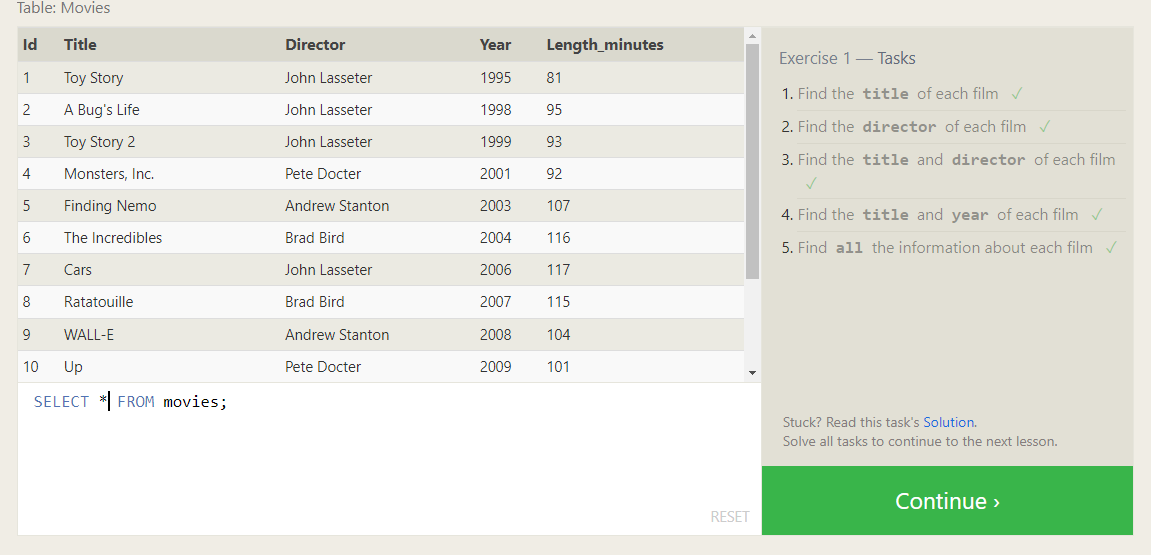
**Road-Map: Day-33: Database: MySQL – Day-1: Tasks**

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

Answers:

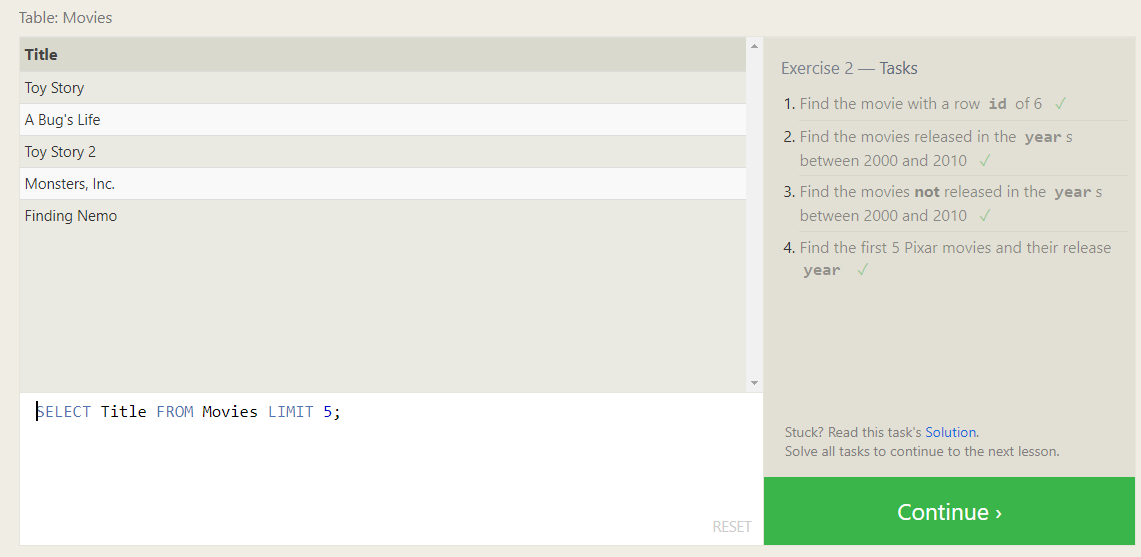
1. SELECT Title FROM Movies;
2. SELECT Director FROM Movies;
3. SELECT Title, Director FROM Movies;
4. SELECT Title, Year FROM Movies;
5. SELECT \* FROM Movies;



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

Answers:

