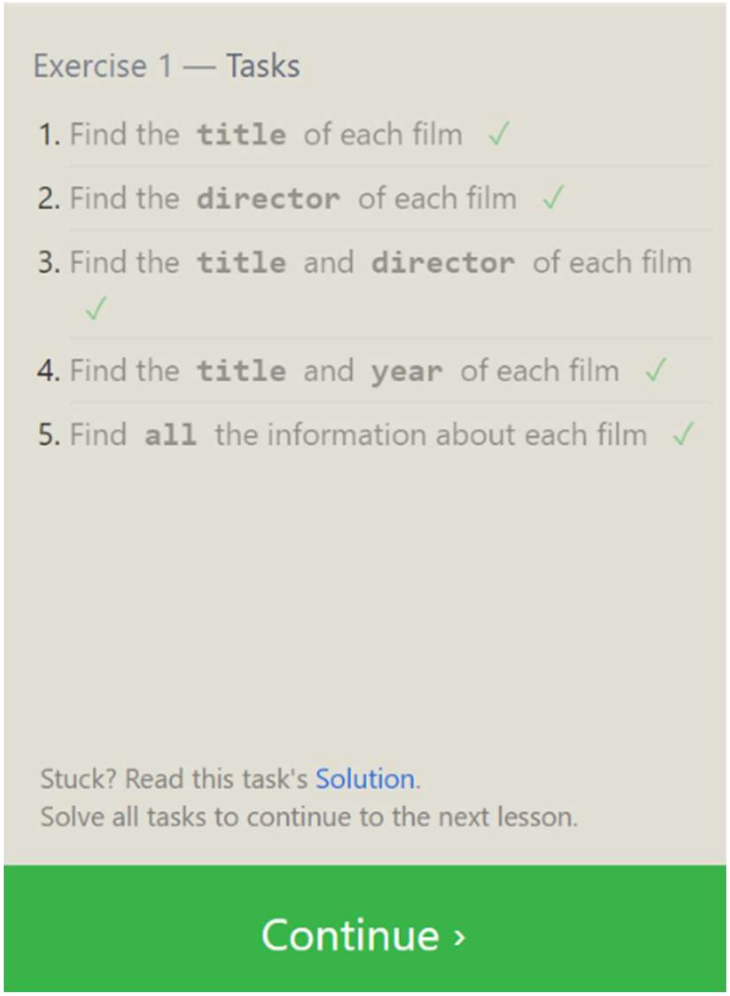
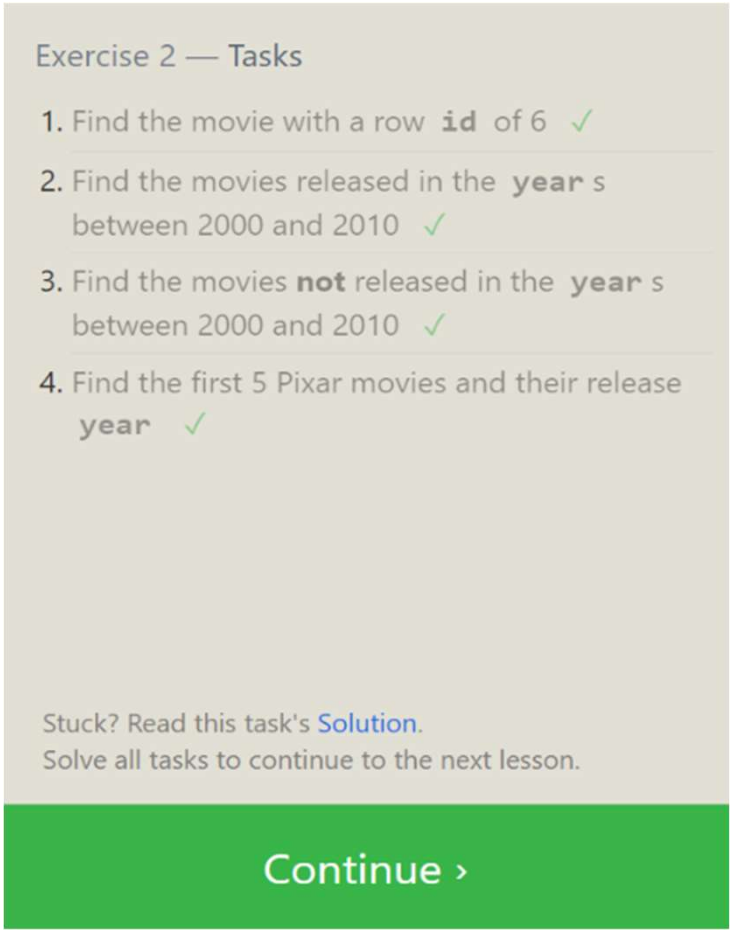
SQL Lesson 1: SELECT queries 101



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;

