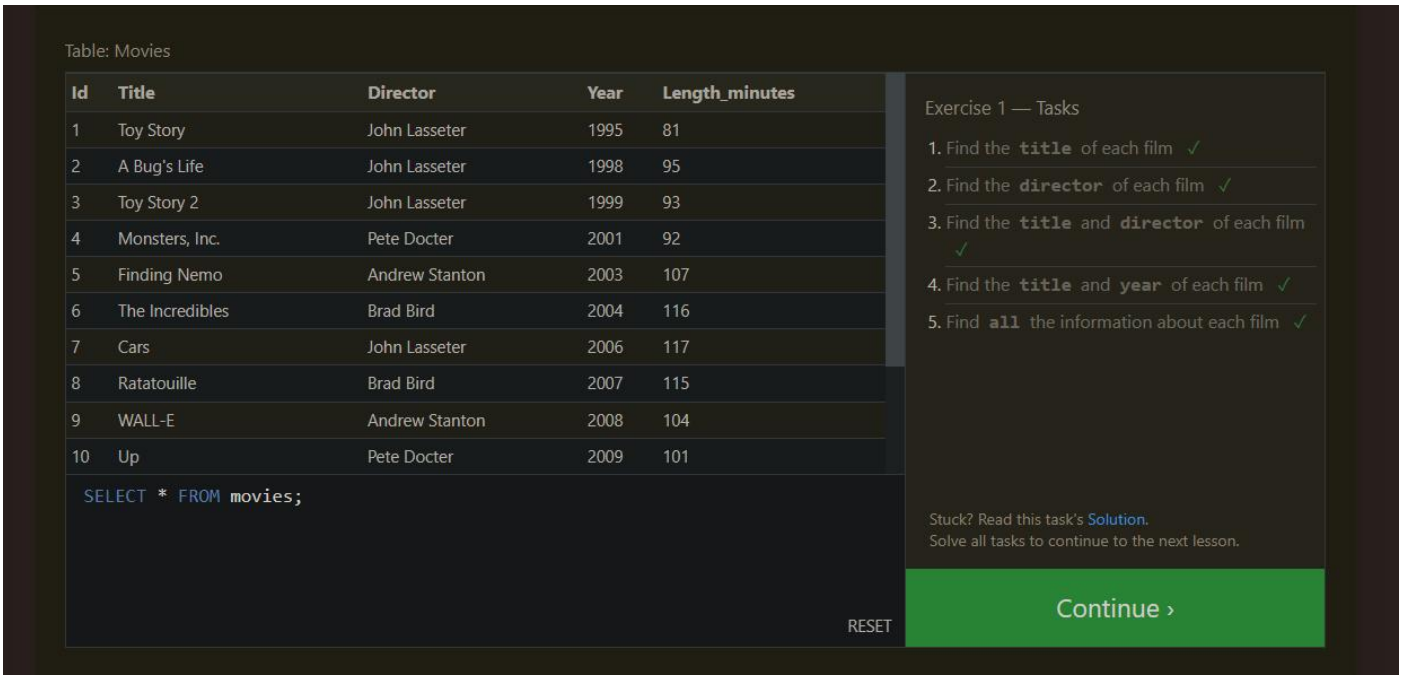
SQLBOLT

SQL Lesson 1: SELECT queries 10



1.SELECT title FROM movies;

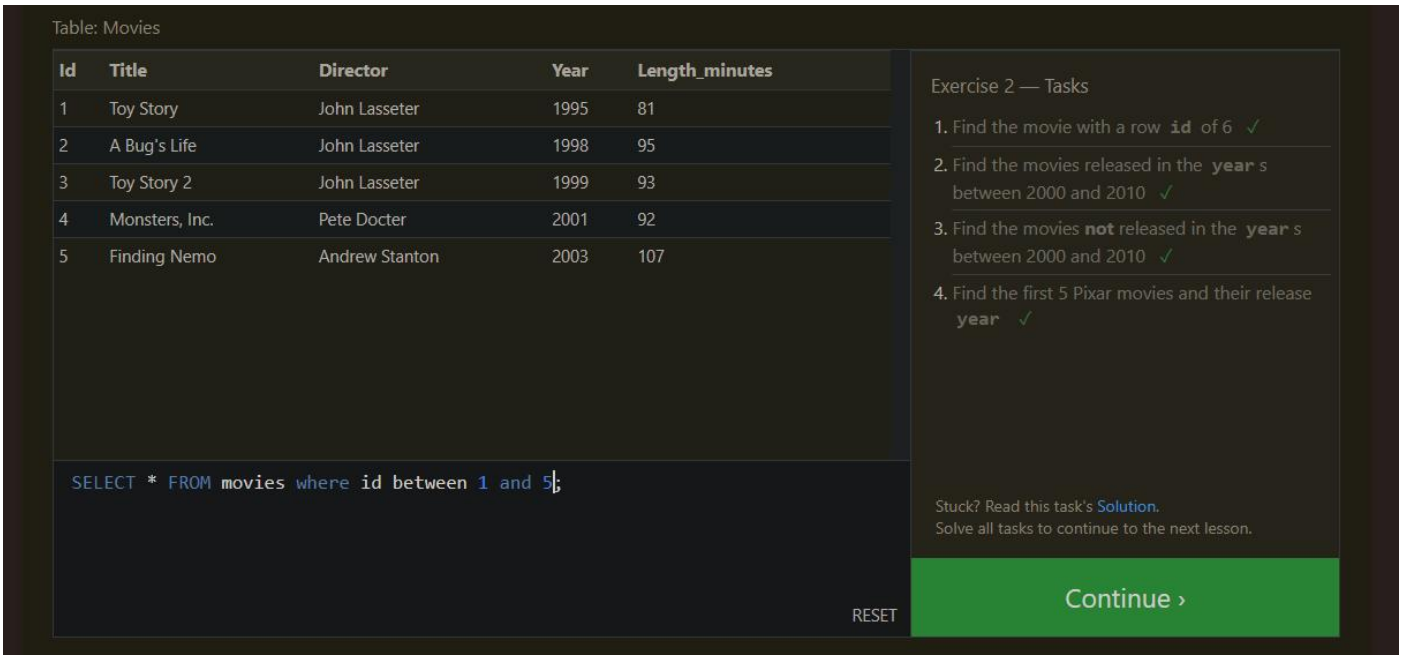
2.SELECT director FROM movies;

3.SELECT title,director FROM movies;

4.SELECT title,year FROM movies;

5.SELECT * FROM movies;

