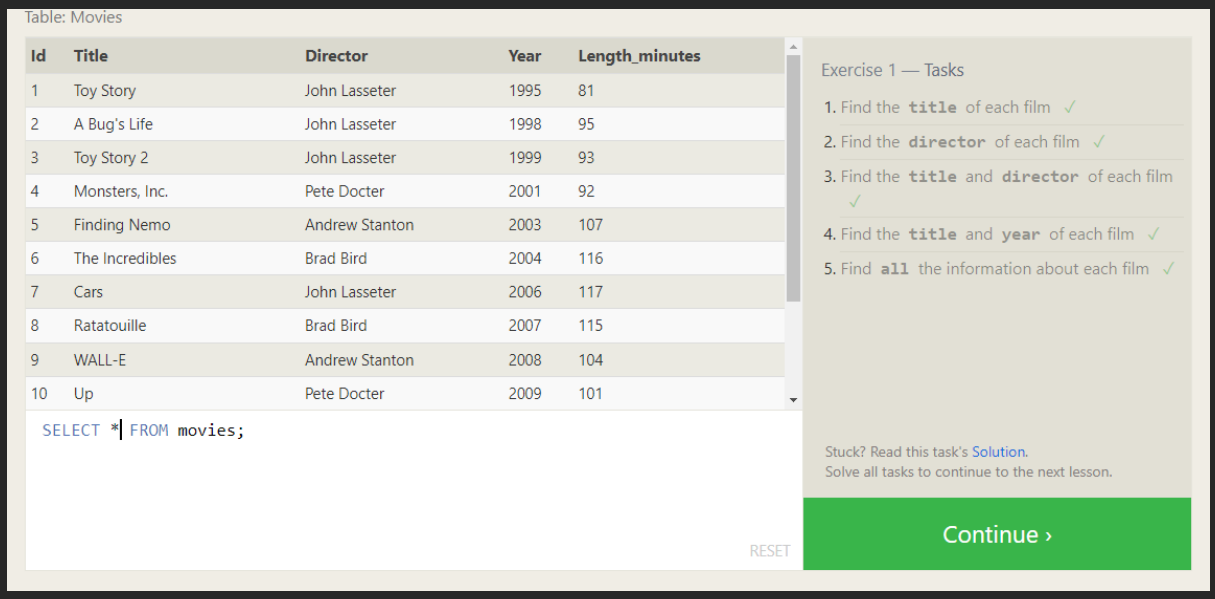
Road-Map : Day-33 : Database : MySql – Day-1 : Tasks

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) :

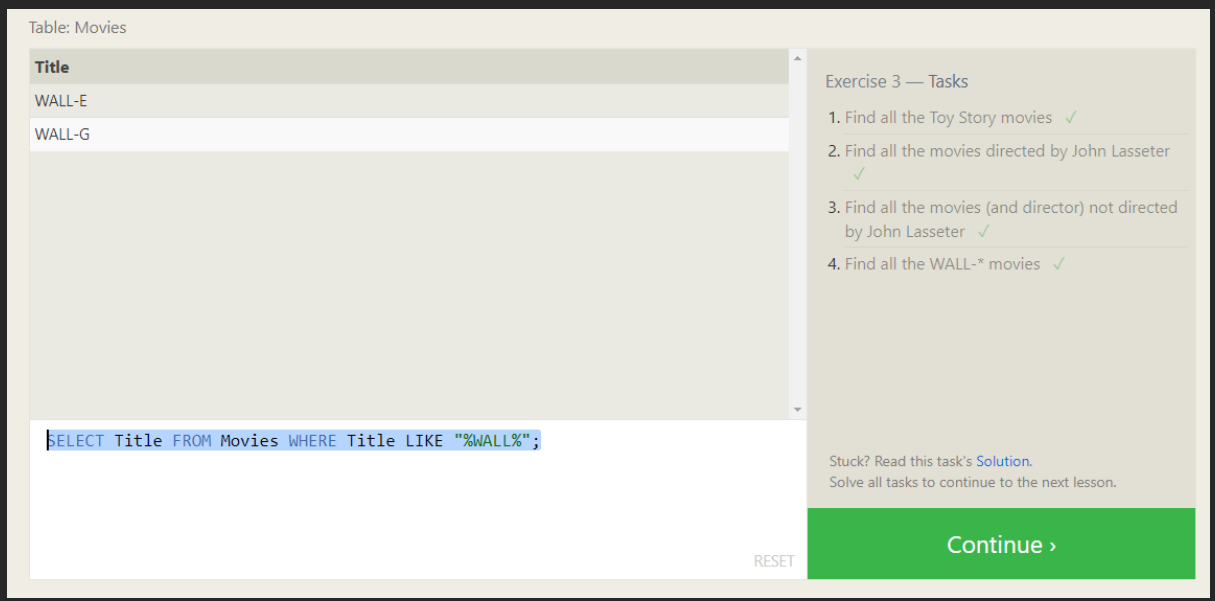
Answers :