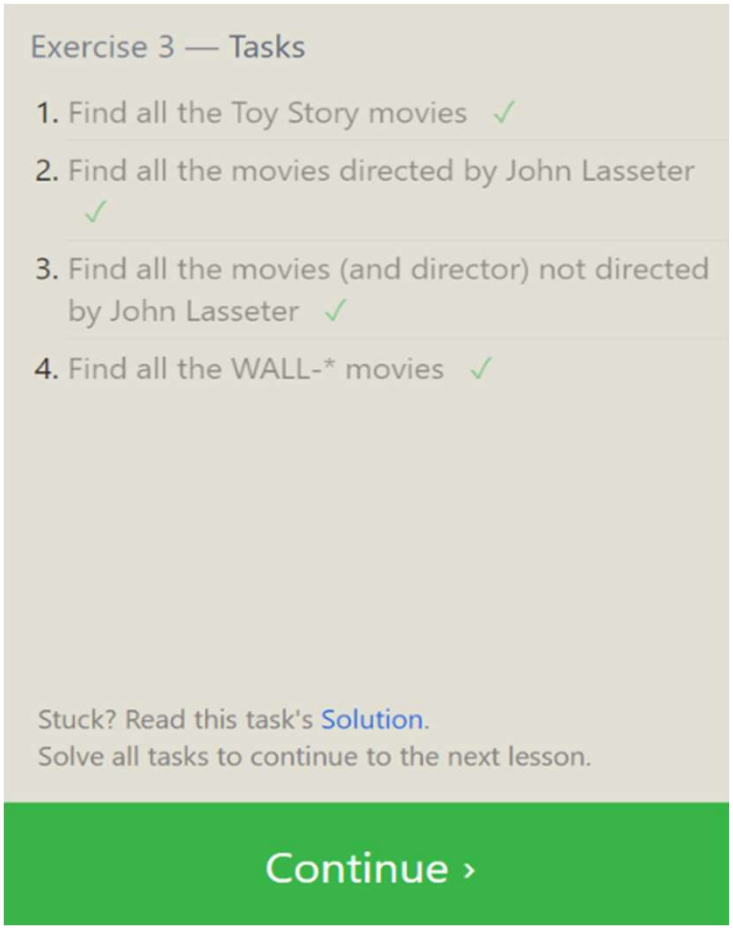
SQL Lesson 2: Queries with constraints (Pt. 1)



Answers:

1. SELECT * FROM movies WHERE id = 6;
2. SELECT * 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;

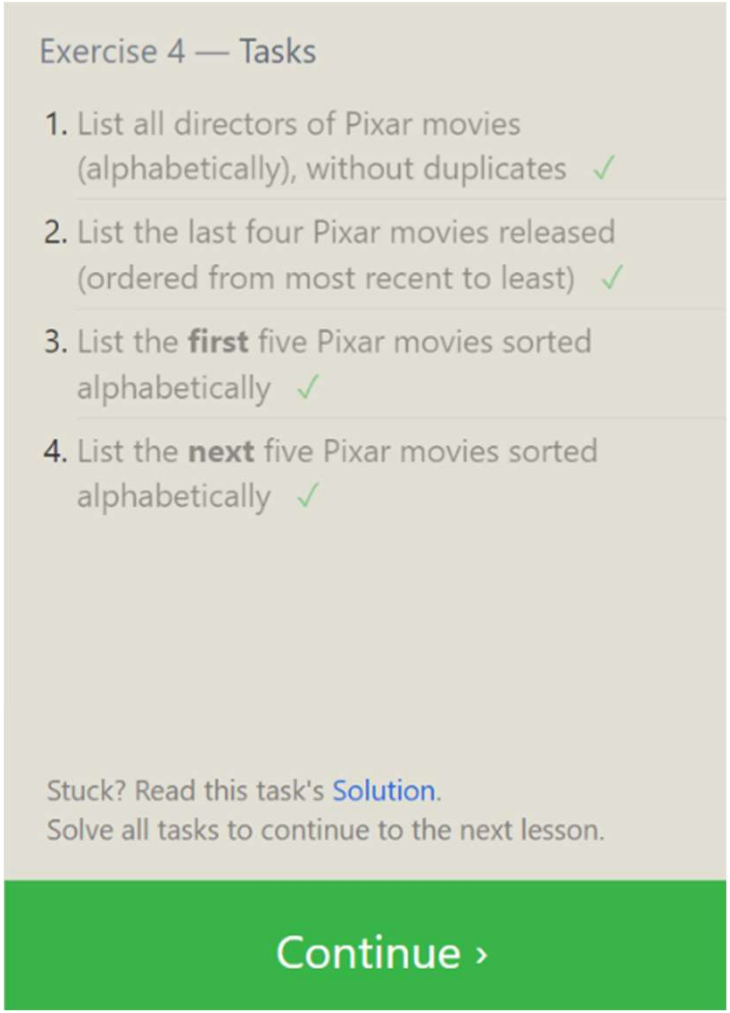
SQL Lesson 3: Queries with Constraints (Pt. 2)



Answers:

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%';

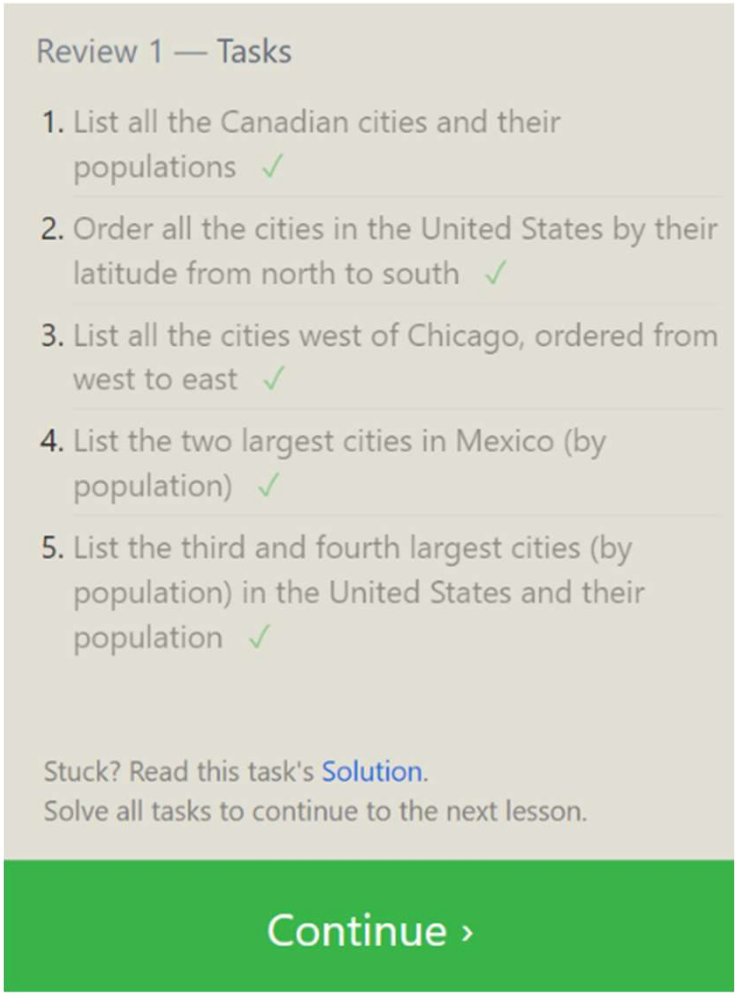
SQL Lesson 4: Filtering and sorting Query results