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



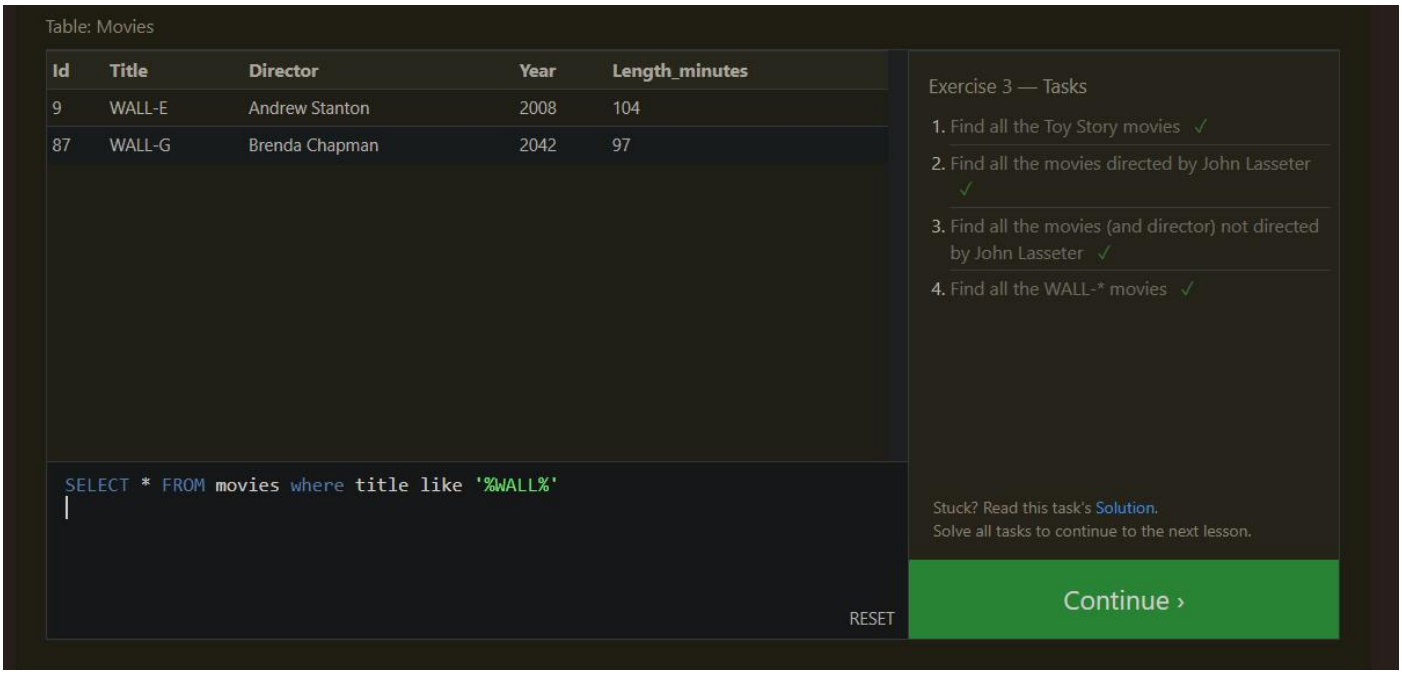
1. SELECT * FROM movies where id=6;

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)



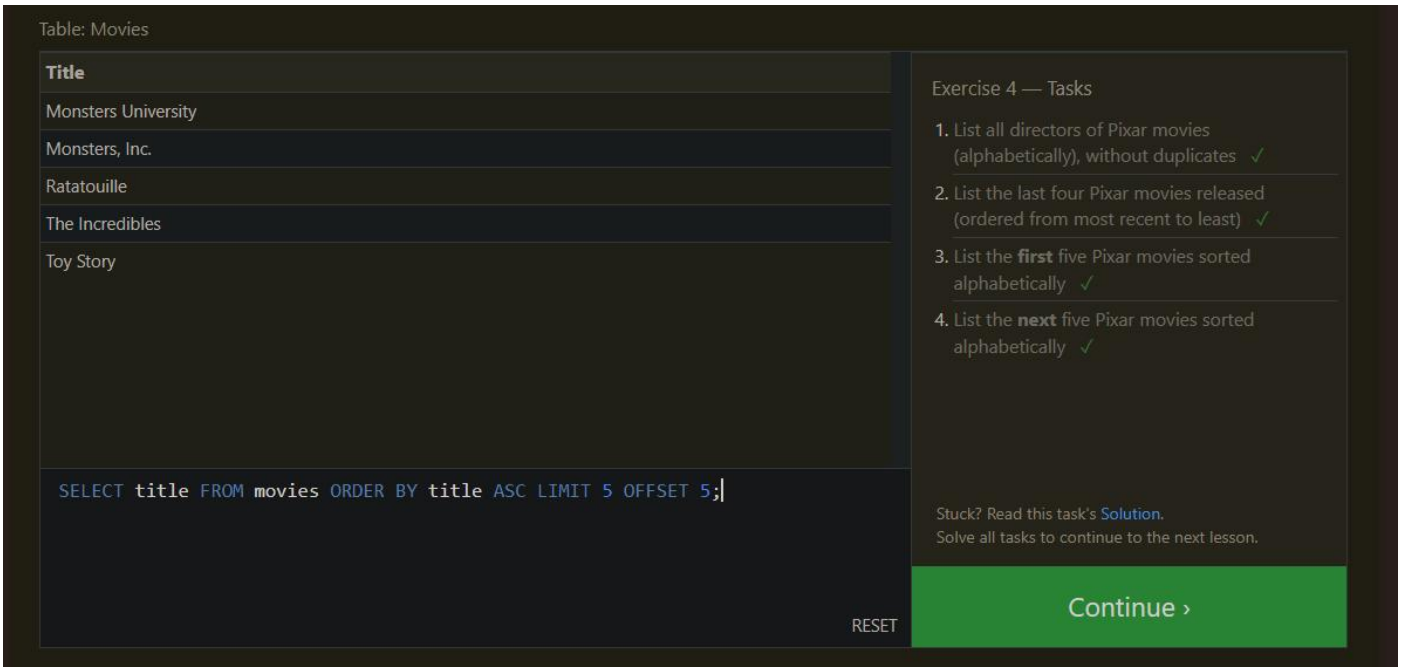
1. SELECT * FROM movies where title like '%toy%'

2. SELECT * FROM movies where director like '%John Lasseter%'

3. SELECT * FROM movies where director not like '%John Lasseter%'

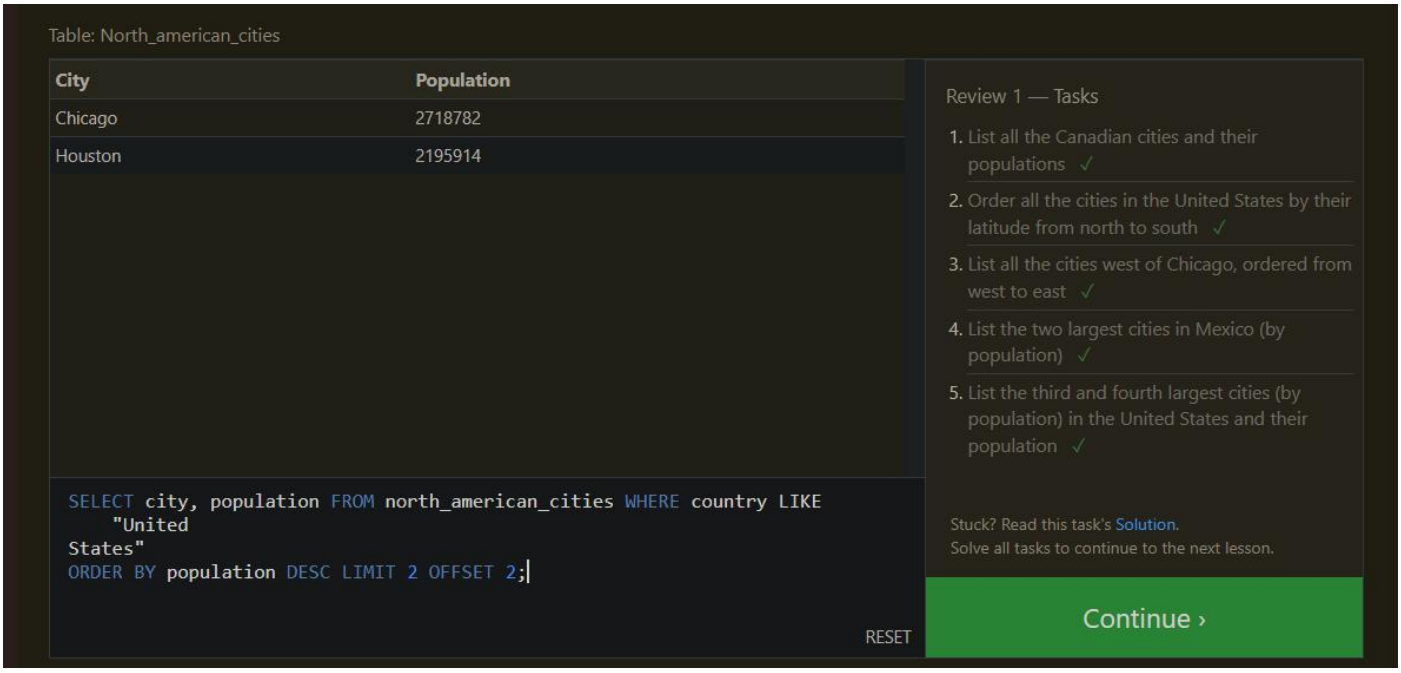
4. SELECT * FROM movies where title like '%WALL%'