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 :

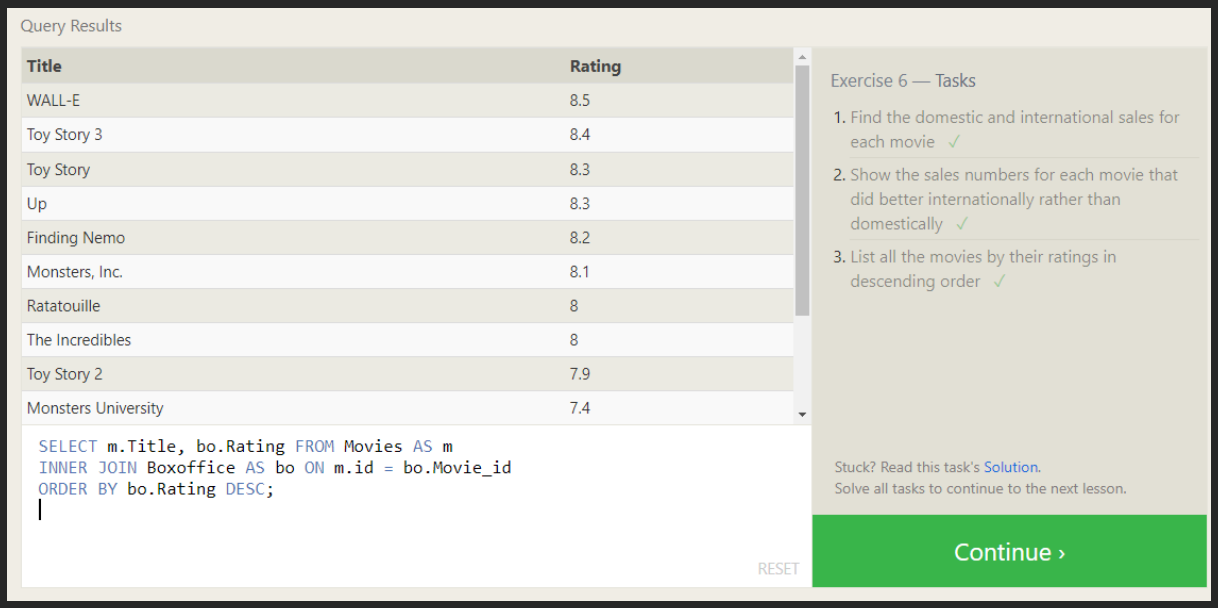
Answers :

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 :

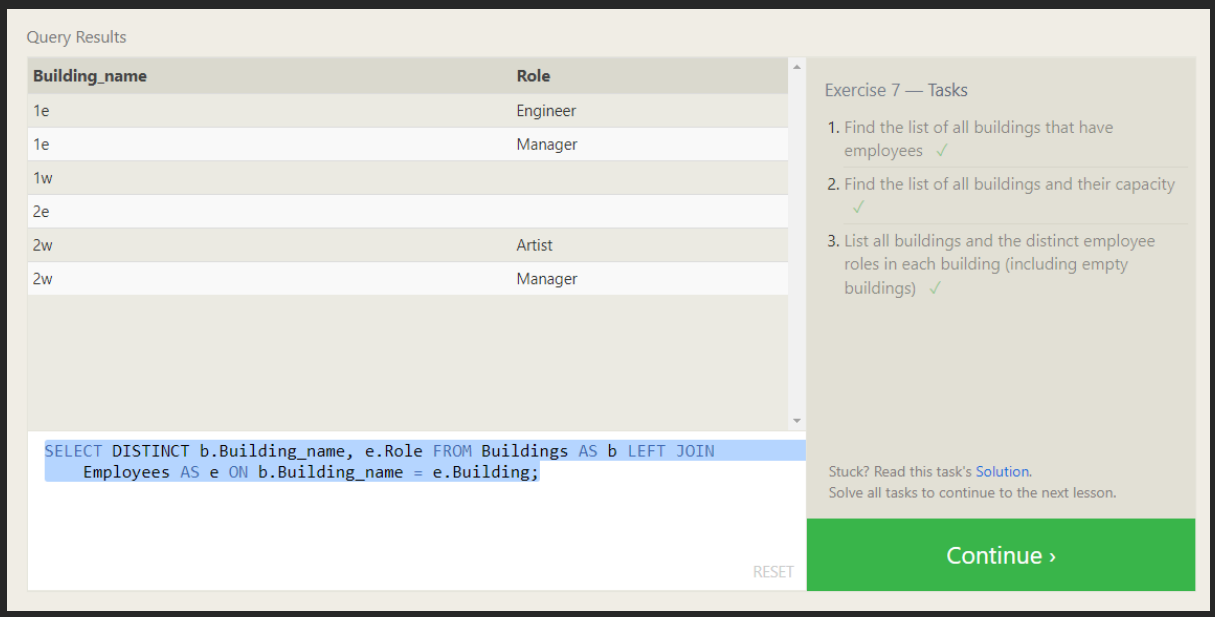
1. SELECT Title, Domestic_sales, International_sales FROM Movies INNER JOIN Boxoffice ON Movies.id = Boxoffice.Movie_id;
2. SELECT m.Title, bo.Domestic_sales, bo.International_sales FROM Movies AS m INNER JOIN Boxoffice AS bo ON m.id = bo.Movie_id WHERE bo.International_sales > bo.Domestic_sales;
3. 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 JOIN's :

