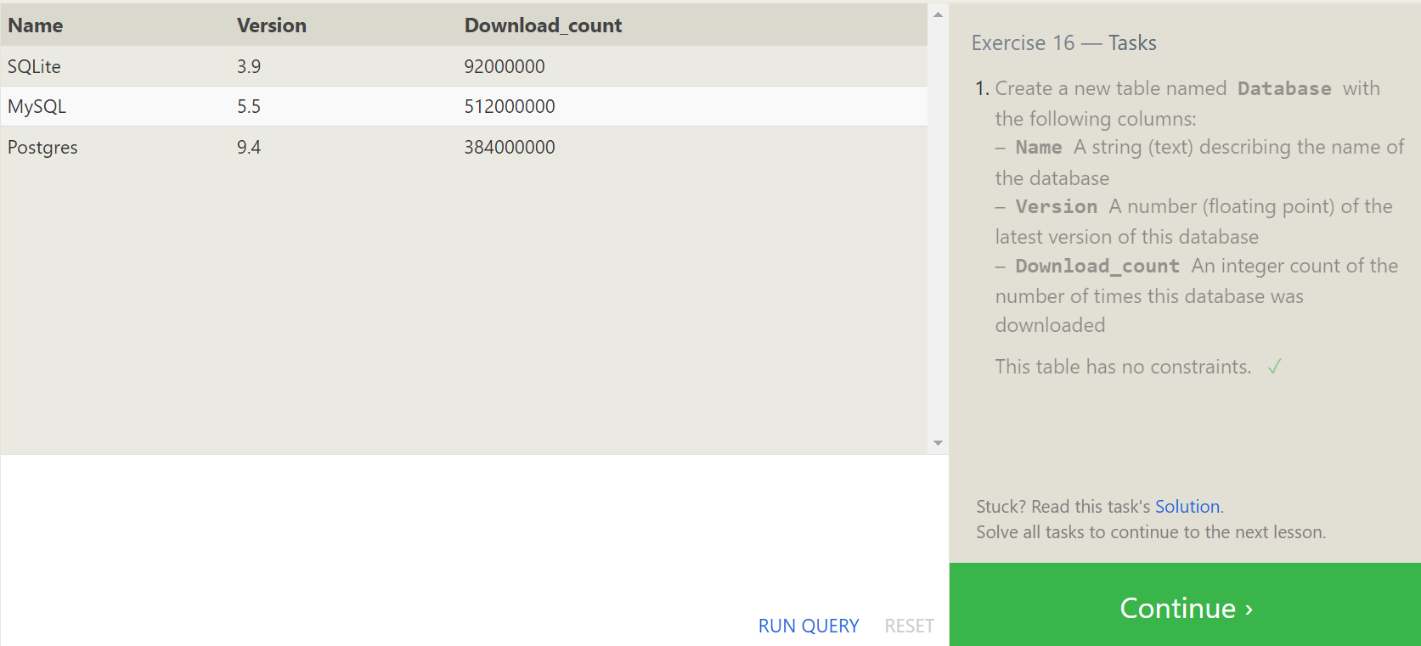
**Answer:**

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;
4. **SQL Lesson 15: Deleting rows:**