1. SELECT \* FROM Movies where id=6;
2. SELECT Title FROM Movies WHERE Year BETWEEN 2000 AND 2010;
3. SELECT Title FROM Movies WHERE Year NOT BETWEEN 2000 AND 2010;
4. SELECT Title FROM Movies LIMIT 5;



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

Answers:

1. SELECT Title FROM Movies WHERE Title LIKE "%Toy%";
2. SELECT Title FROM Movies WHERE Director = "John Lasseter";
3. SELECT Title FROM Movies WHERE Director != "John Lasseter";
4. SELECT Title FROM Movies WHERE Title LIKE "%WALL%";



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

Answers :

1. SELECT DISTINCT Director FROM Movies ORDER BY Director;
2. SELECT Title 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;



**SQL Review: Simple SELECT Queries :**

Answers :

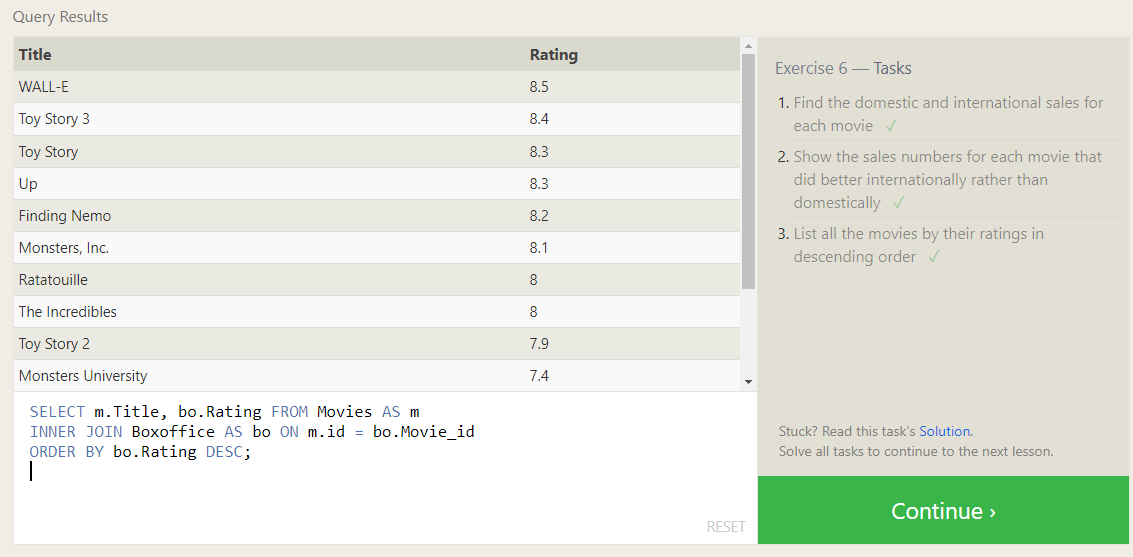
1. SELECT City, Population FROM north\_american\_cities WHERE Country = "Canada";
2. SELECT City FROM north\_american\_cities WHERE Country="United States" ORDER BY Latitude DESC;
3. SELECT City FROM north\_american\_cities WHERE Longitude < -87.629798 ORDER BY Longitude;
4. SELECT City FROM north\_american\_cities WHERE Country="Mexico" ORDER BY Population DESC LIMIT 2;
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 :**

Answers :

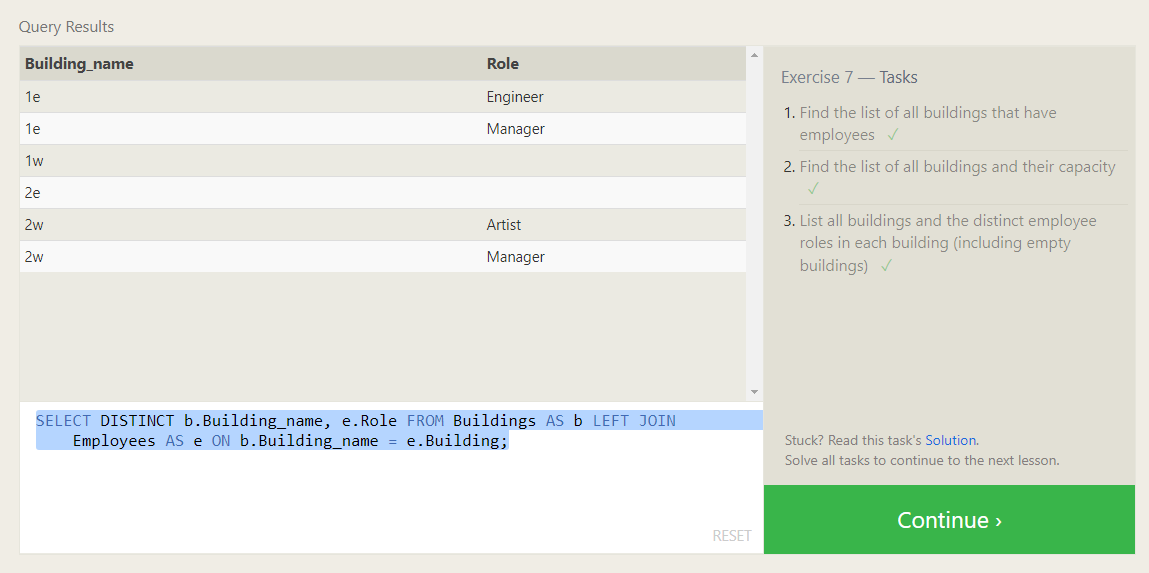
1. SELECT Title, Domestic\_sales, International\_sales FROM Movies INNER JOIN Boxoffice ON Movies.id = Boxoffice.Movie\_id;
2. SELECT m.Title, bo.Domestic\_sales, bo.International\_sales FROM Movies AS m INNER JOIN Boxoffice AS bo ON m.id = bo.Movie\_id WHERE bo.International\_sales > bo.Domestic\_sales;
3. SELECT m.Title, bo.Rating FROM Movies AS m INNER JOIN Boxoffice AS bo ON m.id = bo.Movie\_id ORDER BY bo.Rating DESC;