Answers :

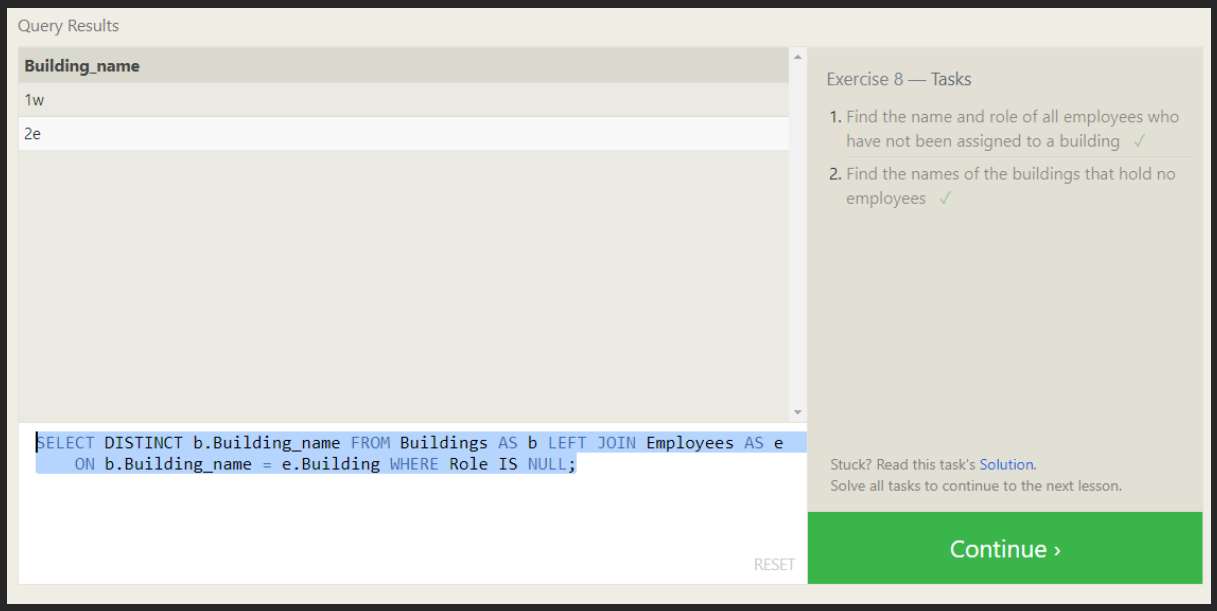
1. SELECT DISTINCT building FROM employees;
2. SELECT Building_name, Capacity FROM Buildings;
3. SELECT DISTINCT b.Building_name, e.Role FROM Buildings AS b LEFT JOIN Employees AS e ON b.Building_name = e.Building;



SQL Lesson 8: A short note on NULLs :

Answers :