Answers:

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;

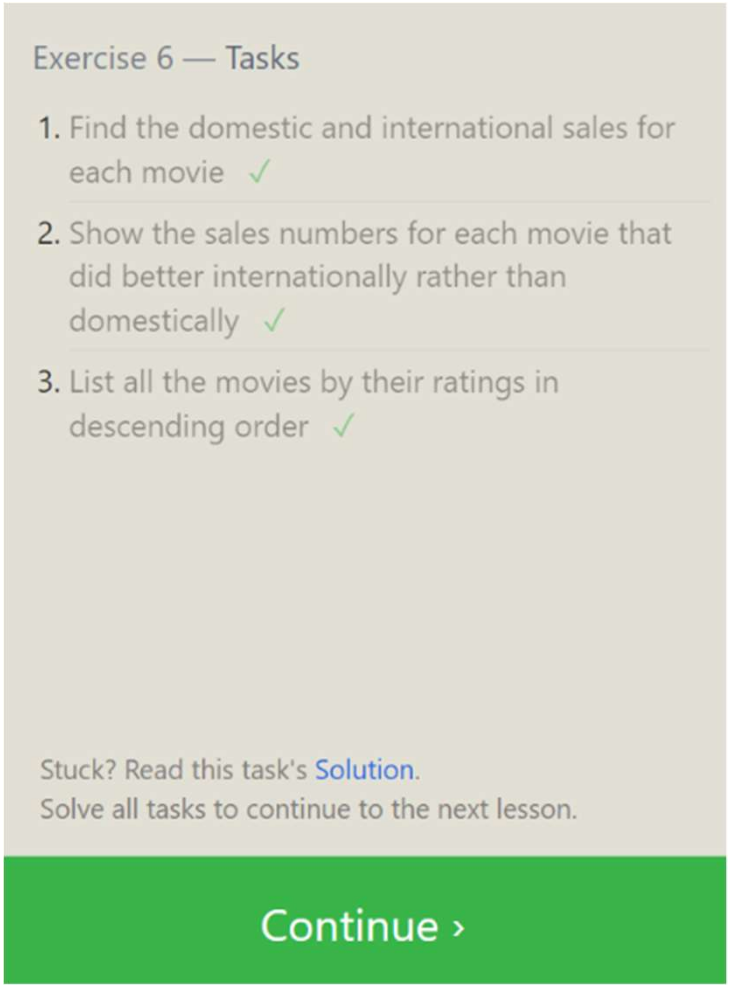
SQL Review: Simple SELECT Queries



Answers:

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;

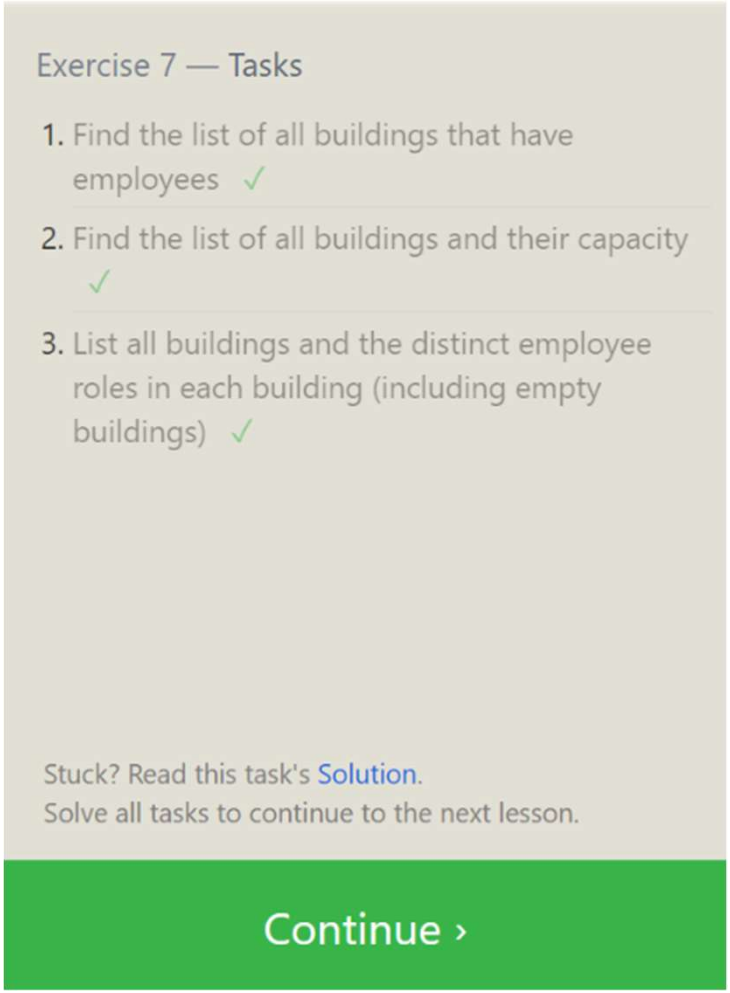
SQL Lesson 6: Multi-table queries with JOINs