**SQL Lesson 7: OUTER JOINs :**

Answers :

1. SELECT DISTINCT building FROM employees;
2. SELECT Building\_name, Capacity FROM Buildings;
3. SELECT DISTINCT b.Building\_name, e.Role FROM Buildings AS b LEFT JOIN Employees AS e ON b.Building\_name = e.Building;

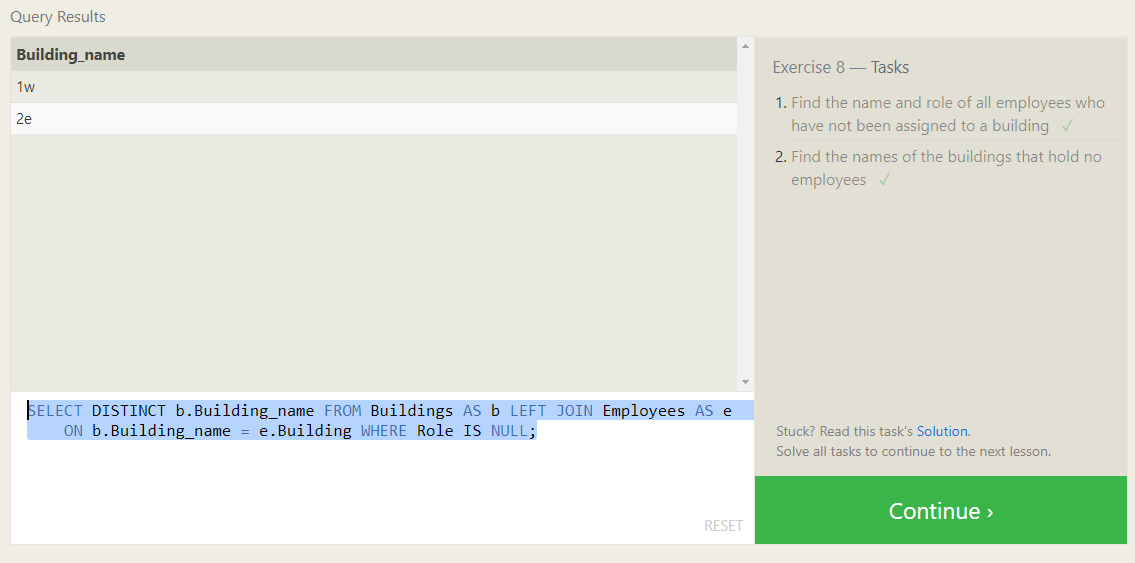


**SQL Lesson 8: A short note on NULLs :**

Answers :

1. SELECT e.Name, e.Role FROM employees AS e LEFT JOIN Buildings AS b ON e.Building = B.Building\_name WHERE B.Building\_name IS NULL ;
2. SELECT DISTINCT b.Building\_name FROM Buildings AS b LEFT JOIN Employees AS e

ON b.Building\_name = e.Building WHERE Role IS NULL;



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

Answers :