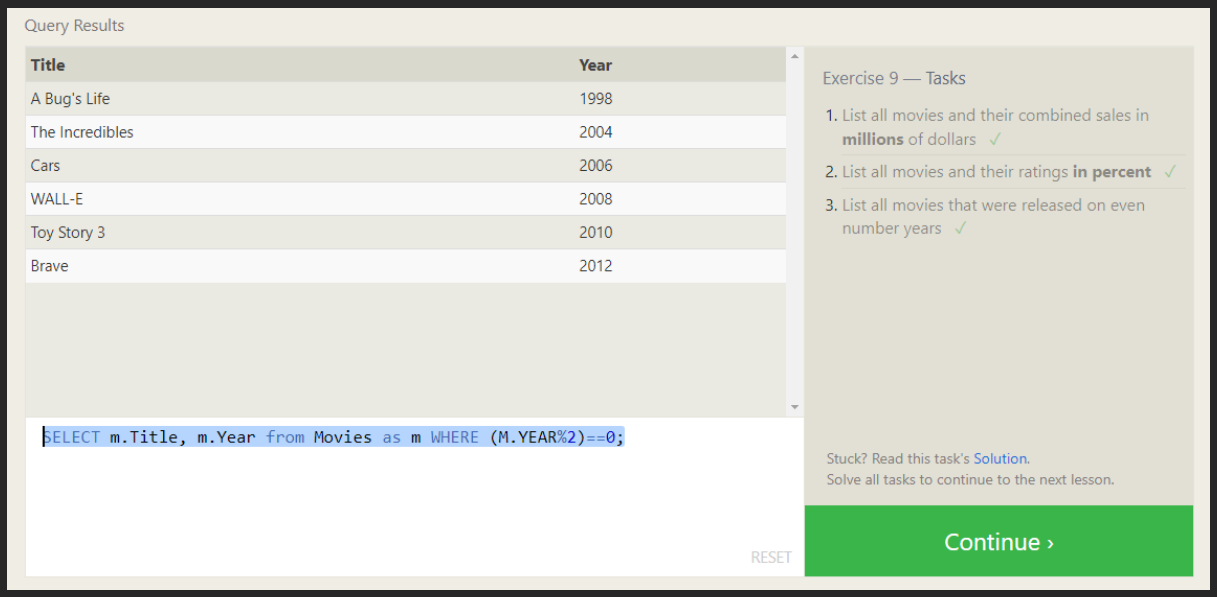
1. SELECT e.Name, e.Role FROM employees AS e LEFT JOIN Buildings AS b ON e.Building = B.Building_name WHERE B.Building_name IS NULL ;
2. SELECT DISTINCT b.Building_name FROM Buildings AS b LEFT JOIN Employees AS e ON b.Building_name = e.Building WHERE Role IS NULL



SQL Lesson 9: Queries with expressions :

Answers :

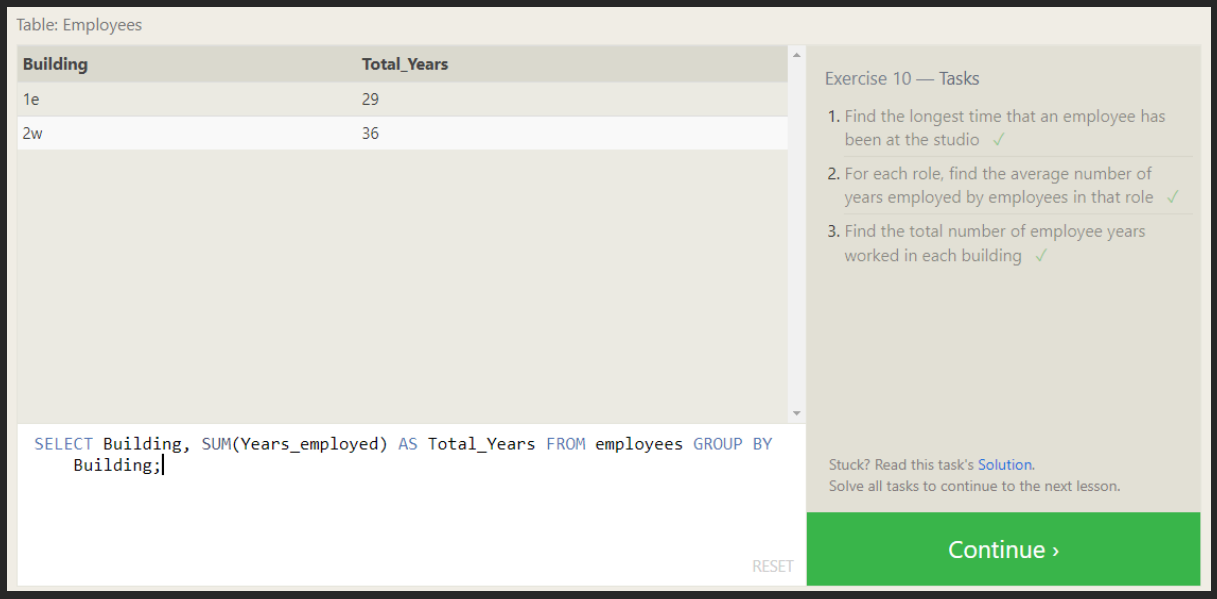
1. SELECT DISTINCT m.Title, (b.Domestic_sales+b.International_sales)/1000000 AS Millions from Movies as m JOIN Boxoffice AS b ON m.Id = b.Movie_id ORDER BY Millions DESC;
2. SELECT m.Title, (b.Rating)*10 AS Rating from Movies as m JOIN Boxoffice AS b ON m.Id = b.Movie_id ORDER BY b.Rating DESC;
3. SELECT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