SQL Lesson 4: Filtering and sorting Query results



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 Lesson5 Review: Simple SELECT Queries



1. SELECT city, population FROM north_american_cities WHERE country = "Canada";

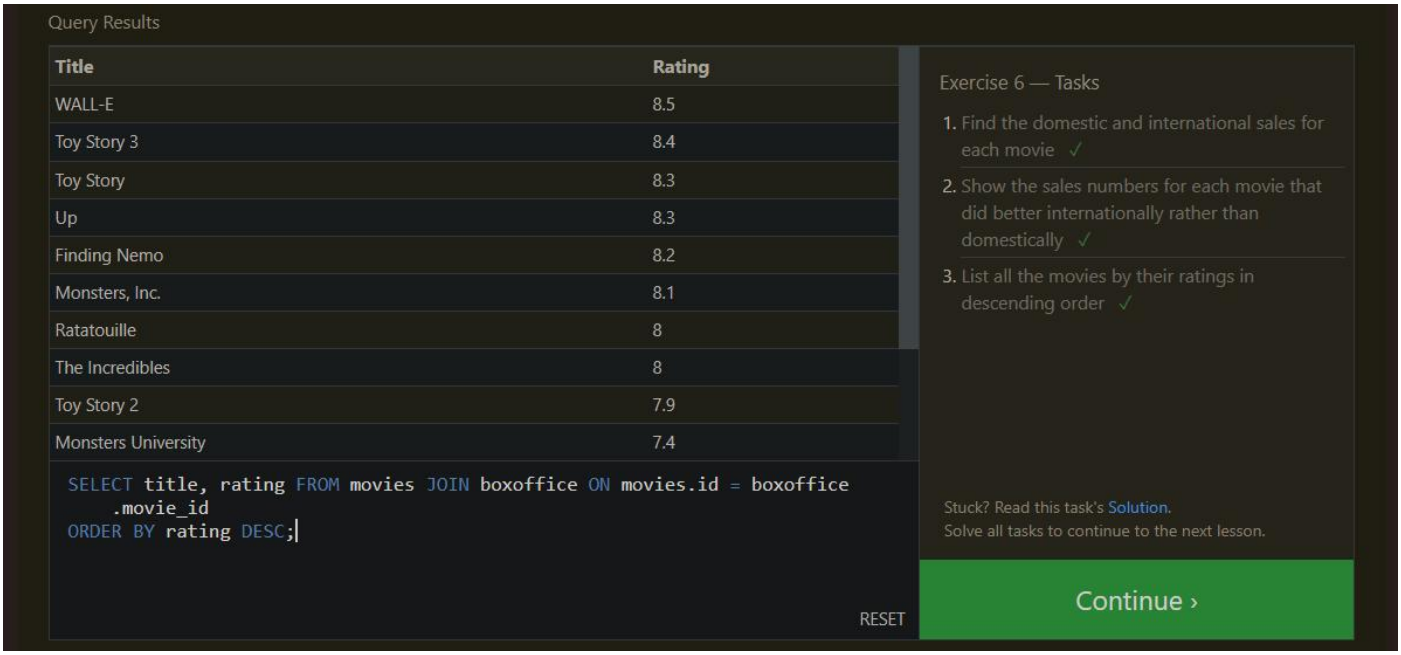
2. SELECT city, latitude FROM north_american_cities WHERE country = "United States" ORDER BY latitude DESC;

3.SELECT City, Latitude FROM north_american_cities WHERE Longitude < -87.629798 ORDER BY Longitude;

3. SELECT city, population FROM north_american_cities WHERE country LIKE "Mexico" ORDER BY population DESC LIMIT 2;

4. 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

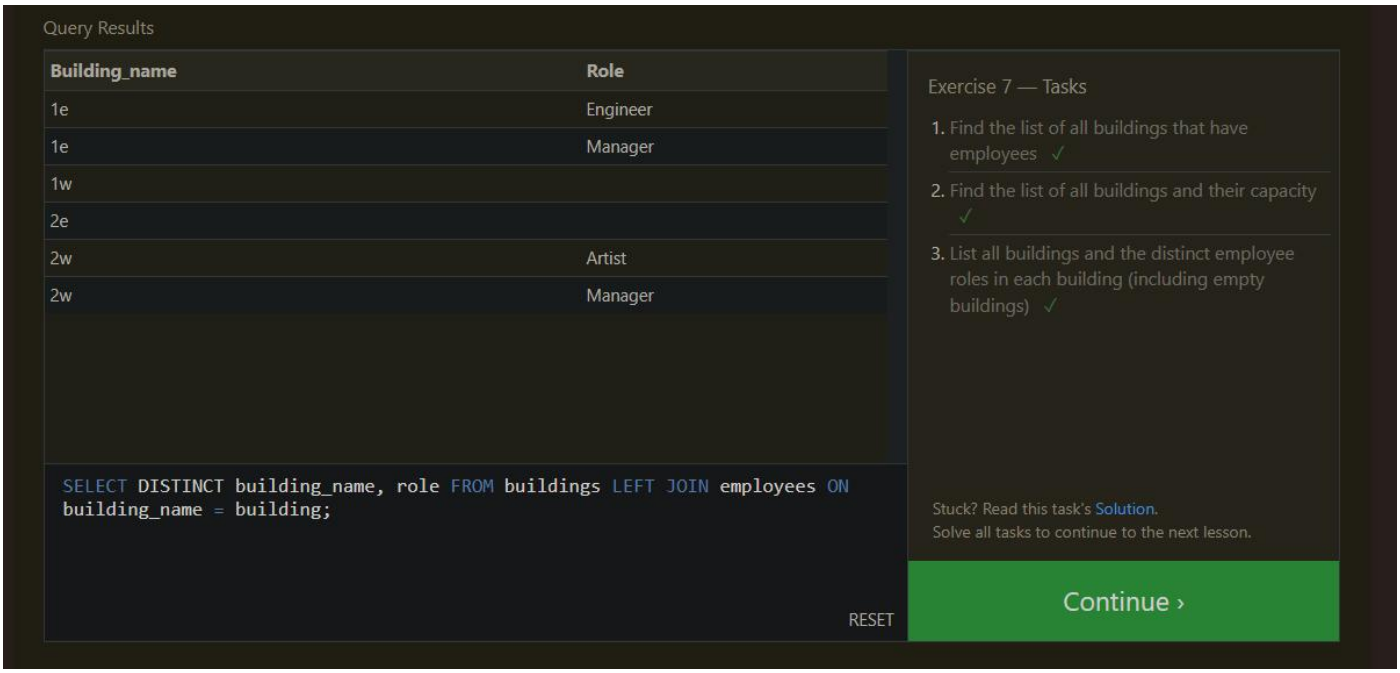


1. SELECT title, domestic_sales, international_sales FROM movies JOIN boxoffice ON movies.id = boxoffice.movie_id;

2. SELECT Title, Domestic_sales, International_sales FROM movies INNER JOIN Boxoffice ON Id = Movie_id WHERE Domestic_sales < International_sales;