****

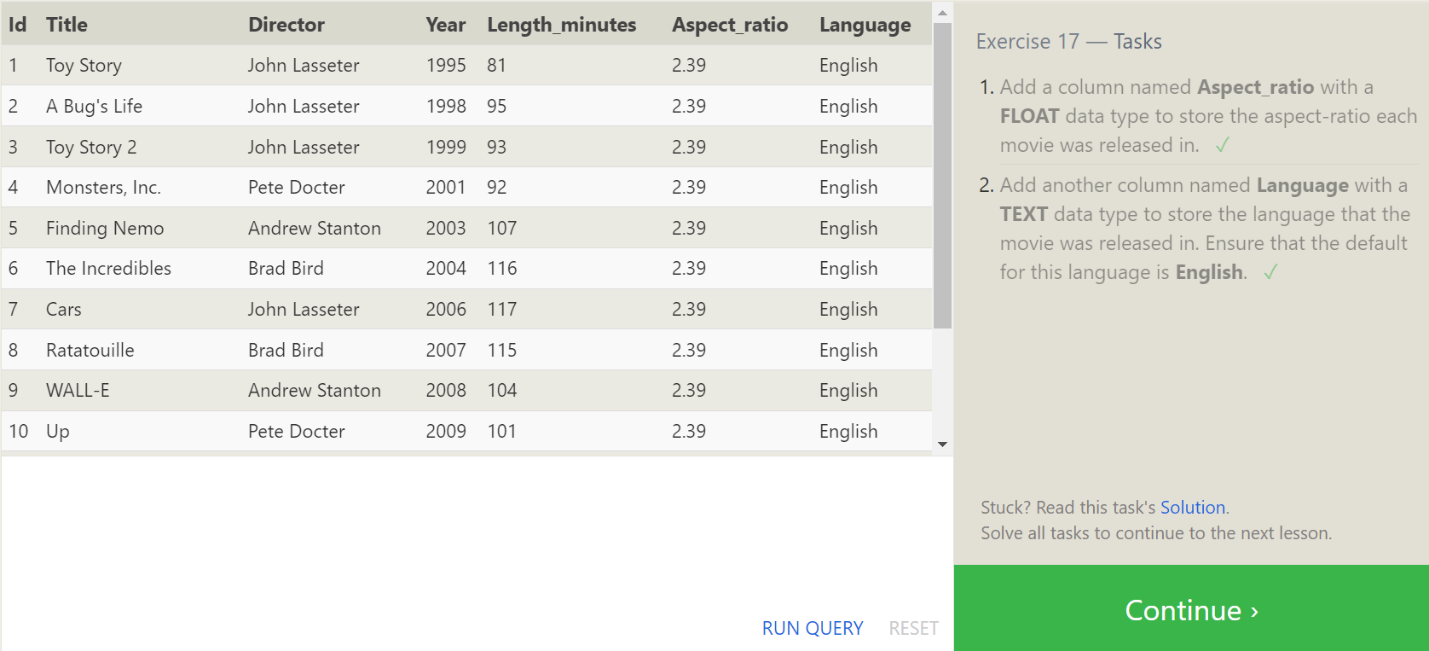
**Answer:**

1. DELETE FROM Movies where Year < 2005;
2. DELETE FROM Movies where Director = "Andrew Stanton";
3. **SQL Lesson 16: Creating tables:**

****

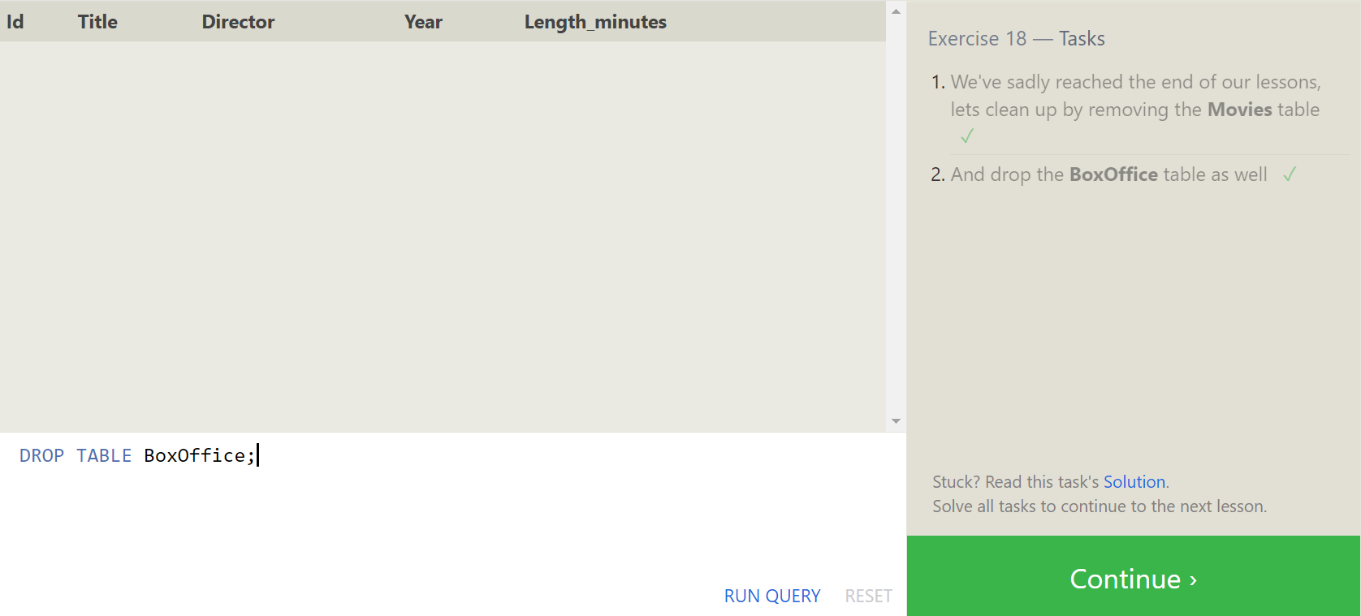
**Answer:**

1. CREATE TABLE Database (Name TEXT, Version FLOAT,Download\_count INTEGER);
2. **SQL Lesson 17: Altering tables:**

****

**Answer:**

1. ALTER TABLE Movies ADD COLUMN Aspect\_ratio FLOAT DEFAULT 2.39;
2. ALTER TABLE Movies ADD COLUMN Language TEXT DEFAULT "English";
3. **SQL Lesson 18: Dropping tables:**

****

**Answer:**

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