Answers:

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;

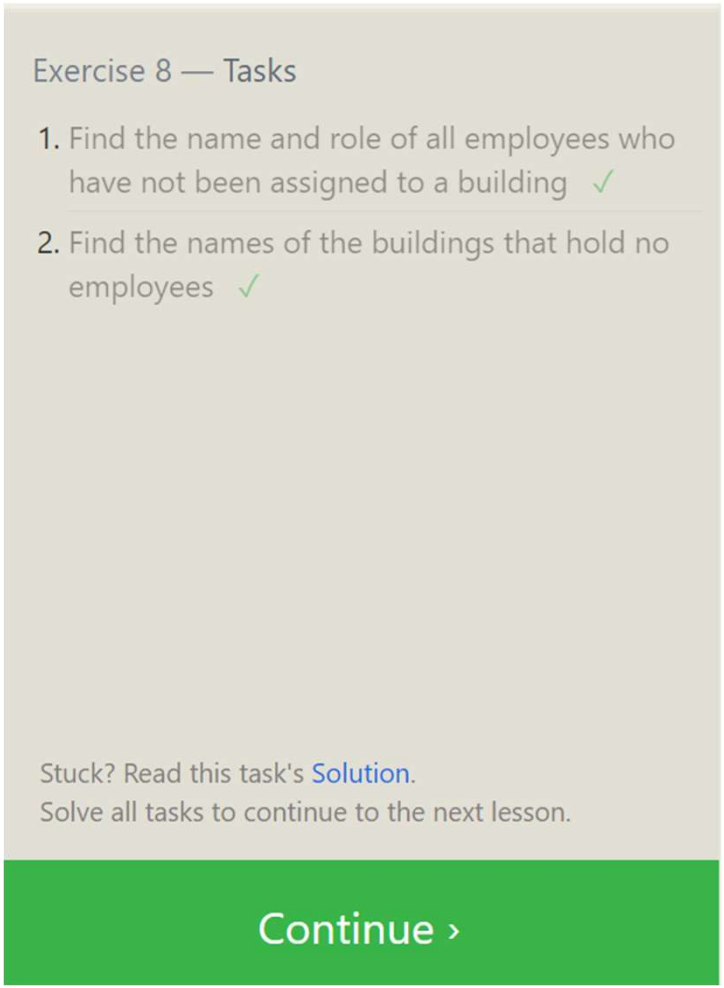
SQL Lesson 7: OUTER JOINs



Answers:

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;

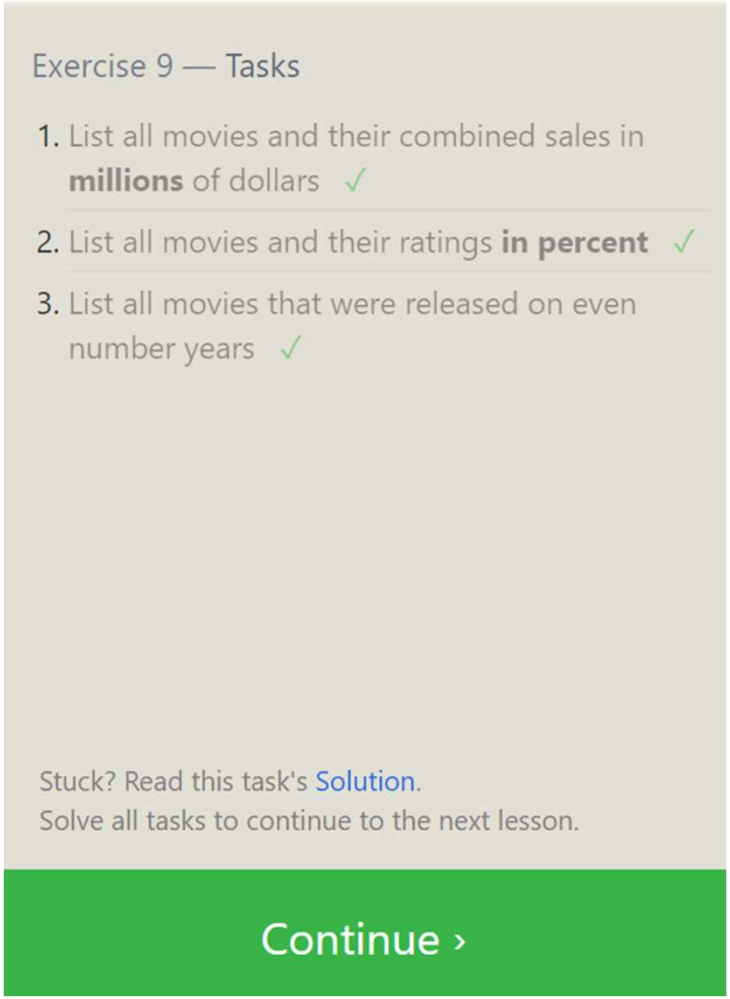
SQL Lesson 8: A short note on NULLs



Answers:

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;