3. SELECT title, rating FROM movies JOIN boxoffice ON movies.id = boxoffice.movie_id ORDER BY rating DESC;

SQL Lesson 7: OUTER JOINs

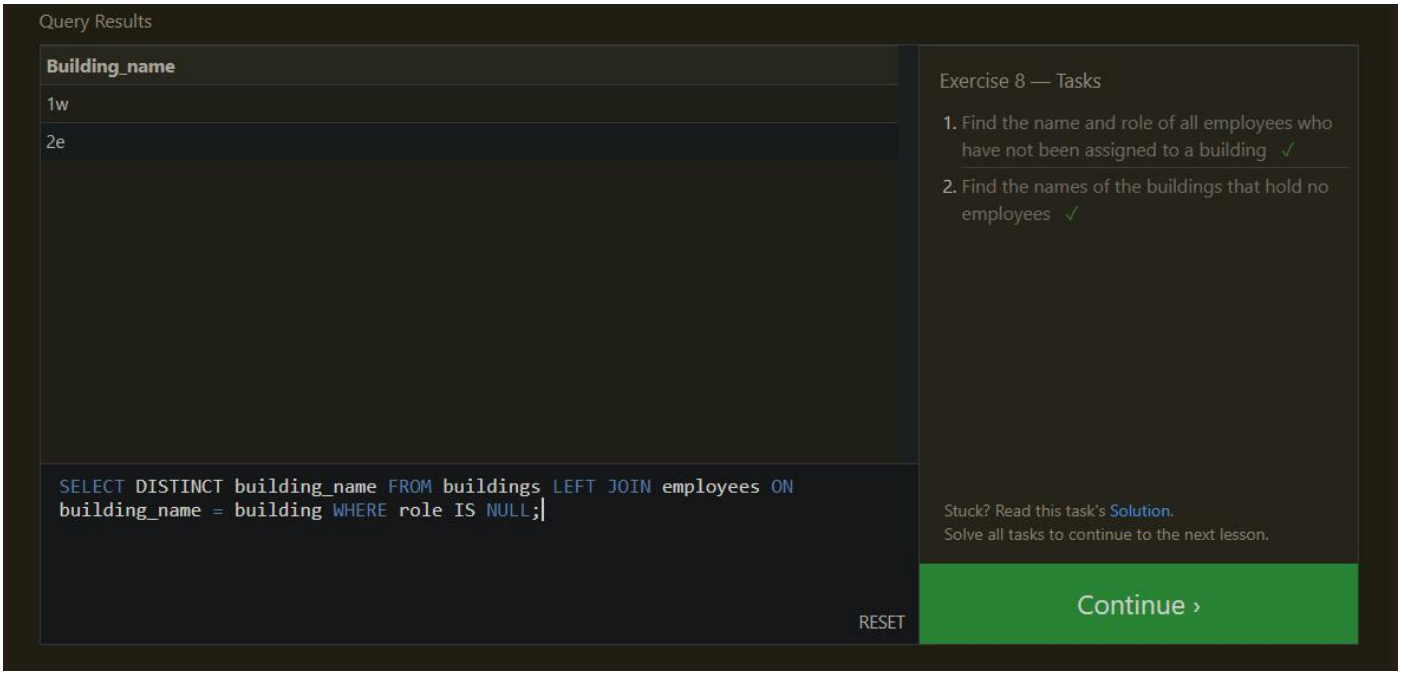


1. SELECT DISTINCT building FROM employees;

2. SELECT * FROM buildings;

3. SELECT DISTINCT building_name, role FROM buildings LEFT JOIN employees ON building_name = building;

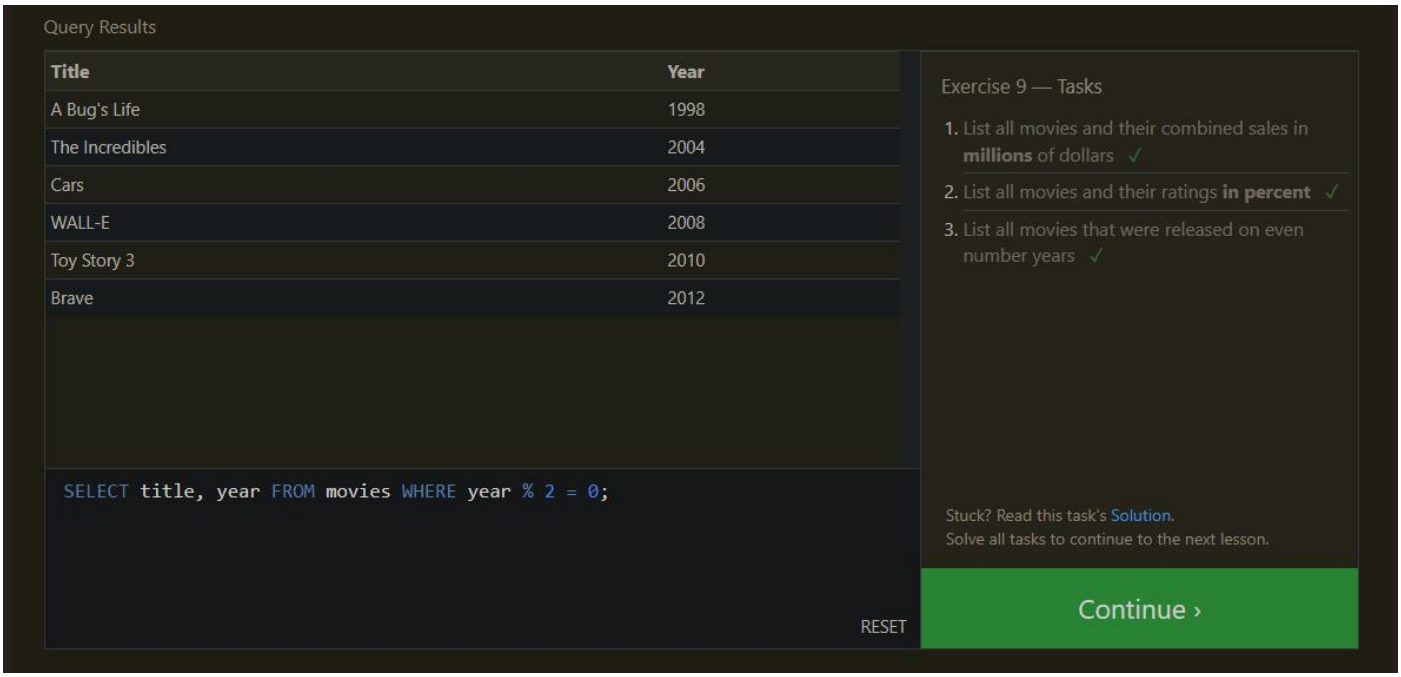
SQL Lesson 8: A short note on NULLs