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

Answers :

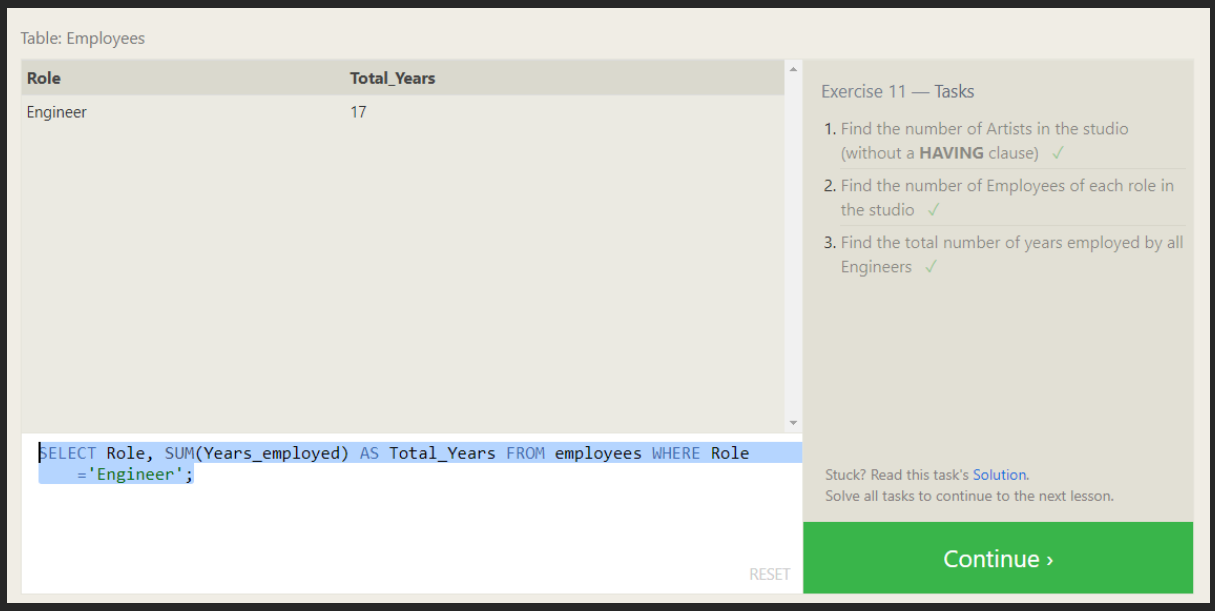
1. SELECT Name,MAX(Years_employed) FROM Employees;
2. SELECT Role, AVG(Years_employed) as Average_years_employed FROM Employees GROUP BY Role;
3. SELECT Building, SUM(Years_employed) AS Total_Years FROM employees GROUP BY Building;



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

Answers :

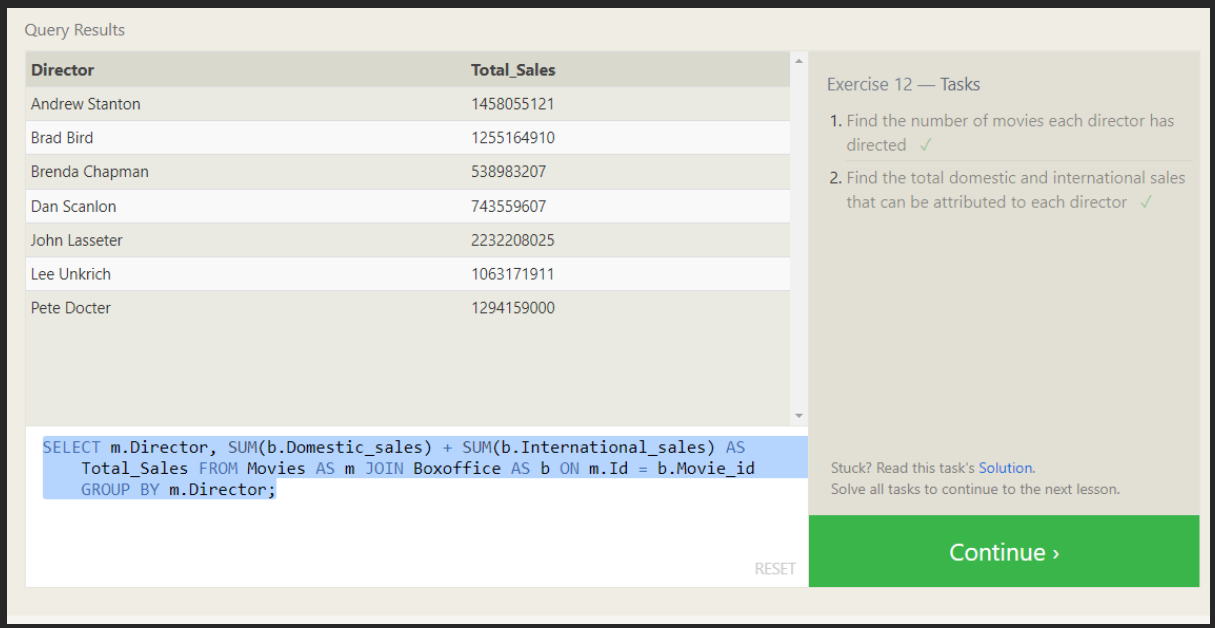
1. SELECT Role, COUNT(Role) AS Artist_Count From employees WHERE Role = 'Artist';
2. SELECT Role, COUNT(Role) AS Total_Employees FROM Employees GROUP BY Role;
3. SELECT Role, SUM(Years_employed) AS Total_Years FROM employees WHERE Role='Engineer';