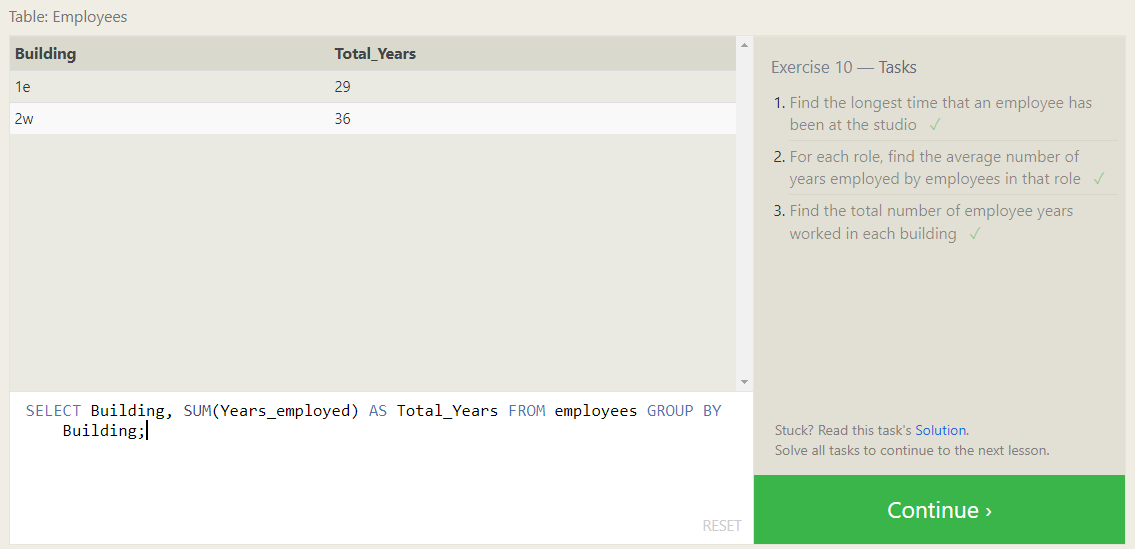
1. SELECT DISTINCT m.Title, (b.Domestic\_sales+b.International\_sales )/1000000 AS Millions from Movies as m JOIN Boxoffice AS b ON m.Id = b.Movie\_id ORDER BY Millions DESC;
2. SELECT m.Title, (b.Rating )\*10 AS Rating from Movies as m JOIN Boxoffice AS b ON m.Id = b.Movie\_id ORDER BY b.Rating DESC;
3. SELECT m.Title, m.Year from Movies as m WHERE (M.YEAR%2)==0;



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

Answers :

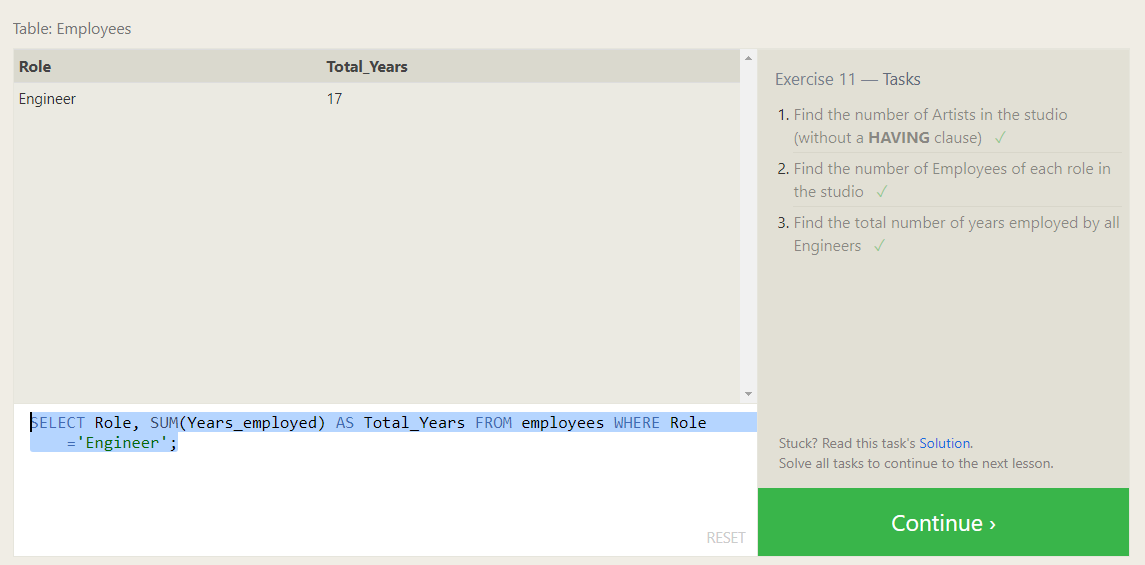
1. SELECT Name,MAX(Years\_employed) FROM Employees;
2. SELECT Role, AVG(Years\_employed) as Average\_years\_employed FROM Employees GROUP BY Role;
3. SELECT Building, SUM(Years\_employed) AS Total\_Years FROM employees GROUP BY Building;