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

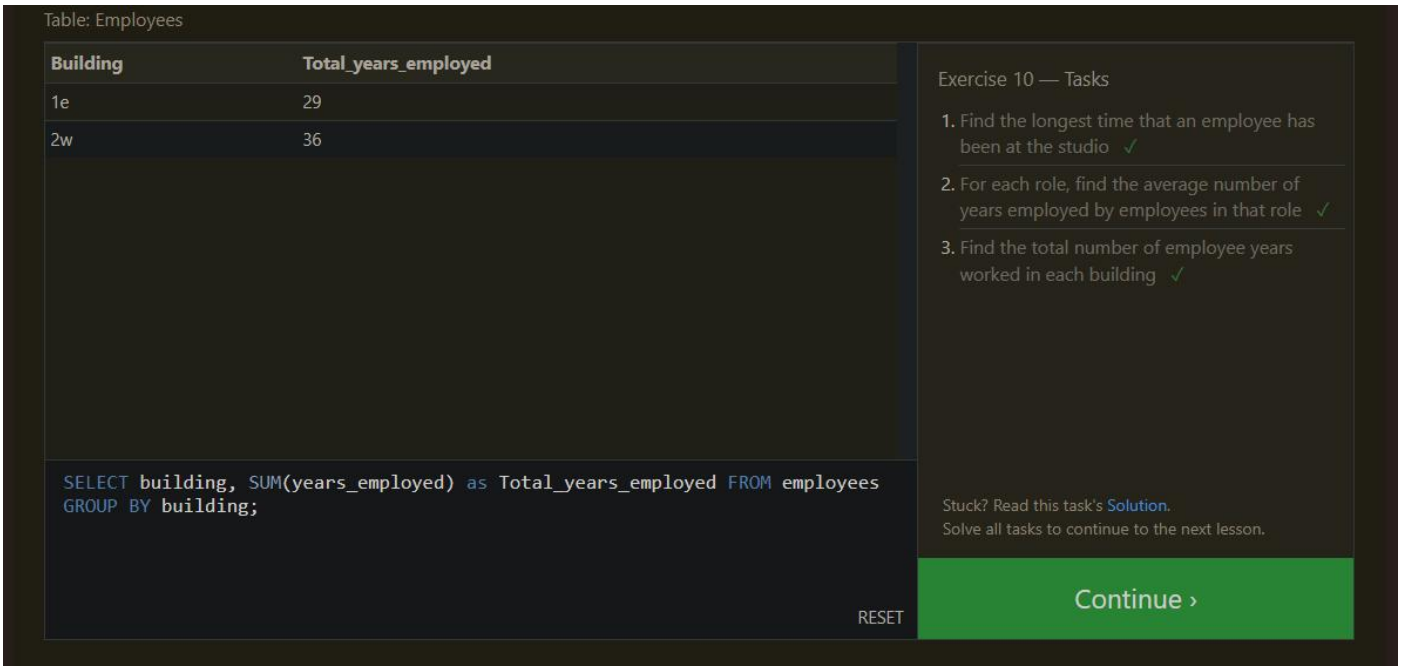


1.SELECT title, (domestic_sales + international_sales) / 1000000 AS gross_sales_millions FROM movies JOIN boxoffice ON movies.id = boxoffice.movie_id;

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)

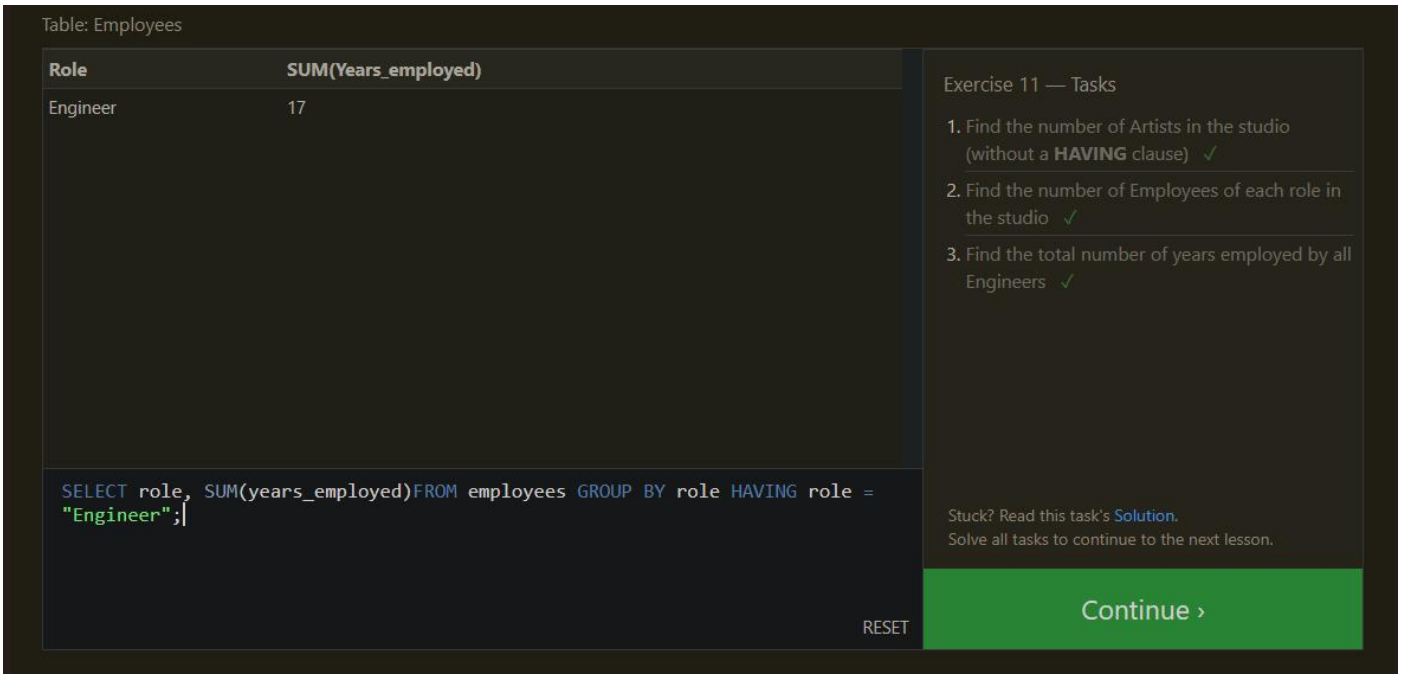


1.SELECT MAX(years_employed) as Max_years_employed FROM employees;

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)