SQL Lesson 12: Order of execution of a Query :

Answers :

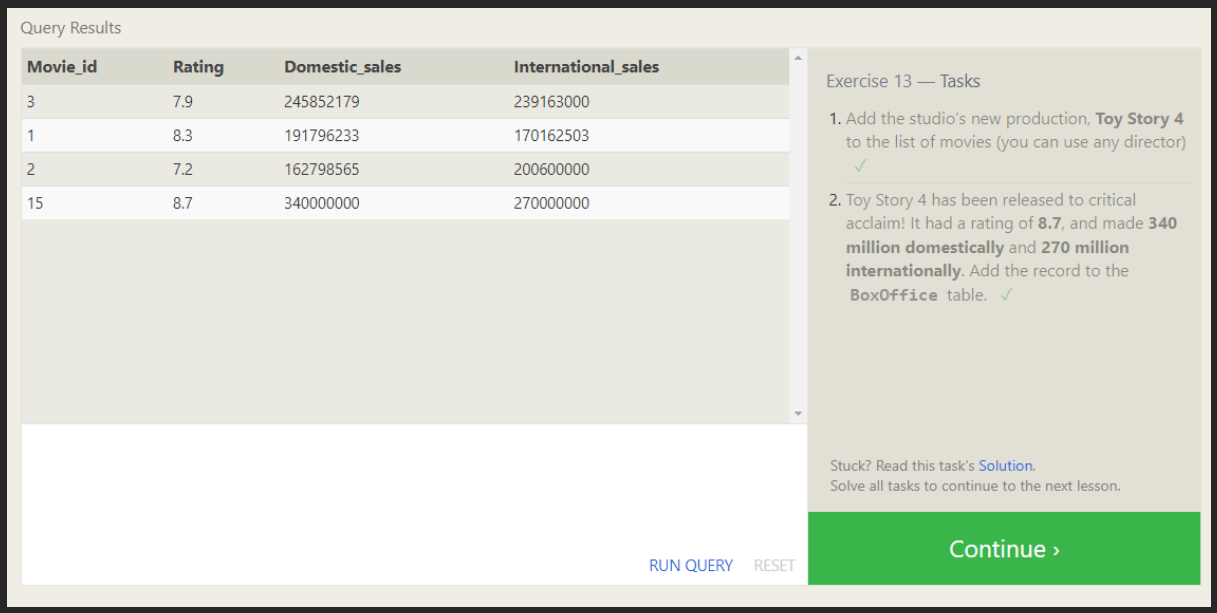
1. SELECT Director, COUNT(Director) AS Total_Movies FROM movies GROUP BY Director;
2. SELECT m.Director, SUM(b.Domestic_sales) + SUM(b.International_sales) AS Total_Sales FROM Movies AS m JOIN Boxoffice AS b ON m.Id = b.Movie_id GROUP BY m.Director;



SQL Lesson 13: Inserting rows

Answers :