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

Answers :

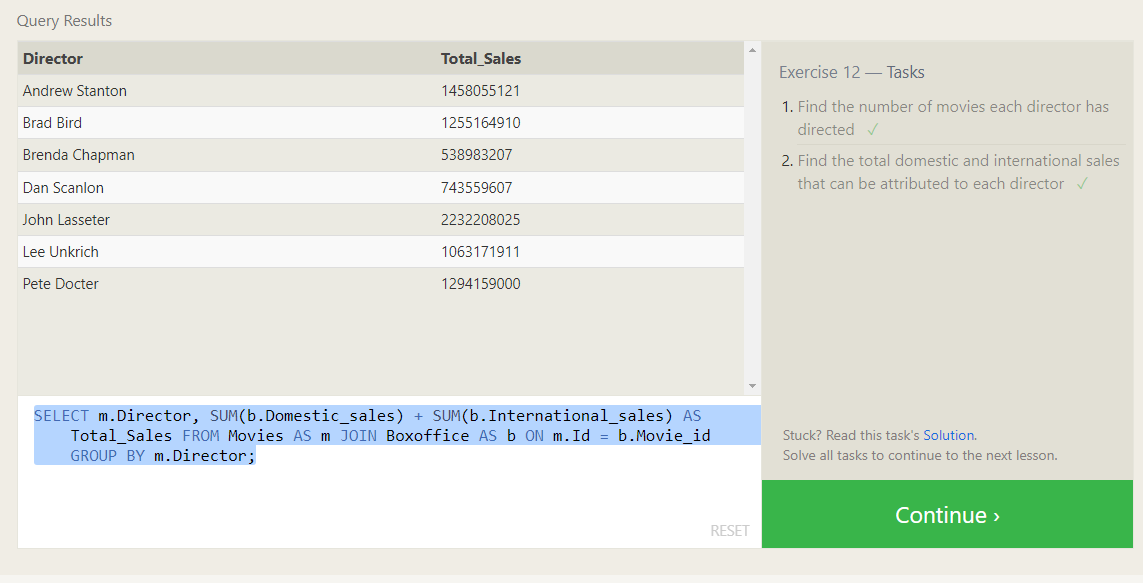
1. SELECT Role, COUNT(Role) AS Artist\_Count From employees WHERE Role = 'Artist';
2. SELECT Role, COUNT(Role) AS Total\_Employees FROM Employees GROUP BY Role;
3. SELECT Role, SUM(Years\_employed) AS Total\_Years FROM employees WHERE Role='Engineer';



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

Answers :

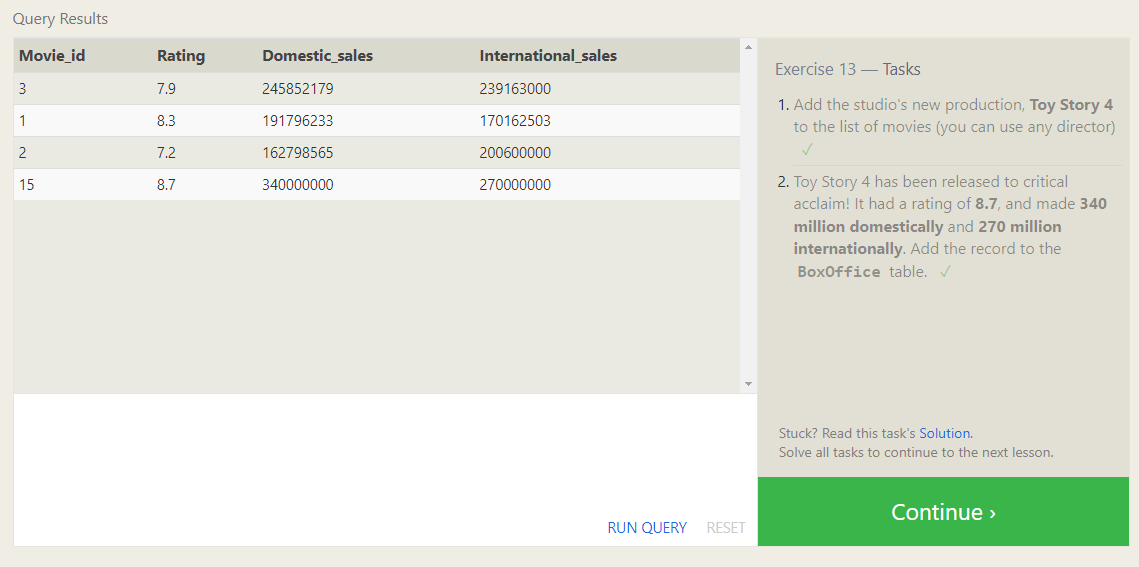
1. SELECT Director, COUNT(Director) AS Total\_Movies FROM movies GROUP BY Director;
2. SELECT m.Director, SUM(b.Domestic\_sales) + SUM(b.International\_sales) AS Total\_Sales FROM Movies AS m JOIN Boxoffice AS b ON m.Id = b.Movie\_id GROUP BY m.Director;