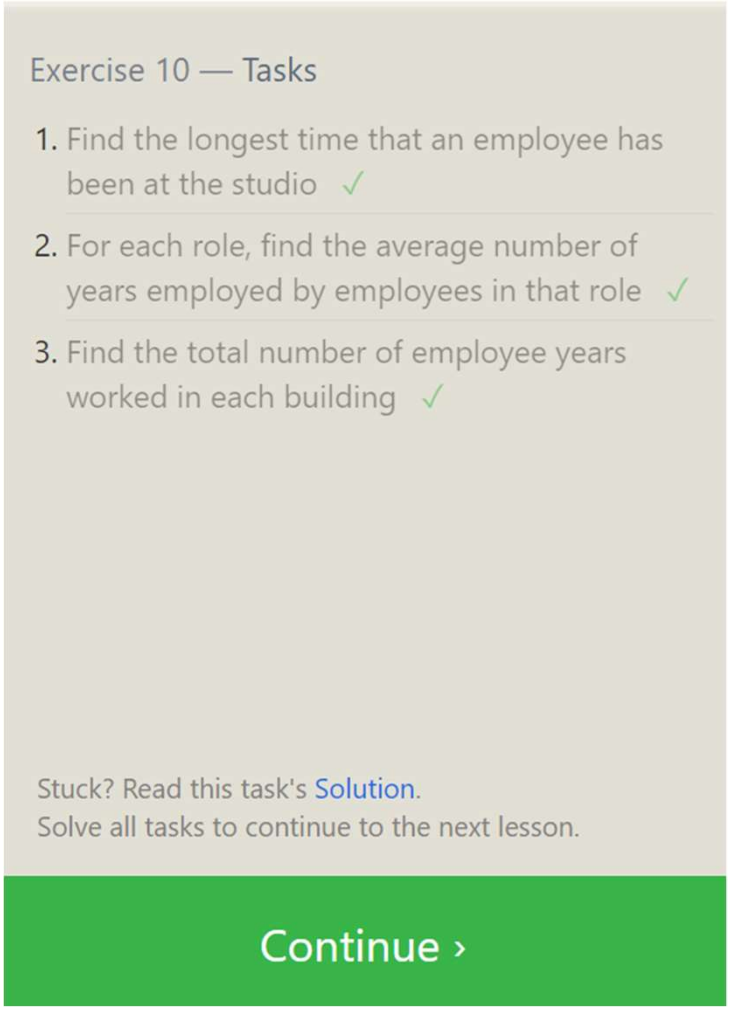
SQL Lesson 9: Queries with expressions



Answers:

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;

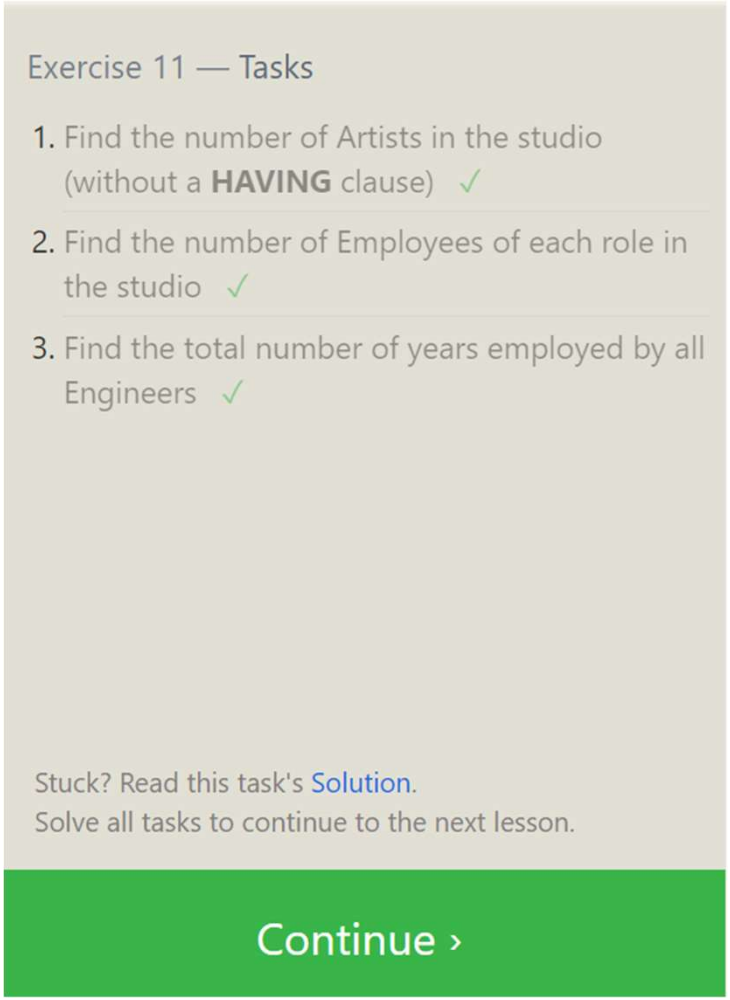
SQL Lesson 10: Queries with aggregates (Pt. 1)



Answers:

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;