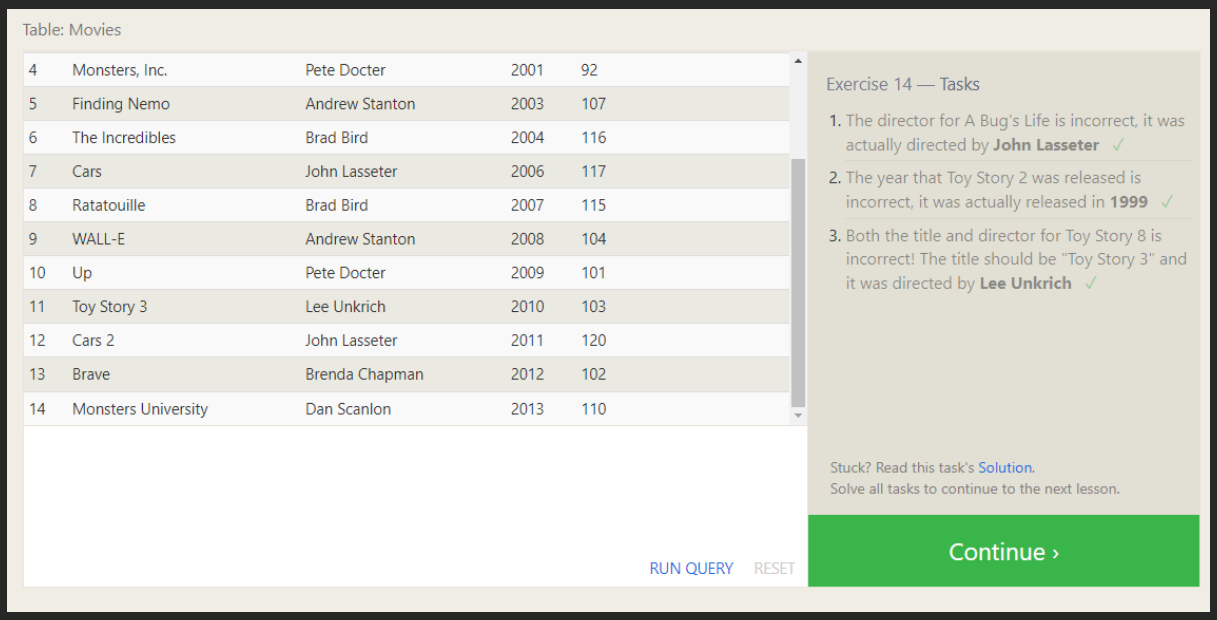
1. INSERT INTO Movies VALUES(15, 'Toy Story 4', 'John Lasseter', 2002, 90);
2. INSERT INTO BoxOffice VALUES(15, 8.7, 340000000, 270000000);



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';