**SQL Lesson 13: Inserting rows :**

Answers :

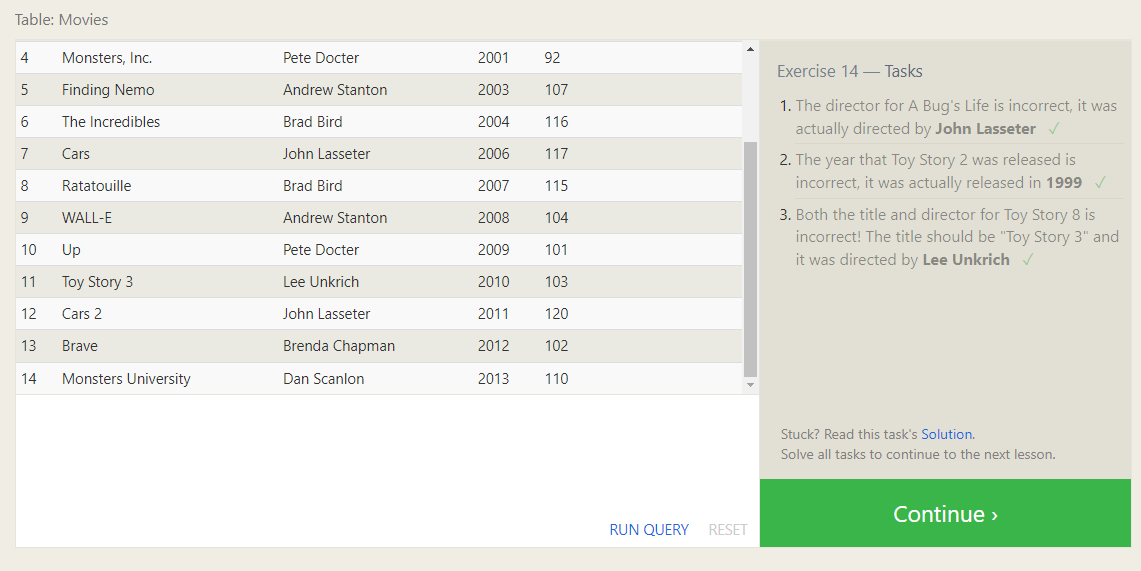
1. INSERT INTO Movies VALUES(15, 'Toy Story 4', 'John Lasseter', 2002, 90);
2. INSERT INTO BoxOffice VALUES(15, 8.7, 340000000, 270000000);



**SQL Lesson 14: Updating rows :**

Answers :

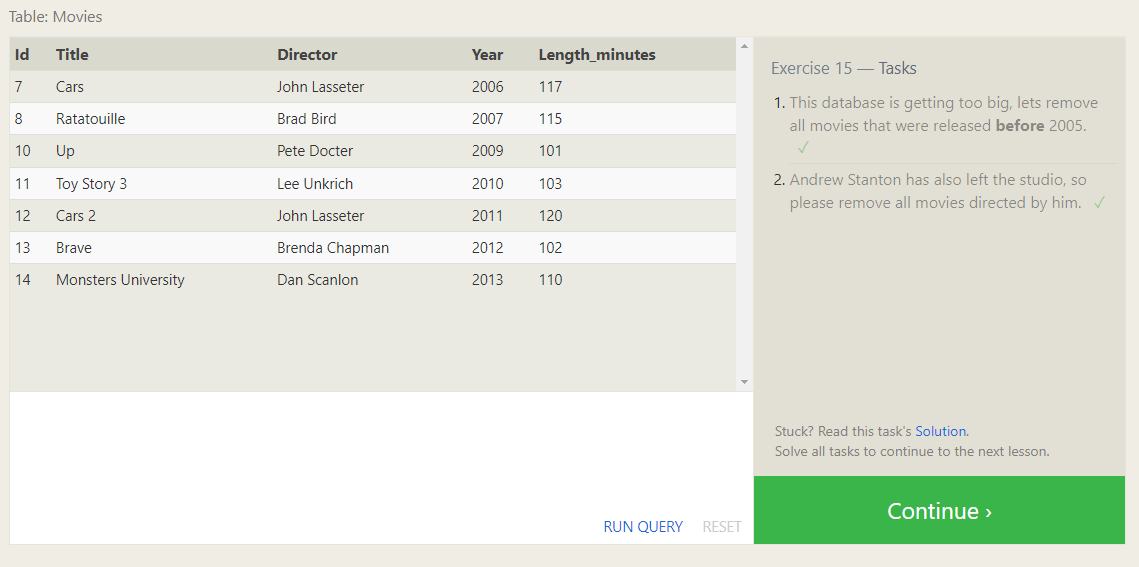
1. UPDATE movies SET director = "John Lasseter" WHERE Title = "A Bug's Life";
2. UPDATE movies SET Year = 1999 WHERE Title = "Toy Story 2";
3. UPDATE movies SET Title = "Toy Story 3", Director = "Lee Unkrich" WHERE ID = 11;