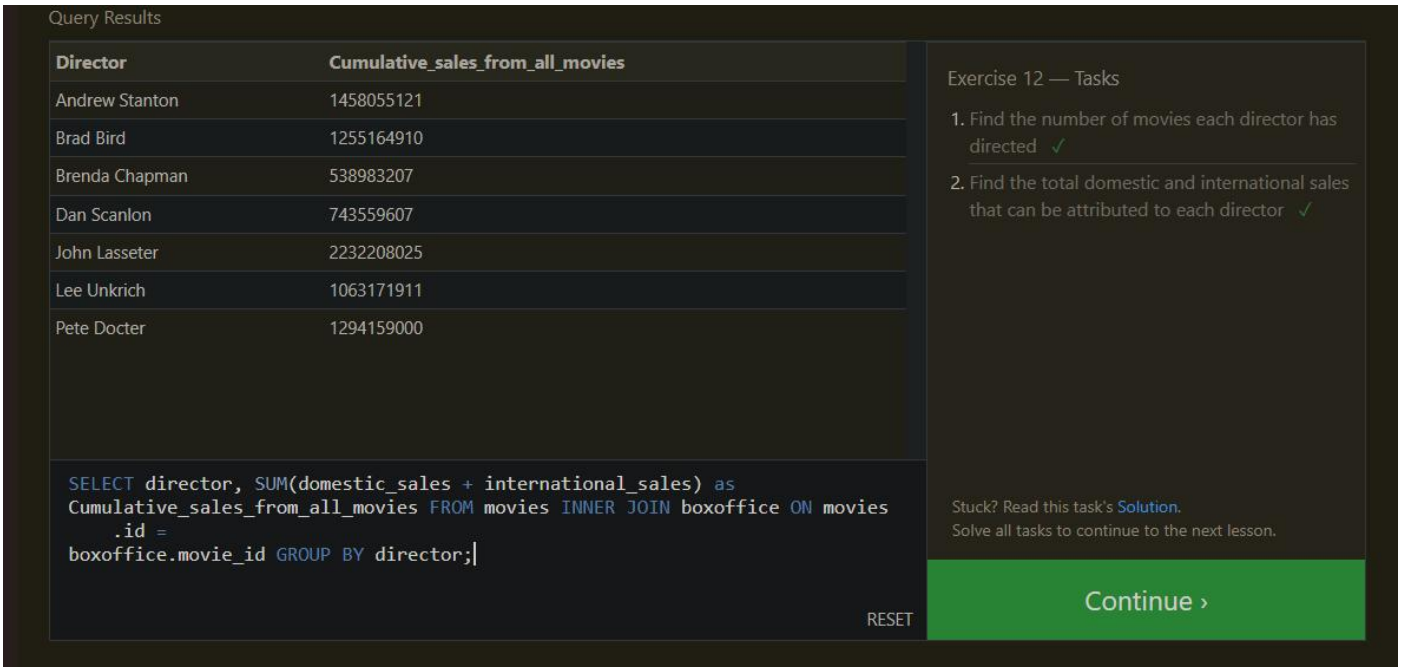


1. SELECT role, COUNT(*) as Number_of_artists FROM employees WHERE role = "Artist";

2. SELECT role, COUNT(*) FROM employees GROUP BY role;

3. SELECT role, SUM(years_employed)FROM employees GROUP BY role HAVING role = "Engineer";

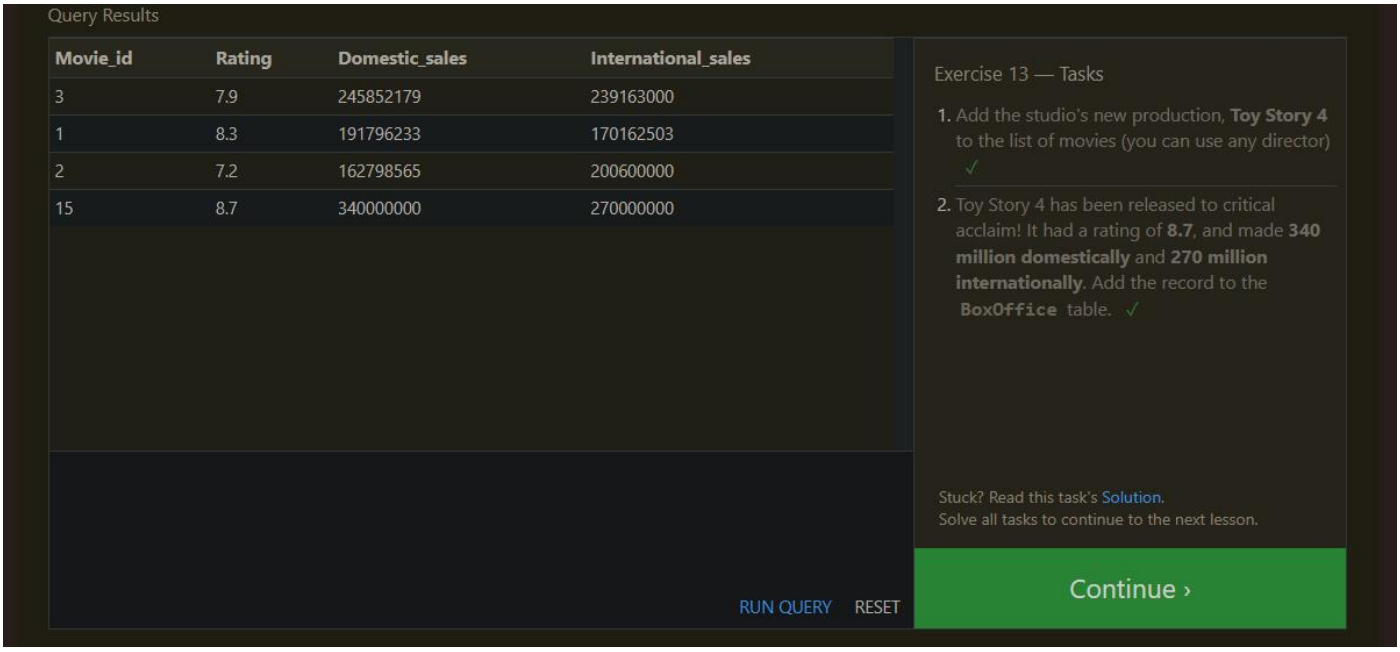
SQL Lesson 12: Order of execution of a Query



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;