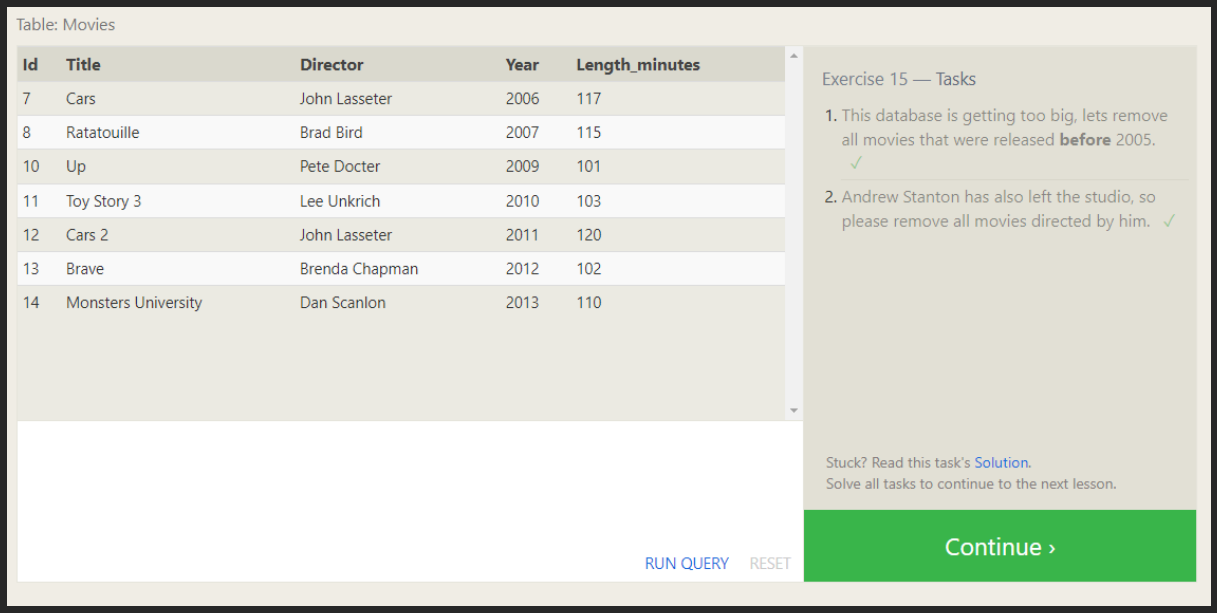


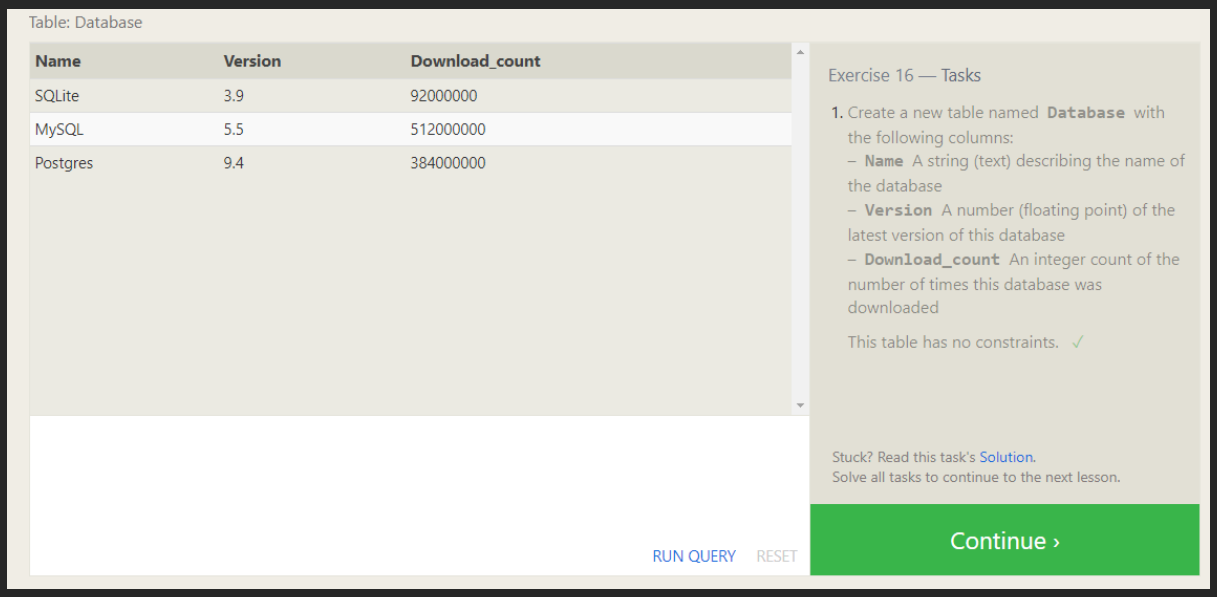
Table: Movies

Id	Title	Director	Year	Length_minutes
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

SQL Lesson 16: Creating tables :

Answers :

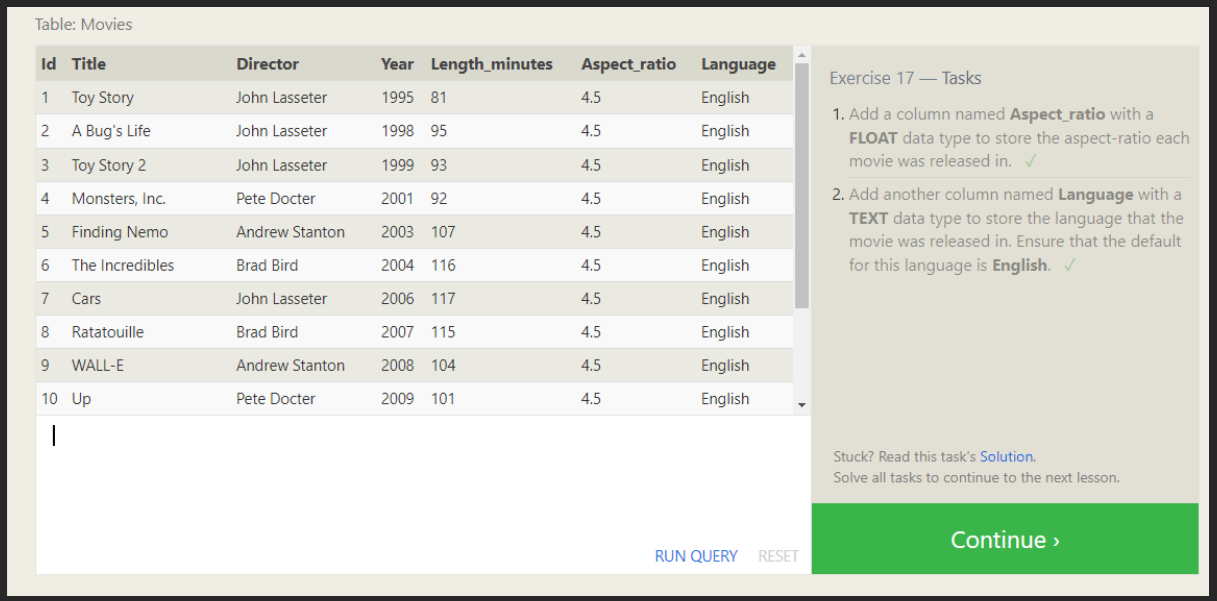
1. CREATE TABLE Database (Name varchar(20), Version float ,Download_count int);



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;