**SQL Lesson 15: Deleting rows :**

Answers :

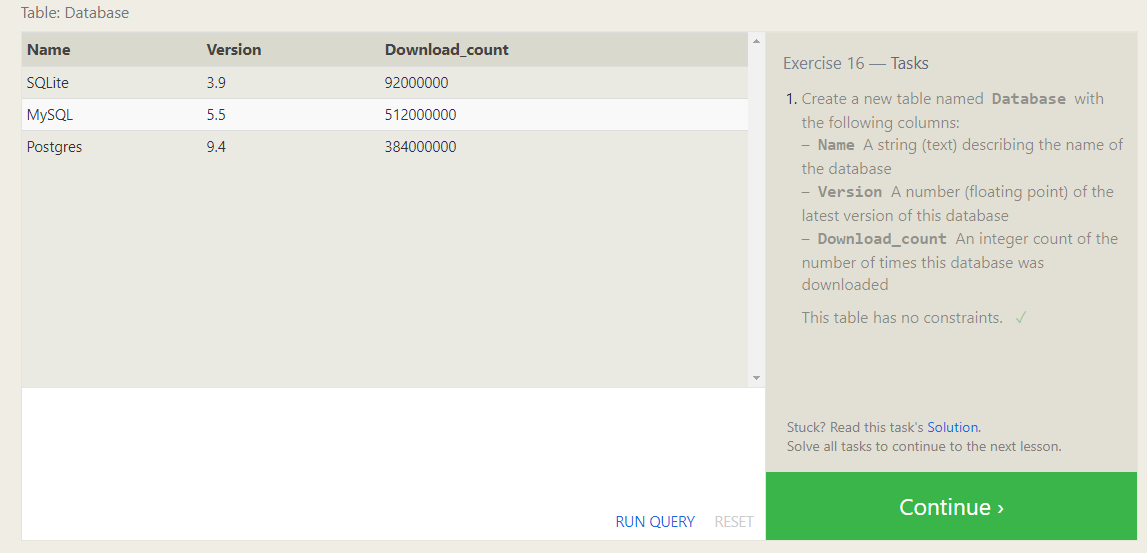
1. DELETE FROM Movies WHERE Year < 2005;
2. DELETE FROM Movies WHERE Director = 'Andrew Stanton';



**SQL Lesson 16: Creating tables :**

Answers :

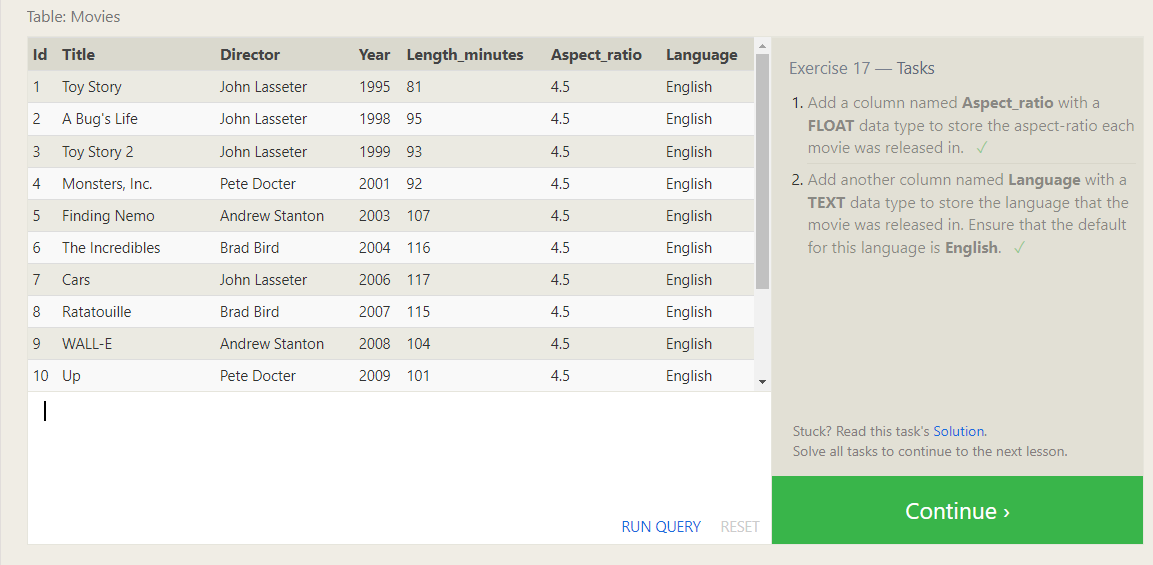
1. CREATE TABLE Database ( Name varchar(20), Version float ,Download\_count int);