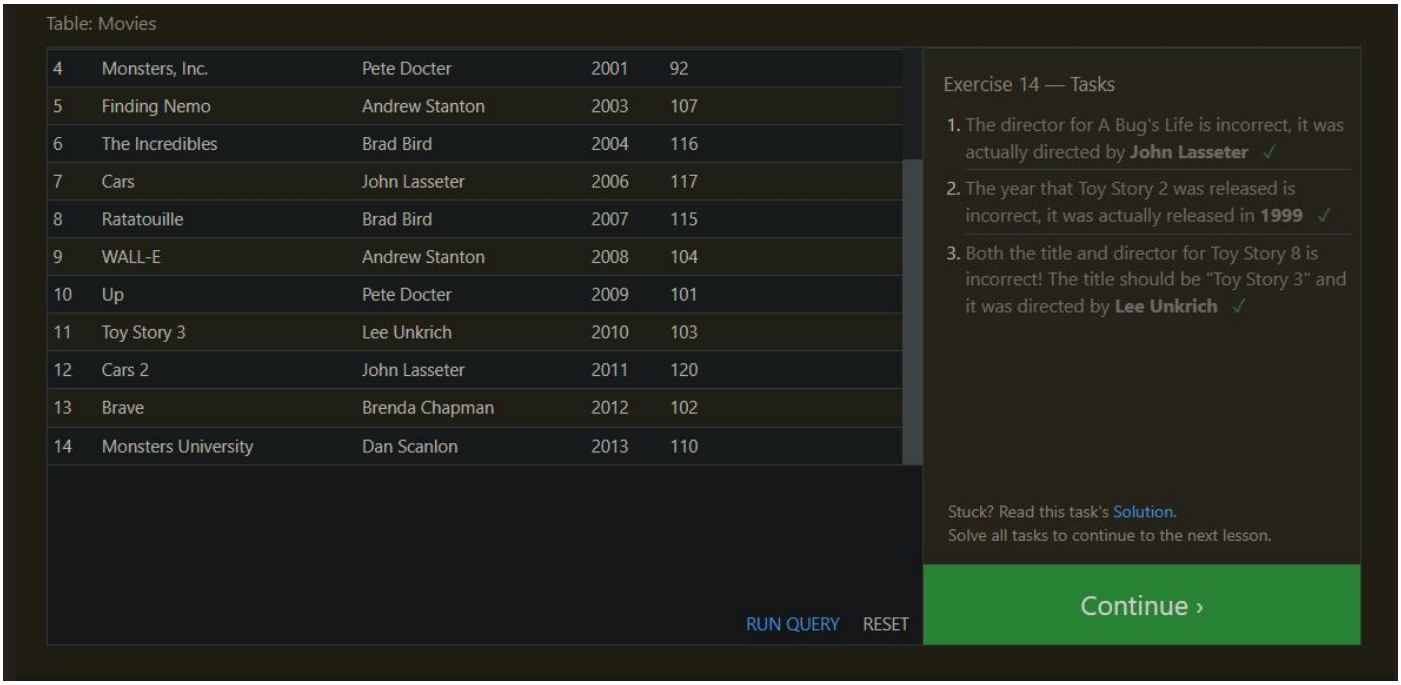
SQL Lesson 13: Inserting rows



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

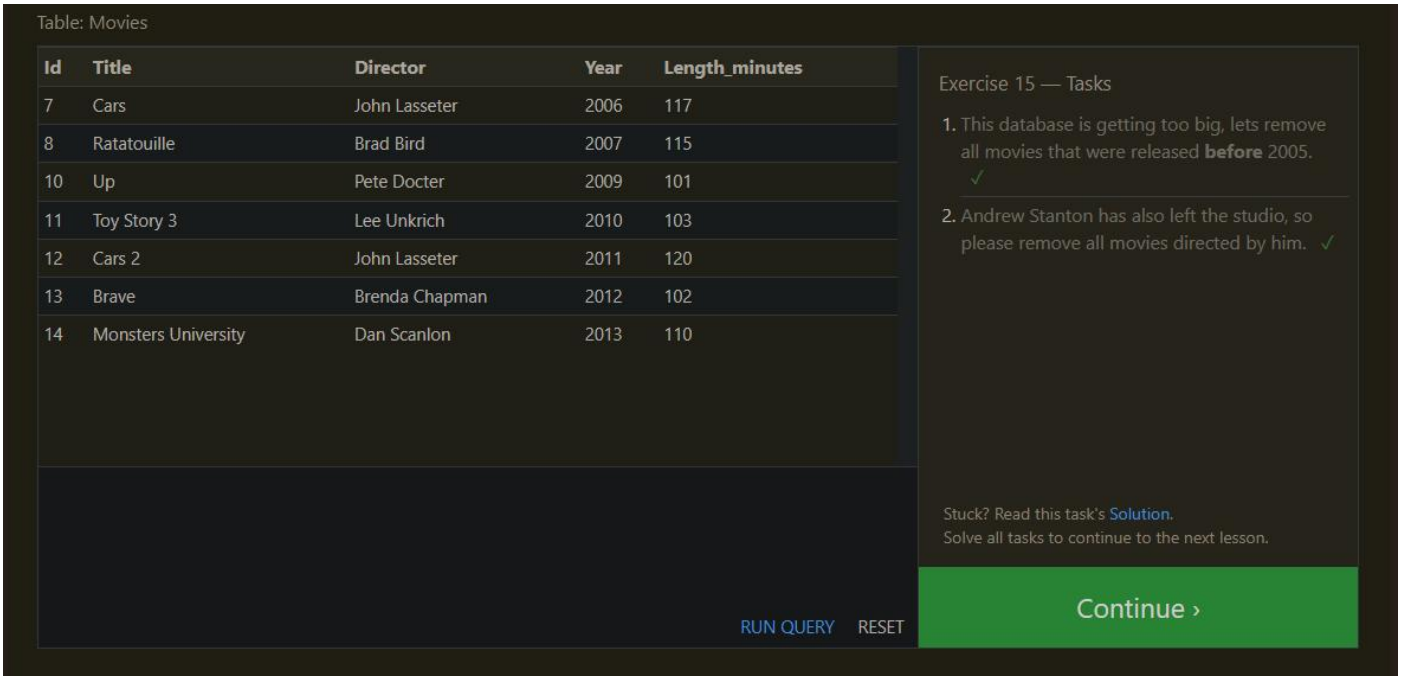


1. UPDATE movies SET director = "John Lasseter" WHERE id = 2;

2. UPDATE movies SET year = 1999 WHERE id = 3;

3.UPDATE movies SET title = "Toy Story 3", director = "Lee Unkrich" WHERE id = 11;