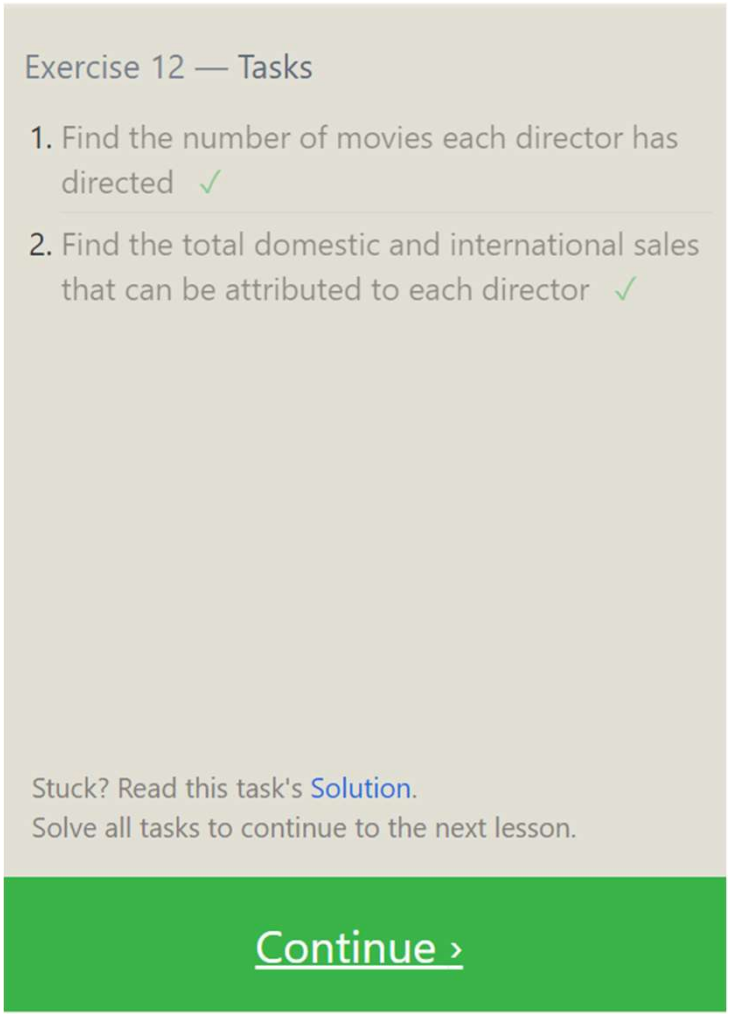
SQL Lesson 11: Queries with aggregates (Pt. 2)



Answers:

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";

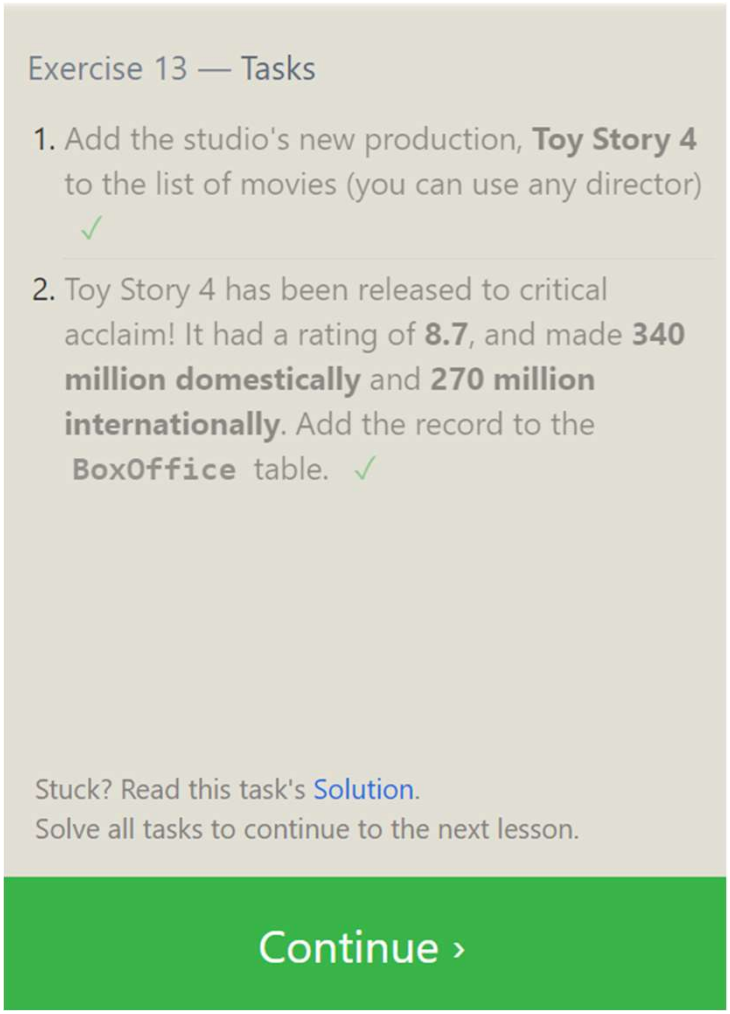
SQL Lesson 12: Order of execution of a Query



Answers:

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;