**SQL Lesson 17: Altering tables :**

Answers :

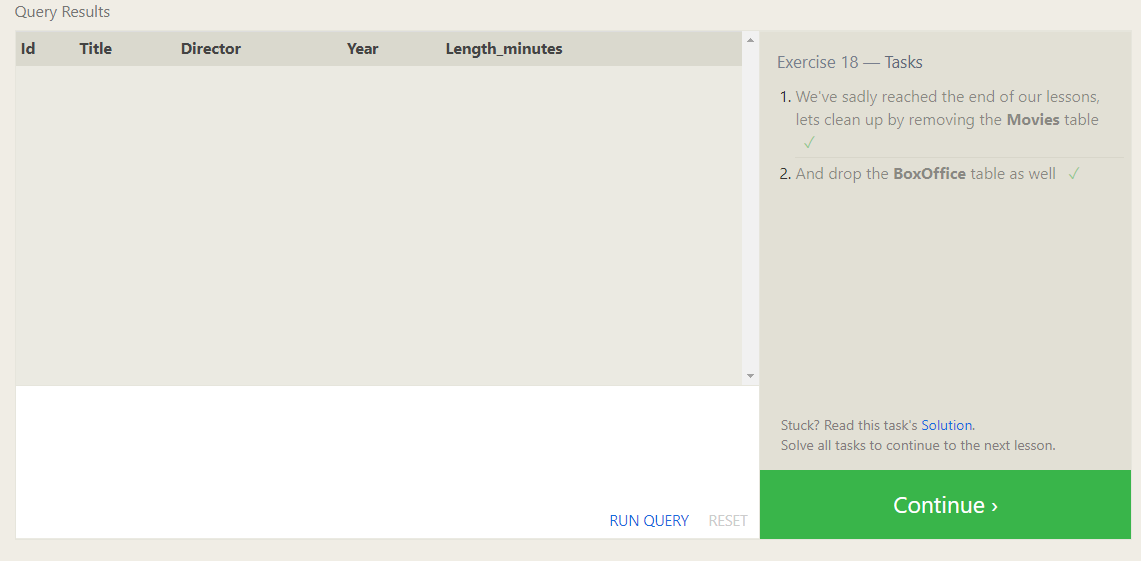
1. ALTER TABLE Movies ADD COLUMN Aspect\_ratio FLOAT DEFAULT 4.5;
2. ALTER TABLE Movies ADD COLUMN Language TEXT DEFAULT "English";

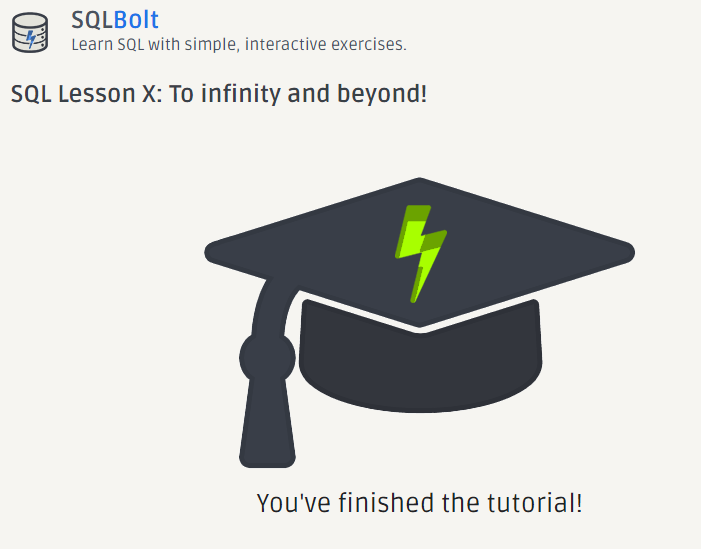


**SQL Lesson 18: Dropping tables :**

Answers :

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





--------------------------------End--------------------------------