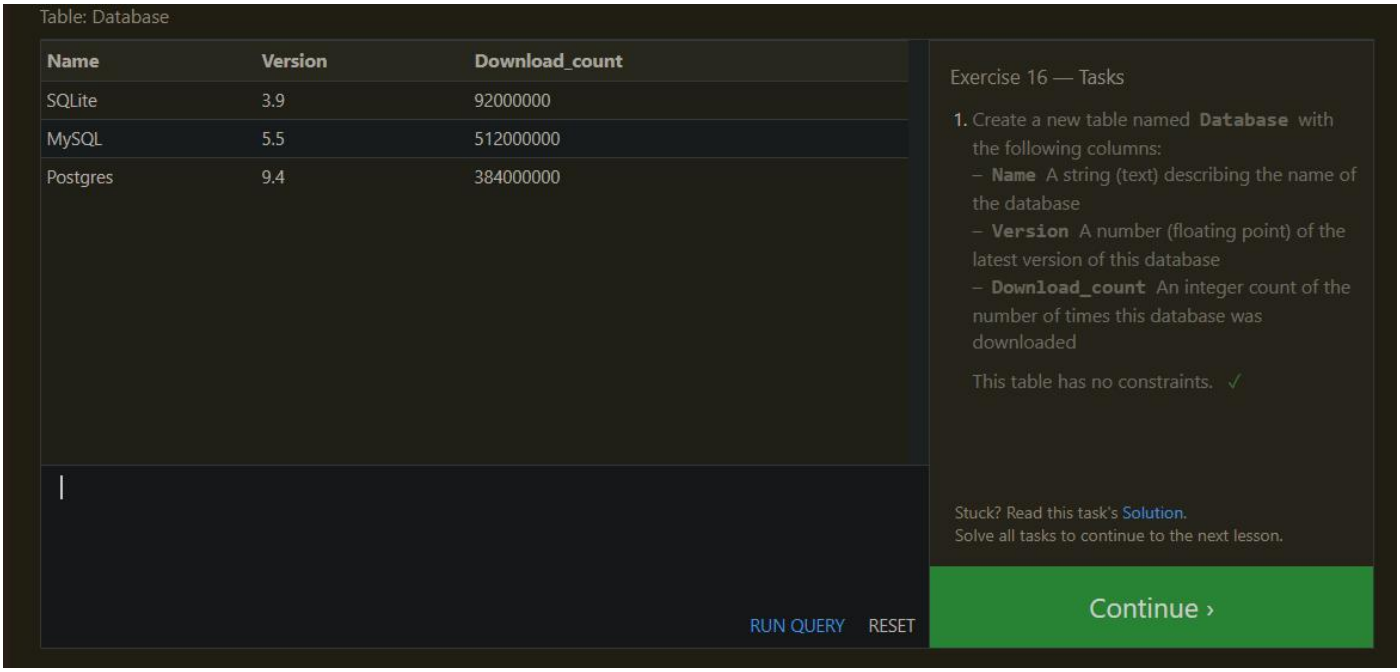
SQL Lesson 15: Deleting rows



1. DELETE FROM movies where year < 2005;

2. DELETE FROM movies where director = "Andrew Stanton";

SQL Lesson 16: Creating tables




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

SQL Lesson 17: Altering tables

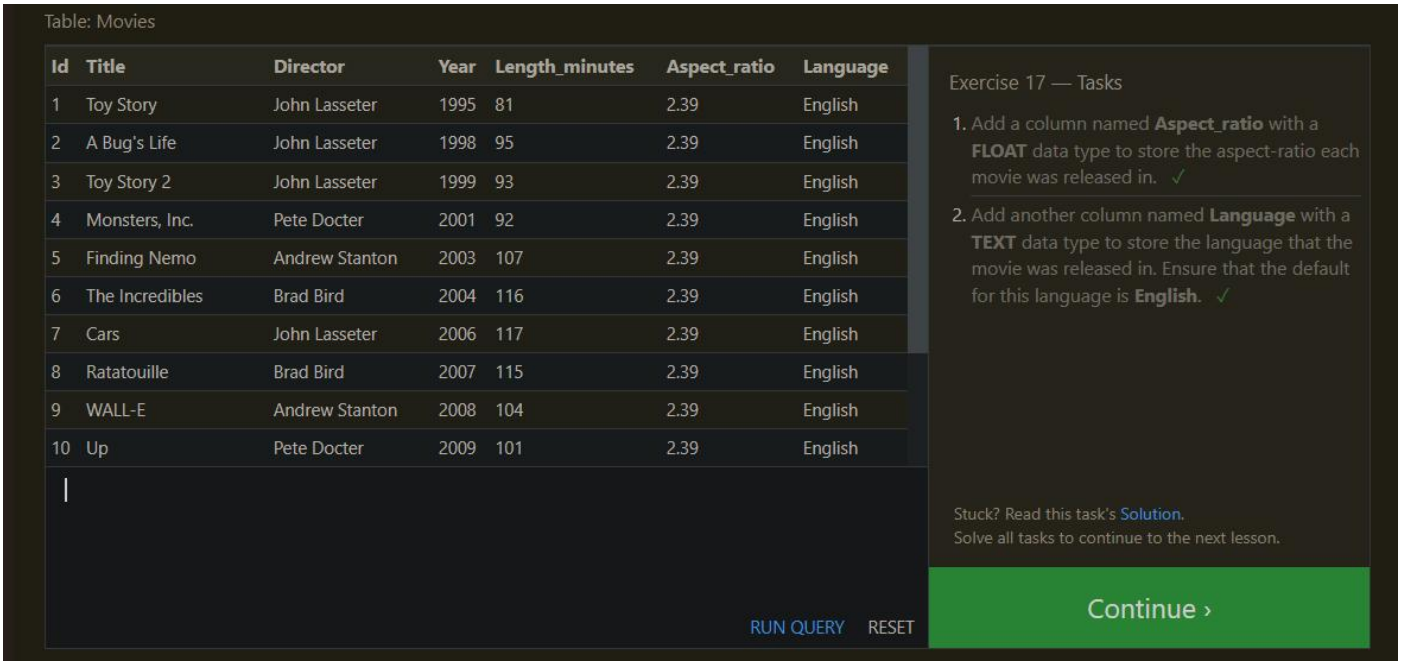


Table: Movies

Id	Title	Director	Year	Length_minutes	Aspect_ratio	Language
1	Toy Story	John Lasseter	1995	81	2.39	English
2	A Bug's Life	John Lasseter	1998	95	2.39	English
3	Toy Story 2	John Lasseter	1999	93	2.39	English
4	Monsters, Inc.	Pete Docter	2001	92	2.39	English
5	Finding Nemo	Andrew Stanton	2003	107	2.39	English
6	The Incredibles	Brad Bird	2004	116	2.39	English
7	Cars	John Lasseter	2006	117	2.39	English
8	Ratatouille	Brad Bird	2007	115	2.39	English
9	WALL-E	Andrew Stanton	2008	104	2.39	English
10	Up	Pete Docter	2009	101	2.39	English

RUN QUERY RESET

Exercise 17 — Tasks

1. Add a column named **Aspect_ratio** with a **FLOAT** data type to store the aspect-ratio each movie was released in. ✓
2. Add another column named **Language** with a **TEXT** data type to store the language that the movie was released in. Ensure that the default for this language is **English**. ✓

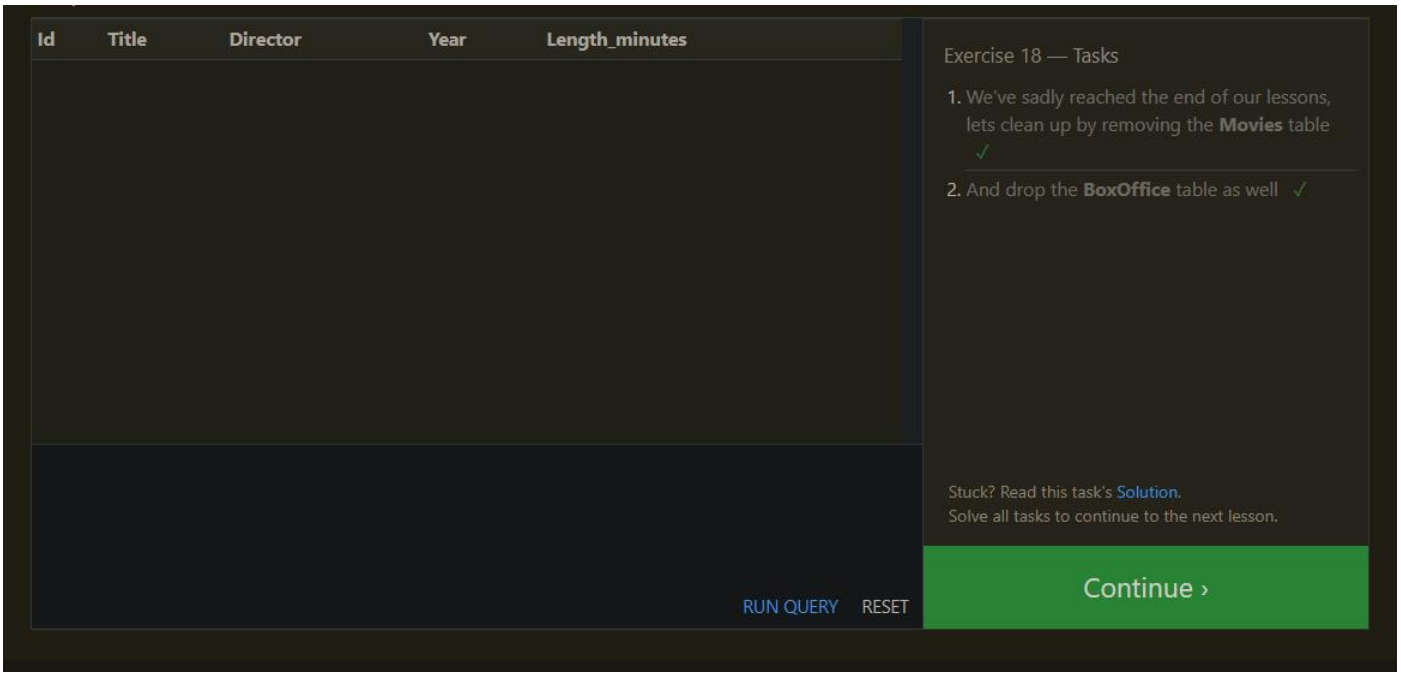
Stuck? Read this task's Solution.
Solve all tasks to continue to the next lesson.

Continue ›

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



1. DROP TABLE Movies;

2. DROP TABLE BoxOffice;