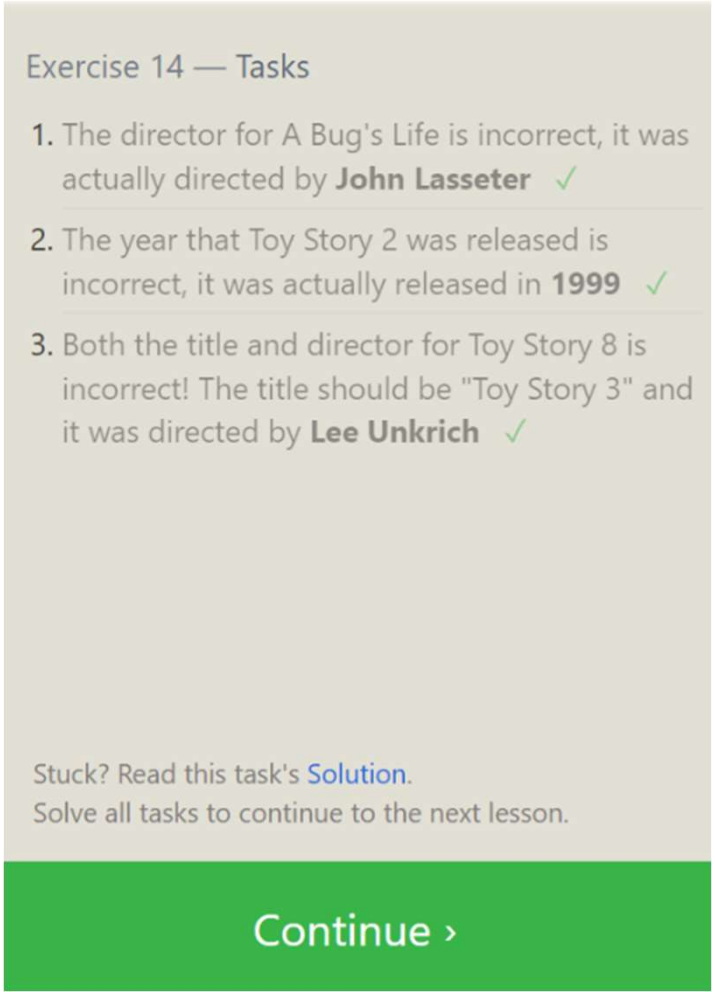
SQL Lesson 13: Inserting rows



Answers:

1. INSERT INTO Movies VALUES (4, "Toy Story 4", "El Directore", 2015, 90);
2. INSERT INTO Boxoffice VALUES (4, 8.7, 340000000, 270000000);

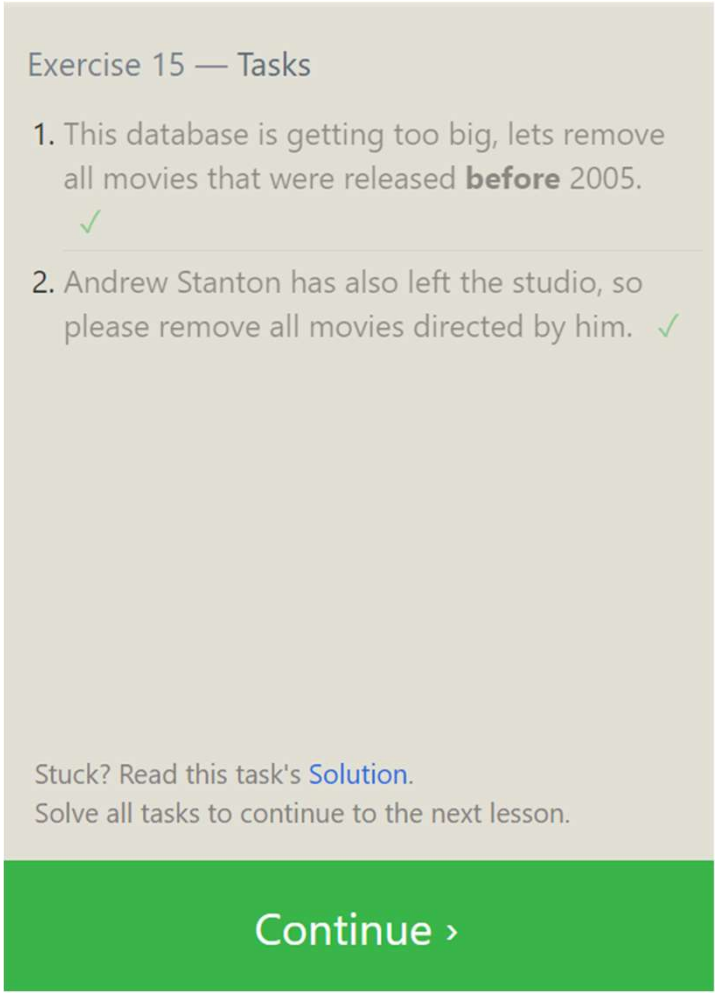
SQL Lesson 14: Updating rows



Answers:

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;

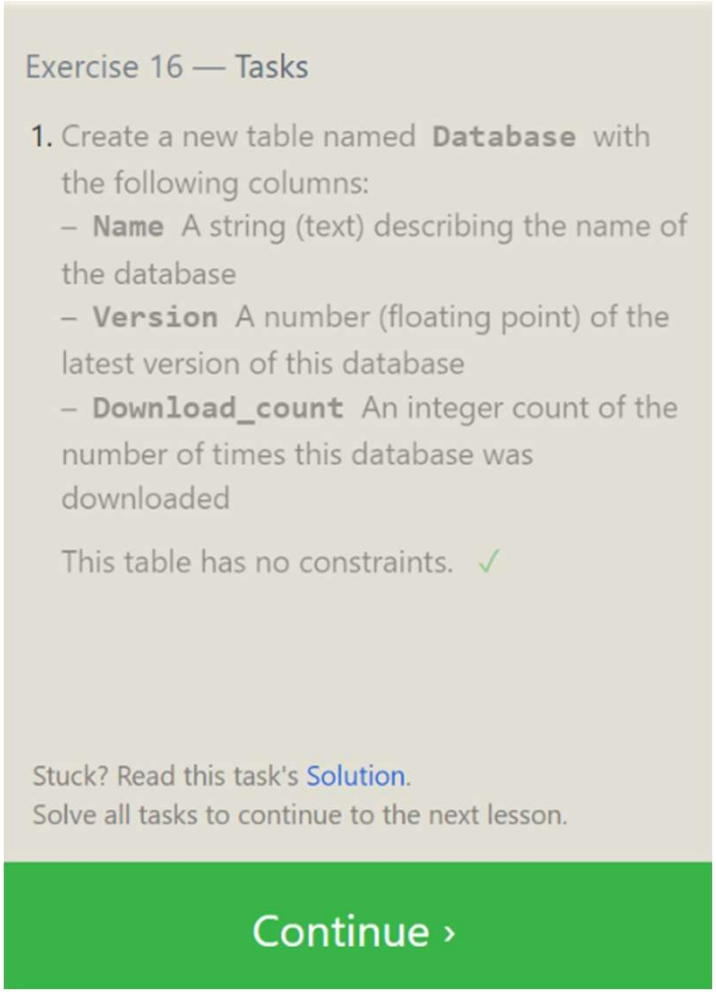
SQL Lesson 15: Deleting rows



Answers:

1. DELETE FROM Movies where Year < 2005;
2. DELETE FROM Movies where Director = "Andrew Stanton";

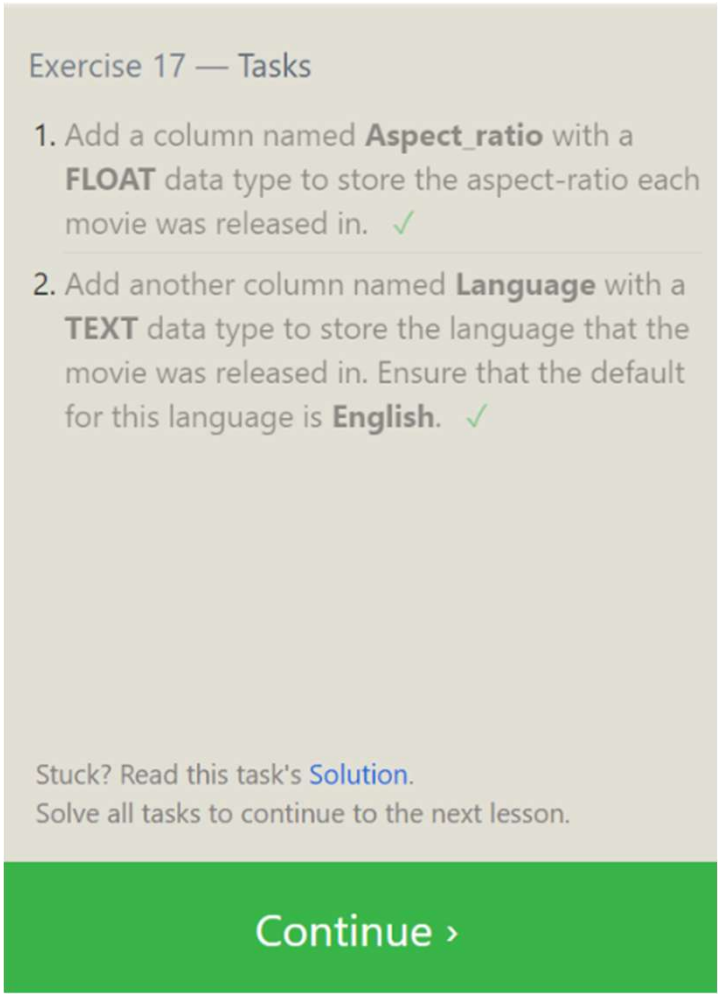
SQL Lesson 16: Creating tables



Answers:

1. CREATE TABLE Database (Name TEXT, Version FLOAT, Download_count INTEGER);

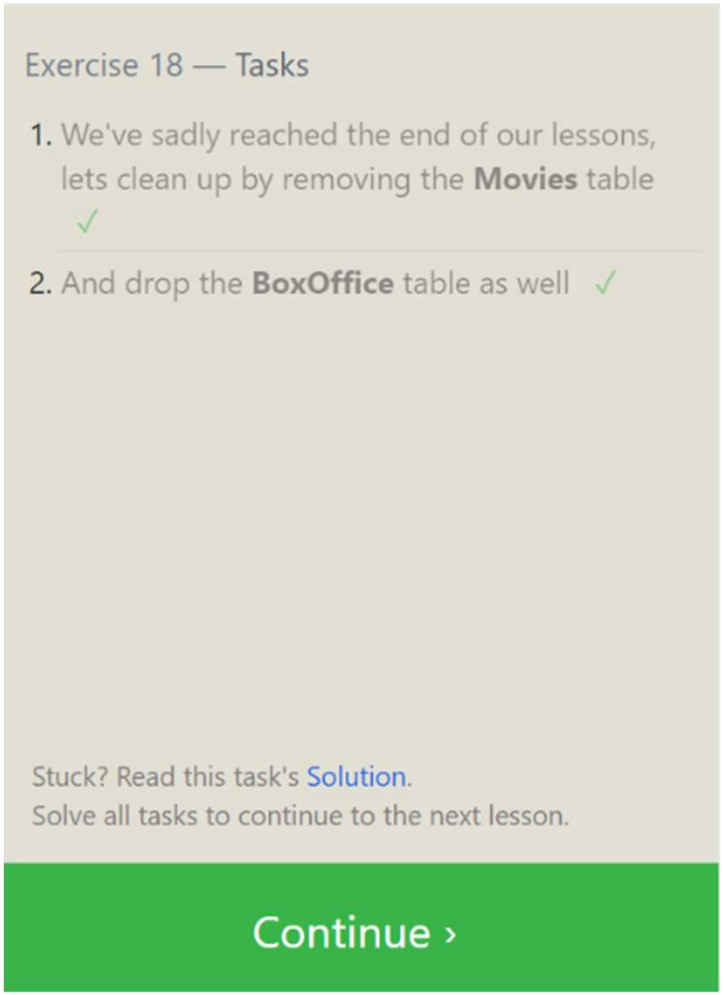
SQL Lesson 17: Altering tables



Answers:

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



Answers:

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

SQL Lesson X: To infinity and beyond!

SQLBolt

Learn SQL with simple, interactive exercises.

SQL Lesson X: To infinity and beyond!

You've finished the tutorial!